# MPC5645S Microcontroller Reference Manual

## Devices Supported:

**MPC5645S**

MPC5645SRM
Rev. 8
March 2015

This page is intentionally left blank.

# Chapter 1
# Introduction


1.1 The MPC5645S microcontroller ..... 55
1.2 MPC5645S device summary ..... 55
1.3 Device block diagram ..... 57
1.4 Feature summary ..... 58
1.5 Feature details ..... 61
1.5.1 Low-power operation ..... 61
1.5.2 e200z4d core ..... 64
1.5.3 Crossbar switch (XBAR) ..... 64
1.5.4 Enhanced Direct Memory Access (eDMA) ..... 65
1.5.5 Interrupt Controller (INTC) ..... 66
1.5.6 QuadSPI serial flash memory controller ..... 66
1.5.7 System Integration Unit Lite (SIUL) ..... 67
1.5.8 On-chip flash memory with ECC ..... 67
1.5.9 SRAM ..... 68
1.5.10 On-chip graphics SRAM ..... 68
1.5.11 Memory Protection Unit (MPU) ..... 69
1.5.12 2D Graphics Accelerator (GFX2D) ..... 69
1.5.13 Display Control Unit (DCU3) ..... 69
1.5.14 Display Control Unit Lite (DCULite) ..... 71
1.5.15 Timing Controller (TCON) and RSDS interface ..... 71
1.5.16 RLE decoder ..... 71
1.5.17 DRAM controller ..... 72
1.5.18 Video Input Unit (VIU2) ..... 72
1.5.19 Boot Assist Module (BAM) ..... 73
1.5.20 Enhanced Modular Input/Output System (eMIOS) ..... 73
1.5.21 Analog-to-Digital Converter (ADC) ..... 73
1.5.22 Deserial Serial Peripheral Interface (DSPI) ..... 74
1.5.23 FlexCAN ..... 75
1.5.24 Serial communication interface module (LINFlex) ..... 76
1.5.25 Inter-Integrated Circuit (I2C) controller modules ..... 77
1.5.26 System clocks and clock generation modules ..... 77
1.5.27 Periodic Interrupt Timer (PIT) ..... 78
1.5.28 Real Time Counter (RTC) ..... 78
1.5.29 System Timer Module (STM) ..... 78
1.5.30 Software Watchdog Timer (SWT) ..... 79
1.5.31 Stepper Motor Controller (SMC) ..... 79
1.5.32 Stepper Stall Detect (SSD) ..... 79
1.5.33 Sound Generator Module (SGM) ..... 80
1.5.34 IEEE 1149.1 JTAG controller (JTAGC) ..... 80
1.5.35 Nexus Development Interface (NDI) ..... 80
1.6 How to use the MPC5645S documents ..... 81
1.6.1 The MPC5645S document set ..... 81
1.6.2 Reference manual content ..... 82

1.7 Using the MPC5645S ....83
1.7.1 Hardware design ....83
1.7.2 Input/output pins ....84
1.7.3 Software design ....85
1.7.4 Other features ....85

## Chapter 2
## Memory Map

## Chapter 3
## Signal Description

3.1 Introduction ....91
3.2 Package pinouts ....91
3.3 Signal descriptions ....95
3.3.1 Pad configuration during reset phases ....95
3.3.2 Voltage supply pins ....95
3.3.3 System pins ....97
3.3.4 Nexus pins ....98
3.3.5 DRAM interface ....99
3.3.6 VIU muxing ....102
3.3.7 SGM muxing ....103
3.3.8 RSDS special function muxing ....103
3.3.9 Functional ports ....105

## Chapter 4
## Safety

4.1 Register Protection ....131
4.1.1 Introduction ....131
4.1.1.1 Overview ....131
4.1.1.2 Features ....131
4.1.1.3 Modes of Operation ....132
4.1.2 External Signal Description ....132
4.1.3 Memory Map and Register Description ....132
4.1.3.1 Memory Map ....133
4.1.3.2 Register description ....134
4.1.4 Functional Description ....136
4.1.4.1 General ....136
4.1.4.2 Change Lock Settings ....137
4.1.4.3 Access Errors ....139
4.1.5 Reset ....140
4.2 Software Watchdog Timer (SWT) ....140
4.2.1 Overview ....140
4.2.2 Features ....140
4.2.3 Modes of operation ....141
4.2.4 External signal description ....142

4.2.5 Memory map and register description ....142
4.2.5.1 Memory map ....142
4.2.6 Register summary ....143
4.2.6.1 SWT Control Register (SWT_CR) ....143
4.2.6.2 SWT Interrupt Register (SWT_IR) ....144
4.2.6.3 SWT Time-Out Register (SWT_TO) ....145
4.2.7 Functional Description ....147

## Chapter 5
## Analog-to-Digital Converter (ADC)

5.1 Overview ....149
5.1.1 Device-specific features ....149
5.1.2 Device-specific implementation ....150
5.2 Introduction ....150
5.3 Register descriptions ....151
5.3.1 Introduction ....151
5.3.2 Control logic registers ....153
5.3.2.1 Main Configuration Register (MCR) ....153
5.3.2.2 Main Status Register (MSR) ....155
5.3.3 Interrupt registers ....157
5.3.3.1 Interrupt Status Register (ISR) ....157
5.3.3.2 Channel Pending Registers (CEOCFR[1..2]) ....158
5.3.3.3 Interrupt Mask Register (IMR) ....159
5.3.3.4 Channel Interrupt Mask Register (CIMR[1..2]) ....160
5.3.3.5 Watchdog Threshold Interrupt Status Register (WTISR) ....161
5.3.3.6 Watchdog Threshold Interrupt Mask Register (WTIMR) ....162
5.3.4 DMA registers ....163
5.3.4.1 DMA Enable Register (DMAE) ....163
5.3.4.2 DMA Channel Select Register (DMAR[1..2]) ....163
5.3.5 Threshold registers ....164
5.3.5.1 Introduction ....164
5.3.5.2 Threshold Control Register (TRCx, x = [0..3]) ....165
5.3.5.3 Threshold Register (THRHLR[0:3]) ....165
5.3.6 Conversion timing registers CTR[1..2] ....166
5.3.7 Mask registers ....168
5.3.7.1 Introduction ....168
5.3.7.2 Normal Conversion Mask Registers (NCMR[1..2]) ....168
5.3.7.3 Injected Conversion Mask Registers (JCMR[1..2]) ....169
5.3.8 Delay registers ....170
5.3.8.1 Decode Signals Delay Register (DSDR) ....170
5.3.8.2 Power-down Exit Delay Register (PDEDR) ....170
5.3.9 Data registers ....171
5.3.9.1 Introduction ....171
5.3.9.2 Channel Data Register (CDR*n*) ....172
5.4 Functional description ....173

5.4.1 Analog channel conversion ....173
5.4.1.1 Normal conversion ....173
5.4.1.2 Start of normal conversion ....173
5.4.1.3 Normal conversion operating modes ....174
5.4.1.4 Injected channel conversion ....175
5.4.1.5 Abort conversion ....176
5.4.2 Analog clock generator and conversion timings ....177
5.4.3 ADC sampling and conversion timing ....177
5.4.4 Programmable analog watchdog ....179
5.4.4.1 Introduction ....179
5.4.5 DMA functionality ....180
5.4.6 Interrupts ....180
5.4.7 External decode signals delay ....181
5.4.8 Power-down mode ....181
5.4.9 Auto-clock-off mode ....181

## Chapter 6
## Boot Assist Module (BAM)

6.1 Overview ....183
6.2 Features ....183
6.3 Boot modes ....183
6.4 Memory map ....183
6.5 Functional description ....184
6.5.1 Entering boot modes ....184
6.5.2 Reset Configuration Half Word Source (RCHW) ....185
6.5.3 Single Chip boot mode ....187
6.5.3.1 Boot and alternate boot ....187
6.5.4 Boot through BAM ....187
6.5.4.1 Executing BAM ....187
6.5.4.2 BAM software flow ....187
6.5.4.3 BAM resources ....189
6.5.4.4 Download and execute the new code ....189
6.5.4.5 Download 64-bit password and password check ....190
6.5.4.6 Download start address, VLE bit and code size ....191
6.5.4.7 Download data ....192
6.5.4.8 Execute code ....192
6.5.5 Boot from UART ....192
6.5.5.1 Configuration ....192
6.5.5.2 Protocol ....193
6.5.6 Bootstrap with CAN ....193
6.5.6.1 Configuration ....193
6.5.6.2 Protocol ....194
6.5.7 Flash memory password swapping ....195
6.5.8 Interrupts ....195

## Chapter 7
## CAN Sampler


7.1 Introduction ....197
7.2 Main features ....197
7.3 Register description ....198
7.3.1 CAN Sampler Control Register (CR) ....198
7.3.2 CAN Sampler Sample Registers 0–11 ....199
7.4 Functional description ....200
7.4.1 Enabling/disabling the CAN Sampler ....201
7.4.2 Selecting the Rx port ....201
7.4.3 Baud rate generation ....201


## Chapter 8
## Clock Description


8.1 Clock architecture ....203
8.2 Auxiliary clocks ....205
8.3 Clock Generation Module (MC_CGM) ....206
8.3.1 Introduction ....206
8.3.1.1 Overview ....206
8.3.1.2 Features ....207
8.3.2 External signal description ....208
8.3.3 Memory map and register definition ....208
8.3.3.1 Register Descriptions ....214
8.3.4 Functional description ....226
8.3.4.1 System clock generation ....226
8.3.4.2 Dividers Functional Description ....233
8.3.4.3 DRAM Controller Clock ....234
8.3.4.4 Output Clock Multiplexing ....234
8.3.4.5 Output Clock Division Selection ....234
8.4 Oscillators ....235
8.4.1 Pierce oscillator (FXOSC) ....235
8.4.1.1 Introduction ....235
8.4.1.2 Features ....235
8.4.1.3 Modes of operation ....235
8.4.1.4 Block diagram ....235
8.4.1.5 External signal description ....236
8.4.1.6 Memory map and register definition ....237
8.4.1.7 Functional description ....237
8.4.2 External crystal oscillator (SXOSC) ....238
8.4.2.1 Features ....238
8.4.2.2 Functional description ....238
8.4.2.3 Register description ....239
8.4.3 SIRC digital interface ....240
8.4.3.1 Introduction ....240
8.4.3.2 Slow Internal RC Oscillator (128 kHz) ....240

8.4.3.3 Register description ....241
8.4.4 FIRC digital interface ....242
8.4.4.1 Introduction ....242
8.4.4.2 Functional Description (16 MHz) ....242
8.4.4.3 Register Description ....242
8.5 Frequency-modulated phase-locked loop (FMPLL) ....244
8.5.1 Introduction ....244
8.5.2 Overview ....244
8.5.3 Features ....244
8.5.4 Memory map ....245
8.5.5 Register description ....245
8.5.5.1 Control register (CR) ....245
8.5.5.2 Modulation register (MR) ....248
8.5.6 Functional description ....249
8.5.6.1 Normal mode ....249
8.5.6.2 Progressive clock switching ....250
8.5.6.3 Normal Mode with frequency modulation ....251
8.5.6.4 Powerdown mode ....252
8.5.6.5 1:1 Mode (FMPLL0 only) ....252
8.5.7 Recommendations ....252
8.6 Clock Monitor Unit (CMU) ....253
8.6.1 Introduction ....253
8.6.2 Main features ....253
8.6.3 Block diagram ....254
8.6.4 Memory map and register description ....254
8.6.4.1 Control Status Register (CMU_CSR) ....255
8.6.4.2 Frequency Display Register (CMU_FDR) ....256
8.6.4.3 High Frequency Reference Register FMPLL0 (CMU_HFREFR) ....256
8.6.4.4 Low Frequency Reference Register FMPLL0 (CMU_LFREFR) ....257
8.6.4.5 Interrupt Status Register (CMU_ISR) ....257
8.6.4.6 Measurement Duration Register (CMU_MDR) ....258
8.6.5 Functional description ....258
8.6.5.1 Crystal clock monitor ....259
8.6.5.2 PLL clock monitor ....259
8.6.5.3 Frequency meter ....259

## Chapter 9
## Crossbar Switch (XBAR)

9.1 Information specific to this device ....261
9.1.1 Device-specific block diagram ....261
9.1.2 XBAR Master ID Numbers ....261
9.1.3 Unsupported features ....262
9.2 Introduction ....262
9.2.1 Overview ....262
9.2.2 Features ....262

9.2.3 Limitations ....263
9.2.4 General Operation ....263
9.3 XBAR registers ....264
9.3.1 Register summary ....264
9.3.2 XBAR Register Descriptions ....266
9.3.2.1 Master Priority Register ....266
9.3.2.2 Slave General Purpose Control Register ....269
9.3.2.3 Master General Purpose Control Register ....271
9.3.3 Coherency ....272
9.4 Function ....272
9.4.1 Arbitration ....272
9.4.1.1 Arbitration During Undefined Length Bursts ....273
9.4.1.2 Fixed Priority Operation ....273
9.4.1.3 Round-Robin Priority Operation ....274
9.4.2 Priority Assignment ....274
9.4.2.1 Context Switching ....274
9.4.2.2 Priority Elevation ....275
9.4.3 Master Port Functionality ....275
9.4.3.1 General ....275
9.4.3.2 Master Port Decoders ....277
9.4.3.3 Master Port Capture Unit ....277
9.4.3.4 Master Port Registers ....277
9.4.3.5 Master Port State Machine ....277
9.4.4 Slave Port Functionality ....278
9.4.4.1 General ....278
9.4.4.2 Slave Port Muxes ....279
9.4.4.3 Slave Port Registers ....280
9.4.4.4 Slave Port State Machine ....280
9.5 Initialization/Application Information ....285
9.6 Interface ....285
9.6.1 Overview ....286
9.6.2 Master Ports ....286
9.6.2.1 Ignored Accesses ....286
9.6.2.2 Terminated Accesses ....286
9.6.2.3 Taken Accesses ....286
9.6.2.4 Stalled Accesses ....286
9.6.2.5 Error Response Terminated Accesses ....287
9.6.3 Slave Ports ....287

## Chapter 10
## Deserial Serial Peripheral Interface (DSPI)

10.1 Introduction ....289
10.2 Block diagram ....289
10.3 Overview ....289
10.4 Features ....290

10.5 Modes of operation ....291
10.5.1 Master mode ....291
10.5.2 Slave mode ....291
10.5.3 Module disable mode ....291
10.5.4 External stop mode ....292
10.5.5 Debug mode ....292
10.6 Device-specific information ....292
10.7 External signal description ....292
10.7.1 Signal overview ....292
10.7.2 Signal names and descriptions ....293
10.7.2.1 Peripheral Chip Select / Slave Select (CS_0) ....293
10.7.2.2 Peripheral Chip Selects 1–2 (CS1:2) ....293
10.7.2.3 Serial Input (SIN_x) ....293
10.7.2.4 Serial Output (SOUT_x) ....293
10.7.2.5 Serial Clock (SCK_*x*) ....293
10.8 Memory map and register description ....293
10.8.1 Memory map ....293
10.8.2 Register description ....294
10.8.2.1 DSPI Module Configuration Register (DSPIx_MCR) ....294
10.8.2.2 DSPI Transfer Count Register (DSPIx_TCR) ....297
10.8.2.3 DSPI Clock and Transfer Attributes Registers 0–7 (DSPIx_CTARn) ....298
10.8.2.4 DSPI Status Register (DSPIx_SR) ....304
10.8.2.5 DSPI DMA / Interrupt Request Select and Enable Register (DSPIx_RSER) ....306
10.8.2.6 DSPI PUSH TX FIFO Register (DSPIx_PUSHR) ....308
10.8.2.7 DSPI POP RX FIFO Register (DSPIx_POPR) ....310
10.8.2.8 DSPI Transmit FIFO Registers 0–4 (DSPIx_TXFRn) ....311
10.8.2.9 DSPI Receive FIFO Registers 0–4 (DSPIx_RXFRn) ....311
10.9 Functional description ....312
10.9.1 Modes of operation ....313
10.9.1.1 Master Mode ....313
10.9.1.2 Slave Mode ....314
10.9.1.3 Module Disable Mode ....314
10.9.1.4 External Stop Mode ....314
10.9.1.5 Debug Mode ....314
10.9.2 Start and Stop of DSPI Transfers ....314
10.9.3 Serial Peripheral Interface (SPI) Configuration ....315
10.9.3.1 SPI Master Mode ....316
10.9.3.2 SPI Slave Mode ....316
10.9.3.3 FIFO Disable Operation ....316
10.9.3.4 Transmit First In First Out (TX FIFO) Buffering Mechanism ....316
10.9.3.5 Receive First In First Out (RX FIFO) Buffering Mechanism ....317
10.9.4 DSPI Baud Rate and Clock Delay Generation ....318
10.9.4.1 Baud Rate Generator ....319
10.9.4.2 CS to SCK Delay (tCSC) ....319

10.9.4.3 After SCK Delay (tASC) ....319
10.9.4.4 Delay after Transfer (tDT) ....320
10.9.5 Transfer Formats ....321
10.9.5.1 Classic SPI Transfer Format (CPHA = 0) ....322
10.9.5.2 Classic SPI Transfer Format (CPHA = 1) ....323
10.9.5.3 Modified SPI Transfer Format (MTFE = 1, CPHA = 0) ....324
10.9.5.4 Modified SPI Transfer Format (MTFE = 1, CPHA = 1) ....325
10.9.5.5 Continuous Selection Format ....326
10.9.5.6 Clock Polarity Switching between DSPI Transfers ....327
10.9.6 Continuous Serial Communications Clock ....328
10.9.7 Interrupts/DMA Requests ....329
10.9.7.1 End of Queue Interrupt Request (EOQF) ....330
10.9.7.2 Transmit FIFO Fill Interrupt or DMA Request (TFFF) ....330
10.9.7.3 Transfer Complete Interrupt Request (TCF) ....330
10.9.7.4 Transmit FIFO Underflow Interrupt Request (TFUF) ....331
10.9.7.5 Receive FIFO Drain Interrupt or DMA Request (RFDF) ....331
10.9.7.6 Receive FIFO Overflow Interrupt Request (RFOF) ....331
10.9.7.7 FIFO Overrun Request (TFUF) or (RFOF) ....331
10.9.8 Power Saving Features ....331
10.9.8.1 External Stop Mode ....331
10.9.8.2 Module Disable Mode ....332
10.9.8.3 Slave Interface Signal Gating ....332
10.10 Initialization and Application Information ....332
10.10.1How to Change Queues ....332
10.10.2Baud Rate Settings ....333
10.10.3Delay Settings ....334
10.10.4Calculation of FIFO Pointer Addresses ....334
10.10.4.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO ... 335
10.10.4.2 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO .. 335

## Chapter 11
## Display Control Unit (DCU3)

11.1 Introduction ....337
11.1.1 Overview ....338
11.1.2 Features ....339
11.1.3 Modes of operation ....340
11.2 External signal description ....341
11.2.1 Overview ....341
11.2.2 Detailed signal descriptions ....341
11.3 Memory map and register definition ....342
11.3.1 Memory map ....342
11.3.2 Register map ....342
11.3.3 Register summary ....348

11.3.4 Register descriptions ....359
11.3.4.1 Control Descriptor L0_1 Register (CtrlDescL0_1) ....359
11.3.4.2 Control Descriptor L0_2 Register ....360
11.3.4.3 Control Descriptor L0_3 Register ....361
11.3.4.4 Control Descriptor L0_4 Register ....362
11.3.4.5 Control Descriptor L0_5 Register ....364
11.3.4.6 Control Descriptor L0_6 Register ....365
11.3.4.7 Control Descriptor L0_7 Register ....367
11.3.4.8 Control Descriptor Cursor 1 Register (CtrlDescCursor_1) ....367
11.3.4.9 Control Descriptor Cursor 2 Register (CtrlDescCursor_2) ....368
11.3.4.10 Control Descriptor Cursor 3 Register (CtrlDescCursor_3) ....369
11.3.4.11 Control Descriptor Cursor 4 Register (CtrlDescCursor_4) ....369
11.3.4.12 DCU3 Mode Register (DCU_MODE) ....370
11.3.4.13 BGND Register ....372
11.3.4.14 DISP_SIZE Register ....373
11.3.4.15 HSYN_PARA Register ....373
11.3.4.16 VSYN_PARA Register ....374
11.3.4.17 SYN_POL Register ....375
11.3.4.18 Threshold Register ....376
11.3.4.19 Interrupt Status Register (INT_STATUS) ....377
11.3.4.20 Interrupt Mask Register (INT_MASK) ....378
11.3.4.21 COLBAR Registers ....380
11.3.4.22 Divide Ratio Register (DIV_RATIO) ....384
11.3.4.23 SIGN_CALC_1 Register ....385
11.3.4.24 SIGN_CALC_2 Register ....386
11.3.4.25 CRC_VAL Register ....386
11.3.4.26 PDI Status Register ....387
11.3.4.27 PDI Status Mask Register ....388
11.3.4.28 PARR_ERR_STATUS Register ....389
11.3.4.29 Mask PARR_ERR status register ....392
11.3.4.30 THRESHOLD_INP_BUF_1 Register ....394
11.3.4.31 THRESHOLD_INP_BUF_2 Register ....395
11.3.4.32 LUMA Component Register ....395
11.3.4.33 Red Chroma Components ....396
11.3.4.34 Green Chroma Component Register ....396
11.3.4.35 BLUE chroma Component Register ....397
11.3.4.36 CRC_POS Register ....398
11.3.4.37 FG0_FCOLOR Register ....398
11.3.4.38 FG0_bcolor ....399
11.3.4.39 LYR_INTPOL_EN ....401
11.3.4.40 LYR_LUMA_COMPONENT ....401
11.3.4.41 LYR_CHROMA_RED ....402
11.3.4.42 LYR_CHROMA_GREEN ....402
11.3.4.43 LYR_CHROMA_BLUE ....403
11.3.4.44 COMP_IMSIZE ....404

11.3.4.45 Global Protection Register ....404
11.3.4.46 Soft Lock Bit Register L0 ....405
11.3.4.47 Soft Lock Bit Register L1 ....407
11.3.4.48 Soft Lock DISP_SIZE Register ....408
11.3.4.49 Soft Lock HSYNC/VSYNC PARA Register ....409
11.3.4.50 Soft Lock POL Register ....410
11.3.4.51 Soft Lock L0_TRANSP Register ....410
11.3.4.52 Soft Lock L1_TRANSP Register ....412
11.4 Functional description ....412
11.4.1 Graphic sources ....413
11.4.2 TFT LCD panel configuration ....413
11.4.3 DCU3 mode selection and background color ....415
11.4.4 Layer configuration and blending ....416
11.4.4.1 Blending priority of layers ....417
11.4.4.2 Control Descriptors ....419
11.4.4.3 Layer size and positioning ....419
11.4.4.4 Graphics and data format ....420
11.4.4.5 Alpha and chroma-key blending ....424
11.4.4.6 Transparency mode and blending ....430
11.4.4.7 Luminance mode ....433
11.4.4.8 Tile mode ....434
11.4.5 Hardware cursor ....435
11.4.6 CLUT/tile RAM ....436
11.4.7 Gamma correction ....437
11.4.8 Temporal dithering ....438
11.4.9 Special DDR mode ....439
11.4.10Run Length Encoding (RLE) mode ....439
11.4.10.1 RLE decoding scheme ....440
11.5 Timing, Error and Interrupt Management ....440
11.5.1 Synchronizing to panel frame rate ....441
11.5.2 Managing the DCU3 FIFOs and DMA activity ....441
11.5.3 Error detection ....443
11.5.4 Interrupt generation ....443
11.6 Register protection ....444
11.6.1 Operation of scheme ....444
11.6.2 List of protected registers ....445
11.7 Safety Mode ....445
11.7.1 CRC area description ....447
11.7.1.1 Configuring the CRC calculation ....447
11.7.2 Summary of operation ....448
11.8 Parallel Data Interface (camera interface) ....449
11.8.1 PDI interface description ....449
11.8.1.1 Introduction ....449
11.8.1.2 PDI interaction with other modules ....449
11.8.1.3 Features ....451

11.8.1.4 ITU-R BT.656 sync information extraction ....451
11.8.1.5 Normal and Narrow Mode ....453
11.8.1.6 Modes of operation based on sync extraction ....455
11.8.1.7 Mode of operation depending on PDI[17:0] ....459
11.8.1.8 PDI-related Interrupts ....460
11.9 Switch between DCU mode and PDI mode (top-level description) ....461
11.9.1 Changes in the configuration ....462
11.9.1.1 PDI slave mode ....462
11.9.1.2 PDI sync detection/validation ....462
11.9.1.3 Other assumptions ....463
11.10 DCU3 initialization ....463
11.11 Glossary ....464

## Chapter 12
## Display Control Unit Lite (DCULite)

12.1 Introduction ....465
12.1.1 Overview ....465
12.1.2 Features ....467
12.1.3 Modes of operation ....468
12.2 External signal description ....468
12.2.1 Detailed signal descriptions ....469
12.3 Memory map and register definition ....470
12.3.1 Memory map ....470
12.3.2 Register map ....470
12.3.3 Register summary ....473
12.3.4 Register descriptions ....483
12.3.4.1 Control Descriptor L0_1 Register (CtrlDescL0_1) ....483
12.3.4.2 Control Descriptor L0_2 Register ....484
12.3.4.3 Control Descriptor L0_3 Register ....485
12.3.4.4 Control Descriptor L0_4 Register ....486
12.3.4.5 Control Descriptor L0_5 Register ....488
12.3.4.6 Control Descriptor L0_6 Register ....488
12.3.4.7 Control Descriptor L0_7 Register ....489
12.3.4.8 Control Descriptor Cursor 1 Register (CtrlDescCursor_1) ....490
12.3.4.9 Control Descriptor Cursor 2 Register (CtrlDescCursor_2) ....490
12.3.4.10 Control Descriptor Cursor 3 Register (CtrlDescCursor_3) ....491
12.3.4.11 Control Descriptor Cursor_4 Register (CtrlDescCursor_4) ....492
12.3.4.12 DCULite Mode Register (DCU_MODE) ....492
12.3.4.13 BGND Register ....494
12.3.4.14 DISP_SIZE Register ....495
12.3.4.15 HSYN_PARA Register ....496
12.3.4.16 VSYN_PARA Register ....496
12.3.4.17 SYN_POL Register ....497
12.3.4.18 Threshold Register (THRESHOLD) ....498
12.3.4.19 Interrupt Status Register (INT_STATUS) ....499

12.3.4.20 Interrupt Mask Register (INT_MASK) ....500
12.3.4.21 COLBAR registers ....501
12.3.4.22 Clock Divider Ratio (DIV_RATIO) register ....505
12.3.4.23 SIGN_CALC_1 register ....506
12.3.4.24 SIGN_CALC_2 register ....507
12.3.4.25 CRC_VAL register ....507
12.3.4.26 PDI Status Register (PDI_STATUS) ....508
12.3.4.27 PDI Status Mask Register (MASK_PDI_STATUS) ....509
12.3.4.28 PARR_ERR Status Register (PARR_ERR_STATUS) ....510
12.3.4.29 MASK_PARR_ERR_STATUS register ....512
12.3.4.30 THRESHOLD_INPUT BUF_1 Register ....513
12.3.4.31 Luma Component Register (LUMA) ....514
12.3.4.32 Red Chroma Components (RED) ....514
12.3.4.33 Green Chroma Component Register (GREEN) ....515
12.3.4.34 Blue Chroma Component Register (BLUE) ....515
12.3.4.35 CRC_POS Register ....516
12.3.4.36 FG0_fcolor Register ....517
12.3.4.37 FG0_bcolor ....517
12.3.4.38 LYR_INTPOL_EN ....518
12.3.4.39 LYR_LUMA_COMPONENT ....518
12.3.4.40 LYR_CHROMA_RED ....519
12.3.4.41 LYR_CHROMA_GREEN ....520
12.3.4.42 LYR_CHROMA_BLUE ....520
12.3.4.43 COMP_IMSIZE ....521
12.3.4.44 Global Protection Register ....521
12.3.4.45 Soft Lock Bit Register L0 ....522
12.3.4.46 Soft Lock Bit Register L1 ....523
12.3.4.47 Soft Lock DISP_SIZE Register ....525
12.3.4.48 Soft Lock HSYNC/VSYNC PARA Register ....526
12.3.4.49 Soft Lock POL Register ....527
12.3.4.50 Soft Lock L0_TRANSP Register ....527
12.3.4.51 Soft Lock L1_TRANSP Register ....528
12.4 Functional description ....529
12.4.1 Graphic sources ....530
12.4.2 TFT LCD panel configuration ....530
12.4.3 DCULite mode selection and background color ....532
12.4.4 Layer configuration and blending ....533
12.4.4.1 Blending priority of layers ....534
12.4.4.2 Control Descriptors ....536
12.4.4.3 Layer size and positioning ....536
12.4.4.4 Graphics and data format ....537
12.4.4.5 Alpha and Chroma-key blending ....540
12.4.4.6 Transparency mode and blending ....547
12.4.4.7 Luminance mode ....550
12.4.4.8 Tile mode ....551

12.4.5 Hardware cursor ....552
12.4.6 CLUT RAM ....553
12.4.7 Gamma correction ....554
12.4.8 Temporal dithering ....555
12.4.9 Special DDR mode ....556
12.4.10 Run Length Encoding (RLE) mode ....556
12.4.10.1 RLE decoding scheme ....557
12.5 Timing, error and interrupt management ....557
12.5.1 Synchronizing to panel frame rate ....558
12.5.2 Managing the DCULite FIFOs and DMA activity ....558
12.5.3 Error detection ....560
12.5.4 Interrupt generation ....560
12.6 Register protection ....561
12.6.1 Operation of scheme ....561
12.6.2 List of protected registers ....562
12.7 Safety mode ....562
12.7.1 CRC area description ....564
12.7.1.1 Configuring the CRC calculation ....564
12.7.2 Summary of operation ....565
12.8 Parallel Data Interface (camera interface) ....566
12.8.1 PDI interface description ....566
12.8.1.1 Introduction ....566
12.8.1.2 PDI interaction with other modules ....566
12.8.1.3 Features ....568
12.8.1.4 ITU-R BT.656 sync information extraction ....568
12.8.1.5 Normal and Narrow Mode ....570
12.8.1.6 Modes of Operation Based on Sync Extraction ....572
12.8.1.7 Mode of operation depending on PDI[17:0] ....576
12.8.1.8 PDI-related Interrupts ....577
12.9 Switch between DCU mode and PDI mode (top-level description) ....578
12.9.1 Changes in the configuration ....579
12.9.1.1 PDI slave mode ....579
12.9.1.2 PDI sync detection/validation ....579
12.9.1.3 Other assumptions ....580
12.10 DCULite initialization ....580
12.11 Glossary ....580

## Chapter 13
## DRAM Controller (DRAMC)

13.1 Introduction ....583
13.1.1 Overview ....584
13.2 Features ....584
13.3 Memory map and register definition ....585
13.3.1 Memory map ....585
13.3.2 Register descriptions ....586

13.3.2.1 DRAMC System Configuration Register (DRAMC_SCR) .....586
13.3.2.2 Timing Configuration .....590
13.3.2.3 DRAMC Command Register (DRAMC_CMD) .....595
13.3.2.4 DRAMC Compact Command Register (DRAMC_CCMD) .....595
13.3.2.5 DQS Config Offset Count Register (DRAMC_DQS_OC) .....597
13.3.2.6 DQS Config Offset Time Register (DRAMC_DQS_OT) .....598
13.3.2.7 DQS Delay Status (DRAMC_DQS_DS) .....598
13.3.2.8 DRAMC Extra Attributes (DRAMC_EXTRA) .....599
13.4 Functional description .....600
13.4.1 Interfacing with the DRAM .....600
13.4.1.1 Connecting the DRAM .....600
13.4.2 Programming DRAM Device Internal Configuration Register .....600
13.4.3 DRAM Command Engine .....600
13.4.4 Write Buffer .....601
13.4.5 Timing Manager .....601
13.4.6 DRAM Read Block and DRAM Write Block .....601
13.4.7 Bus Interface .....602

## Chapter 14
## DRAMC Priority Manager

14.1 Introduction .....603
14.2 Features .....604
14.3 Detailed signal description .....605
14.4 Memory map and register definition .....606
14.4.1 Memory map .....606
14.4.2 Register descriptions .....608
14.4.2.1 prioman_config1, prioman_config2 (CFG1, CFG2) .....608
14.4.2.2 HPCFG .....609
14.4.2.3 Lookup table main upper registers (MLUTU0–MLUTU4) .....610
14.4.2.4 Lookup table main lower registers (MLUTL0–MLUTL4) .....611
14.4.2.5 Lookup table alternate upper registers (ALUTU0–ALUTU4) .....612
14.4.2.6 Lookup table alternate lower registers (ALUTL0–ALUTL4) .....613
14.4.2.7 Performance monitor config register (PMCFG) .....614
14.4.2.8 Event Time Timer (EVTMR) .....615
14.4.2.9 Event Time Preset (EVPRST) .....615
14.4.2.10 Performance monitor address registers .....616
14.4.2.11 Counter registers .....617
14.5 Functional description .....624
14.5.1 Description of operation — overview .....625
14.5.2 Description of operation — block diagram .....625
14.5.3 Congestion detector .....627

## Chapter 15
## e200z4d Core

15.1 Overview .....629

15.2 Features ....630
15.2.1 Execution Unit Features ....631
15.2.1.1 Instruction Unit Features ....631
15.2.1.2 Integer Unit Features ....632
15.2.1.3 Load/Store Unit Features ....632
15.2.2 L1 Cache Features ....632
15.2.3 Memory Management Unit Features ....633
15.2.4 System Bus (Core Complex Interface) Features ....633
15.2.5 Nexus 3+ Features ....633
15.3 Programming model ....634
15.3.1 Register set ....634
15.3.2 Instruction set ....636
15.3.3 Interrupts and exception handling ....637
15.4 Microarchitecture summary ....639
15.5 Availability of detailed documentation ....640

## Chapter 16
## eDMA Channel Mux (DMACHMUX)

16.1 Introduction ....641
16.1.1 Overview ....641
16.1.2 Features ....641
16.1.3 Modes of operation ....642
16.2 External signal description ....642
16.2.1 Overview ....642
16.3 Memory map and register definition ....642
16.3.1 Register descriptions ....643
16.3.1.1 Channel configuration registers ....643
16.4 Functional description ....646
16.4.1 DMA Channels with periodic triggering capability ....646
16.4.2 DMA Channels with no triggering capability ....648
16.4.3 "Always Enabled" DMA Sources ....648
16.5 Initialization/application information ....649
16.5.1 Reset ....649
16.5.2 Enabling and configuring sources ....649

## Chapter 17
## Enhanced Modular IO Subsystem (eMIOS)

17.1 Introduction ....653
17.2 Features ....654
17.3 Modes of operation ....654
17.4 Device-specific information ....654
17.4.1 Unsupported features ....655
17.4.2 Device-specific configuration ....655
17.4.3 eMIOS clocking configuration ....655
17.4.4 Channel types ....656

17.4.5 Unified Channel Block ....658
17.5 External signal description ....659
17.5.1 Overview ....659
17.5.2 Detailed signal descriptions ....659
17.5.2.1 emiosi[n] - eMIOS200 Channel Input Signal ....659
17.5.2.2 emioso[n] - eMIOS200 Channel Output Signal ....659
17.5.2.3 emios_flag_out[n] - eMIOS200 Channel Flag Signal ....659
17.6 Memory map and register description ....659
17.6.1 Memory map ....659
17.6.1.1 Unified Channel memory map ....660
17.6.2 Register description ....661
17.6.2.1 eMIOS200 Module Configuration Register (MCR) ....661
17.6.2.2 eMIOS200 Global FLAG Register (GFR) ....662
17.6.2.3 eMIOS200 Output Update Disable (OUDR) ....663
17.6.2.4 eMIOS200 Disable Channel (UCDIS) ....664
17.6.2.5 eMIOS200 UC A Register (CADR[n]) ....666
17.6.2.6 eMIOS200 UC B Register (CBDR[n]) ....666
17.6.2.7 eMIOS200 UC Counter Register (CCNTR[n]) ....667
17.6.2.8 eMIOS200 UC Control Register (CCR[n]) ....668
17.6.2.9 eMIOS200 UC Status Register (CSR[n]) ....672
17.6.2.10 eMIOS200 UC Alternate A Register (ALTCADR[n]) ....673
17.7 Functional description ....674
17.7.1 Unified Channel (UC) ....676
17.7.1.1 UC Modes of Operation ....678
17.7.1.2 Input Programmable Filter (IPF) ....693
17.7.1.3 Clock Prescaler (CP) ....694
17.7.1.4 Effect of Freeze on the Unified Channel ....694
17.7.2 Global Clock Prescaler Submodule (GCP) ....694
17.7.2.1 Effect of Freeze on the GCP ....695
17.8 Initialization/application information ....695
17.8.1 Considerations ....695
17.8.2 Application Information ....695
17.8.2.1 Time Base Generation ....696
17.8.2.2 Coherent Accesses ....698
17.8.2.3 Channel/Modes Initialization ....698

## Chapter 18
## Enhanced Direct Memory Access (eDMA)

18.1 Introduction ....699
18.1.1 Overview ....700
18.1.2 Features ....700
18.2 Memory map/register definition ....701
18.2.1 Register descriptions ....702
18.2.1.1 DMA Control Register (DMACR) ....702
18.2.1.2 DMA Error Status (DMAES) ....705

18.2.1.3 DMA Enable Request (DMAERQH, DMAERQL) ....708
18.2.1.4 DMA Enable Error Interrupt (DMAEEIH, DMAEEIL) ....709
18.2.1.5 DMA Set Enable Request (DMASERQ) ....710
18.2.1.6 DMA Clear Enable Request (DMACERQ) ....711
18.2.1.7 DMA Set Enable Error Interrupt (DMASEEI) ....711
18.2.1.8 DMA Clear Enable Error Interrupt (DMACEEI) ....712
18.2.1.9 DMA Clear Interrupt Request (DMACINT) ....713
18.2.1.10 DMA Clear Error (DMACERR) ....713
18.2.1.11 DMA Set START Bit (DMASSRT) ....714
18.2.1.12 DMA Clear DONE Status (DMACDNE) ....714
18.2.1.13 DMA Interrupt Request (DMAINTH, DMAINTL) ....715
18.2.1.14 DMA Error (DMAERRH, DMAERRL) ....716
18.2.1.15 DMA Hardware Request Status (DMAHRSH, DMAHRSL) ....717
18.2.1.16 DMA General Purpose Output Register (DMAGPOR) ....718
18.2.1.17 DMA Channel n Priority (DCHPRIn), n = 0,..., {15,31,63} ....719
18.2.1.18 Transfer Control Descriptor (TCD) ....720
18.3 Functional description ....732
18.3.1 DMA microarchitecture ....732
18.3.2 DMA basic data flow ....733
18.3.3 DMA performance ....736
18.4 Initialization/application information ....739
18.4.1 DMA initialization ....739
18.4.2 DMA programming errors ....740
18.4.3 DMA arbitration mode considerations ....741
18.4.3.1 Fixed group arbitration, fixed channel arbitration ....741
18.4.3.2 Round-robin group arbitration, fixed channel arbitration ....741
18.4.3.3 Round-robin group arbitration, round-robin channel arbitration ....741
18.4.3.4 Fixed group arbitration, round-robin channel arbitration ....742
18.4.4 DMA transfer ....742
18.4.4.1 Single request ....742
18.4.4.2 Multiple requests ....743
18.4.5 TCD status ....745
18.4.5.1 Minor loop complete ....745
18.4.5.2 Active channel TCD reads ....746
18.4.5.3 Preemption status ....746
18.4.6 Channel linking ....746
18.4.7 Dynamic programming ....747
18.4.7.1 Dynamic priority changing ....747
18.4.7.2 Dynamic channel linking and dynamic scatter/gather ....747
18.4.8 Hardware request release timing ....748

## Chapter 19
## Error Correction Status Module (ECSM)

19.1 Introduction ....749
19.2 Overview ....749

19.3 Features ....749
19.4 Memory map and register description ....749
19.4.1 Memory map ....749
19.4.2 Register description ....750
19.4.2.1 Miscellaneous User-Defined Control Register (MUDCR) ....750
19.4.2.2 ECC registers ....751
19.4.2.3 ECC Configuration Register (ECR) ....752
19.4.2.4 ECC Status Register (ESR) ....753
19.4.2.5 ECC Error Generation Register (EEGR) ....755
19.4.2.6 Flash ECC Address Register (FEAR) ....758
19.4.2.7 Flash ECC Master Number Register (FEMR) ....759
19.4.2.8 Flash ECC Attributes (FEAT) register ....760
19.4.2.9 Flash ECC Data Register (FEDR) ....760
19.4.2.10 RAM ECC Address Register (REAR) ....761
19.4.2.11 RAM ECC Syndrome Register (RESR) ....762
19.4.2.12 RAM ECC Master Number Register (REMR) ....764
19.4.2.13 RAM ECC Attributes (REAT) register ....765
19.4.2.14 RAM ECC Data Register (REDR) ....765
19.4.3 High Priority Enables ....766
19.4.4 Supervisor mode access protection ....767

## Chapter 20
## FlexCAN

20.1 Introduction ....769
20.1.1 Overview ....769
20.1.2 FlexCAN module features ....770
20.1.3 Modes of operation ....771
20.2 Device-specific information ....771
20.3 External signal description ....772
20.3.1 Overview ....772
20.3.2 Signal descriptions ....772
20.3.2.1 CAN Rx ....772
20.3.2.2 CAN Tx ....772
20.4 Memory map and register description ....772
20.4.1 FlexCAN memory mapping ....772
20.4.2 Message Buffer Structure ....774
20.4.3 Rx FIFO structure ....777
20.4.4 Register descriptions ....779
20.4.4.1 Module Configuration Register (MCR) ....779
20.4.4.2 Control Register (CTRL) ....783
20.4.4.3 Free Running Timer (TIMER) ....786
20.4.4.4 Rx Global Mask (RXGMASK) ....787
20.4.4.5 Rx 14 Mask (RX14MASK) ....788
20.4.4.6 Rx 15 Mask (RX15MASK) ....788
20.4.4.7 Error Counter Register (ECR) ....789

20.4.4.8 Error and Status Register (ESR) ....790
20.4.4.9 Interrupt Mask Register High (IMRH) ....793
20.4.4.10 Interrupt Mask Register Low (IMRL) ....794
20.4.4.11 Interrupt Flag Register High (IFRH) ....794
20.4.4.12 Interrupt Flag Register Low (IFRL) ....795
20.4.4.13 Rx Individual Mask Registers (RXIMR0–RXIMR63) ....796
20.5 Functional description ....797
20.5.1 Overview ....797
20.5.2 Transmit process ....798
20.5.3 Arbitration process ....798
20.5.4 Receive process ....799
20.5.5 Matching Process ....801
20.5.6 Data Coherence ....802
20.5.6.1 Transmission Abort Mechanism ....802
20.5.6.2 Message Buffer deactivation ....803
20.5.6.3 Message Buffer lock mechanism ....804
20.5.7 Rx FIFO ....804
20.5.8 CAN protocol related features ....805
20.5.8.1 Remote frames ....805
20.5.8.2 Overload frames ....806
20.5.8.3 Time stamp ....806
20.5.8.4 Protocol timing ....806
20.5.8.5 Arbitration and Matching Timing ....809
20.5.9 Modes of operation: Details ....810
20.5.9.1 Freeze mode ....810
20.5.9.2 Module Disable mode ....810
20.5.10 Interrupts ....811
20.5.11 Bus interface ....811
20.6 Initialization/application information ....812
20.6.1 FlexCAN initialization sequence ....812
20.6.2 FlexCAN Addressing and RAM size configurations ....813

## Chapter 21
## Flash Memory

21.1 Introduction ....815
21.1.1 Block diagram ....815
21.1.2 Flash memory block segmentation ....816
21.1.3 Features ....817
21.1.4 Modes of operation ....818
21.1.4.1 Flash User Mode ....818
21.1.4.2 Low Power Mode ....818
21.1.4.3 Power Down Mode ....818
21.1.4.4 User Test Mode (UTest) ....818
21.2 External signal description ....818
21.3 Memory map and registers ....819

21.3.1 Module Memory Map ....819
21.3.2 Register descriptions ....821
21.3.2.1 Module Configuration Register (MCR) ....821
21.3.2.2 Low/Mid Address Space Block Locking Register (LML) ....825
21.3.2.3 High Address Space Block Locking Register (HBL) ....826
21.3.2.4 Secondary Low/Mid Address Space Block Locking Register (SLL) ....827
21.3.2.5 Low/Mid Address Space Block Select Register (LMS) ....828
21.3.2.6 High Address Space Block Select Register (HBS) ....829
21.3.2.7 Address Register (ADR) ....830
21.3.2.8 Platform Flash Configuration Registers (PFCRP0 and PFCRP1) ....831
21.3.2.9 Platform Flash Access Protection Register (PFAPR) ....834
21.3.2.10 PFlash Supervisor Access Control Register (PFSACC) ....835
21.3.2.11 PFlash Data Access Control Register (PFDACC) ....837
21.3.2.12 User Test Register 0 (UT0) ....837
21.3.2.13 User Test Register 1 (UT1) ....839
21.3.2.14 User Test Register 2 (UT2) ....840
21.3.2.15 User Multiple Input Signature Register [0:4] (UM*n*) ....840
21.3.2.16 Nonvolatile private censorship PassWord 0 register (NVPWD0) ....842
21.3.2.17 Nonvolatile private censorship PassWord 1 register (NVPWD1) ....842
21.3.2.18 Nonvolatile System Censoring Information 0 register (NVSCC0) ....843
21.3.2.19 Nonvolatile System Censoring Information 1 register (NVSCC1) ....844
21.3.2.20 Nonvolatile User Options register (NVUSRO) ....845
21.4 Functional description ....846
21.4.1 User mode ....846
21.4.1.1 Read and write ....846
21.4.1.2 Flash programming ....847
21.4.1.3 Flash Erase ....851
21.4.2 Low Power mode ....854
21.4.3 Power Down mode ....855
21.4.4 UTest Mode ....855
21.4.4.1 Array Integrity Self Check ....855
21.4.4.2 Factory Margin Read ....856
21.4.4.3 ECC Logic Check ....857
21.4.5 PFLASH2P ....858
21.4.5.1 Line Read Buffers and Prefetch Operation ....858
21.4.5.2 Instruction / Data Prefetch Triggering ....859
21.4.5.3 Per-Master Prefetch Triggering ....859
21.4.5.4 Buffer Allocation ....859
21.4.5.5 Buffer Invalidation ....860
21.5 Initialization Information ....860
21.6 Application information ....860
21.6.1 Background ....860
21.6.2 Flash memory setting recommendations ....861

## Chapter 22
## Graphics Accelerator Gasket (GXG)


22.1 Introduction ....863
22.1.1 Block Diagram ....864
22.1.2 Features ....865
22.2 External Signal Description ....865
22.3 Memory Map and Register Definition ....866
22.3.1 Memory Map ....866
22.3.2 Registers Description ....868
22.3.2.1 Window Configuration (GXGCNFG0-3) ....868
22.3.2.2 Window Destination Base Address (GXGBASE0-3) ....869
22.3.2.3 Window First Address (GXGFRST0-3) ....870
22.3.2.4 Window Last Address (GXGLAST0-3) ....870
22.3.2.5 ....GFX2D Stride Setting (GXGSTRIDE) 871
22.4 Functional Description ....872
22.4.1 IPS to AHB bridge ....872
22.4.2 AXI to AHB bridge ....872
22.4.3 1x2 AXI bus matrix ....872
22.4.4 Address Filter ....873
22.4.5 Byte Swapper ....873
22.4.6 Frame buffer color depth converter ....876
22.4.7 Alpha buffer write suppressor ....879
22.4.8 Serial flash exclusive access ....879


## Chapter 23
## Graphics Static RAM (GSRAM)


23.1 Introduction ....881
23.1.1 Overview ....881
23.1.2 Features ....881
23.1.3 Modes of operation ....882
23.2 External signal description ....882
23.2.1 Memory map ....882
23.2.2 Register descriptions ....883
23.2.2.1 PRAM2P Control Register (PRAM2P_CR) ....883
23.2.2.2 PRAM2P Status Register (PRAM2P_SR) ....884
23.2.2.3 PRAM2P Fill Region Begin Address Register (PRAM2P_BEG) ....884
23.2.2.4 PRAM2P Fill Region End Address Register (PRAM2P_END) ....885
23.2.2.5 PRAM2P Fill Register (PRAM2P_FIL) ....885
23.3 Functional description ....886
23.4 Initialization information ....887
23.5 Application information ....887


## Chapter 24
## IEEE 1149.1 Test Access Port Controller (JTAGC)


24.1 Introduction ....889

24.2 Block diagram ....889
24.3 Overview ....889
24.4 Features ....890
24.5 Modes of operation ....890
24.5.1 Reset ....890
24.5.2 IEEE 1149.1-2001 defined test modes ....890
24.5.2.1 Bypass Mode ....891
24.5.2.2 TAP Sharing Mode ....891
24.6 External signal description ....891
24.7 Memory map and register description ....892
24.7.1 Instruction Register ....892
24.7.2 Bypass Register ....892
24.7.3 Device Identification Register ....892
24.7.4 Boundary Scan Register ....893
24.8 Functional description ....893
24.8.1 JTAGC reset configuration ....893
24.8.2 IEEE 1149.1-2001 (JTAG) Test Access Port ....893
24.8.3 TAP Controller State Machine ....894
24.8.3.1 Selecting an IEEE 1149.1-2001 Register ....896
24.8.4 JTAGC Instructions ....896
24.8.4.1 BYPASS Instruction ....897
24.8.4.2 ACCESS_AUX_TAP_*x* Instructions ....897
24.8.4.3 EXTEST — External Test Instruction ....897
24.8.4.4 IDCODE Instruction ....898
24.8.4.5 SAMPLE Instruction ....898
24.8.4.6 SAMPLE/PRELOAD Instruction ....898
24.8.5 Boundary Scan ....898
24.9 e200z0 OnCE Controller ....899
24.9.1 e200z0 OnCE Controller Block Diagram ....899
24.9.2 e200z0 OnCE Controller Functional Description ....899
24.9.2.1 Enabling the TAP Controller ....899
24.9.3 e200z0 OnCE Controller Register Description ....900
24.9.3.1 OnCE Command Register (OCMD) ....900
24.10 Initialization/Application Information ....901

## Chapter 25
## Inter-Integrated Circuit Bus Controller Module ($I^2C$)

25.1 Introduction ....903
25.1.1 Overview ....903
25.1.2 Features ....903
25.1.3 Block diagram ....903
25.2 Modes of operation ....904
25.3 External signal description ....904
25.3.1 Overview ....904
25.3.2 Detailed signal descriptions ....904

25.3.2.1 SCL ....904
25.3.2.2 SDA ....904
25.4 Memory map and register description ....905
25.4.1 Overview ....905
25.4.2 Module memory map ....905
25.4.3 Register description ....905
25.4.3.1 I2C Bus Address Register ....905
25.4.3.2 I2C Bus Frequency Divider Register ....906
25.4.3.3 I2C Bus Control Register ....915
25.4.3.4 I2C Bus Status Register ....916
25.4.3.5 I2C Bus Data I/O Register ....918
25.4.3.6 I2C Bus Interrupt Configuration Register ....918
25.5 Functional description ....919
25.5.1 General ....919
25.5.2 I-Bus Protocol ....919
25.5.2.1 START Signal ....919
25.5.2.2 Slave Address Transmission ....920
25.5.2.3 Data Transfer ....920
25.5.2.4 STOP Signal ....921
25.5.2.5 Repeated START Signal ....921
25.5.2.6 Arbitration Procedure ....921
25.5.2.7 Clock Synchronization ....921
25.5.2.8 Handshaking ....922
25.5.2.9 Clock Stretching ....922
25.5.3 Interrupts ....922
25.5.3.1 General ....922
25.5.3.2 Interrupt Description ....923
25.6 Initialization/Application Information ....923
25.6.1 I2C Programming Examples ....923
25.6.1.1 Initialization Sequence ....923
25.6.1.2 Generation of START ....923
25.6.1.3 Post-Transfer Software Response ....924
25.6.1.4 Generation of STOP ....925
25.6.1.5 Generation of Repeated START ....925
25.6.1.6 Slave Mode ....925
25.6.1.7 Arbitration Lost ....926
25.6.2 DMA application information ....928

## Chapter 26
## Interrupt Controller (INTC)

26.1 Introduction ....929
26.2 Features ....929
26.3 Block diagram ....930
26.4 Modes of operation ....930
26.4.1 Normal mode ....930

26.4.1.1 Software vector mode ....930
26.4.1.2 Hardware vector mode ....931
26.4.1.3 Debug mode ....931
26.4.1.4 Stop Mode ....931
26.5 Memory map and register description ....932
26.5.1 Module memory map ....932
26.5.2 Register description ....932
26.5.2.1 INTC Module Configuration Register (INTC_MCR) ....933
26.5.2.2 INTC Current Priority Register for Processor (INTC_CPR) ....933
26.5.2.3 INTC Interrupt Acknowledge Register (INTC_IACKR) ....935
26.5.2.4 INTC End-of-Interrupt Register (INTC_EOIR) ....935
26.5.2.5 INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0_3–INTC_SSCIR4_7) 936
26.5.2.6 INTC Priority Select Registers (INTC_PSR0_3–INTC_PSR206_238) ....937
26.6 Functional description ....939
26.6.1 Interrupt Request Sources ....950
26.6.1.1 Peripheral Interrupt Requests ....950
26.6.1.2 Software configurable Interrupt Requests ....950
26.6.1.3 Unique Vector for Each Interrupt Request Source ....951
26.6.2 Priority Management ....951
26.6.2.1 Current Priority and Preemption ....951
26.6.2.2 Last-In First-Out (LIFO) ....952
26.6.3 Handshaking with Processor ....952
26.6.3.1 Software Vector Mode Handshaking ....952
26.6.3.2 Hardware Vector Mode Handshaking ....954
26.7 Initialization/Application Information ....954
26.7.1 Initialization Flow ....954
26.7.2 Interrupt Exception Handler ....955
26.7.2.1 Software Vector Mode ....955
26.7.2.2 Hardware Vector Mode ....956
26.7.3 ISR, RTOS, and Task Hierarchy ....956
26.7.4 Order of Execution ....957
26.7.5 Priority Ceiling Protocol ....958
26.7.5.1 Elevating Priority ....958
26.7.5.2 Ensuring coherency ....958
26.7.6 Selecting Priorities According to Request Rates and Deadlines ....959
26.7.7 Software configurable Interrupt Requests ....959
26.7.7.1 Scheduling a Lower Priority Portion of an ISR ....959
26.7.7.2 Scheduling an ISR on Another Processor ....960
26.7.8 Lowering Priority Within an ISR ....960
26.7.9 Negating an Interrupt Request Outside of its ISR ....961
26.7.9.1 Negating an Interrupt Request as a Side Effect of an ISR ....961
26.7.9.2 Negating Multiple Interrupt Requests in One ISR ....961
26.7.9.3 Proper Setting of Interrupt Request Priority ....961
26.7.10Examining LIFO contents ....961

# Chapter 27
# LIN Controller (LINFlexD)


27.1 Introduction ....963
27.2 Main features ....963
27.2.1 LIN mode features ....964
27.2.2 UART mode features ....964
27.3 The LIN protocol ....965
27.3.1 Dominant and recessive logic levels ....965
27.3.2 LIN frames ....965
27.3.3 LIN header ....966
27.3.3.1 Break field ....966
27.3.3.2 Sync ....966
27.3.4 Response ....967
27.3.4.1 Data field ....967
27.3.4.2 Identifier ....967
27.3.4.3 Checksum ....967
27.4 LINFlexD and software intervention ....968
27.5 Summary of operating modes ....968
27.6 Controller-level operating modes ....969
27.6.1 Initialization mode ....969
27.6.2 Normal mode ....970
27.6.3 Sleep (low-power) mode ....970
27.7 LIN modes ....970
27.7.1 Master mode ....970
27.7.1.1 LIN header transmission ....970
27.7.1.2 Data transmission (transceiver as publisher) ....971
27.7.1.3 Data reception (transceiver as subscriber) ....971
27.7.1.4 Error detection and handling ....971
27.7.2 Slave mode ....971
27.7.2.1 Data transmission (transceiver as publisher) ....972
27.7.2.2 Data reception (transceiver as subscriber) ....972
27.7.2.3 Data discard ....973
27.7.2.4 Error detection and handling ....973
27.7.2.5 Valid header ....974
27.7.2.6 Valid message ....974
27.7.2.7 Overrun ....974
27.7.3 Slave mode with identifier filtering ....974
27.7.3.1 Filter submodes ....974
27.7.3.2 Identifier filter submode configuration ....975
27.7.4 Slave mode with automatic resynchronization ....977
27.7.4.1 Automatic resynchronization method ....977
27.7.4.2 Deviation error on the sync field ....978
27.8 Test modes ....978
27.8.1 Loop Back mode ....978
27.8.2 Self Test mode ....979

27.9 UART mode ....979
27.9.1 Data frame structure ....979
27.9.1.1 8-bit data frame ....979
27.9.1.2 9-bit data frame ....979
27.9.1.3 16-bit data frame ....980
27.9.1.4 17-bit data frame ....980
27.9.2 Buffer ....980
27.9.3 UART transmitter ....981
27.9.4 UART receiver ....982
27.10 Memory map and register description ....984
27.10.1 LIN control register 1 (LINCR1) ....986
27.10.2 LIN interrupt enable register (LINIER) ....989
27.10.3 LIN status register (LINSR) ....990
27.10.4 LIN error status register (LINESR) ....993
27.10.5 UART mode control register (UARTCR) ....994
27.10.6 UART mode status register (UARTSR) ....997
27.10.7 LIN timeout control status register (LINTCSR) ....999
27.10.8 LIN output compare register (LINOCR) ....1000
27.10.9 LIN timeout control register (LINTOCR) ....1001
27.10.10 LIN fractional baud rate register (LINFBRR) ....1002
27.10.11 LIN integer baud rate register (LINIBRR) ....1002
27.10.12 LIN checksum field register (LINCFR) ....1003
27.10.13 LIN control register 2 (LINCR2) ....1004
27.10.14 Buffer identifier register (BIDR) ....1005
27.10.15 Buffer data register least significant (BDRL) ....1006
27.10.16 Buffer data register most significant (BDRM) ....1007
27.10.17 Identifier filter enable register (IFER) ....1008
27.10.18 Identifier filter match index (IFMI) ....1008
27.10.19 Identifier filter mode register (IFMR) ....1009
27.10.20 Identifier filter control registers (IFCR0–IFCR15) ....1010
27.10.21 Global control register (GCR) ....1011
27.10.22 UART preset timeout register (UARTPTO) ....1012
27.10.23 UART current timeout register (UARTCTO) ....1013
27.10.24 DMA Tx enable register (DMATXE) ....1014
27.10.25 DMA Rx enable register (DMARXE) ....1014
27.11 DMA interface ....1015
27.11.1 Master node, TX mode ....1015
27.11.2 Master node, RX mode ....1018
27.11.3 Slave node, TX mode ....1021
27.11.4 Slave node, RX mode ....1023
27.11.5 UART node, TX mode ....1026
27.11.6 UART node, RX mode ....1028
27.11.7 Use cases and limitations ....1030
27.12 Functional description ....1031
27.12.1 8-bit timeout counter ....1031

27.12.1.1 LIN timeout mode ..... 1031
27.12.1.2 Output compare mode ..... 1031
27.12.2 Interrupts ..... 1031
27.12.3 Fractional baud rate generation ..... 1032
27.13 Programming considerations ..... 1033
27.13.1 Master node ..... 1034
27.13.2 Slave node ..... 1035
27.13.3 Extended frames ..... 1038
27.13.4 Timeout ..... 1039
27.13.5 UART mode ..... 1039

## Chapter 28
## Memory Protection Unit (MPU)

28.1 Introduction ..... 1041
28.1.1 Overview ..... 1041
28.1.2 Features ..... 1043
28.1.3 Modes of operation ..... 1043
28.1.4 External signal description ..... 1044
28.2 Memory map and register description ..... 1044
28.2.1 Memory map ..... 1044
28.2.2 Register description ..... 1046
28.2.2.1 MPU Control/Error Status Register (MPU_CESR) ..... 1046
28.2.2.2 MPU Error Address Register, Slave Port n (MPU_EARn) ..... 1047
28.2.2.3 MPU Error Detail Register, Slave Port n (MPU_EDRn) ..... 1047
28.2.2.4 MPU Region Descriptor n (MPU_RGDn) ..... 1048
28.2.2.5 MPU Region Descriptor Alternate Access Control n (MPU_RGDAACn) 1053
28.3 Functional description ..... 1055
28.3.1 Access evaluation macro ..... 1055
28.3.1.1 Access Evaluation — Hit Determination ..... 1056
28.3.1.2 Access Evaluation — Privilege Violation Determination ..... 1056
28.3.2 Putting It All Together and AHB Error Terminations ..... 1057
28.4 Initialization information ..... 1058
28.5 Opcode pre-fetch cycles and the execute permission ..... 1058
28.6 Application information ..... 1058

## Chapter 29
## Mode Entry Module (MC_ME)

29.1 Introduction ..... 1061
29.1.1 Overview ..... 1061
29.1.2 Features ..... 1063
29.1.3 Modes of Operation ..... 1063
29.2 External Signal Description ..... 1064
29.3 Memory Map and Register Definition ..... 1064
29.3.1 Memory map ..... 1065
29.3.2 Register description ..... 1074

29.3.2.1 Global Status Register (ME_GS) ..... 1075
29.3.2.2 Mode Control Register (ME_MCTL) ..... 1077
29.3.2.3 Mode Enable Register (ME_ME) ..... 1078
29.3.2.4 Interrupt Status Register (ME_IS) ..... 1079
29.3.2.5 Interrupt Mask Register (ME_IM) ..... 1080
29.3.2.6 Invalid Mode Transition Status Register (ME_IMTS) ..... 1081
29.3.2.7 Debug Mode Transition Status Register (ME_DMTS) ..... 1082
29.3.2.8 ***RESET*** Mode Configuration Register (ME_RESET_MC) ..... 1085
29.3.2.9 ***TEST*** Mode Configuration Register (ME_TEST_MC) ..... 1085
29.3.2.10 ***SAFE*** Mode Configuration Register (ME_SAFE_MC) ..... 1086
29.3.2.11 ***DRUN*** Mode Configuration Register (ME_DRUN_MC) ..... 1086
29.3.2.12 ***RUN0...3*** Mode Configuration Registers (ME_RUN0…3_MC) ..... 1087
29.3.2.13 ***HALT*** Mode Configuration Register (ME_HALT_MC) ..... 1087
29.3.2.14 ***STOP*** Mode Configuration Register (ME_STOP_MC) ..... 1088
29.3.2.15 ***STANDBY*** Mode Configuration Register (ME_STANDBY_MC) ..... 1088
29.3.2.16 Peripheral Status Register 0 (ME_PS0) ..... 1090
29.3.2.17 Peripheral Status Register 1 (ME_PS1) ..... 1090
29.3.2.18 Peripheral Status Register 2 (ME_PS2) ..... 1091
29.3.2.19 Peripheral Status Register 3 (ME_PS3) ..... 1091
29.3.2.20 Run Peripheral Configuration Registers (ME_RUN_PC0…7) ..... 1092
29.3.2.21 Low-Power Peripheral Configuration Registers (ME_LP_PC0…7) ..... 1093
29.3.2.22 Peripheral Control Registers (ME_PCTL0…143) ..... 1093
29.4 Functional description ..... 1094
29.4.1 Mode Transition Request ..... 1094
29.4.2 Modes Details ..... 1095
29.4.2.1 ***RESET*** Mode ..... 1095
29.4.2.2 ***DRUN*** Mode ..... 1096
29.4.2.3 ***SAFE*** Mode ..... 1096
29.4.2.4 ***TEST*** Mode ..... 1097
29.4.2.5 ***RUN0...3*** Modes ..... 1098
29.4.2.6 ***HALT*** Mode ..... 1098
29.4.2.7 ***STOP*** Mode ..... 1099
29.4.2.8 ***STANDBY*** Mode ..... 1100
29.4.3 Mode Transition Process ..... 1100
29.4.3.1 Target Mode Request ..... 1100
29.4.3.2 Target Mode Configuration Loading ..... 1101
29.4.3.3 Peripheral Clocks Disable ..... 1102
29.4.3.4 Processor Low-Power Mode Entry ..... 1102
29.4.3.5 Processor and System Memory Clock Disable ..... 1102
29.4.3.6 Clock sources (main voltage regulator independent) switch-on ..... 1103
29.4.3.7 Main Voltage Regulator Switch-On ..... 1103
29.4.3.8 Flash Module Switch-On ..... 1103
29.4.3.9 Clock Sources (Main Voltage Regulator Dependent) Switch-On ..... 1104
29.4.3.10 Power Domain #2 Switch-On ..... 1104
29.4.3.11 Pad Outputs-On ..... 1104

29.4.3.12 Peripheral Clocks Enable ..... 1104
29.4.3.13 Processor and Memory Clock Enable ..... 1104
29.4.3.14 Processor Low-Power Mode Exit ..... 1105
29.4.3.15 System Clock Switching ..... 1105
29.4.3.16 Power Domain #2 Switch-Off ..... 1106
29.4.3.17 Pad Switch-Off ..... 1106
29.4.3.18 Clock Sources Switch-Off ..... 1106
29.4.3.19 Flash Switch-Off ..... 1106
29.4.3.20 Main Voltage Regulator Switch-Off ..... 1107
29.4.3.21 Current Mode Update ..... 1107
29.4.4 Protection of Mode Configuration Registers ..... 1109
29.4.5 Mode Transition Interrupts ..... 1109
29.4.5.1 Invalid Mode Configuration Interrupt ..... 1109
29.4.5.2 Invalid Mode Transition Interrupt ..... 1109
29.4.5.3 ***SAFE*** Mode Transition Interrupt ..... 1111
29.4.5.4 Mode Transition Complete interrupt ..... 1111
29.4.6 Peripheral Clock Gating ..... 1111
29.4.7 Application Example ..... 1112

## Chapter 30
## Nexus Development Interface (NDI)

30.1 Introduction ..... 1115
30.2 Block diagram ..... 1115
30.3 Features ..... 1116
30.4 Modes of operation ..... 1117
30.4.1 Nexus reset ..... 1117
30.4.2 Operating mode ..... 1118
30.4.2.1 Disabled-Port Mode ..... 1118
30.4.2.2 Censored Mode ..... 1118
30.4.2.3 Stop Mode ..... 1118
30.5 External signal description ..... 1118
30.5.1 Nexus Signal Reset States ..... 1118
30.6 Memory map and register description ..... 1118
30.6.1 Nexus Debug Interface Registers ..... 1119
30.6.2 Register Description ..... 1119
30.6.2.1 Nexus Device ID Register (DID) ..... 1119
30.6.2.2 Port Configuration Register (PCR) ..... 1120
30.6.2.3 Development Control Register 1, 2 (DC1, DC2) ..... 1122
30.6.2.4 Development Status Register (DS) ..... 1124
30.6.2.5 Read/Write Access Control/Status (RWCS) ..... 1125
30.6.2.6 Read/Write Access Address (RWA) ..... 1127
30.6.2.7 Read/Write Access Data (RWD) ..... 1127
30.6.2.8 Watchpoint Trigger Register (WT) ..... 1128
30.7 Functional Description ..... 1129
30.7.1 NPC_HNDSHK module ..... 1129

30.7.2 Enabling Nexus Clients for TAP Access ..... 1130
30.7.3 Configuring the NDI for Nexus Messaging ..... 1131
30.7.4 Programmable MCKO Frequency ..... 1131
30.7.5 Nexus Messaging ..... 1132
30.7.6 EVTO Sharing ..... 1132
30.7.7 Debug Mode Control ..... 1132
30.7.7.1 EVTI Generated Break Request ..... 1132
30.7.8 Nexus Reset Control ..... 1133
30.8 Initialization / application information ..... 1133
30.8.1 Relationship between TCK and system clock frequency ..... 1133

## Chapter 31
## OpenVG Graphics Accelerator (GFX2D)

31.1 Introduction ..... 1135
31.1.1 Block Diagram ..... 1135
31.1.2 Features ..... 1136
31.1.2.1 Frame buffer ..... 1136
31.1.2.2 2D bitmap graphics (separate 2D unit) ..... 1136
31.1.2.3 Vector graphics ..... 1137
31.2 External Signal Description ..... 1138
31.3 Memory Map and Register Definition ..... 1139
31.3.1 G12_COMMANDSTREAM ..... 1139
31.3.2 G12_MMUCOMMANDSTREAM ..... 1140
31.3.3 G12_REVISION ..... 1140
31.3.4 G12_SYSSTATUS ..... 1141
31.3.5 G12_IRQSTATUS ..... 1141
31.3.6 G12_IRQENABLE ..... 1142
31.3.7 G12_IRQ_ACTIVE_CNT ..... 1142
31.3.8 G12_CLOCKEN ..... 1143
31.3.9 MMU_READ_ADDR ..... 1143
31.3.10MMU_READ_DATA ..... 1144
31.3.11G12_FIFOFREE ..... 1144
31.4 Command Stream registers ..... 1145
31.4.1 G2D_BASE0-3 ..... 1152
31.4.2 G2D_CFG0-3 ..... 1152
31.4.3 G2D_SCISSORX ..... 1153
31.4.4 G2D_SCISSORY ..... 1153
31.4.5 G2D_FOREGROUND, G2D_BACKGROUND ..... 1153
31.4.6 G2D_ALPHABLEND ..... 1153
31.4.7 G2D_ROP ..... 1154
31.4.8 G2D_CONFIG ..... 1155
31.4.9 G2D_INPUT ..... 1156
31.4.10G2D_MASK ..... 1156
31.4.11G2D_BLENDERCFG ..... 1156
31.4.12G2D_BLEND_A0-3, GD2_BLEND_C0-7 ..... 1157

31.4.13VGV1_VTX0-1 ..... 1158
31.4.14VGV1_TILEOFS ..... 1158
31.4.15VGV1_FILL ..... 1159
31.4.16VGV1_SCISSORX ..... 1159
31.4.17VGV1_SCISSORY ..... 1159
31.4.18VGV1_CFG1 ..... 1159
31.4.19VGV1_CFG2 ..... 1160
31.4.20VGV1_DIRTYBASE ..... 1160
31.4.21VGV1_CBASE1 ..... 1160
31.4.22VGV1_UBASE2 ..... 1160
31.4.23VGV2_C1X ..... 1161
31.4.24VGV2_C1Y ..... 1161
31.4.25VGV2_C2X, VGV2_C2Y ..... 1161
31.4.26VGV2_C3X, VGV2_C3Y ..... 1161
31.4.27VGV2_C4X, VGV2_C4Y ..... 1161
31.4.28VGV2_C(1-4)XREL, VGV2_C(1-4)YREL ..... 1162
31.4.29VGV2_XFXX ..... 1162
31.4.30VGV2_XFYX ..... 1162
31.4.31VGV2_XFXY ..... 1162
31.4.32VGV2_XFYY ..... 1163
31.4.33VGV2_XFXA ..... 1163
31.4.34VGV2_XFYA ..... 1163
31.4.35VGV2_XFSTXX ..... 1163
31.4.36VGV2_XFSTYX ..... 1163
31.4.37VGV2_XFSTXY ..... 1164
31.4.38VGV2_XFSTYY ..... 1164
31.4.39VGV2_BBOXMINX, VGV2_BBOXMINY ..... 1164
31.4.40VGV2_BBOXMAXX, VGV2_BBOXMAXY ..... 1164
31.4.41VGV2_SCALE ..... 1164
31.4.42VGV2_BIAS ..... 1165
31.4.43VGV2_ACCURACY ..... 1165
31.4.44VGV2_THINRADIUS ..... 1165
31.4.45VGV2_ARCCOS ..... 1165
31.4.46VGV2_ARCSIN ..... 1165
31.4.47VGV2_ARCTAN ..... 1166
31.4.48VGV2_RADIUS ..... 1166
31.4.49VGV2_MITER ..... 1166
31.4.50VGV2_CLIP ..... 1166
31.4.51VGV2_MODE ..... 1167
31.4.52VGV2_ACTION ..... 1168
31.4.53VGV3_CONTROL ..... 1168
31.4.54VGV3_MODE ..... 1169
31.4.55VGV3_WRITEADDR ..... 1169
31.4.56VGV3_WRITE ..... 1169
31.4.57VGV3_WRITEIFPAUSED ..... 1169

31.4.58VGV3_NEXTADDR ..... 1170
31.4.59VGV3_NEXTCMD ..... 1170
31.4.60VGV3_VGBYPASS ..... 1170
31.4.61VGV3_WRITES8, VGV3_WRITES16, VGV3_WRITES32, VGV3_WRITEF32, VGV3_WRITERAW 1171
31.4.62VGV3_WRITEDMI ..... 1171
31.4.63VGV3_LAST ..... 1171
31.4.64FBC_BASE ..... 1172
31.4.65FBC_DATA ..... 1172
31.4.66FBC_WIDTH ..... 1172
31.4.67FBC_HEIGHT ..... 1172
31.4.68FBC_STRIDE ..... 1172
31.4.69FBC_START ..... 1173
31.4.70G2D_CONST0-7 ..... 1173
31.4.71GRADW_CONST0-B ..... 1173
31.4.72G2D_GRADIENT ..... 1174
31.4.73GRADW_TEXCFG ..... 1174
31.4.74GRADW_TEXSIZE ..... 1175
31.4.75GRADW_TEXBASE ..... 1175
31.4.76GRADW_BORDERCOLOR ..... 1175
31.4.77GRADW_INST0-7 ..... 1176
31.4.78G2D_XY ..... 1176
31.4.79G2D_WIDTHHEIGHT ..... 1177
31.4.80G2D_SXY, G2D_SXY2 ..... 1177
31.4.81G2D_VGSPAN ..... 1177
31.4.82G2D_IDLE ..... 1177
31.4.83G2D_COLOR ..... 1178
31.5 MMU Command Stream registers ..... 1179
31.5.1 MH_MMU_CONFIG ..... 1181
31.5.2 MH_MMU_VA_RANGE ..... 1181
31.5.3 MH_MMU_PT_BASE ..... 1182
31.5.4 MH_MMU_PAGE_FAULT ..... 1182
31.5.5 MH_MMU_TRAN_ERROR ..... 1182
31.5.6 MH_MMU_INVALIDATE ..... 1183
31.5.7 MH_MMU_MPU_BASE ..... 1183
31.5.8 MH_MMU_MPU_END ..... 1183
31.5.9 MH_ARBITER_CONFIG ..... 1184
31.5.10MH_CLNT_AXI_ID_REUSE ..... 1185
31.5.11MH_INTERRUPT_MASK ..... 1185
31.5.12MH_INTERRUPT_STATUS ..... 1185
31.5.13MH_INTERRUPT_CLEAR ..... 1186
31.5.14MH_AXI_ERROR ..... 1186
31.5.15MH_PERFCOUNTER0_SELECT ..... 1187
31.5.16MH_PERFCOUNTER0_CONFIG ..... 1187
31.5.17MH_PERFCOUNTER0_LOW ..... 1187

31.5.18MH_PERFCOUNTER0_HI ..... 1187
31.5.19MH_PERFCOUNTER1_SELECT ..... 1188
31.5.20MH_PERFCOUNTER1_CONFIG ..... 1188
31.5.21MH_PERFCOUNTER1_LOW ..... 1188
31.5.22MH_PERFCOUNTER1_HI ..... 1188
31.5.23MH_DEBUG_CTRL ..... 1189
31.5.24MH_DEBUG_DATA ..... 1189
31.5.25MH_AXI_HALT_CONTROL ..... 1189
31.5.26Performance Counters ..... 1190
31.6 Functional Description ..... 1193
31.6.1 Bus interface ..... 1193
31.6.2 Input unit ..... 1193
31.6.3 Arbiter ..... 1193
31.6.4 Burst cache ..... 1193
31.6.5 2D+VG unit ..... 1193
31.6.6 2D+VG unit level architecture ..... 1194
31.6.6.1 Command Handler ..... 1194
31.6.6.2 Geometry Engine ..... 1195
31.6.6.3 Rasterizer ..... 1195
31.6.6.4 2D unit ..... 1195
31.6.6.5 Gradient and texturing unit ..... 1195

## Chapter 32
## Periodic Interrupt Timer (PIT)

32.1 Introduction ..... 1197
32.2 Signal description ..... 1197
32.3 Memory map and register description ..... 1198
32.3.1 Memory map ..... 1198
32.3.2 Register descriptions ..... 1198
32.3.2.1 PIT Module Control Register (PITMCR) ..... 1199
32.3.2.2 Timer Load Value Register (LDVAL) ..... 1199
32.3.2.3 Current Timer Value Register (CVAL) ..... 1200
32.3.2.4 Timer Control Register (TCTRL) ..... 1201
32.3.2.5 Timer Flag Register (TFLG) ..... 1202
32.4 Functional description ..... 1203
32.4.1 General ..... 1203
32.4.1.1 Timers ..... 1203
32.4.1.2 Debug Mode ..... 1204
32.4.2 Interrupts ..... 1204
32.5 Initialization and Application Information ..... 1205
32.5.1 Example Configuration ..... 1205

## Chapter 33
## Peripheral Bridge (PBRIDGE)

33.1 Introduction ..... 1207

33.1.1 Overview ....1207
33.1.2 Features ....1207
33.2 Functional description ....1207
33.2.1 Access support ....1207
33.2.1.1 Peripheral write buffering ....1207
33.2.1.2 Read cycles ....1207
33.2.1.3 Write cycles ....1208
33.2.2 General operation ....1208

## Chapter 34
## Power Control Unit (MC_PCU)

34.1 Introduction ....1209
34.1.1 Overview ....1209
34.1.2 Features ....1210
34.2 External Signal Description ....1210
34.3 Memory map and register definition ....1211
34.3.1 Memory Map ....1211
34.3.2 Register Descriptions ....1212
34.3.2.1 Power Domain #0 Configuration Register (PCU_PCONF0) ....1213
34.3.2.2 Power Domain #1 Configuration Register (PCU_PCONF1) ....1214
34.3.2.3 Power Domain #2 Configuration Register (PCU_PCONF2) ....1215
34.3.2.4 Power Domain Status Register (PCU_PSTAT) ....1215
34.4 Functional Description ....1216
34.4.1 General ....1216
34.4.2 Reset / Power-On Reset ....1216
34.4.3 MC_PCU Configuration ....1216
34.4.4 Mode Transitions ....1216
34.4.4.1 ***DRUN***, ***SAFE***, ***TEST***, ***RUN0...3***, ***HALT***, and ***STOP*** Mode Transition ..1216
34.4.4.2 ***STANDBY*** Mode Transition ....1217
34.4.4.3 Power Saving for Memories During ***STANDBY*** Mode ....1218
34.5 Initialization Information ....1219
34.6 Application Information ....1219
34.6.1 ***STANDBY*** Mode Considerations ....1219

## Chapter 35
## Quad Serial Peripheral Interface (QuadSPI)

35.1 Preface ....1221
35.1.1 Conventions ....1221
35.1.2 Glossary for QuadSPI module ....1221
35.2 Introduction ....1222
35.2.1 Overview ....1224
35.2.2 Features ....1224
35.2.3 QuadSPI modes of operation ....1224
35.2.3.1 Normal Mode ....1224
35.2.3.2 Module Disable Mode ....1224

35.2.3.3 Stop Mode ..... 1224
35.3 External Signal Description ..... 1225
35.3.1 Overview ..... 1225
35.3.2 Detailed Signal Description ..... 1225
35.3.2.1 PCSFA - Peripheral Chip Select Flash A ..... 1225
35.3.2.2 PCSFB - Peripheral Chip Select Flash B ..... 1225
35.3.2.3 SCKFA — Serial Clock Flash A ..... 1225
35.3.2.4 SCKFB — Serial Clock Flash B ..... 1225
35.3.2.5 IOFA[3:0] - Data IO Flash A ..... 1225
35.3.2.6 IOFA[3:0] - Data IO Flash B ..... 1226
35.3.3 Driving of External Signals ..... 1226
35.4 Interrupt Signals ..... 1228
35.5 Memory map and register definition ..... 1229
35.5.1 Memory map ..... 1229
35.5.2 Serial Flash Address Assignment ..... 1230
35.5.3 AMBA Bus Register Memory Map ..... 1231
35.5.4 Register descriptions ..... 1231
35.5.4.1 Register write access ..... 1231
35.5.4.2 Module Configuration Register (QSPI_MCR) ..... 1232
35.5.4.3 Latency Configuration Register (QSPI_LCR) ..... 1233
35.5.4.4 Serial Flash Address Register (QSPI_SFAR) ..... 1234
35.5.4.5 Instruction Code Register (QSPI_ICR) ..... 1235
35.5.4.6 Sampling Register (QSPI_SMPR) ..... 1236
35.5.4.7 RX Buffer Status Register (QSPI_RBSR) ..... 1237
35.5.4.8 RX Buffer Control Register (QSPI_RBCT) ..... 1238
35.5.4.9 TX Buffer Status Register (QSPI_TBSR) ..... 1238
35.5.4.10 TX Buffer Data Register (QSPI_TBDR) ..... 1239
35.5.4.11 AMBA Control Register (QSPI_ACR) ..... 1240
35.5.4.12 Status Register (QSPI_SFMSR) ..... 1241
35.5.4.13 Flag Register (QSPI_SFMFR) ..... 1242
35.5.4.14 Interrupt and DMA Request Select and Enable Register (QSPI_SFMRSER) ... 1244
35.5.4.15 RX Buffer Data Registers 0–31 (QSPI_RBDR0–QSPI_RBDR31) ..... 1246
35.5.5 AHB Bus Register Memory Map Descriptions ..... 1247
35.5.5.1 AHB Bus Access Considerations ..... 1247
35.5.5.2 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A ..1247
35.5.5.3 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B ..1247
35.5.5.4 Memory Mapped Serial Flash Data - Parallel Flash Mode ..... 1248
35.5.5.5 AHB RX data buffer (QSPI_ARDB0 to QSPI_ARDB31) ..... 1249
35.6 Functional Description ..... 1250
35.6.1 Serial Flash Access Schemes ..... 1250
35.6.2 Modes of Operation ..... 1251
35.6.3 Normal Mode ..... 1251
35.6.3.1 Issuing SFM Commands ..... 1251
35.6.3.2 Flash Programming ..... 1252

35.6.3.3 Flash Read ..... 1253
35.6.3.4 Byte Ordering of Serial Flash Read Data ..... 1255
35.6.3.5 Normal Mode Interrupt and DMA Requests ..... 1258
35.6.3.6 TX Buffer Operation ..... 1260
35.6.4 Power Saving Features ..... 1260
35.6.4.1 Stop Mode ..... 1261
35.6.4.2 Module Disable Mode ..... 1262
35.6.4.3 Leaving Power Saving Modes ..... 1263
35.6.4.4 Slave Bus Signal Gating ..... 1263
35.7 Initialization/Application Information ..... 1263
35.7.1 Power Up and Reset ..... 1263
35.7.2 Available Status/Flag Information ..... 1263
35.7.2.1 IP Commands ..... 1263
35.7.2.2 AHB Commands ..... 1264
35.7.2.3 Overview of error flags ..... 1264
35.7.2.4 IP bus and AHB access command collisions ..... 1265
35.7.3 Exclusive Access to Serial Flash for AHB Commands ..... 1265
35.7.3.1 RX Buffer Read via QSPI_ARDB Registers ..... 1266
35.7.3.2 RX Buffer Read via QSPI_RDBR Registers ..... 1266
35.7.4 Command arbitration ..... 1266
35.7.5 Flash Device Selection ..... 1267
35.7.6 Continuous Mode Commands ..... 1267
35.7.7 DMA Usage ..... 1267
35.7.7.1 DMA Usage in Normal Mode ..... 1267
35.8 Byte Ordering - Endianess ..... 1269
35.8.1 Programming Flash Data ..... 1269
35.8.2 Reading Flash Data into the RX Buffer ..... 1269
35.8.2.1 Readout of the RX Buffer via QSPI_RBDRn ..... 1270
35.8.2.2 Readout of the RX Buffer via ARDBn ..... 1270
35.8.3 Reading Flash Data into the AHB Buffer ..... 1270
35.8.3.1 Readout of the AHB Buffer via Memory Mapped Read ..... 1270
35.9 Serial Flash Devices ..... 1271
35.9.1 Supported Instruction Codes in Winbond Devices ..... 1271
35.9.2 Instruction Codes in Spansion Devices ..... 1274
35.9.3 Instruction Codes in Macronix Devices ..... 1278
35.9.4 Instruction Codes in Numonyx Devices ..... 1281
35.9.5 Serial Flash Clock Frequency Limitations ..... 1284
35.10 Internal Sampling of Serial Flash Input Data ..... 1285

## Chapter 36
## Real-Time Clock (RTC/API)

36.1 Overview ..... 1289
36.2 Features ..... 1289
36.3 Device-specific information ..... 1291
36.4 Modes of operation ..... 1291

36.4.1 Functional mode ..... 1291
36.4.2 Debug mode ..... 1292
36.5 Memory map and register descriptions ..... 1292
36.5.1 RTC Supervisor Control Register (RTCSUPV) ..... 1292
36.5.2 RTC Control Register (RTCC) ..... 1292
36.5.3 RTC Status Register (RTCS) ..... 1294
36.5.4 RTC Counter Register (RTCCNT) ..... 1295
36.6 RTC functional description ..... 1296
36.7 API functional description ..... 1297

## Chapter 37
## Reset Generation Module (MC_RGM)

37.1 Introduction ..... 1299
37.1.1 Overview ..... 1299
37.1.2 Features ..... 1300
37.1.3 Modes of operation ..... 1301
37.2 External Signal Description ..... 1302
37.3 Memory Map and Register Definition ..... 1302
37.3.1 Register Descriptions ..... 1304
37.3.1.1 Functional Event Status Register (RGM_FES) ..... 1305
37.3.1.2 Destructive Event Status Register (RGM_DES) ..... 1306
37.3.1.3 Functional Event Reset Disable Register (RGM_FERD) ..... 1307
37.3.1.4 Destructive Event Reset Disable Register (RGM_DERD) ..... 1308
37.3.1.5 Functional Event Alternate Request Register (RGM_FEAR) ..... 1309
37.3.1.6 Destructive Event Alternate Request Register (RGM_DEAR) ..... 1311
37.3.1.7 Functional Event Short Sequence Register (RGM_FESS) ..... 1312
37.3.1.8 ***STANDBY*** Reset Sequence Register (RGM_STDBY) ..... 1313
37.3.1.9 Functional Bidirectional Reset Enable Register (RGM_FBRE) ..... 1314
37.4 Functional description ..... 1315
37.4.1 Reset state machine ..... 1315
37.4.1.1 ***PHASE0*** Phase ..... 1316
37.4.1.2 ***PHASE1*** Phase ..... 1317
37.4.1.3 ***PHASE2*** Phase ..... 1317
37.4.1.4 ***PHASE3*** Phase ..... 1317
37.4.1.5 ***IDLE*** Phase ..... 1317
37.4.2 Destructive Resets ..... 1318
37.4.3 External Reset ..... 1318
37.4.4 Functional Resets ..... 1319
37.4.5 ***STANDBY*** Entry Sequence ..... 1319
37.4.6 Alternate Event Generation ..... 1319
37.4.7 Boot Mode Capturing ..... 1320

## Chapter 38
## Run-Length Encoding Decoder (RLE_DEC)

38.1 Introduction ..... 1321

38.1.1 Overview ..... 1321
38.1.2 Features ..... 1322
38.1.3 RLE_DEC modes of operation ..... 1322
38.1.3.1 Normal Mode ..... 1322
38.1.3.2 Module Disable Mode ..... 1322
38.1.3.3 Stop Mode ..... 1323
38.2 External signal description ..... 1323
38.3 Interrupt and DMA request signals ..... 1323
38.4 Memory map and register definition ..... 1323
38.4.1 Memory map ..... 1323
38.4.2 AMBA Bus Register Memory Map ..... 1323
38.4.3 Register descriptions ..... 1324
38.4.3.1 Module Configuration Register (RLE_DEC_MCR) ..... 1324
38.4.3.2 Image Configuration Register (RLE_DEC_ICR) ..... 1325
38.4.3.3 Compressed Image Size Register (RLE_DEC_CISR) ..... 1326
38.4.3.4 Decompressed Image Coordinates Register (RLE_DEC_DICR) ..... 1326
38.4.3.5 Status Register (RLE_DEC_SR) ..... 1327
38.4.3.6 Interrupt Request Status Register (RLE_DEC_ISR) ..... 1327
38.4.3.7 Interrupt Request Enable Register (RLE_DEC_RIER) ..... 1328
38.4.3.8 Start Pixel Coordinate Register of Image (RLE_DEC_SPCR) ..... 1329
38.4.3.9 End Pixel Coordinate Register of Image (RLE_DEC_EPCR) ..... 1330
38.4.4 Crossbar switch memory map descriptions ..... 1331
38.4.4.1 Rx FIFO address range ..... 1331
38.4.4.2 Tx FIFO address range ..... 1331
38.4.4.3 Memory mapped Rx FIFO ..... 1331
38.4.4.4 Memory mapped Tx FIFO ..... 1331
38.5 Functional description ..... 1332
38.5.1 RLE encoding format ..... 1332
38.5.2 RLE decoding process ..... 1332
38.5.3 Image coordinates’ example ..... 1333
38.5.4 Modes of operation ..... 1333
38.5.5 Normal mode ..... 1334
38.5.6 Power-saving features ..... 1334
38.5.6.1 Module Disable Mode ..... 1334

## Chapter 39
## Sound Generator Module (SGM)

39.1 Introduction ..... 1335
39.2 Features ..... 1335
39.3 Device-specific configuration ..... 1336
39.4 Block diagram ..... 1337
39.5 External signal description ..... 1338
39.6 Memory map and register definition ..... 1338
39.6.1 Memory map ..... 1338
39.6.2 Register descriptions ..... 1340

39.6.2.1 SGM Control Register (SGMCTL) ....1340
39.6.2.2 SGM Configuration Register (SGMCFG) ....1342
39.6.2.3 Clock Configuration Register for Resampler (CLKRSP) ....1345
39.6.2.4 Clock Configuration Register for Channel 3 (CLKCH3) ....1346
39.6.2.5 DDS Configuration Register for Channel 3 (DDSCH3) ....1346
39.6.2.6 Envelope Configuration Register of Attack Phase for Channel 3 (ECRACH3) 1347
39.6.2.7 Envelope Configuration Register of Release Phase for Channel 3 (ECRRCH3) 1348
39.6.2.8 Envelope Configuration Register of sustain Timing for Channel 3 (ECRSCH3) 1349
39.6.2.9 Inter-Note No-Output Phase Timing for Channel 3 (NTCH3) ....1350
39.6.2.10 Target Note Pulse Count for Channel 3 (TPCCH3) ....1350
39.6.2.11 PlayBack Timing Configuration Register for Channel 3(PTCCH3) ....1351
39.6.2.12 Dead Time Configuration Register for Channel 3(DTCCH3) ....1352
39.6.2.13 Repeat Number Configuration Register for Channel 3(RNCCH3) ....1352
39.6.2.14 Volume Control Register for Wave mode (VCRWAV) ....1353
39.6.2.15 SGM TimeOut Count Register (SGMTOCR) ....1354
39.6.2.16 Mixer Configuration Register (MIXCR) ....1354
39.6.2.17 Clock Configuration Register for PWM (CLKPWM) ....1355
39.6.2.18 PWM Configuration Register (PWMCR) ....1356
39.6.2.19 Data FIFO Register 1(DFIFO1) ....1356
39.6.2.20 Data FIFO Register 2(DFIFO2) ....1357
39.6.2.21 FIFO WaterMark (FIFOWM) ....1358
39.6.2.22 FIFO Read Pointer (FIFORP) ....1358
39.6.2.23 FIFO Write Pointer (FIFOWP) ....1359
39.6.2.24 SGM Status Register (SGMST) ....1359
39.6.2.25 SGM Interrupt Control Register for FIFO and DMA (SGMICFD) ....1361
39.6.2.26 SGM Interrupt Control Register (SGMIC) ....1363
39.6.2.27 SGM Interrupt Status Register for FIFO and DMA (SGMISFD) ....1365
39.6.2.28 SGM Interrupt Status Register (SGMIS) ....1366
39.6.2.29 I2S Enable Register (I2SEN) ....1368
39.6.2.30 I2S Control Register (I2SCTL) ....1369
39.6.2.31 I2S Output Data Format Control Register (I2SDFC) ....1370
39.6.2.32 I2S Clock Prescaler Register (I2SPRS) ....1372
39.6.2.33 I2S Interrupt Control Register (I2SINTC) ....1372
39.6.2.34 I2S Status Register (I2SST) ....1373
39.7 Functional description ....1374
39.7.1 Wave mode ....1374
39.7.1.1 State Machine of SGM channel in Wave mode ....1376
39.7.2 DDS mode ....1377
39.7.2.1 DDS Concept ....1377
39.7.2.2 Wavetable initialization ....1378
39.7.2.3 Generating the tone ....1378
39.7.2.4 Updating the configuration buffers ....1380

39.7.2.5 State Machine of SGM channel in DDS mode ....1381
39.7.3 SGM architecture ....1382
39.7.4 SGM clocking ....1382
39.7.5 Channel controller ....1383
39.7.5.1 DDS Controller ....1383
39.7.5.2 ASR Envelope Controller ....1384
39.7.5.3 Sample Format Converter ....1384
39.7.5.4 Volume Control ....1384
39.7.6 Re-sampling Block ....1385
39.7.7 Mixer ....1385
39.7.8 I2S Interface ....1386
39.7.8.1 Features ....1386
39.7.8.2 Clock Choices ....1387
39.7.8.3 I2S Block Diagram ....1387
39.7.8.4 Supported Protocol modes ....1387
39.7.9 PWM Output ....1394
39.8 Interrupts and DMA ....1395
39.9 Initialization and application information ....1396
39.9.1 Wave Mode Use Cases ....1396
39.9.1.1 Speech ....1396
39.9.1.2 Alarm sound ....1397
39.9.2 DDS Mode Use Cases ....1398
39.9.2.1 Polyphonic sound ....1399
39.9.2.2 Polyphonic alarm sound ....1402

## Chapter 40
## Static RAM (SRAM)

40.1 Introduction ....1405
40.1.1 Modes of operation ....1405
40.1.1.1 Normal (functional) mode ....1405
40.1.1.2 Standby mode ....1405
40.2 External signal description ....1405
40.3 Memory map and registers ....1405
40.4 Functional description ....1406
40.5 SRAM ECC mechanism ....1406
40.5.1 Access timing ....1406
40.5.2 Reset effects on SRAM accesses ....1408
40.6 DMA requests ....1408
40.7 Interrupt Requests ....1408
40.8 Initialization and application information ....1409
40.8.1 Example code ....1409

## Chapter 41
## Stepper Motor Controller (SMC)

41.1 Introduction ....1411

41.1.1 Features ..... 1411
41.1.2 Modes of operation ..... 1411
41.1.2.1 Functional modes ..... 1411
41.1.2.2 PWM channel configuration modes ..... 1411
41.1.2.3 PWM alignment modes ..... 1412
41.1.2.4 Low-power modes ..... 1412
41.1.3 Block diagram ..... 1413
41.2 External signal description ..... 1414
41.2.1 M0C0M/M0C0P/M0C1M/M0C1P — PWM Output Pins for Motor 0 ..... 1414
41.2.2 M1C0M/M1C0P/M1C1M/M1C1P — PWM Output Pins for Motor 1 ..... 1415
41.2.3 M2C0M/M2C0P/M2C1M/M2C1P — PWM Output Pins for Motor 2 ..... 1415
41.2.4 M3C0M/M3C0P/M3C1M/M3C1P — PWM Output Pins for Motor 3 ..... 1415
41.2.5 M4C0M/M4C0P/M4C1M/M4C1P — PWM Output Pins for Motor 4 ..... 1415
41.2.6 M5C0M/M5C0P/M5C1M/M5C1P — PWM Output Pins for Motor 5 ..... 1415
41.3 Memory map and register definition ..... 1415
41.3.1 Module memory map ..... 1415
41.3.2 Register description ..... 1417
41.3.2.1 Motor Controller Control Register 0 (MCCTL0) ..... 1418
41.3.2.2 Motor Controller Control Register 1 (MCCTL1) ..... 1419
41.3.2.3 Motor Controller Period Register (MCPER) ..... 1420
41.3.2.4 Motor Controller Channel Control Register (MCCC0..11) ..... 1420
41.3.2.5 Motor Controller Duty Cycle Register (MCDC0..11) ..... 1421
41.3.2.6 Short-circuit Detector Time-out Register (MCSDTO) ..... 1423
41.3.2.7 Short-circuit Detector Enable Register 0 (MCSDE0) ..... 1423
41.3.2.8 Short-circuit Detector Enable Register 1 (MCSDE1) ..... 1424
41.3.2.9 Short-circuit Detector Enable Register 2 (MCSDE2) ..... 1424
41.3.2.10 Short-circuit Detector Interrupt Enable Register 0 (MCSDIEN0) ..... 1425
41.3.2.11 Short-circuit Detector Interrupt Enable Register 1 (MCSDIEN1) ..... 1425
41.3.2.12 Short-circuit Detector Interrupt Enable Register 2 (MCSDIEN2) ..... 1426
41.3.2.13 Short-circuit Detector Interrupt Register 0 (MCSDI0) ..... 1426
41.3.2.14 Short-circuit Detector Interrupt Register 1 (MCSDI1) ..... 1427
41.3.2.15 Short-circuit Detector Interrupt Register 2 (MCSDI2) ..... 1427
41.4 Functional Description ..... 1428
41.4.1 Modes of Operation ..... 1428
41.4.1.1 PWM Output Modes ..... 1428
41.4.1.2 Relationship Between PWM Mode and PWM Channel Enable ..... 1431
41.4.1.3 Relationship Between Sign, Duty, Dither, RECIRC, Period, and PWM Mode Functions 1431
41.4.2 PWM Duty Cycle ..... 1441
41.4.3 Motor Controller Counter Clock Source ..... 1441
41.4.4 Output Switching Delay ..... 1442
41.4.5 Operation in SMC stop mode ..... 1442
41.4.6 Short-circuit detection ..... 1442
41.5 Reset ..... 1447
41.6 Interrupts ..... 1447

# Chapter 42
# Stepper Stall Detect (SSD)


42.1 Introduction ..... 1449
42.1.1 Overview ..... 1449
42.1.2 Features ..... 1451
42.1.3 Modes of Operation ..... 1451
42.1.3.1 Disabled Mode ..... 1451
42.1.3.2 Normal Mode ..... 1451
42.1.3.3 Power Down Modes ..... 1451
42.2 External Signal Description ..... 1452
42.3 Memory Map and Register Definition ..... 1452
42.3.1 Memory Map ..... 1452
42.3.2 Register Summary ..... 1453
42.3.3 Register Descriptions ..... 1454
42.3.3.1 SSD Control and Status Register (CONTROL) ..... 1454
42.3.3.2 Interrupt Enable and Flag Register (IRQ) ..... 1455
42.3.3.3 Integration Accumulator Register (ITGACC) ..... 1456
42.3.3.4 Down Counter Register (DCNT) ..... 1456
42.3.3.5 Blanking Counter Load Register (BLNCNTLD) ..... 1457
42.3.3.6 Integration Counter Load Register (ITGCNTLD) ..... 1457
42.3.3.7 SSD Prescale and Divider Register (PRESCALE) ..... 1458
42.4 Functional Description ..... 1460
42.4.1 Main Building Blocks of the SSD ..... 1460
42.4.1.1 Analog Block ..... 1460
42.4.1.2 Analog Wrapper + Port Control ..... 1462
42.4.1.3 Register Interface ..... 1464
42.4.1.4 BIS Control ..... 1465
42.4.2 Stepper Stall Detection Measurement ..... 1468
42.4.2.1 Overview of the SSD Measurement ..... 1468
42.4.2.2 Details of the SSD Measurement ..... 1469
42.4.3 Additional Modes of Operation ..... 1471
42.4.3.1 Blanking with No Drive ..... 1471
42.4.3.2 Integration with No Drive ..... 1471
42.5 Initialization Information ..... 1471
42.5.1 Analog Block Startup Time ..... 1471
42.5.2 Analog Block Polarity Switching Time ..... 1471
42.5.3 SSD Startup ..... 1472
42.6 Application Information ..... 1472
42.6.1 Current Flow Examples ..... 1472
42.6.2 Setting of the PRESCALE Register ..... 1474
42.6.2.1 Timing Resolution Considerations ..... 1474
42.6.2.2 Offset Cancellation Considerations ..... 1475
42.6.3 Watching Internal States of the SSD ..... 1475
42.6.4 Stepper Motor Transition Considerations ..... 1476
42.6.4.1 SSD Phase-In and Phase-Out ..... 1476

42.6.4.2 Changing of SSD Internal States ....1476
42.6.5 Legacy Modes - Separate Blanking and Integration Phase ....1477

## Chapter 43
## System Integration Unit Lite (SIUL)

43.1 Introduction ....1479
43.2 Overview ....1479
43.3 Features ....1480
43.4 External signal description ....1481
43.4.1 Detailed signal descriptions ....1481
43.4.1.1 General-purpose I/O pins (GPIO[0:184]) ....1481
43.4.1.2 External interrupt request input pins (EIRQ[0:23]) ....1481
43.4.1.3 Special function output pins configuration (PCR[185:281]) ....1482
43.5 Memory Map and Register Description ....1482
43.5.1 SIU memory map ....1483
43.5.2 Register protection ....1484
43.5.3 Register description ....1484
43.5.3.1 MCU ID Register #1 (MIDR1) ....1485
43.5.3.2 MCU ID Register #2 (MIDR2) ....1485
43.5.3.3 Interrupt Status Flag Register (ISR) ....1486
43.5.3.4 Interrupt Request Enable Register (IRER) ....1487
43.5.3.5 Interrupt Rising-Edge Event Enable Register (IREER) ....1487
43.5.3.6 Interrupt Falling-Edge Event Enable Register (IFEER) ....1488
43.5.3.7 Interrupt Filter Enable Register (IFER) ....1489
43.5.3.8 Pad Configuration Registers (PCR0–PCR184) ....1489
43.5.3.9 Pad Configuration Registers (PCR185–PCR281) ....1491
43.5.3.10 Pad Selection for Multiplexed Inputs Registers (PSMI0_3–PSMI50_53) .1493
43.5.3.11 GPIO Pad Data Output Registers (GPDO0_3 - GPDO184) ....1497
43.5.3.12 GPIO Pad Data Input Registers (GPDI0_3–GPDI184) ....1498
43.5.3.13 Parallel GPIO Pad Data Out Registers (PGPDO0–PGPDO5) ....1499
43.5.3.14 Parallel GPIO Pad Data In Register (PGPDI0–PGPDI5) ....1500
43.5.3.15 Masked Parallel GPIO Pad Data Out Register (MPGPDO0–MPGPDO11) .... 1501
43.5.3.16 Interrupt Filter Maximum Counter Registers (IFMC0–IFMC23) ....1502
43.5.3.17 Interrupt Filter Clock Prescaler Register (IFCPR) ....1503
43.6 Functional description ....1504
43.6.1 General ....1504
43.6.2 Pad control ....1504
43.6.3 General purpose input and output pads (GPIO) ....1504
43.6.4 External interrupts ....1505
43.6.4.1 External interrupt management ....1506
43.7 Pin muxing ....1506

## Chapter 44
## System Status and Configuration Module (SSCM)


44.1 Introduction ..... 1507
44.1.1 Overview ..... 1507
44.1.2 Features ..... 1507
44.1.3 Modes of operation ..... 1508
44.2 Memory map and register description ..... 1508
44.2.1 Memory map ..... 1508
44.2.2 Register description ..... 1508
44.2.2.1 System Status Register (STATUS) ..... 1508
44.2.2.2 System Memory Configuration Register ..... 1509
44.2.2.3 Error Configuration ..... 1510
44.2.2.4 Debug Status Port Register ..... 1511
44.2.2.5 Password Comparison Registers ..... 1512
44.3 Functional Description ..... 1514
44.4 Initialization/Application Information ..... 1514
44.4.1 Reset ..... 1514


## Chapter 45
## System Timer Module (STM)


45.1 Introduction ..... 1515
45.1.1 Overview ..... 1515
45.1.2 Features ..... 1515
45.1.3 Modes of operation ..... 1515
45.2 External signal description ..... 1515
45.3 Memory map and register definition ..... 1515
45.3.1 Memory map ..... 1515
45.3.2 Register descriptions ..... 1516
45.3.2.1 STM Control Register (STM_CR) ..... 1516
45.3.2.2 STM Count Register (STM_CNT) ..... 1517
45.3.2.3 STM Channel Control Register (STM_CCR*n*) ..... 1518
45.3.2.4 STM Channel Interrupt Register (STM_CIR*n*) ..... 1518
45.3.2.5 STM Channel Compare Register (STM_CMP*n*) ..... 1519
45.4 Functional description ..... 1519


## Chapter 46
## Timing Controller (TCON)


46.1 Introduction ..... 1521
46.1.1 Features ..... 1521
46.1.2 Modes of operation ..... 1522
46.2 External signal descriptions ..... 1522
46.3 Memory map and register definition ..... 1522
46.3.1 Memory map ..... 1522
46.3.2 Register summary ..... 1523
46.3.3 Register descriptions ..... 1525

46.3.3.1 Control Register 1 (TCON_CTRL1) ....1526
46.3.3.2 Bit Mapping Control (TCON_BMC) ....1528
46.3.3.3 TCON_COMP0 - TCON_COMP3 ....1529
46.3.3.4 TCON_COMP0_MSK - TCON_COMP3_MSK ....1530
46.3.3.5 TCON_PULSE0 - TCON_PULSE5 ....1531
46.3.3.6 TCON_PULSE0_MSK - TCON_PULSE5_MSK ....1532
46.3.3.7 TCON_SMX0 - TCON_SMX13 ....1532
46.3.3.8 TCON_OMUX_LOW ....1535
46.3.3.9 TCON_OMUX_HIGH ....1536
46.3.3.10 TCON_LUT0 - TCON_LUT13 ....1538
46.3.3.11 TCON_DATA0_DLY - TCON_DATA12_DLY ....1539
46.3.3.12 TCON_CTRL2 ....1542
46.4 Functional description ....1542
46.4.1 Modes of operation ....1543
46.4.1.1 RSDS mode ....1543
46.4.1.2 TTL mode ....1543
46.4.1.3 Bypass mode ....1543
46.4.2 Timing signal generator ....1544
46.4.2.1 Comparator ....1544
46.4.2.2 Pulse generator ....1544
46.4.2.3 Toggle generator ....1545
46.4.2.4 Signal Mixer (SMX) ....1547
46.4.2.5 Output Crossbar Mux ....1548
46.4.3 Data inversion control ....1548
46.4.4 Bit Mapping Control (BMC) ....1548
46.4.4.1 Bit mapping in TTL mode ....1549
46.4.4.2 Bit mapping in RSDS mode ....1549
46.4.4.3 Bit mapping examples ....1550
46.4.4.4 Clock mapping in RSDS mode ....1552
46.4.4.5 Clock mapping in TTL mode ....1553
46.4.5 Clock/Data Skew Adjustment ....1555
46.5 RSDS interface description ....1555
46.5.1 Introduction ....1555
46.5.2 Features ....1556
46.5.3 Data path signals ....1556
46.5.3.1 pad_p ....1556
46.5.3.2 pad_n ....1556
46.5.4 Cell description ....1556
46.5.4.1 RSDS_ref - RSDS Reference cell ....1556
46.5.4.2 RSDS_Tx - RSDS transmitter cell ....1556
46.5.5 Functionality and modes of operation: ....1557
46.5.6 General ....1558
46.5.7 Timing Diagrams ....1558
46.6 Initialization/application information ....1560
46.6.1 TCON Initialization ....1560

## Chapter 47
## Video Input Unit (VIU2)


47.1 Introduction ..... 1561
47.1.1 Features ..... 1562
47.2 Memory map and register definition ..... 1562
47.2.1 Memory map ..... 1562
47.2.2 Register Summary ..... 1562
47.2.3 Register descriptions ..... 1565
47.2.3.1 SCR ..... 1565
47.2.3.2 LUMA_COMP ..... 1567
47.2.3.3 CHROMA_RED ..... 1568
47.2.3.4 CHROMA_GREEN ..... 1569
47.2.3.5 CHROMA_BLUE ..... 1569
47.2.3.6 DMA_ADDR ..... 1570
47.2.3.7 DMA_INC ..... 1570
47.2.3.8 INVSZ ..... 1571
47.2.3.9 HPALRM ..... 1572
47.2.3.10 ALPHA ..... 1572
47.2.3.11 HFACTOR ..... 1573
47.2.3.12 VFACTOR ..... 1573
47.2.3.13 VID_SIZE ..... 1574
47.2.3.14 LUT_ADDR ..... 1574
47.2.3.15 LUT_DATA ..... 1575
47.3 Functional Description ..... 1576
47.3.1 ITU656 ..... 1576
47.3.2 Input Synchronizer ..... 1577
47.3.3 ITU Decoder ..... 1577
47.3.4 Down Scaling ..... 1578
47.3.5 Brightness and Contrast Adjust ..... 1579
47.3.6 YUV to RGB Conversion ..... 1580
47.3.7 Round and Dither ..... 1580
47.3.7.1 Round ..... 1580
47.3.7.2 Dither ..... 1580
47.3.8 Output Formatter ..... 1580
47.3.9 DMA and De-interlace ..... 1581
47.3.10Error Case ..... 1582
47.4 Initialization/Application Information ..... 1583
47.4.1 Initialization Information ..... 1583
47.4.2 Application Information ..... 1584
47.4.2.1 Register Configuration Timing Window ..... 1584


## Chapter 48
## Voltage Regulators and Power Supplies


48.1 Introduction ..... 1585
48.2 Power-up sequencing ..... 1585

48.3 Voltage regulators ..... 1586
48.3.1 Block diagram ..... 1587
48.3.2 External signals ..... 1587
48.3.3 Detailed signal descriptions ..... 1588
48.3.3.1 VDDR ..... 1588
48.3.3.2 VRC ..... 1588
48.4 Memory map and register definition ..... 1588
48.4.1 Voltage Regulator Control Register (VREG_CTL) ..... 1588
48.5 Functional description ..... 1589
48.5.1 High Power or Main Regulator (HPREG) ..... 1589
48.5.2 Low Power Regulator (LPREG) ..... 1589
48.5.3 Ultra Low Power Regulator (ULPREG) ..... 1589
48.5.4 Low Voltage Detectors (LVD) and Power On Reset (POR) ..... 1589
48.5.5 VREG digital interface ..... 1590
48.6 GPIO power supply configuration ..... 1590
48.7 Power domain organization ..... 1592

## Chapter 49
## Wakeup Unit (WKPU)

49.1 Overview ..... 1595
49.2 Features ..... 1596
49.3 External signal description ..... 1597
49.4 Memory map and register description ..... 1597
49.4.1 Memory map ..... 1597
49.4.2 Register description ..... 1598
49.4.2.1 NMI Status Flag Register (NSR) ..... 1599
49.4.2.2 NMI Configuration Register (NCR) ..... 1599
49.4.2.3 Wakeup/Interrupt Status Flag Register (WISR) ..... 1601
49.4.2.4 Interrupt Request Enable Register (IRER) ..... 1601
49.4.2.5 Wakeup Request Enable Register (WRER) ..... 1602
49.4.2.6 Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER) ..... 1602
49.4.2.7 Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER) ..... 1603
49.4.2.8 Wakeup/Interrupt Filter Enable Register (WIFER) ..... 1603
49.4.2.9 Wakeup/Interrupt Pullup Enable Register (WIPUER) ..... 1604
49.5 Functional description ..... 1604
49.5.1 WKPU behavior ..... 1604
49.5.2 Non-maskable interrupts ..... 1605
49.5.2.1 NMI management ..... 1606
49.5.3 External wakeups and interrupts ..... 1607
49.5.3.1 External interrupt management ..... 1608
49.5.3.2 On-chip wakeup management ..... 1608

## Chapter 50
## Device Performance Optimization

50.1 Introduction ..... 1609

50.2 Features ..... 1609
50.3 Configuring hardware features ..... 1610
50.3.1 Branch target buffer (BTB) ..... 1610
50.3.1.1 Description ..... 1610
50.3.1.2 Recommended configuration ..... 1610
50.3.2 Frequency-modulated PLL0 ..... 1611
50.3.2.1 Description ..... 1611
50.3.2.2 Recommended configuration ..... 1611
50.3.3 Flash bus interface unit ..... 1612
50.3.3.1 Description ..... 1612
50.3.3.2 Recommended configuration ..... 1612
50.3.4 Crossbar switch ..... 1612
50.3.4.1 Description ..... 1612
50.3.4.2 Recommended configuration ..... 1613
50.3.5 Cache ..... 1613
50.3.5.1 Description ..... 1613
50.3.5.2 Recommended configuration ..... 1613
50.3.6 Memory management unit (MMU) ..... 1615
50.3.6.1 Description ..... 1615
50.3.7 DRAMC Priority Manager ..... 1616
50.3.7.1 Description ..... 1616
50.4 Application software ..... 1616
50.4.1 Compiler optimizations ..... 1616
50.4.2 Signal processing extension ..... 1617
50.4.3 Hardware single precision floating point ..... 1618
50.4.4 Variable length encoding ..... 1618
50.5 Peripherals and general application guidelines ..... 1619
50.6 Performance optimization checklist ..... 1620

## Appendix A
## Registers Under Protection

## Appendix B
## Revision History

B.1 Changes between revisions 4 and 5 ..... 1639
B.2 Changes between revisions 3 and 4 ..... 1640
B.3 Changes between revisions 2 and 3 ..... 1649
B.4 Changes between revisions 1 and 2 ..... 1658

This page is intentionally left blank.

# Preface

## Overview

The primary objective of this document is to define the functionality of the MPC5645S microcontroller for use by software and hardware developers. The MPC5645S is built on Power Architecture® technology and integrates technologies that are important for today's instrument cluster applications.

The information in this document is subject to change without notice, as described in the disclaimers on the title page. As with any technical documentation, it is the reader's responsibility to be sure he or she is using the most recent version of the documentation.

To locate any published errata or updates for this document, visit the Freescale Web site at **freescale.com**.

## Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the MPC5645S device. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power Architecture.

## Chapter organization and device-specific information

This document includes chapters that describe:

- The device as a whole
- The functionality of the individual modules on the device

In the latter, any device-specific information may be presented in the section "Information specific to this device" at the beginning of the chapter.

## References

In addition to this reference manual, the following documents provide additional information on the operation of the MPC5645S:

- IEEE-ISTO 5001-2003 Standard for a Global Embedded Processor Interface (Nexus)
- IEEE 1149.1-2001 standard - IEEE Standard Test Access Port and Boundary-Scan Architecture
- Power Architecture Book E V1.0 (http://www.freescale.com/files/32bit/doc/user_guide/BOOK_EUM.pdf)

This page is intentionally left blank.

# Chapter 1
# Introduction

## 1.1 The MPC5645S microcontroller

The MPC5645S represents a new generation of 32-bit microcontrollers targeting single-chip automotive instrument cluster applications. MPC5645S devices are part of the MPC56xxS family of Power Architecture®-based devices. This family has been designed with an emphasis on providing cost-effective and high quality graphics capabilities in order to satisfy the increasing market demand for color Thin Film Transistor (TFT) displays within the vehicle cockpit. Traditional cluster functions, such as gauge drive, real time counter, and sound generation are also integrated on each device.

The MPC5645S:

- Includes 2 MB internal flash memory, 1 MB internal graphics SRAM and 64 KB system SRAM
- Offers high processing performance operating at speeds up to 125 MHz
- Is optimized for low power consumption

The MPC5645S is designed to reduce development and production costs of TFT-based instrument cluster displays by providing a single-chip solution with the processing and storage capacity to host and execute real-time application software and drive TFT displays directly.

The MPC5645S features a 2D OpenVG and raster graphics accelerator, Video Input Unit (VIU2) and two on-chip Display Control Units (DCU3 and DCULite) designed to drive two color TFT displays simultaneously. The MPC5645S includes a enhanced QuadSPI serial flash controller and an optional DRAM controller allowing graphics RAM expansion externally.

The MPC5645S is compatible with the existing development infrastructure of current Power Architecture devices and are supported with software drivers, operating systems and configuration code to assist with application development.

## 1.2 MPC5645S device summary

Table 1-1 summarizes the MPC5645S device.

**Table 1-1. MPC5645S device comparison**

| Feature MPC | 5645S | | |
|---|---|---|---|
| Package | 176 LQFP | 208 LQFP | 416 TEPBGA |
| CPU | e200z4d<br>4 KB Instruction-Cache<br>16-entry Memory Management Unit (MMU)<br>Floating Point Unit (FPU)<br>Signal Processing Extension (SPE) | | |
| Execution speed | Static–125 MHz | | |
| Flash memory (ECC) | 2 MB | | |
| RAM (ECC) | 64 KB | | |
| On-chip graphics RAM (no ECC) | 1 MB | | |
| MPU | 16 entry | | |
| eDMA | 16 channels | | |
| DRAM controller | No | | Yes |
| OpenVG Graphics Accelerator (GFX2D) | Yes (OpenVG 1.1) | | |
| Display Control Unit (DCU3) | Yes | | |
| Display Control Unit Lite (DCULite) | No | Yes | |
| Timing Controller (TCON) and RSDS interface | No | Yes | |
| Video Input Unit (VIU2) | Yes | | |
| QuadSPI serial flash interface | Yes | | |
| Stepper Motor Controller (SMC) | 4 motors | 6 motors | |
| Stepper Stall Detect (SSD) | Yes | | |
| Sound Generator Module (SGM) | Yes | | |
| 32 kHz external crystal oscillator | Yes | | |
| Real Time Counter and Autonomous Periodic Interrupt (RTC/API) | Yes | | |
| Periodic interrupt timer (PIT) | 8 ch, 32-bit | | |
| Software Watchdog Timer (SWT) | Yes | | |
| System Timer Module (STM) | 4 ch, 32-bit | | |
| Timed I/O[1] | 20 ch, 16-bit: IC / OC / OPWM<br>8 ch, 16-bit: IC / OC<br>4 ch, 16-bit: IC / OC / OPWM / QDEC | | |
| Analog-to-Digital Converter (ADC)[2] | 16 channels, 10-bit | 20 channels, 10-bit | |
| CAN (64 mailboxes) | 3 × FlexCAN | | |
| CAN sampler | Yes | | |
| Serial communication interface | 3 × LINFlex | 4 × LINFlex | |

**Table 1-1. MPC5645S device comparison (continued)**

| Feature MPC | 5645S | | |
|---|---|---|---|
| Package | 176 LQFP | 208 LQFP | 416 TEPBGA |
| SPI | 2 × DSPI | 3 × DSPI | |
| $I^2C$ | 4 | | |
| GPIO | 128 | 150 | 177[3] |
| Debug | Nexus Class 3 (4×MDO)[4] | | Nexus Class 3 (12×MDO) |

1 IC-Input Capture, OC-Output Compare, OPWM-Output Pulse Width Modulation, QDEC- Quadrature Decode Mode

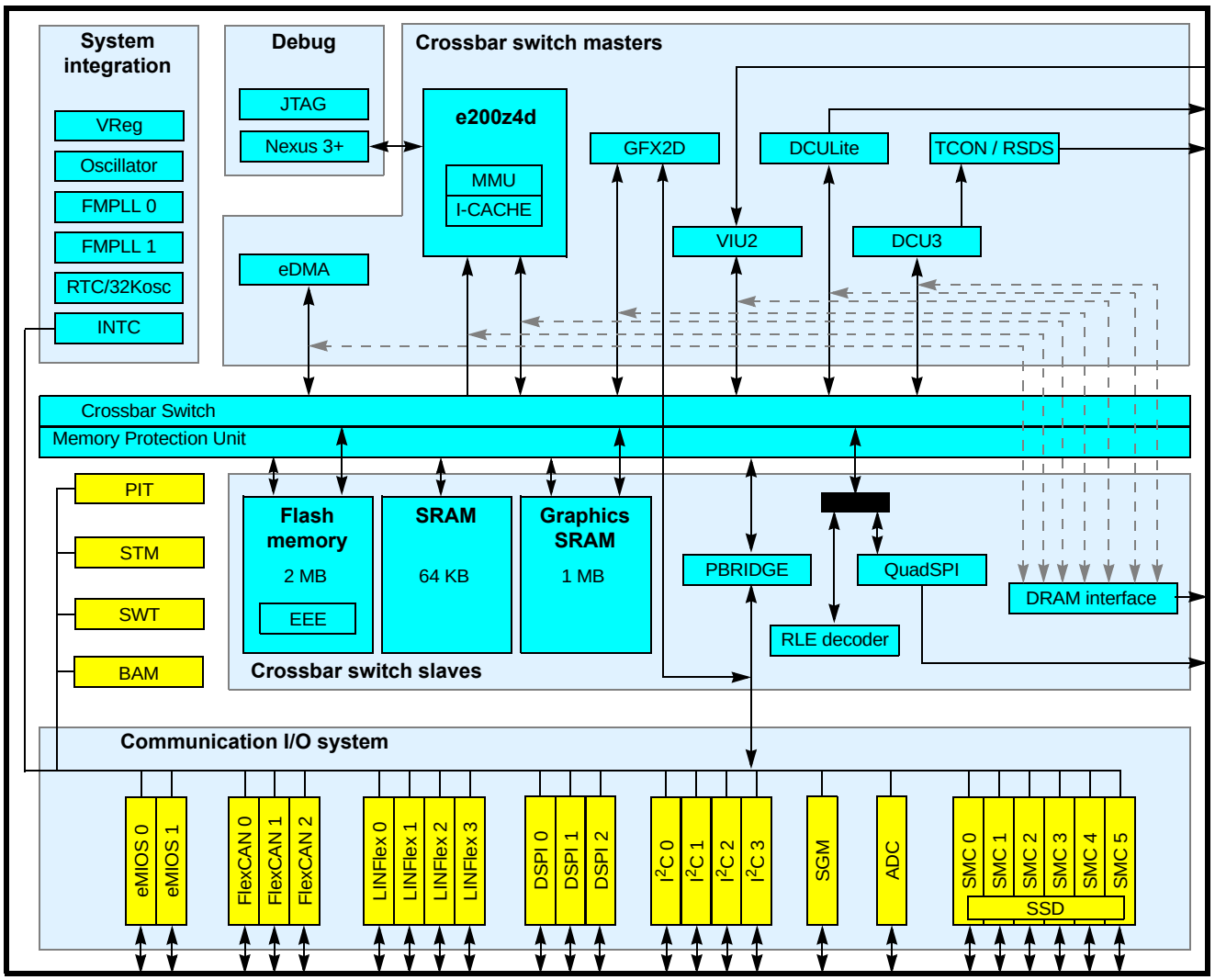
2 Support for external multiplexer enabling up to 8 channels

3 The 416-pin GPIO count does not include the DRAM interface, which is dedicated to DRAM only.

4 Nexus pins are multiplexed with other functional pins on 176 LQFP and 208 LQFP package options.

## 1.3 Device block diagram

Figure 1-1 shows a top-level block diagram of the MPC5645S.




**ADC** – Analog-to-Digital Converter
**BAM** – Boot Assist Module
**eDMA** – Enhanced Direct Memory Access Controller
**DCU3** – Display Control Unit
**DCULite** – Display Control Unit Lite
**DSPI** – Serial Peripherals Interface
**eMIOS** – Enhanced Modular Input/Output System
**FlexCAN** – Controller Area Network Controller
**FMPLL** – Frequency-Modulated Phase-Locked Loop
**GFX2D** – OpenVG Graphics Accelerator
**INTC** – Interrupt Controller
**JTAG** – Joint Test Action Group interface
**MMU** – Memory Management Unit
**QuadSPI** – Quad IO serial flash interface
**PBRIDGE** – Peripheral Bridge
**PIT** – Periodic Interrupt Timer
**RLE** – Run Length Encoding
**RSDS** – Reduced-Swing Differential Signal interface
**RTC** – Real Time Clock
**SGM** – Sound Generator Module
**SMC** – Stepper Motor Controller
**SSD** – Stepper Stall Detect
**STM** – System Timer Module
**SWT** – Software Watchdog Timer
**TCON** – Timing Controller
**VIU2** – Video Input Unit
**VReg** – Voltage regulator

**Figure 1-1. MPC5645S block diagram**

## 1.4 Feature summary

- Dual-issue, 32-bit Power Architecture Book E compliant CPU core complex (e200z4d)

  - Memory Management Unit (MMU)
  - 4 KB, 2/4-way instruction cache
- 2 MB on-chip ECC flash memory with:
  - Flash memory controller
  - Prefetch buffers
- 64 KB on-chip ECC SRAM
- 1 MB on-chip non-ECC graphics SRAM with two-port graphics SRAM controller
- Memory Protection Unit (MPU) with up to 16 region descriptors and 32-byte region granularity to provide basic memory access permission and ensure separation between different codes and data
- Interrupt Controller (INTC) with 181 peripheral interrupt sources and eight software interrupts
- Two Frequency-Modulated Phase-Locked Loops (FMPLLs)
  - Primary FMPLL (FMPLL0) provides a system clock up to 125 MHz
  - Auxiliary FMPLL (FMPLL1) is available for use as an alternate, modulated or non-modulated clock source to eMIOS modules, QuadSPI and as alternate clock to the DCU and DCU-Lite for pixel clock generation
- Crossbar switch architecture enables concurrent access of peripherals, flash memory or RAM from multiple bus masters
- 16-channel Enhanced Direct Memory Access controller (eDMA) with multiple transfer request sources using a DMA channel multiplexer
- Boot Assist Module (BAM) with 8 KB dedicated ROM for embedded boot code supports boot options including download of boot code via a serial link (CAN or SCI)
- Two Display Control Units (DCU3 and DCULite) for direct drive of up to two TFT LCD displays up to XGA resolution
- Timing Controller (TCON) and RSDS interface for the DCU3 module
- 2D OpenVG 1.1 and raster graphics accelerator (GFX2D)
- Video Input Unit (VIU2) supporting 8/10-bit ITU656 video input, YUV to RGB conversion, video down-scaling, de-interlacing, contrast adjustment and brightness adjustment.
- DRAM controller supporting DDR1, DDR2, and LPDDR1 DRAMs
- Stepper Motor Controller (SMC)
  - High-current drivers for up to six instrument cluster gauges driven in full dual H-bridge configuration
  - Stepper motor return-to-zero and stall detection module
  - Stepper motor short circuit detection
- Sound Generator Module (SGM)
  - 4-channel mixer
  - Supports PCM wave playback and synthesized tones
  - Optional PWM or $I^2S$ outputs
- Two 16-channel Enhanced Modular Input Output System (eMIOS) modules

- Support a range of 16-bit Input Capture, Output Compare, Pulse Width Modulation and Quadrature Decode functions

- 10-bit Analog-to-Digital Converter (ADC) with a maximum conversion time of 1 µs
  - Up to 20 internal channels
  - Up to 8 external channels
- Three Deserial Serial Peripheral Interface (DSPI) modules for full-duplex, synchronous, communications with external devices
- QuadSPI serial flash memory controller
  - Supports single, dual and quad IO serial flash memory
  - Interfaces to external, memory-mapped serial flash memories
  - Supports simultaneous addressing of 2 external serial flashes to achieve up 80 MB/s read bandwidth
- RLE decoder supporting memory to memory decoding of RLE data in conjunction with eDMA
- Four local interconnect network (LINFlex) controller modules
  - Capable of autonomous message handling (master), autonomous header handling (slave mode), and UART support
  - Compliant with LIN protocol rev 2.1
- Three controller-area network (FlexCAN) modules
  - Compliant with the CAN protocol version 2.0 C
  - 64 configurable buffers
  - Programmable bit rate of up to 1 Mb/s
- Four Inter-Integrated Circuit ($I^2C$) internal bus controllers with master/slave bus interface
- Low-power loop controlled pierce crystal oscillator supporting 4–16MHz external crystal or resonator
- Real Time Counter (RTC) with clock source from internal 128 kHz or 16 MHz oscillator supporting autonomous wake-up with 1 ms resolution with maximum timeout of 2 seconds
  - Support for real time counter (RTC) with clock source from external 32 KHz crystal oscillator, supporting wake-up with 1 s resolution and maximum timeout of one hour
  - RTC optionally clocked by fast 4–16 MHz external oscillator
- System timers:
  - Four-channel 32-bit System Timer Module (STM)
  - Eight-channel 32-bit Periodic Interrupt Timer (PIT) module (including ADC trigger)
  - Software Watchdog Timer (SWT)
- System Integration Unit Lite (SIUL) module to manage external interrupts, GPIO and pad control
- System Status and Configuration Module (SSCM)
  - Provides information for identification of the device, last boot mode, or debug status
  - Provides an entry point for the censorship password mechanism
- Clock Generation Module (MC_CGM) to generate system clock sources and provide a unified register interface, enabling access to all clock sources

- Clock Monitor Unit (CMU)
  - Monitors the integrity of the fast (4–16 MHz) external crystal oscillator and the primary FMPLL (FMPLL0)
  - Acts as a frequency meter, measuring the frequency of one clock source and comparing it to a reference clock
- Mode Entry Module (MC_ME)
  - Controls the device power mode, i.e., RUN, HALT, STOP, or STANDBY
  - Controls mode transition sequences
  - Manages the power control, voltage regulator, clock generation and clock management modules
- Power Control Unit (MC_PCU) to implement standby mode entry/exit and control connections to power domains
- Reset Generation Module (MC_RGM) to manage reset assertion and release to the device at initial power-up
- Nexus Development Interface (NDI) per IEEE-ISTO 5001-2008 Class 3 standard with additional Class 4 features:
  - Watchpoint Triggering
  - Processor Overrun Control
- Device/board boundary-scan testing supported per Joint Test Action Group (JTAG) of IEEE (IEEE 1149.1)
- On-chip voltage regulator controller for regulating the 3.3–5 V supply voltage down to 1.2 V for core logic (requires external ballast transistor)
- Package:[1]
  - 176 LQFP, 0.5 mm pitch, 24 mm × 24 mm outline
  - 208 LQFP, 0.5 mm pitch, 28 mm × 28 mm outline
  - 416 TEPBGA, 1mm ball pitch, 27 mm × 27 mm outline

## 1.5 Feature details

### 1.5.1 Low-power operation

The MPC5645S is designed for optimized low-power operation and dynamic power management of the CPU and peripherals. Power management features include software-controlled clock gating of peripherals and multiple power domains to minimize leakage in low-power modes.

There are three low-power modes:

- STANDBY
- STOP

1. See the device comparison table for package offerings for each device in the family.

- HALT

and five dynamic power modes — RUN[0..3] and DRUN. All low-power modes use clock gating to halt the clock for all or part of the device.

STANDBY mode turns off the power to the majority of the chip to offer the lowest power consumption mode.

The device can be awakened from STANDBY mode via from any of up to 23 I/O pins, a reset or from a periodic wake-up using a low power oscillator. If required, it is possible to enable the internal 16 MHz oscillator, the external 4–16 MHz oscillator, and the external 32 KHz oscillator.

In STANDBY mode the contents of the CPU, on-chip peripheral registers, and potentially some of the volatile memory are lost. The two possible configurations in STANDBY mode are:

- The device retains 64 KB of the on-chip SRAM, but the content of the graphics SRAM is lost.
- The device retains 8 KB of the on-chip SRAM, but the content of the graphics SRAM is lost.

STOP mode maintains power to the entire device allowing the retention of all on-chip registers and memory, and providing a faster recovery low power mode than the lowest-power STANDBY mode. There is no need to reconfigure the device before executing code. The clocks to the CPU and peripherals are halted and can be optionally stopped to the oscillator or PLL at the expense of a slower start-up time.

STOP is entered from RUN mode only. Wake-up from STOP mode is triggered by an external event or by the internal periodic wake-up, if enabled.

RUN modes are the main operating modes where the entire device can be powered and clocked and from which most processing activity is done. Four dynamic RUN modes are supported—RUN0 - RUN3. The ability to configure and select different RUN modes enables different clocks and power configurations to be supported with respect to each other and to allow switching between different operating conditions. The necessary peripherals, clock sources, clock speed, and system clock prescalers can be independently configured for each of the four RUN modes of the device.

HALT mode is a reduced activity, low power mode intended for moderate periods of lower processing activity. In this mode the CPU system clocks are stopped but user-selected peripheral tasks can continue to run. It can be configured to provide more efficient power management features (switch-off PLL, flash memory, main regulator, etc.) at the cost of longer wake up latency. The system returns to RUN mode as soon as an event or interrupt is pending.

The following table summarizes the operating modes of the MPC5645S.

**Table 1-2. Operating mode summary**

| Operating mode | SoC features | | | | | Clock sources | | | | | | Periodic Wake-up | Wake-up input | VREG mode | Wake-up time[1] | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CPU GFX accelerator DRAM controller | Peripherals | Flash | SRAM | Graphics RAM | Primary PLL | Auxiliary PLL | 16MHz IRC | 4-16MHz OSC | 128KHz IRC | 32KHz X OSC | | | | VREG start-up | IRC Wake-up | Flash Recovery | OSC Stabilization | PLL Lock | S/W Reconfig | Mode switch over |
| RUN | On | OP | OP | OP[2] | On | OP | OP | On | OP | On | OP | — | — | FP | — | — | — | — | — | — | — |
| HALT | CG | OP | OP | OP[2] | On | OP | OP | On | OP | On | OP | OP | OP | FP | — | — | — | — | — | — | 30 µs [3] |
| STOP | CG | CG | CG | OP[2] | CG | CG | CG | OP | OP | On | OP | OP | OP | LP | 350 µs | 4 µs | 20 µs | 1 ms | 200 µs | — | 30 µs [3] |
| STAND BY | Off | Off | Off | 64K [4] | Off | Off | Off | OP | OP | On | OP | OP | OP | LP | 350 µs | 8 µs | 100 µs | 1 ms | 200 µs | Var | 28 µs |
| | Off | Off | Off | 8K[5] | Off | Off | Off | OP | OP | On | OP | OP | OP | LP | 200 µs | 8 µs | 100 µs | 1 ms | 200 µs | Var | 28 µs |
| POR | | | | | | | | | | | | | | | 500 µs | 8 µs | 100 µs | 1 ms | 200 µs | | BAM[6] |

Table key:
- On — Powered and clocked
- OP — Optionally configurable to be enabled or disabled (clock gated)
- CG — Clock Gated, Powered but clock stopped
- Off — Powered off and clock gated
- FP — VREG Full Performance mode
- LP — VREG Low Power mode, reduced output capability of VREG but lower power consumption
- Var — Variable duration, based on the required reconfiguration and execution clock speed
- BAM — Boot Assist Module Software and Hardware used for device start-up and configuration

1 A high level summary of some key durations that need to be considered when recovering from low power modes. This does not account for all durations at wake up. Other delays will be necessary to consider including, but not limited to the external supply start-up time.
IRC Wake-up time must not be added to the overall wake-up time as it starts in parallel with the VREG.
All other wake-up times must be added to determine the total start-up time.

2 Either 64 KB or 8 KB available.

3 Flash configured in disabled mode.

4 64 KB of the RAM contents is retained, but not accessible in STANDBY mode.

5 8 KB of the RAM contents is retained, but not accessible in STANDBY mode.

6 Dependent on boot option after reset.

Additional notes on low power operation:

- Fast wake-up using the on-chip 16 MHz internal RC oscillator allows rapid execution from RAM on exit from low power modes
- The 16 MHz internal RC oscillator supports low speed code execution and clocking of peripherals when it is selected as the system clock and can also be used as the PLL input clock source to provide fast start-up without the external oscillator delay
- The device includes an internal voltage regulator that includes the following features:
  - Regulates input to generate all internal supplies
  - Manages power gating

- External ballast transistor for high power regulator
- Low-Power and Ultra-Low-Power regulators support operation when in STOP and STANDBY modes, respectively, to minimize power consumption
- Startup on-chip regulators in <350µs for rapid exit of STOP and STANDBY modes
- Low voltage detection on main supply and 1.2 V regulated supplies

### 1.5.2 e200z4d core

The e200z4d Power Architecture® core provides the following features:

- Dual issue, 32-bit *Power Architecture Book E* compliant CPU
- Implements the VLE APU for reduced code footprint
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
  - Dedicated branch address calculation adder
  - Branch target prefetching using 8-entry BTB
- Supports independent instruction and data accesses to different memory subsystems, such as SRAM and Flash memory via independent Instruction and Data BIUs.
- Load/store unit
  - 2 cycle load latency
  - Fully pipelined
  - Big and Little endian support
  - Misaligned access support
- 64-bit General Purpose Register file
- Dual AHB 2.v6 64-bit System buses
- Memory Management Unit (MMU) with 16-entry fully-associative TLB and multiple page size support
- 4 KB, 2/4-Way Set Associative Instruction Cache
- Signal Processing Extension (SPE1.1) APU supporting SIMD fixed-point operations using the 64-bit General Purpose Register file
- Embedded Floating-Point (EFP2) APU supporting scalar and vector SIMD single-precision floating-point operations, using the 64-bit General Purpose Register file
- Nexus Class 3 real-time Development Unit
- Dynamic power management of execution units, cache and MMU

### 1.5.3 Crossbar switch (XBAR)

The XBAR multi-port crossbar switch supports simultaneous connections between seven master ports and eight slave ports. The crossbar supports a 32-bit address bus width and a 64-bit data bus width.

The crossbar allows concurrent transactions to occur from any master port to any slave port but one of those transfers must be an instruction fetch from internal flash. If a slave port is simultaneously requested by more than one master port, arbitration logic selects the higher priority master and grants it ownership of the slave port. All other masters requesting that slave port are stalled until the higher priority master completes its transactions. Requesting masters having equal priority are granted access to a slave port in round-robin fashion, based upon the ID of the last master to be granted access.

The crossbar provides the following features:

- Seven master ports:
  - e200z4d core instruction port
  - e200z4d core complex load/store data port
  - eDMA controller
  - DCU
  - DCULite
  - VIU
  - 2D Graphics Accelerator (GFX2D)
- Seven slave ports:
  - Platform Flash Controller (2 Ports)
  - Platform SRAM Controller
  - Graphics SRAM Controller (2 Ports)
  - QuadSPI serial flash Controller and RLE Decoder
  - Peripheral Bridge
- 32-bit internal address bus, 64-bit internal data bus
- Programmable Arbitration Priority
  - Requesting masters can be treated with equal priority and will be granted access to a slave port in round-robin fashion, based upon the ID of the last master to be granted access, or a priority order can be assigned by software at application run time
- Temporary dynamic priority elevation of masters

### 1.5.4 Enhanced Direct Memory Access (eDMA)

The eDMA module is a controller capable of performing complex data movements via 16 programmable channels, with minimal intervention from the host processor. The hardware micro architecture includes a DMA engine which performs source and destination address calculations, and the actual data movement operations, along with an SRAM-based memory containing the transfer control descriptors (TCD) for the channels. This implementation is utilized to minimize the overall block size. The eDMA module provides the following features:

- 16 channels support independent 8-, 16-, or 32-bit single value or block transfers
- Supports variable sized queues and circular queues
- Source and destination address registers are independently configured to post-increment or remain constant

- Each transfer is initiated by a peripheral, CPU, periodic timer interrupt or eDMA channel request
- Each DMA channel can optionally send an interrupt request to the CPU on completion of a single value or block transfer
- DMA transfers possible between system memories, QuadSPI, RLE Decoder, DSPIs, $I^2C$, ADC, eMIOS, and General Purpose I/Os (GPIOs)
- Programmable DMA Channel Mux allows assignment of any DMA source to any available DMA channel with up to a total of 64 potential request sources

### 1.5.5 Interrupt Controller (INTC)

The INTC (interrupt controller) provides priority-based preemptive scheduling of interrupt requests, suitable for statically scheduled hard real-time systems.

For high priority interrupt requests, the time from the assertion of the interrupt request from the peripheral to when the processor is executing the interrupt service routine (ISR) has been minimized. The INTC provides a unique vector for each interrupt request source for quick determination of which ISR needs to be executed. It also provides an ample number of priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. To allow the appropriate priorities for each source of interrupt request, the priority of each interrupt request is software configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the priority ceiling protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource can not preempt each other.

Multiple processors can assert interrupt requests to each other through software setable interrupt requests. These same software setable interrupt requests also can be used to break the work involved in servicing an interrupt request into a high priority portion and a low priority portion. The high priority portion is initiated by a peripheral interrupt request, but then the ISR asserts a software setable interrupt request to finish the servicing in a lower priority ISR. Therefore these software setable interrupt requests can be used instead of the peripheral ISR scheduling a task through the RTOS. The INTC provides the following features:

- Unique 9-bit vector for each of the possible 128 separate interrupt sources
- Eight software triggerable interrupt sources
- 16 priority levels with fixed hardware arbitration within priority levels for each interrupt source
- Ability to modify the ISR or task priority
  - Modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
- External non maskable interrupt directly accessing the main CPU critical interrupt mechanism
- 32 external interrupts

### 1.5.6 QuadSPI serial flash memory controller

The QuadSPI module enables use of external serial flash memories supporting single, dual, and quad modes of operation. It features the following:

- Maximum serial clock frequency 80 MHz

- Memory mapped read access for AHB crossbar switch masters
- Automatic serial flash read command generation by CPU, eDMA, DCU, or DCULite read access on AHB bus
- Supports single, dual, and quad serial flash read commands
- Simultaneous mode:
  - Supports concurrent read of two external serial flashes
  - The quad data streams from the two flashes can be recombined in the QuadSPI to achieve up to 80 MB/s read bandwidth with 80 MHz serial flash
- 16×64-bit buffer with speculative fetch and buffer flush mechanisms to maximize read bandwidth of serial flash
- DMA support
- All Serial Flash program, erase, read, and configuration commands available via IP bus interface

### 1.5.7 System Integration Unit Lite (SIUL)

The SIUL controls MCU reset configuration, pad configuration, external interrupt, general purpose I/O (GPIO), internal peripheral multiplexing, and the system reset operation.

The GPIO features the following:

- Up to four levels of internal pin multiplexing, allowing exceptional flexibility in the allocation of device functions for each package
- Centralized general purpose input output (GPIO) control
- All GPIO pins can be independently configured to support pull-up, pull down, or no pull
- Reading and writing to GPIO supported both as individual pins and 16-bit wide ports
- All peripheral pins can be alternatively configured as both general purpose input or output pins except ADC channels which support alternative configuration as general purpose inputs
- Direct readback of the pin value supported on all digital output pins through the SIU
- Configurable digital input filter that can be applied to up to 24 general purpose input pins for noise elimination on external interrupts
- Register configuration protected against change with soft lock for temporary guard or hard lock to prevent modification until next reset

### 1.5.8 On-chip flash memory with ECC

The MPC5645S microcontroller has the following flash memory features:

- 2 MB of flash memory
  - Typical flash memory access time: 0 wait-state for buffer hits, 3 wait-states for page buffer miss at 125 MHz
  - Two 4×128-bit page buffers with programmable prefetch control
    - One set of page buffers can be allocated for code-only, fixed partitions of code and data, all available for any access

- One set of page buffers allocated to Display Controller Units, Graphics Accelerator, and the eDMA

— 64-bit ECC with single-bit correction, double-bit detection for data integrity

- Small block flash arrangement to support features such as boot block, EEPROM Emulation, operating system block
  - — 8×16 KB
  - — 2×64 KB
  - — 2×128 KB
  - — 6×256 KB
- Hardware managed flash writes, erase and verify sequence
- Censorship protection scheme to prevent Flash content visibility

### 1.5.9 SRAM

The MPC5645S microcontroller has 64 KB general-purpose on-chip SRAM with the following features:

- Typical SRAM access time: 1 wait-state for reads and 32-bit writes
- 32-bit ECC with single-bit correction, double bit detection for data integrity
- Supports byte (8-bit), half word (16-bit), word (32-bit), and double-word (64-bit) writes for optimal use of memory
- User transparent ECC encoding and decoding for byte, half word, and word accesses
- Separate internal power domains applied to 56 KB and 8 KB SRAM blocks during STANDBY modes to retain contents during low power mode

### 1.5.10 On-chip graphics SRAM

The MPC5645S microcontroller has 1 MB on-chip graphics SRAM with the following features:

- Two crossbar slave ports:
  - — One dedicated to the 2D Graphics Accelerator (GFX2D) access
  - — One dedicated to all other crossbar masters
- Usable as general purpose SRAM
- Supports byte (8-bit), half word (16-bit), word (32-bit), and double-word (64-bit) writes for optimal use of memory
- RAM controller with hardware RAM fill function supporting all-zeroes or all-ones SRAM initialization
- Independent data buffers (one per AHB port) for maximum system performance
  - — Optimized for burst transfers (read + write)
  - — Programmable read prefetch capabilities

### 1.5.11 Memory Protection Unit (MPU)

The MPU features the following:

- Sixteen region descriptors for per master protection
- Start and end address defined with 32-byte granularity
- Overlapping regions supported
- Protection attributes can optionally include process ID
- Protection offered for 4 concurrent read ports
- Read and write attributes for all masters
- Execute and supervisor/user mode attributes for processor masters

### 1.5.12 2D Graphics Accelerator (GFX2D)

- Native vector graphics rendering
  - Compatible with OpenVG1.1
  - Complete hardware OpenVG 1.1 rendering pipeline
  - Both geometry and pixel processing
  - Adaptive processing of Bezier curves and strokes
- 16-sample edge anti-aliasing
  - High image quality, font scalability, etc.
  - 4× Rotated Grid Supersampling (RGSS) AA for Flash
- 3D perspective texturing, reflections, and shadowing
- Shading (linear or radial gradient)
- Separate 2D engine for BitBlt, fill, and ROP operations
- Significant performance improvement when compared to software or 3D GPU-based OpenVG implementations

### 1.5.13 Display Control Unit (DCU3)

The DCU3 is a display controller designed to drive TFT LCD displays up to WVGA resolution using direct blit graphics and video.

The DCU3 generates all the necessary signals required to drive the TFT LCD displays: up to 24-bit RGB data bus, Pixel Clock, Data Enable, Horizontal-Sync, and Vertical-Sync.

The flexible architecture of the DCU3 enables the display of OpenVG-rendered frame buffer content and direct blit rendered graphics simultaneously.

An optional Timing Controller (TCON) and RSDS interface is available to directly drive the row and column drivers of a display panel.

Internal memory resource of the device allows to easily handle complex graphics contents (pictures, icons, languages, fonts).

The DCU3 supports 4-plane blending and 16 graphics layers. Control Descriptors (CDs) associated with each of the 16 layers enable effective merging of different resolutions into one plane to optimize use of internal memory buffers. A layer may be constructed from graphic content of various resolutions including indexed colors of 1, 2, 4, and 8 bpp, direct colors of 16, 24, and 32 bpp, and a YUV 4:2:2 color space. The ability of the DCU3 to handle input data in resolutions as low as 1bpp, 2bpp, and 4bpp enables a highly efficient use of internal memory resources of the MPC5645S. A special tiled mode can be enabled on any of the 16 layers to repeat a pattern optimizing graphic memory usage.

A hardware cursor can be managed independently of the layers at blending level increasing the efficient use of the internal DCU3 resources.

To secure the content of all critical information to be displayed, a safety mode can be activated to check the integrity of critical data along the whole system data path from the memory to the TFT pads.

The DCU3 features the following:

- Display color depth: up to 24 bpp
- Generation of all RGB and control signals for TFT
- Four-plane blending
- Maximum number of Input Layers: 16 (fixed priority)
- Dynamic Look-Up-Table (Color and Gamma Look-Up)
- $\alpha$–blending range: up to 256 levels
- Transparency Mode
- Gamma Correction
- Tiled mode on all the layers
- Hardware Cursor
- Supports YCrCb 4:2:2 input data format
- RLE decode inline supporting direct read of RLE compressed images from system memory
- Critical display content integrity monitoring for Functional Safety support
- Internal Direct Memory Access (DMA) module to transfer data from internal and / or external memory
- Support displays up to 800 x 480 pixel resolutions

The DCU3 also features a Parallel Data Interface (PDI) to receive external digital video or graphic content into the DCU3. The PDI input is directly injected into the DCU3 background plane FIFO. When the PDI is activated, all the DCU3 synchronization is extracted from the external video stream to guarantee the synchronization of the two video sources.

The PDI can be used to:

- Connect a video camera output directly to the PDI
- Connect a secondary display driver as slave with a minimum of extra cost
- Connect a device gathering various Video sources
- Provide flexibility to allow the DCU to be used in slave mode (external synchronization)

The PDI features the following:

- Supported color modes:
  - 8-bit mono
  - 8-bit color multiplexed
  - RGB565
  - 16-bit/18-bit RAW color
- Supported synchronization modes:
  - embedded ITU-R BT.656-4 (RGB565 mode 2)
  - HSYNC, VSYNC
  - Data Enable
- Direct interface with DCU3 background plane FIFO
- Synchronization generation for the DCU3

### 1.5.14 Display Control Unit Lite (DCULite)

The DCULite is a display controller designed to enable the MPC5645S to drive a second TFT LCD display up to XGA resolution using direct blit graphics and video. The DCULite includes all features of the DCU3, including the PDI with the following exceptions:

- Reduced from 4-plane to 2-plane blending
- Reduced from 16 layers to 4 layers
- Reduced CLUT size

### 1.5.15 Timing Controller (TCON) and RSDS interface

The TCON enables direct drive of the row and column drivers of display panels enabling emulation of TCON ICs used in display panels.

- Programmable Timing Generation unit featuring 12 waveform generators allowing high degree of flexibility in panel waveform generation
- Reduced Swing Differential Signaling (RSDS) interface for RGB data and pixel clock
- Conforms to “RSDS ‘Intra Panel’ Interface Specification” Rev. 1.0 (National Semiconductor)

### 1.5.16 RLE decoder

The RLE decoder is a crossbar slave sharing a slave port with the QuadSPI module. The platform eDMA is used to stream compressed image data into and extract decompressed data out of the RLE Decoder.

- Lossless decompression
- Pixel formats supported: 8 bpp, 16 bpp, 24 bpp, and 32 bpp
- AHB mapped read and write registers in RLE_DEC to achieve higher throughput
- Programmable fill levels of read and write buffers for initiating burst transfers
- Crop feature: Support for selectively reading out a part of decompressed image data taking complete compressed data for the full image as input

### 1.5.17 DRAM controller

The DRAM controller is a multi-port DRAM controller supporting LPDDR1, DDR-1, and DDR-2 memories. The DRAM controller listens to the incoming requests to the seven buses in parallel and then sends commands to the DRAM from the highest priority bus at the current time

The seven incoming 64-bit buses are:

- DCU3
- DCULite
- e200z4d core — instruction bus
- e200z4d core — data bus
- VIU2
- GFX2D
- eDMA

The DRAM controller features the following:

- Supports CAS latency of 2, 3, and 4 clock cycles
- Master buses
  - 7 incoming master buses
  - Supports 16-byte and 32-byte bursts
  - Supports byte enables
  - Supports 4-bit priority signal for each bus
- Write buffer contains five 32-byte entries
- Supports 16-wide and 32-wide DDR1, DDR2, and LPDDR1 DRAM devices
- Controller supports one chip select, 8-bank DRAM system
- Supports dynamic on-die termination in the host device and in the DRAM
- Supports memory sizes as small as 64 Mbit

### 1.5.18 Video Input Unit (VIU2)

The VIU2 is a crossbar master module accepting an ITU656 compatible video input stream on a parallel interface, converting the pixel data to RGB or YUV format and transferring the video image to internal frame buffer memory or external DRAM if available.

- Supports 8-bit/10-bit ITU656 video input
- Output formats:
  - RGB888
  - RGB565
  - 8-bit monochrome
  - YCrCb 4:2:2
- Video downscaling
- Contrast and Brightness adjustment

- De-interlace for interlaced video image
- Internal DMA engine for data transfer to memory

### 1.5.19 Boot Assist Module (BAM)

The BAM is a block of read-only memory that is programmed once by Freescale. The BAM program is executed every time the MCU is powered-on or reset in normal mode. The BAM supports different modes of booting. They are:

- Booting from internal flash memory
- Serial boot loading (A program is downloaded into RAM via FlexCAN or LINFlex and then executed)
- Booting from external memory

Additionally the BAM:

- Enables and manages the transition of the MCU from reset to user code execution
- Configures device for serial bootload
- Enables multiple bootcode starting locations out of reset through implementation of search for valid Reset Configuration Halfword

### 1.5.20 Enhanced Modular Input/Output System (eMIOS)

This device has two eMIOS modules, each with 16 channels supporting a range of 16-bit Input Capture, Output Compare, Pulse Width Modulation, and Quadrature Decode functions.

- Selectable clock source from primary FMPLL, secondary FMPLL, external 4 - 16 MHz oscillator or 16 MHz Internal RC oscillator on a per module basis
- Timed I/O channels with 16-bit counter resolution
- Buffered updates
- Support for shifted PWM outputs to minimize occurrence of concurrent edges
- Edge aligned output pulse width modulation
  - Programmable pulse period and duty cycle
  - Supports 0% and 100% duty cycle
  - Shared or independent time bases
- Programmable phase shift between channels
- 4 channels of Quadrature Decode
- DMA transfer support

### 1.5.21 Analog-to-Digital Converter (ADC)

The ADC features the following:

- 10-bit A/D resolution
- 0–5 V or 0–3.3 V common mode conversion range

- Supports conversions speeds of up to 1µs
- 20 internal and 8 external channels support
- Up to 20 single-ended inputs channels
  - 10 channels configured as input only pins
    - 10-bit ± 2 counts accuracy (TUE)
  - 10 channels configured to have alternate function as general purpose input/output pins
    - 10-bit ± 3 counts accuracy (TUE)
- External multiplexer support to increase up to 27 channels
  - Automatic 1 × 8 multiplexer control
  - External multiplexer connected to a dedicated input channel
  - Shared register between the 8 external channels
- Result register available for every non-multiplexed channel
- Configurable Left or Right aligned result format
- Supports for one-shot, scan, and injection conversion modes
- Injection mode status bit implemented on adjacent 16-bit register for each result
  - Supports Access to Result and injection status with single 32-bit read
- Independently enabling of function for channels:
  - Pre-sampling
  - Offset error cancellation
  - Offset Refresh
- Conversion Triggering support
  - Internal conversion triggering from periodic interrupt timer (PIT)
- Four configurable analog comparator channels offering range comparison with triggered alarm
  - Greater than
  - Less than
  - Out of range
- All unused analog pins available as general purpose input pins
- Selected unused analog pins available as general purpose pins
- Power Down mode
- Optional support for DMA transfer of results

### 1.5.22 Deserial Serial Peripheral Interface (DSPI)

The DSPI modules provide a synchronous serial interface for communication between the MCU and external devices.

The DSPI features:

- Full duplex, synchronous transfers
- Master or slave operation

- Programmable master bit rates
- Programmable clock polarity and phase
- End-of-transmission interrupt flag
- Programmable transfer baud rate
- Programmable data frames from 4 to 16 bits
- Up to 3 chip select lines available, depending on package and pin multiplexing, enable 8 external devices to be selected using external muxing from a single DSPI
- Eight clock and transfer attributes registers
- Chip select strobe available as alternate function on one of the chip select pins for de-glitching
- FIFOs for buffering up to 4 transfers on the transmit and receive side
- General purpose I/O functionality on pins when not used for SPI
- Queueing operation possible through use of eDMA

### 1.5.23 FlexCAN

The MPC5645S includes up to three controller area network (FlexCAN) modules. The FlexCAN module is a communication controller implementing the CAN protocol according to Bosch Specification version 2.0B. The CAN protocol was designed to be used primarily as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness, and required bandwidth.

Each FlexCAN module offers the following:

- Compliant with CAN protocol specification, Version 2.0B active
- 64 mailboxes, each configurable as transmit or receive
  - Mailboxes configurable while module remains synchronized to CAN bus
- Transmit features
  - Supports configuration of multiple mailboxes to form message queues of scalable depth
  - Arbitration scheme according to message ID or message buffer number
  - Internal arbitration to guarantee no inner or outer priority inversion
  - Transmit abort procedure and notification
- Receive features
  - Individual programmable filters for each mailbox
  - 8 mailboxes configurable as a 6-entry receive FIFO
  - 8 programmable acceptance filters for receive FIFO
- Programmable clock source
  - System clock
  - Direct oscillator clock to avoid PLL jitter
- Listen only mode capabilities
- CAN Sampler

- Can catch the 1st message sent on the CAN network while the MCU is stopped, which guarantees a clean startup of the system without missing messages on the CAN network
- The CAN sampler is connected to one of the CAN RX pins

### 1.5.24 Serial communication interface module (LINFlex)

The MPC5645S devices include up to four LINFlex modules and support for LIN Master mode, LIN Slave mode, and UART mode. The modules are LIN state machine compliant to the LIN 1.3, 2.0, and 2.1 Specifications and handle LIN frame transmission and reception without CPU intervention.

Other features include:

- Autonomous LIN frame handling
- Message buffer to store identifier and up to eight data bytes
- Supports message length of up to 64 bytes
- Detection and flagging of LIN errors
- Sync field; Delimiter; ID parity; Bit, Framing; Checksum and Timeout errors
- Classic or extended checksum calculation
- Configurable Break duration of up to 36-bit times
- Programmable Baud rate prescalers (13-bit mantissa, 4-bit fractional)
- Diagnostic features
  - Loop back
  - Self Test
  - LIN bus stuck dominant detection
- Interrupt driven operation with 16 interrupt sources
- LIN slave mode features
  - Autonomous LIN header handling
  - Autonomous LIN response handling
  - Discarding of irrelevant LIN responses using up to 16 ID filters
- UART mode
  - Full-duplex operation
  - Standard non return-to-zero (NRZ) mark/space format
  - Data buffers with 4-byte receive, 4-byte transmit
  - Configurable word length (8-bit or 9-bit words)
  - Error detection and flagging
    - Parity, noise, and framing errors
  - Interrupt driven operation with 4 interrupts sources
  - Separate transmitter and receiver CPU interrupt sources
  - 16-bit programmable baud-rate modulus counter and 16-bit fractional
  - Two receiver wakeup methods

### 1.5.25 Inter-Integrated Circuit ($I^2C$) controller modules

The MPC5645S includes four $I^2C$ modules. Each module features the following:

- Two-wire bi-directional serial bus for on-board communications
- Compatibility with $I^2C$ bus standard
- Multi-master operation
- Software-programmable for one of 256 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt-driven, byte-by-byte data transfer
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus-busy detection

### 1.5.26 System clocks and clock generation modules

The system clock on the MPC5645S can be derived from an external oscillator, an on-chip FMPLL, or the internal 16 MHz oscillator.

The source system clock frequency can be changed via an on-chip programmable clock divider (÷1 to ÷32). An additional programmable peripheral bus clock divider (ratios ÷1 to ÷15) is also available.

The MPC5645S has two on-chip FMPLLs (primary and secondary). Each features the following:

- Input clock frequency from 4 MHz to 16 MHz
- Lock detect circuitry continuously monitors lock status
- Loss Of Clock (LOC) detection for reference and feedback clocks
- On-chip loop filter (for improved electromagnetic interference performance and reduction of number of external components required)
- Support for frequency ramping from PLL

The primary FMPLL module is for use as a system clock source. The secondary FMPLL is available for use as an alternate, modulated or non-modulated clock source to eMIOS modules and as alternate clock to the DCU for pixel clock generation.

The fast external oscillator provides the following features:

- Input frequency range 4–16 MHz
- Square-wave input mode
- Oscillator input mode 3.3 V (5.0 V)
- Automatic level control
- Low power consumption

- PLL reference

The MPC5645S also includes the following oscillators:

- 32 KHz low power external oscillator for slow execution, low power, and RTC
- Dedicated internal 128 kHz RC oscillator for low power mode operation and self wake-up
  - ±10% accuracy across voltage and temperature (after factory trimming)
  - Trimming registers to support improved accuracy with in-application calibration
- Dedicated 16 MHz internal RC oscillator
  - Used as default clock source out of reset
  - Provides a clock for rapid start-up from low power modes
  - Provides a back-up clock in the event of PLL or External Oscillator clock failure
  - Offers an independent clock source for the SWT
  - ±5% accuracy across voltage and temperature (after factory trimming)
  - Trimming registers to support frequency adjustment with in-application calibration

### 1.5.27 Periodic Interrupt Timer (PIT)

The PIT features the following:

- Eight general purpose interrupt timers
- Two dedicated interrupt timers for triggering ADC conversions
- 32-bit counter resolution
- Clocked by system clock frequency

### 1.5.28 Real Time Counter (RTC)

The Real Timer Counter supports wake-up from Low Power modes or Real Time Clock generation

- Configurable resolution for different timeout periods
  - 1 s resolution for >1 hour period
  - 1 ms resolution for 2 second period
- Selectable clock sources from external 32 KHz crystal, external 4–16 MHz crystal, internal 128 kHz RC oscillator or divided internal 16 MHz RC oscillator

### 1.5.29 System Timer Module (STM)

The STM is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel

- Counter can be stopped in debug mode

### 1.5.30 Software Watchdog Timer (SWT)

The SWT features the following:

- Watchdog supporting software activation or enabled out of Reset
- Supports normal or windowed mode
- Watchdog timer value writable once after reset
- Watchdog supports optional halting during low power modes
- Configurable response on timeout: reset, interrupt, or interrupt followed by reset
- Clock source: 128 kHz RC oscillator

### 1.5.31 Stepper Motor Controller (SMC)

The SMC module is a PWM motor controller suitable to drive instruments in a cluster configuration or any other loads requiring a PWM signal. The motor controller has twelve PWM channels associated with two pins each (24 pins in total) driving up to 6 stepper motors.

The SMC module includes the following features:

- 10/11-bit PWM counter
- 11-bit resolution with selectable PWM dithering function
- Left, right, or center aligned PWM
- Output slew rate control
- Output Short Circuit Detection

This module is suited for, but not limited to, driving small stepper and air core motors used in instrumentation applications. This module can be used for other motor control or PWM applications that match the frequency, resolution, and output drive capabilities of the module.

### 1.5.32 Stepper Stall Detect (SSD)

The SSD module provides a circuit to measure and integrate the induced voltage on the non-driven coil of a stepper motor using full steps when the gauge pointer is returning to zero (RTZ).

The SSD module features the following:

- Programmable full step state
- Programmable integration polarity
- Blanking (recirculation) state
- 16-bit integration accumulator register
- 16-bit modulus down counter with interrupt

### 1.5.33 Sound Generator Module (SGM)

The SGM features the following:

- 4-channel audio mixer
- Each channel capable of independent Tone generation or Wave playback
- Individual channel volume control (8-bit resolution)
- Tone Mode:
  - Programmable Tone frequency
  - Programmable amplitude envelope: attack, duration, and decay
  - Programmable number of tone pulses and inter-tone duration
- Wave Mode:
  - One FIFO per channel working in conjunction with eDMA
  - Supports standard audio sampling rates (4 kHz, 8 kHz, 11.025 kHz, 16 kHz, 22.050 kHz, 32 kHz, 44.100 kHz, 48 kHz)
  - Same sample rate applies to all channels
  - 8-bit, 12-bit, 16-bit input data formats
  - Programmable wave duration and inter-wave duration
  - Repeat mode with programmable number of wave playbacks
- SGM Output:
  - 16-bit PWM channel
  - Integrated $I^2S$ master interface for connection to external audio DAC

### 1.5.34 IEEE 1149.1 JTAG controller (JTAGC)

JTAGC features the following:

- Backward compatible to standard JTAG IEEE 1149.1-2001 test access port (TAP) interface
- Support for boundary scan testing

### 1.5.35 Nexus Development Interface (NDI)

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001-2008 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace

Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.

- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins provides visibility when debugging.
- Watchpoint Trigger enablement of Program and/or Data Trace Messaging enhances debug capability.
- Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
- Address Translation Messaging via program correlation messages displays updates to the TLB for use by the debugger in correlating virtual and physical address information.
- Auxiliary interface for higher data input/output.
- Registers for Program Trace, Data Trace, Ownership Trace, and Watchpoint Trigger.
- All features are controllable and configurable via the JTAG port.
- Nexus Auxiliary port is supported on the 416BGA package.

## 1.6 How to use the MPC5645S documents

This section:

- Describes how the MPC5645S documents provide information on the microcontroller
- Makes recommendations on how to use the documents in a system design

### 1.6.1 The MPC5645S document set

The MPC5645S document set comprises:

- This reference manual (provides information on the features of the logical blocks on the device and how they are integrated with each other)
- The device data sheet (specifies the electrical characteristics of the device)
- The device product brief

The following reference documents (available online at www.freescale.com) are also available to support the CPU on this device:

- *Programmer's Reference Manual for Freescale Embedded Processors*
- *e200z4 Power Architecture Core Reference Manual*
- *Variable-Length Encoding (VLE) Programming Environments Manual*

The aforementioned documents describe all of the functional and electrical characteristics of the MPC5645S microcontroller.

Depending on your task, you may need to refer to multiple documents to make design decisions. However, in general the use of the documents can be divided up as follows:

- Use the reference manual (this document) during software development and when allocating functions during system design.

- Use the data sheet when designing hardware and optimizing power consumption.
- Use the CPU reference documents when:
  - Configuring CPU memory, branch and cache optimizations
  - Doing detailed software development in assembly language
  - Debugging complex software interactions

### 1.6.2 Reference manual content

The content in this document focuses on the functionality of the microcontroller rather than its performance. Most chapters describe the functionality of a particular on-chip module, such as a CAN controller or timer. The remaining chapters describe how these modules are integrated into the memory map, how they are powered and clocked, and the pin-out of the device.

In general, when an individual module is enabled for use all of the detail required to configure and operate it is contained in the dedicated chapter. In some cases there are multiple implementations of this module, however, there is only one chapter for each type of module in use. For this reason, the address of registers in each module is normally provided as an offset from a base address which can be found in Chapter 2, "Memory Map". The benefit of this approach is that software developed for a particular module can be easily reused on this device and on other related devices that use the same modules.

The steps to enable a module for use varies but typically these require configuration of the integration features of the microcontroller. The module will normally have to be powered and enabled at system level, then a clock may have to be explicitly chosen and finally if required the input and output connections to the external system must be configured.

The primary integration chapters of the reference manual contain most of the information required to enable the modules. There are special cases where a chapter may describe module functionality and some integration features for convenience — for example, the microcontroller input/output (SIUL) module. Integration and functional content is provided in the manual as shown in Table 1-3.

**Table 1-3. Reference manual integration and functional content**

| Chapter | Integration content | Functional content |
|---|---|---|
| Overview | • The main features on chip<br>• A summary of the functions provided by each module | — |
| Memory Map | How the memory map is allocated, including:<br>• Internal RAM<br>• Flash memory<br>• External memory-mapped resources and the location of the registers used by the peripherals[1] | — |
| Signal Description | How the signals from each of the modules are combined and brought to a particular pin on a package | — |
| Boot Assist Module | CPU boot sequence from reset | Implementation of the boot options if internal flash memory is not used |

**Table 1-3. Reference manual integration and functional content (continued)**

| Chapter | Integration content | Functional content |
|---|---|---|
| Clock Architecture | Clocking architecture of the device (which clock is available for the system and each peripheral) | — |
| eDMA Channel Mux | Source values for module eDMA channels | How to connect a module eDMA channel to the eDMA module |
| Interrupt Controller | Interrupt vector table | Operation of the module |
| Mode Entry Module | Module numbering for control and status | Operation of operating modes |
| System Integration Unit Lite | How input signals are mapped to individual modules including external interrupt pins | Operation of GPIO |
| Voltage regulators and power supplies | Power distribution to the MCU and in particular to different I/O banks | — |
| Wakeup Unit | Allocation of inputs to the Wakeup Unit | Operation of the wakeup feature |

[1] To find the address of a register in a particular module take the start address of the module given in the memory map and add the offset for the register given in the module chapter.

## 1.7 Using the MPC5645S

There are many different approaches to designing a system using the MPC5645S so the guidance in this section is provided as an example of how the documents can be applied in this task.

Familiarity with the MPC5645S modules can help ensure that its features are being optimally used in a system design. Therefore, the current chapter is a good starting point. Further information on the detailed features of a module are provided within the module chapters. These, combined with the current chapter, should provide a good introduction to the functions available on the MCU.

### 1.7.1 Hardware design

The MPC5645S requires that certain pins are connected to particular power supplies, system functions and other voltage levels for operation.

The MPC5645S internal logic operates from 1.2 V (nominal) supplies that are normally supplied by the on-chip voltage regulator from a 5 V or 3.3 V supply. The 5 V and 3.3 V supplies are also used to supply the input/output pins on the MCU. This means that different input/output ports can operate at different voltages simultaneously. Chapter 3, “Signal Description,” describes the power supply pin names, numbers and their purpose. For more detail on the voltage supply of each pin, see Chapter 48, “Voltage Regulators and Power Supplies”; that chapter also describes the use of the required external ballast transistor to generate the 1.2 V. For specifications of the voltage ranges and limits and decoupling of the power supplies see the MPC5645S data sheet.

Certain pins have dedicated functions that affect the behavior of the MCU after reset. These include pins to force test or alternate boot conditions and debug features. These are described in Chapter 3, “Signal Description,” and a hardware designer should take care that these pins are connected to allow correct operation.

Beyond power supply and pins that have special functions there are also pins that have special system purposes such as oscillator and reset pins. These are also described in Chapter 3, “Signal Description”. The reset pin is bidirectional and its function is closely tied to the reset generation module [Chapter 37, “Reset Generation Module (MC_RGM)”]. The crystal oscillator pins are dedicated to this function but the oscillator is not started automatically after reset. The oscillator module is described in Section 8.4.1, “Pierce oscillator (FXOSC),” along with the internal clock architecture and the other oscillator sources on chip.

### 1.7.2 Input/output pins

The majority of the pins on the MCU are input/output pins which may either operate as general purpose pins or be connected to a particular on-chip module. The arrangement allows a function to be available on several pins. The system designer should allocate the function for the pin before connecting to external hardware. The software should then choose the correct function to match the hardware. The pad characteristics can vary depending on the functions on the pad. Chapter 3, “Signal Description,” describes each pad type (for example, SLOW, MEDIUM, or SMD). Two pads may be able to carry the same function but have different pad types. The electrical specification of the pads is described in the data sheet dependent on the function enabled and the pad type.

In addition to general purpose input/output pins the MPC5645S also includes dedicated pins that have a single function and connect to an external DRAM device.

There are five modules that configure the various functions available:

- System Integration Unit Lite (SIUL)
- Wakeup Unit (WKPU)
- Timing controller (TCON)
- 32 KHz oscillator (SXOSC)
- DRAM controller

The SIUL configures the digital pin functions. Each pin has a register (PCR) in the module that allows selection of the output functions that is connected to the pin. The available settings for the PCR are described in Section 3.3.9, “Functional ports”. Inputs are selected using the PSMI registers; these are described in Chapter 43, “System Integration Unit Lite (SIUL)”. (PSMI registers connect a module to one of several pins, whereas the PCR registers connect a pin to one of several modules).

The DRAM controller pins have a single function and use the related PCRs to configure the correct slew rate control for a connected memory.

The WKPU provides the ability to cause interrupts and wake the MCU from low power modes and operates independently from the SIUL.

In addition to digital I/O functions there are "special functions" that provide analog functionality. These are listed in Section 3.3.9, “Functional ports”. The special functions are enabled independently from the digital I/O which means that the digital function on the pin must be disabled when the special function is active. The TCON module and the SXOSC oscillator are enabled in the modules. The ADC functions are enabled using the PCRs. The RSDS function of the TCON module has a separate set of PCRs for configuration of the pin when that special function is enabled.

### 1.7.3 Software design

Certain modules provide system integration functions, and other modules (such as timers and CAN modules) provide specific functions.

From reset, the modules involved in configuring the system for application software are:

- Boot Assist Module (BAM) — determines the selected boot source
- Reset Generation Module (MC_RGM) — determines the behavior of the MCU when various reset sources are triggered and reports the source of the reset
- Mode Entry Module (MC_ME) — controls which operating mode the MCU is in and configures the peripherals and clocks and power supplies for each of the modes
- Power Control Unit (MC_PCU) — determines which power domains (see Chapter 48, “Voltage Regulators and Power Supplies”) are active
- Clock Generation Module (MC_CGM) — chooses the clock source for the system and many peripherals

After reset, the MCU will automatically select the appropriate reset source and begin to execute code. At this point the system clock is the 16 MHz FIRC oscillator, the CPU is in supervisor mode and the Memory Management Unit (MMU) makes available a small area of memory at the selected reset vector. Initialization is required before most memory and peripherals may be used and before the SRAM can be read (since it is protected by ECC the syndrome will generally be uninitialized after reset and reads would fail the check). Accessing disabled features causes error conditions or interrupts.

A typical startup routine would involve initializing the software environment starting with the MMU configuration and including stacks, heaps, variable initialization and so on and configuring the MCU for the application.

The MMU translates physical memory addresses for use by the CPU and it must be configured before any peripherals or memories are available for use by the CPU. See the *e200z4 Power Architecture Core Reference Manual* for details on how to configure the MMU.

The MC_ME module enables the modules and other features like clocks. It is therefore an essential part of the initialization and operation software. In general, the software will configure an MC_ME mode to make certain peripherals, clocks, and memory active and then switch to that mode.

Chapter 8, “Clock Description,” includes a graphic of the clock architecture of the MCU. This can be used to determine how to configure the MC_CGM module. In general software will configure the module to enable the required clocks and PLLs and route these to the active modules.

After these steps are complete it is possible to configure the input/output pins and the modules for the application.

### 1.7.4 Other features

The MC_ME module manages low power modes and so it is likely that it will be used to switch into different configurations (module sets, clocks) depending on the application requirements.

The MCU includes two other features (both described in Chapter 4, “Safety”) to improve the integrity of the application:

- It is possible to enable a software watchdog (SWT) immediately at reset or afterwards to help detect code runaway.
- Individual register settings can be protected from unintended writes using the features of the Register Protection module. The protected registers are shown in Appendix A, “Registers Under Protection”.

Other integration functionality is provided by the System Status and Configuration Module (SSCM).

# Chapter 2
# Memory Map

Table 2-1 shows the system memory map for the MPC5645S. All addresses on the MPC5645S, including those that are reserved, are identified in the table. The addresses represent the physical addresses assigned to each IP block.

**Table 2-1. MPC5645S system memory map**

| Start address | End address | Size (KB) | Region |
|---|---|---|---|
| On-chip flash memories (Code Flash)<br>Code Flash 0: FL-2048_4_0 (8×16k, 2×64k, 2×128k, 6×256k) | | | |
| 0x00000000 | 0x00003FFF | 16 | Code Flash Array 0 |
| 0x00004000 | 0x00007FFF | 16 | Code Flash Array 0 |
| 0x00008000 | 0x0000BFFF | 16 | Code Flash Array 0 |
| 0x000C000 | 0x0000FFFF | 16 | Code Flash Array 0 |
| 0x00010000 | 0x00013FFF | 16 | Code Flash Array 0 |
| 0x00014000 | 0x00017FFF | 16 | Code Flash Array 0 |
| 0x00018000 | 0x0001BFFF | 16 | Code Flash Array 0 |
| 0x001C000 | 0x0001FFFF | 16 | Code Flash Array 0 |
| 0x00020000 | 0x0002FFFF | 64 | Code Flash Array 0 |
| 0x00030000 | 0x0003FFFF | 64 | Code Flash Array 0 |
| 0x00040000 | 0x0005FFFF | 128 | Code Flash Array 0 |
| 0x00060000 | 0x0007FFFF | 128 | Code Flash Array 0 |
| 0x00080000 | 0x000BFFFF | 256 | Code Flash Array 0 |
| 0x000C0000 | 0x000FFFFF | 256 | Code Flash Array 0 |
| 0x00100000 | 0x0013FFFF | 256 | Code Flash Array 0 |
| 0x00140000 | 0x0017FFFF | 256 | Code Flash Array 0 |
| 0x00180000 | 0x001BFFFF | 256 | Code Flash Array 0 |
| 0x001C0000 | 0x001FFFFF | 256 | Code Flash Array 0 |
| 0x00200000 | 0x007FFFFF | Reserved | |
| On-chip flash memories (shadow for code flash) | | | |
| 0x00F00000 | 0x00FFBFFF | Reserved | |
| 0x00FFC000 | 0x00FFFFFF | 16 | Code Flash Array 0 Shadow Sector |
| Emulation mapping | | | |
| 0x01000000 | 0x1FFFFFFF | 507904 | Flash memory emulation mapping |
| 0x20000000 | 0x3FFFFFFF | 524288 | DRAM (lower 512 MB) |
| SRAM | | | |

**Table 2-1. MPC5645S system memory map (continued)**

| Start address | End address | Size (KB) | Region |
|---|---|---|---|
| 0x40000000 | 0x40001FFF | 8 | SRAM (Power Domain PD0 — Always ON) |
| 0x40002000 | 0x4000FFFF | 56 | SRAM (RAM Domain PD2) |
| 0x40010000 | 0x5FFFFFFF | Reserved | |
| 0x60000000 | 0x600FFFFF | 1024 | 1 MB graphics SRAM |
| 0x60100000 | 0x6FFFFFFF | Reserved | |
| External, Memory Mapped Serial Flash for QuadSPI | | | |
| 0x70000000 | 0x77FFFFFF | 131072 | External Serial Flash Memory A |
| 0x78000000 | 0x7FFFFFFF | 131072 | External Serial Flash Memory B |
| 0x80000000 | 0x8FFFFFFF | 262144 | External Serial Flash Memory A // B |
| 0x90000000 | 0x900001FF | 0.5 | QuadSPI RX Buffer |
| 0x90000200 | 0x90003FFF | Reserved | |
| 0x90004000 | 0x900041FF | 0.5 | RLE Decoder |
| 0x90004200 | 0x9FFFFFFF | Reserved | |
| 0xA0000000 | 0xBFFFFFFF | 524288 | DRAM (upper 512 MB) |
| PBRIDGE(1) - Off Platform Peripherals (mirrored to PBRIDGE(0) memory range 0xFFE80000-0xFFEFFFFF) | | | |
| 0xC0000000 | 0xC3F87FFF | Reserved | |
| 0xC3F88000 | 0xC3F8BFFF | 16 | Code Flash 0 Configuration (CFLASH0) |
| 0xC3F8C000 | 0xC3F8FFFF | Reserved | |
| 0xC3F90000 | 0xC3F93FFF | 16 | System Integration Unit Lite (SIUL) |
| 0xC3F94000 | 0xC3F97FFF | 16 | Wakeup Unit (WKPU) |
| 0xC3F98000 | 0xC3F9BFFF | Reserved | |
| 0xC3F9C000 | 0xC3F9FFFF | Reserved | |
| 0xC3FA0000 | 0xC3FA3FFF | 16 | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 0xC3FA4000 | 0xC3FA7FFF | 16 | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 0xC3FA8000 | 0xC3FD7FFF | Reserved | |
| 0xC3FD8000 | 0xC3FDBFFF | 16 | System Status and Configuration Module (SSCM) |
| 0xC3FDC000 | 0xC3FDFFFF | 16 | Mode Entry Module (MC_ME) |
| 0xC3FE0000 | 0xC3FE3FFF | 16 | Clock Generation Module (MC_CGM) |
| 0xC3FE4000 | 0xC3FE7FFF | 16 | Reset Generation Module (MC_RGM) |
| 0xC3FE8000 | 0xC3FEBFFF | 16 | Power Control Unit (MC_PCU) |
| 0xC3FEC000 | 0xC3FEFFFF | 16 | Real Time Counter & Correction Timer |
| 0xC3FF0000 | 0xC3FF3FFF | 16 | Periodic Interrupt Timer (PIT/RTI) |
| 0xC3FF4000 | 0xC3FFFFFF | Reserved | |

**Table 2-1. MPC5645S system memory map (continued)**

| Start address | End address | Size (KB) | Region |
|---|---|---|---|
| PBRIDGE(0) - Off Platform Peripherals (new range) | | | |
| 0xFFE00000 | 0xFFE03FFF | 16 | Analog-to-Digital Converter 0 (ADC0) |
| 0xFFE04000 | 0xFFE13FFF | Reserved | |
| 0xFFE14000 | 0xFFE17FFF | 16 | RLE Decoder |
| 0xFFE18000 | 0xFFE23FFF | Reserved | |
| 0xFFE24000 | 0xFFE27FFF | 16 | Video Input Unit (VIU2) |
| 0xFFE28000 | 0xFFE2BFFF | 16 | DRAM Controller (DRAMC) |
| 0xFFE2C000 | 0xFFE2FFFF | Reserved | |
| 0xFFE30000 | 0xFFE33FFF | 16 | Inter-IC Bus Interface Controller 0 (I2C0) |
| 0xFFE34000 | 0xFFE37FFF | 16 | Inter-IC bus interface Controller 1 (I2C1) |
| 0xFFE38000 | 0xFFE3BFFF | 16 | Inter-IC bus interface Controller 2 (I2C2) |
| 0xFFE3C000 | 0xFFE3FFFF | 16 | Inter-IC Bus Interface Controller 3 (I2C3) |
| 0xFFE40000 | 0xFFE43FFF | 16 | LINFlex 0 |
| 0xFFE44000 | 0xFFE47FFF | 16 | LINFlex 1 |
| 0xFFE48000 | 0xFFE4BFFF | 16 | LINFlex 2 |
| 0xFFE4C000 | 0xFFE4FFFF | 16 | LINFlex 3 |
| 0xFFE50000 | 0xFFE53FFF | 16 | OpenVG graphics accelerator (GFX2D) |
| 0xFFE54000 | 0xFFE57FFF | 16 | Graphics Accelerator Gasket (GXG) |
| 0xFFE58000 | 0xFFE5BFFF | 16 | Display Control Unit Lite (DCULite) |
| 0xFFE5C000 | 0xFFE5FFFF | 16 | Display Control Unit (DCU3) |
| 0xFFE60000 | 0xFFE60FFF | 4 | Stepper Motor Controller (SMC) |
| 0xFFE61000 | 0xFFE617FF | 2 | Stepper Stall Detect 0 (SSD0) |
| 0xFFE61800 | 0xFFE61FFF | 2 | Stepper Stall Detect 1 (SSD1) |
| 0xFFE62000 | 0xFFE627FF | 2 | Stepper Stall Detect 2 (SSD2) |
| 0xFFE62800 | 0xFFE62FFF | 2 | Stepper Stall Detect 3 (SSD3) |
| 0xFFE63000 | 0xFFE637FF | 2 | Stepper Stall Detect 4 (SSD4) |
| 0xFFE63800 | 0xFFE63FFF | 2 | Stepper Stall Detect 5 (SSD5) |
| 0xFFE64000 | 0xFFE6FFFF | Reserved | |
| 0xFFE70000 | 0xFFE73FFF | 16 | CAN sampler |
| 0xFFE74000 | 0xFFE77FFF | Reserved | |
| 0xFFE78000 | 0xFFE7BFFF | 16 | Sound Generator Module (SGM) |
| 0xFFE7C000 | 0xFFE7FFFF | 16 | Timing Controller (TCON) |
| 0xFFE80000 | 0xFFEFFFFF | Reserved | |

**Table 2-1. MPC5645S system memory map (continued)**

| Start address | End address | Size (KB) | Region |
|---|---|---|---|
| PBRIDGE(0) - On Platform Peripherals | | | |
| 0xFFF00000 | 0xFFF03FFF | 16 | PBRIDGE0 |
| 0xFFF04000 | 0xFFF07FFF | 16 | XBAR |
| 0xFFF08000 | 0xFFF0BFFF | Reserved | |
| 0xFFF0C000 | 0xFFF0FFFF | 16 | Graphics RAM Controller |
| 0xFFF10000 | 0xFFF13FFF | 16 | Memory Protection Unit (MPU) |
| 0xFFF14000 | 0xFFF37FFF | Reserved | |
| 0xFFF38000 | 0xFFF3BFFF | 16 | Software Watchdog Timer 0 (SWT0) |
| 0xFFF3C000 | 0xFFF3FFFF | 16 | System Timer Module 0 (STM0) |
| 0xFFF40000 | 0xFFF43FFF | 16 | Error Correction Status Module (ECSM) |
| 0xFFF44000 | 0xFFF47FFF | 16 | 1st Direct Memory Access Controller (eDMA) |
| 0xFFF48000 | 0xFFF4BFFF | 16 | Interrupt Controller (INTC) |
| 0xFFF4C000 | 0xFFF7FFFF | Reserved | |
| PBRIDGE(0) - Off Platform Peripherals | | | |
| 0xFFF80000 | 0xFFF8FFFF | Reserved | |
| 0xFFF90000 | 0xFFF93FFF | 16 | DSPI 0 |
| 0xFFF94000 | 0xFFF97FFF | 16 | DSPI 1 |
| 0xFFF98000 | 0xFFF9BFFF | 16 | DSPI 2 |
| 0xFFF9C000 | 0xFFF9FFFF | Reserved | |
| 0xFFFA0000 | 0xFFFA3FFF | 16 | QuadSPI0 |
| 0xFFFA4000 | 0xFFFBFFFF | Reserved | |
| 0xFFFC0000 | 0xFFFC3FFF | 16 | FlexCAN 0 (CAN0) |
| 0xFFFC4000 | 0xFFFC7FFF | 16 | FlexCAN 1 (CAN1) |
| 0xFFFC8000 | 0xFFFCBFFF | 16 | FlexCAN 2 (CAN2) |
| 0xFFFCC000 | 0xFFFDBFFF | Reserved | |
| 0xFFFDC000 | 0xFFFDFFFF | 16 | DMA Channel Mux (DMA_MUX) |
| 0xFFFE0000 | 0xFFFFBFFF | Reserved | |
| 0xFFFFC000 | 0xFFFFFFFF | 16 | Boot Assist Module (BAM) |

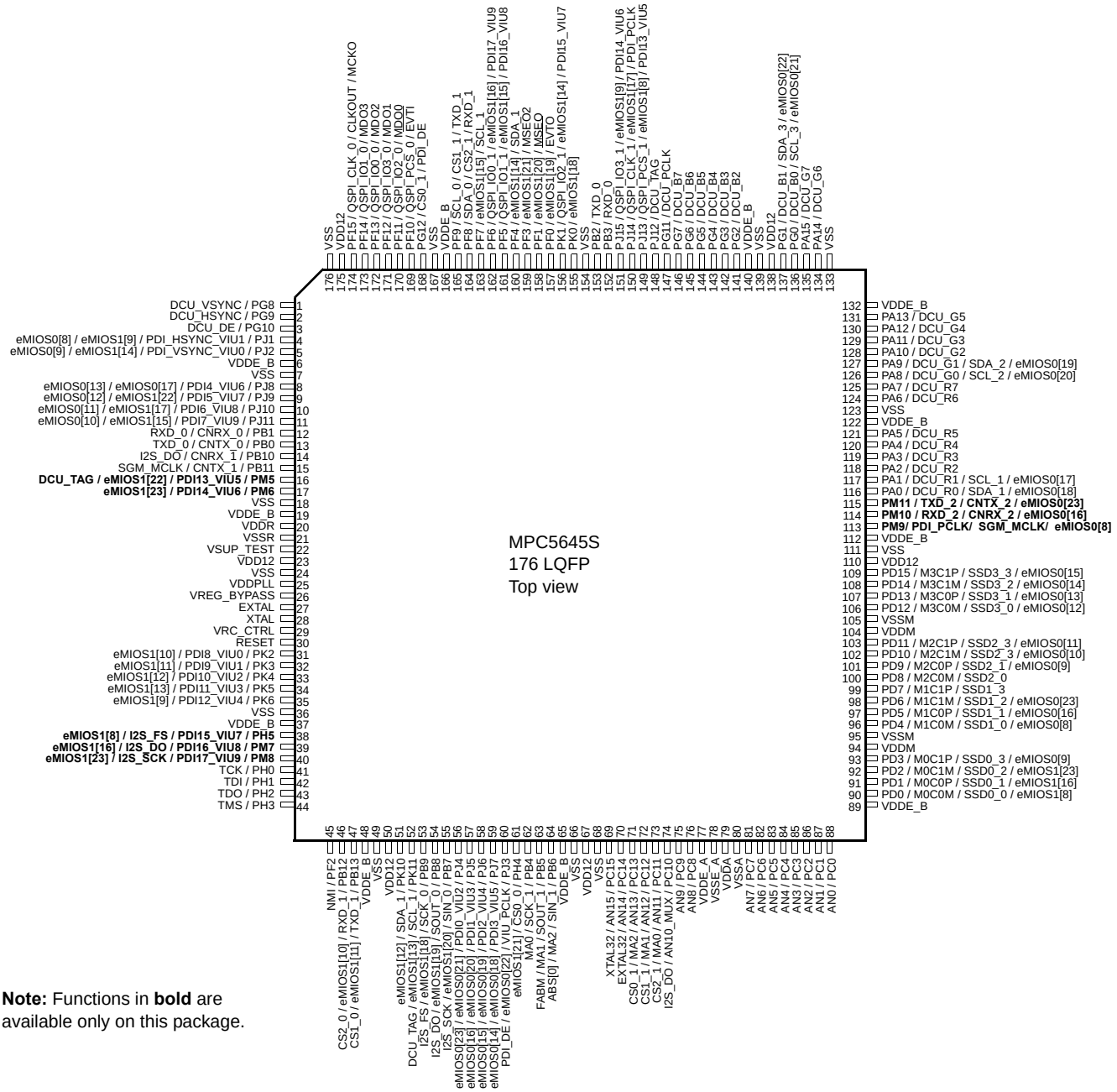
# Chapter 3
# Signal Description

## 3.1 Introduction

The following sections provide signal descriptions and related information about the functionality and configuration.

## 3.2 Package pinouts

The 176- and 208-pin LQFP pinouts and the 416 TEPBGA ballmap are provided in the following figures.




**Note:** Functions in **bold** are available only on this package.

**Figure 3-1. 176-pin LQFP pinout**

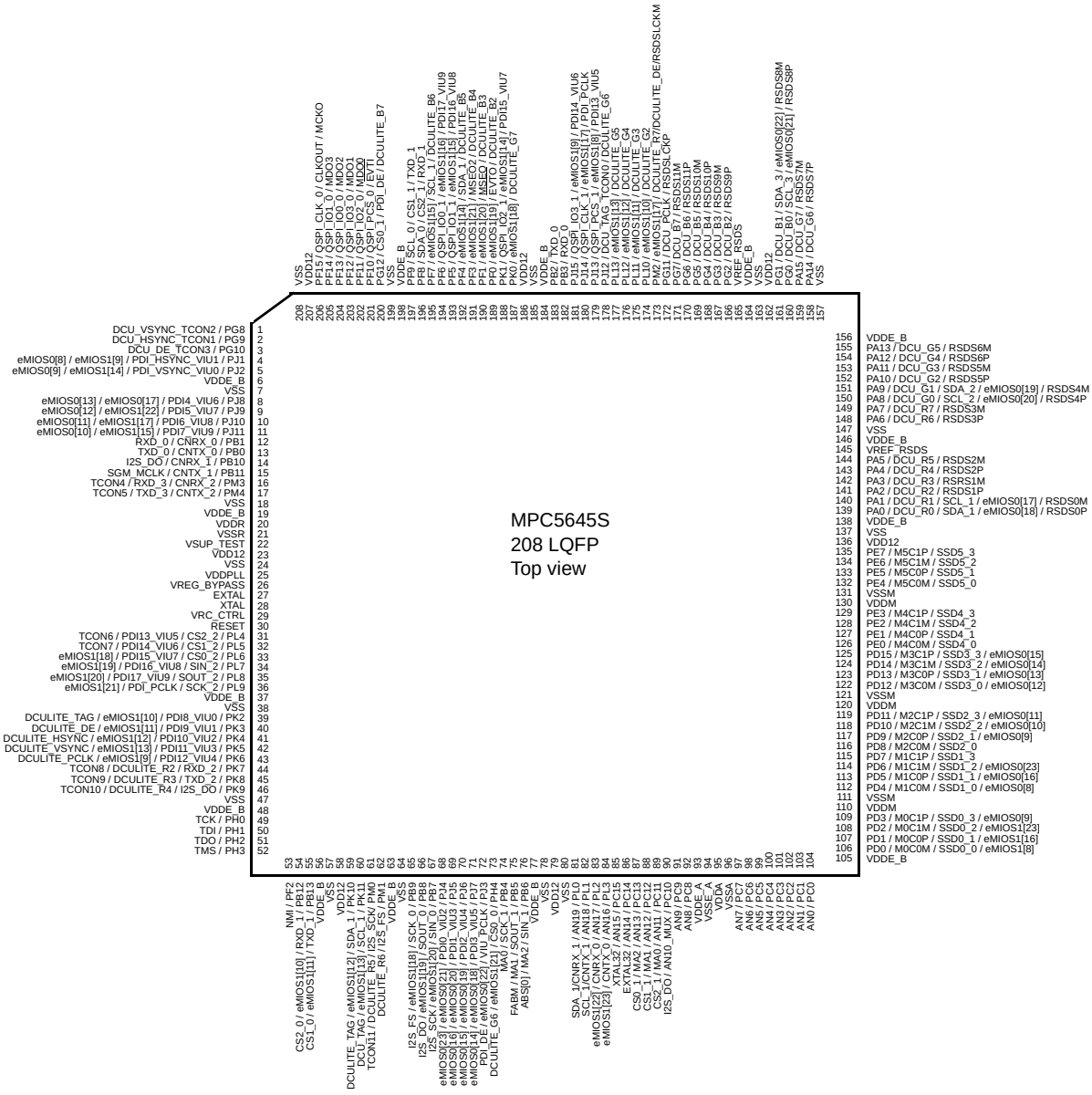
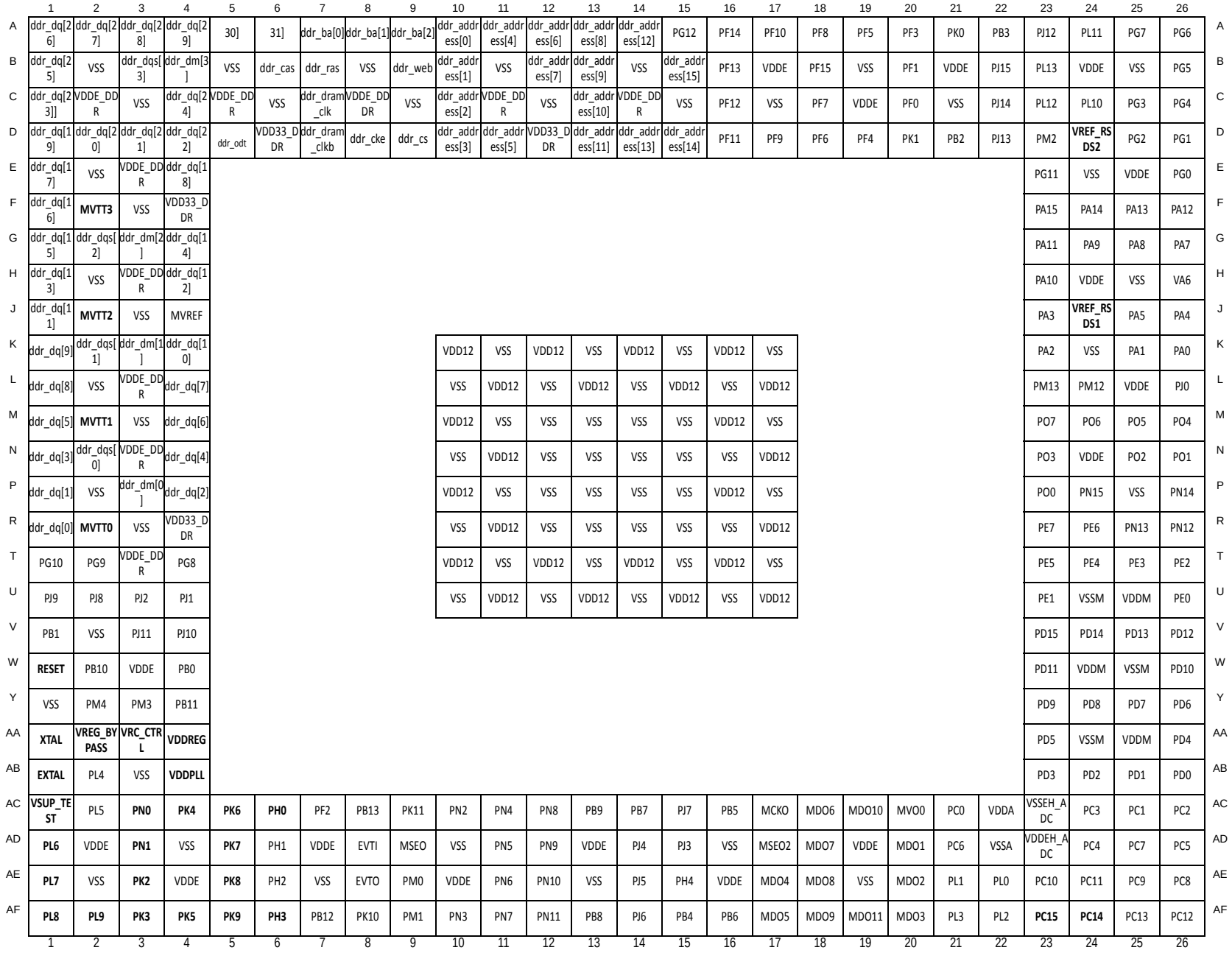



**Figure 3-2. 208-pin LQFP pinout**



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | ddr_dq[26] | ddr_dq[27] | ddr_dq[28] | ddr_dq[29] | 30] | 31] | ddr_ba[0] | ddr_ba[1] | ddr_ba[2] | ddr_address[0] | ddr_address[4] | ddr_address[6] | ddr_address[8] | ddr_address[12] | PG12 | PF14 | PF10 | PF8 | PF5 | PF3 | PK0 | PB3 | PJ12 | PL11 | PG7 | PG6 | A |
| B | ddr_dq[25] | VSS | ddr_dqs[3] | ddr_dm[3] | VSS | ddr_cas | ddr_ras | VSS | ddr_web | ddr_address[1] | VSS | ddr_address[7] | ddr_address[9] | VSS | ddr_address[15] | PF13 | VDDE | PF15 | VSS | PF1 | VDDE | PJ15 | PL13 | VDDE | VSS | PG5 | B |
| C | ddr_dq[23]] | VDDE_DDR | VSS | ddr_dq[24] | VDDE_DDR | VSS | ddr_dram_clk | VDDE_DDR | VSS | ddr_address[2] | VDDE_DDR | VSS | ddr_address[10] | VDDE_DDR | VSS | PF12 | VSS | PF7 | VDDE | PF0 | VSS | PJ14 | PL12 | PL10 | PG3 | PG4 | C |
| D | ddr_dq[19] | ddr_dq[20] | ddr_dq[21] | ddr_dq[22] | ddr_odt | VDD33_DDR | ddr_dram_clkb | ddr_cke | ddr_cs | ddr_address[3] | ddr_address[5] | VDD33_DDR | ddr_address[11] | ddr_address[13] | ddr_address[14] | PF11 | PF9 | PF6 | PF4 | PK1 | PB2 | PJ13 | PM2 | **VREF_RSDS2** | PG2 | PG1 | D |
| E | ddr_dq[17] | VSS | VDDE_DDR | ddr_dq[18] | | | | | | | | | | | | | | | | | | | PG11 | VSS | VDDE | PG0 | E |
| F | ddr_dq[16] | **MVTT3** | VSS | VDD33_DDR | | | | | | | | | | | | | | | | | | | PA15 | PA14 | PA13 | PA12 | F |
| G | ddr_dq[15] | ddr_dqs[2] | ddr_dm[2] | ddr_dq[14] | | | | | | | | | | | | | | | | | | | PA11 | PA9 | PA8 | PA7 | G |
| H | ddr_dq[13] | VSS | VDDE_DDR | ddr_dq[12] | | | | | | | | | | | | | | | | | | | PA10 | VDDE | VSS | VA6 | H |
| J | ddr_dq[11] | **MVTT2** | VSS | MVREF | | | | | | | | | | | | | | | | | | | PA3 | **VREF_RSDS1** | PA5 | PA4 | J |
| K | ddr_dq[9] | ddr_dqs[1] | ddr_dm[1] | ddr_dq[10] | | | | | | VDD12 | VSS | VDD12 | VSS | VDD12 | VSS | VDD12 | VSS | | | | | | PA2 | VSS | PA1 | PA0 | K |
| L | ddr_dq[8] | VSS | VDDE_DDR | ddr_dq[7] | | | | | | VSS | VDD12 | VSS | VDD12 | VSS | VDD12 | VSS | VDD12 | | | | | | PM13 | PM12 | VDDE | PJ0 | L |
| M | ddr_dq[5] | **MVTT1** | VSS | ddr_dq[6] | | | | | | VDD12 | VSS | VSS | VSS | VSS | VSS | VDD12 | VSS | | | | | | PO7 | PO6 | PO5 | PO4 | M |
| N | ddr_dq[3] | ddr_dqs[0] | VDDE_DDR | ddr_dq[4] | | | | | | VSS | VDD12 | VSS | VSS | VSS | VSS | VSS | VDD12 | | | | | | PO3 | VDDE | PO2 | PO1 | N |
| P | ddr_dq[1] | VSS | ddr_dm[0] | ddr_dq[2] | | | | | | VDD12 | VSS | VSS | VSS | VSS | VSS | VDD12 | VSS | | | | | | PO0 | PN15 | VSS | PN14 | P |
| R | ddr_dq[0] | **MVTT0** | VSS | VDD33_DDR | | | | | | VSS | VDD12 | VSS | VSS | VSS | VSS | VSS | VDD12 | | | | | | PE7 | PE6 | PN13 | PN12 | R |
| T | PG10 | PG9 | VDDE_DDR | PG8 | | | | | | VDD12 | VSS | VDD12 | VSS | VDD12 | VSS | VDD12 | VSS | | | | | | PE5 | PE4 | PE3 | PE2 | T |
| U | PJ9 | PJ8 | PJ2 | PJ1 | | | | | | VSS | VDD12 | VSS | VDD12 | VSS | VDD12 | VSS | VDD12 | | | | | | PE1 | VSSM | VDDM | PE0 | U |
| V | PB1 | VSS | PJ11 | PJ10 | | | | | | | | | | | | | | | | | | | PD15 | PD14 | PD13 | PD12 | V |
| W | **RESET** | PB10 | VDDE | PB0 | | | | | | | | | | | | | | | | | | | PD11 | VDDM | VSSM | PD10 | W |
| Y | VSS | PM4 | PM3 | PB11 | | | | | | | | | | | | | | | | | | | PD9 | PD8 | PD7 | PD6 | Y |
| AA | **XTAL** | **VREG_BYPASS** | **VRC_CTRL** | **VDDREG** | | | | | | | | | | | | | | | | | | | PD5 | VSSM | VDDM | PD4 | AA |
| AB | **EXTAL** | PL4 | VSS | **VDDPLL** | | | | | | | | | | | | | | | | | | | PD3 | PD2 | PD1 | PD0 | AB |
| AC | **VSUP_TEST** | PL5 | **PN0** | **PK4** | **PK6** | **PH0** | PF2 | PB13 | PK11 | PN2 | PN4 | PN8 | PB9 | PB7 | PJ7 | PB5 | MCKO | MDO6 | MDO10 | MVO0 | PC0 | VDDA | VSSEH_ADC | PC3 | PC1 | PC2 | AC |
| AD | **PL6** | VDDE | **PN1** | VSS | **PK7** | PH1 | VDDE | EVTI | MSEO | VSS | PN5 | PN9 | VDDE | PJ4 | PJ3 | VSS | MSEO2 | MDO7 | VDDE | MDO1 | PC6 | VSSA | VDDEH_ADC | PC4 | PC7 | PC5 | AD |
| AE | **PL7** | VSS | **PK2** | VDDE | **PK8** | PH2 | VSS | EVTO | PM0 | VDDE | PN6 | PN10 | VSS | PJ5 | PH4 | VDDE | MDO4 | MDO8 | VSS | MDO2 | PL1 | PL0 | PC10 | PC11 | PC9 | PC8 | AE |
| AF | **PL8** | **PL9** | **PK3** | **PK5** | **PK9** | **PH3** | PB12 | PK10 | PM1 | PN3 | PN7 | PN11 | PB8 | PJ6 | PB4 | PB6 | MDO5 | MDO9 | MDO11 | MDO3 | PL3 | PL2 | **PC15** | **PC14** | PC13 | PC12 | AF |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | |

**Figure 3-3. 416-pin TEPBGA ballmap**

## 3.3 Signal descriptions

The following sections provide signal descriptions and related information about the signals' functionality and configuration.

### 3.3.1 Pad configuration during reset phases

All pads have a fixed configuration under reset.

During the power-up phase, all pads are forced to tristate.

After power-up phase, all pads are floating with the following exceptions:

- PB[5] (FAB) is pull-down. Without external strong pull-up the device starts fetching from flash memory.
- RESET pad is driven low. This is released only after PHASE2 reset completion.
- Fast (4-16 MHz) external oscillator pads (EXTAL, XTAL) are tristate.
- The following pads are pull-up:
  - PB[6]
  - PH[0]
  - PH[1]
  - PH[3]

**NOTE**

TCK pin should have an external pull-down resistor of 4.7 K.

### 3.3.2 Voltage supply pins

Voltage supply pins are used to provide power to the device. Two dedicated pins are used for 1.2 V regulator stabilization.

**Table 3-1. Voltage supply pin descriptions**

| Supply pin | Function | Pin number | | |
|---|---|---|---|---|
| | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| $V_{DD12}$[1] | 1.2 V core supply (1.08 V - 1.32 V) | 23, 50, 67, 110, 138, 175 | 23, 58, 79, 136, 162, 186, 207 | K10,K12,K14,K16,L11,L13,L15,L17,M10,M16,N11,N17,P10,P16,R11,R17,T10,T12,T14,T16,U11,U13,U15,U17 |
| $V_{SS}$ | 1.2 V ground | 7, 18, 36, 49, 66, 68, 111, 123, 133, 139, 154, 167, 176 | 7, 18, 38, 47, 57, 64, 78, 80, 137, 147, 157, 163, 185, 199, 208 | AB3,AD10,AD16,AD4,AE13,AE19,AE2,AE7,B11,B14,B19,B2,B25,B5,B8,C12,C15,C17,C21,C3,C6,C9,E2,E24,F3,H2,H25,J3,K11,K13,K15,K17,K24,L10,L12,L14,L16,L2,M11,M12,M13,M14,M15,M17,M3,N10,N12,N13,N14,N15,N16,P11,P12,P13,P14,P15,P17,P2,P25,R10,R12,R13,R14,R15,R16,R3,T11,T13,T15,T17,U10,U12,U14,U16,V2,Y1 |
| | VDD12 ground and VDDPLL ground (VSSPLL) | 24 | 24 | — |
| $V_{DDE_B}$ | 3.3 V I/O supply. This supply is shared with internal flash, 16 MHz IRC oscillator and 4–16MHz crystal oscillator. | 6, 19, 37, 48, 65, 89, 112, 122, 132, 140, 166 | 6, 19, 37, 48, 56, 63, 77, 105, 138, 146, 156, 164, 184, 198 | AD13,AD19,AD2,AD7,AE10,AE16,AE4,B17,B21,B24,C19,E25,H24,L25,N24,W3 |
| $V_{DDA}$[2] | 3.3 V/5 V reference voltage and analog supply for A/D converter. This supply is shared with the SXOSC. | 79 | 95 | AC22 |
| $V_{SSA}$ | Reference ground and analog ground for A/D converter | 80 | 96 | AD22 |
| $V_{DDR}$ | Voltage regulator VREG supply | 20 | 20 | AA4 |
| $V_{SSR}$ | Voltage regulator ground | 21 | 21 | — |
| $V_{DDE_A}$[2] | 3.3 V/5 V I/O supply. This supply is shared with the SXOSC. | 77 | 93 | AD23 |
| $V_{SSE_A}$ | 3.3 V/5 V I/O supply ground | 78 | 94 | AC23 |
| $V_{DDM}$ | Stepper motor 3.3 V/5 V pad supply. SSD shares this supply. | 94, 104 | 110, 120, 130 | U25,W24,AA25 |
| $V_{SSM}$ | Stepper motor ground | 95, 105 | 111, 121, 131 | U24,W24,AA24 |

**Table 3-1. Voltage supply pin descriptions (continued)**

| Supply pin | Function | Pin number | | |
|---|---|---|---|---|
| | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| $V_{DDPLL}$ | 1.2 V PLL supply | 25 | 25 | AB4 |
| $V_{SUP_TEST}$[3] | 9 V - 12 V flash test analog write signal | 22 | 22 | AC1 |
| $V_{DD_DR}$ | 1.8V, 2.5V, and 3.3V DDR SDRAM supply | — | — | C2,C5,C8,C11,C14, E3,H3,L3,N3,T3 |
| $V_{DD33_DR}$ | Functional supply for SDRAM pads (where available must be >= VDD_DR) | — | — | D6, D12, F4, R4 |

1 Decoupling capacitors must be connected between these pins and the nearest $V_{SS}$ pin.

2 VDDA must be at the same voltage as VDDE_A.

3 This signal needs to be connected to ground during normal operation.

## 3.3.3 System pins

The system pins are listed in Table 3-2.

**Table 3-2. System pin descriptions**

| System pin | Function | I/O direction | Pad type | RESET configuration[1] | Pin number | | |
|---|---|---|---|---|---|---|---|
| | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| $\overline{\text{RESET}}$ | Bidirectional reset with Schmitt-Trigger characteristics and noise filter. | I/O | M | Input, weak pull up | 30 | 30 | W1 |
| EXTAL | Analog input to the oscillator amplifier circuit. Input for the clock generator in bypass mode. | I | X | — | 27 | 27 | AB1 |
| XTAL | Analog output of the oscillator amplifier circuit. Needs to be grounded if oscillator bypass mode is used. | O | X | — | 28 | 28 | AA1 |
| EXTAL32 | Analog input of the 32KHz oscillator amplifier circuit. | O | S | — | 70 | 86 | AF24 |
| XTAL32 | Analog output of the 32 KHz oscillator amplifier circuit. Input for the clock generator in bypass mode. | I | S | — | 69 | 85 | AF23 |

**Table 3-2. System pin descriptions (continued)**

| System pin | Function | I/O direction | Pad type | RESET configuration[1] | Pin number | | |
|---|---|---|---|---|---|---|---|
| | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| NMI | Non-Maskable Interrupt | I/O | S | Input, none | 45 | 53 | AC7 |
| VRC_CTRL | Voltage Regulator external NPN Ballast base control pin | | Analog | — | 29 | 29 | AA3 |
| VREF_RSDS[2] | RSDS interface reference voltage | | Analog | — | — | 145, 165 | J24,D24 |
| VREG_BYPASS[3] | Pin used for factory testing | I | — | — | 26 | 26 | AA2 |

1 Reset configuration is given as I/O direction and pull direction (for example, “Input, pullup”).

2 Although this signal is not a supply for RSDS pads, it needs to be terminated in an external capacitor with a value of 47 pF.

3 VREG_BYPASS should be connected to ground during normal operation.

## 3.3.4 Nexus pins

On the 176 LQFP and the 208 LQFP package options a reduced set of Nexus pins are optionally available, multiplexed with GPIO pins.

On the 416 TEPBGA package option all Nexus pins are dedicated to Nexus only.

**Table 3-3. Nexus pins**

| System pin | Function | Pad type | PCR | Pin number[1] | | |
|---|---|---|---|---|---|---|
| | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| EVTI | Nexus Event In | M | PCR[80] | 169 | 201 | A17 |
| EVTO | Nexus Event Out | M | PCR[70] | 157 | 189 | C20 |
| MCKO | Nexus Msg Clock Out | F | PCR[85] | 174 | 206 | B18 |
| MSEO[0] | Nexus Msg Start/End Out | M | PCR[71] | 158 | 190 | B20 |
| MSEO[2] | Nexus Msg Start/End Out | M | PCR[73] | 159 | 191 | A20 |
| MDO[0] | Nexus Msg Data Out | M | PCR[81] | 170 | 202 | D16 |
| MDO[1] | Nexus Msg Data Out | M | PCR[82] | 171 | 203 | C16 |
| MDO[2] | Nexus Msg Data Out | M | PCR[83] | 172 | 204 | B16 |
| MDO[3] | Nexus Msg Data Out | M | PCR[84] | 173 | 205 | A16 |
| EVTI | Nexus Event In | M | PCR[197] | n/a | n/a | AD8 |
| EVTO | Nexus Event Out | M | PCR[198] | n/a | n/a | AE8 |
| MCKO | Nexus Msg Clock Out | F | PCR[200] | n/a | n/a | AC17 |
| MSEO[0] | Nexus Msg Start/End Out | M | PCR[199] | n/a | n/a | AD9 |
| MSEO[2] | Nexus Msg Start/End Out | M | PCR[201] | n/a | n/a | AD17 |

Table 3-3. Nexus pins (continued)

| System pin | Function | Pad type | PCR | Pin number[1] | | |
|---|---|---|---|---|---|---|
| | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| MDO[0] | Nexus Msg Data Out | M | PCR[185] | n/a | n/a | AC20 |
| MDO[1] | Nexus Msg Data Out | M | PCR[186] | n/a | n/a | AD20 |
| MDO[2] | Nexus Msg Data Out | M | PCR[187] | n/a | n/a | AE20 |
| MDO[3] | Nexus Msg Data Out | M | PCR[188] | n/a | n/a | AF20 |
| MDO[4] | Nexus Msg Data Out | M | PCR[189] | n/a | n/a | AE17 |
| MDO[5] | Nexus Msg Data Out | M | PCR[190] | n/a | n/a | AF17 |
| MDO[6] | Nexus Msg Data Out | M | PCR[191] | n/a | n/a | AC18 |
| MDO[7] | Nexus Msg Data Out | M | PCR[192] | n/a | n/a | AD18 |
| MDO[8] | Nexus Msg Data Out | M | PCR[193] | n/a | n/a | AE18 |
| MDO[9] | Nexus Msg Data Out | M | PCR[194] | n/a | n/a | AF18 |
| MDO[10] | Nexus Msg Data Out | M | PCR[195] | n/a | n/a | AC19 |
| MDO[11] | Nexus Msg Data Out | M | PCR[196] | n/a | n/a | AF19 |

1 On the 176 LQFP and 208 LQFP package options the Nexus pins are multiplexed with other GPIO. On the 416 TEPBGA package, there are additional dedicated Nexus pins.

### 3.3.5 DRAM interface

The DRAM interface pins are listed in Table 3-4.

Table 3-4. DRAM interface pin summary

| Port pin[1] | Function | I/O direction | Pad type | PCR | RESET config[2] | Pin number |
|---|---|---|---|---|---|---|
| | | | | | | 416 TEPBGA |
| **DRAM Data Bus** | | | | | | |
| DDR_DQ[31] | DRAM Data Bus [31] | I/O | DDR | PCR[237] | None, None | A6 |
| DDR_DQ[30] | DRAM Data Bus [30] | I/O | DDR | PCR[238] | None, None | A5 |
| DDR_DQ[29] | DRAM Data Bus [29] | I/O | DDR | PCR[239] | None, None | A4 |
| DDR_DQ[28] | DRAM Data Bus [28] | I/O | DDR | PCR[240] | None, None | A3 |
| DDR_DQ[27] | DRAM Data Bus [27] | I/O | DDR | PCR[241] | None, None | A2 |
| DDR_DQ[26] | DRAM Data Bus [26] | I/O | DDR | PCR[242] | None, None | A1 |
| DDR_DQ[25] | DRAM Data Bus [25] | I/O | DDR | PCR[243] | None, None | B1 |
| DDR_DQ[24] | DRAM Data Bus [24] | I/O | DDR | PCR[244] | None, None | C4 |
| DDR_DQ[23] | DRAM Data Bus [23] | I/O | DDR | PCR[245] | None, None | C1 |
| DDR_DQ[22] | DRAM Data Bus [22] | I/O | DDR | PCR[246] | None, None | D4 |
| DDR_DQ[21] | DRAM Data Bus [21] | I/O | DDR | PCR[247] | None, None | D3 |

**Table 3-4. DRAM interface pin summary (continued)**

| Port pin[1] | Function | I/O direction | Pad type | PCR | RESET config[2] | Pin number |
|---|---|---|---|---|---|---|
| | | | | | | 416 TEPBGA |
| DDR_DQ[20] | DRAM Data Bus [20] | I/O | DDR | PCR[248] | None, None | D2 |
| DDR_DQ[19] | DRAM Data Bus [19] | I/O | DDR | PCR[249] | None, None | D1 |
| DDR_DQ[18] | DRAM Data Bus [18] | I/O | DDR | PCR[250] | None, None | E4 |
| DDR_DQ[17] | DRAM Data Bus [17] | I/O | DDR | PCR[251] | None, None | E1 |
| DDR_DQ[16] | DRAM Data Bus [16] | I/O | DDR | PCR[252] | None, None | F1 |
| DDR_DQ[15] | DRAM Data Bus [15] | I/O | DDR | PCR[253] | None, None | G1 |
| DDR_DQ[14] | DRAM Data Bus [14] | I/O | DDR | PCR[254] | None, None | G4 |
| DDR_DQ[13] | DRAM Data Bus [13] | I/O | DDR | PCR[255] | None, None | H1 |
| DDR_DQ[12] | DRAM Data Bus [12] | I/O | DDR | PCR[256] | None, None | H4 |
| DDR_DQ[11] | DRAM Data Bus [11] | I/O | DDR | PCR[257] | None, None | J1 |
| DDR_DQ[10] | DRAM Data Bus [10] | I/O | DDR | PCR[258] | None, None | K4 |
| DDR_DQ[9] | DRAM Data Bus [9] | I/O | DDR | PCR[259] | None, None | K1 |
| DDR_DQ[8] | DRAM Data Bus [8] | I/O | DDR | PCR[260] | None, None | L1 |
| DDR_DQ[7] | DRAM Data Bus [7] | I/O | DDR | PCR[261] | None, None | L4 |
| DDR_DQ[6] | DRAM Data Bus [6] | I/O | DDR | PCR[262] | None, None | M4 |
| DDR_DQ[5] | DRAM Data Bus [5] | I/O | DDR | PCR[263] | None, None | M1 |
| DDR_DQ[4] | DRAM Data Bus [4] | I/O | DDR | PCR[264] | None, None | N4 |
| DDR_DQ[3] | DRAM Data Bus [3] | I/O | DDR | PCR[265] | None, None | N1 |
| DDR_DQ[2] | DRAM Data Bus [2] | I/O | DDR | PCR[266] | None, None | P4 |
| DDR_DQ[1] | DRAM Data Bus [1] | I/O | DDR | PCR[267] | None, None | P1 |
| DDR_DQ[0] | DRAM Data Bus [0] | I/O | DDR | PCR[268] | None, None | R1 |
| **DRAM Data Strobes** | | | | | | |
| DDR_DQS[3] | DRAM Data Strobe [3] | I/O | DDR | PCR[232] | None, None | B3 |
| DDR_DQS[2] | DRAM Data Strobe [2] | I/O | DDR | PCR[231] | None, None | G2 |
| DDR_DQS[1] | DRAM Data Strobe [1] | I/O | DDR | PCR[230] | None, None | K2 |
| DDR_DQS[0] | DRAM Data Strobe [0] | I/O | DDR | PCR[229] | None, None | N2 |
| **DRAM Data Enables** | | | | | | |
| DDR_DM[3] | DRAM Data Enable [3] | Output | DDR | PCR[236] | Output, None | B4 |
| DDR_DM[2] | DRAM Data Enable [2] | Output | DDR | PCR[235] | Output, None | G3 |
| DDR_DM[1] | DRAM Data Enable [1] | Output | DDR | PCR[234] | Output, None | K3 |

**Table 3-4. DRAM interface pin summary (continued)**

| Port pin[1] | Function | I/O direction | Pad type | PCR | RESET config[2] | Pin number |
|---|---|---|---|---|---|---|
| | | | | | | 416 TEPBGA |
| DDR_DM[0] | DRAM Data Enable [0] | Output | DDR | PCR[233] | Output, None | P3 |
| **DRAM Address** | | | | | | |
| DDR_A[15] | DRAM address [15] | Output | DDR | PCR[217] | Output, None | B15 |
| DDR_A[14] | DRAM address [14] | Output | DDR | PCR[216] | Output, None | D15 |
| DDR_A[13] | DRAM address [13] | Output | DDR | PCR[215] | Output, None | D14 |
| DDR_A[12] | DRAM address [12] | Output | DDR | PCR[214] | Output, None | A14 |
| DDR_A[11] | DRAM address [11] | Output | DDR | PCR[213] | Output, None | D13 |
| DDR_A[10] | DRAM address [10] | Output | DDR | PCR[212] | Output, None | C13 |
| DDR_A[9] | DRAM address [9] | Output | DDR | PCR[211] | Output, None | B13 |
| DDR_A[8] | DRAM address [8] | Output | DDR | PCR[210] | Output, None | A13 |
| DDR_A[7] | DRAM address [7] | Output | DDR | PCR[209] | Output, None | B12 |
| DDR_A[6] | DRAM address [6] | Output | DDR | PCR[208] | Output, None | A12 |
| DDR_A[5] | DRAM address [5] | Output | DDR | PCR[207] | Output, None | D11 |
| DDR_A[4] | DRAM address [4] | Output | DDR | PCR[206] | Output, None | A11 |
| DDR_A[3] | DRAM address [3] | Output | DDR | PCR[205] | Output, None | D10 |
| DDR_A[2] | DRAM address [2] | Output | DDR | PCR[204] | Output, None | C10 |
| DDR_A[1] | DRAM address [1] | Output | DDR | PCR[203] | Output, None | B10 |
| DDR_A[0] | DRAM address [0] | Output | DDR | PCR[202] | Output, None | A10 |
| **DRAM Bank Address** | | | | | | |
| DDR_BA[2] | DRAM Bank Address[2] | Output | DDR | PCR[220] | Output, None | A9 |

Table 3-4. DRAM interface pin summary (continued)

| Port pin[1] | Function | I/O direction | Pad type | PCR | RESET config[2] | Pin number |
|---|---|---|---|---|---|---|
| | | | | | | 416 TEPBGA |
| DDR_BA[1] | DRAM Bank Address[1] | Output | DDR | PCR[219] | Output, None | A8 |
| DDR_BA[0] | DRAM Bank Address[0] | Output | DDR | PCR[218] | Output, None | A7 |
| **DRAM Control** | | | | | | |
| DDR_CAS | $\overline{\text{Column Address Strobe}}$ | Output | DDR | PCR[221] | Output, None | B6 |
| DDR_RAS | $\overline{\text{Row Address Strobe}}$ | Output | DDR | PCR[227] | Output, None | B7 |
| DDR_WEB | $\overline{\text{Write Enable}}$ | Output | DDR | PCR[228] | Output, None | B9 |
| DDR_ODT | DRAM On-die termination | Output | DDR | PCR[226] | Output, Pull Down | D5 |
| DDR_CLK | DRAM Clock | Output | DDR | PCR[225] | Output, None | C7 |
| DDR_CLKB | DRAM Clock bar | Output | DDR | NA | Output, None | D7 |
| DDR_CK | DRAM Clock Enable | Output | DDR | PCR[222] | Output, Pull Down | D8 |
| DDR_CS | DRAM Chip Select | Output | DDR | PCR[223] | Output, None | D9 |
| MVREF | DDR Reference Voltage | Input | — | NA | — | J4 |
| MVTT | DRAM Termination Voltage | Input | — | NA | — | F2,J2,M2,R2 |

1 These port pins are disabled and unpowered on packages where the DRAM interface is not bonded out.

2 Reset configuration is given as I/O direction and pull direction (for example, “Input, pullup”).

### 3.3.6 VIU muxing

The DCU3, DCULite and VIU2 modules share the same pins for input video. It is, however, possible to feed independent video streams to VIU2 and DCU3 (operating in narrow mode). Figure 3-4 explains the pin sharing arrangement.

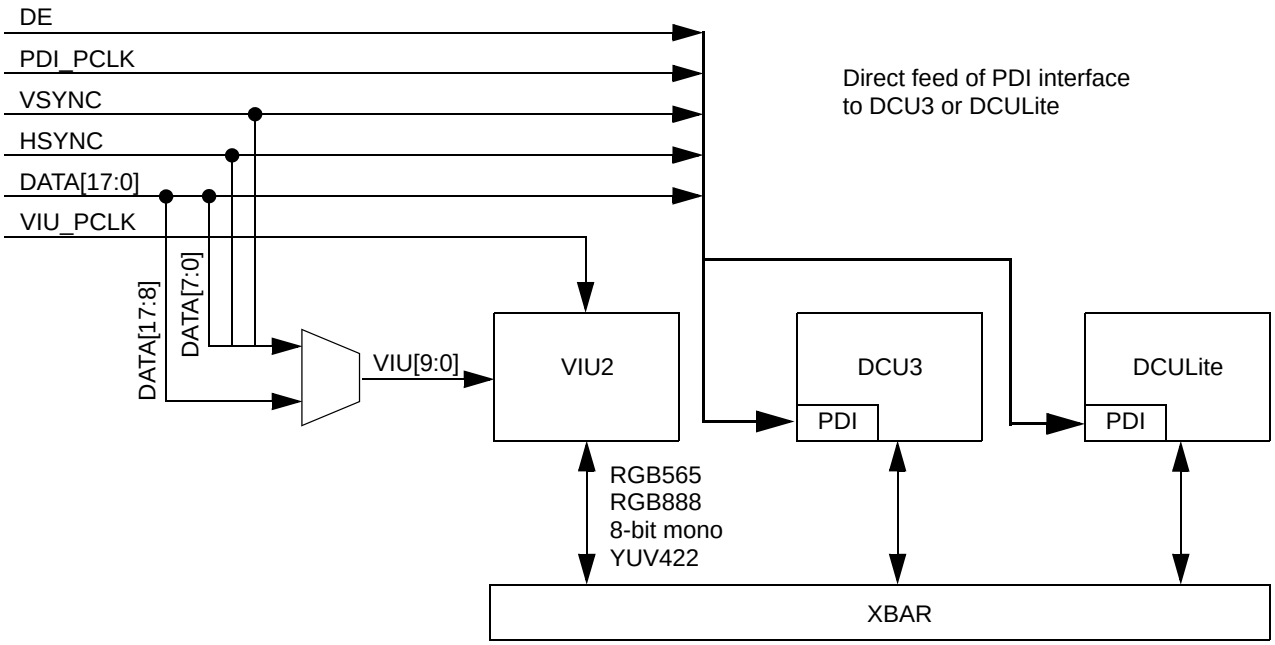



**Figure 3-4. VIU2, DCU3, and DCULite pin sharing**

VIU input data selection is done based on select bit (bit 0) of Miscellaneous control register (0xC3FE0340).

- VIU pix data: VIU[9:0]
- Select bit 1'b0: PDI[7:0],HSYNC,VSYNC
- Select bit 1'b1: PDI[17:8]

### 3.3.7 SGM muxing

The SGM shares pins between the PWM output signals and the I2S bus signals as shown in the "Port pin summary" table. When the PWM function is enabled in the SGM (SGMCTL[PWME]) the PWM (PWMO, PWMOA) signals are available. When the PWM function is disabled the I2S bus signals (I2S_DO, I2S_SCK) are available.

### 3.3.8 RSDS special function muxing

Ports PA[0:15], PG[0:7], PG[11] and PM[2] have the RSDS signalling option as a special function. The SIUL allocates pad control registers to these functions (PCR[270:282]), but because these pads share a common pin with the normal GPIO pins they do not operate in the same way as the normal GPIO ports. PG[11] in particular has a special configuration separate from the other pads.

The special-function pads are output-only, and the associated PCR[OBE] bit is controlled by the TCON_CTRL1 register (TCON_BYPASS and RSDS_MODE bits). However, the alternate function selection is taken from the associated normal GPIO pad. This allows selection of the DCU3 function as the alternate function of the pad and then the TCON module to select if the output style is TCON/RSDS or digital RGB format.

Therefore, when the TCON bypass is active (bypass disabled with or without RSDS active), it is important not to configure the normal GPIO ports for output operation with a non-DCU3 alternate function on ports PA[0:15] and PG[0:7].

For PG[11], the PCR[282] OBE bit is fully controlled by the TCON module and will become an output whenever the DCU3 alternate option is selected. Therefore, only select the DCU3 function on this pin when ready to configure it as a clock for a TFT panel.

## 3.9 Functional ports

The functional port pins are listed in Table 5. Note that most port pins have options to independently configure slew rate control and after reset this will be set to the slowest option. In addition different Pad Types have different slew rate performance and this may mean that the default slew rate needs to be changed to match the performance required for each signal. Refer to column "Pad Type" when selecting and configuring port pins to determine if slew rate adjustment may be required.

**Table 5. Port pin summary**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| **PORT A** | | | | | | | | | | | |
| PA[0] | PCR[0] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[0]<br>DCU_R0<br>SDA_1<br>eMIOS0[18] | RSDS0P | SIUL<br>DCU3<br>$I^2C$_1<br>PWM/Timer | I/O | M / RSDS | None, none | 116 | 139 | K26 |
| PA[1] | PCR[1] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[1]<br>DCU_R1<br>SCL_1<br>eMIOS0[17] | RSDS0M | SIUL<br>DCU3<br>$I^2C$_1<br>PWM/Timer | I/O | M / RSDS | None, none | 117 | 140 | K25 |
| PA[2] | PCR[2] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[2]<br>DCU_R2<br>—<br>— | RSDS1P | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 118 | 141 | K23 |
| PA[3] | PCR[3] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[3]<br>DCU_R3<br>—<br>— | RSDS1M | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 119 | 142 | J23 |
| PA[4] | PCR[4] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[4]<br>DCU_R4<br>—<br>— | RSDS2P | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 120 | 143 | J26 |
| PA[5] | PCR[5] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[5]<br>DCU_R5<br>—<br>— | RSDS2M | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 121 | 144 | J25 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PA[6] | PCR[6] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[6]<br>DCU_R6<br>—<br>— | RSDS3P | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 124 | 148 | H26 |
| PA[7] | PCR[7] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[7]<br>DCU_R7<br>—<br>— | RSDS3M | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 125 | 149 | G26 |
| PA[8] | PCR[8] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[8]<br>DCU_G0<br>SCL_2<br>eMIOS0[20] | RSDS4P | SIUL<br>DCU3<br>$I^2C$_2<br>PWM/Timer | I/O | M / RSDS | None, none | 126 | 150 | G25 |
| PA[9] | PCR[9] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[9]<br>DCU_G1<br>SDA_2<br>eMIOS0[19] | RSDS4M | SIUL<br>DCU3<br>$I^2C$_2<br>PWM/Timer | I/O | M / RSDS | None, none | 127 | 151 | G24 |
| PA[10] | PCR[10] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[10]<br>DCU_G2<br>—<br>— | RSDS5P | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 128 | 152 | H23 |
| PA[11] | PCR[11] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[11]<br>DCU_G3<br>—<br>— | RSDS5M | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 129 | 153 | G23 |
| PA[12] | PCR[12] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[12]<br>DCU_G4<br>—<br>— | RSDS6P | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 130 | 154 | F26 |
| PA[13] | PCR[13] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[13]<br>DCU_G5<br>—<br>— | RSDS6M | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 131 | 155 | F25 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PA[14] | PCR[14] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[14]<br>DCU_G6<br>—<br>— | RSDS7P | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 134 | 158 | F24 |
| PA[15] | PCR[15] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[15]<br>DCU_G7<br>—<br>— | RSDS7M | SIUL<br>DCU3<br>—<br>— | I/O | M / RSDS | None, none | 135 | 159 | F23 |
| **PORT B** | | | | | | | | | | | |
| PB[0] | PCR[16] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[16]<br>CANTX_0<br>TXD_0<br>— | — | SIUL<br>FlexCAN_0<br>LINFlex_0<br>— | I/O | S | None, none | 13 | 13 | W4 |
| PB[1] | PCR[17] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[17]<br>CANRX_0<br>RXD_0<br>— | — | SIUL<br>FlexCAN_0<br>LINFlex_0<br>— | I/O | S | None, none | 12 | 12 | V1 |
| PB[2] | PCR[18] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[18]<br>TXD_0<br>—<br>— | — | SIUL<br>LINFlex_0<br>—<br>— | I/O | S | None, none | 153 | 183 | D21 |
| PB[3] | PCR[19] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[19]<br>RXD_0<br>—<br>— | — | SIUL<br>LINFlex_0<br>—<br>— | I/O | S | None, none | 152 | 182 | A22 |
| PB[4] | PCR[20] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[20]<br>SCK_1<br>MA0<br>— | — | SIUL<br>DSPI_1<br>ADC<br>— | I/O | S | None, none | 62 | 74 | AF15 |
| PB[5] | PCR[21] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[21]<br>SOUT_1<br>MA1<br>FABM | — | SIUL<br>DSPI_1<br>ADC<br>Control | I/O | S | Input, pull-down | 63 | 75 | AC16 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PB[6] | PCR[22] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[22]<br>SIN_1<br>MA2<br>ABS[0] | — | SIUL<br>DSPI_1<br>ADC<br>Control | I/O | S | Input, pull-up | 64 | 76 | AF16 |
| PB[7] | PCR[23] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[23]<br>SIN_0<br>eMIOS1[20]<br>I2S_SCK/PWMO | — | SIUL<br>DSPI_0<br>PWM/Timer<br>SGM | I/O | S | None, none | 55 | 67 | AC14 |
| PB[8] | PCR[24] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[24]<br>SOUT_0<br>eMIOS1[19]<br>I2S_DO/PWMOA | — | SIUL<br>DSPI_0<br>PWM/Timer<br>SGM | I/O | S | None, none | 54 | 66 | AF13 |
| PB[9] | PCR[25] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[25]<br>SCK_0<br>eMIOS1[18]<br>I2S_FS | — | SIUL<br>DSPI_0<br>PWM/Timer<br>SGM | I/O | M | None, none | 53 | 65 | AC13 |
| PB[10] | PCR[26] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[26]<br>CANRX_1<br>I2S_DO/PWMOA<br>— | — | SIUL<br>FlexCAN_1<br>SGM<br>— | I/O | S | None, none | 14 | 14 | W2 |
| PB[11] | PCR[27] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[27]<br>CANTX_1<br>SGM_MCLK<br>— | — | SIUL<br>FlexCAN_1<br>SGM<br>— | I/O | S | None, none | 15 | 15 | Y4 |
| PB[12] | PCR[28] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[28]<br>RXD_1<br>eMIOS1[10]<br>CS2_0 | — | SIUL<br>LINFlex_1<br>PWM/Timer<br>DSPI_0 | I/O | S | None, none | 46 | 54 | AF7 |
| PB[13] | PCR[29] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[29]<br>TXD_1<br>eMIOS1[11]<br>CS1_0 | — | SIUL<br>LINFlex_1<br>PWM/Timer<br>DSPI_0 | I/O | S | None, none | 47 | 55 | AC8 |
| PB[14] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PB[15] | — | — | Reserved | — | — | — | — | — | — | — | — |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| **PORT C** | | | | | | | | | | | |
| PC[0] | PCR[30] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[30]<br>—<br>—<br>— | ANS[0] | SIUL<br>—<br>—<br>— | I/O | J | None, none | 88 | 104 | AC21 |
| PC[1] | PCR[31] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[31]<br>—<br>—<br>— | ANS[1] | SIUL<br>—<br>—<br>— | I/O | J | None, none | 87 | 103 | AC25 |
| PC[2] | PCR[32] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[32]<br>—<br>—<br>— | ANS[2] | SIUL<br>—<br>—<br>— | I/O | J | None, none | 86 | 102 | AC26 |
| PC[3] | PCR[33] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[33]<br>—<br>—<br>— | ANS[3] | SIUL<br>—<br>—<br>— | I/O | J | None, none | 85 | 101 | AC24 |
| PC[4] | PCR[34] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[34]<br>—<br>—<br>— | ANS[4] | SIUL<br>—<br>—<br>— | I/O | J | None, none | 84 | 100 | AD24 |
| PC[5] | PCR[35] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[35]<br>—<br>—<br>— | ANS[5] | SIUL<br>—<br>—<br>— | I/O | J | None, none | 83 | 99 | AD26 |
| PC[6] | PCR[36] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[36]<br>—<br>—<br>— | ANS[6] | SIUL<br>—<br>—<br>— | I/O | J | None, none | 82 | 98 | AD21 |
| PC[7] | PCR[37] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[37]<br>—<br>—<br>— | ANS[7] | SIUL<br>—<br>—<br>— | I/O | J | None, none | 81 | 97 | AD25 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PC[8] | PCR[38] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[38]<br>—<br>—<br>— | ANS[8] | SIUL<br>—<br>—<br>— | I/O | J | None, none | 76 | 92 | AE26 |
| PC[9] | PCR[39] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[39]<br>—<br>—<br>— | ANS[9] | SIUL<br>—<br>—<br>— | I/O | J | None, none | 75 | 91 | AE25 |
| PC[10] | PCR[40] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[40]<br>—<br>I2S_DO/PWMO<br>— | ANS[10] | SIUL<br>—<br>SGM<br>— | I/O | J | None, none | 74 | 90 | AE23 |
| PC[11] | PCR[41] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[41]<br>—<br>MA0<br>CS2_1 | ANS[11] | SIUL<br>—<br>ADC<br>DSPI_1 | I/O | J | None, None | 73 | 89 | AE24 |
| PC[12] | PCR[42] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[42]<br>—<br>MA1<br>CS1_1 | ANS[12] | SIUL<br>—<br>ADC<br>DSPI_1 | I/O | J | None, None | 72 | 88 | AF26 |
| PC[13] | PCR[43] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[43]<br>—<br>MA2<br>CS0_1 | ANS[13] | SIUL<br>—<br>ADC<br>DSPI_1 | I/O | J | None, None | 71 | 87 | AF25 |
| PC[14] | PCR[44] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[44]<br>—<br>—<br>— | ANS[14]<br>EXTAL32 | SIUL<br>—<br>—<br>— | I/O | J | None, None | 70 | 86 | AF24 |
| PC[15] | PCR[45] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[45]<br>—<br>—<br>— | ANS[15]<br>XTAL32 | SIUL<br>—<br>—<br>— | I/O | J | None, None | 69 | 85 | AF23 |
| **PORT D** | | | | | | | | | | | |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PD[0] | PCR[46] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[46]<br>M0C0M<br>SSD0_0<br>eMIOS1[8] | — | SIUL<br>SMD<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 90 | 106 | AB26 |
| PD[1] | PCR[47] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[47]<br>M0C0P<br>SSD0_1<br>eMIOS1[16] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 91 | 107 | AB25 |
| PD[2] | PCR[48] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[48]<br>M0C1M<br>SSD0_2<br>eMIOS1[23] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 92 | 108 | AB24 |
| PD[3] | PCR[49] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[49]<br>M0C1P<br>SSD0_3<br>eMIOS0[9] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 93 | 109 | AB23 |
| PD[4] | PCR[50] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[50]<br>M1C0M<br>SSD1_0<br>eMIOS0[8] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 96 | 112 | AA26 |
| PD[5] | PCR[51] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[51]<br>M1C0P<br>SSD1_1<br>eMIOS0[16] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 97 | 113 | AA23 |
| PD[6] | PCR[52] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[52]<br>M1C1M<br>SSD1_2<br>eMIOS0[23] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 98 | 114 | Y26 |
| PD[7] | PCR[53] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[53]<br>M1C1P<br>SSD1_3<br>— | — | SIUL<br>SMC<br>SSD<br>— | I/O | SMD | None,<br>None | 99 | 115 | Y25 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PD[8] | PCR[54] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[54]<br>M2C0M<br>SSD2_0<br>— | — | SIUL<br>SMC<br>SSD<br>— | I/O | SMD | None,<br>None | 100 | 116 | Y24 |
| PD[9] | PCR[55] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[55]<br>M2C0P<br>SSD2_1<br>eMIOS0[9] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 101 | 117 | Y23 |
| PD[10] | PCR[56] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[56]<br>M2C1M<br>SSD2_2<br>eMIOS0[10] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 102 | 118 | W26 |
| PD[11] | PCR[57] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[57]<br>M2C1P<br>SSD2_3<br>eMIOS0[11] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 103 | 119 | W23 |
| PD[12] | PCR[58] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[58]<br>M3C0M<br>SSD3_0<br>eMIOS0[12] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 106 | 122 | V26 |
| PD[13] | PCR[59] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[59]<br>M3C0P<br>SSD3_1<br>eMIOS0[13] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 107 | 123 | V25 |
| PD[14] | PCR[60] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[60]<br>M3C1M<br>SSD3_2<br>eMIOS0[14] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 108 | 124 | V24 |
| PD[15] | PCR[61] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[61]<br>M3C1P<br>SSD3_3<br>eMIOS0[15] | — | SIUL<br>SMC<br>SSD<br>PWM/Timer | I/O | SMD | None,<br>None | 109 | 125 | V23 |
| **PORT E** | | | | | | | | | | | |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PE[0] | PCR[62] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[62]<br>M4C0M<br>SSD4_0<br>— | — | SIUL<br>SMC<br>SSD<br>— | I/O | SMD | None,<br>None | — | 126 | U26 |
| PE[1] | PCR[63] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[63]<br>M4C0P<br>SSD4_1<br>— | — | SIUL<br>SMC<br>SSD<br>— | I/O | SMD | None,<br>None | — | 127 | U23 |
| PE[2] | PCR[64] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[64]<br>M4C1M<br>SSD4_2<br>— | — | SIUL<br>SMC<br>SSD<br>— | I/O | SMD | None,<br>None | — | 128 | T26 |
| PE[3] | PCR[65] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[65]<br>M4C1P<br>SSD4_3<br>— | — | SIUL<br>SMC<br>SSD<br>— | I/O | SMD | None,<br>None | — | 129 | T25 |
| PE[4] | PCR[66] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[66]<br>M5C0M<br>SSD5_0<br>— | — | SIUL<br>SMC<br>SSD<br>— | I/O | SMD | None,<br>None | — | 132 | T24 |
| PE[5] | PCR[67] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[67]<br>M5C0P<br>SSD5_1<br>— | — | SIUL<br>SMC<br>SSD<br>— | I/O | SMD | None,<br>None | — | 133 | T23 |
| PE[6] | PCR[68] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[68]<br>M5C1M<br>SSD5_2<br>— | — | SIUL<br>SMC<br>SSD<br>— | I/O | SMD | None,<br>None | — | 134 | R24 |
| PE[7] | PCR[69] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[69]<br>M5C1P<br>SSD5_3<br>— | — | SIUL<br>SMC<br>SSD<br>— | I/O | SMD | None,<br>None | — | 135 | R23 |
| **PORT F** | | | | | | | | | | | |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PF[0] | PCR[70] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[70]<br>eMIOS1[19]<br>EVTO<br>DCULITE_B2 | — | SIUL<br>PWM/Timer<br>NEXUS<br>DCULite | I/O | M | None,<br>None | 157 | 189 | C20 |
| PF[1] | PCR[71] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[71]<br>eMIOS1[20]<br>MSEO<br>DCULITE_B3 | — | SIUL<br>PWM/Timer<br>NEXUS<br>DCULite | I/O | M | None,<br>None | 158 | 190 | B20 |
| PF[2] | PCR[72] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[72]<br>NMI<br>—<br>— | — | SIUL<br>NMI<br>—<br>— | I/O | S | None,<br>None | 45 | 53 | AC7 |
| PF[3] | PCR[73] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[73]<br>eMIOS1[21]<br>MSEO<br>DCULITE_B4 | — | SIUL<br>PWM/Timer<br>NEXUS<br>DCULite | I/O | M | None,<br>None | 159 | 191 | A20 |
| PF[4] | PCR[74] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[74]<br>eMIOS1[14]<br>SDA_1<br>DCULITE_B5 | — | SIUL<br>PWM/Timer<br>$I^2C$_1<br>DCULite | I/O | M | None,<br>None | 160 | 192 | D19 |
| PF[5] | PCR[75] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[75]<br>QUADSPI_IO1_B<br>eMIOS1[15]<br>VIU8_PDI16 | — | SIUL<br>QuadSPI<br>PWM/Timer<br>VIU2/PDI | I/O | M | None,<br>None | 161 | 193 | A19 |
| PF[6] | PCR[76] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[76]<br>QUADSPI_IO0_B<br>eMIOS1[16]<br>VIU9_PDI17 | — | SIUL<br>QuadSPI<br>PWM/Timer<br>VIU2/PDI | I/O | M | None,<br>None | 162 | 194 | D18 |
| PF[7] | PCR[77] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[77]<br>eMIOS1[15]<br>SCL_1<br>DCULITE_B6 | — | SIUL<br>PWM/Timer<br>$I^2C$_1<br>DCULite | I/O | M | None,<br>None | 163 | 195 | C18 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PF[8] | PCR[78] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[78]<br>SDA_0<br>CS2_1<br>RXD_1 | — | SIUL<br>I2C_0<br>DSPI_1<br>LINFlex_1 | I/O | S | None,<br>None | 164 | 196 | A18 |
| PF[9] | PCR[79] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[79]<br>SCL_0<br>CS1_1<br>TXD_1 | — | SIUL<br>I2C_0<br>DSPI_1<br>LINFlex_1 | I/O | S | None,<br>None | 165 | 197 | D17 |
| PF[10] | PCR[80] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[80]<br>QUADSPI_PCS_A<br>—<br>EVTI | — | SIUL<br>QuadSPI<br>—<br>NEXUS | I/O | M | None,<br>None | 169 | 201 | A17 |
| PF[11] | PCR[81] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[81]<br>QUADSPI_IO2_A<br>—<br>MDO0 | — | SIUL<br>QuadSPI<br>—<br>NEXUS | I/O | M | None,<br>None | 170 | 202 | D16 |
| PF[12] | PCR[82] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[82]<br>QUADSPI_IO3_A<br>—<br>MDO1 | — | SIUL<br>QuadSPI<br>—<br>NEXUS | I/O | M | None,<br>None | 171 | 203 | C16 |
| PF[13] | PCR[83] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[83]<br>QUADSPI_IO0_A<br>—<br>MDO2 | — | SIUL<br>QuadSPI<br>—<br>NEXUS | I/O | M | None,<br>None | 172 | 204 | B16 |
| PF[14] | PCR[84] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[84]<br>QUADSPI_IO1_A<br>—<br>MDO3 | — | SIUL<br>QuadSPI<br>—<br>NEXUS | I/O | M | None,<br>None | 173 | 205 | A16 |
| PF[15] | PCR[85] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[85]<br>QUADSPI_CLK_A<br>CLKOUT<br>MCKO | — | SIUL<br>QuadSPI<br>Control<br>NEXUS | I/O | F | None,<br>None | 174 | 206 | B18 |
| **PORT G** | | | | | | | | | | | |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PG[0] | PCR[86] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[86]<br>DCU_B0<br>SCL_3<br>eMIOS0[21] | RSDS8P | SIUL<br>DCU3<br>$I^2C$_3<br>PWM/Timer | I/O | M | None,<br>None | 136 | 160 | E26 |
| PG[1] | PCR[87] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[87]<br>DCU_B1<br>SDA_3<br>eMIOS0[22] | RSDS8M | SIUL<br>DCU3<br>$I^2C$_3<br>PWM/Timer | I/O | M | None,<br>None | 137 | 161 | D26 |
| PG[2] | PCR[88] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[88]<br>DCU_B2<br>—<br>— | RSDS9P | SIUL<br>DCU3<br>—<br>— | I/O | M | None,<br>None | 141 | 166 | D25 |
| PG[3] | PCR[89] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[89]<br>DCU_B3<br>—<br>— | RSDS9M | SIUL<br>DCU3<br>—<br>— | I/O | M | None,<br>None | 142 | 167 | C25 |
| PG[4] | PCR[90] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[90]<br>DCU_B4<br>—<br>— | RSDS10P | SIUL<br>DCU3<br>—<br>— | I/O | M | None,<br>None | 143 | 168 | C26 |
| PG[5] | PCR[91] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[91]<br>DCU_B5<br>—<br>— | RSDS10M | SIUL<br>DCU3<br>—<br>— | I/O | M | None,<br>None | 144 | 169 | B26 |
| PG[6] | PCR[92] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[92]<br>DCU_B6<br>—<br>— | RSDS11P | SIUL<br>DCU3<br>—<br>— | I/O | M | None,<br>None | 145 | 170 | A26 |
| PG[7] | PCR[93] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[93]<br>DCU_B7<br>—<br>— | RSDS11M | SIUL<br>DCU3<br>—<br>— | I/O | M | None,<br>None | 146 | 171 | A25 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PG[8] | PCR[94] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[94]<br>DCU_VSYNC<br>—<br>— | — | SIUL<br>DCU3<br>—<br>— | I/O | M | None,<br>None | 1 | 1 | T4 |
| PG[9] | PCR[95] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[95]<br>DCU_HSYNC<br>—<br>— | — | SIUL<br>DCU3<br>—<br>— | I/O | M | None,<br>None | 2 | 2 | T2 |
| PG[10] | PCR[96] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[96]<br>DCU_DE<br>—<br>— | — | SIUL<br>DCU3<br>—<br>— | I/O | M | None,<br>None | 3 | 3 | T1 |
| PG[11] | PCR[97] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[97]<br>DCU_PCLK<br>—<br>— | RSDSCLKP | SIUL<br>DCU3<br>—<br>— | I/O | F | None,<br>None | 147 | 172 | E23 |
| PG[12] | PCR[98] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[98]<br>CS0_1<br>PDI_DE<br>DCULITE_B7 | — | SIUL<br>DSPI_1<br>PDI<br>DCULite | I/O | M | None,<br>None | 168 | 200 | A15 |
| PG[13] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PG[14] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PG[15] | — | — | Reserved | — | — | — | — | — | — | — | — |
| **PORT H** | | | | | | | | | | | |
| PH[0][6] | PCR[99] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[99]<br>TCK<br>—<br>— | — | SIUL<br>JTAG<br>—<br>— | I/O | S | Input,<br>Pull Up | 41 | 49 | AC6 |
| PH[1][6] | PCR[100] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[100]<br>TDI<br>—<br>— | — | SIUL<br>JTAG<br>—<br>— | I/O | S | Input,<br>Pull Up | 42 | 50 | AD6 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PH[2][6] | PCR[101] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[101]<br>TDO<br>—<br>— | — | SIUL<br>JTAG<br>—<br>— | I/O | M | Output, None | 43 | 51 | AE6 |
| PH[3][6] | PCR[102] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[102]<br>TMS<br>—<br>— | — | SIUL<br>JTAG<br>—<br>— | I/O | S | Input, Pull Up | 44 | 52 | AF6 |
| PH[4] | PCR[103] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[103]<br>CS0_0<br>eMIOS1[21]<br>DCULITE_G6 | — | SIUL<br>DSPI_0<br>PWM/Timer<br>DCULite | I/O | M | None, None | 61 | 73 | AE15 |
| PH[5] | PCR[104] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[104]<br>VIU7_PDI15<br>I2S_FS<br>eMIOS1[8] | — | SIUL<br>VIU2/PDI<br>SGM<br>PWM/Timer | I/O | S | None, None | 38 | — | — |
| PH[6] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PH[7] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PH[8] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PH[9] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PH[10] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PH[11] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PH[12] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PH[13] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PH[14] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PH[15] | — | — | Reserved | — | — | — | — | — | — | — | — |
| **PORT J** | | | | | | | | | | | |
| PJ[0] | PCR[105] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[105]<br>DCULITE_B6<br>—<br>I2S_DO / PWMO | — | SIUL<br>DCULite<br>—<br>SGM | I/O | M | None, None | — | — | L26 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PJ[1] | PCR[106] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[106]<br>VIU1_PDI_HSYNC<br>eMIOS1[9]<br>eMIOS0[8] | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>PWM/Timer | I/O | S | None,<br>None | 4 | 4 | U4 |
| PJ[2] | PCR[107] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[107]<br>VIU0_PDI_VSYNC<br>eMIOS1[14]<br>eMIOS0[9] | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>PWM/Timer | I/O | S | None,<br>None | 5 | 5 | U3 |
| PJ[3] | PCR[108] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[108]<br>VIU_PCLK<br>eMIOS0[22]<br>PDI_DE | — | SIUL<br>VIU2<br>PWM/Timer<br>PDI | I/O | S | None,<br>None | 60 | 72 | AD15 |
| PJ[4] | PCR[109] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[109]<br>VIU2_PDI0<br>eMIOS0[21]<br>eMIOS0[23] | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>PWM/Timer | I/O | S | None,<br>None | 56 | 68 | AD14 |
| PJ[5] | PCR[110] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[110]<br>VIU3_PDI1<br>eMIOS0[20]<br>eMIOS0[16] | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>PWM/Timer | I/O | M | None,<br>None | 57 | 69 | AE14 |
| PJ[6] | PCR[111] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[111]<br>VIU4_PDI2<br>eMIOS0[19]<br>eMIOS0[15] | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>PWM/Timer | I/O | S | None,<br>None | 58 | 70 | AF14 |
| PJ[7] | PCR[112] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[112]<br>VIU5_PDI3<br>eMIOS0[18]<br>eMIOS0[14] | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>PWM/Timer | I/O | S | None,<br>None | 59 | 71 | AC15 |
| PJ[8] | PCR[113] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[113]<br>VIU6_PDI4<br>eMIOS0[17]<br>eMIOS0[13] | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>PWM/Timer | I/O | S | None,<br>None | 8 | 8 | U2 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PJ[9] | PCR[114] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[114]<br>VIU7_PDI5<br>eMIOS1[22]<br>eMIOS0[12] | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>PWM/Timer | I/O | S | None,<br>None | 9 | 9 | U1 |
| PJ[10] | PCR[115] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[115]<br>VIU8_PDI6<br>eMIOS1[17]<br>eMIOS0[11] | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>PWM/Timer | I/O | S | None,<br>None | 10 | 10 | V4 |
| PJ[11] | PCR[116] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[116]<br>VIU9_PDI7<br>eMIOS1[15]<br>eMIOS0[10] | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>PWM/Timer | I/O | S | None,<br>None | 11 | 11 | V3 |
| PJ[12] | PCR[117] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[117]<br>DCU_TAG<br>—<br>DCULITE_G6 | — | SIUL<br>DCU3<br>—<br>DCULite | I/O | M | None,<br>None | 148 | 178 | A23 |
| PJ[13] | PCR[118] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[118]<br>QUADSPI_PCS_B<br>eMIOS1[8]<br>VIU5_PDI13 | — | SIUL<br>QuadSPI<br>PWM/Timer<br>VIU2/PDI | I/O | M | None,<br>None | 149 | 179 | D22 |
| PJ[14] | PCR[119] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[119]<br>QUADSPI_CLK_B<br>eMIOS1[17]<br>PDI_PCLK | — | SIUL<br>QuadSPI<br>PWM/Timer<br>PDI | I/O | F | None,<br>None | 150 | 180 | C22 |
| PJ[15] | PCR[120] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[120]<br>QUADSPI_IO3_B<br>eMIOS1[9]<br>VIU6_PDI14 | — | SIUL<br>QuadSPI<br>PWM/Timer<br>VIU2/PDI | I/O | M | None,<br>None | 151 | 181 | B22 |
| **PORT K** | | | | | | | | | | | |
| PK[0] | PCR[121] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[121]<br>eMIOS1[18]]<br>—<br>DCULITE_G7 | — | SIUL<br>PWM/Timer<br>—<br>DCULite | I/O | M | None,<br>None | 155 | 187 | A21 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PK[1] | PCR[122] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[122]<br>QUADSPI_IO2_B<br>eMIOS1[14]<br>VIU7_PDI15 | — | SIUL<br>QuadSPI<br>PWM/Timer<br>VIU2/PDI | I/O | M | None,<br>None | 156 | 188 | D20 |
| PK[2] | PCR[123] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[123]<br>VIU0_PDI8<br>eMIOS1[10]<br>DCULITE_TAG | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>DCULite | I/O | M | None,<br>None | 31 | 39 | AE3 |
| PK[3] | PCR[124] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[124]<br>VIU1_PDI9<br>eMIOS1[11]<br>DCULITE_DE | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>DCULite | I/O | M | None,<br>None | 32 | 40 | AF3 |
| PK[4] | PCR[125] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[125]<br>VIU2_PDI10<br>eMIOS1[12]<br>DCULITE_HSYNC | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>DCULite | I/O | M | None,<br>None | 33 | 41 | AC4 |
| PK[5] | PCR[126] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[126]<br>VIU3_PDI11<br>eMIOS1[13]<br>DCULITE_VSYNC | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>DCULite | I/O | M | None,<br>None | 34 | 42 | AF4 |
| PK[6] | PCR[127] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[127]<br>VIU4_PDI12<br>eMIOS1[9]<br>DCULITE_PCLK | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>DCULite | I/O | F | None,<br>None | 35 | 43 | AC5 |
| PK[7] | PCR[128] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[128]<br>RXD_2<br>DCULITE_R2<br>TCON[8] | — | SIUL<br>LINFlex_2<br>DCULite<br>TCON | I/O | M | None,<br>None | — | 44 | AD5 |
| PK[8] | PCR[129] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[129]<br>TXD_2<br>DCULITE_R3<br>TCON[9] | — | SIUL<br>LINFlex_2<br>DCULite<br>TCON | I/O | M | None,<br>None | — | 45 | AE5 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PK[9] | PCR[130] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[130]<br>I2S_DO / PWMO<br>DCULITE_R4<br>TCON[10] | — | SIUL<br>SGM<br>DCULite<br>TCON | I/O | M | None,<br>None | — | 46 | AF5 |
| PK[10] | PCR[131] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[131]<br>SDA_1<br>eMIOS1[12]<br>DCULITE_TAG | — | SIUL<br>$I^2C$_1<br>PWM/Timer<br>DCULite | I/O | S | None,<br>None | 51 | 59 | AF8 |
| PK[11] | PCR[132] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[132]<br>SCL_1<br>eMIOS1[13]<br>DCU_TAG | — | SIUL<br>$I^2C$_1<br>PWM/Timer<br>DCU3 | I/O | S | None,<br>None | 52 | 60 | AC9 |
| PK[12] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PK[13] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PK[14] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PK[15] | — | — | Reserved | — | — | — | — | — | — | — | — |
| **PORT L** | | | | | | | | | | | |
| PL[0] | PCR[133] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[133]<br>—<br>CANRX_1<br>SDA_1 | ANS[19] | SIUL<br>—<br>FlexCAN_1<br>I2C1 | I/O | M /<br>ANALOG | None,<br>None | — | 81 | AE22 |
| PL[1] | PCR[134] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[134]<br>—<br>CANTX_1<br>SCL_1 | ANS[18] | SIUL<br>—<br>FlexCAN_1<br>I2C1 | I/O | M /<br>ANALOG | None,<br>None | — | 82 | AE21 |
| PL[2] | PCR[135] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[135]<br>—<br>CANRX_0<br>eMIOS1[22] | ANS[17] | SIUL<br>—<br>FlexCAN_0<br>PWM/Timer | I/O | S /<br>ANALOG | None,<br>None | — | 83 | AF22 |
| PL[3] | PCR[136] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[136]<br>—<br>CANTX_0<br>eMIOS1[23] | ANS[16] | SIUL<br>—<br>FlexCAN_0<br>PWM/Timer | I/O | S /<br>ANALOG | None,<br>None | — | 84 | AF21 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PL[4] | PCR[137] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[137]<br>CS2_2<br>VIU5_PDI13<br>TCON[6] | — | SIUL<br>DSPI_2<br>VIU2/PDI<br>TCON | I/O | M | None,<br>None | — | 31 | AB2 |
| PL[5] | PCR[138] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[138]<br>CS1_2<br>VIU6_PDI14<br>TCON[7] | — | SIUL<br>DSPI_2<br>VIU2/PDI<br>TCON | I/O | M | None,<br>None | — | 32 | AC2 |
| PL[6] | PCR[139] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[139]<br>CS0_2<br>VIU7_PDI15<br>eMIOS1[18] | — | SIUL<br>DSPI_2<br>VIU2/PDI<br>PWM/Timer | I/O | S | None,<br>None | — | 33 | AD1 |
| PL[7] | PCR[140] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[140]<br>SIN_2<br>VIU8_PDI16<br>eMIOS1[19] | — | SIUL<br>DSPI_2<br>VIU2/PDI<br>PWM/Timer | I/O | S | None,<br>None | — | 34 | AE1 |
| PL[8] | PCR[141] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[141]<br>SOUT_2<br>VIU9_PDI17<br>eMIOS1[20] | — | SIUL<br>DSPI_2<br>VIU2/PDI<br>PWM/Timer | I/O | S | None,<br>None | — | 35 | AF1 |
| PL[9] | PCR[142] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[142]<br>SCK_2<br>PDI_PCLK<br>eMIOS1[21] | — | SIUL<br>DSPI_2<br>PDI<br>PWM/Timer | I/O | S | None,<br>None | — | 36 | AF2 |
| PL[10] | PCR[143] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[143]<br>eMIOS1[10]<br>DCULITE_G2<br>— | — | SIUL<br>PWM/Timer<br>DCULite<br>— | I/O | M | None,<br>None | — | 174 | C24 |
| PL[11] | PCR[144] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[144]<br>eMIOS1[11]<br>DCULITE_G3<br>— | — | SIUL<br>PWM/Timer<br>DCULite<br>— | I/O | M | None,<br>None | — | 175 | A24 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PL[12] | PCR[145] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[145]<br>eMIOS1[12]<br>DCULITE_G4<br>— | — | SIUL<br>PWM/Timer<br>DCULite<br>— | I/O | M | None,<br>None | — | 176 | C23 |
| PL[13] | PCR[146] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[146]<br>eMIOS1[13]<br>DCULITE_G5<br>— | — | SIUL<br>PWM/Timer<br>DCULite<br>— | I/O | M | None,<br>None | — | 177 | B23 |
| PL[14] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PL[15] | — | — | Reserved | — | — | — | — | — | — | — | — |
| **PORT M** | | | | | | | | | | | |
| PM[0] | PCR[147] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[147]<br>I2S_SCK / PWMO<br>DCULITE_R5<br>TCON[11] | — | SIUL<br>SGM<br>DCULite<br>TCON | I/O | M | None,<br>None | — | 61 | AE9 |
| PM[1] | PCR[148] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[148]<br>I2S_FS<br>DCULITE_R6<br>— | — | SIUL<br>SGM<br>DCULite<br>— | I/O | M | None,<br>None | — | 62 | AF9 |
| PM[2] | PCR[149] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[149]<br>eMIOS1[17]<br>DCULITE_R7<br>DCULITE_DE | RSDSCLKM | SIUL<br>PWM/Timer<br>DCULite<br>DCULite | I/O | M | None,<br>None | — | 173 | D23 |
| PM[3] | PCR[150] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[150]<br>CANRX_2<br>RXD_3<br>TCON[4] | — | SIUL<br>FlexCAN_2<br>LINFlex_3<br>TCON | I/O | M | None,<br>None | — | 16 | Y3 |
| PM[4] | PCR[151] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[151]<br>CANTX_2<br>TXD_3<br>TCON[5] | — | SIUL<br>FlexCAN_2<br>LINFlex_3<br>TCON | I/O | M | None,<br>None | — | 17 | Y2 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PM[5] | PCR[152] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[152]<br>VIU5_PDI13<br>eMIOS1[22]<br>DCU_TAG | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>DCU3 | I/O | M | None,<br>None | 16 | — | — |
| PM[6] | PCR[153] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[153]<br>VIU6_PDI14<br>eMIOS1[23]<br>DCULITE_TAG | — | SIUL<br>VIU2/PDI<br>PWM/Timer<br>DCULite | I/O | M | None,<br>None | 17 | — | — |
| PM[7] | PCR[154] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[154]<br>VIU8_PDI16<br>I2S_DO / PWMOA<br>eMIOS1[16] | — | SIUL<br>VIU2/PDI<br>SGM<br>PWM/Timer | I/O | S | None,<br>None | 39 | — | — |
| PM[8] | PCR[155] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[155]<br>VIU9_PDI17<br>I2S_SCK / PWMO<br>eMIOS1[23] | — | SIUL<br>VIU2/PDI<br>SGM<br>PWM/Timer | I/O | S | None,<br>None | 40 | — | — |
| PM[9] | PCR[156] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[156]<br>PDI_PCLK<br>SGM_MCLK<br>eMIOS0[8] | — | SIUL<br>PDI<br>SGM<br>PWM/Timer | I/O | M | None,<br>None | 113 | — | — |
| PM[10] | PCR[157] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[157]<br>RXD_2<br>CANRX_2<br>eMIOS0[16] | — | SIUL<br>LINFlex_2<br>FlexCAN_2<br>PWM/Timer | I/O | S | None,<br>None | 114 | — | — |
| PM[11] | PCR[158] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[158]<br>TXD_2<br>CANTX_2<br>eMIOS0[23] | — | SIUL<br>LINFlex_2<br>FlexCAN_2<br>PWM/Timer | I/O | S | None,<br>None | 115 | — | — |
| PM[12] | PCR[159] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[159]<br>DCULITE_B7<br>—<br>I2S_SCK / PWMO | — | SIUL<br>DCULite<br>—<br>SGM | I/O | M | None,<br>None | — | — | L24 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PM[13] | PCR[160] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[160]<br>DCULITE_PCLK<br>—<br>SGM_MCLK | — | SIUL<br>DCULite<br>—<br>SGM | I/O | F | None,<br>None | — | — | L23 |
| PM[14] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PM[15] | — | — | Reserved | — | — | — | — | — | — | — | — |
| **PORT N** | | | | | | | | | | | |
| PN[0] | PCR[161] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[161]<br>DCULITE_HSYNC<br>—<br>TCON[4] | — | SIUL<br>DCULite<br>—<br>TCON | I/O | M | None,<br>None | — | — | AC3 |
| PN[1] | PCR[162] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[162]<br>DCULITE_VSYNC<br>—<br>TCON[5] | — | SIUL<br>DCULite<br>—<br>TCON | I/O | M | None,<br>None | — | — | AD3 |
| PN[2] | PCR[163] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[163]<br>DCULITE_R0<br>RXD_2<br>VIU0_PDI8 | — | SIUL<br>DCULite<br>LINFlex_2<br>VIU2/PDI | I/O | M | None,<br>None | — | — | AC10 |
| PN[3] | PCR[164] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[164]<br>DCULITE_R1<br>TXD_2<br>VIU1_PDI9 | — | SIUL<br>DCULite<br>LINFlex_2<br>VIU2/PDI | I/O | M | None,<br>None | — | — | AF10 |
| PN[4] | PCR[165] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[165]<br>DCULITE_R2<br>—<br>TCON[6] | — | SIUL<br>DCULite<br>—<br>TCON | I/O | M | None,<br>None | — | — | AC11 |
| PN[5] | PCR[166] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[166]<br>DCULITE_R3<br>—<br>TCON[7] | — | SIUL<br>DCULite<br>—<br>TCON | I/O | M | None,<br>None | — | — | AD11 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PN[6] | PCR[167] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[167]<br>DCULITE_R4<br>—<br>TCON[8] | — | SIUL<br>DCULite<br>—<br>TCON | I/O | M | None,<br>None | — | — | AE11 |
| PN[7] | PCR[168] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[168]<br>DCULITE_R5<br>—<br>TCON[9] | — | SIUL<br>DCULite<br>—<br>TCON | I/O | M | None,<br>None | — | — | AF11 |
| PN[8] | PCR[169] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[169]<br>DCULITE_R6<br>—<br>TCON[10] | — | SIUL<br>DCULite<br>—<br>TCON | I/O | M | None,<br>None | — | — | AC12 |
| PN[9] | PCR[170] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[170]<br>DCULITE_R7<br>—<br>TCON[11] | — | SIUL<br>DCULite<br>—<br>TCON | I/O | M | None,<br>None | — | — | AD12 |
| PN[10] | PCR[171] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[171]<br>DCULITE_G0<br>RXD_3<br>VIU2_PDI10 | — | SIUL<br>DCULite<br>LINFlex_3<br>VIU2/PDI | I/O | M | None,<br>None | — | — | AE12 |
| PN[11] | PCR[172] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[172]<br>DCULITE_G1<br>TXD_3<br>VIU3_PDI11 | — | SIUL<br>DCULite<br>LINFlex_3<br>VIU2/PDI | I/O | M | None,<br>None | — | — | AF12 |
| PN[12] | PCR[173] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[173]<br>DCULITE_G2<br>—<br>eMIOS0[17] | — | SIUL<br>DCULite<br>—<br>PWM/Timer | I/O | M | None,<br>None | — | — | R26 |
| PN[13] | PCR[174] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[174]<br>DCULITE_G3<br>—<br>eMIOS0[18] | — | SIUL<br>DCULite<br>—<br>PWM/Timer | I/O | M | None,<br>None | — | — | R25 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PN[14] | PCR[175] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[175]<br>DCULITE_G4<br>—<br>eMIOS0[19] | — | SIUL<br>DCULite<br>—<br>PWM/Timer | I/O | M | None,<br>None | — | — | P26 |
| PN[15] | PCR[176] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[176]<br>DCULITE_G5<br>—<br>eMIOS0[20] | — | SIUL<br>DCULite<br>—<br>PWM/Timer | I/O | M | None,<br>None | — | — | P24 |
| **PORT P** | | | | | | | | | | | |
| PP[0] | PCR[177] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[177]<br>DCULITE_G6<br>—<br>eMIOS0[21] | — | SIUL<br>DCULite<br>—<br>PWM/Timer | I/O | M | None,<br>None | — | — | P23 |
| PP[1] | PCR[178] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[178]<br>DCULITE_G7<br>—<br>eMIOS0[22] | — | SIUL<br>DCULite<br>—<br>PWM/Timer | I/O | M | None,<br>None | — | — | N26 |
| PP[2] | PCR[179] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[179]<br>DCULITE_B0<br>CANRX_2<br>VIU4_PDI12 | — | SIUL<br>DCULite<br>FlexCAN_2<br>VIU2/PDI | I/O | M | None,<br>None | — | — | N25 |
| PP[3] | PCR[180] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[180]<br>DCULITE_B1<br>CANTX_2<br>PDI_DE | — | SIUL<br>DCULite<br>FlexCAN_2<br>PDI | I/O | M | None,<br>None | — | — | N23 |
| PP[4] | PCR[181] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[181]<br>DCULITE_B2<br>—<br>eMIOS0[11] | — | SIUL<br>DCULite<br>—<br>PWM/Timer | I/O | M | None,<br>None | — | — | M26 |
| PP[5] | PCR[182] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[182]<br>DCULITE_B3<br>—<br>eMIOS0[13] | — | SIUL<br>DCULite<br>—<br>PWM/Timer | I/O | M | None,<br>None | — | — | M25 |

**Table 5. Port pin summary (continued)**

| Port pin | PCR | Alternate function[1] | Function | Special function[2] | Peripheral[3] | I/O direction | Pad Type[4] | RESET config[5] | Pin number | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 176 LQFP | 208 LQFP | 416 TEPBGA |
| PP[6] | PCR[183] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[183]<br>DCULITE_B4<br>—<br>eMIOS0[15] | — | SIUL<br>DCULite<br>—<br>PWM/Timer | I/O | M | None,<br>None | — | — | M24 |
| PP[7] | PCR[184] | Option 0<br>Option 1<br>Option 2<br>Option 3 | GPIO[184]<br>DCULITE_B5<br>—<br>I2S_FS | — | SIUL<br>DCULite<br>—<br>SGM | I/O | M | None,<br>None | — | — | M23 |
| PP[8] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PP[9] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PP[10] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PP[11] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PP[12] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PP[13] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PP[14] | — | — | Reserved | — | — | — | — | — | — | — | — |
| PP[15] | — | — | Reserved | — | — | — | — | — | — | — | — |

1 Alternate functions are chosen by setting the values of the PCR[PA] bitfields inside the SIUL module.
PCR[PA] = 00 selects Option 0
PCR[PA] = 01 selects Option 1
PCR[PA] = 10 selects Option 2
PCR[PA] = 11 selects Option 3
This is intended to select the output functions. To use one of the input functions, the PCR[IBE] bit must be written to '1', regardless of the values selected in the PCR[PA] bitfields. For this reason, the value corresponding to an input only function is reported as "—".

2 Special functions are enabled independently from the standard digital pin functions. Enabling standard I/O functions in the PCR registers may interfere with their functionality. ADC functions are enabled using the PCR[APC] bit; other functions are enabled by enabling the respective module.

3 Using the PSMI registers in the System Integration Unit Lite (SIUL), different pads can be multiplexed to the same peripheral input. Please see the SIUL chapter of the *MPC5645S Microcontroller Reference Manual* for details.

4 See the "Pad types" table for an explanation of the letters in this column.

5 Reset configuration is given as I/O direction and pull, e.g., "Input, pullup".

6 Out of reset pins PH[0:3] are available as JTAG pins (TCK, TDI, TDO and TMS respectively). It is up to the user to configure pins PH[0:3] when needed.

### 3.3.10 TCON Pins

The location of TCON[0:3] pins is mentioned in the following table:

**Table 6. Location of TCON[0:3] Pins**

| Function | Port Name | PCR | Ball Number |
|---|---|---|---|
| DCU_TAG / TCON0 | PK11 | PCR[132] | Ball No: AC09 |
| DCU_TAG / TCON0 | PJ12 | PCR[117] | Ball No: A23 |
| DCU_HSYNC/TCON1 | PG9 | PCR[95] | Ball No: T02 |
| DCU_VSYNC / TCON2 | PG8 | PCR[94] | Ball No: T04 |
| DCU_DE / TCON3 | PG10 | PCR[96] | Ball No: T01 |
| DCU_TAG/TCON3 | PM5 | PCR[152] | NC (only on 176) |

**Table 3-7. Pad type descriptions[1]**

| Abbreviation | Description |
|---|---|
| F | Fast (pad with slew rate) |
| J | Fast / medium / slow / I/O pad with analog feature (GPIO and analog functionality) |
| M | Medium (pad with slew rate control) |
| M / RSDS | Medium (pad with slew rate control) shared with RSDS alternate function |
| S | Slow (pad with slew rate control) |
| SMD | Stepper motor driver (pad with reduced slew rate control) |
| X | Oscillator |

1 The pad descriptions refer to the different Pad Configuration Register (PCR) types. Refer to the SIUL chapter in the device reference manual for the features available for each pad type.

# Chapter 4
# Safety

## 4.1 Register Protection

### 4.1.1 Introduction

#### 4.1.1.1 Overview

The Register Protection module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

The protection module is located between the module under protection and the PBRIDGE. This is shown in Figure 4-1.

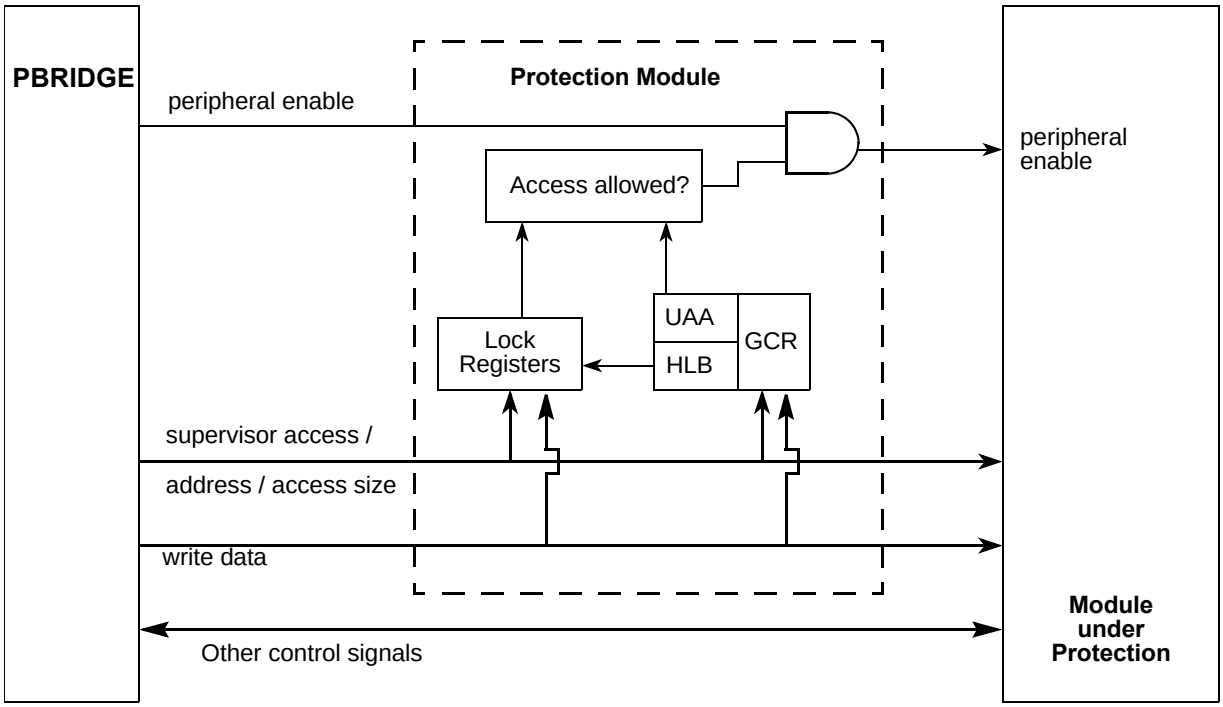



**Figure 4-1. Register Protection Block Diagram**

#### 4.1.1.2 Features

The Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Address mirror automatically sets corresponding lock bit

- Once configured lock bits can be protected from changes

#### 4.1.1.3 Modes of Operation

The Register Protection module is operable when the module under protection is operable. For further details about the availability please see the module's chapter in this document.

### 4.1.2 External Signal Description

There are no external signals.

### 4.1.3 Memory Map and Register Description

This section provides a detailed description of the memory map of a module using the Register Protection. The original 16 KB module memory space is divided into five areas as shown in Figure 4-2.

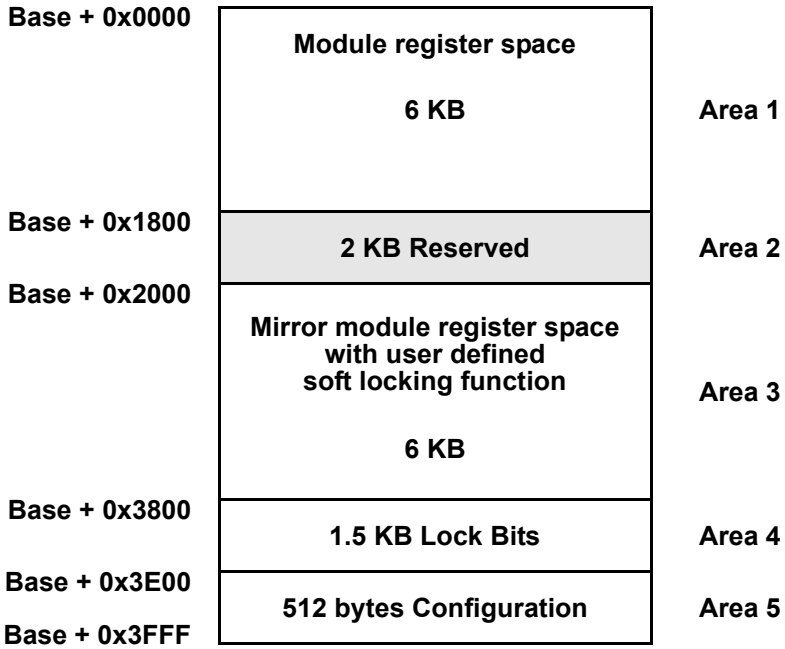



**Figure 4-2. Register Protection Memory Diagram**

Area 1 is 6 KB large and holds the normal functional module registers and is transparent for all read/write operations.

Area 2 is 2 KB starting at address 0x1800 is a reserved area, which shall not be accessed.

Area 3 is 6 KB large, starting at address 0x2000 and is a mirror of area 1. A read/write access to these 0x2000+X addresses will read/write the register at address X. As a side effect, a write access to address 0x2000+X will set the optional Soft Lock Bits for this address X in the same cycle as the register at address X is written. Not all registers in area 1 need to have protection defined by associated Soft Lock Bits. For

unprotected registers at address Y, accesses to address 0x2000+Y will be identical to accesses at address Y. Only for registers implemented in area 1 and defined as protectable Soft Lock Bits will be available in area 4.

Area 4 is 1.5 KB large and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.

Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

#### 4.1.3.1 Memory Map

Table 4-1 gives an overview on the **Register Protection** registers implemented.

**Table 4-1. Register protection memory map**

| Address offset | Register name | Location |
|---|---|---|
| 0x0000 | Module Register 0 (MR0) | on page 134 |
| 0x0001 | Module Register 1 (MR1) | on page 134 |
| 0x0002 | Module Register 2 (MR2) | on page 134 |
| 0x0003 - 0x17FF | Module Register 3 (MR3) - Module Register 6143(MR6143) | on page 134 |
| 0x1800 - 0x1FFF | Reserved | |
| 0x2000 | Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0) | on page 134 |
| 0x2001 | Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1) | on page 134 |
| 0x2002 - 0x37FF | Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143) | on page 134 |
| 0x3800 | Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3 | on page 134 |
| 0x3801 | Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7 | on page 134 |
| 0x3802 - 0x3DFF | Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 - Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143 | on page 134 |
| 0x3E00 - 0x3FFB | Reserved | |
| 0x3FFC | Global Configuration Register (GCR) | on page 135 |

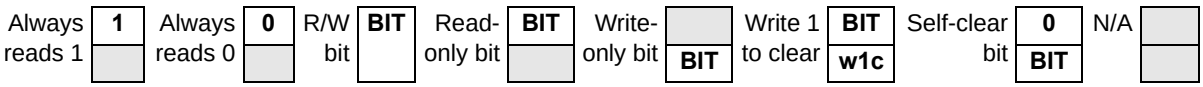
**NOTE**

Reserved registers in area #2 will be handled according to the protected IP (module under protection).

### 4.1.3.2 Register description

This section describes in address order all the Register Protection registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

**Figure 4-3. Key to Register Fields**




#### 4.1.3.2.1 Module Registers (MR0-6143)

This is the lower 6K module memory space which holds all the functional registers of the module that is protected by the Register Protection module.

#### 4.1.3.2.2 Module Register and Set Soft Lock Bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBR*n*.SLB*m*, according to the mapping described in Table 4-2.

#### 4.1.3.2.3 Soft Lock Bit Register (SLBR0-1535)

These registers hold the Soft Lock Bits for the protected registers in memory area #1.

Address: 0x3800-0x3DFF — Access: Read always, Supervisor write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | SLB0 | SLB1 | SLB2 | SLB3 |
| W | WE0 | WE1 | WE2 | WE3 | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4-4. Soft Lock Bit Register (SLBR*n*)**

**Table 4-2. SLBR*n* Field Descriptions**

| Field D | escription |
|---|---|
| WE0<br>WE1<br>WE2<br>WE3 | Write Enable Bits for Soft Lock Bits (SLB):<br>WE0 enables writing to SLB0<br>WE1 enables writing to SLB1<br>WE2 enables writing to SLB2<br>WE3 enables writing to SLB3<br><br>1 Value is written to SLB<br>0 SLB is not modified |
| SLB0<br>SLB1<br>SLB2<br>SLB3 | Soft Lock Bits for one MR$n$ register:<br>SLB0 can block accesses to MR[$n$ *4 + 0]<br>SLB1 can block accesses to MR[$n$ *4 + 1]<br>SLB2 can block accesses to MR[$n$ *4 + 2]<br>SLB3 can block accesses to MR[$n$ *4 + 3]<br><br>1 Associated MR$n$ byte is locked against write accesses<br>0 Associated MR$n$ byte is unprotected and writable |

Table 4-3 gives some examples of how SLBR*n*.SLB and MR*n* go together:

**Table 4-3. Soft Lock Bits vs. Protected Address**

| Soft Lock Bit | Protected address |
|---|---|
| SLBR0.SLB0 | MR0 |
| SLBR0.SLB1 | MR1 |
| SLBR0.SLB2 | MR2 |
| SLBR0.SLB3 | MR3 |
| SLBR1.SLB0 | MR4 |
| SLBR1.SLB1 | MR5 |
| SLBR1.SLB2 | MR6 |
| SLBR1.SLB3 | MR7 |
| SLBR2.SLB0 | MR8 |
| ... | ... |

#### 4.1.3.2.4 Global Configuration Register (GCR)

This register is used to make global configurations related with the Register Protection.

Address: 0x3FFC Access: Read Always Supervisor write

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

**Figure 4-5. Global Configuration Register (GCR)**

**Table 4-4. GCR Field Descriptions**

| Field D | escription |
|---|---|
| HLB | Hard Lock Bit.<br>This register can not be cleared once it is set by software. It can only be cleared by a system reset.<br><br>1 All SLB bits are write protected and can not be modified<br>0 All SLB bits are accessible and can be modified. |
| UAA | User Access Allowed.<br><br>1 The registers in the module under protection can be accessed in the User mode without any additional restrictions.<br>0 The registers in the module under protection can only be written in supervisor mode. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module. |

**NOTE**

The GCR.UAA bit has no effect on the allowed access modes for the registers in the Register Protection module.

## 4.1.4 Functional Description

### 4.1.4.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)
- unprotected (address == multiples of 1)

Which addresses are protected and the protection size depend on the SoC and/or module. Therefore this section can just give examples for various protection configurations.

For all addresses that are protected there are SLBR*n*.SLB*m* bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBR*n*.SLB*m* bit is always 0b0 no matter what software is writing to.

### 4.1.4.2 Change Lock Settings

To change the setting whether an address is locked or unlocked the corresponding SLBR*n*.SLB*m* bit needs to be changed. This can be done using the following methods:

- Modify the SLBR*n*.SLB*m* directly by writing to area #4
- Set the SLBR*n*.SLB*m* bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

#### 4.1.4.2.1 Change Lock Settings Directly Via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each SLBR*n*.SLB*m* bit has a mask bit SLBR*n*.WE*m* which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

Figure 4-6 shows two modification examples. In the left example there is a write access to the SLBR*n* register specifying a mask value which allows modification of all SLBR*n*.SLB*m* bits. The example on the right specifies a mask which only allows modification of the bits SLBR*n*.SLB[3:1].

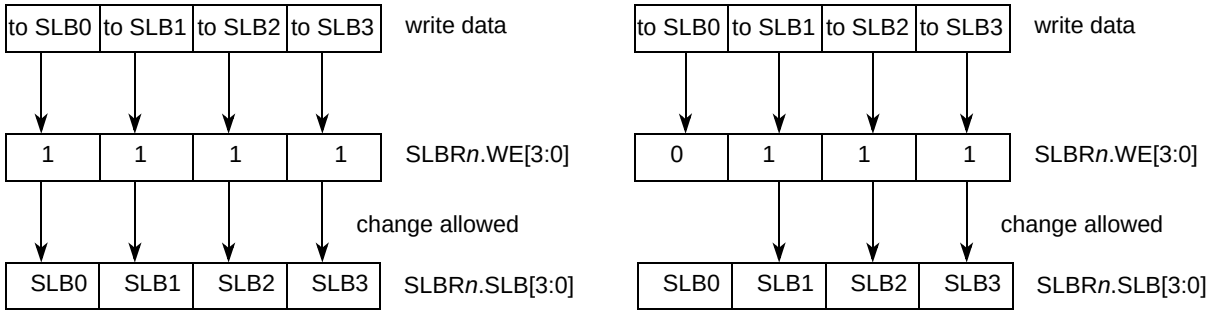



**Figure 4-6. Change Lock Settings Directly Via Area #4**

Figure 4-6 showed four registers that can be protected 8-bit wise. In Figure 4-7 registers with 16-bit protection and in Figure 4-8 registers with 32-bit protection are shown:

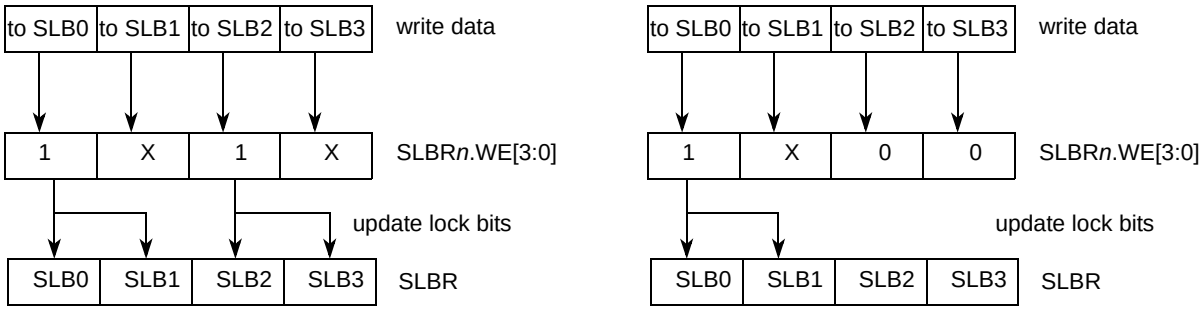



**Figure 4-7. Change Lock Settings for 16-bit Protected Addresses**

On the right side of Figure 4-7 it is shown that the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[1] also. This is done as the address reflected by SLBR*n*.SLB[0] is protected 16-bit wise.

Note that in this case the write enable SLBR*n*.WE[0] must be set while SLBR*n*.WE[1] does not matter. As the enable bits SLBR*n*.WE[3:2] are cleared the lock bits SLBR*n*.SLB[3:2] remain unchanged.

In the example on the left side of Figure 4-7 the data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] and the data written to SLBR*n*.SLB[2] is mirrored to SLBR*n*.SLB[3] as for both registers the write enables are set.

In Figure 4-8 a 32-bit wise protected register is shown. When SLBR*n*.WE[0] is set the data written to SLBR*n*.SLB[0] is automatically written to SLBR*n*.SLB[3:1] also. Otherwise SLBR*n*.SLB[3:0] remains unchanged.

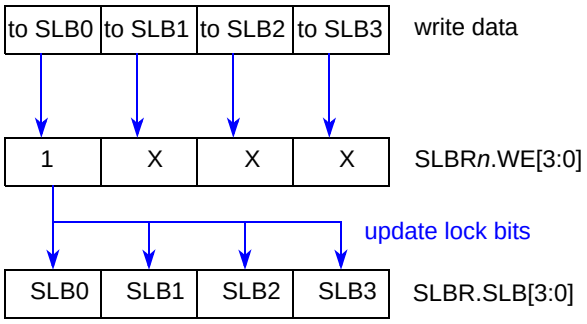



**Figure 4-8. Change Lock Settings for 32-bit Protected Addresses**

In Figure 4-9 an example is shown which has a mixed protection size configuration:

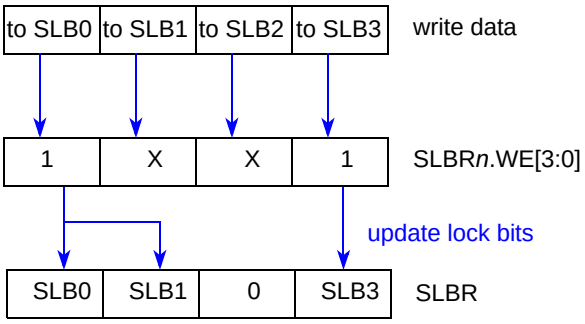



**Figure 4-9. Change Lock Settings for Mixed Protection**

The data written to SLBR*n*.SLB[0] is mirrored to SLBR*n*.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBR*n*.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBR*n*.SLB[3] is written to SLBR*n*.SLB[3].

#### 4.1.4.2.2 Enable Locking Via Mirror Module Space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. Figure 4-10 shows one example:

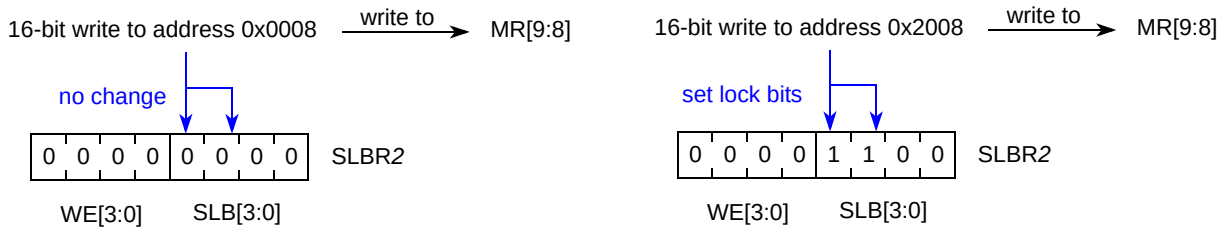



**Figure 4-10. Enable Locking Via Mirror Module Space (Area #3)**

When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (left part of Figure 4-7).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of Figure 4-7).

Figure 4-11 shows an example where some addresses are protected and some are not:

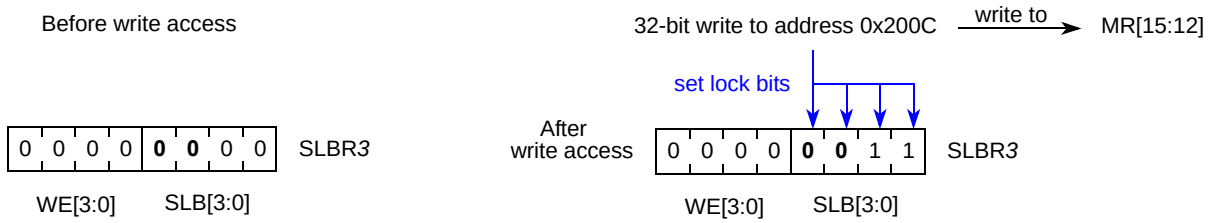



**Figure 4-11. Enable Locking for Protected and Unprotected Addresses**

In the example in Figure 4-11 addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

**NOTE**

Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

#### 4.1.4.2.3 Write Protection for Locking Bits

Changing the locking bits through any of the procedures mentioned in Section 4.1.4.2.1, “Change Lock Settings Directly Via Area #4” and Section 4.1.4.2.2, “Enable Locking Via Mirror Module Space (Area #3)” is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there was a system reset.

### 4.1.4.3 Access Errors

The protection module generates transfer errors under several circumstances. For the area definition refer to Figure 4-2.

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.

2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.
6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set will result in a transfer error.

### 4.1.5 Reset

The reset state of each individual bit is shown within the Register Description section (See Section 4.1.3.2, “Register description”). In summary, after reset, locking for all MR*n* registers is disabled. The registers can be accessed in Supervisor Mode only.

## 4.2 Software Watchdog Timer (SWT)

### 4.2.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing sequence. Writing the sequence resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial time-out, a reset is always generated on a second consecutive time-out.

The SWT provides a window functionality. When this functionality is programmed, the servicing action should take place within the defined window. When occurring outside the defined period, the SWT will generate a reset.

### 4.2.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- The unique SWT counter clock is the undivided low power internal oscillator (IRC 128 khz), no other clock source can be selected
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Master access protection
- Hard and soft configuration lock bits

- The SWT is started on PHASE1 exit and counts unconditionally during PHASE2 to monitor flash boot sequence. SWT is held in reset during PHASE3 but can start again at PHASE3 exit depending on the value of the shadow flash configuration bit NVUSR0[WATCHDOG_EN].

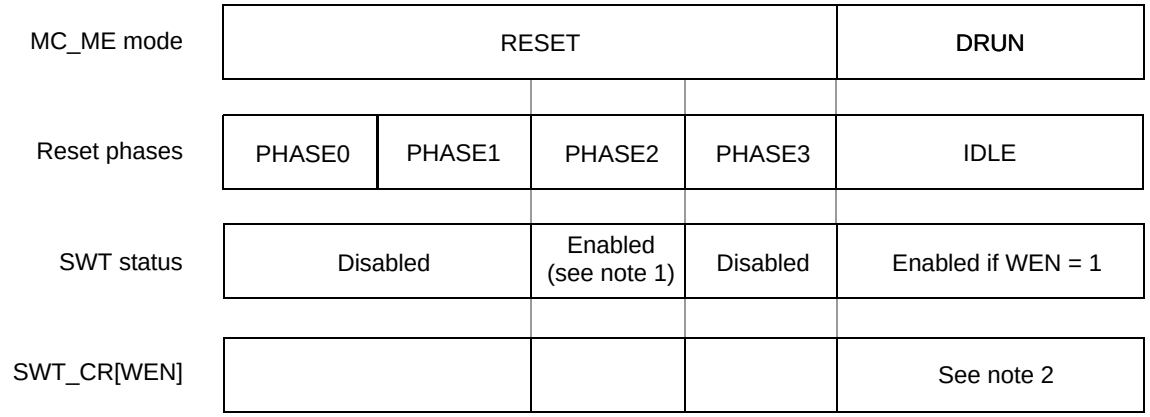
### 4.2.3 Modes of operation

When enabled, the SWT is always active in all modes except STANDBY when it is always disabled and STOP when it may be disabled by the SWT_CR[STP] bit. If the STP bit is set, the counter is stopped in STOP mode, otherwise it continues to run. On exit from STOP mode, the SWT will continue from the state it was before entering STOP mode.

To simplify software development it is possible to temporarily suspend the SWT counter while the MCU is stopped by a debugger. While the MCU is running the SWT counter runs normally. This feature is enabled by setting the SWT_CR[FRZ] bit and is only available when the CPU has debug mode active (see the CPU reference manual for more information on debug mode and support).

Software watchdog is not available in STANDBY mode. As soon as out of STANDBY mode, the SWT behaves as in a usual “out of reset” situation.

Figure 4-12 shows the operation timing diagram of the SWT. Table 4-5describes the SWT operation after reset.




Notes:
1) The SWT is started on PHASE1 exit and counts unconditionally during PHASE2 to monitor the flash memory boot sequence.
2) Value copied from configuration bit NVUSR0[WATCHDOG_EN] in the shadow flash memory (software can modify it later)

**Figure 4-12. SWT operation timing diagram**

**Table 4-5. SWT operation after reset**

| SWT_CR [WEN] | MCU mode | CPU debug active | SWT_CR [FRZ] | SWT_CR [STP] | SWT operation |
|---|---|---|---|---|---|
| 0 | — | No | 0 or 1 | 0 or 1 | Off |

Table 4-5. SWT operation after reset (continued)

| SWT_CR [WEN] | MCU mode | CPU debug active | SWT_CR [FRZ] | SWT_CR [STP] | SWT operation |
|---|---|---|---|---|---|
| 1 | Normal (MC_ME modes DRUN, RUN0:3, HALT, SAFE) | No | 0 or 1 | 0 or 1 | Running |
| | Debug[1] (MC_ME modes DRUN, RUN0:3, HALT, SAFE) | Yes | 0 | 0 or 1 | Running |
| | | Yes | 1 | 0 or 1 | Halted |
| | STOP (MC_ME mode STOP) | No | 0 or 1 | 0 | Running |
| | | No | 0 or 1 | 1 | Halted |
| 0 or 1 | STANDBY | No | No | No | Off |

1 SWT Debug Mode occurs when the processor is stopped due to user specified debug criteria such as breakpoint.

## 4.2.4 External signal description

The SWT module does not have any external interface signals.

## 4.2.5 Memory map and register description

The SWT programming model has six 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. Other types of invalid accesses include: writes to read only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid access otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_CR are set then the SWT_CR, SWT_TO and SWT_WN registers are read only.

### 4.2.5.1 Memory map

The SWT memory map is shown in Table 4-6.

Table 4-6. SWT memory map

| Address offset | Register name | Location |
|---|---|---|
| 0x0000 | SWT Control Register (SWT_CR) | on page 143 |
| 0x0004 | SWT Interrupt Register (SWT_IR) | on page 144 |
| 0x0008 | SWT Time-out Register (SWT_TO) | on page 145 |
| 0x000C | SWT Window Register (SWT_WN) | on page 145 |
| 0x0010 | SWT Service Register (SWT_SR) | on page 146 |
| 0x0014 | SWT Counter Output Register (SWT_CO) | on page 146 |
| 0x0018–0x3FFF | Reserved | |

## 4.2.6 Register summary

The following sections detail the individual registers within the SWT programming model.

### 4.2.6.1 SWT Control Register (SWT_CR)

The SWT_CR contains fields for configuring and controlling the SWT. This register is read only if either the SWT_CR.HLK or SWT_CR.SLK bits are set.

Offset 0x0000 Access: Read/Write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RIA | WND | ITR | HLK | SLK | CSL | STP | FRZ | WEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | See note |

**Note:** SWT_CR[WEN] value is copied from configuration bit NVUSR0[WATCHDOG_EN] during reset

**Figure 4-13. SWT Control Register (SWT_CR)**

The SWT_CR reset value is 0x8000_011A or 0x8000_011B, corresponding to MAP0 = 1 (only CPU access allowed), RIA = 1 (reset on invalid SWT access), SLK = 1 (soft lock), CSL = 1 (IRC clock source for counter), FRZ = 1 (freeze available while debugging), WEN = 0 or 1 (copied from configuration bit NVUSR0[WATCHDOG_EN]).

**Table 4-7. SWT_CR field descriptions**

| Field | Description |
|---|---|
| MAPn | Master Access Protection for Master n. The platform bus master assignments are device specific.<br>0 = Access for the master is not enabled<br>1 = Access for the master is enabled |
| RIA | Reset on Invalid Access.<br>0 = Invalid access to the SWT generates a bus error<br>1 = Invalid access to the SWT causes a system reset if WEN=1 |
| WND | Window Mode.<br>0 = Regular mode, service sequence can be done at any time<br>1 = Windowed mode, the service sequence is only valid when the down counter is less than the value in the SWT_WN register. |
| ITR | Interrupt Then Reset.<br>0 = Generate a reset on a time-out<br>1 = Generate an interrupt on an initial time-out, reset on a second consecutive time-out |

**Table 4-7. SWT_CR field descriptions (continued)**

| Field | Description |
|---|---|
| HLK | Hard Lock. This bit is only cleared at reset.<br>0 = SWT_CR, SWT_TO and SWT_WN are read/write registers if SLK=0<br>1 = SWT_CR, SWT_TO and SWT_WN are read only registers |
| SLK | Soft Lock. This bit is cleared by writing the unlock sequence to the service register.<br>0 = SWT_CR, SWT_TO and SWT_WN are read/write registers if HLK=0<br>1 = SWT_CR, SWT_TO and SWT_WN are read only registers |
| CSL | Clock Selection. Selects the LP IRC 128 khz oscillator clock that drives the internal timer.<br>CSL bit can be written.The status of the bit has no effect on counter clock selection on this device.<br>0 = System clock. (Not applicable on this device)<br>1 = Oscillator clock. |
| STP | STOP Mode Control. Allows the watchdog timer to be stopped when the device enters STOP mode.<br>0 = SWT counter continues to run in STOP mode<br>1 = SWT counter is stopped in STOP mode |
| FRZ | Freeze available during debug. This function is only operational when the CPU is in an active debug mode.<br>0 = SWT counter continues to run independent of the CPU status<br>1 = SWT counter is stopped when the CPU is stopped by a debugger |
| WEN | Watchdog Enabled.<br>0 = SWT is disabled<br>1 = SWT is enabled |

### 4.2.6.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Offset 0x0004 Access: Read/Write

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

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

**Figure 4-14. SWT Interrupt Register (SWT_IR)**

**Table 4-8. SWT_IR Field Descriptions**

| Field | Description |
|---|---|
| TIF | Time-out Interrupt Flag. The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect.<br>0 = No interrupt request.<br>1 = Interrupt request due to an initial time-out. |

### 4.2.6.3 SWT Time-Out Register (SWT_TO)

The SWT Time-Out (SWT_TO) register contains the 32-bit time-out period. This register is read-only if either the SWT_CR.HLK or SWT_CR.SLK bits are set.

Offset 0x008 Access: Read/Write

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

**Figure 4-15. SWT Time-Out Register (SWT_TO)**

The default counter value (SWT_TO_RST) is 1920 (0x0000_0780 hexadecimal) which corresponds to 15 ms with a 128 kHz clock.

**Table 4-9. SWT_TO Register Field Descriptions**

| Field | Description |
|---|---|
| WTO | Watchdog time-out period in clock cycles. An internal 32-bit down counter is loaded with this value or 0x100 which ever is greater when the service sequence is written or when the SWT is enabled. |

#### 4.2.6.3.1 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read only if either the SWT_CR.HLK or SWT_CR.SLK bits are set.

Offset 0x00C Access: Read/Write

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

**Figure 4-16. SWT Window Register (SWT_WN)**

**Table 4-10. SWT_WN Register Field Descriptions**

| Field | Description |
|---|---|
| WST | Window start value. When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value. |

### 4.2.6.3.2 SWT Service Register (SWT_SR)

The SWT Time-Out (SWT_SR) service register is the target for service sequence writes used to reset the watchdog timer.

Offset 0x010 Access: Read/Write

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

**Figure 4-17. SWT Service Register (SWT_SR)**

**Table 4-11. SWT_SR Field Descriptions**

| Field | Description |
|---|---|
| WSC | Watchdog Service Code.This field is used to service the watchdog and to clear the soft lock bit (SWT_CR.SLK). To service the watchdog, the value 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_CR.SLKSWT_CR.), the value 0xC520 followed by 0xD928 is written to the WSC field. |

### 4.2.6.3.3 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read only register that shows the value of the internal down counter when the SWT is disabled.

Offset 0x014 Access: Read Only

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

**Figure 4-18. SWT Counter Output Register (SWT_CO)**

Table 4-12. SWT_CO Register Field Descriptions

| Field | Description |
|---|---|
| CNT | Watchdog Count. When the watchdog is disabled (SWT_CR.WENSWT_CR.=0) this field shows the value of the internal down counter. When the watchdog is enabled the value of this field is 0x0000_0000. Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter. |

### 4.2.7 Functional Description

The SWT is a 32-bit timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes a a control register (SWT_CR), an interrupt register (SWT_IR), time-out register (SWT_TO), a window register (SWT_WN), a service register (SWT_SR) and a counter output register (SWT_CO).

The SWT_CR includes bits to enable the timer, set configuration options and lock configuration of the module. The watchdog is enabled by setting the SWT_CR.WEN bit. The reset value of the SWT_CR.WEN bit is 0 when exiting RESET mode if the flash user option bit 31 (WATCHDOG_EN) is ‘0’. If the reset value of WATCHDOG_EN is 1, the SWT_CR.WEN bit is set and the watchdog starts operation automatically after reset is released.

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100 in which case the time-out period is set to 0x100. This time-out period is loaded into an internal 32-bit down counter when the SWT is enabled and each time a valid service sequence is written. The SWT_CR.CSL bit selects which clock (system or oscillator) is used to drive the down counter.

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked the SWT_CR, SWT_TO and SWT_WN registers are read only. The hard lock is enabled by setting the SWT_CR.HLK bit which can only be cleared by a reset. The soft lock is enabled by setting the SWT_CR.SLK bit and is cleared by writing the unlock sequence to the service register. The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR.WSC field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR.WEN bit to be set.

When enabled, the SWT requires periodic execution of the watchdog servicing sequence. The service sequence is a write of 0xA602 followed by a write of 0xB480 to the SWT_SR.WSC field. Writing the service sequence loads the internal down counter with the time-out period. There is no timing requirement between the two writes. The service sequence logic ignores unlock sequence writes and recognizes the 0xA602, 0xB480 sequence regardless of previous writes. Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

If window mode is enabled (SWT_CR.WND bit is set), the service sequence must be performed in the last part of the time-out period defined by the window register. The window is open when the down counter is less than the value in the SWT_WN register. Outside of this window, service sequence writes are invalid

accesses and generate a bus error or reset depending on the value of the SWT_CR.RIA bit. For example, if the SWT_TO register is set to 5000 and SWT_WN register is set to 1000 then the service sequence must be performed in the last 20% of the time-out period. There is a short lag in the time it takes for the window to open due to synchronization logic in the watchdog design. This delay could be up to three system plus four counter clock cycles.

The interrupt then reset bit (SWT_CR.ITR) controls the action taken when a time-out occurs. If the SWT_CR.ITR bit is not set, a reset is generated immediately on a time-out. If the SWT_CR.ITR bit is set, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period. If the service sequence is not written before the second consecutive time-out, the SWT generates a system reset. The interrupt is indicated by the time-out interrupt flag (SWT_IR.TIF). The interrupt request is cleared by writing a one to the SWT_IR.TIF bit.

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR.WEN cleared) and the value of the SWT_CO read to determine if the internal down counter is working properly.

# Chapter 5
# Analog-to-Digital Converter (ADC)

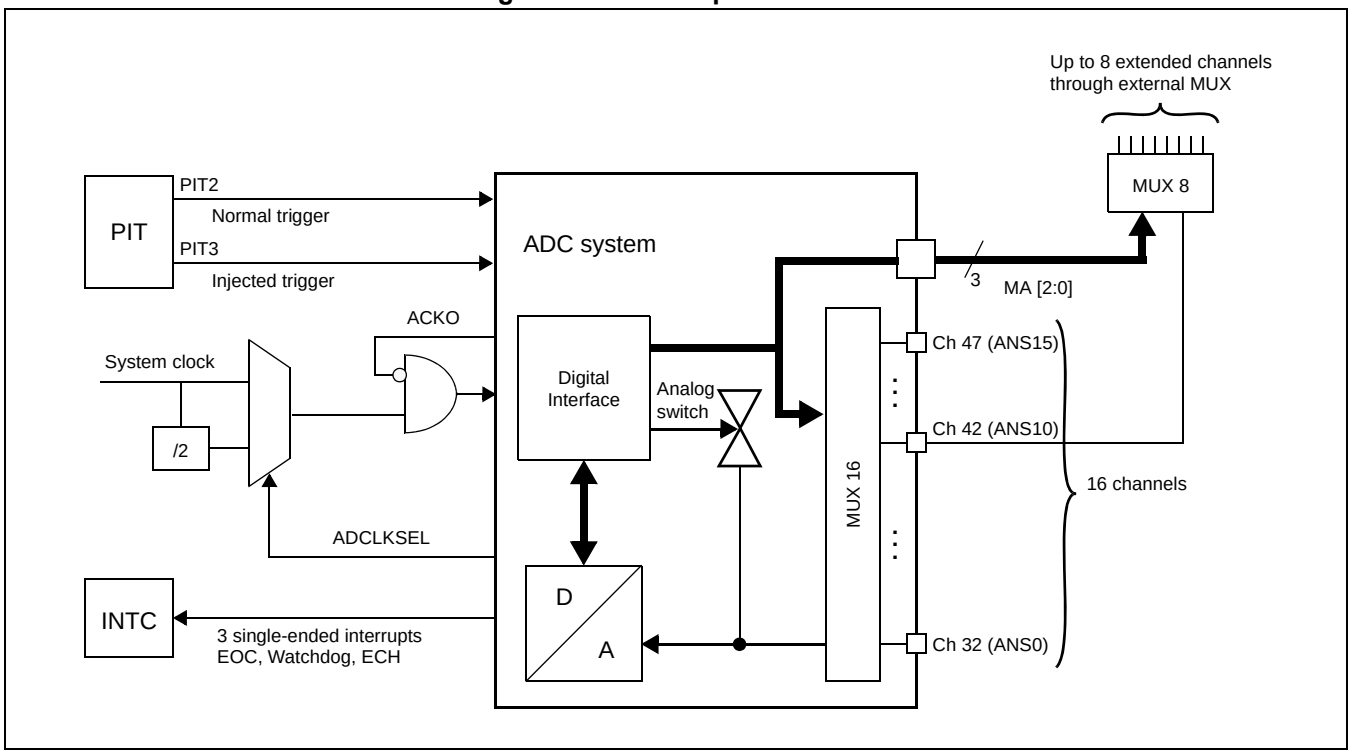
## 5.1 Overview

### 5.1.1 Device-specific features

- 10-bit resolution
- 20 channels, expandable to 27 channels via external multiplexing
- Minimum conversion time of 1 µs
- Conversion triggering sources:
  - Software
  - PIT channel 2 (used to trigger normal conversion)
  - PIT channel 3 (used to trigger injection conversion)
- 2 different sampling and conversion time registers CTR[1:2] (extended internal channels, external channels)
- As many as 23 data registers for storing converted data. Conversion information, such as mode of operation (normal, injected), is associated to data value.
- Conversions on external channels managed in the same way as internal channels, making it transparent to the application
- External decode signals (3 bits) to control the external analog multiplexers
- One Shot/Scan Modes
- Chain Injection Mode
- Hardware-triggered DMA transfer requests
- Power-down mode
- 2 different Abort functions allow to abort either single-channel conversion or chain conversion
- 4 programmable analog watchdogs with interrupt capability
  - Allows continuous hardware monitoring of as many as 4 analog input channels
  - Alternate analog thresholds
- Auto-clock-off

### 5.1.2 Device-specific implementation

**Figure 5-1. ADC implementation**




## 5.2 Introduction

The analog-to-digital converter (ADC) block provides accurate and fast conversions for a wide range of applications. An ADC analog part has its corresponding digital interface (ADCDig).

The ADC digital interface contains advanced features for normal or injected conversion. An injected conversion can be triggered by software or hardware (PIT3). A normal conversion can be triggered by software or hardware (PIT2).

There are two types of input channels:

- Internal extended, ANS (internally multiplexed standard accuracy channels)
- External, ANX (externally multiplexed standard accuracy channels)

The mask registers present within the ADCDig can be programmed to configure which channel has to be converted.

Three external decode signals MA[2:0] (multiplexer address) are provided for external channel selection and are available as alternate functions on GPIO.

A conversion timing register for configuring different sampling and conversion times is associated to each channel type.

Analog watchdogs allow continuous hardware monitoring.

## 5.3 Register descriptions

### 5.3.1 Introduction

Table 5-1 lists the ADC registers with their address offsets and reset values.

**Table 5-1. ADC digital registers**

| Address offset (hex.) | Register name | Location |
|---|---|---|
| 000 | Main Configuration Register (MCR) | on page 153 |
| 004 | Main Status Register (MSR) | on page 155 |
| 008 .. 00C | Reserved | — |
| 010 | Interrupt Status Register (ISR) | on page 157 |
| 014 | Reserved | — |
| 018 | Channel Pending Register (CEOCFR1) | on page 158 |
| 01C | Channel Pending Register (CEOCFR2) | on page 158 |
| 020 | Interrupt Mask Register (IMR) | on page 159 |
| 024 | Reserved | — |
| 028 | Channel Interrupt Mask Register (CIMR1) | on page 160 |
| 02C | Channel Interrupt Mask Register (CIMR2) | on page 160 |
| 030 | Watchdog Threshold Interrupt Status Register (WTISR) | on page 161 |
| 034 | Watchdog Threshold Interrupt Mask Register (WTIMR) | on page 162 |
| 038 .. 03C | Reserved | — |
| 040 | DMA Enable Register (DMAE) | on page 163 |
| 044 | Reserved | — |
| 048 | DMA Channel Select Register 1 (DMAR1) | on page 163 |
| 04C | DMA Channel Select Register 2 (DMAR2) | on page 163 |
| 050 | Threshold Control Register 0 (TRC0) | on page 165 |
| 054 | Threshold Control Register 1 (TRC1) | on page 165 |
| 058 | Threshold Control Register 2 (TRC2) | on page 165 |
| 05C | Threshold Control Register 3 (TRC3) | on page 165 |
| 060 | Threshold Register 0 (THRHLR0) | on page 165 |
| 064 | Threshold Register 1 (THRHLR1) | on page 165 |
| 068 | Threshold Register 2 (THRHLR2) | on page 165 |
| 06C | Threshold Register 3 (THRHLR3) | on page 165 |
| 070 .. 094 | Reserved | — |
| 098 | Conversion Timing Register 1 (CTR1) | on page 166 |

**Table 5-1. ADC digital registers (continued)**

| Address offset (hex.) | Register name | Location |
|---|---|---|
| 09C | Conversion Timing Register 2 (CTR2) | on page 166 |
| 0A0 .. 0A4 | Reserved | — |
| 0A8 | Normal Conversion Mask Register 1 (NCMR1) | on page 168 |
| 0AC | Normal Conversion Mask Register 2 (NCMR2) | on page 168 |
| 0B0 .. 0B4 | Reserved | — |
| 0B8 | Injected Conversion Mask Register 1 (JCMR1) | on page 169 |
| 0BC | Injected Conversion Mask Register 2 (JCMR2) | on page 169 |
| 0C0 | Reserved | — |
| 0C4 | Decode Signals Delay Register (DSDR) | on page 170 |
| 0C8 | Power-down Exit Delay Register (PDEDR) | on page 171 |
| 0CC .. 17C | Reserved | — |
| 180 | Channel 32 Data Register (CDR32) | on page 172 |
| 184 | Channel 33 Data Register (CDR33) | on page 172 |
| 188 | Channel 34 Data Register (CDR34) | on page 172 |
| 18C | Channel 35 Data Register (CDR35) | on page 172 |
| 190 | Channel 36 Data Register (CDR36) | on page 172 |
| 194 | Channel 37 Data Register (CDR37) | on page 172 |
| 198 | Channel 38 Data Register (CDR38) | on page 172 |
| 19C | Channel 39 Data Register (CDR39) | on page 172 |
| 1A0 | Channel 40 Data Register (CDR40) | on page 172 |
| 1A4 | Channel 41 Data Register (CDR41) | on page 172 |
| 1A8 | Channel 42 Data Register (CDR42) | on page 172 |
| 1AC | Channel 43 Data Register (CDR43) | on page 172 |
| 1B0 | Channel 44 Data Register (CDR44) | on page 172 |
| 1B4 | Channel 45 Data Register (CDR45) | on page 172 |
| 1B8 | Channel 46 Data Register (CDR46) | on page 172 |
| 1BC | Channel 47 Data Register (CDR47) | on page 172 |
| 1C0 | Channel 48 Data Register (CDR48) | on page 172 |
| 1C4 | Channel 49 Data Register (CDR49) | on page 172 |
| 1C8 | Channel 50 Data Register (CDR50) | on page 172 |
| 1CC | Channel 51 Data Register (CDR51) | on page 172 |
| 1D0 .. 1FC | Reserved | — |
| 200 | Channel 64 Data Register (CDR64) | on page 172 |

**Table 5-1. ADC digital registers (continued)**

| Address offset (hex.) | Register name | Location |
|---|---|---|
| 204 | Channel 65 Data Register (CDR65) | on page 172 |
| 208 | Channel 66 Data Register (CDR66) | on page 172 |
| 20C | Channel 67 Data Register (CDR67) | on page 172 |
| 210 | Channel 68 Data Register (CDR68) | on page 172 |
| 214 | Channel 69 Data Register (CDR69) | on page 172 |
| 218 | Channel 70 Data Register (CDR70) | on page 172 |
| 21C | Channel 71 Data Register (CDR71) | on page 172 |
| 220 .. 2FC | Reserved | — |

**Table 5-2. Bit access descriptions**

| Access type | Description |
|---|---|
| read/write (rw) | Software can read and write to these bits. |
| read-only (r) | Software can only read these bits. |
| write-only (w) | Software can only write to these bits. |
| write 1 to clear (w1c) | Software can clear bits by writing ‘1’. |

## 5.3.2 Control logic registers

### 5.3.2.1 Main Configuration Register (MCR)

The Main Configuration Register (MCR) provides configuration settings for the ADC.

Address: Base + 0x0000 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OWREN | WLSIDE | MODE | EDGLEV | TRGEN | EDGE | 0 | NSTART | 0 | JTRGEN | JEDGE | JSTART | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ADCLK SEL | ABORT CHAIN | ABORT | ACKO | 0 | 0 | 0 | 0 | PWDN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 5-2. Main Configuration Register (MCR)**

**Table 5-3. Main Configuration Register (MCR) field descriptions**

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>OWREN: Overwrite enable<br>This bit enables or disables the functionality to overwrite unread converted data.<br>0 Prevents overwrite of unread converted data; new result is discarded.<br>1 Enables converted data to be overwritten by a new conversion.</td></tr>
<tr><td>1</td><td>WLSIDE: Write left/right-aligned<br>0 The conversion data is written right-aligned.<br>1 Data is left-aligned (from 15 to (15 – resolution + 1)).</td></tr>
<tr><td>2</td><td>MODE: One Shot/Scan<br>0 One Shot Mode—Configures the normal conversion of one chain.<br>1 Scan Mode—Configures continuous chain conversion mode; when the programmed chain conversion is finished it restarts immediately.</td></tr>
<tr><td>3</td><td>EDGLEV: Edge or level selection for external start trigger<br>0 Edge configuration for external trigger usage.<br>1 Level configuration for external trigger usage.</td></tr>
<tr><td>4</td><td>TRGEN: External trigger enable. This bit must be set to use external triggering to start a conversion.<br>0 An external trigger cannot be used to start a conversion.<br>1 An external trigger can start a conversion.</td></tr>
<tr><td>5</td><td>Start trigger edge/ level detection. The following table shows the interaction between the EDGE bit and the TRGEN and EDGLEV bits.<br>Table 0-1
<table>
<tr><th>TRGEN</th><th>EDGLEV</th><th>EDGE</th><th>Trigger Detection</th></tr>
<tr><td>0</td><td>n</td><td>n</td><td>External triggering disabled</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>External trigger on falling edge of trigger</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>External trigger on rising edge of trigger</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>External trigger on low edge of trigger</td></tr>
</table></td></tr>
<tr><td>6</td><td>Reserved<br>Must be kept at 0.</td></tr>
<tr><td>7</td><td>NSTART: Normal Start conversion<br>Setting this bit starts the chain or scan conversion. Resetting this bit during scan mode causes the current chain conversion to finish, then stops the operation.<br>This bit stays high while the conversion is ongoing (or pending during injection mode).<br>0 Causes the current chain conversion to finish and stops the operation.<br>1 Starts the chain or scan conversion.</td></tr>
<tr><td>8</td><td>Reserved<br>Write of any value has no effect, read value is always 0.</td></tr>
<tr><td>9</td><td>JTRGEN: Injection external trigger enable<br>0 External trigger disabled for channel injection (injected conversion cannot be started using an external signal).<br>1 External trigger enabled for channel injection.</td></tr>
</table>

**Table 5-3. Main Configuration Register (MCR) field descriptions (continued)**

| Bit | Description |
|---|---|
| 10 | JEDGE: Injection trigger edge selection<br>Edge selection for external trigger, if JTRGEN = 1.<br>0 Selects falling edge for the external trigger.<br>1 Selects rising edge for the external trigger. |
| 11 | JSTART: Injection start<br>Setting this bit will start the configured injected analog channels to be converted by software. Resetting this bit has no effect, as the injected chain conversion cannot be interrupted. |
| 12:13 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 14 | Reserved<br>Must be kept at 0. |
| 15:22 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 23 | ADCLKSEL: Analog clock frequency selector<br>If this bit is set the AD_clk frequency is equal to ipg_clk frequency. Otherwise, it is half of ipg_clk frequency. This bit can be written in power-down only. |
| 24 | ABORTCHAIN: Abort Chain<br>When this bit is set, the ongoing Chain Conversion is aborted. This bit is reset by hardware as soon as a new conversion is requested.<br>0 Conversion is not affected.<br>1 Aborts the ongoing chain conversion. |
| 25 | ABORT: Abort Conversion<br>When this bit is set, the ongoing conversion is aborted and a new conversion is invoked. This bit is reset by hardware as soon as a new conversion is invoked.<br>0 Conversion is not affected.<br>1 Aborts the ongoing conversion. |
| 26 | ACKO: Auto-clock-off enable<br>If set, this bit enables the Auto clock off feature.<br>0 Auto clock off is disabled.<br>1 Auto clock off is enabled. |
| 27:28 | Reserved<br>Must be kept at 0. |
| 29:30 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 31 | PWDN: Power-down enable<br>When this bit is set, the analog module is requested to enter Power Down mode. When ADC status is PWDN, resetting this bit starts ADC transition to IDLE mode.<br>0 ADC is in normal mode.<br>1 ADC has been requested to power down. |

### 5.3.2.2 Main Status Register (MSR)

The Main Status Register (MSR) provides status bits for the ADC.

Address: Base + 0x0004 Access: User read-only

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N START | J ABORT | 0 | 0 | J START | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CHADDR[0:6] | | | | | | | 0 | 0 | 0 | ACK0 | 0 | 0 | ADCSTATUS[0:2] | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 5-3. Main Status Register (MSR)**

**Table 5-4. Main Status Register (MSR) field descriptions**

| Field | Description |
|---|---|
| 0:6 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 7 | NSTART<br>This status bit is used to signal that a Normal conversion is ongoing. |
| 8 | JABORT<br>This status bit is used to signal that an Injected conversion has been aborted. This bit is reset when a new injected conversion starts. |
| 9:10 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 11 | JSTART<br>This status bit is used to signal that an Injected conversion is ongoing. |
| 12:14 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 15 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 16:22 | CHADDR[0:6]: Channel under measure address<br>This status bit is used to signal which channel is under measure. |
| 23:25 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 26 | ACKO: Auto-clock-off enable<br>This status bit is used to signal if the Auto-clock-off feature is on. |

**Table 5-4. Main Status Register (MSR) field descriptions (continued)**

| Field | Description |
|---|---|
| 27:28 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 29:31 | ADCSTATUS[0:2]<br>The value of this parameter depends on ADC status:<br>000 IDLE<br>001 Power-down<br>010 Wait state<br>011 —<br>100 Sample<br>101 —<br>110 Conversion<br>111 — |

## 5.3.3 Interrupt registers

### 5.3.3.1 Interrupt Status Register (ISR)

The Interrupt Status Register (ISR) contains interrupt status bits for the ADC.

Address: Base + 0x0010 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | JEOC | JECH | EOC | ECH |
| W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-4. Interrupt Status Register (ISR)**

**Table 5-5. Interrupt Status Register (ISR) field descriptions**

| Field | Description |
|---|---|
| 0:24 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 25:26 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 27 | Reserved<br>Write of any value has no effect, read value is always 0 |
| 28 | End of Injected Channel Conversion interrupt (JEOC) flag. It is the interrupt of the digital end of conversion for the injected channel; active when set. When this bit is set, a JEOC interrupt has occurred. |

**Table 5-5. Interrupt Status Register (ISR) field descriptions (continued)**

| Field | Description |
|---|---|
| 29 | End of Injected Chain Conversion interrupt (JECH) flag. It is the interrupt of the digital end of chain conversion for the injected channel; active when set. When this bit is set, a JECH interrupt has occurred. |
| 30 | End of Channel Conversion interrupt (EOC) flag. It is the interrupt of the digital end of conversion. When this bit is set, an EOC interrupt has occurred. |
| 31 | End of Chain Conversion interrupt (ECH) flag. It is the interrupt of the digital end of chain conversion. When this bit is set, an ECH interrupt has occurred. |

### 5.3.3.2 Channel Pending Registers (CEOCFR[1..2])

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-1.

CEOCFR1 = End of conversion pending interrupt for channel 32 to 63 (extended internal channels)

CEOCFR2 = End of conversion pending interrupt for channel 64 to 95 (external channels)

Address: Base + 0x0018 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EOC_CH63 | EOC_CH62 | EOC_CH61 | EOC_CH60 | EOC_CH59 | EOC_CH58 | EOC_CH57 | EOC_CH56 | EOC_CH55 | EOC_CH54 | EOC_CH53 | EOC_CH52 | EOC_CH51 | EOC_CH50 | EOC_CH49 | EOC_CH48 |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EOC_CH47 | EOC_CH46 | EOC_CH43 | EOC_CH44 | EOC_CH43 | EOC_CH42 | EOC_CH41 | EOC_CH40 | EOC_CH39 | EOC_CH38 | EOC_CH37 | EOC_CH36 | EOC_CH35 | EOC_CH34 | EOC_CH33 | EOC_CH32 |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-5. Channel Pending Register 1 (CEOCFR1)**

Address: Base + 0x001C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EOC_CH95 | EOC_CH94 | EOC_CH93 | EOC_CH92 | EOC_CH91 | EOC_CH90 | EOC_CH89 | EOC_CH88 | EOC_CH87 | EOC_CH86 | EOC_CH85 | EOC_CH84 | EOC_CH83 | EOC_CH82 | EOC_CH81 | EOC_CH80 |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EOC_CH79 | EOC_CH78 | EOC_CH77 | EOC_CH76 | EOC_CH75 | EOC_CH74 | EOC_CH73 | EOC_CH72 | EOC_CH71 | EOC_CH70 | EOC_CH69 | EOC_CH68 | EOC_CH67 | EOC_CH66 | EOC_CH65 | EOC_CH64 |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-6. Channel Pending Register 2 (CEOCFR2)**

**Table 5-6. Channel Pending Registers (CEOCFR[1..2]) field descriptions**

| Field | Description |
|---|---|
| 31 | EOC_CH0<br>When set, the measure of channel 0 is completed. |
| n | EOC_CHn<br>When set, the measure of channel n is completed. |

### 5.3.3.3 Interrupt Mask Register (IMR)

The Interrupt Mask Register (IMR) contains the interrupt enable bits for the ADC.

Address: Base + 0x0020 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | MSK JEOC | MSK JECH | MSK EOC | MSK ECH |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-7. Interrupt Mask Register (IMR)**

**Table 5-7. Interrupt Mask Register (IMR) field descriptions**

| Field | Description |
|---|---|
| 0:24 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 25:26 | Reserved<br>Must be kept at 0. |
| 27 | Reserved<br>Must be kept at 0. |
| 28 | MSKJEOC: Mask bit for JEOC<br>When set, the JEOC interrupt is enabled. |
| 29 | MSKJECH: Mask bit for JECH<br>When set, the JECH interrupt is enabled. |
| 30 | MSKEOC: Mask bit for EOC<br>When set, the EOC interrupt is enabled. |
| 31 | MSKECH: Mask bit for ECH<br>When set, the ECH interrupt is enabled. |

### 5.3.3.4 Channel Interrupt Mask Register (CIMR[1..2])

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-1.

CIMR1 = Enable bits for channels 32 to 63 (extended internal channels)

CIMR2 = Enable bits for channels 64 to 95 (external channels)

Address: Base + 0x0028 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CIM 63 | CIM 62 | CIM 61 | CIM 60 | CIM 59 | CIM 58 | CIM 57 | CIM 56 | CIM 55 | CIM 54 | CIM 53 | CIM 52 | CIM 51 | CIM 50 | CIM 49 | CIM 48 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CIM 47 | CIM 46 | CIM 43 | CIM 44 | CIM 43 | CIM 42 | CIM 41 | CIM 40 | CIM 39 | CIM 38 | CIM 37 | CIM 36 | CIM 35 | CIM 34 | CIM 33 | CIM 32 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-8. Channel Interrupt Mask Register 1 (CIMR1)**

Address: Base + 0x002C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CIM 95 | CIM 94 | CIM 93 | CIM 92 | CIM 91 | CIM 90 | CIM 89 | CIM 88 | CIM 87 | CIM 86 | CIM 85 | CIM 84 | CIM 83 | CIM 82 | CIM 81 | CIM 80 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CIM 79 | CIM 78 | CIM 77 | CIM 76 | CIM 75 | CIM 74 | CIM 73 | CIM 72 | CIM 71 | CIM 70 | CIM 69 | CIM 68 | CIM 67 | CIM 66 | CIM 65 | CIM 64 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-9. Channel Interrupt Mask Register 2 (CIMR2)**

**Table 5-8. Channel Interrupt Mask Register (CIMR[1..2]) field descriptions**

| Field | Description |
|---|---|
| 31 | CIM0: Interrupt enable<br>When set (CIM0 = 1), interrupt for channel 0 is enabled. |
| n | CIMn: Interrupt enable<br>When set (CIMn = 1), interrupt for channel n is enabled. |

### 5.3.3.5 Watchdog Threshold Interrupt Status Register (WTISR)

Address: Base + 0x0030 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | WDG 3H | WDG 2H | WDG 1H | WDG 0H | WDG 3L | WDG 2L | WDG 1L | WDG 0L |
| W | | | | | | | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-10. Watchdog Threshold Interrupt Status Register (WTISR)**

**Table 5-9. Watchdog Threshold Interrupt Status Register (WTISR) field descriptions**

| Field Des | cription |
|---|---|
| 0:23 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 24:27 | WDGxH [x = 0..3]<br>This corresponds to the status flag generated on the converted value being higher than the programmed higher threshold. |
| 28:31 | WDGxL [x = 0..3]<br>This corresponds to the status flag generated on the converted value being lower than the programmed lower threshold. |

### 5.3.3.6 Watchdog Threshold Interrupt Mask Register (WTIMR)

Reset value: 0x0000_0000

Address: Base + 0x0034 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | MSK WDG 3H | MSK WDG 2H | MSK WDG 1H | MSK WDG 0H | MSK WDG 3L | MSK WDG 2L | MSK WDG 1L | MSK WDG 0L |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-11. Watchdog Threshold Interrupt Mask Register (WTIMR)**

**Table 5-10. Watchdog Threshold Interrupt Mask Register (WTIMR) field descriptions**

| Field Des | cription |
|---|---|
| 0:23 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 24:27 | MSKWDGxH [x = 0..3]<br>This corresponds to the mask bit for the interrupt generated on the converted value being higher than the programmed higher threshold. When set the interrupt is enabled. |
| 28:31 | MSKWDGxL [x = 0..3]<br>This corresponds to the mask bit for the interrupt generated on the converted value being lower than the programmed lower threshold. When set the interrupt is enabled. |

## 5.3.4 DMA registers

### 5.3.4.1 DMA Enable Register (DMAE)

The DMA Enable (DMAE) register sets up the DMA for use with the ADC.

Address: Base + 0x0040 Access: User read/write

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

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

**Figure 5-12. DMA Enable Register (DMAE)**

**Table 5-11. DMA Enable Register (DMAE) field descriptions**

| Field | Description |
|---|---|
| 0:29 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 30 | DCLR: DMA clear sequence enable<br>0 DMA request cleared by Acknowledge from DMA controller<br>1 DMA request cleared on read of data registers |
| 31 | DMAEN: DMA global enable<br>0 DMA feature is disabled.<br>1 DMA feature is enabled. |

### 5.3.4.2 DMA Channel Select Register (DMAR[1..2])

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-1.

DMAR1 = Enable bits for channels 32 to 63 (extended internal channels)

DMAR2 = Enable bits for channels 64 to 95 (external channels)

Reset value: 0x0000_0000

Address: Base + 0x0048 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DMA 63 | DMA 62 | DMA 61 | DMA 60 | DMA 59 | DMA 58 | DMA 57 | DMA 56 | DMA 55 | DMA 54 | DMA 53 | DMA 52 | DMA 51 | DMA 50 | DMA 49 | DMA 48 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DMA 47 | DMA 46 | DMA 43 | DMA 44 | DMA 43 | DMA 42 | DMA 41 | DMA 40 | DMA 39 | DMA 38 | DMA 37 | DMA 36 | DMA 35 | DMA 34 | DMA 33 | DMA 32 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-13. DMA Channel Select Register 1 (DMAR1)**

Address: Base + 0x004C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DMA 95 | DMA 94 | DMA 93 | DMA 92 | DMA 91 | DMA 90 | DMA 89 | DMA 88 | DMA 87 | DMA 86 | DMA 85 | DMA 84 | DMA 83 | DMA 82 | DMA 81 | DMA 80 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DMA 79 | DMA 78 | DMA 77 | DMA 76 | DMA 75 | DMA 74 | DMA 73 | DMA 72 | DMA 71 | DMA 70 | DMA 69 | DMA 68 | DMA 67 | DMA 66 | DMA 65 | DMA 64 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-14. DMA Channel Select Register 2 (DMAR2)**

**Table 5-12. DMA Channel Select Register (DMAR[1..2]) field descriptions**

| Field Des | cription |
|---|---|
| 31 | DMA0: DMA enable<br>When set (DMA0 = 1), channel 0 is enabled to transfer data in DMA mode. |
| n | DMAn: DMA enable<br>When set (DMAn = 1), channel n is enabled to transfer data in DMA mode. |

## 5.3.5 Threshold registers

### 5.3.5.1 Introduction

These four registers are used to store the user programmable lower and upper thresholds' values. The inverter bit and the mask bit for mask the interrupt are stored in the TRC registers.

### 5.3.5.2 Threshold Control Register (TRCx, x = [0..3])

Reset value: 0x0000_0000

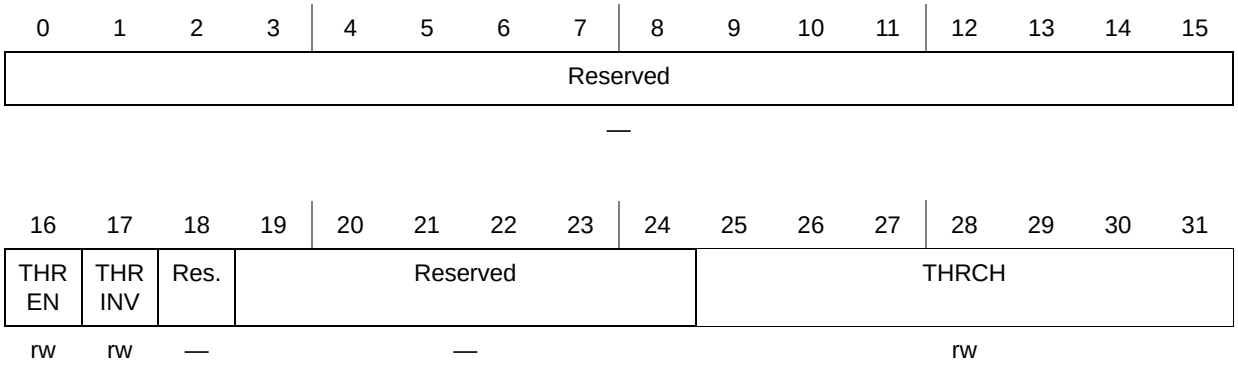



**Figure 5-15. Threshold Control Register (TRCx, x = [0..3])**

**Table 5-13. Threshold Control Register (TRCx, x = [0..3]) field descriptions**

| Field Des | cription |
|---|---|
| 0:15 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 16 | THREN: Threshold enable<br>When set, this bit enables the threshold detection feature for the selected channel. |
| 17 | THRINV: Invert the output pin<br>Setting this bit inverts the behavior of the threshold output pin. |
| 18 | Reserved<br>Must be kept at 0. |
| 19:24 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 25:31 | THRCH: Choose the channel for threshold comparison. |

### 5.3.5.3 Threshold Register (THRHLR[0:3])

The four THRHLR*n* registers are used to store the user-programmable thresholds' 10-bit values.

Address: Base + 0x0060 (THRHLR0)
Base + 0x0064 (THRHLR1)
Base + 0x0068 (THRHLR2)
Base + 0x006C (THRHLR3)

Access: User read/write

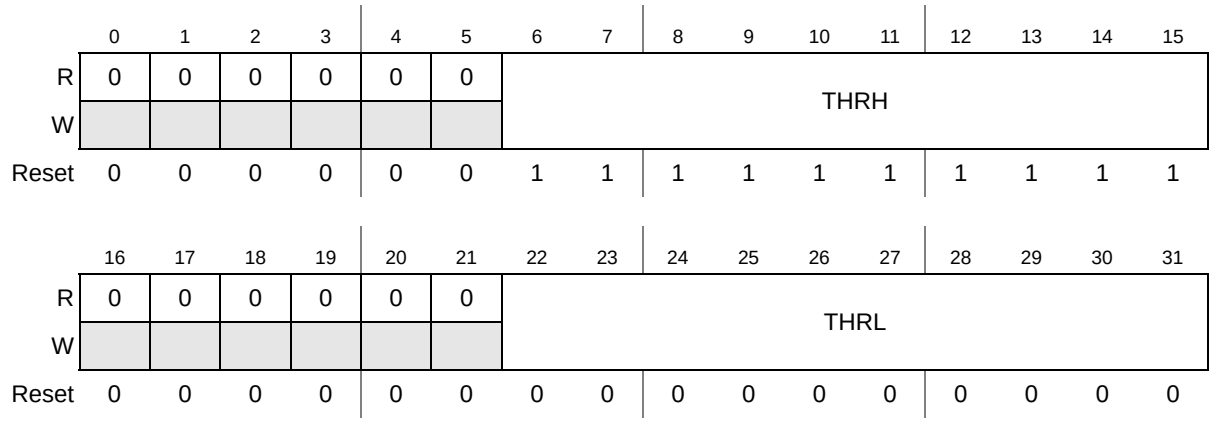



**Figure 5-16. Threshold Register (THRHLR[0:3])**

**Table 5-14. Threshold Register (THRHLR[0:3]) field descriptions**

| Field De | scription |
|---|---|
| 0:5 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 6:15 | THRH: High threshold value for channel *n*. |
| 16:21 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 22:31 | THRL: Low threshold value for channel *n*. |

## 5.3.6 Conversion timing registers CTR[1..2]

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-1.

CTR1 = associated to extended internal channels (from 32 to 63)

CTR2 = associated to external channels (from 64 to 95)

Address: Base + 0x0098 (CTR1)
Base + 0x009C (CTR2)

Access: User read/write

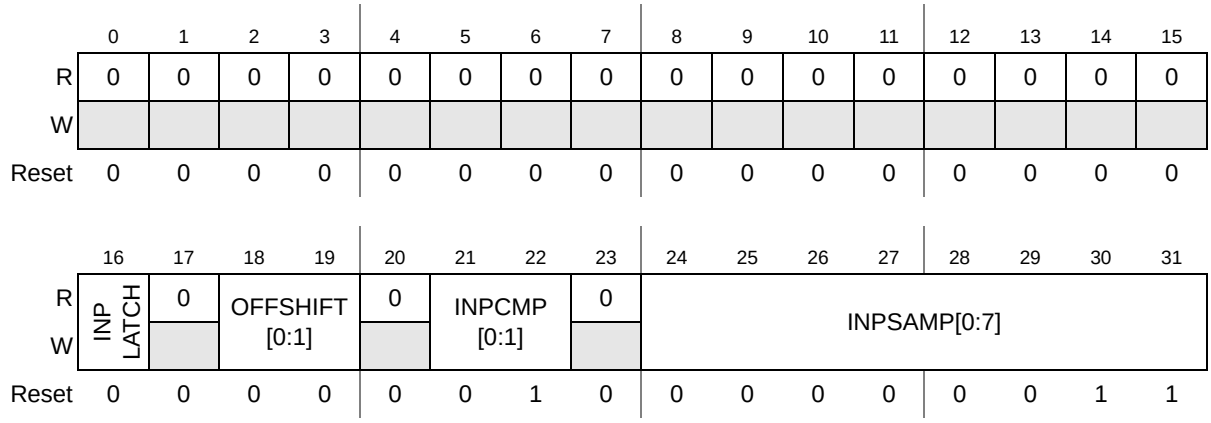



**Figure 5-17. Conversion timing registers CTR[1..2]**

**Table 5-15. Conversion timing registers CTR[1..2] field descriptions**

| Field De | scription |
|---|---|
| 0:15 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 16 | INPLATCH<br>Configuration bit for latching phase duration |
| 17 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 18:19 | OFFSHIFT[0:1]<br>Configuration for offset shift characteristic<br>00 No shift (that is the transition between codes 000h and 001h) is reached when the $A_{VIN}$ (analog input voltage) is equal to 1 LSB.<br>01 Transition between code 000h and 001h is reached when the $A_{VIN}$ is equal to1/2 LSB<br>10 Transition between code 00h and 001h is reached when the $A_{VIN}$ is equal to 0<br>11 Not used |
| 20 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 21:22 | INPCMP[0:1]<br>Configuration bits for comparison phase duration |
| 23 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 24:31 | INPSAMP[0:7]<br>Configuration bits for sampling phase duration |

## 5.3.7 Mask registers

### 5.3.7.1 Introduction

These registers are used to program which of the 96 input channels must be converted during Normal and Injected conversion.

### 5.3.7.2 Normal Conversion Mask Registers (NCMR[1..2])

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-1.

NCMR1 = Enable bits of normal sampling for channel 32 to 63 (extended internal channels)

Reset value: 0x0000_0000

Address: Base + 0x00A8 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH63 | CH62 | CH61 | CH60 | CH59 | CH58 | CH57 | CH56 | CH55 | CH54 | CH53 | CH52 | CH51 | CH50 | CH49 | CH48 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH47 | CH46 | CH45 | CH44 | CH43 | CH42 | CH41 | CH40 | CH39 | CH38 | CH37 | CH36 | CH35 | CH34 | CH33 | CH32 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-18. Normal Conversion Mask Register 1 (NCMR1)**

Address: Base + 0x00AC Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH95 | CH94 | CH93 | CH92 | CH91 | CH90 | CH89 | CH88 | CH87 | CH86 | CH85 | CH84 | CH83 | CH82 | CH81 | CH80 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH79 | CH78 | CH77 | CH76 | CH75 | CH74 | CH73 | CH72 | CH71 | CH70 | CH69 | CH68 | CH67 | CH66 | CH65 | CH64 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-19. Normal Conversion Mask Register 2 (NCMR2)**

**Table 5-16. Normal Conversion Mask Registers (NCMR[1..2]) field descriptions**

| Field Des | cription |
|---|---|
| 31 | CH0: Sampling enable<br>When set Sampling is enabled for channel 0. |
| n | CHn: Sampling enable<br>When set Sampling is enabled for channel n. |

### 5.3.7.3 Injected Conversion Mask Registers (JCMR[1..2])

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-1.

JCMR1 = Enable bits of injected sampling for channel 32 to 63 (extended internal channels)

JCMR2 = Enable bits of injected sampling for channel 64 to 95 (external channels)

Reset value: 0x0000_0000

Address: Base + 0x00B8 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH63 | CH62 | CH61 | CH60 | CH59 | CH58 | CH57 | CH56 | CH55 | CH54 | CH53 | CH52 | CH51 | CH50 | CH49 | CH48 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH47 | CH46 | CH45 | CH44 | CH43 | CH42 | CH41 | CH40 | CH39 | CH38 | CH37 | CH36 | CH35 | CH34 | CH33 | CH32 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-20. Injected Conversion Mask Register 1 (JCMR1)**

Address: Base + 0x00BC Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH95 | CH94 | CH93 | CH92 | CH91 | CH90 | CH89 | CH88 | CH87 | CH86 | CH85 | CH84 | CH83 | CH82 | CH81 | CH80 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH79 | CH78 | CH77 | CH76 | CH75 | CH74 | CH73 | CH72 | CH71 | CH70 | CH69 | CH68 | CH67 | CH66 | CH65 | CH64 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-21. Injected Conversion Mask Register 2 (JCMR2)**

**Table 5-17. Injected Conversion Mask Registers (JCMR[1..2]) field descriptions**

| Field Des | cription |
|---|---|
| 31 | CH0: Sampling enable<br>When set, sampling is enabled for channel 0. |
| n | CHn: Sampling enable<br>When set, sampling is enabled for channel n. |

## 5.3.8 Delay registers

### 5.3.8.1 Decode Signals Delay Register (DSDR)

Reset value: 0x0000_0000

Address: Base + 0x00C4 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DSD[0:7] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-22. Decode Signals Delay Register (DSDR)**

**Table 5-18. Decode Signals Delay Register (DSDR) field descriptions**

| Field Des | cription |
|---|---|
| 0:23 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 24:31 | DSD[0:7]: The delay between the external decode signals and the start of the sampling phase. It is used to take into account the settling time of the external multiplexer.<br>The decode signal delay is calculated as: DSD × 1/frequency of ADC clock.<br>**Note:** When ADC clock = Peripheral Clock/2, the DSD should be incremented by 2, to see an additional ADC clock cycle delay on the decode signal.<br>For example:<br>DSD = 0; 0 ADC clock cycle delay<br>DSD = 2; 1 ADC clock cycle delay<br>DSD = 4; 2 ADC clock cycles delay |

**NOTE**

For the case when ADC clock = Peripheral Clock/2, the ADC Multiplexer Decode Signal Delay Register (ADC_DSDR) should be incremented by 2, to see an additional ADC clock cycle delay on the decode signal. For example:

– ADC_DSDR = 0 - 0 ADC clock cycle delay
– ADC_DSDR = 2 - 1 ADC clock cycle delay
– ADC_DSDR = 4 - 2 ADC clock cycles delay"

### 5.3.8.2 Power-down Exit Delay Register (PDEDR)

As a programming guideline:

1. Reset PWDN bit

2. Wait for the power up time suggested by ADC power up parameter in the DS (tADC_PU) before any conversion is initiated.

Reset value: 0x0000_0000

Address: Base + 0x00C8 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDED[0:7] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-23. Power-down Exit Delay Register (PDEDR)**

**Table 5-19. Power-down Exit Delay Register (PDEDR) field descriptions**

| Field Des | cription |
|---|---|
| 0:23 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 24:31 | PDED[0:7]: The delay between the power-down bit reset and the start of conversion<br>The power down delay is calculated as: PDED x 1/frequency of ADC clock |

## 5.3.9 Data registers

### 5.3.9.1 Introduction

ADC conversion results are stored in data registers. There is one register per channel.

The 0 to 31 range shown below is the maximum range for the channel type. For the exact number of available channels, please refer to Table 5-1.

CDR[32..63] = Extended internal channels

CDR[64..95] = External channels

Each data register also gives information regarding the corresponding result as described below.

### 5.3.9.2 Channel Data Register (CDR*n*)

Address: See Table 5-1 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | VALID | OVERW | RESULT[0:1] | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | CDATA[0:9] | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5-24. Channel Data Register (CDR*n*)**

**Table 5-20. CDR*n* field descriptions**

| Field | Description |
|---|---|
| 0:11 | Reserved<br>Write of any value has no effect, read value is always 0. |
| 12 | VALID<br>Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read. |
| 13 | OVERW: Overwrite data<br>This bit signals that the previous converted data has been overwritten by a new conversion. This functionality depends on the value of MCR[OWREN]:<br>– When OWREN = 0, then OVERW is frozen to 0 and CDATA field is protected against being overwritten until being read.<br>– When OWREN = 1, then OVERW flags the CDATA field overwrite status.<br>0 Converted data has not been overwritten<br>1 Previous converted data has been overwritten before having been read |

Table 5-20. CDR*n* field descriptions (continued)

| Field Des | cription |
|---|---|
| 14:15 | RESULT[0:1]<br>This bit reflects the mode of conversion for the corresponding channel.<br>00 Data is a result of Normal conversion mode.<br>01 Data is a result of Injected conversion mode.<br>10 Reserved.<br>11 Reserved. |
| 16:21 | Reserved<br>Write of any value has no effect, read value is always 0 |
| 22:31 | CDATA[0:9]: Channel 0-95 converted data |

## 5.4 Functional description

### 5.4.1 Analog channel conversion

Two conversion modes are available within the ADCDig:

- Normal conversion
- Injected conversion

#### 5.4.1.1 Normal conversion

This is the normal conversion that the user programs by configuring the normal conversion mask registers (NCMR). Each channel can be individually enabled by setting ‘1’ in the corresponding field of NCMR registers. Mask registers must be programmed before starting the conversion and cannot be changed until the conversion of all the selected channels ends (NSTART bit in the Main Status Register (MSR) is reset).

#### 5.4.1.2 Start of normal conversion

By programming the configuration bits in the Main Configuration Register (MCR), the normal conversion can be started in two ways:

- By software (TRGEN reset)—If the external trigger enable bit is reset, the conversion chain starts when the NSTART bit in the MCR is set.
- By trigger (TRGEN set)—An on-chip internal signal triggers an ADC conversion. The settings in the MCR select how conversions are triggered based on these internal signals:
  - If the EDGLEV (edge/level selection) bit in the MCR is cleared, then a rising/falling edge (depending on the EDGE bit in MCR) detected in the signal sets the NSTART bit in the MSR and starts the programmed conversion. EDGE = 0 selects a falling edge. EDGE = 1 selects a rising edge.
  - If the EDGLEV bit in the MCR is set, the conversion is started if and only if the NSTART bit in the MCR is set and the programmed level on the trigger signal is detected. The level is selected using the EDGE bit in the MCR. EDGE = 0 means that the start of conversion is

enabled if the signal is low. If EDGE = 1, the start of conversion is enabled when the signal is high.

**Table 5-21. Configurations for starting normal conversion**

| Type of conversion start | MCR MS | | | | R | Result |
|---|---|---|---|---|---|---|
| | TRGEN | NSTART | EDGLEV | EDGE | NSTART | |
| Software | 0 | 1 | — | — | 1 | Conversion chain starts |
| Trigger | 1 | — | 0 | 0 | 1 | A falling edge detected in a trigger signal sets the NSTART bit in the MSR and starts the programmed conversion. |
| | | | | 1 | | A rising edge detected in a trigger signal sets the NSTART bit in the MSR and starts the programmed conversion. |
| Trigger | 1 | 1 | 1 | 0 | 1 | The conversion is started if the programmed level on the trigger signal is detected: the start of conversion is enabled if the external pin is low. |
| | | | | 1 | 1 | The conversion is started if the programmed level on the trigger signal is detected: the start of conversion is enabled if the external pin is high. |

The NSTART status bit in the MSR is automatically set when the normal conversion starts. At the same time the NSTART bit in the MCR is reset, allowing the software to program a new start of conversion. In that case the new requested conversion starts after the running conversion is completed.

If the content of all the normal conversion mask registers is zero (that is, no channel is selected) the conversion operation is considered completed and the interrupt ECH (see further) is immediately issued after the start of conversion.

### 5.4.1.3 Normal conversion operating modes

Two operating modes are available for the normal conversion:

- One Shot
- Scan

To enter one of these modes, it is necessary to program the MODE bit in the MCR. The first phase of the conversion process involves sampling the analog channel and the next phase involves the conversion phase when the sampled analog value is converted to digital as shown in Figure 5-25.

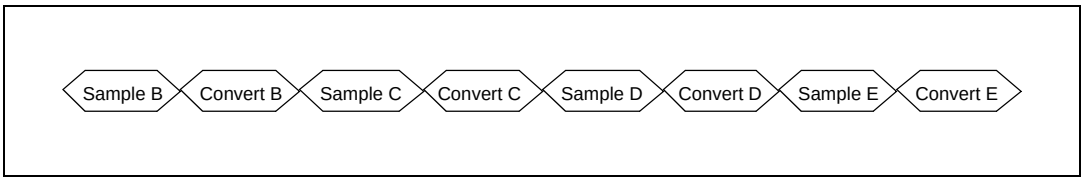



**Figure 5-25. Normal conversion flow**

In **One Shot Mode** (MODE = 0) a sequential conversion specified in the NCMR registers is performed only once. At the end of each conversion, the digital result of the conversion is stored in the corresponding data register.

**Example 5-1. One Shot Mode (MODE = 0)**

```
Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in
the One Shot Mode. MODE = 0 is set for One Shot mode. Conversion starts from the channel B
followed by conversion of channels D-E. At the end of conversion of channel E the scanning of
channels stops.
```

The NSTART status bit in the MSR is automatically set when the Normal conversion starts. At the same time the NSTART bit in the MCR is reset, allowing the software to program a new start of conversion. In that case the new requested conversion starts after the running conversion is completed.

In **Scan Mode** (MODE = 1), a sequential conversion of N channels specified in the NCMR registers is continuously performed. As in the previous case, at the end of each conversion the digital result of the conversion is stored into the corresponding data register.

The NSTART status bit in the MSR is automatically set when the Normal conversion starts. Unlike One Shot Mode, the NSTART bit in the MCR is not reset. It can be reset by software when the user needs to stop scan mode. In that case, the ADC completes the current scan conversion and, after the last conversion, also resets the NSTART bit in the MSR.

**Example 5-2. Scan Mode (MODE = 1)**

```
Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in
the Scan Mode. MODE = 1 is set for Scan Mode. Conversion starts from the channel B followed by
conversion of the channels D-E. At the end of conversion of channel E the scanning of channel
B starts followed by conversion of the channels D-E. This sequence repeats itself till the
NSTART bit in the MCR is reset by software.
```

If the conversion is started by an external trigger and EDGLEV is ‘0’, the NSTART bit in the MCR is not set. As a consequence, once started the only way to stop scan mode conversion is to set the MODE bit to ‘0’.

At the end of each conversion an End Of Conversion interrupt is issued (if enabled by the corresponding mask bit) and at the end of the conversion sequence an End Of Chain interrupt is issued (if enabled by the corresponding mask bit).

#### 5.4.1.4 Injected channel conversion

A conversion chain can be injected into the ongoing Normal conversion by configuring the Injected Conversion Mask Registers (JCMR). As Normal conversion, each channel can be individually selected. This injected conversion can only occur in One Shot mode and interrupts the normal conversion. When an injected conversion is inserted, ongoing channel conversion is aborted and the injected channel request is processed. After the last channel in the injected chain is converted, normal conversion resumes from the channel at which the normal conversion was stopped as shown in Figure 5-26.

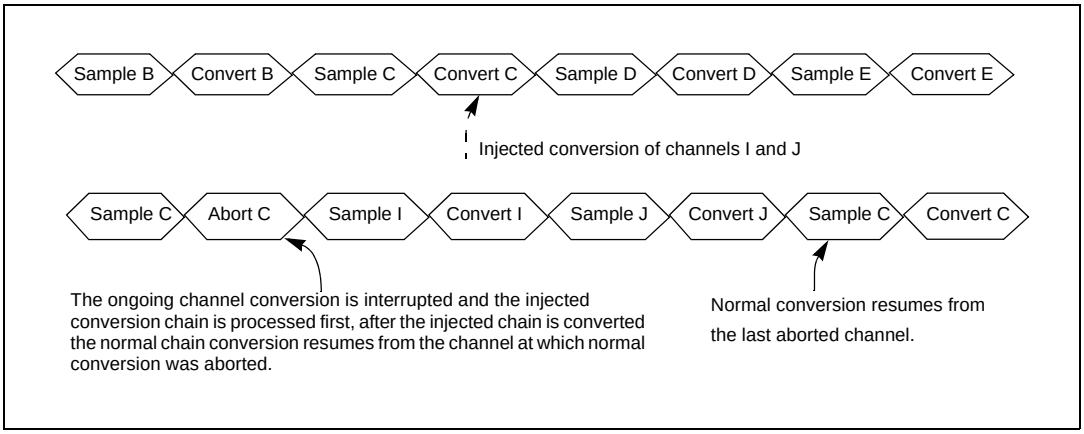



**Figure 5-26. Injected sample/conversion sequence**

The injected conversion can be started by software setting the JSTART bit in the MCR; the current conversion is suspended and the injected chain is converted. At the end of the chain, the JSTART bit in the MSR is reset and the normal chain conversion is resumed.

The JSTART status bit in the MSR is automatically set when the Injected conversion starts. At the same time the JSTART bit in the MCR is reset, allowing the software to program a new start of conversion. In that case the new requested conversion starts after the running injected conversion is completed.

At the end of each injected conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit) and at the end of the sequence an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit).

If the content of all the injected conversion mask registers is zero (that is, no channel is selected) the interrupt JECH is immediately issued after the start of conversion.

Once started, injected chain conversion cannot be interrupted by any other conversion type (it can, however, be aborted; see Section 5.4.1.5, “Abort conversion”).

### 5.4.1.5 Abort conversion

Two different abort functions are provided.

- The user can abort the ongoing conversion by setting the ABORT bit in the MCR. The current conversion is aborted and the conversion of the next channel of the chain is immediately started (generating a new start pulse to the Analog ADC). In the case of an abort operation, the NSTART/JSTART bit remains set and the ABORT bit is reset after the conversion of the next channel starts. The EOC corresponding to the aborted channel is not generated. This behavior is true for normal or triggered/Injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported generating an ECH interrupt.
- It is also possible to abort the current chain conversion by setting the ABORTCHAIN bit in the MCR. In that case the behavior of the ADC depends on the MODE bit. In fact, if scan mode is disabled, the NSTART bit is automatically reset together with the ABORTCHAIN bit. Otherwise, if the MODE bit is set to ‘1’, a new chain conversion is started. The EOC of the current aborted conversion is not generated but an ECH interrupt is generated to signal the end of the chain.

When a chain conversion abort is requested (ABORTCHAIN bit is set) while an injected conversion is running over a suspended Normal conversion, both injected chain and Normal conversion chain are aborted (both the NSTART and JSTART bits are also reset).

**NOTE**

Setting either the ABORT or ABORTCHAIN bits when no conversion is taking place can cause undetermined operation of the next programmed conversion chain.

**NOTE**

The arrival of injected conversion chain will suspend the ongoing normal conversion chain and the normal chain resumes after the injected chain is over.

In real, the current conversion of normal chain is suspended only when it is in the sampling phase. If it is in the conversion phase, the current conversion will be finished and the injected conversion starts after it.

### 5.4.2 Analog clock generator and conversion timings

The clock frequency can be selected by programming the ADCLKSEL bit in the MCR. When this bit is set to '1' the ADC clock has the same frequency as the system clock. Otherwise, the ADC clock is half of the system clock frequency. The ADCLKSEL bit can be written only in power-down mode.

When the internal divider is not enabled (ADCCLKSEL = 1), it is important that the associated clock divider in the clock generation module is '1'. This is needed to ensure 50% clock duty cycle.

The direct clock should basically be used only in low power mode when the device is using only the 16 MHz fast internal RC oscillator, but the conversion still requires a 16 MHz clock (an 8 MHz clock is not fast enough).

In all other cases, the ADC should use the clock divided by two internally.

### 5.4.3 ADC sampling and conversion timing

In order to support different loading and switching times, several different Conversion Timing registers (CTR) are present. There is one register per channel type. INPLATCH and INPCMP configurations are limited when the system clock frequency is greater than 20 MHz.

When a conversion is started, the ADC connects the internal sampling capacitor to the respective analog input pin, allowing the capacitance to charge up to the input voltage value. The time to load the capacitor is referred to as sampling time. After completion of the sampling phase, the evaluation phase starts and all the bits corresponding to the resolution of the ADC are estimated to provide the conversion result.

The conversion times are programmed via the bit fields of the CTR. Bit fields INPLATCH, INPCMP and INPSAMPLE are used to define the total conversion duration ($T_{conv}$) and in particular the partition between sampling phase duration ($T_{sample}$) and total evaluation phase duration ($T_{eval}$).

The sampling phase duration is:

$$T_{sample} = (INPSAMPLE - ndelay) \bullet T_{ck}$$

$$INPSAMPLE \geq 3$$

where ndelay is equal to 0.5 if INPSAMPLE is less than or equal to 06h, otherwise it is 1. INPSAMPLE must be greater than or equal to 3 (hardware requirement).

The total evaluation phase duration is:

$$T_{eval} = 10 \bullet T_{biteval} = 10 \bullet (INPCMP \bullet T_{ck})$$

$$(INPCMP \geq 1) \quad \text{and} \quad (INPLATCH < INPCMP)$$

INPCMP must be greater than or equal to 1 and INPLATCH must be less than INCMP (hardware requirements).

The total conversion duration is (not including external multiplexing):

$$T_{conv} = T_{sample} + T_{eval} + (ndelay \bullet T_{ck})$$

The timings refer to the unit $T_{ck}$, where $f_{ck}$ = (1/2 x ADC peripheral set clock). The maximum clock frequency is specified in Table 5-22.

**NOTE**

At 32 MHz, the minimum sampling time must be at least 180ns.

**Table 5-22. Max AD_clk frequency and related configuration settings**

| INPLATCH | INPCMP | INPSAMPLE | AD_clk $f_{max}$ (MHz) | $T_{sample}$ min. (ns) |
|---|---|---|---|---|
| 0 | 1h | 3h | 20 | 125 |
| 0 | 1h | 4h | 20 + 4% | 168 |
| 1 | 2h | 4h | 20 + 4% | 168 |
| 1 | 2h | 5h | 20 + 4% | 135 |
| 1 | 3h | 6h | 32 + 4% | 132 |
| 1 | 3h | 7h | 40 + 4% | 128 |
| 1 | 3h | 8h | 50 + 4% | 134 |
| 1 | 3h | 9h | 60 + 4% | 128 |

Table 5-23 lists the possible combinations by configuring the AD_clk at 60 MHz.

**Table 5-23. ADC sampling and conversion timing**

| INPLATCH | INPCMP | INPSAMP | $T_{sample}$[1] | $T_{eval}$ | ndelay | $T_{conv}$ |
|---|---|---|---|---|---|---|
| 1 | 11 | 0000 1001 | 8 * Tck | 30 * Tck | 1 * Tck | 39 * Tck[2] |
| 1 | 11 | 0000 1010 | 9 * Tck | 30 * Tck | 1 * Tck | 40 * Tck |
| 1 | 11 | 0000 1011 | 10 * Tck | 30 * Tck | 1 * Tck | 41 * Tck |
| 1 | 11 | 0000 1100 | 11 * Tck | 30 * Tck | 1 * Tck | 42 * Tck |
| 1 | 11 | 0000 1101 | 12 * Tck | 30 * Tck | 1 * Tck | 43 * Tck |
| 1 | 11 | 0000 1110 | 13 * Tck | 30 * Tck | 1 * Tck | 44 * Tck |
| 1 | 11 | 0000 1111 | 14 * Tck | 30 * Tck | 1 * Tck | 45 * Tck |
| ... | ... | ... | ... | ... | ... | ... |
| 1 | 11 | 1111 1100 | 251 * Tck | 30 * Tck | 1 * Tck | 282 * Tck |
| 1 | 11 | 1111 1101 | 252 * Tck | 30 * Tck | 1 * Tck | 283 * Tck |
| 1 | 11 | 1111 1110 | 253 * Tck | 30 * Tck | 1 * Tck | 284 * Tck |
| 1 | 11 | 1111 1111 | 254 * Tck | 30 * Tck | 1 * Tck | 285 * Tck |

1 Represents the number of clock cycles that this operation will last

2 The ADC minimum conversion time at 60 MHz frequency is 39 * Tck; that corresponds to 650 ns.

## 5.4.4 Programmable analog watchdog

### 5.4.4.1 Introduction

The analog watchdogs are used for determining whether the result of a channel conversion lies within a given guarded area (as shown in Figure 5-27) specified by an upper and a lower threshold value named THRH and THRL respectively.

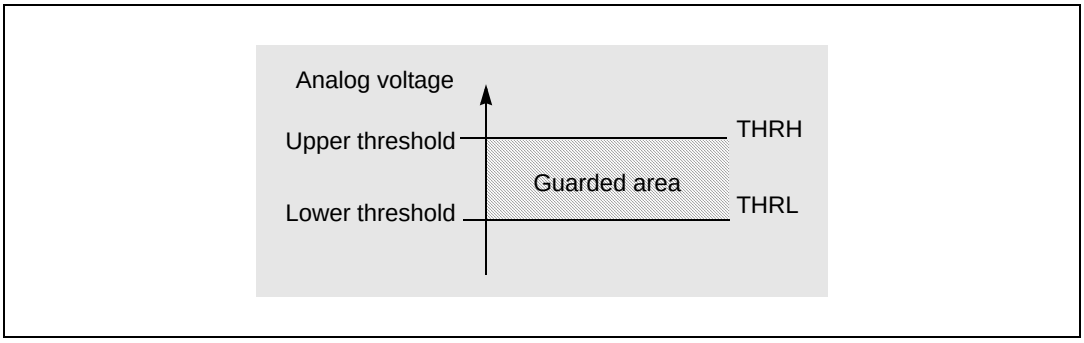



**Figure 5-27. Guarded area**

After the conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside that guarded area then corresponding threshold violation interrupts are generated. The comparison result is stored as WDGxH and WDGxL bits in the WTISR as explained in Table 5-24. Depending on the mask bits MSKWDGxL and MSKWDGxH in the WTIMR, an interrupt is generated on threshold violation.

**Table 5-24. Values of WDGxH and WDGxL fields**

| WDGxH | WDGxL | Converted data |
|---|---|---|
| 1 | 0 | converted data > THRH |
| 0 | 1 | converted data < THRL |
| 0 | 0 | THRL <= converted data <= THRH |

The channel on which the analog watchdog is to be applied is selected by the THRCH field in the TRC registers. The analog watchdog is enabled by setting the corresponding THREN bit in the same register.

The lower and higher threshold values for the analog watchdog are programmed using the registers THRHLR.

For example, if channel number 3 is to be monitored with threshold values in THRHLR1, then the THRCH field is programmed in the TRC1 register to select channel number 3.

A set of threshold registers (THRHLRx and TRCx) can be linked only to a single channel for a particular THRCH value. If another channel is to be monitored with same threshold values, then the THRCH field in the TRCx register has to be programmed again.

NOTE

If the higher threshold for the analog watchdog is programmed lower than the lower threshold and the converted value is less than the lower threshold, then the WDGxL interrupt for the low threshold violation is set, else if the converted value is greater than the lower threshold (consequently also greater than the higher threshold) then the interrupt WDGxH for high threshold violation is set. Thus, the user should avoid that situation as it could lead to misinterpretation of the watchdog interrupts.

### 5.4.5 DMA functionality

A DMA request can be programmed after the conversion of every channel by setting the respective masking bit in the DMAR registers. The DMAR masking registers must be programmed before starting any conversion. There is one DMAR per channel type.

The DMA transfers can be enabled using the DMAEN bit of DMAE register. When the DCLR bit of DMAE register is set then the DMA request is cleared on the reading of the register for which DMA transfer has been enabled.

### 5.4.6 Interrupts

The ADC generates the following maskable interrupt signals:

- ADC_EOC interrupt requests
  - EOC (end of conversion)
  - ECH (end of chain)
  - JEOC (end of injected conversion)

— JECH (end of injected chain)

- WDGxL and WDGxH (watchdog threshold) interrupt requests

Interrupts are generated during the conversion process to signal events such as End Of Conversion as explained in register description for CEOCFR. Two 7-bit registers named CEOCFR (Channel Pending Registers) and IMR (Interrupt Mask Register) are provided in order to check and enable the interrupt request to EIC module.

Interrupts can be individually enabled on a channel by channel base by programming the CIMR (Channel Interrupt Mask Register).

Several Channel Interrupt Pending Registers are also provided in order to signal which of the channels' measurement has been completed.

The analog watchdog interrupts are handled by two 8-bit registers WTISR (Watchdog Threshold Interrupt Status Register) and WTIMR (Watchdog Threshold Interrupt Mask Register) in order to check and enable the interrupt request to the EIC module. The Watchdog interrupt source sets two pending bits WDGxH and WDGxL in the WTISR for each of the four channels being monitored.

The CEOCFR contains the interrupt pending request status. If the user wants to clear a particular interrupt event status, then writing a '1' to the corresponding status bit clears the pending interrupt flag (at this write operation all the other bits of the CEOCFR must be maintained at '0').

### 5.4.7 External decode signals delay

The ADC provides several external decode signals to select which external channel has to be converted. In order to take into account the control switching time of the external analog multiplexer, a Decode Signals Delay register (DSDR) is provided. The delay between the decoding signal selection and the actual start of conversion can be programmed by writing the field DSD[0:7].

### 5.4.8 Power-down mode

The analog part of the ADC can be put in low power mode by setting the PWDN bit in the MCR. After releasing the reset signal the ADC analog module is kept in power-down mode by default, so this state must be exited before starting any operation by resetting the appropriate bit in the MCR.

The power-down mode can be requested at any time by setting the PWDN bit in the MCR. If a conversion is ongoing, the ADC hard macrocell cannot immediately enter the power-down mode. In fact, the ADC enters power-down mode only after completing the ongoing conversion. Otherwise, the ongoing operation should be aborted manually by resetting the NSTART bit and using the ABORTCHAIN bit.

Bit ADCSTATUS[0] in the MSR is set only when ADC enters power-down mode.

After the power-down phase is completed the process ongoing before the power-down phase must be restarted manually (by setting the appropriate START bit).

Resetting PWDN bit and setting NSTART or JSTART bit during the same cycle is forbidden.

### 5.4.9 Auto-clock-off mode

To reduce power consumption during the IDLE mode of operation (without going into power-down mode), an “auto-clock-off” feature can be enabled by setting the ACKO bit in the MCR. When enabled, the analog clock is automatically switched off when no operation is ongoing, that is, no conversion is programmed by the user.

# Chapter 6
# Boot Assist Module (BAM)

This chapter describes the Boot Assist Module (BAM).

## 6.1 Overview

The Boot Assist Module is a block of read-only memory containing VLE code which is executed according to the boot mode of the device.

The BAM allows to download code into internal SRAM through the following serial protocol and execute it afterwards:

- FlexCAN (without autobaud)
- LINFlex (without autobaud)

## 6.2 Features

The BAM provides the following features:

- Locate serial communication interface for downloading application boot code
- Detect application boot code
- MPC5645S in static mode if internal flash is not initialized or invalid
- System can recover from Static mode only by Reset
- Configures single MMU TBL entry to enable access to startup code
- Programmable 64-bit password protection for serial boot mode
- Serial boot loads the application boot code from a FlexCAN or LINFlex bus into internal SRAM
- Censorship protection for internal flash module

## 6.3 Boot modes

The MPC5645S supports the following boot modes:

- Single Chip (SC) - The device boots from the first bootable section[1] of the flash memory main array.
- Serial Boot (SBL) - The device downloads boot code from either LINFlex or FlexCAN interface and then executes it.

If booting is not possible with the selected configuration (e.g., if no Boot ID is found in the selected boot location) then the device enters the static mode.

## 6.4 Memory map

The BAM code resides in a reserved 8 KB ROM mapped from address 0xFFFF_C000.

1. Section with valid boot ID

The RAM location where to download the code can be any 4 byte aligned location starting from the address 0x4000_0100.

## 6.5 Functional description

### 6.5.1 Entering boot modes

The MPC5645S detects the boot mode based on external pins and device status (see Figure 6-1).

To boot either from FlexCAN or LINFlex, the device must be forced into an Alternate Boot Loader Mode via the FAB (Force Alternate Boot Mode) pin which must be asserted before initiating the reset sequence. The type of alternate boot mode is selected according to the ABS (Alternate Boot Selector) pin (see Table 6-1).

**NOTE**

The watchdog (SWT) is disabled at the start of BAM execution. In the case of an unexpected issue during BAM execution the CPU may be stalled and it will be necessary to generate an external reset to recover.

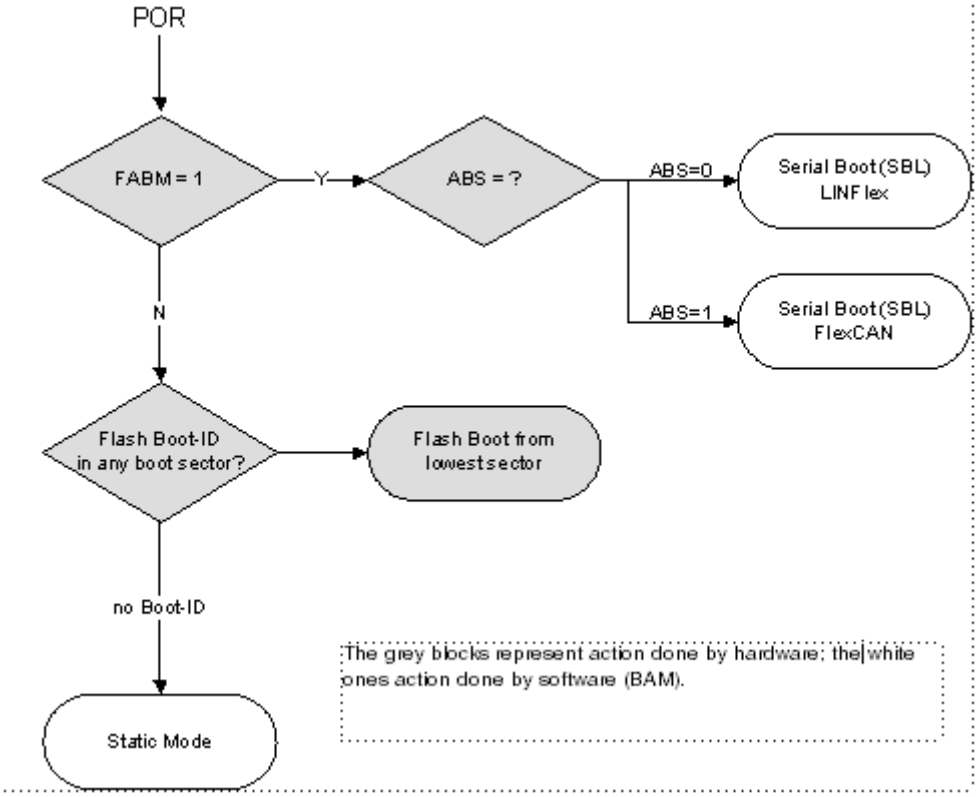



**Figure 6-1. Boot mode selection**

**Table 6-1. Hardware configuration to select boot mode**

| FAB A | BS | Standby-RAM Boot Flag | Boot ID | Boot Mode |
|---|---|---|---|---|
| 1 | 0 | 0 | — | LINFlex |
| 1 | 1 | 0 | — | FlexCAN |
| 0 | - | 0 | valid | SC (Single Chip) |
| 0 | - | 0 | not found | Static Mode |
| - | - | 1 | - | From Backup RAM (0x40000000)[1] |

[1] After the device exits standby, it boots from Backup RAM if the Standby-RAM boot flag is set. If this flag is not set, the device boots internally (SC) or using the BAM depending on the state of the FABM and ABS pins.

## 6.5.2 Reset Configuration Half Word Source (RCHW)

MPC5645S flash memory is partitioned into boot sectors shown in Table 6-3.

Each boot sector contains the Reset Configuration Half-Word (RCHW) at offset 0x00.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | VLE | BOOT_ID | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-2. Reset Configuration Half Word (RCHW)**

**Table 6-2. RCHW field descriptions**

| Field De | scription |
|---|---|
| VLE | Selects the VLE or Book E instruction set at application start address.<br>0 Selects the classic PowerPC instruction set<br>1 Selects the VLE instruction set |
| BOOT_ID | Is valid if its value is 0x5A, then the sector is considered bootable |

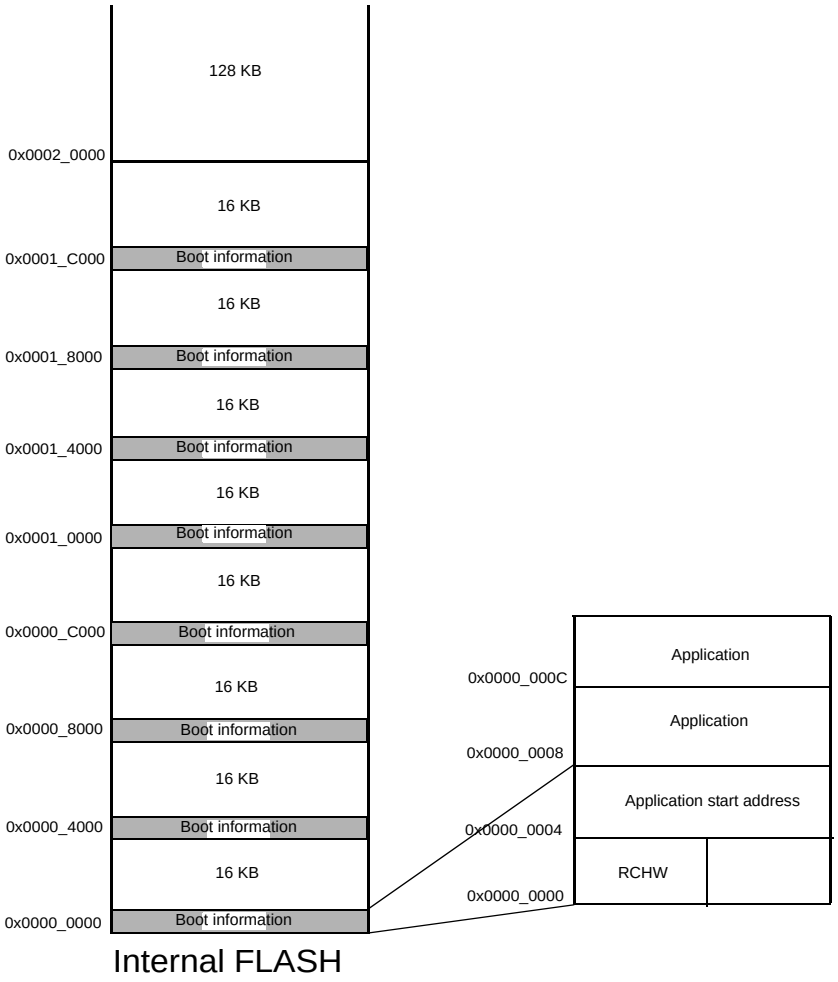



**Figure 6-3. MPC5645S flash memory partitioning and RCHW search**

**Table 6-3. Flash boot sector**

| Block | Address |
|---|---|
| 0 | 0x0000_0000 |
| 1 | 0x0000_4000 |
| 2 | 0x0000_8000 |
| 3 | 0x0000_C000 |
| 4 | 0x0001_0000 |
| 5 | 0x0001_4000 |
| 6 | 0x0001_8000 |
| 7 | 0x0001_C000 |

### 6.5.3 Single Chip boot mode

In single chip mode the hardware searches flash boot sector for a valid boot ID. As soon the device detects the first bootable sector, it jumps within this sector and reads the 32-bit word at offset 0x4. This word is the address where the startup code is located (reset boot vector).

The hardware then sets up a 4 KB MMU entry at this address. The instruction set for this entry is selected by the VLE bit in the RCHW. Then the device executes this startup code. A user application should have a valid instruction at the reset boot vector address. The user application should set up the MMU to allow access to peripherals and memory as required.

If a valid RCHW is not found, the BAM code is executed. In this case BAM moves the MPC5645S into static mode.

#### 6.5.3.1 Boot and alternate boot

Some applications require an alternate boot sector so that the main boot can be erased and reprogrammed in the field.

When an alternate boot is needed, user can create two bootable sectors; the lowest sector shall be the main boot sector and the highest shall be the alternate boot sector. The alternate boot sector does not need to be consecutive to the main boot sector.

This scheme allows to ensure that there is always one active boot sector by erasing one of the boot sectors only:

- Sector shall be activated (that is, program a valid BOOT_ID instead of 0xFF as initially programmed).
- Sector shall be deactivated writing to 0 some of the bits BOOT-ID bit field (bit1 and/or bit3, and/or bit4, and/or bit6).

### 6.5.4 Boot through BAM

#### 6.5.4.1 Executing BAM

Single Chip mode is managed by hardware and BAM does not participate.

BAM is executed only in the following two cases:

- Serial boot mode has been selected by FAB pin
- Hardware has not found a valid Boot-ID in any Flash boot locations

If one of these conditions is true, the device fetches code at location 0xFFFF_C000 and BAM application starts.

#### 6.5.4.2 BAM software flow

With reference to Figure 6-4 a description of BAM logic flow is done.

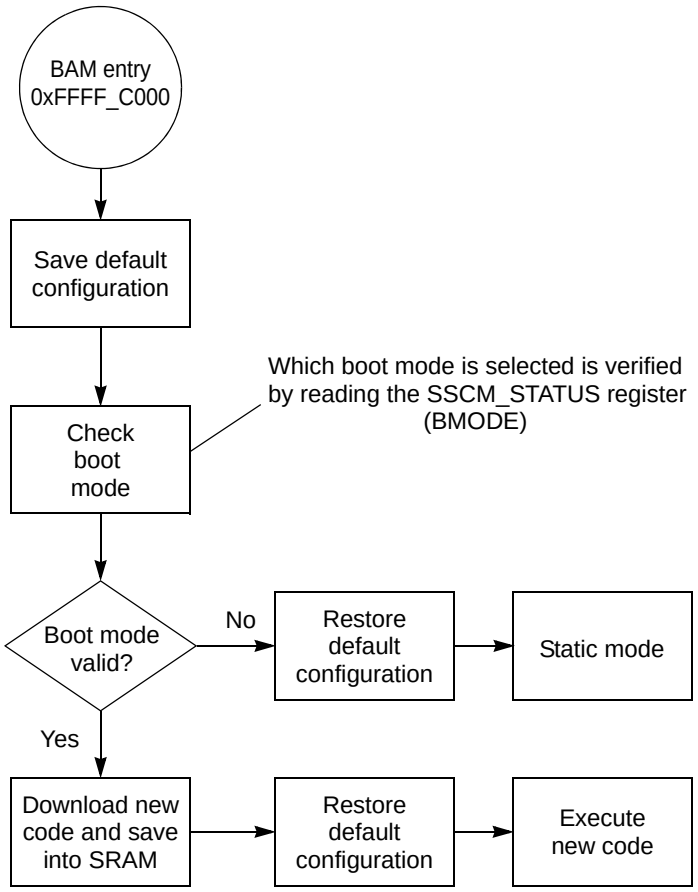



**Figure 6-4. BAM logic flow**

The first action is to save the initial device configuration. In this way is possible to restore the initial configuration after downloading the new code but before executing it. This allows the new code being executed as the device was just coming out of reset.

The BMODE field of the SSCM STATUS register (see Section 44.2.2.1, “System Status Register (STATUS)”) indicates which boot has to be executed (see Table 6-4).

If BMODE field shows either a single-chip value (011) or the reserved values, the boot mode is not considered valid and the BAM pushes the device into static mode.

In all other cases the code of the relative boot is called. Data is downloaded and saved into proper SRAM location.

**Table 6-4. Fields of SSCM STATUS Register Used by BAM**

| Field De | scription |
|---|---|
| BMODE | BMODE Device Boot Mode.<br>000 FlexRay Boot Serial Boot Loader (future use)<br>001 CAN Serial Boot Loader<br>010 SCI Serial Boot Loader<br>011 Single Chip<br>other values are reserved |

Then, the initial device configuration is restored and the code jumps to the address of downloaded code. At this point BAM has finished its task.

If there is any error (that is, communication error, wrong boot selected, etc.), BAM restores the default configuration and puts the device into static mode. Static mode means the device enters the low power mode SAFE and the processor executes a wait instruction. It is needed if the device cannot boot in the mode which was selected. During BAM execution and after, the mode reported by the field S_CURRENT_MODE of the register ME_GS in the module MC_ME Module is "DRUN".

#### 6.5.4.3 BAM resources

BAM uses/initializes the following MCU resources:

- MC_ME and MC_CGM to initialize mode and clock sources
- FlexCAN 0, LINFlex 0 and their pads when performing serial boot mode
- SSCM to check the boot mode and during password check (see Table 6-4 and Figure 6-5)
- External oscillator
- SWT (the BAM disables it)

As already mentioned, the initial configuration is restored before executing the downloaded code.

#### 6.5.4.4 Download and execute the new code

From high level perspective, the download protocol follows steps:

1. Send 64 bit password
2. Send start address, size of downloaded code in bytes and VLE bit
3. Download data
4. Execute code from start address.

Each step must be complete before the next step starts.

The communication is done in half duplex manner, any transmission from host is followed by the MCU transmission:

- host sends data to MCU and start waiting
- MCU echoes to host the data received
- MCU verifies if echoes is correct
  - if data is correct host can continue to send data
  - if data is not correct host stops to transmit and MCU need to be reset.

All multi-byte data structures are sent with MSB first.

A more detailed description of these steps follows.

#### 6.5.4.5 Download 64-bit password and password check

The first 64 bits received represent the password. This password is sent to the Password Check procedure which verifies if it is correct.

The password check data flow is shown in Figure 6-5 where:

- SSCM_STATUS.SEC = 1 means flash secured
- SSCM_STATUS.PUB = 1 means flash with public access.

In case of flash with public access, the received password is compared with the public password 0xFEED_FACE_CAFE_BEEF.

If public access is not allowed but the flash is not secured, the received password is compared with the value saved on NVPWD0 and NVPWD1 registers.

In both of the previous cases, the comparison is done by the BAM code. The BAM code does not enforce any compliance checks on the password itself. If comparison fails the BAM forces the MCU into static mode.

In the case of public access not allowed and flash secured, the password is written into SSCM.PWCMPH-L registers.

After a fixed time waiting, comparison is done by hardware. Then BAM verifies again SSCM_STATUS's SEC flag:

- SEC = 0, flash is now unsecured and BAM continues its task
- SEC = 1, flash is still secured because password was wrong; BAM puts MCU to static mode.

This fixed time depends on the external crystal oscillator frequency (FXOSC). With FXOSC of 12 MHz, the fixed time is 350 ms.

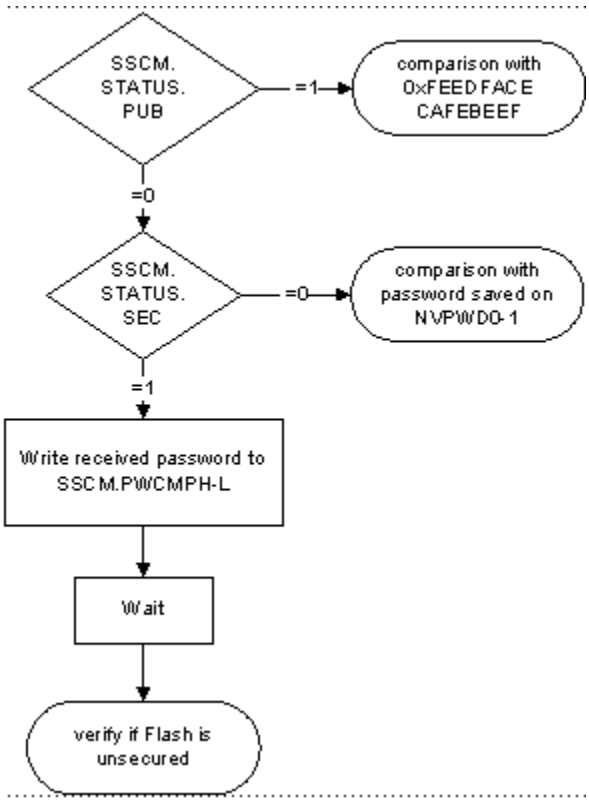



**Figure 6-5. Password Check Flow**

### 6.5.4.6 Download start address, VLE bit and code size

The next 8 bytes received by the MCU contain a 32-bit Start Address, the VLE mode bit and a 31-bit code Length as shown in Figure 6-6.

The VLE bit (Variable Length Instruction) is used to indicate for which instruction set the code has been compiled. The VLE bit should be set to 1 for VLE instruction set or 0 for BookE.

The Start Address defines where the received data will be stored and where the MCU will branch after the download is finished. The two LSB bits of the Start Address are ignored by the BAM program, such that the loaded code should be 32-bit word aligned.

The Length defines how many data bytes have to be loaded.

| START_ADDRESS[31:16] |
|---|

| START_ADDRESS[15:0] |
|---|

| VLE | CODE_LENGTH[30:16] |
|---|---|

| CODE_LENGTH[15:0] |
|---|

**Figure 6-6. Start address, VLE bit and download size in bytes**

#### 6.5.4.7 Download data

Each byte of data received is stored into device's SRAM, starting from the address specified in the previous protocol step.

The address increments until the number of bytes of data received matches the number of bytes specified in the previous protocol step.

Since the SRAM is protected by 32-bit wide Error Correction Code (ECC), BAM always writes bytes into SRAM grouped into 32-bit words. If the last byte received does not fall onto a 32-bit boundary, BAM fills it with 0 bytes.

Then a "dummy" word (0x0000_0000) is written to avoid ECC error during core prefetch.

#### 6.5.4.8 Execute code

The BAM program waits for the last echo message transmission being completed.

Then it restores the initial MCU configuration and jumps to the loaded code at Start Address which was received in step 2 of the protocol.

At this point BAM has finished its tasks and MCU is controlled by new code executing from SRAM.

### 6.5.5 Boot from UART

#### 6.5.5.1 Configuration

Boot from UART protocol is implemented by the LINFlex 0 module. The pins used are:

- LINFlex_TX corresponds to pin PB[2]
- LINFlex_RX corresponds to pin PB[3].

The system clock is driven by an external oscillator.

The LINFlex controller is configured to operate at a baud rate = system clock frequency/833, using an 8-bit data frame without parity bit and 1 stop bit.

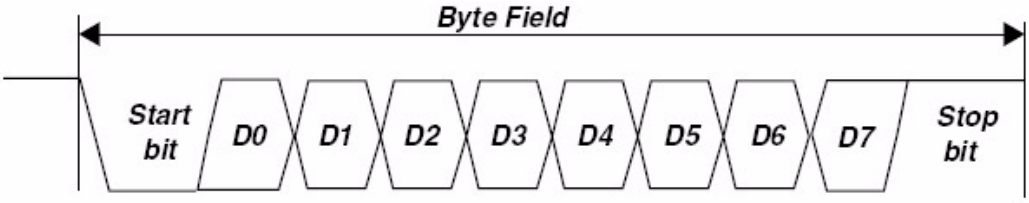



**Figure 6-7. LINFlex Bit Timing in UART Mode**

### 6.5.5.2 Protocol

Table 6-5 summarizes the protocol and BAM action during this boot mode.

See also Section 6.5.7, "Flash memory password swapping," for information on password swapping.

**Table 6-5. UART Boot Mode Download Protocol (Autobaud Disabled)**

| Protocol step | Host sent message | BAM response message | Action |
|---|---|---|---|
| 1 | 64-bit password (MSB first) | 64-bit password | Password checked for validity and compared against stored password. |
| 2 | 32-bit store address | 32-bit store address | Load address is stored for future use. |
| 3 | VLE bit + 31-bit number of bytes (MSB first) | VLE bit + 31-bit number of bytes (MSB first) | Size of download is stored for future use. Verify if VLE bit is set to 1 |
| 4 | 8 bits of raw binarydata | 8 bits of raw binary data | 8-bit data are packed into 32-bit words. This word is saved in SRAM starting from the "Load address". "Load address" increments until the number of data received and stored matches the size as specified in the previous step. |
| 5 | none | none | Branch to dowloaded code |

## 6.5.6 Bootstrap with CAN

### 6.5.6.1 Configuration

Boot from FlexCAN protocol is implemented by the FlexCAN_0 module. The pins used are:

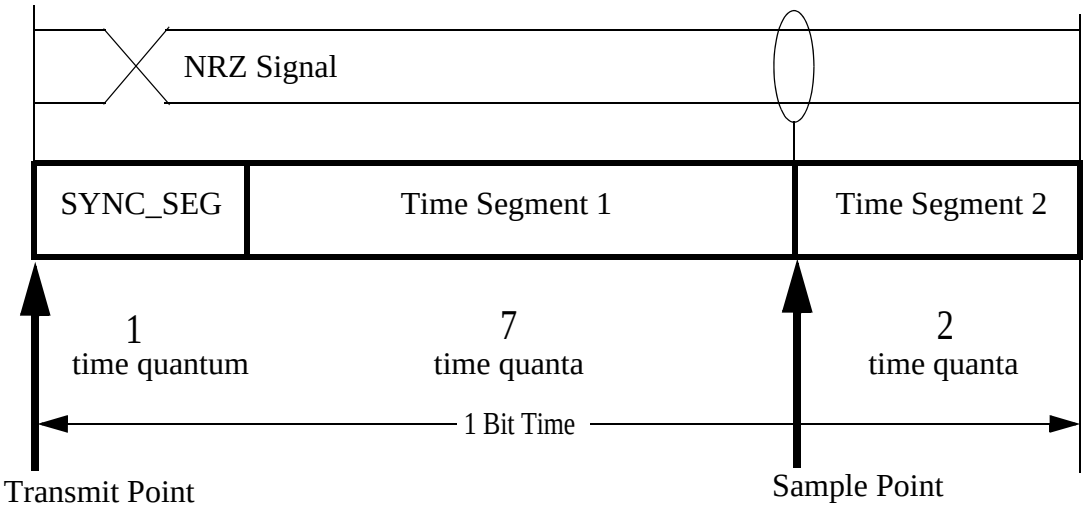
- CAN_TX corresponds to pin PB[0]
- CAN_RX corresponds to pin PB[1].

Boot from FlexCAN uses the system clock driven by an external oscillator.

The FlexCAN controller is configured to operate at a baud rate = system clock frequency/40.

It uses the standard 11-bit identifier format detailed in FlexCAN 2.0A specification.

FlexCAN controller bit timing is programmed with 10 time quanta, and the sample point is 2 time quanta before the end, as shown in Figure 6-8.




**1 time quantum = 4 system clock periods**

**Figure 6-8. FlexCAN Bit Timing**

### 6.5.6.2 Protocol

Table 6-6 summarizes the protocol and BAM action during this boot mode. All data are transmitted byte wise.

See also Section 6.5.7, “Flash memory password swapping,” for information on password swapping.

**Table 6-6. FlexCAN Boot Mode Download Protocol (Autobaud Disabled)**

| Protocol step | Host sent message | BAM response message | Action |
|---|---|---|---|
| 1 | CAN ID 0x011+<br>64-bit password | CAN ID 0x001+<br>64-bit password | Password checked for validity and compared against stored password. |
| 2 | CAN ID 0x012+<br>32-bit store address+<br>VLE bit+<br>31-bit number of bytes | CAN ID 0x002+<br>32-bit store address+<br>VLE bit+<br>31-bit number of bytes | Load address is stored for future use.<br>Size of download is stored for future use.<br>Verify if VLE bit is set to 1 |
| 3 | CAN ID 0x013+<br>8 to 64 bits of raw binary data | CAN ID 0x003+<br>8 to 64 bits of raw binary data | 8-bit data are packed into 32-bit words. These words are saved in SRAM starting from the “Load address”.<br>“Load address” increments until the number of data received and stored matches the size as specified in the previous step. |
| 5 | none | none | Branch to dowloaded code |

**Table 6-7. System clock frequency related to external clock frequency**

| $f_{osc}$ [MHz] f | $_{rc}/f_{osc}$ [1] | $f_{sys}$ [MHz] |
|---|---|---|
| 4 - 8 | 4 - 2 | 16 - 32 |
| 8 - 12 | 2 - 4/3 | 32 - 48 |
| 12 - 16 | 4/3 - 1 | 36 - 48 |
| 16 - 24 | 1 - 2/3 | 32 - 48 |
| > 24 | < 2/3 | > 24 |

1 These values and consequently the $f_{sys}$ suffer from the precision of the RC internal oscillator used to measure $f_{osc}$ through the CMU module.

### 6.5.7 Flash memory password swapping

When the chip uses the BAM to boot using CAN or UART, the required flash memory password is different depending on whether the flash memory is secured or unsecured. This difference affects the way in which the NVPWD0, NVPWD1, and NVSCI0 registers must be programmed.

When the flash memory is secured:

- The registers are programmed as follows:
  - NVPWD0 = 0x87654321
  - NVPWD1 = 0x12345678
  - NVSCI0 = 0x55AA1111
- To download the code via serial boot, the provided password is 0x1234_5678_8765_4321 (NVPWD1 followed by NVPWD0).

When the flash memory is unsecured:

- The registers are programmed as follows:
  - NVPWD0 = 0x87654321
  - NVPWD1 = 0x12345678
  - NVSCI0 = 0x55AA55AA
- To download the code via serial boot, the provided password is 0x8765_4321_1234_5678 (NVPWD0 followed by NVPWD1).

### 6.5.8 Interrupts

No interrupts are generated by or are enabled by the BAM.

This page is intentionally left blank.

# Chapter 7
# CAN Sampler

## 7.1 Introduction

The CAN Sampler peripheral has been designed to store the first identifier of CAN message "detected" on the CAN bus while no precise clock (Crystal) is running at that time on the device, typically in Low Power modes (STOP, HALT or STANDBY) or in RUN mode with crystal switched off.

Depending on both CAN baudrate and Low Power mode used, it is possible to catch either the first or the second CAN frame by sampling one of two CAN Rx ports and storing all samples in internal registers.

After selection of the mode (first or second frame), the CAN Sampler stores samples of the 48 bits or skips the first frame and stores samples of the 48 bits of second frame using the 16-MHz IRC oscillator and the 5-bit clock prescaler.

After completion, Software has to process the sampled data in order to rebuild the 48 minimal bits.

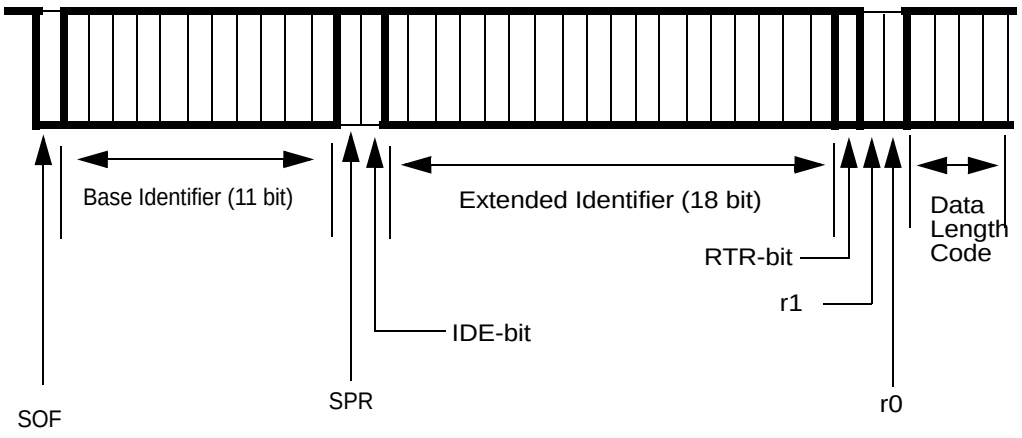



**Figure 7-1. Extended CAN data frame**

## 7.2 Main features

- Store 384 samples, equivalent to 48 CAN bit @8 samples/bit
- Sample frequency from 500 kHz up to 16 MHz, equivalent at 8 samples/bit to CAN baud rates of 62.5 Kbps to 2 Mbps
- User selectable CAN Rx sample port, CAN0RX, CAN1RX, or CAN2RX
- 16 MHz IRC clock
- 5-bit clock prescaler

- Configurable trigger mode (immediate, next frame)
- Flexible samples processing by software
- Very low power consumption

## 7.3 Register description

The CAN Sampler registers are listed in Table 7-1.

**Table 7-1. CAN Sampler registers**

| Register Name | Address Offset | Reset Value | Location |
|---|---|---|---|
| Control Register (CR) | 00h | 0000 0000h | on page 198 |
| Sample registers 0 | 04h | xxxx xxxxh[1] | on page 199 |
| Sample register 1 | 08h | xxxx xxxxh[1] | on page 199 |
| ..... | .... | .... | |
| Sample register 11 | 30h | xxxx xxxxh[1] | on page 199 |

[1] The initialization data is unknown. They will be filled only after first CAN sampling.

### 7.3.1 CAN Sampler Control Register (CR)

Address Offset: 0x00 Reset value: 0000 0000h

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RX_COMPLETE | BUSY | Active_CK | 0 | 0 | 0 | mode | CAN_RX_SEL | | | BRP | | | | | CAN_SMPLR_EN |
| W | | | | | | | | | | | | | | | | |

**Figure 7-2. Control Register (CR)**

**Table 7-2. CR field descriptions**

| | |
|---|---|
| 0-15 | Reserved |
| 16<br>RX_COMPLETE | 1: CAN frame is stored in the sample registers<br>0: CAN frame has not been stored in the sample registers |
| 17<br>BUSY | This bit indicates the status of sampling<br>1: Sampling is ongoing<br>0: Sampling is complete or has not started |
| 18<br>Active_CK | This bit indicates which is the current clock for sample registers.<br>1: FIRC is the currently selected clock<br>0: The peripheral set clock is the currently selected clock |

**Table 7-2. CR field descriptions (continued)**

| | |
|---|---|
| 19-21 | Reserved<br>These are reserved bits. These bits are always read as ‘0’. |
| 22<br>Mode | 0:Skip the first frame and sample and store the second frame (SF_MODE)<br>1:Sample and store the first frame (FF_MODE) |
| 23-25<br>CAN_RX_SEL | These bits determine which RX bit is sampled.<br>000: Rx0 is selected<br>001: Rx1 is selected<br>010: Rx2 is selected<br>011: Reserved<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Reserved |
| 26-30<br>BRP | Baud Rate Prescaler<br>These bits are used to set the baud rate before going into standby mode<br>00000: Prescaler has 1<br>11111: Presacler has 32 |
| 31<br>CAN_SMPLR_EN | CAN SAMPLER Enable<br>This bit enables the CAN Sampler before going into standby or stop mode.<br>0 CAN Sampler is Disabled<br>1 CAN Sampler is Enabled |

## 7.3.2 CAN Sampler Sample Registers 0–11

The CAN sampler sample registers 0–11 have the same structure; Figure 7-3 and Figure 7-4 show this structure for registers 0 and 11, respectively.

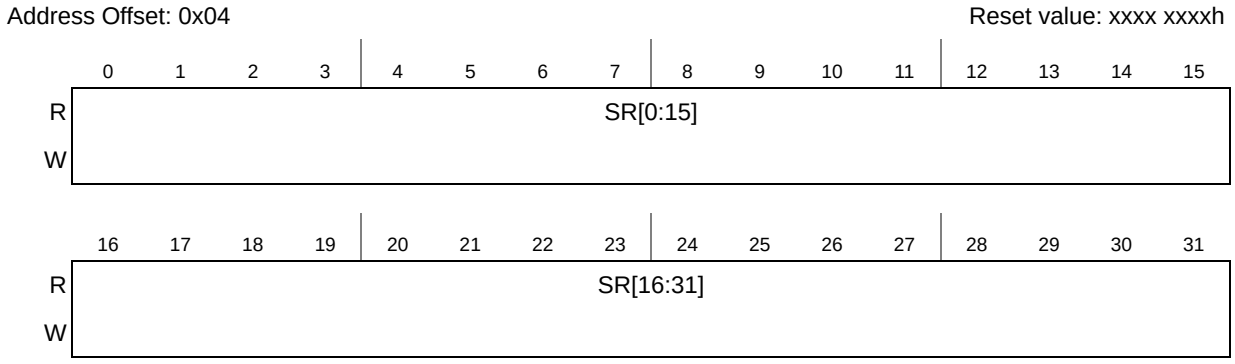



**Figure 7-3. Sample Register 0**

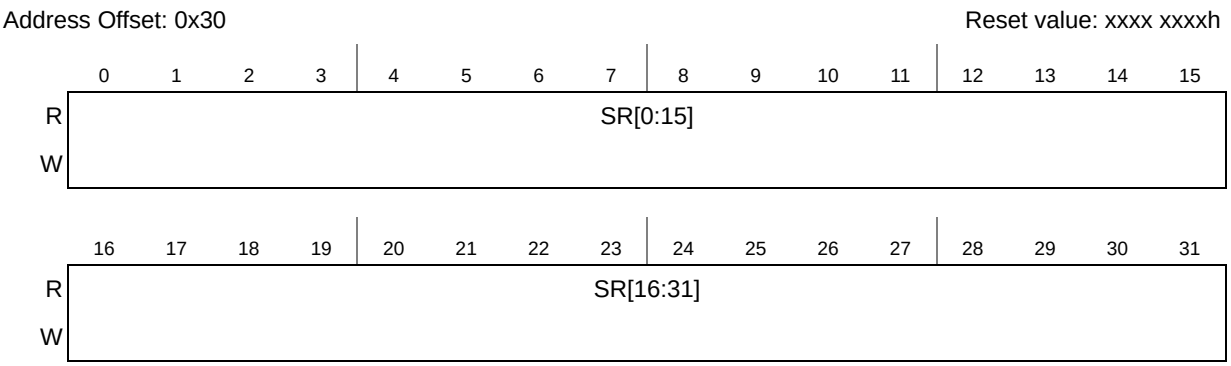



**Figure 7-4. Sample Register 11**

## 7.4 Functional description

As the CAN Sampler is driven by the 16 MHz IRC to sample properly the CAN identifier, two modes are possible depending on both CAN baudrate and Low Power mode used:

- Immediate sampling on falling edge detection (first CAN frame): this mode is used when the IRC 16 MHz is available in LP mode, e.g. STOP or HALT.
- Sampling on next frame (second CAN frame): this mode is used when the IRC 16 MHz is switched off in LP mode, e.g. STANDBY. Due to the start-up times of both the Voltage regulator and the IRC 16 MHz (~10 µs), the CAN sampler would miss the first bits of a CAN identifier sent at 500kbps. Therefore the first identifier is ignored and the sampling is performed on the first falling edge of after interframe space.

The CAN sampler performs sampling on a user selected CAN Rx port, normally when the device is in standby or stop mode storing the samples in internal registers. The user is required to configure the baud rate to achieve 8 samples per CAN nominal bit.It does not perform any sort of filtering on input samples. Thereafter the software must enable the sampler by setting CAN_SMPLR_EN bit in CR register.It then becomes the master controller for accessing the internal registers implemented for storing samples.

The CAN sampler, when enabled, waits for a low pulse on the selected Rx line, taking it as a valid bit of the first CAN frame and generates the RC wakeup request which can be used to start the RC oscillator. Depending upon the mode, it stores the first 8 samples of the 48 bits on selected Rx line or skips the first frame and stores 8 bits for first 48 bits of second frame. In FF_MODE, it samples the CAN Rx line on RC clock and stores the 8 samples of first 48 bits (384 samples). In SF_MODE, it samples the Rx and waits for 11 consecutive dominant bits (11 * 8 samples), taking it as the end of first frame. It then waits for first low pulse on the Rx, taking it as a valid Start of Frame (SOF) of the second frame. The sampler takes 384 samples (48 bytes * 8) using the RC clock (configuring 8 samples per nominal bit) of the second frame, including the SOF bit. These samples are stored in consecutive addresses of the (12 x 32) internal registers. RX_COMPLETE bit is set to ‘1’, indicating that sampling is complete.

Software should now process the sampled data by first becoming master for accessing samples internal registers by resetting CAN_SMPLR_EN bit.The sampler will need to be enabled again to start waiting for a new sampling routine.

### 7.4.1 Enabling/disabling the CAN Sampler

The CAN sampler is disabled on reset and the CPU is able to access the 12 registers used for storing samples. The CAN Sampler must be enabled before entering standby or stop mode by setting CR[CAN_SMPLR_EN].

When the CAN sampler is enabled, the A, D, WEN, CSN and CK to the (12 x 32) block of registers are switched to those generated by the kernel of the sampler. You can monitor CR[Active_CK] to check which is the active clock to the registers.

Any activity on selected Rx line, the sampler enables the 16 MHz RC oscillator. When CAN_SMPLR_EN is reset to 0, the sampler should at least receive 3 RC clock pulses to reset itself, after which the RC can be switched off.

When the software wishes to access the sample registers contents it must first reset the CAN_SMPLR_EN bit by writing a ‘0’. Before accessing the register contents it must monitor Active_CK bit for ‘0’.When this bit is reset it can safely access the (12 x 32) sample registers. While shifting from normal to sample mode and vice versa, the sample register signals must be static and inactive to ensure the data is not corrupt.

### 7.4.2 Selecting the Rx port

One Rx port can be selected per sampling routine; the port to be sampled is selected by CAN_RX_SEL.

**Table 7-3. Internal multiplexer correspondence**

| CAN_RX_SEL | Rx selected |
|---|---|
| 000 | CANRX_0 PB[1] |
| 001 | CANRX_1 PB[10] |
| 010 | CANRX_2 PM[3] |
| 011 | Reserved |
| 100 | Reserved |
| 101 | Reserved |
| 110 | Reserved |
| 111 | Reserved |

### 7.4.3 Baud rate generation

Sampling is performed at a baud rate that is set by the software as a multiple of RC oscillator frequency of 62.5 ns (assuming RC is configured for high frequency mode i.e. 16 MHz). User must set the baud rate prescaler (BRP) such that 8 samples per bit are achieved.

Baud rate setting must be made by software before going into standby or stop mode. This is done by setting BRP bits 5:1 in Control register. The reset value of BRP is 00000 and can be set to max. 11111 which gives a prescale value of BRP+1 thus providing a BRP range of 1 to 32.

- Maximum bitrate supported for sampling is 2Mbps using BRP as 1
- Minimum bitrate supported for sampling is 62.5kbps using BRP as 32

For example, suppose system is transmitting at 125kbps. In this case, nominal bit period:

$$T=1/(125*10^{3})s = 8*10^{-3}*10^{-3}s = 8\mu s \qquad \textit{Eqn. 7-1}$$

To achieve 8 samples per bit

Sample period= 8/8 $\mu$s =1$\mu$s

BRP = 1$\mu$s/62.5ns = 16. Thus, in this case BRP = 01111.

# Chapter 8
# Clock Description

## 8.1 Clock architecture

Figure 8-1 shows the how the system clocks on MPC5645S are generated.

The peripheral groupings mentioned in the figure are listed in Table 8-1. Peripherals not explicitly listed in a peripheral set or using an auxiliary clock use the system clock (or where available an alternative chosen within the peripheral) as their reference.

**Table 8-1. MPC5645S peripheral sets**

| Peripheral set 1 | Peripheral set 2 | Peripheral set 3 | Peripheral set 4 |
|---|---|---|---|
| All LINFlex modules | All FlexCAN modules | ADC | Sound Generation Module |
| All I2C modules | CAN Sampler | — | — |
| Stepper Motor Controller | All DSPI modules | — | — |

**NOTE**

The system clock and modules may not select the FMPLL0 clock directly. The option available is FMPLL0 divided by 2. Therefore, to select a 125 MHz clock, the FMPLL0 must be configured to produce 250 MHz.

The DRAM Controller requires FMPLL0 to be operational and selected as the system clock for correct operation.

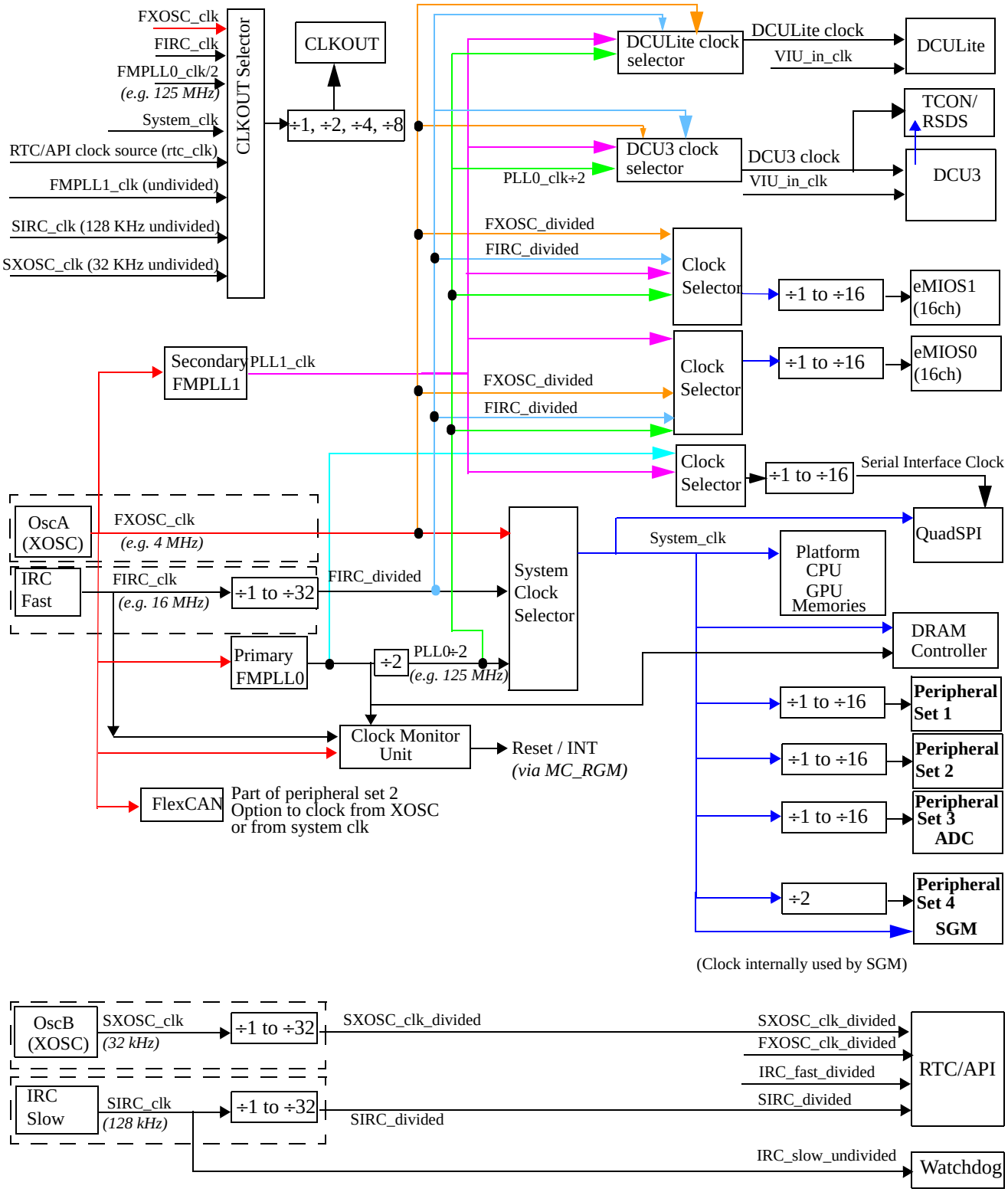



**Figure 8-1. MPC5645S system clock generation**

## 8.2 Auxiliary clocks

This device has five auxiliary clocks configurable using the MC_CGM registers. These auxiliary clocks allow the associated peripherals to operate at clock speeds independent of the system clock (sys_clk). The peripherals also use the undivided system clock to synchronously interface with the rest of the device. The auxiliary clock configuration is:

- Auxiliary Clock 0: DCU3
- Auxiliary Clock 1: eMIOS0
- Auxiliary Clock 2: eMIOS1
- Auxiliary Clock 3: QuadSPI
- Auxiliary Clock 4: DCULite

## 8.3 Clock Generation Module (MC_CGM)

### 8.3.1 Introduction

This document describes the Clock Generation Module (MC_CGM) which includes, but is not limited to, the functionality, pin description, and registers of the MC_CGM module.

#### 8.3.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the SoC blocks. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME documentation for more details). A set of MC_CGM registers controls the clock dividers which are used for divided system and peripheral clock generation. The MC_CGM memory space also includes the control registers of the clock sources themselves, for example PLLs, IRCs, and oscillators. The MC_CGM also selects and generates an output clock.

Figure 8-2 depicts the MC_CGM block diagram.

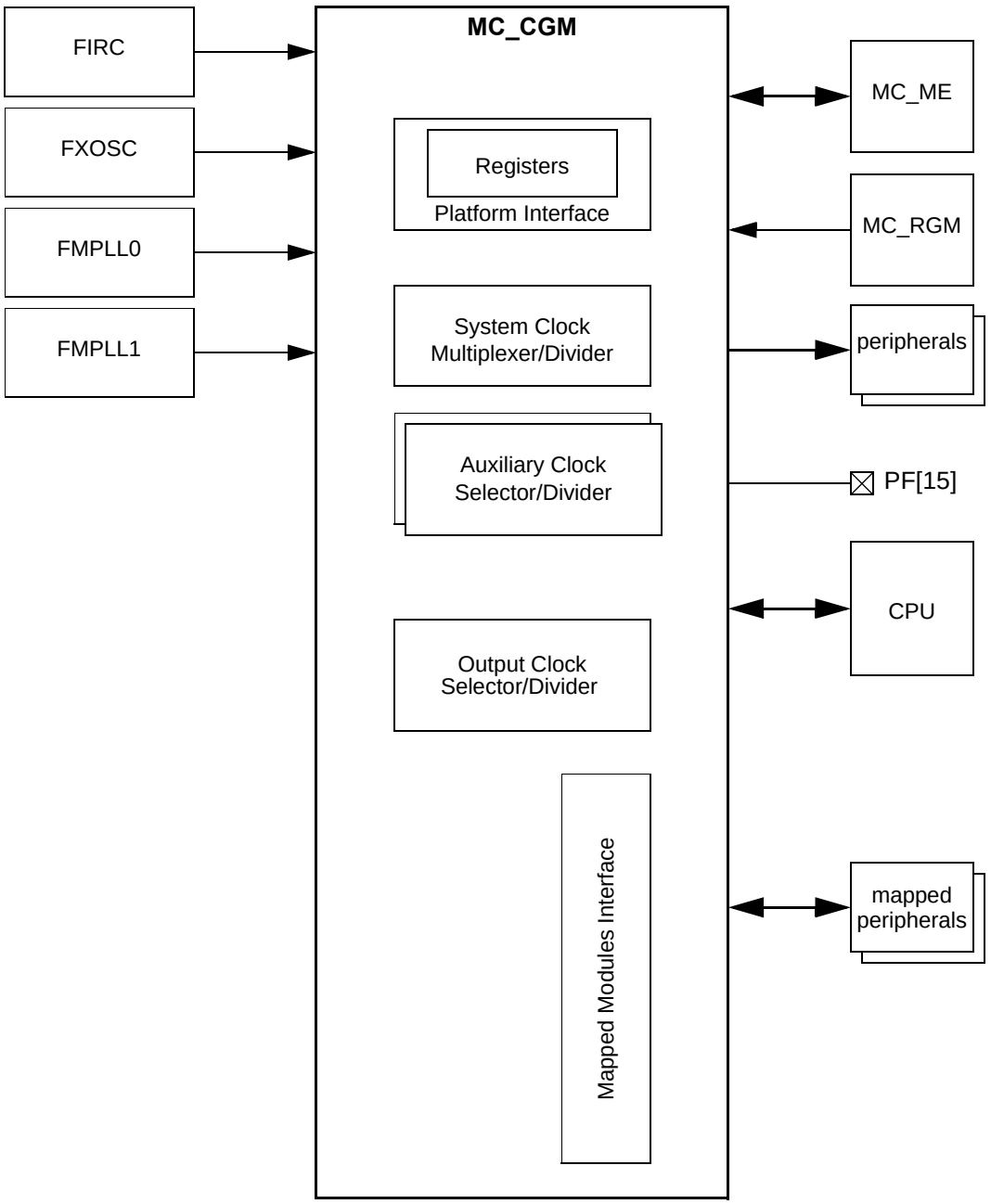



**Figure 8-2. MC_CGM block diagram**

### 8.3.1.2 Features

The MC_CGM includes the following features:

- generates system and peripheral clocks
- selects and enables/disables the system clock supply from system clock sources according to MC_ME control
- contains a set of registers to control clock dividers for divided clock generation

- supports multiple clock sources and maps their address spaces to its memory map
- generates an output clock
- guarantees glitch-less clock transitions when changing the system clock selection
- supports 8, 16 and 32-bit wide read/write accesses

### 8.3.2 External signal description

The MC_CGM delivers an output clock to the PF[15] pin for off-chip use and/or observation.

### 8.3.3 Memory map and register definition

**Table 8-2. MC_CGM register description**

| Address | Name | Description | Size | Access | | | Location |
|---|---|---|---|---|---|---|---|
| | | | | User | Supervisor | Test | |
| 0xC3FE-0340 | CGM_VIU_MUX | VIU2 multiplex select | word | read | read/write | read/write | on page 214 |
| 0xC3FE_0370 | CGM_OC_EN | Output Clock Enable | word | read | read/write | read/write | on page 215 |
| 0xC3FE_0374 | CGM_OCDS_SC | Output Clock Division Select | byte | read | read/write | read/write | on page 216 |
| 0xC3FE_0378 | CGM_SC_SS | System Clock Select Status | byte | read | read | read | on page 217 |
| 0xC3FE_037C | CGM_SC_DC0 | System Clock Divider Configuration 0 | byte | read | read/write | read/write | on page 218 |
| 0xC3FE_037D | CGM_SC_DC1 | System Clock Divider Configuration 1 | byte | read | read/write | read/write | on page 218 |
| 0xC3FE_037E | CGM_SC_DC2 | System Clock Divider Configuration 2 | byte | read | read/write | read/write | on page 218 |
| 0xC3FE_037F | CGM_SC_DC3 | System Clock Divider Configuration 3 | byte | read | read/write | read/write | on page 218 |
| 0xC3FE_0380 | CGM_AC0_SC | Aux Clock 0 Select Control | word | read | read/write | read/write | on page 219 |
| 0xC3FE_0384 | CGM_AC0_DC | Aux Clock 0 Divider Configuration | byte | read | read/write | read/write | on page 220 |
| 0xC3FE_0388 | CGM_AC1_SC | Aux Clock 1 Select Control | word | read | read/write | read/write | on page 220 |
| 0xC3FE_038C | CGM_AC1_DC | Aux Clock 1 Divider Configuration | byte | read | read/write | read/write | on page 221 |
| 0xC3FE_0390 | CGM_AC2_SC | Aux Clock 2 Select Control | word | read | read/write | read/write | on page 222 |
| 0xC3FE_0394 | CGM_AC2_DC | Aux Clock 2 Divider Configuration | byte | read | read/write | read/write | on page 223 |

**Table 8-2. MC_CGM register description (continued)**

| Address | Name | Description | Size | Access | | | Location |
|---|---|---|---|---|---|---|---|
| | | | | User | Supervisor | Test | |
| 0xC3FE_0398 | CGM_AC3_SC | Aux Clock 3 Select Control | word | read | read/write | read/write | on page 223 |
| 0xC3FE_039C | CGM_AC3_DC | Aux Clock 3 Divider Configuration | byte | read | read/write | read/write | on page 224 |
| 0xC3FE_03A0 | CGM_AC4_SC | Aux Clock 4 Select Control | word | read | read/write | read/write | on page 225 |
| 0xC3FE_03A4 | CGM_AC4_DC | Aux Clock 4 Divider Configuration | byte | read | read/write | read/write | on page 226 |

**NOTE**

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

**Table 8-3. MC_CGM memory map**

| Address | Name | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FE_0000 … 0xC3FE_003C | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_0040 … 0xC3FE_005C | SXOSC registers (see Section 8.4, “Oscillators”) | | | | | | | | | | | | | | | | |
| 0xC3FE_0060 … 0xC3FE_007C | FIRC registers (see Section 8.4, “Oscillators”) | | | | | | | | | | | | | | | | |
| 0xC3FE_0080 … 0xC3FE_009C | SIRC registers (see Section 8.4, “Oscillators”) | | | | | | | | | | | | | | | | |

**Table 8-3. MC_CGM memory map (continued)**

| Address | Name | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FE_00A0 … 0xC3FE_00BC | FMPLL0 registers | | | | | | | | | | | | | | | | |
| 0xC3FE_00C0 … 0xC3FE_00DC | FMPLL1 registers | | | | | | | | | | | | | | | | |
| 0xC3FE_00E0 … 0xC3FE_00FC | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_0100 … 0xC3FE_011C | CMU registers (see Section 8.6, “Clock Monitor Unit (CMU)”) | | | | | | | | | | | | | | | | |
| 0xC3FE_0120 … 0xC3FE_013C | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_0140 … 0xC3FE_015C | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_0160 … 0xC3FE_017C | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_0180 … 0xC3FE_019C | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_01A0 … 0xC3FE_01BC | reserved | | | | | | | | | | | | | | | | |

**Table 8-3. MC_CGM memory map (continued)**

| Address | Name | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FE_01C0 … 0xC3FE_01DC | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_01E0 … 0xC3FE_01FC | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_0200 … 0xC3FE_021C | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_0220 … 0xC3FE_023C | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_0240 … 0xC3FE_025C | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_0260 … 0xC3FD_C27C | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_0280 … 0xC3FE_029C | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_02A0 … 0xC3FE_02BC | reserved | | | | | | | | | | | | | | | | |
| 0xC3FE_02C0 … 0xC3FE_02DC | reserved | | | | | | | | | | | | | | | | |

**Table 8-3. MC_CGM memory map (continued)**

| Address | Name | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FE_02E0 … 0xC3FE_02FC | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FE_0300 … 0xC3FE_031C | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FE_0320 … 0xC3FE_033C | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FE_0340 | CGM_VIU_MUX | R | VIUSEL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_0344 … 0xC3FE_036C | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FE_0370 | CGM_OC_EN | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | EN |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_0374 | CGM_OCDS_SC | R | 0 | 0 | SELDIV | | SELCTL | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_0378 | CGM_SC_SS | R | 0 | 0 | 0 | 0 | SELSTAT | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |

**Table 8-3. MC_CGM memory map (continued)**

| Address | Name | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FE_037C | CGM_SC_DC0…3 | R | DE0 | 0 | 0 | 0 | DIV0 | | | | DE1 | 0 | 0 | 0 | DIV1 | | | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | DE2 | 0 | 0 | 0 | DIV2 | | | | DE3 | 0 | 0 | 0 | DIV3 | | | |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_0380 | CGM_AC0_SC | R | 0 | 0 | 0 | 0 | SELCTL | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_0384 | CGM_AC0_DC0 | R | DE0 | 0 | 0 | 0 | DIV0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_0388 | CGM_AC1_SC | R | 0 | 0 | 0 | 0 | SELCTL | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_038C | CGM_AC1_DC0 | R | DE0 | 0 | 0 | 0 | DIV0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_0390 | CGM_AC2_SC | R | 0 | 0 | 0 | 0 | SELCTL | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_0394 | CGM_AC2_DC0 | R | DE0 | 0 | 0 | 0 | DIV0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_0398 | CGM_AC3_SC | R | 0 | 0 | 0 | 0 | SELCTL | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |

**Table 8-3. MC_CGM memory map (continued)**

| Address | Name | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FE_039C | CGM_AC3_DC0 | R | DE0 | 0 | 0 | 0 | DIV0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_03A0 | CGM_AC4_SC | R | 0 | 0 | 0 | 0 | SELCTL | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_03A4 | CGM_AC4_DC0 | R | DE0 | 0 | 0 | 0 | DIV0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_03A8 … 0xC3FE_3FFC | reserved | | | | | | | | | | | | | | | | | |

## 8.3.3.1 Register Descriptions

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the **CGM_OC_EN** register may be accessed as a word at address 0xC3FE_0370, as a half-word at address 0xC3FE_0372, or as a byte at address 0xC3FE_0373.

### 8.3.3.1.1 VIU2 multiplex select register (CGM_VIU_MUX)

Address: 0xC3FE_0340 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VIU SEL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | AUTO_DIV2 | | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-3. VIU2 multiplex select register (CGM_VIU_MUX)**

This register is used to select which pins to use for VIU2 input. For details, see Section 3.3.5, “DRAM interface”.

**Table 8-4. CGM_VIU_MUX field descriptions**

| Field | Description |
|---|---|
| VIUSEL | Selects which pins to use for VIU2 input.<br>0 PDI[7:0], HSYNC, VSYNC<br>1 PDI[17:8] |
| AUTO_DIV2 | 00 System clock automatically set to half the SDRAM clock frequency<br>01 Reserved - do not select<br>10 Reserved - do not select<br>11 Reserved - do not select |

### 8.3.3.1.2 Output Clock Enable Register (CGM_OC_EN)

Address 0xC3FE_0370 Access: User read, Supervisor read/write, Test read/write

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

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

**Figure 8-4. Output Clock Enable Register (CGM_OC_EN)**

This register is used to enable and disable the output clock.

**Table 8-5. Output Clock Enable Register (CGM_OC_EN) Field Descriptions**

| Field | Description |
|---|---|
| EN | **Output Clock Enable control**<br>0 Output Clock is disabled<br>1 Output Clock is enabled |

### 8.3.3.1.3 Output Clock Division Select Register (CGM_OCDS_SC)

Address 0xC3FE_0374 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | SELDIV | | SELCTL | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 8-5. Output Clock Division Select Register (CGM_OCDS_SC)**

This register is used to select the current output clock source and by which factor it is divided before being delivered at the output clock.

**Table 8-6. Output Clock Division Select Register (CGM_OCDS_SC) Field Descriptions**

| Field | Description |
|---|---|
| SELDIV | **Output Clock Division Select**<br>00 output selected Output Clock without division<br>01 output selected Output Clock divided by 2<br>10 output selected Output Clock divided by 4<br>11 output selected Output Clock divided by 8 |
| SELCTL | **Output Clock Source Selection Control** — This value selects the current source for the output clock.<br>0000 16 MHz int. RC osc.<br>0001 4-16 MHz ext. xtal osc.<br>0010 primary PLL/2<br>0011 secondary PLL<br>0100 128 kHz int. RC osc.<br>0101 32 kHz ext. xtal osc.<br>0110 reserved<br>0111 RTC clock<br>1000 system clock<br>1001 reserved<br>1010 reserved<br>1011 reserved<br>1100 reserved<br>1101 reserved<br>1110 reserved<br>1111 reserved |

### 8.3.3.1.4 System Clock Select Status Register (CGM_SC_SS)

Address 0xC3FE_0378 Access: User read, Supervisor read, Test read

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

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

**Figure 8-6. System Clock Select Status Register (CGM_SC_SS)**

This register provides the current system clock source selection.

**Table 8-7. System Clock Select Status Register (CGM_SC_SS) Field Descriptions**

| Field | Description |
|---|---|
| SELSTAT | **System Clock Source Selection Status** — This value indicates the current source for the system clock.<br>0000 16 MHz int. RC osc.<br>0001 reserved<br>0010 reserved<br>0011 div. 4-16 MHz ext. xtal osc.<br>0100 primary PLL/2<br>0101 reserved<br>0110 reserved<br>0111 reserved<br>1000 reserved<br>1001 reserved<br>1010 reserved<br>1011 reserved<br>1100 reserved<br>1101 reserved<br>1110 reserved<br>1111 system clock is disabled |

### 8.3.3.1.5 System Clock Divider Configuration Registers (CGM_SC_DC0…3)

Address 0xC3FE_037C Access: User read, Supervisor read/write, Test read/write

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| R | DE0 | 0 | 0 | 0 | DIV0 | | | | DE1 | 0 | 0 | 0 | DIV1 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| R | DE2 | 0 | 0 | 0 | DIV2 | | | | DE3 | 0 | 0 | 0 | DIV3 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-7. System Clock Divider Configuration Registers (CGM_SC_DC0…3)**

These registers control the system clock dividers. The divided clock is the reference for the associated peripheral set.

**Table 8-8. System Clock Divider Configuration Registers (CGM_SC_DC0…3) Field Descriptions**

| Field | Description |
|---|---|
| DE0 | **Divider 0 Enable**<br>0 Disable system clock divider 0<br>1 Enable system clock divider 0 |
| DIV0 | **Divider 0 Division Value** — The resultant peripheral set 1 clock will have a period DIV0 + 1 times that of the system clock. If the DE0 is set to ‘0’ (Divider 0 is disabled), any write access to the DIV0 field is ignored and the peripheral set 1 clock remains disabled. |
| DE1 | **Divider 1 Enable**<br>0 Disable system clock divider 1<br>1 Enable system clock divider 1 |
| DIV1 | **Divider 1 Division Value** — The resultant peripheral set 2 clock will have a period DIV1 + 1 times that of the system clock. If the DE1 is set to ‘0’ (Divider 1 is disabled), any write access to the DIV1 field is ignored and the peripheral set 2 clock remains disabled. |
| DE2 | **Divider 2 Enable**<br>0 Disable system clock divider 2<br>1 Enable system clock divider 2 |
| DIV2 | **Divider 2 Division Value** — The resultant peripheral set 3 clock will have a period DIV2 + 1 times that of the system clock. If the DE2 is set to ‘0’ (Divider 2 is disabled), any write access to the DIV2 field is ignored and the peripheral set 4 clock remains disabled. |
| DE3 | **Divider 3 Enable**<br>0 Disable system clock divider 3<br>1 Enable system clock divider 3 |
| DIV3 | **Divider 3 Division Value** — The clock for peripheral clock 4 (SGM) is fixed at divide by 2. This field is not writable and always reads as 0. |

### 8.3.3.1.6 Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)

Address 0xC3FE_0380 Access: User read, Supervisor read/write, Test read/write

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

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

**Figure 8-8. Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)**

This register is used to select the current clock source for the following clocks:

- undivided: unused
- divided by auxiliary clock 0 divider 0: DCU3 clock

**Table 8-9. CGM_AC0_SC field descriptions**

| Field | Description |
|---|---|
| SELCTL | **Auxiliary Clock 0 Source Selection Control** — This value selects the current source for auxiliary clock 0.<br>0000 div. 16 MHz int. RC osc.<br>0001 div. 4-16 MHz ext. xtal osc.<br>0010 secondary PLL<br>0011 primary PLL/2<br>0100 reserved<br>0101 reserved<br>0110 reserved<br>0111 reserved<br>1000 reserved<br>1001 reserved<br>1010 reserved<br>1011 reserved<br>1100 reserved<br>1101 reserved<br>1110 reserved<br>1111 reserved |

### 8.3.3.1.7 Auxiliary Clock 0 Divider Configuration Register (CGM_AC0_DC)

Address 0xC3FE_0384 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DE0 | 0 | 0 | 0 | DIV0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 8-9. Auxiliary Clock 0 Divider Configuration Register (CGM_AC0_DC)**

This register controls the auxiliary clock 0 divider.

**Table 8-10. CGM_AC0_DC field descriptions**

| Field | Description |
|---|---|
| DE0 | **Divider 0 Enable**<br>0 Disable auxiliary clock 0 divider 0<br>1 Enable auxiliary clock 0 divider 0 |
| DIV0 | **Divider 0 Division Value** — The resultant DCU3 clock will have a period DIV0 + 1 times that of auxiliary clock 0. If the DE0 is set to 0 (Divider 0 is disabled), any write access to the DIV0 field is ignored and the DCU3 clock remains disabled. |

### 8.3.3.1.8 Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)

Address 0xC3FE_0388 Access: User read, Supervisor read/write, Test read/write

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

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

**Figure 8-10. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)**

This register is used to select the current clock source for the following clocks:

- undivided: unused
- divided by auxiliary clock 1 divider 0: eMIOS0 clock

**Table 8-11. Auxiliary Clock 1 Select Control Register (CGM_AC1_SC) Field Descriptions**

| Field | Description |
|---|---|
| SELCTL | **Auxiliary Clock 1 Source Selection Control** — This value selects the current source for auxiliary clock 1.<br>0000 div. 16 MHz int. RC osc.<br>0001 div. 4-16 MHz ext. xtal osc.<br>0010 secondary PLL<br>0011 primary PLL/2<br>0100 reserved<br>0101 reserved<br>0110 reserved<br>0111 reserved<br>1000 reserved<br>1001 reserved<br>1010 reserved<br>1011 reserved<br>1100 reserved<br>1101 reserved<br>1110 reserved<br>1111 reserved |

### 8.3.3.1.9 Auxiliary Clock 1 Divider Configuration Register (CGM_AC1_DC)

Address 0xC3FE_038C Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DE0 | 0 | 0 | 0 | DIV0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 8-11. Auxiliary Clock 1 Divider Configuration Register (CGM_AC1_DC)**

This register controls the auxiliary clock 1 divider.

**Table 8-12. CGM_AC1_DC field descriptions**

| Field | Description |
|---|---|
| DE0 | **Divider 0 Enable**<br>0 Disable auxiliary clock 1 divider 0<br>1 Enable auxiliary clock 1 divider 0 |
| DIV0 | **Divider 0 Division Value** — The resultant eMIOS0 clock will have a period DIV0 + 1 times that of auxiliary clock 1. If the DE0 is set to 0 (Divider 0 is disabled), any write access to the DIV0 field is ignored and the eMIOS0 clock remains disabled. |

### 8.3.3.1.10 Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)

Address 0xC3FE_0390 Access: User read, Supervisor read/write, Test read/write

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

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

**Figure 8-12. Auxiliary Clock 2 Select Control Register (CGM_AC2_SC)**

This register is used to select the current clock source for the following clocks:

- undivided: unused
- divided by auxiliary clock 2 divider 0: eMIOS1 clock

**Table 8-13. Auxiliary Clock 2 Select Control Register (CGM_AC2_SC) Field Descriptions**

| Field | Description |
|---|---|
| SELCTL | **Auxiliary Clock 2 Source Selection Control** — This value selects the current source for auxiliary clock 2.<br>0000 div. 16 MHz int. RC osc.<br>0001 div. 4-16 MHz ext. xtal osc.<br>0010 secondary PLL<br>0011 primary PLL/2<br>0100 reserved<br>0101 reserved<br>0110 reserved<br>0111 reserved<br>1000 reserved<br>1001 reserved<br>1010 reserved<br>1011 reserved<br>1100 reserved<br>1101 reserved<br>1110 reserved<br>1111 reserved |

### 8.3.3.1.11 Auxiliary Clock 2 Divider Configuration Register (CGM_AC2_DC)

Address 0xC3FE_0394 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DE0 | 0 | 0 | 0 | DIV0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 8-13. Auxiliary Clock 2 Divider Configuration Register (CGM_AC2_DC)**

This register controls the auxiliary clock 2 divider.

**Table 8-14. CGM_AC2_DC field descriptions**

| Field | Description |
|---|---|
| DE0 | **Divider 0 Enable**<br>0 Disable auxiliary clock 2 divider 0<br>1 Enable auxiliary clock 2 divider 0 |
| DIV0 | **Divider 0 Division Value** — The resultant eMIOS1 clock will have a period DIV0 + 1 times that of auxiliary clock 2. If the DE0 is set to 0 (Divider 0 is disabled), any write access to the DIV0 field is ignored and the eMIOS1 clock remains disabled. |

### 8.3.3.1.12 Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)

Address 0xC3FE_0398 Access: User read, Supervisor read/write, Test read/write

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

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

**Figure 8-14. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)**

This register is used to select the current clock source for the following clocks:

- undivided: unused
- divided by auxiliary clock 3 divider 0: QuadSPI clock

See Figure 8-22 for details.

**Table 8-15. Auxiliary Clock 3 Select Control Register (CGM_AC3_SC) Field Descriptions**

| Field | Description |
|---|---|
| SELCTL | **Auxiliary Clock 3 Source Selection Control** — This value selects the current source for auxiliary clock 3.<br>0000 system clock<br>0001 primary PLL<br>0010 secondary PLL<br>0011 reserved<br>0100 reserved<br>0101 reserved<br>0110 reserved<br>0111 reserved<br>1000 reserved<br>1001 reserved<br>1010 reserved<br>1011 reserved<br>1100 reserved<br>1101 reserved<br>1110 reserved<br>1111 reserved |

### 8.3.3.1.13 Auxiliary Clock 3 Divider Configuration Register (CGM_AC3_DC)

Address 0xC3FE_039C Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DE0 | 0 | 0 | 0 | DIV0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 8-15. Auxiliary Clock 3 Divider Configuration Register (CGM_AC3_DC)**

This register controls the auxiliary clock 3 divider.

**Table 8-16. CGM_AC3_DC field descriptions**

| Field | Description |
|---|---|
| DE0 | **Divider 0 Enable**<br>0 Disable auxiliary clock 3 divider 0<br>1 Enable auxiliary clock 3 divider 0 |
| DIV0 | **Divider 0 Division Value** — The resultant QuadSPI clock will have a period DIV0 + 1 times that of auxiliary clock 3. If the DE0 is set to 0 (Divider 0 is disabled), any write access to the DIV0 field is ignored and the QuadSPI clock remains disabled. |

### 8.3.3.1.14 Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

Address 0xC3FE_03A0 Access: User read, Supervisor read/write, Test read/write

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

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

**Figure 8-16. Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)**

This register is used to select the current clock source for the following clocks:

- undivided: unused
- divided by auxiliary clock 4 divider 0: DCULite clock

**Table 8-17. CGM_AC4_SC field descriptions**

| Field | Description |
|---|---|
| SELCTL | **Auxiliary Clock 4 Source Selection Control** — This value selects the current source for auxiliary clock 4.<br>0000 div. 16 MHz int. RC osc.<br>0001 div. 4-16 MHz ext. xtal osc.<br>0010 secondary PLL<br>0011 primary PLL/2<br>0100 reserved<br>0101 reserved<br>0110 reserved<br>0111 reserved<br>1000 reserved<br>1001 reserved<br>1010 reserved<br>1011 reserved<br>1100 reserved<br>1101 reserved<br>1110 reserved<br>1111 reserved |

#### 8.3.3.1.15 Auxiliary Clock 4 Divider Configuration Register (CGM_AC4_DC)

Address 0xC3FE_03A4 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DE0 | 0 | 0 | 0 | DIV0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 8-17. Auxiliary Clock 4 Divider Configuration Register (CGM_AC4_DC)**

This register controls the auxiliary clock 4 divider.

**Table 8-18. CGM_AC4_DC field descriptions**

| Field | Description |
|---|---|
| DE0 | **Divider 0 Enable**<br>0 Disable auxiliary clock 4 divider 0<br>1 Enable auxiliary clock 4 divider 0 |
| DIV0 | **Divider 0 Division Value** — The resultant DCULite clock will have a period DIV0 + 1 times that of auxiliary clock 4. If the DE0 is set to 0 (Divider 0 is disabled), any write access to the DIV0 field is ignored and the DCULite clock remains disabled. |

## 8.3.4 Functional description

### 8.3.4.1 System clock generation

Figure 8-18 shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see MC_ME documentation for more details), and the MC_RGM provides the safe clock request (see MC_RGM documentation for more details). The safe clock request forces the selector to select the 16 MHz int. RC osc. as the system clock and to ignore the system clock select.

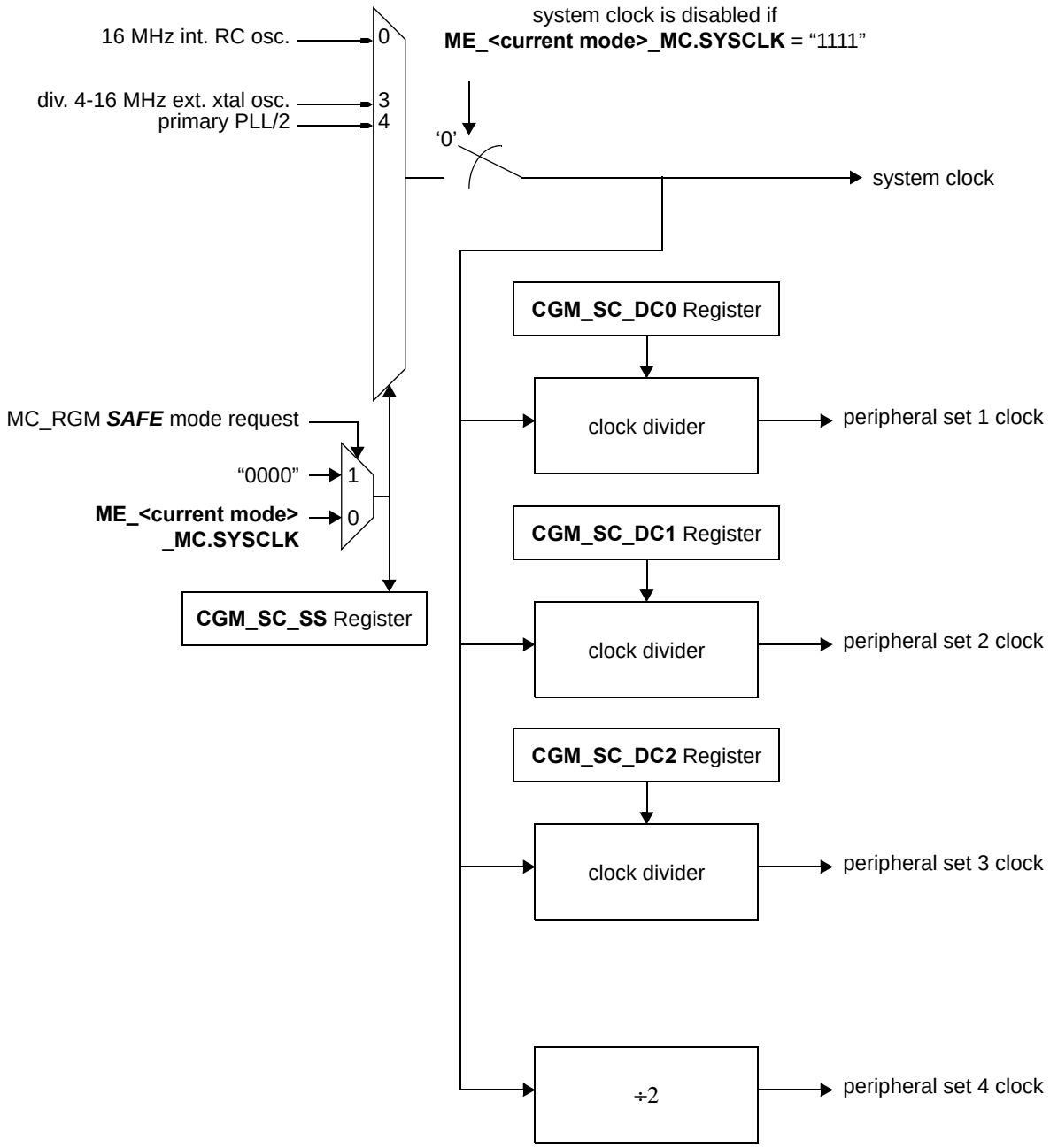



**Figure 8-18. MC_CGM system clock generation overview**

### 8.3.4.1.1 System clock source selection

During normal operation, the system clock selection is controlled

- on a ***SAFE*** mode or reset event, by the MC_RGM
- otherwise, by the MC_ME

#### 8.3.4.1.2 System clock disable

During the ***STOP*** and ***TEST*** modes, the system clock can be disabled by the MC_ME.

#### 8.3.4.1.3 System clock dividers

The MC_CGM generates the following derived clocks from the system clock that are used as the reference clocks for their associated peripherals:

- peripheral set 1 clock - controlled by the **CGM_SC_DC0** register
- peripheral set 2 clock - controlled by the **CGM_SC_DC1** register
- peripheral set 3 clock - controlled by the **CGM_SC_DC2** register
- peripheral set 4 clock - fixed at ÷2

#### 8.3.4.1.4 Auxiliary clock generation

Figure 8-19 (and those following) shows the block diagram of the auxiliary clock generation logic. See:

- Section 8.3.3.1.6, “Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)
- Section 8.3.3.1.8, “Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)
- Section 8.3.3.1.12, “Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)
- Section 8.3.3.1.14, “Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)

### 8.3.4.1.5 Auxiliary clock dividers

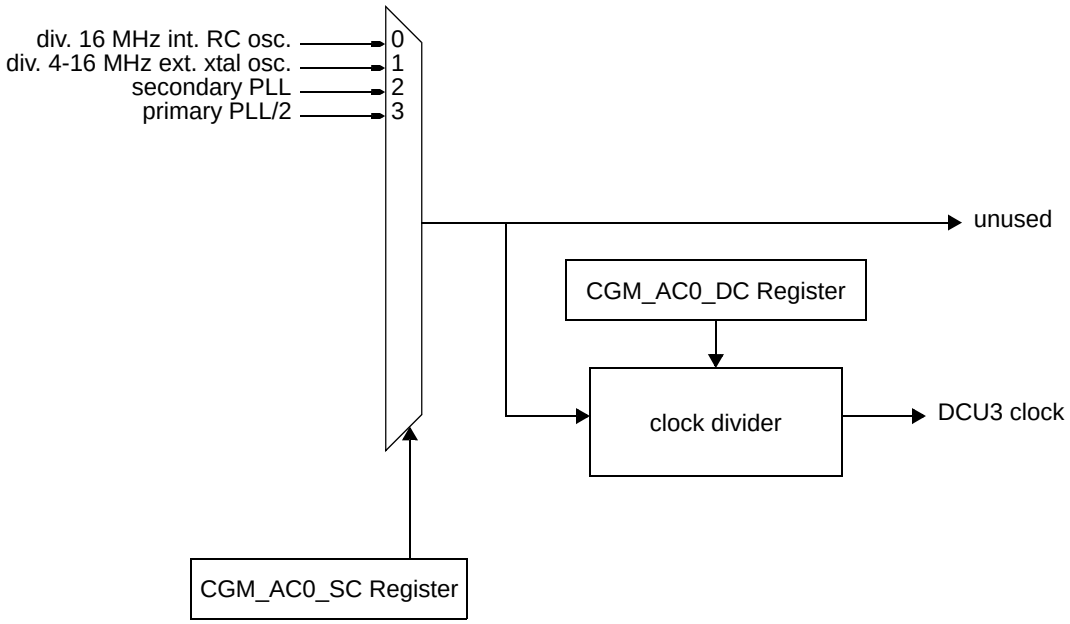



**Figure 8-19. MC_CGM auxiliary clock 0 generation overview**

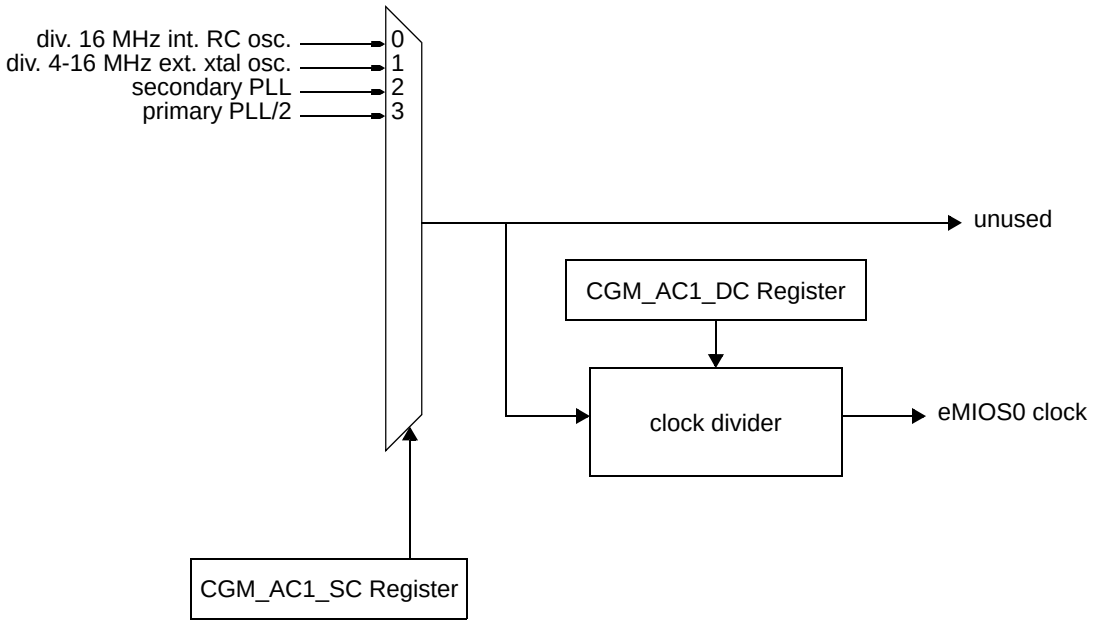



**Figure 8-20. MC_CGM Auxiliary Clock 1 Generation Overview**

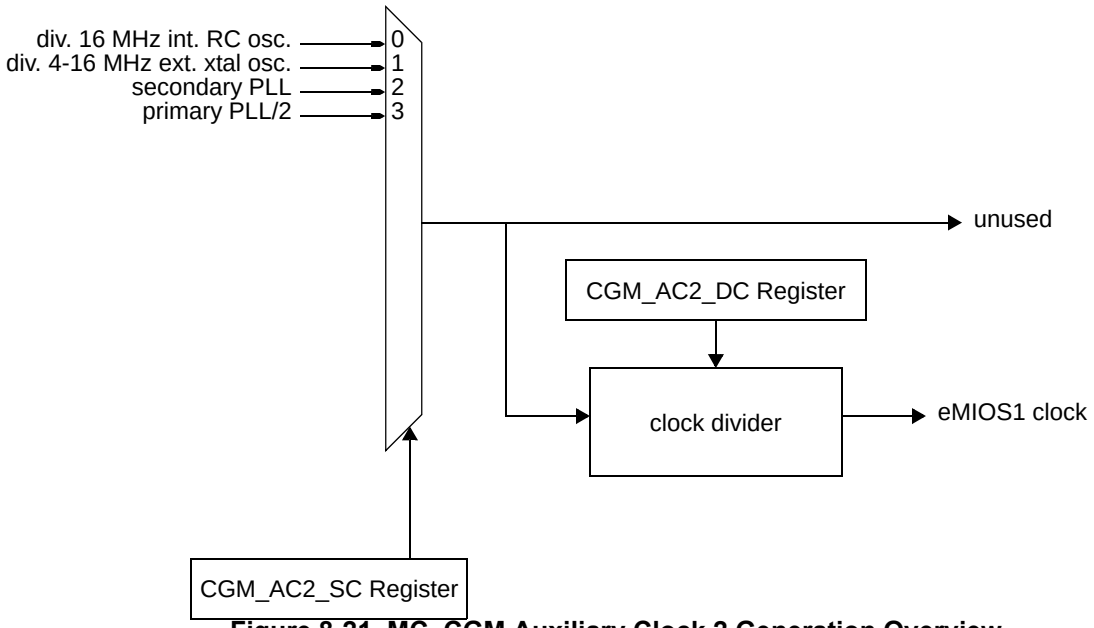



**Figure 8-21. MC_CGM Auxiliary Clock 2 Generation Overview**

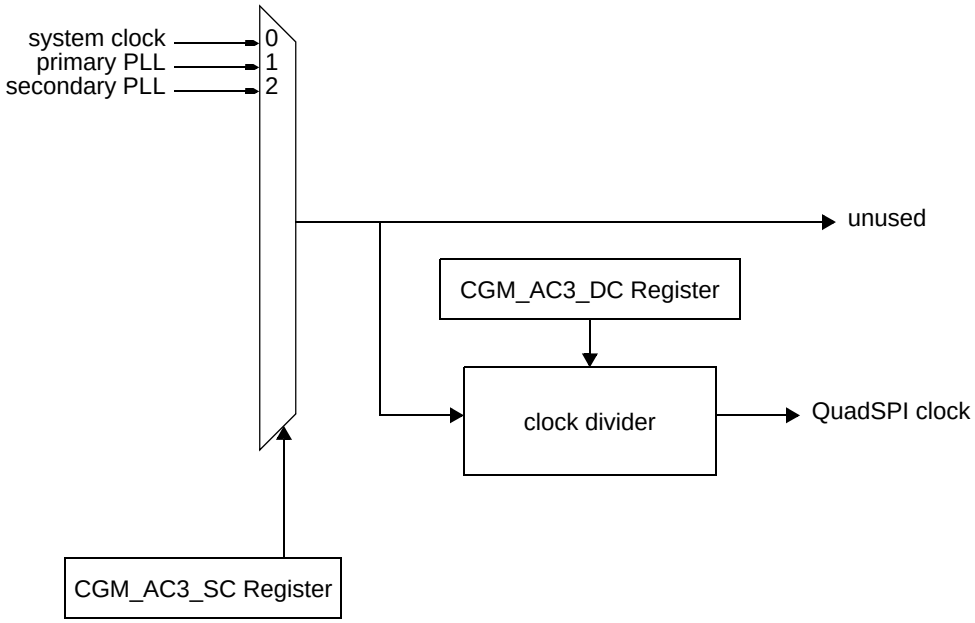



**Figure 8-22. MC_CGM Auxiliary Clock 3 Generation Overview**

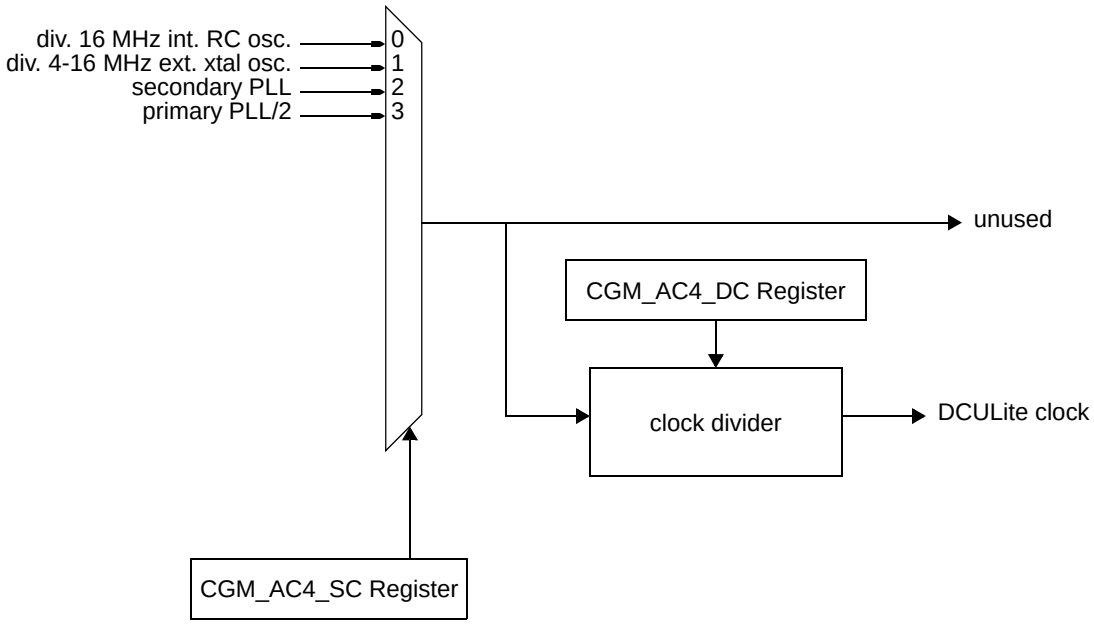



**Figure 8-23. MC_CGM Auxiliary Clock 4 Generation Overview**

### 8.3.4.2 Dividers Functional Description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- Section 8.3.3.1.5, “System Clock Divider Configuration Registers (CGM_SC_DC0…3)
- Section 8.3.3.1.7, “Auxiliary Clock 0 Divider Configuration Register (CGM_AC0_DC)
- Section 8.3.3.1.9, “Auxiliary Clock 1 Divider Configuration Register (CGM_AC1_DC)
- Section 8.3.3.1.11, “Auxiliary Clock 2 Divider Configuration Register (CGM_AC2_DC)
- Section 8.3.3.1.13, “Auxiliary Clock 3 Divider Configuration Register (CGM_AC3_DC)
- Section 8.3.3.1.15, “Auxiliary Clock 4 Divider Configuration Register (CGM_AC4_DC)

The reset value of all counters is ‘1’. If a divider has its **DE** bit in the respective configuration register set to ‘0’ (the divider is disabled), any value in its **DIVn** field is ignored.

### 8.3.4.3 DRAM Controller Clock

For correct operation, the DRAM controller requires two clocks: the system clock and 2x system clock. Since the only clock source available on the device that can provide this is the FMPLL0, the DRAM Controller can only operate when FMPLL0 is selected as the system clock. This is not a limitation because in practice FMPLL0 is the only clock that can provide operating frequencies high enough for the DRAM controller.

### 8.3.4.4 Output Clock Multiplexing

The MC_CGM contains a multiplexing function for a number of clock sources which can then be used as output clock sources. The selection is done via the **CGM_OCDS_SC** register.

### 8.3.4.5 Output Clock Division Selection

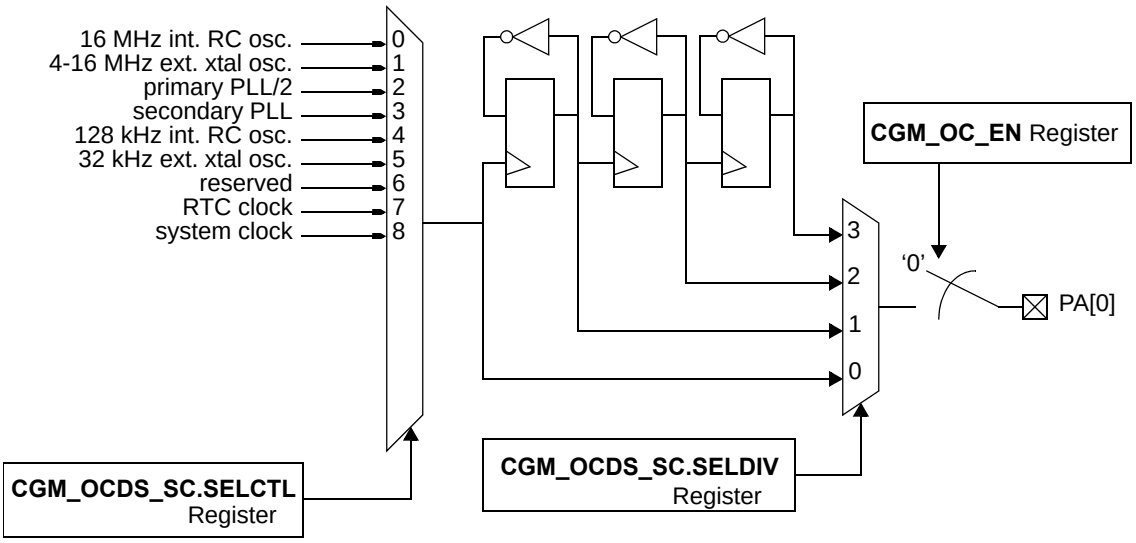



**Figure 8-24. MC_CGM Output Clock Multiplexer and PA[0] Generation**

The MC_CGM provides the following output signals for the output clock generation:

- PA[0] (see Figure 8-24). This signal is generated by using one of the 3-stage ripple counter outputs or the selected signal without division. The non-divided signal is not guaranteed to be 50% duty cycle by the MC_CGM.

the MC_CGM also has an output clock enable register (see Section 8.3.3.1.2, “Output Clock Enable Register (CGM_OC_EN)”) which contains the output clock enable/disable control bit.

## 8.4 Oscillators

### 8.4.1 Pierce oscillator (FXOSC)

#### 8.4.1.1 Introduction

The Pierce oscillator (FXOSC) module provides a robust, low-noise and low-power clock source. The module is operated from the internally generated $V_{DDE_B}$ supply rail (3.3 V) and require the minimum number of external components. It is designed for optimal start-up margin with typical crystal oscillators.

#### 8.4.1.2 Features

The FXOSC contains circuitry to dynamically control current gain in the output amplitude. This ensures a signal with low harmonic distortion, low power and good noise immunity.

- High noise immunity due to input hysteresis
- Low RF emissions with peak-to-peak swing limited dynamically
- Transconductance (gm) sized for optimum start-up margin for typical oscillators
- Dynamic gain control eliminates the need for external current limiting resistor
- Integrated resistor eliminates the need for external bias resistor in loop controlled Pierce mode.
- Low power consumption:
  - Operates from externally provided power $V_{DDE_B}$ (3.3 V)
  - Amplitude control limits power
- Clock monitor

#### 8.4.1.3 Modes of operation

Two modes of operation exist:

1. Loop controlled Pierce (LCP) oscillator
2. External square wave mode

#### 8.4.1.4 Block diagram

Figure 8-25 shows a block diagram of the FXOSC.

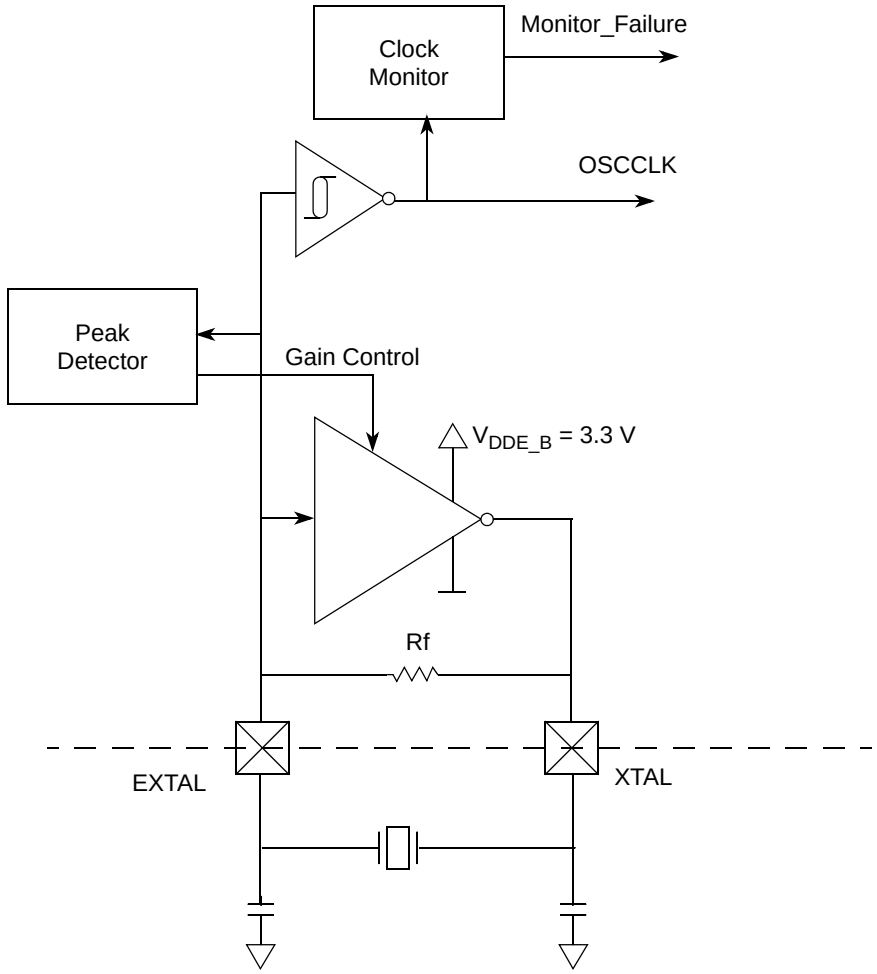



**Figure 8-25. FXOSC block diagram**

### 8.4.1.5 External signal description

This section lists and describes the signals that connect off chip

#### 8.4.1.5.1 $V_{DDE_B}$ and $V_{SS}$ — operating and ground voltage pins

Theses pins provides operating voltage ($V_{DDE_B}$) and ground ($V_{SS}$) for the FXOSC circuitry. This allows the supply voltage to the FXOSC to use an independent bypass capacitor.

#### 8.4.1.5.2 EXTAL and XTAL — input and output pins

These pins provide the interface for either a crystal or a 3.3 V CMOS compatible clock to control the internal clock generator circuitry. EXTAL is the external clock input or the input to the crystal oscillator amplifier. XTAL is the output of the crystal oscillator amplifier. The MCU internal system clock is derived from the EXTAL input frequency. In full stop mode (PSTP = 0), the EXTAL pin is pulled down by an internal resistor of typical 200 kΩ.

**NOTE**

This low-power loop-controlled circuit is not suited for overtone resonators and crystals.

Freescale recommends an evaluation of the layout and performance of the application board and chosen resonator or crystal by the resonator or crystal supplier. The low-power nature of the circuit should be noted when performing this evaluation since the impact of stray capacitance and external disturbances can be larger than seen with higher power oscillators.

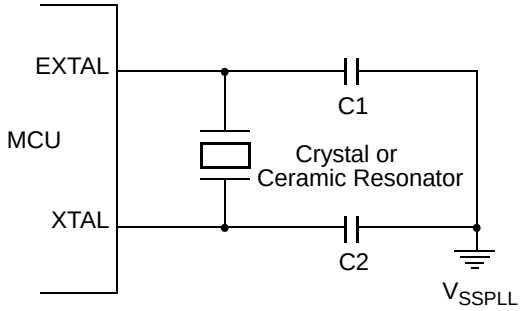



**Figure 8-26. Loop controlled Pierce oscillator connections (LCP mode selected)**

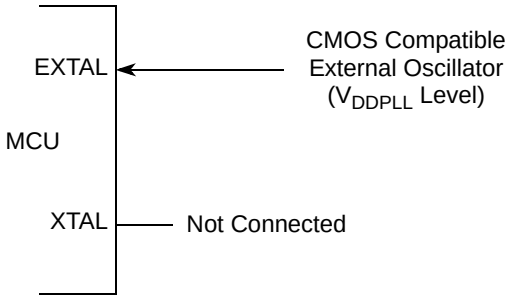



**Figure 8-27. External clock connections**

### 8.4.1.6 Memory map and register definition

The FXOSC does not contain any configuration registers. The oscillator is enabled and disabled using the Mode Entry module.

### 8.4.1.7 Functional description

The FXOSC module has control circuitry to maintain the crystal oscillator circuit voltage level to an optimal level which is determined by the amount of hysteresis being used and the maximum oscillation range.

The oscillator block has two external pins, EXTAL and XTAL. The oscillator input pin, EXTAL, is intended to be connected to either a crystal or an external clock source. The XTAL pin is an output signal that provides crystal circuit feedback.

A buffered EXTAL signal becomes the internal clock. To improve noise immunity, the oscillator is powered by the VDDPLL and VSSPLL power supply pins.

#### 8.4.1.7.1 Gain control

A closed loop control system is utilized whereby the amplifier is modulated to keep the output waveform sinusoidal and to limit the oscillation amplitude. The output peak to peak voltage will be kept above twice the maximum hysteresis level of the input buffer.

#### 8.4.1.7.2 Clock monitor

The clock monitor circuit is based on an internal RC time delay so that it can operate without any MCU clocks. If no OSCCLK edges are detected within this RC time delay, the clock monitor indicates failure which asserts self-clock mode or generates a system reset depending on the state of SCME bit. If the clock monitor is disabled or the presence of clocks is detected no failure is indicated.The clock monitor function is enabled/disabled by the CME control bit, described in the MC_CGM chapter.

## 8.4.2 External crystal oscillator (SXOSC)

### 8.4.2.1 Features

- External crystal oscillator (SXOSC) digital interface
- Oscillator powerdown control and status
- Oscillator clock available interrupt
- Oscillator bypass mode
- Output clock division factors ranging from 1,2,3....32

### 8.4.2.2 Functional description

The crystal oscillator circuit includes an internal oscillator driver and an external crystal circuitry. It can be used as a reference clock to specific modules depending on system needs.

The crystal oscillator is controlled by the OSC_CTL register. The OSCON bit controls the powerdown while S_OSC bit provides the oscillator clock available status.

After system reset, the oscillator is put to power down state and software has to switch on when required. Whenever the crystal oscillator is switched on from off state, OSCCNT counter starts and when it reaches the value EOCV[7:0]*512, oscillator clock is made available to the system. Also an interrupt pending bit I_OSC of OSC_CTL register is set. An interrupt will be generated if the interrupt mask bit M_OSC is set.

The oscillator circuit can be bypassed by writing OSCBYP bit to OSC_CTL register to ‘1’. This bit can only be set by the software. System reset is needed to reset this bit. In this bypass mode, the output clock has the same polarity as external clock applied on EXTAL32 pin and the oscillator status is forced to ‘1’. The bypass configuration is independent of the powerdown mode of the oscillator.

The table below shows the truth table of different configurations of oscillator.

**Table 8-19. Truth table of crystal oscillator**

| ENABLE | BYP | XTAL32 | EXTAL32 | CK_OSCM | OSC MODE |
|---|---|---|---|---|---|
| 0 | 0 | No crystal, High Z | No crystal, High Z | 0 | Power down, IDDQ |
| x | 1 | Ext clock | x | EXTAL32 | Bypass, OSC Disabled |
| 1 | 0 | Crystal | Crystal | EXTAL32 | Normal, OSC Enabled |
| | | Gnd | Ext clock | EXTAL32 | Normal, OSC Enabled |

The crystal oscillator clock can be further divided by a configurable factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by the OSCDIV[4:0] bits of OSC_CTL register.

### 8.4.2.3 Register description

Address offset: 0x0000 Base Address: 0xC3FE0040

Reset value: 0b00000000_10000000_00000000_00000000

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OSCBYP | reserved | | | | | | | EOCV | | | | | | | |
| rs | r | | | | | | | rw | | | | | | | |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M_OSC | reserved | | | OSCDIV | | | | I_OSC | reserved | | | | | S_OSC | OSCON |
| rw | r | | | rw | | | | rc | r | | | | | r | rw |

**Table 8-20. SXOSC Crystal Oscillator Control Register (OSC_CTL)**

**Note:** OSC32 is by default always ON, but can be configured OFF in standby by writing OSCON bit.

**Table 8-21. OSC_CTL field descriptions**

| Field | Description |
|---|---|
| Bit 0 | **OSCBYP**: Crystal Oscillator bypass<br>This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit.<br>0: Oscillator output is used as root clock.<br>1: EXTAL32 is used as root clock. |
| Bits 1-7 | Reserved |

**Table 8-21. OSC_CTL field descriptions**

| | |
|---|---|
| Bits 8-15 | **EOCV[7:0]**: End of Count Value<br>These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV[7:0]*512, oscillator available interrupt request is generated. The reset value of this field depends on the device specification. The OSCCNT counter will be kept under reset if oscillator bypass mode is selected. |
| Bit 16 | **M_OSC**: Crystal oscillator clock interrupt mask<br>0: Crystal oscillator clock interrupt is masked.<br>1: Crystal oscillator clock interrupt is enabled. |
| Bits 17-18 | Reserved |
| Bits 19-23 | **OSCDIV[4:0]**: Crystal oscillator clock division factor<br>These bits specify the crystal oscillator output clock division factor. The output clock is divided by the factor OSCDIV+1. |
| Bit 24 | **I_OSC**: Crystal oscillator clock interrupt<br>This bit is set by hardware when OSCCNT counter reaches the count value EOCV[7:0]*512. It is cleared by software by writing ‘1’.<br>0: No oscillator clock interrupt occurred.<br>1: Oscillator clock interrupt pending. |
| Bits 25-29 | Reserved |
| Bit 30 | **S_OSC**: Crystal oscillator statusl<br>0: Crystal oscillator output clock is not stable.<br>1: Crystal oscillator is providing a stable clock. |
| Bit 31 | **OSCON**: Crystal oscillator powerdown control<br>0: Crystal oscillator is switched off.<br>1: Crystal oscillator is switched on. |

**Note:** OSC_CTL register is writable only in supervisor mode.

## 8.4.3 SIRC digital interface

### 8.4.3.1 Introduction

The SIRC digital interface controls the internal low power 128 kHz RC oscillator (SIRC). It holds control and status registers accessible for application.

### 8.4.3.2 Slow Internal RC Oscillator (128 kHz)

The SIRC provides a low frequency ($f_{SIRC}$) clock in the range of tens of kHz requiring less current consumption. This clock can be used as reference clock when a fixed base time is required for specific modules.

The SIRC is always on in all device modes.

The SIRC clock can be further divided by a configurable division factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by the LPRCDIV[4:0] bits of SIRC_CTL register.

After a power-on reset, the SIRC is trimmed using a factory test value stored in test flash memory. However, after a power-on reset the test flash memory value is not visible at SIRC_CTL[LPRCTRIM] and this field shows a value of zero. Therefore, one should be aware that the SIRC_CTL[LPRCTRIM] does not reflect the current trim value until someone has written to this field. Particular attentionshould be paid to this feature when a read-modify-write operation is initiated on SIRC_CTL, because a SIRCTRIM value of zero may be unintentionally written back and this may alter the SIRC frequency. In this case, the SIRC should be calibrated using the CMU or it should be made sure that you only write to the upper 16 bits of this SIRC_CTL.

In this oscillator, two's complement trimming method is implemented. Thus, the trimming code increases from -16 to 15. As the trimming code increases, the internal time constant increases and frequency reduces. Please refer to device datasheet for average frequency variation of the trimming step.

### 8.4.3.3 Register description

Address offset: 0x0000 Base Address: 0xC3FE_0080

Reset value: 0b00000000_00000000_00000011_00000000

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | LPRCTRIM | | | | |
| r | | | | | | | | | | | rw | | | | |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | LPRCDIV | | | | | reserved | | | S_LPRC | reserved | | | |
| r | | | rw | | | | | r | | | r | r | | | |

**Figure 8-28. Slow Internal RC Control Register (SIRC_CTL)**

**Table 8-22. Slow Internal RC Control Register (SIRC_CTL) field descriptions**

| Field | Description |
|---|---|
| Bits 0-10 | Reserved |
| Bits 11-15 | **LPRCTRIM[4:0]**: Low power RC trimming bits<br>**Note:** All configurations cannot be used. Please refer to the device datasheet. |
| Bits 16-18 | Reserved |
| Bits 19-23 | **LPRCDIV[4:0]**: Low Power RC clock division factor<br>These bits specify the low power RC oscillator output clock division factor. The output clock is divided by the factor LPRCDIV+1. |
| Bits 24-26 | Reserved |
| Bits 27 | **S_LPRC**: Low Power RC clock status<br>0: LPRC is not providing a stable clock.<br>1: LPRC is providing a stable clock. |
| Bits 28-31 | Reserved |

**Note:** SIRC_CTL register is writable only in supervisor mode.

## 8.4.4 FIRC digital interface

### 8.4.4.1 Introduction

The FIRC digital interface controls the fast internal 16 MHz RC oscillator (FIRC). It holds control and status registers accessible for application.

### 8.4.4.2 Functional Description (16 MHz)

The main RC oscillator provides a high-frequency ($f_{FIRC}$) clock. This clock can be used to accelerate the exit from reset and wakeup sequence from low power modes of the system. It is controlled by the MC_ME module based on the current device mode. The clock source status is updated in S_RC bit of ME_GS register. Please refer to MC_ME specification for further details.

The FIRC clock can be further divided by a configurable division factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by the RCDIV[4:0] bits of the FIRC_CTL register.

After a power-on reset, the FIRC is trimmed using a factory test value stored in test flash memory. However, after a power-on reset the test flash memory value is not visible at FIRC_CTL[RCTRIM] and this field shows a value of zero. Therefore, be aware that the FIRC_CTL[RCTRIM] does not reflect the current trim value until you have written to this field. Pay particular attention to this feature when you initiate a read-modify-write operation on FIRC_CTL, because a RCTRIM value of zero may be unintentionally written back and this may alter the FIRC frequency. In this case, you should calibrate the FIRC using the CMU or be sure that you only write to the upper 16 bits of this FIRC_CTL.

In this oscillator, two's complement trimming method is implemented. So the trimming code increases from -32 to 31. As the trimming code increases, the internal time constant increases and frequency reduces. Please refer to device datasheet for average frequency variation of the trimming step.

During the standby mode entry process, the RC oscillator is controlled based on the RCON bit of the ME_STANDBY_MC register. The is the last step in the standby entry sequence. On any system wake-up event, the device exits standby mode and switches on the RC oscillator. The actual powerdown status of the RC oscillator when the device is in standby is provided by the S_RC_STDBY bit of the FIRC_CTL register.

### 8.4.4.3 Register Description

Address offset: 0x0000 Base Address: 0xC3FE_0060

Reset value: 0b00000000_00000000_00000000_00000000

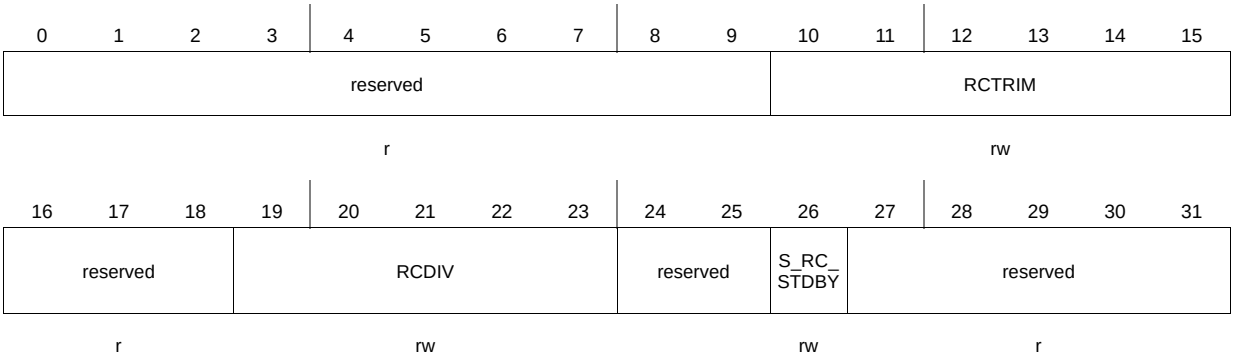



**Table 8-23. FIRC Oscillator Control Register (FIRC_CTL)**

**Table 8-24. FIRC Oscillator Control Register (FIRC_CTL) field descriptions**

| Field | Description |
|---|---|
| Bits 0-9 | Reserved |
| Bits 10-15 | **RCTRIM[5:0]**: Low power RC trimming bits<br>**Note:** Not all configurations can be used. Please refer to the device datasheet. |
| Bits 16-18 | Reserved |
| Bits 19-23 | **RCDIV[4:0]**: Low Power RC clock division factor<br>These bits specify the low power RC oscillator output clock division factor. The output clock is divided by the factor LPRCDIV+1. |
| Bits 24-25 | Reserved |
| Bits 26 | **S_RC_STDBY**: MRC oscillator powerdown status in standby mode<br>This bit specifies whether MRC oscillator is powered down or not during standby mode entry. This bit can be cleared by writing ’ 1‘.<br>0: MRC is not switched off during standby.<br>1: MRC is switched off during standby. |
| Bits 28-31 | Reserved |

**Note:** FIRC_CTL register is writable only in supervisor mode.

## 8.5 Frequency-modulated phase-locked loop (FMPLL)

### 8.5.1 Introduction

This section describes the features and functions of the two independent FMPLL modules (FMPLL_0 and FMPLL_1) implemented in MPC5645S.

### 8.5.2 Overview

The FMPLLs enable the user to generate high speed system clocks from a common 4 MHz to 120 MHz input clock. Further, the FMPLLs support programmable frequency modulation of the system clock. The PLL multiplication factor, output clock divider ratio are all software configurable.

**NOTE**

The user must take care not to program device with frequency higher than allowed (no hardware check).

The FMPLL's block diagram is shown in Figure 8-29.

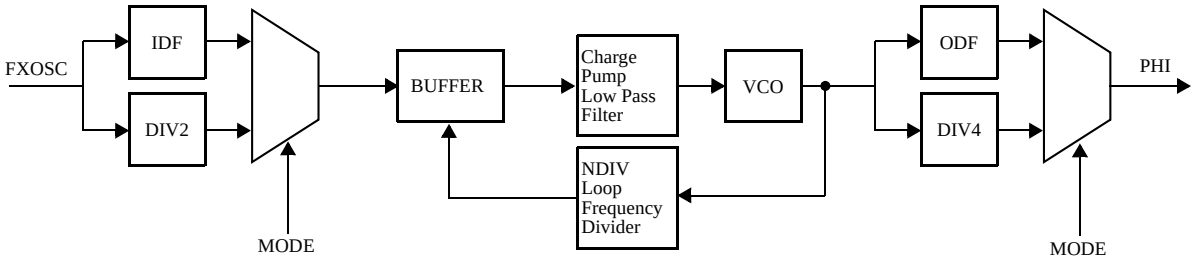



**Figure 8-29. FMPLL block diagram**

### 8.5.3 Features

Each FMPLL has the following major features:

- Input clock frequency from 4 MHz to 120 MHz
- Voltage controlled oscillator (VCO) range from 256 MHz to 512 MHz
- Frequency modulated PLL
  - Modulation enabled/disabled through software
  - Triangle wave modulation
- Programmable modulation depth
  - ±0.25% to ±4% deviation from center spread frequency
  - –0.5% to –8% deviation from down spread frequency
  - Programmable modulation frequency dependent on reference frequency
- Self-clocked mode (SCM) operation
- Five available modes
  - Normal mode

— Progressive clock switching
— Normal Mode with SSCG
— Powerdown mode
— 1:1 mode (FMPLL0 only)

### 8.5.4 Memory map[1]

Table 8-25 shows the memory map locations. Addresses are given as offsets of the module base address.

**Table 8-25. FMPLL Memory Map**

| Address | Register | Access | Location |
|---|---|---|---|
| Base:<br>0xC3FE00A0 (FMPLL0)<br>0xC3FE00C0 (FMPLL1) | | | |
| 0x0000 | Control register (CR) | R/W | on page 243 |
| 0x0004 | Modulation register (MR) | Special | on page 246 |

### 8.5.5 Register description

The PLL operation is controlled by two registers. Those registers can only be written in supervisor mode.

#### 8.5.5.1 Control register (CR)

Offset 0x0000 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | IDF | | | | ODF | | 0 | NDIV | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | en_pll_sw | mode | unlock_once | 0 | i_lock | s_lock | pll_fail_mask | pll_fail_flag | 0 |
| W | | | | | | | | | | | | w1c | | | w1c | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 8-26. Control register (CR)**

1.FMPLL_x are mapped through the ME_CGM Register Slot

**Table 8-27. CR field descriptions**

| Field | Description |
|---|---|
| IDF | The value of this field sets the PLL Input division factor as described in Table 8-28. The reset value is set during integration. |
| ODF | The value of this field sets the PLL Output division factor as described in *Table 8-29.* The reset value is set during integration. |
| NDIV | The value of this field sets the PLL Loop division factor as described in *Table 8-30.* The reset value is set during integration. |
| en_pll_sw | This bit is used to enable progressive clock switching. After the PLL locks, the PLL output initially is divided by 8 then progressively divides down until divide by 1.<br>0 => progressive clock switching disabled<br>1 => progressive clock switching enabled<br>**Note:** The PLL output should not be used if a non-changing clock is needed (such as for serial communications) until the division has finished |
| mode | This bit is used to activate the 1:1 Mode. |
| unlock_once | This bit is a sticky indication of PLL loss of lock condition. Unlock_once is set when the PLL loses lock. Whenever the PLL reacquires lock, unlock_once remains set. Only a power-on reset can clear this bit. |
| i_lock | This bit is set by hardware whenever there is a lock/unlock event.It is cleared by software, writing 1. |
| s_lock | This bit is an indication of whether the PLL has acquired lock.<br>0 => PLL unlocked<br>1 => PLL locked |
| pll_fail_mask | This bit is used to mask the pll_fail output.<br>0 => pll_fail not masked<br>1 => pll_fail masked |
| pll_fail_flag | This bit is asynchronously set by hardware whenever a loss of lock event occurs while PLL is switched on. It is cleared by software, writing 1. |

**Table 8-28. Input divide ratios**

| IDF | Input divide ratio ($R_{INP}$) |
|---|---|
| 0000 | Divide by 1 |
| 0001 | Divide by 2 |
| 0010 | Divide by 3 |
| 0011 | Divide by 4 |
| 0100 | Divide by 5 |
| 0101 | Divide by 6 |
| 0110 | Divide by 7 |
| 0111 | Divide by 8 |
| 1000 | Divide by 9 |
| 1001 | Divide by 10 |
| 1010 | Divide by 11 |

**Table 8-28. Input divide ratios (continued)**

| IDF | Input divide ratio ($R_{INP}$) |
|---|---|
| 1011 | Divide by 12 |
| 1100 | Divide by 13 |
| 1101 | Divide by 14 |
| 1110 | Divide by 15 |
| 1111 | Clock Inhibit |

**Table 8-29. Output divide ratios**

| ODF | Output divide ratio ($R_{OUT}$) |
|---|---|
| 00 | Divide by 2 |
| 01 | Divide by 4 |
| 10 | Divide by 8 |
| 11 | Divide by 16 |

**Table 8-30. Loop divide ratios**

| NDIV | Loop divide ratio ($R_{LOOP}$) |
|---|---|
| 0000000-0011111 | NA |
| 0100000 | Divide by 32 |
| 0100001 | Divide by 33 |
| 0100010 | Divide by 34 |
| ... | ... |
| 1011111 | Divide by 95 |
| 1100000 | Divide by 96 |
| 1100001-1111111 | NA |

### 8.5.5.2 Modulation register (MR)

Offset 0x0004 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | STRB_BYPASS | 0 | SPRD_SEL | MOD_PERIOD | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FM_EN | INC_STEP | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | –[1] | 0 | 0 | 0 | 0 | 0 |

**Table 8-31. Modulation register (MR)**

**Table 8-32. MR field descriptions**

| Field | Description |
|---|---|
| 0<br>STRB_BYPASS | Strobe bypass<br>The STRB_BYPASS signal is used to bypass the STRB signal used inside PLL to latch the correct values for control bits (INC_STEP, MOD_PERIOD and SPRD_SEL).<br>0 = STRB is used to latch PLL modulation control bits<br>1 = STRB is bypassed. In this case control bits need to be static. The control bits must be changed only when PLL is in power down mode. |
| 2<br>SPRD_SEL | Spread type selection<br>The SPRD_SEL control the spread type in Frequency Modulation mode.<br>0 = Center SPREAD<br>1 = Down SPREAD |
| 3-15<br>MOD_PERIOD | Modulation period<br>The MOD_PERIOD field is the binary equivalent of the value modperiod derived from following formula:<br>$\text{modperiod} = \frac{f_{ref}}{4 \times f_{mod}}$<br>where:<br>*fref:* represents the frequency of the feedback divider<br>*fmod*: represents the modulation frequency |

**Table 8-32. MR field descriptions (continued)**

| Field | Description |
|---|---|
| 16<br>FM_EN | Frequency Modulation Enable<br>The FM_EN enables the frequency modulation.<br>0 = Frequency Modulation disabled<br>1= Frequency Modulation enabled |
| 17-31<br>INC_STEP | Increment step<br>The INC_STEP field is the binary equivalent of the value incstep derived from following formula:<br>$$\text{incstep} = \text{round}\left(\frac{(2^{15}-1)\times md \times MDF}{100\times 5\times MODPERIOD}\right)$$<br>where:<br>*md*: represents the peak modulation depth in percentage (Center spread -- pk-pk=+/-md, Downspread -- pk-pk=-2*md)<br>*MDF*: represents the nominal value of loop divider (NDIV in PLL Control Register) |

## 8.5.6 Functional description

### 8.5.6.1 Normal mode

In Normal Mode the PLL inputs are driven by the CR (see Section 8.5.5.1, “Control register (CR)”). This means that, when the PLL is in lock state, the PLL output clock (PHI) is derived by the reference clock (CLKIN) through Equation 8-1:

$$phi = \frac{clkin \cdot R_{LOOP}}{R_{INP} \cdot R_{OUT}}$$

***Eqn. 8-1***

where the value of $R_{LOOP}$, $R_{INP}$, and $R_{OUT}$ are set in the CR and can be derived from Table 8-28, Table 8-29, and Table 8-30. Some examples are given in Table 8-33.

**Table 8-33. Examples of typical PLL settings**

| Crystal Frequency (MHz) | PLL Output Frequency (MHz) | DRAM clock Frequency (MHz) | Register Values | | | VCO Frequency (MHz) |
|---|---|---|---|---|---|---|
| | | | IDF | ODF | NDIV | |
| 8 | 32 | 64 | 0 | 1 | 32 | 256 |
| | 64 | 128 | 0 | 1 | 64 | 512 |
| | 80 | 160 | 0 | 0 | 40 | 320 |
| | 124 | 248 | 0 | 0 | 62 | 496 |

Table 8-33. Examples of typical PLL settings

| Crystal Frequency (MHz) | PLL Output Frequency (MHz) | DRAM clock Frequency (MHz) | Register Values | | | VCO Frequency (MHz) |
|---|---|---|---|---|---|---|
| | | | IDF | ODF | NDIV | |
| 16 | 32 | 64 | 0 | 1 | 32 | 256 |
| | 64 | 128 | 0 | 1 | 64 | 512 |
| | 80 | 160 | 0 | 0 | 40 | 320 |
| | 124 | 248 | 0 | 0 | 62 | 496 |
| | | | | | | |
| 10 | 30 | 60 | 0 | 2 | 48 | 480 |
| | 60 | 120 | 0 | 1 | 48 | 480 |
| | 80 | 160 | 1 | 0 | 64 | 320 |
| | 125 | 250 | 0 | 0 | 50 | 500 |

### 8.5.6.2 Progressive clock switching

Progressive clock switching allows to switch system clock to PLL output clock stepping through different division factors. This means that the current consumption gradually increases and so the voltage regulator has a better response.

This feature can be enabled by programming the en_pll_sw bit in CR. Then, when the input pin pll_select goes high, the output clock ck_pll_div will progressively increase its frequency as described in Table 8-34 and Figure 8-30.

Table 8-34. Progressive clock switching on pll_select rising edge

| Number of PLL output clock cycles | ck_pll_frequency (PLL output clock frequency) |
|---|---|
| 8 | (ck_pll_out frequency)/8 |
| 16 | (ck_pll_out frequency)/4 |
| 32 | (ck_pll_out frequency)/2 |
| onward | (ck_pll_out frequency) |

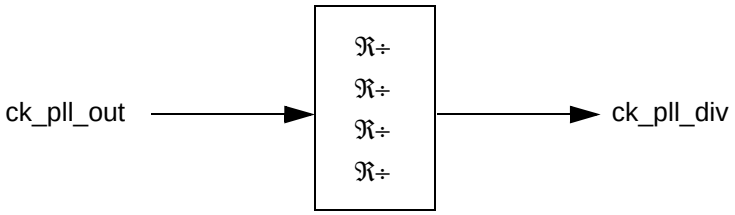



Figure 8-30. Diagram of progressive clock switching

### 8.5.6.3 Normal Mode with frequency modulation

The FMPLL default mode is without frequency modulation enabled. When frequency modulation is enabled, however, two parameters must be set to generate the desired level of modulation: the PERIOD, and the STEP. The modulation waveform is always a triangle wave and its shape is not programmable.

FM modulation shall be activated in two steps:

- **first:** configure the FM modulation characteristics: MOD_PERIOD, INC_STEP.
- **second:** enable the FM modulation by setting the MR[FM_EN] bit. FM modulated mode can be enabled only when PLL is in lock state.

To latch these values inside the PLL, two ways are usable depending on the value of STRB_BYPASS register bit in MR.

If STRB_BYPASS is low, the modulation [parameters are latched in the PLL only when the STRB signal goes high for at least 2 cycles of INFIN clock. The STRB signal is automatically generated in the PLLD when the modulation is enabled (FM_EN goes high) if the PLL is locked (s_lock=1) or when the modulation has been enabled (FM_EN=1) and PLL enters in lock state (s_lock goes high).

If STRB_BYPASS is high, the STRB signal is bypassed. In this case, control bits (MOD_PERIOD[12:0], INC_STEP[14:0], SPREAD_CONTROL) need to be static or hardwired to constant values. The control bits must be changed only when the PLL is in power down mode.

The modulation depth in % is

$$\text{ModulationDepth} = \left(\frac{100 \times 5 \times \text{INCSTEPxMODPERIOD}}{(2^{15} - 1) \times \text{MDF}}\right)$$

**NOTE**

The user must ensure that the product of INCTEP and MODPERIOD is less than ($2^{15}$-1).

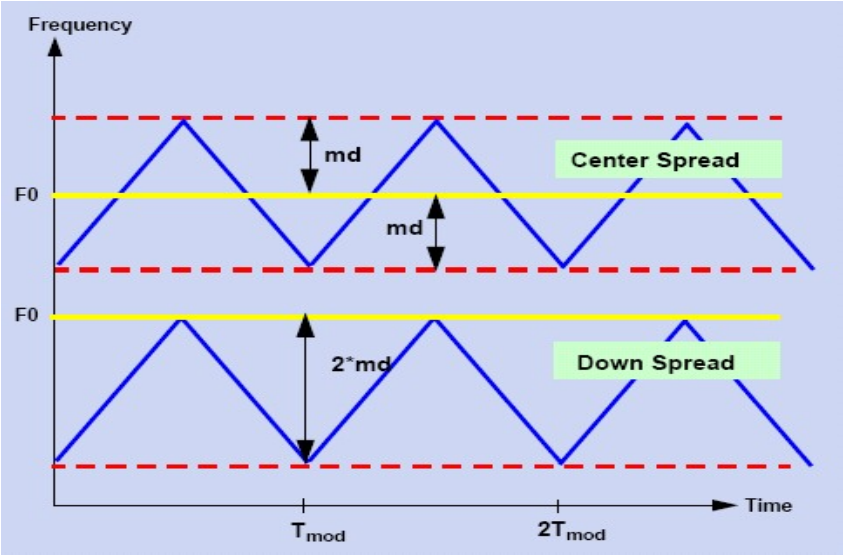



**Figure 8-31. PLL frequency modulation modes**

### 8.5.6.4 Powerdown mode

The PLL can be switched off when not required to achieve lower consumption by programming the registers ME_x_MC register on MC_ME module.

### 8.5.6.5 1:1 Mode (FMPLL0 only)

1:1 Mode is set by asserting the mode bit in CR (see Section 8.5.5.1, “Control register (CR)”). An external input signal (mode_en) has been provided to disable this feature. If mode_en is tied to 0, the mode bit in CR is disabled and there is no way to activate 1:1 Mode.

In 1:1 Mode the inputs of the PLL are driven by CR and MR, but the division factors and the modulation parameters have no influence on the output clock. In fact the dividers and the SSCG control are bypassed inside the PLL. The PLL output clock (phi) frequency is determined by the following relation:

$$\text{phi} = \frac{\text{clkin}}{2}$$

## 8.5.7 Recommendations

To avoid any unpredictable behavior of the PLL clock, it is recommended to respect the following guidelines:

- The PLL VCO frequency should reside in the range 256 MHz to 512 MHz. Care is required when programming the multiplication and division factors to respect this requirement.
- The user must change the multiplication, division factors only when the PLL output clock is not selected as system clock. MOD_PERIOD, INC_STEP, SPREAD_SEL bits should be modified before activating the FM modulated mode. Then strobe has to be generated to enable the new settings. If STRB_BYP is set to 1 then MOD_PERIOD, INC_STEP and SPREAD_SEL can be modified only when PLL is in power down mode.
- Use progressive clock switching.

# 8.6 Clock Monitor Unit (CMU)

## 8.6.1 Introduction

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves two purposes. The main task is to permanently supervise the integrity of the device's system clock sources, i.e., crystal oscillator FXOSC, FIRC, and FMPLL0. If FMPLL0 leaves an upper or lower frequency boundary or the crystal oscillator fails it can detect and forward this kind of event towards the mode and clock management unit. The clock management unit in turn can then switch to a safe mode where it uses a safe fallback clock source such as an on-chip RC oscillator, reset the device or generate the interrupt according to the system needs.

It can also monitor external crystal oscillator clock which must be greater than the internal RC clock divided by a division factor given by RCDIV[1:0] of CMU_CSR register and generates a system clock transition request or an interrupt when enabled.

The second task of the CMU is to provide a frequency meter, which allows to measure the frequency of one clock source vs. a reference clock. This is useful to allow the calibration of the on-chip RC oscillator(s), as well as being able to correct/calculate the time deviation of a counter which is clocked by the RC oscillator.

**NOTE**

The CMU does not monitor SXOSC, SIRC, or FMPLL_1.

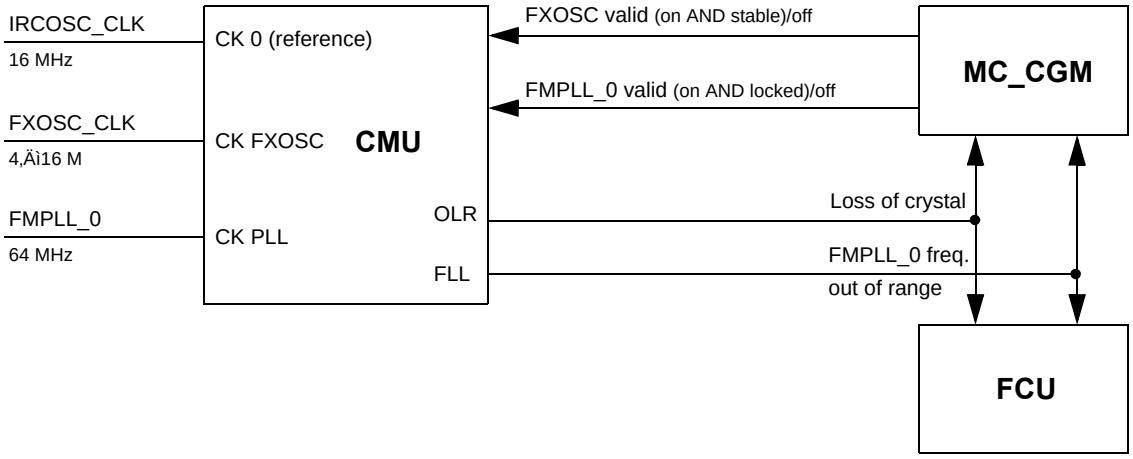



**Figure 8-32. CMU component interaction**

## 8.6.2 Main features

- RC oscillator frequency measurement
- External oscillator clock monitoring with respect to CK_FIRC/n clock
- PLL clock frequency monitoring with respect to CK_FIRC/4 clock
- Event generation for various failures detected inside monitoring unit

## 8.6.3 Block diagram

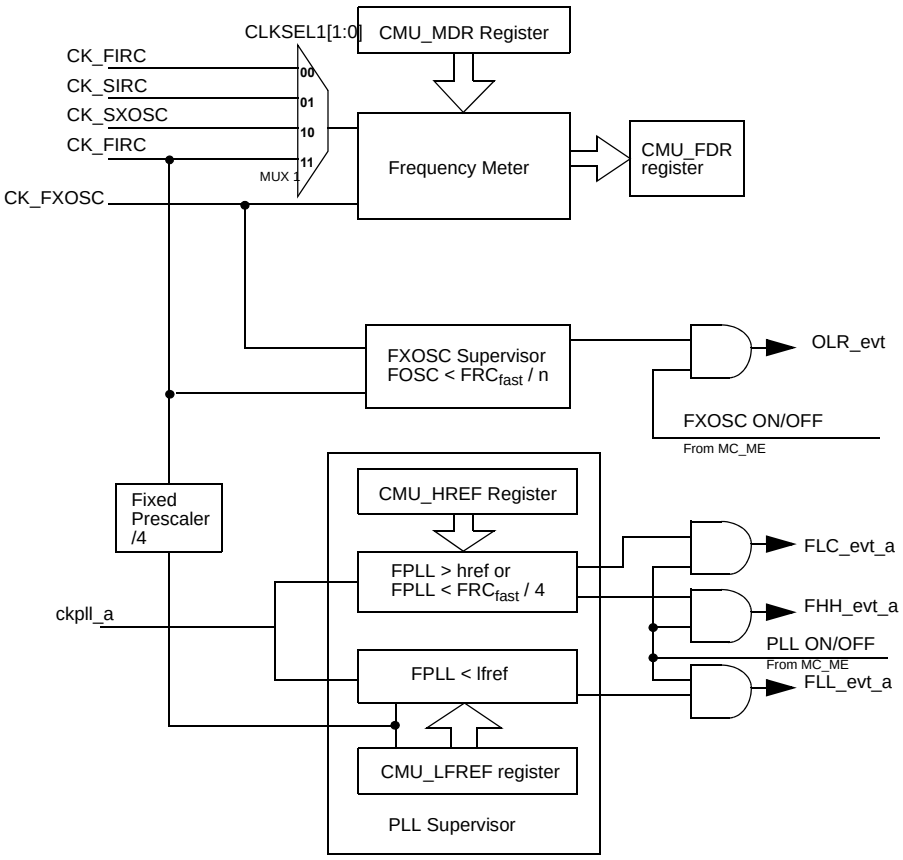



**Figure 8-33. Clock Monitor Unit diagram**

## 8.6.4 Memory map and register description

The memory map of the CMU is shown in the following table.

**Table 8-35. RC Digital Interface Register Set - Base address 0xC3FE_0100**

| Address Offset | Register Name | Location |
|---|---|---|
| 0x00 | Control Status Register (CMU_CSR) | on page 253 |
| 0x04 | Frequency Display Register (CMU_FDR) | on page 254 |
| 0x08 | High Frequency Reference Register FMPLL0(CMU_HFREFR_A) | on page 254 |
| 0x0C | Low Frequency Reference Register FMPLL0(CMU_LFREFR_A) | on page 255 |
| 0x10 | Interrupt Status Register (CMU_ISR) | on page 255 |
| 0x14 | reserved | — |
| 0x18 | Measurement Duration Register (CMU_MDR) | on page 256 |

### 8.6.4.1 Control Status Register (CMU_CSR)

Address offset: 0x00

Reset value: 0x00000006

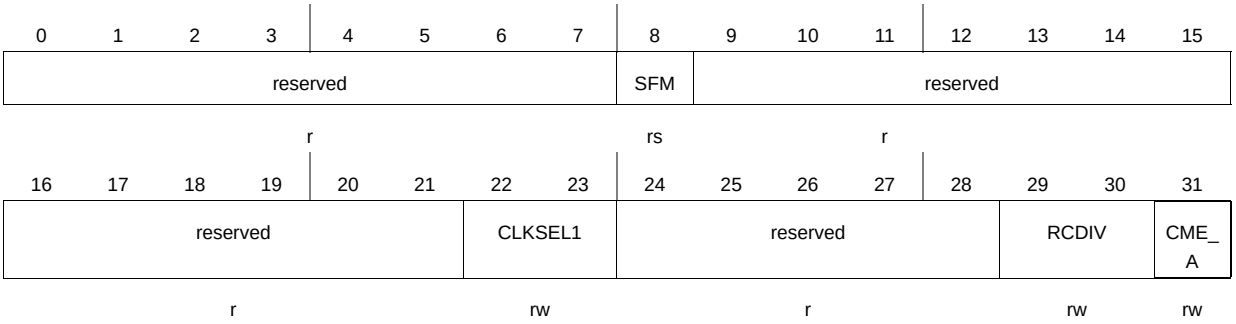



**Table 8-36. Control Status Register (CMU_CSR)**

**Table 8-37. Control Status Register (CMU_CSR) field descriptions**

| Field | Description |
|---|---|
| 8<br>SFM | Start frequency measure<br>The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR register.<br>0: Frequency measurement is completed or not yet started.<br>1: Frequency measurement is not completed. |
| 22-23<br>CLKSEL1 | RC Oscillator(s) selection bit<br>CLKSEL1 selects the clock to be measured by the frequency meter.<br>00: CK_FIRC is selected.<br>01: CK_SIRC is selected.<br>10: CK_SXOSC crystal Oscillator clock is selected.<br>11: CK_FIRC is selected. |
| 29-30<br>RCDIV[1:0] | RC clock division factor<br>These bits specify the RC clock division factor. The output clock is $CK_IRC_{fast}$ divided by the factor $2^{RCDIV}$. This output clock is used to compare with CK_FXOSC for crystal clock monitor feature.The clock division coding is as follows.<br>00: Clock divided by 1 (No division)<br>01: Clock divided by 2<br>10: Clock divided by 4<br>11: Clock divided by 8 |
| 31<br>CME_A | FMPLL0 clock monitor enable<br>0: FMPLL0 monitor is disabled.<br>1: FMPLL0 monitor is enabled. |

### 8.6.4.2 Frequency Display Register (CMU_FDR)

Address offset: 0x04

Reset value: 0x00000000

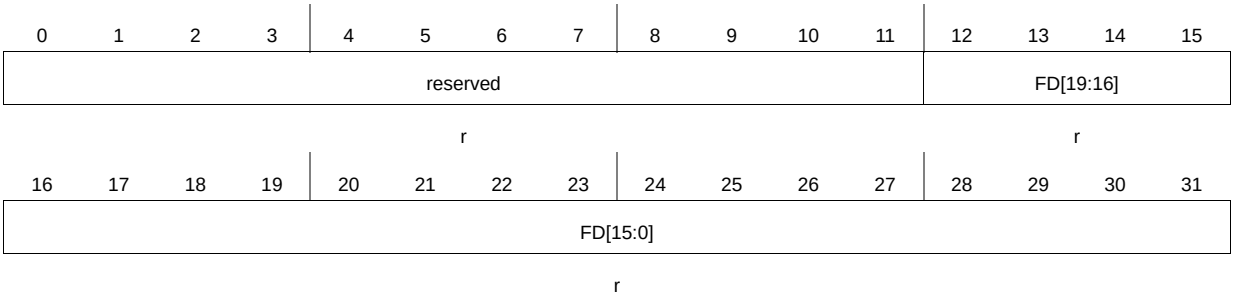



**Table 8-38. Frequency Display Register (CMU_FDR)**

**Table 8-39. Frequency Display Register (CMU_FDR) field descriptions**

| Field | Description |
|---|---|
| 12-31<br>FD | Measured frequency bits<br>This register displays the measured frequency FRC with respect to FOSC. The measured value is given by the following formula: FRC = (FOSC * MD) / n, Where n is the value in CMU_FDR register |

### 8.6.4.3 High Frequency Reference Register FMPLL0 (CMU_HFREFR)

Address offset: 0x08

Reset value: 0x00000FFF




**Table 8-40. High Frequency Reference Register FMPLL0**

**Table 8-41. High Frequency Reference Register FMPLL0 field descriptions**

| Field | Description |
|---|---|
| 20-31<br>HFREF | High Frequency reference value<br>These bits determine the high reference value for the FMPLL0 clock. The reference value is given by: (HFREF[11:0]/16) * ($FRC_{fast}$/4). |

### 8.6.4.4 Low Frequency Reference Register FMPLL0 (CMU_LFREFR)

Address offset: 0x0C

Reset value: 0x00000000

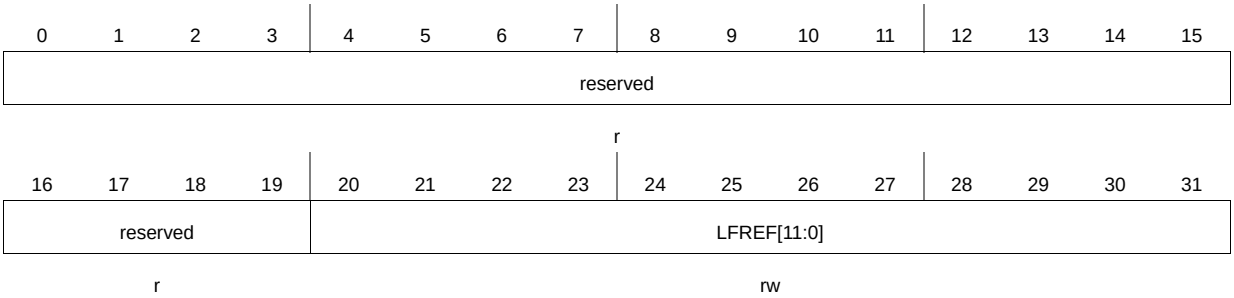



**Table 8-42. Low Frequency Reference Register FMPLL0**

**Table 8-43. Low Frequency Reference Register FMPLL0 field descriptions**

| Field | Description |
|---|---|
| 20-31<br>LFREF | Low Frequency reference value<br>These bits determine the low reference value for the FMPLL0. The reference value is given by: (LFREF[11:0]/16) * ($FRC_{fast}$/4). |

### 8.6.4.5 Interrupt Status Register (CMU_ISR)

Address offset: 0x10

Reset value: 0x00000000

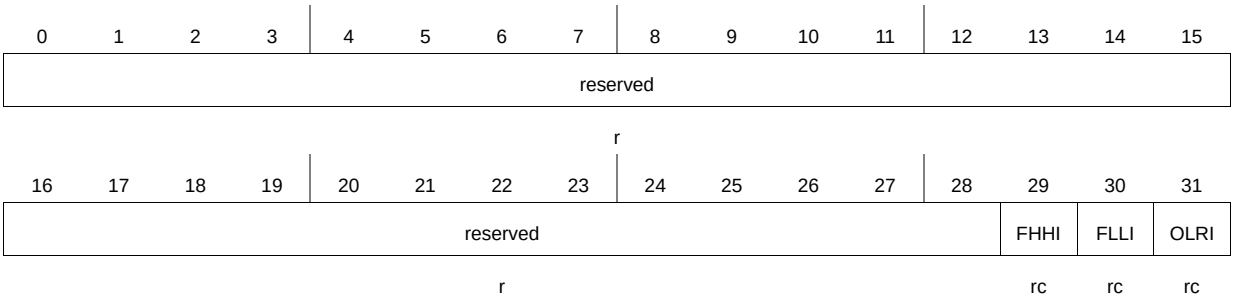



**Table 8-44. Interrupt Status Register (CMU_ISR)**

**Table 8-45. Interrupt Status Register (CMU_ISR) field descriptions**

| Field | Description |
|---|---|
| 29<br>FHHI | FMPLL0 Clock frequency higher than high reference interrupt<br>This bit is set by hardware when CK_FMPLL frequency becomes higher than HFREF value and CK_FMPLL is ‘ON’ as signalled by the MC_ME. It can be cleared by software by writing ‘1’.<br>0: No FHH event.<br>1: FHH event is pending. |

**Table 8-45. Interrupt Status Register (CMU_ISR) field descriptions (continued)**

| | |
|---|---|
| 30<br>FLLI | FMPLL0 Clock frequency less than low reference event<br>This bit is set by hardware when CK_FMPLL frequency becomes lower than LFREF value and CK_FMPLL is 'ON' as signalled by the MC_ME. It can be cleared by software by writing '1'.<br>0: No FLL event.<br>1: FLL event is pending. |
| 31<br>OLRI | Oscillator frequency less than RC frequency event<br>This bit is set by hardware when the frequency of CK_FXOSC is less than CK_FIRC/$2^{RCDIV}$ frequency and CK_FXOSC is 'ON' as signalled by the MC_ME. It can be cleared by software by writing '1'.<br>0: No OLR event.<br>1: OLR event is pending. |

### 8.6.4.6 Measurement Duration Register (CMU_MDR)

Address offset: 0x18

Reset value: 0x00000000

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | MD[19:16] | | | |
| r | | | | | | | | | | | | rw | | | |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

**Table 8-46. Measurement Duration Register (CMU_MDR)**

**Table 8-47. Measurement Duration Register (CMU_MDR) field descriptions**

| Field | Description |
|---|---|
| 12-31<br>MD | Measurement duration bits<br>This register displays the measured duration in term of IRC clock cycles. This value is loaded in the frequency meter downcounter. When SFM bit is set to '1', downcounter starts counting. |

## 8.6.5 Functional description

The names of the clocks involved in this block have the following meaning:

- CK_FXOSC: clock coming from the external crystal oscillator.
- CK_SIRC: clock coming from the low frequency internal RC oscillator.
- CK_FIRC: clock coming from the high frequency internal RC oscillator.
- CK_PLL: clock coming from the PLL.
- FOSC: frequency of external crystal oscillator clock.
- FRCslow: frequency of low frequency internal RC oscillator.

- FRCfast: frequency of high frequency internal RC oscillator.
- FPLL: frequency of FMPLL clock.

### 8.6.5.1 Crystal clock monitor

If FOSC is smaller than FRCfast divided by $2^{RCDIV}$. bits of CMU_CSR and the CK_FXOSC is ‘ON’ as signalled by the MC_ME then:

- An event pending bit OLRI in CMU_ISR is set.
- A failure event OLR is signalled to the MC_RGM which in turn can automatically switch to a safe fallback clock and generate an interrupt or a reset.

**NOTE**

F<sub>XOSC</sub> must be greater than F<sub>FIRC</sub> / 2<sup>RCDIV</sup> by at least 0.5 MHz in order to guarantee correct XOSC monitoring.

### 8.6.5.2 PLL clock monitor

The PLL clock CK_PLL frequency can be monitored by programming CME bit of CMU_CSR register to ‘1’. CK_PLL monitor starts as soon as CME bit is set. This monitor can be disabled at any time by writing CME bit to ‘0’.

If CK_PLL frequency (FPLL) is greater than a reference value determined by the HFREF[11:0] bits of CMU_HFREFR and the CK_PLL is ‘ON’ as signalled by the MC_ME then

- An event pending bit FHHI in CMU_ISR is set,
- A failure event is signalled to the MC_RGM and Fault Collection Unit which in turn can generate an interrupt or a reset.

If FPLL is less than a reference clock frequency (FRC/4) and the CK_PLL is ‘ON’ as signalled by the MC_ME then an event pending bit FLCI in CMU_ISR will be set.

If FPLL is less than a reference value determined by the LFREF[11:0] bits of CMU_LFREFR and the CK_PLL is ‘ON’ as signalled by the MC_ME then

- An event pending bit FLLI in CMU_ISR is set,
- A failure event FLL is signalled to the MC_RGM which can generate an interrupt or a reset.

**NOTE**

F<sub>PLL</sub> must be greater than F<sub>FIRC</sub> / 4 by at least 0.5 MHz in order to guarantee correct PLL monitoring.

### 8.6.5.3 Frequency meter

The purpose of frequency meter is to calibrate the internal RC oscillator (CK_IRC) using a known frequency.

**Hint:** This value can then be stored into the flash so that application software can reuse it later on.

The reference clock will be always the FXOSC. The Frequency Meter returns a precise value of CK_32K, CK_FIRC or CK_SIRC according to the CLKSEL1 bit value. The measure starts when the SFM (Start Frequency Measure) bit in the CMU_CSR is set to ‘1’. The measurement duration is given by the CMU_MDR register in numbers of clock cycles of the selected clock source with a width of 20 bits. The SFM bit is reset to ‘0’ by the hardware once the frequency measurement is done and the count is loaded in the CMU_FDR. The frequency FRC can be derived from the value loaded in the CMU_FDR register as follows:

$$FRC = (FOSC * MD) / n, \qquad \textit{Eqn. 8-2}$$

Where n is the value in the CMU_FDR register and MD is the value in the CMU_MDR.

Frequency Meter by default evaluates CK_FIRC, but the software can swap to CK_SIRC or CK_SXOSC by programming the CLKSEL bits in the CMU_CSR register.The CKON bits indicate which is the actual clock at the output of the multiplexer MUX1

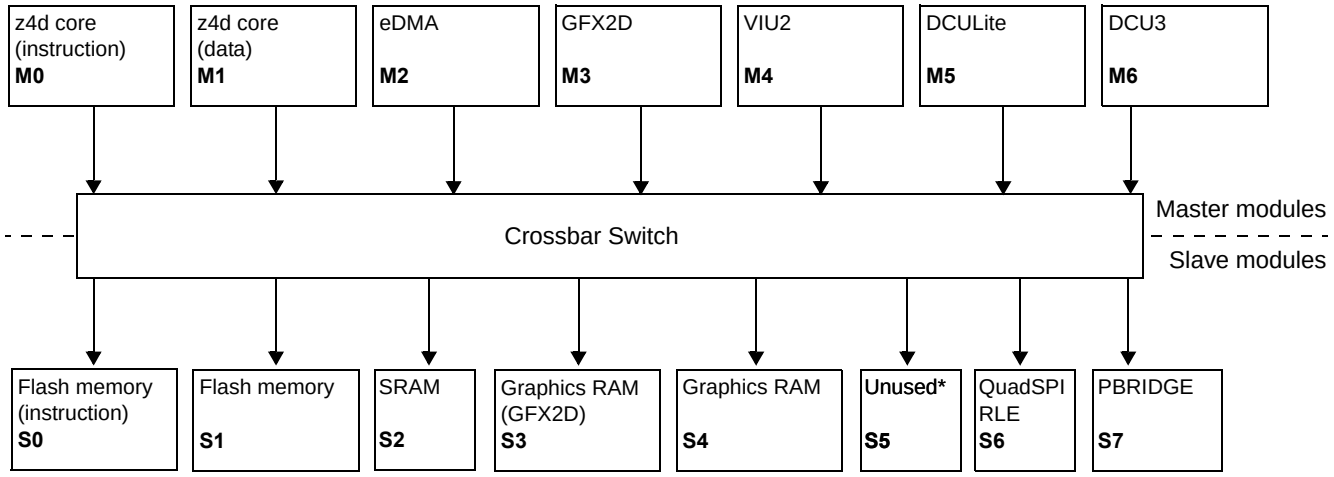
# Chapter 9
# Crossbar Switch (XBAR)

## 9.1 Information specific to this device

This section presents device-specific parameterization, customization, and feature availability information not specifically referenced in the remainder of this chapter.

### 9.1.1 Device-specific block diagram

Figure 9-1 shows the simplified block diagram for the MPC5645S.




* DRAM accesses do not go through the XBAR.

**Figure 9-1. Device-specific block diagram**

### 9.1.2 XBAR Master ID Numbers

The XBAR module refers to individual masters by their physical master port number. Other modules such as MPU and SWT use the master IDs. This is different from the physical master port numbers which are in Figure 9-1 above.

**Table 9-1. XBAR Master ID Numbers**

| Master | Master ID Number | XBAR Port ID |
|---|---|---|
| CPU (instruction) | 0 | 0 |
| CPU (Data) | 0 | 1 |
| eDMA | 2 | 2 |
| GFX2D | 3 | 3 |
| DCULite | 4 | 4 |

Table 9-1. XBAR Master ID Numbers (continued)

| Master | Master ID Number | XBAR Port ID |
|---|---|---|
| VIU2 | 5 | 5 |
| DCU3 | 6 | 6 |
| Nexus ID | 8 | 1 |

### 9.1.3 Unsupported features

- The **max_halt_request** input
- The use of the Alternate Master Priority Registers
- The use of the Alternate General Purpose Control Registers
- Registers for slave 5
- Registers for master 7

## 9.2 Introduction

### 9.2.1 Overview

This section provides an overview of the generic multi-layer AHB crossbar switch (XBAR). The purpose of the XBAR is to concurrently support up to eight simultaneous connections between master ports and slave ports. The XBAR supports a 64-bit address bus width and almost any data bus width at all master and slave ports.

### 9.2.2 Features

The XBAR has the ability to gain control of all the slave ports and prevent any masters from making accesses to the slave ports. This feature is useful when the user wishes to turn off the clocks to the system and needs to ensure that no bus activity will be interrupted.

The XBAR can put each slave port into a low power park mode so that slave port will not dissipate any power transitioning address, control or data signals when not being actively accessed by a master port.

Each slave port can also support multiple master priority schemes. Each slave port has a hardware input which selects the master priority scheme so the user can dynamically change master priority levels on a slave port by slave port basis.

The XBAR will allow for concurrent transactions to occur from any master port to any slave port. It is possible for all master ports and slave ports to be in use at the same time as a result of independent master requests. If a slave port is simultaneously requested by more than one master port, arbitration logic will select the higher priority master and grant it ownership of the slave port. All other masters requesting that slave port will stalled until the higher priority master completes its transactions.

### 9.2.3 Limitations

The XBAR routes bus transactions initiated on the master ports to the appropriate slave ports. There is no provision included to route transactions initiated on the slave ports to other slave ports or to master ports. Simply put, the slave ports do not support the bus request/bus grant protocol, the XBAR assumes it is the sole master of each slave port.

Since the XBAR does not support the bus request/bus grant protocol, if multiple masters are to be connected to a single master port an external arbiter will need to be used. In the case of a single master connecting to a master port the single master's bus grant signal must be tied off in the asserted state.

Each master and slave port is fully AHB-Lite + AMBA V6 extensions compliant. The ports are not fully AHB compliant because the XBAR does not support SPLITs or RETRYs.

### 9.2.4 General Operation

When a master makes an access to the XBAR the access will be immediately taken by the XBAR. If the targeted slave port of the access is available then the access will be immediately presented on the slave port. It is possible to make single clock (zero wait state) accesses through the XBAR. If the targeted slave port of the access is busy or parked on a different master port the requesting master will simply see wait states inserted (**hready** held negated) until the targeted slave port can service the master's request. The latency in servicing the request will depend on each master's priority level and the responding peripheral's access time.

Since the XBAR appears to be just another slave to the master device, the master device will have no knowledge of whether or not it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting it will simply be wait stated.

A master will be given control of the targeted slave port only after a previous access to a different slave port has completed, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when a master has an outstanding request to one slave port that has a long response time, has a pending access to a different slave port, and a lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

Once the master has control of the slave port it is targeting the master will remain in control of that slave port until it gives up the slave port by running an IDLE cycle or by leaving that slave port for its next access. The master could also lose control of the slave port if another higher priority master makes a request to the slave port; however, if the master is running a locked or fixed length burst transfer it will retain control of the slave port until that transfer is completed. Based on the AULB bit in the MGPCR (Master General Purpose Control Register) the master will either retain control of the slave port when doing undefined length incrementing burst transfers or will lose the bus to a higher priority master.

The XBAR will terminate all master IDLE transfers (as opposed to allowing the termination to come from one of the slave busses). Additionally, when no master is requesting access to a slave port the XBAR will drive IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port. When the XBAR is controlling the slave bus (that is, during low power park or halt mode) the **hmaster** field will indicate 4'b0000.

When a slave bus is being IDLEd by the XBAR it can park the slave port on the master port indicated by the PARK bits in the SGPCR (Slave General Purpose Control Register). This can be done in an attempt to save the initial clock of arbitration delay that would otherwise be seen if the master had to arbitrate to gain control of the slave port. The slave port can also be put into low power park mode in attempt to save power.

## 9.3 XBAR registers

This section provides information on XBAR registers.

### 9.3.1 Register summary

There are four registers that reside in each slave port of the XBAR and one register that resides in each master port of the XBAR. These registers are IP bus compliant registers. Read and write transfers both require two IP bus clock cycles. The registers can only be read from and written to in supervisor mode. Additionally, these registers can only be read from or written to by 32-bit accesses.

The registers are fully decoded and an error response is returned if an unimplemented location is accessed within the XBAR.

The slave registers also feature a bit, which when written with a 1, will prevent the registers from being written to again. The registers will still be readable, but future write attempts will have no effect on the registers and will be terminated with an error response.

The memory map for the XBAR program-visible registers is shown in Table 9-2. Table 9-3 shows the XBAR register summary.

**Table 9-2. XBAR registers**

| Address offset | Use | Location |
|---|---|---|
| 0x000 | Master Priority Register for Slave port 0 (MPR0) | on page 264 |
| 0x004 | Reserved | — |
| 0x010 | General Purpose Control Register for Slave port 0 (SGPCR0) | on page 267 |
| 0x014 | Reserved | — |
| 0x100 | Master Priority Register for Slave port 1 (MPR1) | on page 264 |
| 0x104 | Reserved | — |
| 0x110 | General Purpose Control Register for Slave port 1 (SGPCR1) | on page 267 |
| 0x114 | Reserved | — |
| 0x200 | Master Priority Register for Slave port 2 (MPR2) | on page 264 |
| 0x204 | Reserved | — |
| 0x210 | General Purpose Control Register for Slave port 2 (SGPCR2) | on page 267 |
| 0x214 | Reserved | — |
| 0x300 | Master Priority Register for Slave port 3 (MPR3) | on page 264 |
| 0x304 | Reserved | — |
| 0x310 | General Purpose Control Register for Slave port 3 (SGPCR3) | on page 267 |
| 0x314 | Reserved | — |
| 0x400 | Master Priority Register for Slave port 4 (MPR4) | on page 264 |
| 0x404 | Reserved | — |
| 0x410 | General Purpose Control Register for Slave port 4 (SGPCR4) | on page 267 |

**Table 9-2. XBAR registers (continued)**

| Address offset | Use | Location |
|---|---|---|
| 0x414 | Reserved | — |
| 0x500 | Master Priority Register for Slave port 5 (MPR5) | on page 264 |
| 0x504 | Reserved | — |
| 0x510 | General Purpose Control Register for Slave port 5 (SGPCR5) | on page 267 |
| 0x514 | Reserved | — |
| 0x600 | Master Priority Register for Slave port 6 (MPR6) | on page 264 |
| 0x604 | Reserved | — |
| 0x610 | General Purpose Control Register for Slave port 6 (SGPCR6) | on page 267 |
| 0x614 | Reserved | — |
| 0x700 | Master Priority Register for Slave port 7 (MPR7) | on page 264 |
| 0x704 | Reserved | — |
| 0x710 | General Purpose Control Register for Slave port 7 (SGPCR7) | on page 267 |
| 0x714 | Reserved | — |
| 0x800 | General Purpose Control Register for Master port 0 (MGPCR0) | on page 269 |
| 0x900 | General Purpose Control Register for Master port 1 (MGPCR1) | on page 269 |
| 0xA00 | General Purpose Control Register for Master port 2 (MGPCR2) | on page 269 |
| 0xB00 | General Purpose Control Register for Master port 3 (MGPCR3) | on page 269 |
| 0xC00 | General Purpose Control Register for Master port 4 (MGPCR4) | on page 269 |
| 0xD00 | General Purpose Control Register for Master port 5 (MGPCR5) | on page 269 |
| 0xE00 | General Purpose Control Register for Master port 6 (MGPCR6) | on page 269 |
| 0xF00 | General Purpose Control Register for Master port 7 (MGPCR7) | on page 269 |

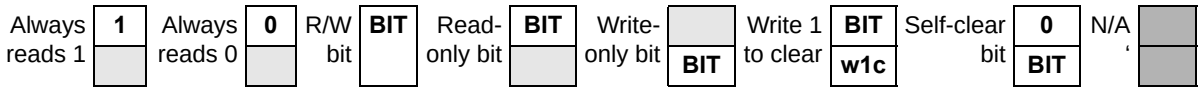



**Figure 9-2. Key to Register Fields**

**Table 9-3. XBAR Register Summary**

| Name | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| MPRn ($BASE + 0x000 + n*0x100) | R | 0 | MSTR_7 | | | 0 | MSTR_6 | | | 0 | MSTR_5 | | | 0 | MSTR_4 | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | MSTR_3 | | | 0 | MSTR_2 | | | 0 | MSTR_1 | | | 0 | MSTR_0 | | |
| | W | | | | | | | | | | | | | | | | |
| SGPCRn ($BASE + 0x010 + n*0x100) | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | RO | HLP | | | | | | | HPE7 | HPE6 | HPE5 | HPE4 | HPE3 | HPE2 | HPE1 | HPE0 |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | ARB | | | | PCTL | | | PARK | | |
| MGPCRn ($BASE + 0x800 + n*0x100 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |

**Table 9-3. XBAR Register Summary (continued)**

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

**Note:** for n = 0 to 7

## 9.3.2 XBAR Register Descriptions

The following paragraphs provide detailed descriptions of the various XBAR registers.

Table 9-4 provides a key to the terms found in XBAR registers.

**Table 9-4. Register Terms**

| Term | Description |
|---|---|
| Gray bit | Unimplemented bit; always reads as zero;writing has no effect |
| **Access** | |
| S | Supervisor mode only |
| — | Supervisor or user mode |
| **Type** | |
| r | Read only; writing to this bit has no effect |
| w | Write only |
| rw | Standard read/write bit. Only software can change a bit's value (other than a hardware reset). |
| rwm | A read/write bit that may be modified by hardware in some fashion other than reset. |
| w1c | A status bit that can be read and cleared by writing a logic 1 |
| slfclr | Self-clearing bit. Writing a 1 has some effect on module, but it always reads as a 0. |
| **Reset** | |
| 0 | Resets to a logic 0 |
| 1 | Resets to a logic 1 |
| u | Unaffected by reset |
| ? | Reset state is unknown. |

### 9.3.2.1 Master Priority Register

The Master Priority Register (MPR) sets the priority of each master port on a per slave port basis and resides in each slave port.

| MPRn | Master Priority Register n | Addr<br>$BASE + 0x000 + n*100<br>Wait State: 0<br>Access: S |
|---|---|---|

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MSTR_7 | | | | MSTR_6 | | | | MSTR_5 | | | | MSTR_4 | | |
| TYPE: | r | rw | rw | rw | r | rw | rw | rw | r | rw | rw | rw | r | rw | rw | rw |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MSTR_3 | | | | MSTR_2 | | | | MSTR_1 | | | | MSTR_0 | | |
| TYPE: | r | rw | rw | rw | r | rw | rw | rw | r | rw | rw | rw | r | rw | rw | rw |
| RESET: | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Note:** for n = 0 to 7

**Figure 9-3. Master Priority Register n**

**Table 9-5. Master Priority Register Descriptions**

| Name | Description | Settings |
|---|---|---|
| Bit 0 | **Master Priority Register Reserved -** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility. | NA |
| MSTR_7 | **Master 7 Priority -** These bits set the arbitration priority for master port 7 on the associated slave port.<br><br>These bits are initialized by hardware reset.<br>The reset value is 111 | 000This master has the highest priority when accessing the slave port.<br><br>111This master has the lowest priority when accessing the slave port. |
| Bit 4 | **Master Priority Register Reserved -** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility. | NA |
| MSTR_6 | **Master 6 Priority -** These bits set the arbitration priority for master port 6 on the associated slave port.<br><br>These bits are initialized by hardware reset.<br>The reset value is 110 | 000This master has the highest priority when accessing the slave port.<br><br>111This master has the lowest priority when accessing the slave port. |
| Bit 8 | **Master Priority Register Reserved -** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility. | NA |
| MSTR_5 | **Master 5 Priority -** These bits set the arbitration priority for master port 5 on the associated slave port.<br><br>These bits are initialized by hardware reset.<br>The reset value is 101 | 000This master has the highest priority when accessing the slave port.<br><br>111This master has the lowest priority when accessing the slave port. |
| Bit 12 | **Master Priority Register Reserved -** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility. | NA |

**Table 9-5. Master Priority Register Descriptions (continued)**

| Name | Description | Settings |
|---|---|---|
| MSTR_4 | **Master 4 Priority -** These bits set the arbitration priority for master port 4 on the associated slave port.<br><br>These bits are initialized by hardware reset.<br>The reset value is 100 | 000This master has the highest priority when accessing the slave port.<br><br>111This master has the lowest priority when accessing the slave port. |
| Bit 16 | **Master Priority Register Reserved -** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility. | NA |
| MSTR_3 | **Master 3 Priority -** These bits set the arbitration priority for master port 3 on the associated slave port.<br><br>These bits are initialized by hardware reset.<br>The reset value is 011 | 000This master has the highest priority when accessing the slave port.<br><br>111This master has the lowest priority when accessing the slave port. |
| Bit 20 | **Master Priority Register Reserved -** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility. | NA |
| MSTR_2 | **Master 2 Priority -** These bits set the arbitration priority for master port 2 on the associated slave port.<br><br>These bits are initialized by hardware reset.<br>The reset value is 010 | 000This master has the highest priority when accessing the slave port.<br><br>111This master has the lowest priority when accessing the slave port. |
| Bit 24 | **Master Priority Register Reserved -** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility. | NA |
| MSTR_1 | **Master 1 Priority -** These bits set the arbitration priority for master port 1 on the associated slave port.<br><br>These bits are initialized by hardware reset.<br>The reset value is 001 | 000This master has the highest priority when accessing the slave port.<br><br>111This master has the lowest priority when accessing the slave port. |
| Bit 28 | **Master Priority Register Reserved -** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility. | NA |
| MSTR_0 | **Master 0 Priority -** These bits set the arbitration priority for master port 0 on the associated slave port.<br><br>These bits are initialized by hardware reset.<br>The reset value is 000 | 000This master has the highest priority when accessing the slave port.<br><br>111This master has the lowest priority when accessing the slave port. |

The Master Priority Register can only be accessed in supervisor mode with 32-bit accesses. Once the RO (Read Only) bit has been set in the slave General Purpose Control Register the Master Priority Register can only be read from, attempts to write to it will have no effect on the MPR and result in an error response.

**NOTE**

No two available master ports may be programmed with the same priority level. Attempts to program two or more available masters with the same priority level will result in an error response and the MPR will not be updated.

### 9.3.2.2 Slave General Purpose Control Register

The Slave General Purpose Control Register (SGPCR) controls several features of each slave port.

The Read Only (RO) bit will prevent any registers associated with this slave port from being written to once set. This bit may be written with 0 as many times as the user desires, but once it is written to a 1 only a reset condition will allow it to be written again.

The Halt Low Priority (HLP) bit will set the priority of the **max_halt_request** input to the lowest possible priority for initial arbitration of the slave ports. By default it is the highest priority. Setting this bit will not effect the **max_halt_request** from attaining highest priority once it has control of the slave ports.

The PCTL bits determine how the slave port will park when no master is actively making a request. The available options are to park on the master defined by the PARK bits, park on the last master to use the slave port, or go into a low power park mode which will force all the outputs of the slave port to inactive states when no master is requesting an access. The low power park feature can result in an overall power savings if a the slave port is not saturated; however, it will force an extra clock of latency whenever any master tries to access it when it is not in use because it will not be parked on any master.

The PARK bits determine which master the slave will park on when no master is making an active request and the **max_halt_request** input is negated. Please use caution to only select master ports that are actually present in the design. If the user programs the PARK bits to a master not present in the current design implementation undefined behavior will result.

**SGPCRn** — **Slave General Purpose Control Register n** — **Addr $BASE + 0x010 + n*100, Wait State: 0, Access: S**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RO | HLP | | | | | | | HPE 7 | HPE 6 | HPE 5 | HPE 4 | HPE 3 | HPE 2 | HPE 1 | HPE 0 |
| TYPE: | rw | rw | r | r | r | r | r | r | rw | rw | rw | rw | rw | rw | rw | rw |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Note: Once the RO bit is written to a 1, only hardware reset will return it to a 0.

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | ARB | | | | PCTL | | | PARK | | |
| TYPE: | r | r | r | r | r | r | rw | rw | r | r | rw | rw | r | rw | rw | rw |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Note: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

**Note:** for n = 0 to 7

**Figure 9-4. Slave General Purpose Control Register n**

**Table 9-6. Slave General Purpose Control Register Descriptions**

| Name | Description | Setting |
|---|---|---|
| RO | **Read Only -** This bit is used to force all of a slave port's registers to be read only. Once written to 1 it can only be cleared by hardware reset.<br><br>This bit is initialized by hardware reset.<br>The reset value is 0 | 0 All this slave port's registers can be written.<br>1 All this slave port's registers are read only and cannot be written (attempted writes have no effect and result in an error response). |
| HLP | **Halt Low Priority -** This bit is used to set the initial arbitration priority of the max_halt_request input.<br><br>This bit is initialized by hardware reset.<br>The reset value is 0 | 0 The max_halt_request input has the highest priority for arbitration on this slave port<br>1 The max_halt_request input has the lowest initial priority for arbitration on this slave port. |
| Bits 2–7 | **Slave General Purpose Control Register Reserved -** These bits are reserved for future expansion. They read as zero and should be written with zero for upward compatibility. | NA |
| HPEx | **High Priority Enable -**These bits are used to enable the mX_high_priority inputs for the respective master.<br><br>These bits are initialized by hardware reset.<br>The reset value is 0 | 0 The mX_high_priority input is disabled on this slave port<br>1 The mX_high_priority input is enabled on this slave port. |
| Bits 16–21 | **Slave General Purpose Control Register Reserved -** These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility. | NA |
| ARB | **Arbitration Mode -** These bits are used to select the arbitration policy for the slave port.<br><br>These bits are initialized by hardware reset.<br>The reset value is 00 | 00 Fixed Priority.<br>01 Round Robin (rotating) Priority.<br>10 Reserved<br>11 Reserved |
| Bits 24–25 | **Slave General Purpose Control Register Reserved -** These bits are reserved for future expansion. They are read as zero and should be written with zero for upward compatibility. | NA |
| PCTL | **Parking Control -** These bits determine the parking control used by this slave port.<br><br>These bits are initialized by hardware reset.<br>The reset value is 00. | 00 When no master is making a request the arbiter will park the slave port on the master port defined by the PARK bit field.<br>01 When no master is making a request the arbiter will park the slave port on the last master to be in control of the slave port.<br>10 When no master is making a request the arbiter will park the slave port on no master and will drive all outputs to a constant safe state.<br>11 Reserved |
| Bit 28 | **Slave General Purpose Control Register Reserved -** This bit is reserved for future expansion. It is read as zero and should be written with zero for upward compatibility. | NA |

**Table 9-6. Slave General Purpose Control Register Descriptions (continued)**

| Name | Description | Setting |
|---|---|---|
| PARK | **PARK -** These bits are used to determine which master port this slave port parks on when no masters are actively making requests and the PCTL bits are set to 00.<br><br>These bits are initialized by hardware reset.<br>The reset value is 000 | 000Park on Master Port 0<br>001Park on Master Port 1<br>010Park on Master Port 2<br>011Park on Master Port 3<br>100Park on Master Port 4<br>101Park on Master Port 5<br>110Park on Master Port 6<br>111Park on Master Port 7 |

The SGPCR can only be accessed in supervisor mode with 32-bit accesses. Once the RO (Read Only) bit has been set in the SGPCR the SGPCR can only be read, attempts to write to it will have no effect on the SGPCR and result in an error response.

### 9.3.2.3 Master General Purpose Control Register

The Master General Purpose Control Register (MGPCR) presently controls only whether or not the master's undefined length burst accesses will be allowed to complete uninterrupted or whether they can be broken by requests from higher priority masters.

The AULB (Arbitrate on Undefined Length Bursts) bit field determines whether (and when) or not the XBAR will arbitrate away the slave port the master owns when the master is performing undefined length burst accesses.

If the user has configured the XBAR to have less than eight master ports only the registers associated with the remaining master ports will be present, all other registers will become reserved locations in memory.

**MGPCRn** — **Master General Purpose Control Register n** — **Addr $BASE + 0x800 + n*100, Wait State: 0, Access: S**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | |
| TYPE: | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Note: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | AULB | | |
| TYPE: | r | r | r | r | r | r | r | r | r | r | r | r | r | rw | rw | rw |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Note: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

**Note:** for n = 0 to 7

**Figure 9-5. Master General Purpose Control Register n**

**Table 9-7. Master General Purpose Control Register Descriptions**

| Name | Description | Setting |
|---|---|---|
| Bits 0–28 | **Master General Purpose Control Register Reserved -** These bits are reserved for future expansion. They read as zero and should be written with zero for upward compatibility. | NA |
| AULB | **Arbitrate on Undefined Length Bursts -** These bits are used to select the arbitration policy during undefined length bursts by this master.<br><br>These bits are initialized by hardware reset. The reset value is 000 | 000No arbitration will be allowed during an undefined length burst.<br>001Arbitration will be allowed at any time during an undefined length burst.<br>010Arbitration will be allowed after four beats of an undefined length burst.<br>011Arbitration will be allowed after eight beats of an undefined length burst.<br>100Arbitration will be allowed after 16 beats of an undefined length burst.<br>101Reserved<br>110Reserved<br>111Reserved |

The MGPCR can only be accessed in supervisor mode with 32-bit accesses.

### 9.3.3 Coherency

Since the content of the registers has a real time effect on the operation of the XBAR it is important for the user to understand that any register modifications take effect as soon as the register is written. The values of the registers do not track with slave port related AHB accesses but instead track only with IP bus accesses.

The exception to this rule are the AULB bits in the MGPCR. These update of these bits is only recognized when the master on that master port runs an IDLE cycle, even though the IP bus cycle to write them will have long since terminated successfully. If the AULB bits in the MGPCR are written in between two burst accesses the new AULB encodings will not take effect until an IDLE cycle has been initiated by the master on that master port.

## 9.4 Function

This section describes in more detail the functionality of the XBAR.

### 9.4.1 Arbitration

The XBAR supports two arbitration schemes; a simple fixed-priority comparison algorithm, and a simple round-robin fairness algorithm. The arbitration scheme is independently programmable for each slave port.

#### 9.4.1.1 Arbitration During Undefined Length Bursts

Arbitration points during an undefined length burst are defined by the current master's MGPCR AULB field setting. When a defined length is imposed on the burst via the AULB bits the undefined length burst will be treated as a single or series of single back to back fixed length burst accesses.

Example: A master runs an undefined length burst and the AULB bits in the MGPCR indicate arbitration will occur after the fourth beat of the burst. The master runs two sequential beats and then starts what will be an 12 beat undefined length burst access to a new address within the same slave port region as the previous access. The XBAR will not allow an arbitration point until the fourth overall access (second beat of the second burst). At that point all remaining accesses will be open for arbitration until the master loses control of the slave port.

Assume the master loses control of the slave port after the fifth beat of the second burst. Once the master regains control of the slave port no arbitration point will be available until after the master has run four more beats of its burst. After the fourth beat of the (now continued) burst (ninth beat of the second burst from the master's perspective) is taken all beats of the burst will once again be open for arbitration until the master loses control of the slave port.

Assume the master again loses control of the slave port on the fifth beat of the third (now continued) burst (10th beat of the second burst from the master's perspective). Once the master regains control of the slave port it will be allowed to complete its final two beats of its burst without facing arbitration.

Note that fixed length burst accesses are not affected by the AULB bits. All fixed length burst accesses lock out arbitration until the last beat of the fixed length burst.

#### 9.4.1.2 Fixed Priority Operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the MPR (Master Priority Register). If two masters both request access to a slave port the master with the highest priority in the selected priority register will gain control over the slave port.

Any time a master makes a request to a slave port the slave port checks to see if the new requesting master's priority level is higher than that of the master that currently has control over the slave port (unless the slave port is in a parked state). The slave port does an arbitration check at every clock edge to ensure that the proper master (if any) has control of the slave port.

If the new requesting master's priority level is higher than that of the master that currently has control of the slave port the new requesting master will be granted control over the slave port at the next clock edge. The exception to this rule is if the master that currently has control over the slave port is running a fixed length burst transfer or a locked transfer. In this case the new requesting master will have to wait until the end of the burst transfer or locked transfer before it will be granted control of the slave port. If the master is running an undefined length burst transfer the new requesting master must wait until an arbitration point for the undefined length burst transfer before it will be granted control of the slave port. Arbitration points for an undefined length burst are defined in the MGPCR for each master.

If the new requesting master's priority level is lower than that of the master that currently has control of the slave port the new requesting master will be forced to wait until the master that currently has control

of the slave port either runs an IDLE cycle or runs a non IDLE cycle to a location other than the current slave port.

#### 9.4.1.3 Round-Robin Priority Operation

When operating in round-robin mode, each master is assigned a relative priority based on the master number.This relative priority is compared to the ID of the last master to perform a transfer on the slave bus. The highest priority requesting master will become owner of the slave bus as the next transfer boundary (accounting for locked and fixed-length burst transfers). Priority is based on how far ahead the ID of the requesting master is to the ID of the last master (ID is defined by master port number, not the **hmaster** field).

Once granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line will be granted access to the slave port at the next assertion of **sX_hready**, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the XBAR is implemented with master ports 0, 1, 4 and 5. If the last master of the slave port was master 1, and master 0, 4 and 5 make simultaneous requests, they will be serviced in the order 4, 5 and then 0.

Parking may still be used in a round-robin mode, but will not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff will occur to the next master in line after one cycle of arbitration. If the slave port is put into low power park mode the round-robin pointer will be reset to point at master port 0, giving it the highest priority.

Each master port has an **mX_high_priority** input which can be enabled by writing the correct data to the SGPCR or ASGPCR. If a master's **mX_high_priority** input is enabled for a slave port programmed for round-robin mode, that master can force the slave port into fixed priority mode by asserting its **mX_high_priority** input while making a request to that particular slave port. While that (or any enabled) master's **mX_high_priority** input is asserted while making an access attempt to that particular slave port, the slave port will remain in fixed priority mode. Once that (or any enabled) master's **mX_high_priority** input is negated, or the master no longer attempts to make accesses to that particular slave port, the slave port will revert back to round-robin priority mode and the pointer will be set on the last master to access the slave port.

### 9.4.2 Priority Assignment

Each master port needs to be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within a register (MPR or AMPR) the XBAR will respond with an error and the registers will not be updated.

#### 9.4.2.1 Context Switching

The XBAR has a hardware input per slave port (**sX_ampr_sel**) which is used to select which registers the master priority levels and general purpose control bits will be taken from. When **sX_ampr_sel** is 0 the MPR and SGPCR will be selected, when **sX_ampr_sel** is 1 the AMPR and the ASGPCR will be selected. This hardware input is useful for context switching so the user does not have to rewrite the MPR or SGPCR

if a particular slave port would temporarily benefit from modifying the master priority levels or functions affected by the bits in the SGPCR.

#### 9.4.2.2 Priority Elevation

The XBAR has a hardware input per master port (**mX_high_priority**) which is used to temporarily elevate the master's priority level on all slave ports. When the master's **mX_high_priority** input is asserted the master port will automatically have higher priority than all other master ports that do not have their **mX_high_priority** input asserted regardless of the priority levels programmed in the MPR and AMPR. If multiple master ports have their **mX_high_priority** input asserted they will have higher priority than all master ports which do not have their **mX_high_priority** inputs asserted. The MPR or AMPR priority level (dependent on the state of **sX_ampr_sel**) will determine which master port that has its **mX_high_priority** input asserted has the highest priority on a slave port by slave port basis.

This functionality is useful because it allows the user to automatically elevate a master port's priority level throughout the XBAR in order to quickly perform temporary tasks such as servicing interrupts.

Please note that the HPEx bits must be set in the SGPCR or ASGPCR in the slave port in order for the **mX_high_priority** inputs to be received by the slave port.

### 9.4.3 Master Port Functionality

#### 9.4.3.1 General

Each master port consists of two decoders, a capture unit, a register slice, a mux and a small state machine.

The first decoder is used to decode the **mX_hsel_slv** and control signals coming directly from the master, telling the state machine where the master's next access will be and if it is in fact a legal access. The second decoder receives its input from the capture unit, so it may be looking directly at the signals coming from the master or it may be looking at captured signals coming from the master, depending entirely on the state of the targeted slave port. The second decoder is then used to generate the access requests that go to the slave ports.

The capture unit is used to capture the address and control information coming from the master in the event that the targeted slave port cannot immediately service the master. The capture unit is controlled by outputs from the state machine which tell it to either pass through the original master signals or the captured signals.

The register slice contains the registers associated with the specific master port. The registers have a quasi-IP bus interface at this level for reads and writes and the outputs feed directly into the state machine.

The mux is used simply to select which slave's read data is sent back to the master. The mux is controlled by the state machine.

The state machine controls all aspects of the master port. It knows which slave port the master wants to make a request to and controls when that request is made. It also has knowledge of each slave port, knowing whether or not the slave port is ready to accept an access from the master port. This will determine whether or not the master may immediately have its request taken by the slave port or whether the master port will have to capture the master's request and queue it at the slave port boundary.

A block diagram of a master port can be seen in Figure 9-6.

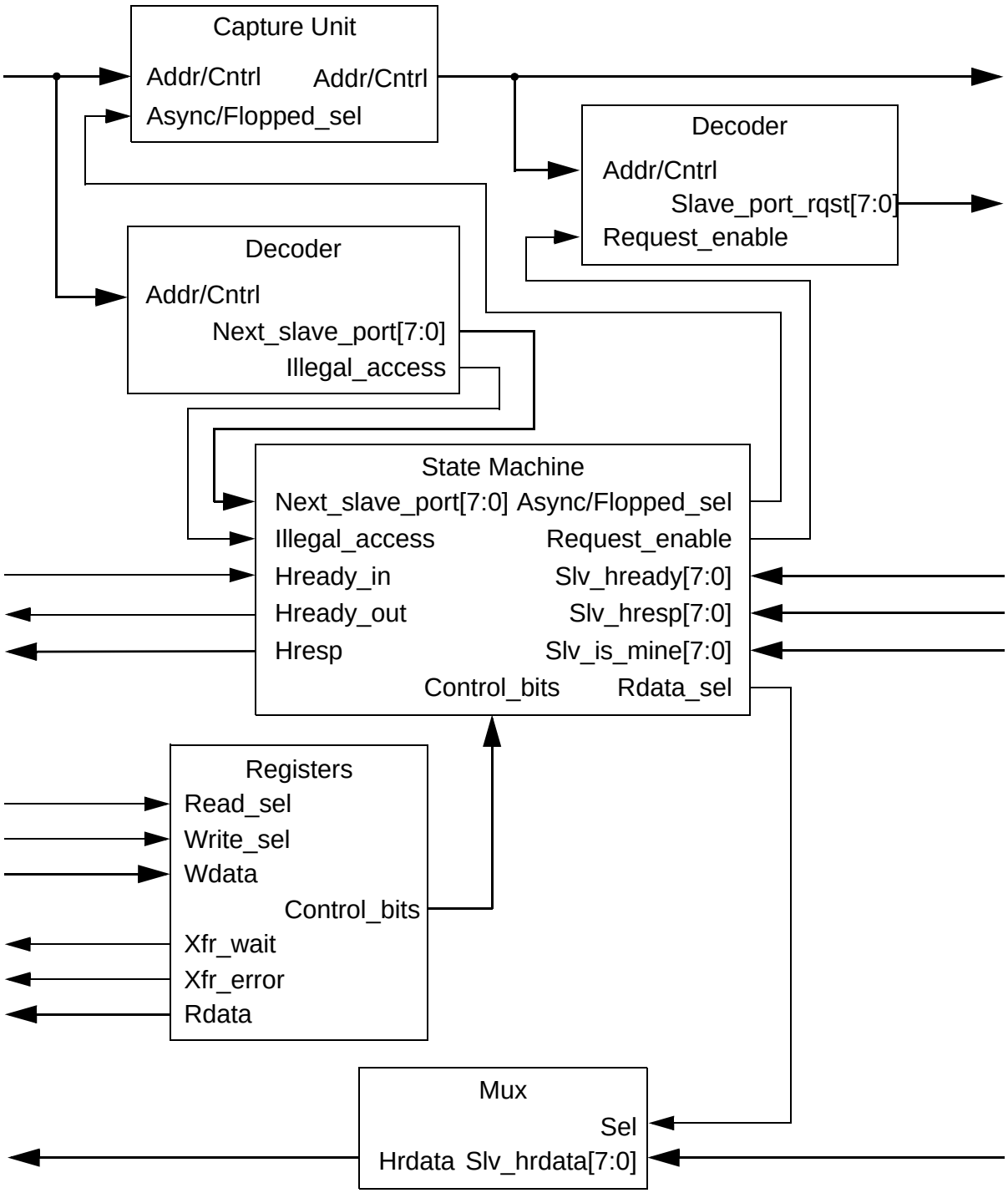



**Figure 9-6. XBAR Master Port Block Diagram**

#### 9.4.3.2 Master Port Decoders

The decoders are very simple as they ensure an access request is allowed to be made and that the slave port targeted is actually present in the design. The decoders feeding the state machine are always enabled. The decoders that select the slave are enabled only when the master port controlling state machine wants to make a request to a slave port. This is necessary so that if a master port is making an access to a slave port and is being wait stated, and its next access is to a different slave port, the request to the second slave port can be held off until the access to the first slave port is terminated.

The decoders also output a “hole decode” or illegal access signal which tells the state machine that the master is trying to access a slave port that does not exist.

#### 9.4.3.3 Master Port Capture Unit

The capture unit simply captures the state of the master’s address and control signals if the XBAR cannot immediately pass the master’s request through to the proper slave port. The capture unit consists of a set of flops and a mux which selects either the asynchronous path from address and control or the flopped (captured) address and control information.

#### 9.4.3.4 Master Port Registers

The registers in the master port are only those registers associated with this particular master port. The read and write interface for the registers is a quasi-IP bus interface. It is not a full IP bus interface at this level because not all the IP bus signals are routed this deep in the design.

There is a register control block at the same level of the master port and slave port instantiations in the XBAR. This control block ensures that all accesses are 64-bit supervisor accesses before passing them on to the master ports.

The register outputs are connected directly to the state machine.

#### 9.4.3.5 Master Port State Machine

##### 9.4.3.5.1 Master Port State Machine States

The master side state machine’s main function is to monitor the activities of the master port. The state machine has six states: **busy**, **idle**, **waiting**, **stalled**, **steady state**, **first cycle error response** and **second cycle error response**.

The **busy** state is used when the master runs a BUSY cycle to the master port. The master port maintains its request to the slave port if it currently owns the slave port; however, if it loses control of the slave port it will no longer maintain its request. If the master port loses control of the slave port it will not be allowed to make another request to the slave port until it runs a NSEQ or SEQ cycle.

The **idle** state is used when the master runs a valid IDLE cycle to the master port. The master port makes no requests to the slave ports (disables the slave port decoder) and terminates the IDLE cycle.

The **waiting** state is used when the **hsel** signal is negated to the master port, indicating the master is running valid cycles to a local slave other than the XBAR. In this case the max disables the slave port decoder and holds **hresp** and **hready** negated.

The **stalled** state is used when the master makes a request to a slave port that is not immediately ready to receive the request. In this case the state machine will direct the capture unit to send out the captured address and control signals and will enable the slave port decoder to indicate a pending request to the appropriate slave port.

The **steady state** state is used when the master port and slave port are in fully asynchronous mode, making the XBAR completely transparent in the access. The state machine selects the appropriate slave's **hresp, hready** and **hrdata** to pass back to the master.

The **first cycle error response** and **second cycle error response** states are self explanatory. The XBAR will respond with an error response to the master if the master tries to access an unimplemented memory location through the XBAR (that is, a slave port that does not exist).

#### 9.4.3.5.2 Master Port State Machine Slave Swapping

The design of the master side state machine is fairly straightforward. The one real decision to be made is how to handle the master moving from one slave port access to another slave port access. The approach that was taken is to minimize or eliminate when possible any "bubbles" that would be inserted into the access due to switching slave ports.

The state machine will not allow the master to request access to another slave port until the current access being made is terminated. This prevents a single master from owning two slave ports at the same time (the slave port it is currently accessing and the slave port it wishes to access next).

The state machine also maintains watch on the slave port the master is accessing as well as the slave port the master wishes to switch to. If the new slave port is parked on the master then the master will be able to make the switch without incurring any delays. The termination of the current access will also act as the launch of the new access on the new slave port. If the new slave port is not parked on the master then the master will incur a minimum one clock delay before it can launch its access on the new slave port.

This is the same for switching from the **busy, idle** or **waiting** state to actively accessing a slave port. If the slave port is parked on the master the state machine will go to the **steady state** state and the access will begin immediately. If the slave port is not parked on the master (serving another master, parked on another master or in low power park mode) then the state machine will transition to the **stalled** state and at least a one clock penalty will be paid.

### 9.4.4 Slave Port Functionality

#### 9.4.4.1 General

Each slave port consists of a register slice, a bank of muxes and a state machine.

The register slice contains the registers associated with the specific slave port. The registers have a quasi-IP bus interface at this level for reads and writes and the outputs feed directly into the state machine.

The muxes are a series of 8 to 1 muxes that take in all the address, control and write data information from each of the master ports and then pass the correct master's signals to the slave port. The state machine controls all the muxes.

The state machine is where the main slave port arbitration occurs, it decides which master is in control of the slave port and which master will be in control of the slave port in the next bus cycle.

A block diagram of a slave port can be seen in Figure 9-7.

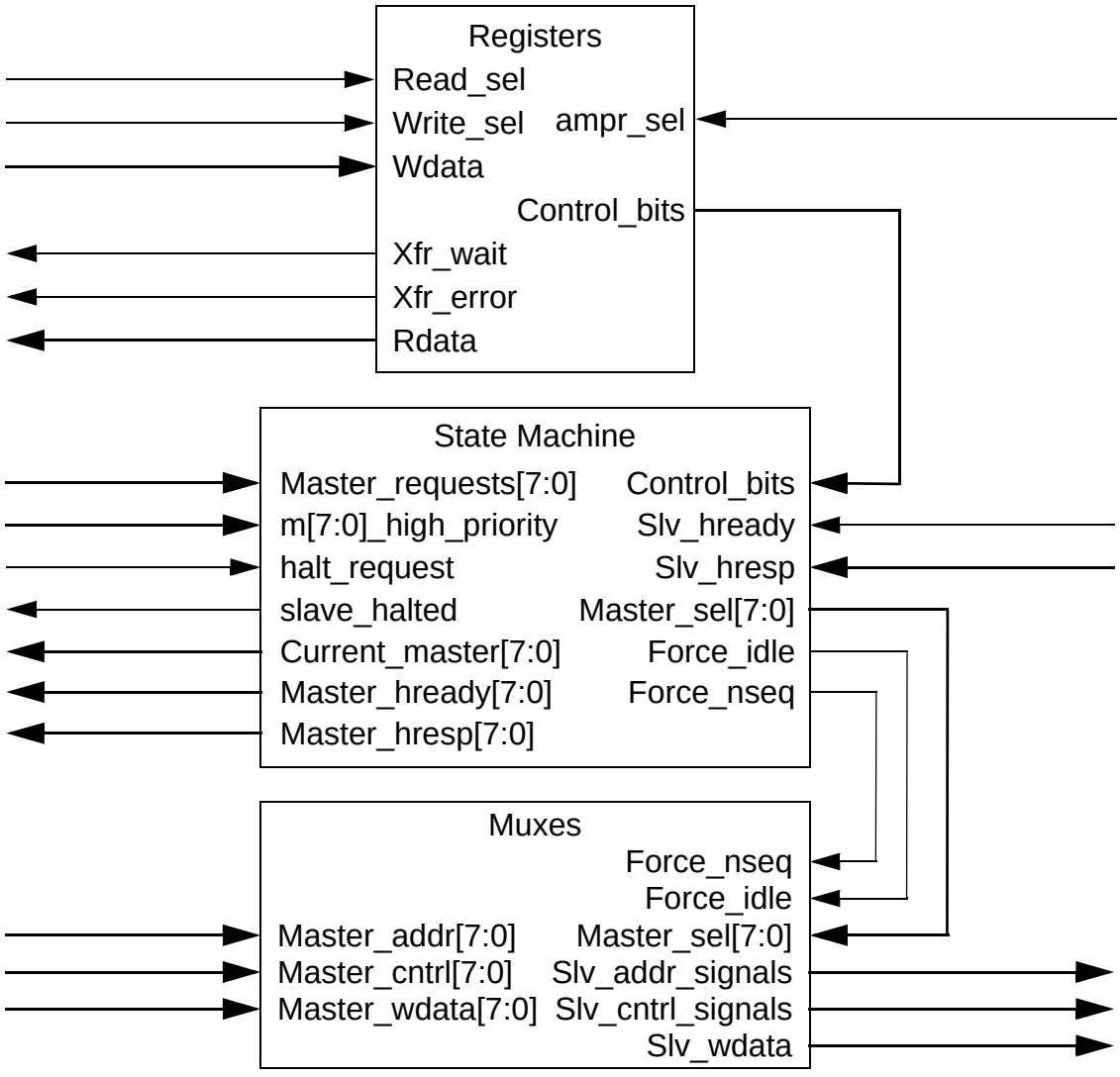



**Figure 9-7. XBAR Slave Port Block Diagram**

### 9.4.4.2 Slave Port Muxes

The block diagram (Figure 9-7) shows only one block for all the muxes. In reality that block instantiates many 8 to 1 muxes, one for each master-to-slave signal in fact. All the muxes are designed in an AND - OR fashion, so that if no master is selected the output of the muxes will be zero. (This is an important feature for low power park mode.)

The muxes also have an override signal which is used by the slave port to asynchronously force IDLE cycles onto the slave bus. When the state machine forces an IDLE cycle it zeros out **htrans** and **hmastlock**, making sure the slave bus sees a valid IDLE cycle being run by the XBAR.

The enable to the mux controlling **htrans** also contains an additional control signal from the state machine so that a NSEQ transaction can be forced. This is done any time the slave port switches masters to ensure that no IDLE-SEQ, BUSY-SEQ or NSEQ-SEQ transactions are seen on the slave port when they shouldn't be. If the state machine indicates to run both an IDLE and an NSEQ cycle, the IDLE directive will have priority.

**NOTE**

IDLE-SEQ is in fact an illegal access, but a possible scenario given the multi-master environment in the XBAR unless corrected by the XBAR.

#### 9.4.4.3 Slave Port Registers

There is a register control block at the same level of the master port and slave port instantiations in the XBAR. This control block ensures that all accesses are 64-bit supervisor accesses before passing them on to the master and slave ports.

The registers in the slave port are only those registers associated with this particular slave port. The read and write interface for the registers is a quasi-IP bus interface. It is not a full IP bus interface at this level because not all the IP bus signals are routed this deep in the design.

The register outputs are connected directly to the slave state machine with the **sX_ampr_sel** input determining which priority register values, halt priority value, arbitration algorithm and parking control bits are passed to the state machine. The registers can be read from an unlimited number of times. The registers can only be written to as long as the RO bit is written to 0 in the SGPCR, once it is written to a 1 only a hardware reset will allow the registers to be written again.

#### 9.4.4.4 Slave Port State Machine

##### 9.4.4.4.1 Slave Port State Machine States

At the heart of the slave port is the state machine. The state machine is simplicity itself, requiring only four states - **steady state**, **transition state, transition hold state** and **hold state**. Either the slave port is owned by the same master it was in the last clock cycle (either by active use or by parking), it is transitioning to a new master (either for active use or parking), it is transitioning to a new master during wait states or it is being held on the same master pending a transition to a new master.

##### 9.4.4.4.2 Slave Port State Machine Arbitration

The real work in the state machine is determining which master port will be in control of the slave port in the next clock cycle, the arbitration. Each master is programmed with a fixed 3 bit priority level. A fourth priority bit is derived from the **mX_high_priority** inputs on the master ports, effectively making each master's priority level a 4 bit field with **mX_high_priority** being the MSB. The XBAR uses these bits in determining priority levels when programmed for fixed priority mode of operation or when one of the enabled **mX_high_priority** inputs is asserted.

Arbitration always occurs on a clock edge, but only occurs on edges when a change in mastership will not violate AHB-Lite protocols. Valid arbitrations points include any clock cycle in which **sX_hready** is asserted (provide the master is not performing a burst or locked cycle) and any wait state in which the master owning the bus indicates a transfer type of IDLE (provided the master is not performing a locked cycle).

Since arbitration can occur on every clock cycle the slave port masks off all master requests if the current master is performing a locked transfer or a protected burst transfer, guaranteeing that no matter how low its priority level it will be allowed to finish its locked or protected portion of a burst sequence.

#### 9.4.4.4.3 Slave Port State Machine Master Handoff

The only times the slave port will switch masters when programmed for fixed priority mode of operation is when a higher priority master makes a request or when the current master is the highest priority and it gives up the slave port by either running and IDLE cycle to the slave port or running a valid access to a location other than the slave port.

If the current master loses control of the slave port because a higher priority master takes it away, the slave port will not incur any wasted cycles. The current master has its current cycle terminated by the slave port at the same time the new master's address and control information are recognized by the slave port. This appears as a seamless transition on the slave port.

If the current master is being wait-stated when the higher priority master makes its request, then the current master will be allowed to make one more transaction on the slave bus before giving it up to the new master. Figure 9-8 illustrates the effect of a higher priority master taking control of the bus when the slave port is programmed for a fixed priority mode of operation.

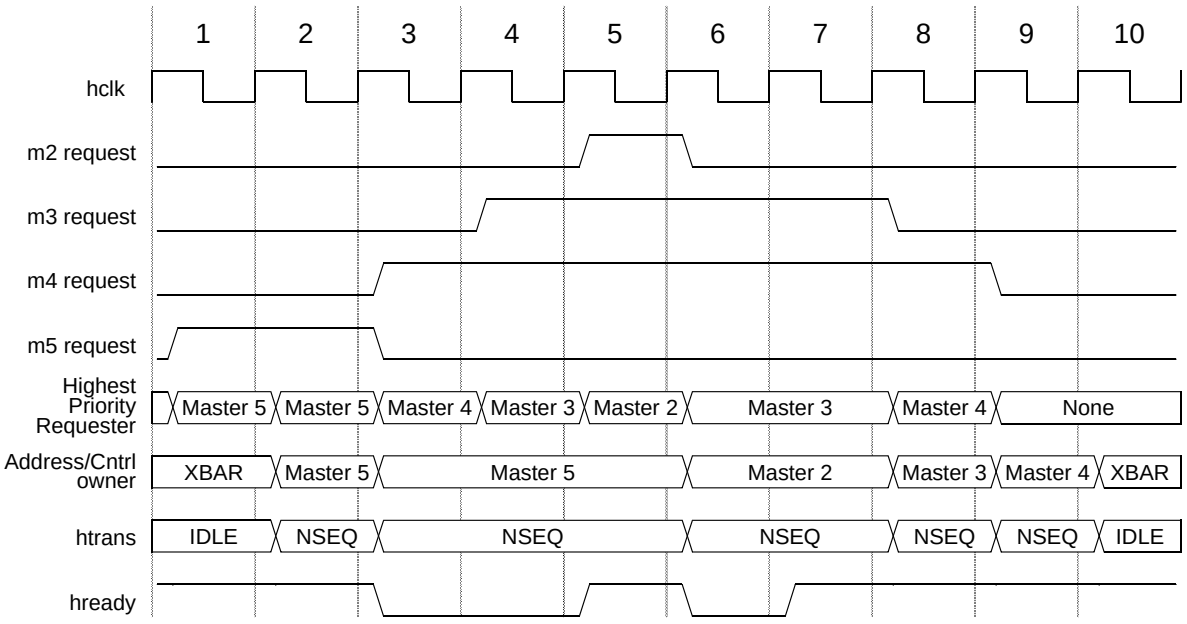



**Figure 9-8. Low to high priority mastership change**

If the current master is the highest priority master and it gives up the slave port by running an IDLE cycle or by running a valid cycle to another location other than the slave port the next highest priority master will gain control of the slave port. If the current access incurs any wait states then the transition will be seamless and no bandwidth will be lost; however, if the current transaction is terminated without wait states then one IDLE cycle will be forced onto the slave bus by the XBAR before the new master will be able to take control of the slave port. If no other master is requesting the bus then IDLE cycles will be run by the XBAR but no bandwidth will truly be lost since no master is making a request. Figure 9-9 illustrates the effect of a higher priority master giving up control of the bus.

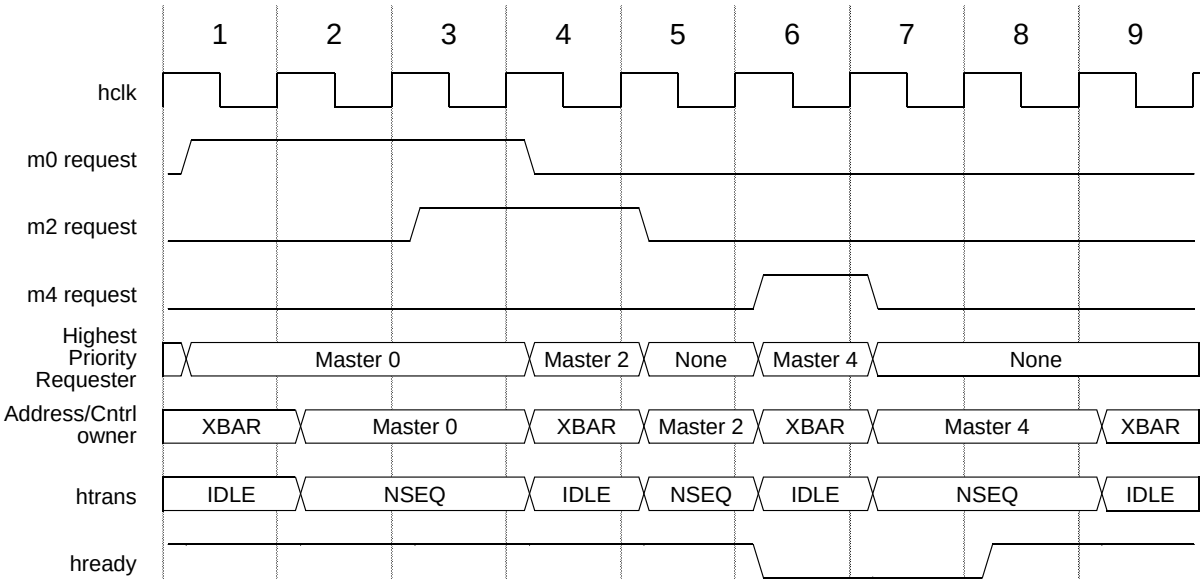



**Figure 9-9. High to low priority mastership change**

When the slave port is programmed for round-robin mode of arbitration then the slave port will switch masters any time there is more than one master actively making a request to the slave port. This will happen because any master other than the one which presently owns the bus will be considered to have higher priority. Figure 9-10 shows an example of round-robin mode of operation.

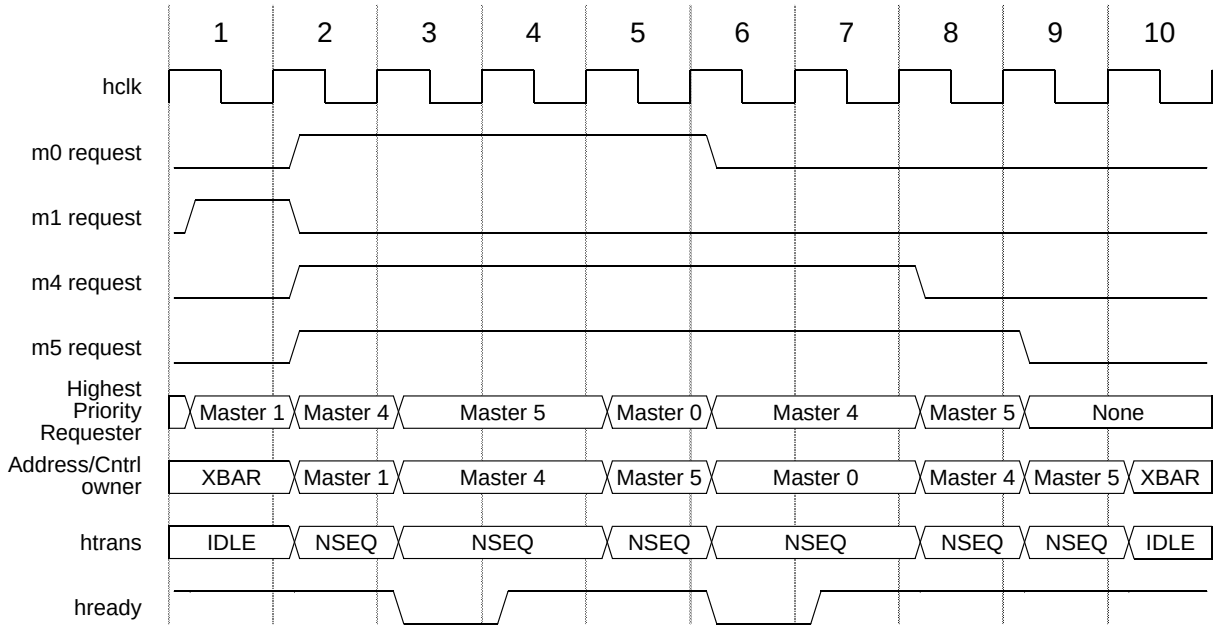



**Figure 9-10. Round-robin mastership change**

#### 9.4.4.4.4 Slave Port State Machine Parking

If no master is currently making a request to the slave port then the slave port will be parked. It will park in one of four places, dictated by the PCTL and PARK bits in the GPCR or AGPCR (depending on the state of the **sX_ampr_sel**) and the locked state of the last master to access it.

If the last master to access the slave port ran a locked cycle and continues to run locked cycles even after leaving the slave port the slave port will park on that master without regard to the bit settings in the GPCR and without regard to pending requests from other masters. This is done so a master can run a locked transfer to the slave port, leave it, and return to it and be guaranteed that no other master has had access to it (provided the master maintains all transfers are locked transfers). If locking is not an issue for parking the GPCR bits will dictate the parking method.

If the PCTL bits are set for “low power park” mode then the slave port will enter low power park mode. It will not recognize any master as being in control of it and it will not select any master’s signals to pass through to the slave bus. In this case all slave bus activity will effectively halt because all slave bus signals being driven from the XBAR will be 0. This of course can save quite a bit of power if the slave port will not be in use for some time. The down side is that when a master does make a request to the slave port it will be delayed by one clock since it will have to arbitrate to acquire ownership of the slave port.

If the PCTL bits are set to “park on last” mode then the slave port will park on the last master to access it, passing all that masters signals through to the slave bus. The XBAR will asynchronously force **htrans[1:0]**, **hmaster[3:0]**, **hburst[2:0]** and **hmastlock** to 0 for all access that the master does not run to the slave port. When that master access the slave port again it will not pay any arbitration penalty; however, if any other master wishes to access the slave port a one clock arbitration penalty will be imposed.

If the PCTL bits are set to “use PARK/APARK” mode then the slave port will park on the master designated by the PARK bits. The behavior here is the same as for the “park on last” mode with the exception that a specific master will be parked on instead of the last master to access the slave port. If the master designated by the PARK bits tries to access the slave port it will not pay an arbitration penalty while any other master will pay a one clock penalty. Figure 9-11 illustrates parking on a specific master.

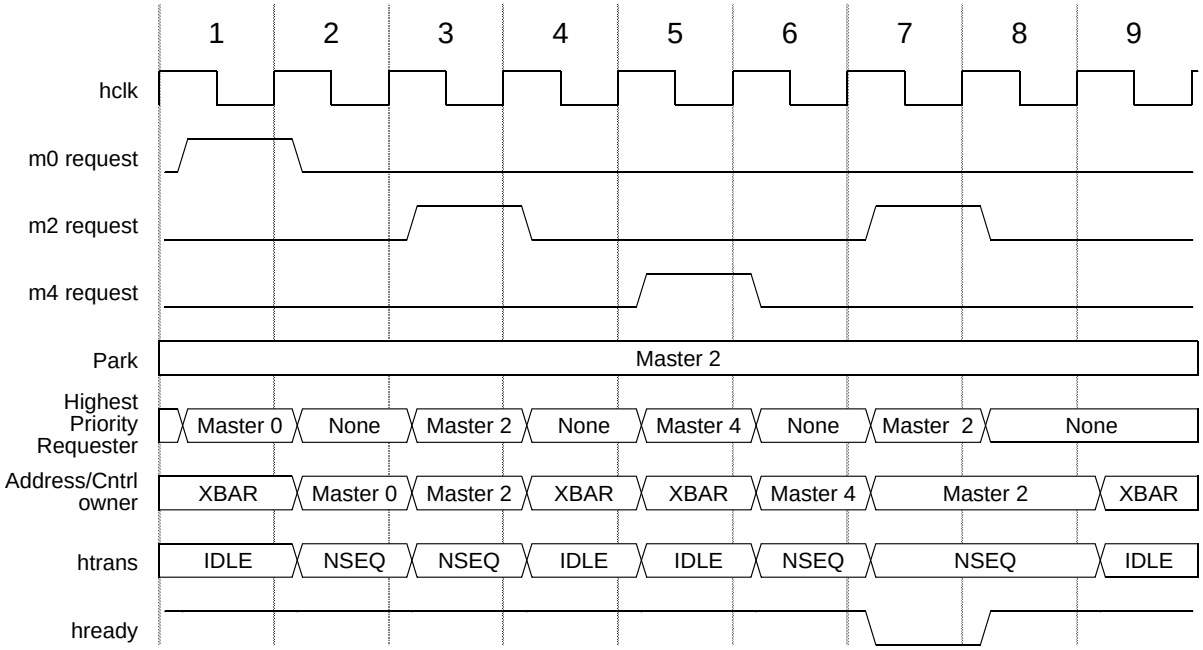



**Figure 9-11. Parking on a specific master**

Figure 9-12 illustrates parking on the last master. Note that in cycle 6 simultaneous requests are made by master 2 and master 4. Although master 2 has higher priority, the slave bus is parked on master 4 so master 4’s access will be taken first. The slave port parks on master 2 once it has given control to master 2. This same situation can occur when parking on a specific master as well.

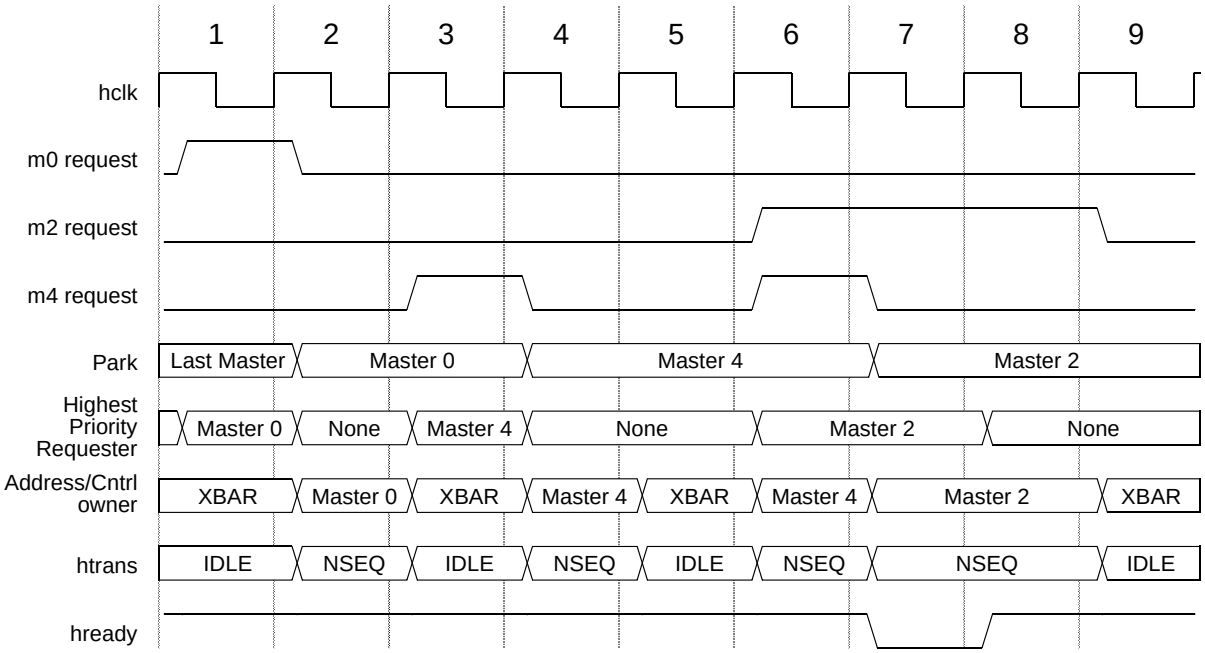



**Figure 9-12. Parking on last master**

#### 9.4.4.4.5 Slave Port State Machine Halt Mode

If the **max_halt_request** input is asserted the slave port will eventually halt all slave bus activity and go into halt mode, which is almost identical to low power park mode. The HLP bit in the GPCR controls the priority level of the **max_halt_request** in the arbitration algorithm. If the HLP bit is cleared then the **max_halt_request** will have the highest priority of any master and will gain control of the slave port at the next arbitration point (most likely the next bus cycle, unless the current master is running a locked or fixed length burst transfer). If the HLP bit is set then the slave port will wait until no masters are actively making requests before moving to halt mode.

Regardless of the state of the HLP bit, once the slave port has gone into halt mode as a result of **max_halt_request** being asserted, it will remain in halt mode until **max_halt_request** is negated, regardless of the priority level of any masters that may make requests.

In halt mode no master is selected to own the slave port so all the outputs of the slave port are set to 0.

## 9.5 Initialization/Application Information

No initialization is required by or for the XBAR. Hardware reset ensures all the register bits used by the XBAR are properly initialized.

## 9.6 Interface

This section provides information on the XBAR interface.

### 9.6.1 Overview

The main goal of the XBAR is to increase overall system performance by allowing multiple masters to communicate in parallel with multiple slaves. In order to maximize data throughput it is essential to keep arbitration delays to a minimum.

This section examines data throughput from the point of view of masters and slaves, detailing when the XBAR will stall the masters or insert bubbles on the slave side.

### 9.6.2 Master Ports

Master accesses will receive one of four responses from the XBAR. They will either be ignored, terminated, taken, stalled or responded to with an error.

#### 9.6.2.1 Ignored Accesses

A master access will be ignored if the **hsel** input of the XBAR is not asserted. The XBAR will respond to IDLE transfers when the **hsel** input is asserted but will not allow the access to pass through the XBAR.

#### 9.6.2.2 Terminated Accesses

A master access will be terminated if the **hsel** input of the XBAR is asserted and the transfer type is IDLE. The XBAR will terminated the access and it will not be allowed to pass through the XBAR.

#### 9.6.2.3 Taken Accesses

A master access will be taken if the **hsel** input of the XBAR is asserted and the transfer type is non IDLE and the slave port to which the access decodes is either currently servicing the master or is parked on the master. In this case the XBAR will be completely transparent and the master's access will be immediately seen on the slave bus and no arbitration delays will be incurred.

#### 9.6.2.4 Stalled Accesses

A master access will be stalled if the **hsel** input of the XBAR is asserted and the transfer type is non IDLE and the access decodes to a slave port that is busy serving another master, parked on another master or is in low power park mode. The XBAR will indicate to the master that the address phase of the access has been taken but will then queue the access to the appropriate slave port to enter into arbitration for access to that slave port.

If the slave port is currently parked on another master or is in low power park mode and no other master is requesting access to the slave port then only one clock of arbitration will be incurred. If the slave port is currently serving another master of a lower priority and the master has a higher priority than all other requesting masters then the master will gain control over the slave port as soon as the data phase of the current access is completed (burst and locked transfers excluded). If the slave port is currently servicing another master of a higher priority then the master will gain control of the slave port once the other master releases control of the slave port if no other higher priority master is also waiting for the slave port.

#### 9.6.2.5 Error Response Terminated Accesses

A master access will be responded to with an error if the **hsel** input of the XBAR is asserted and the transfer type is non IDLE and the access decodes to a location not occupied by a slave port. This is the only time the XBAR will respond with an error response. All other error responses received by the master are the result of error responses on the slave ports being passed through the XBAR.

### 9.6.3 Slave Ports

The goal of the XBAR with respect to the slave ports is to keep them 100% saturated when masters are actively making requests. In order to do this the XBAR must not insert any bubbles onto the slave bus unless absolutely necessary.

There is only one instance when the XBAR will force a bubble onto the slave bus when a master is actively making a request. This occurs when a higher priority master has control of the slave port and is running single clock (zero wait state) accesses while a lower priority master is stalled waiting for control of the slave port. When the higher priority master either leaves the slave port or runs an IDLE cycle to the slave port the XBAR will take control of the slave bus and run a single IDLE cycle before giving the slave port to the lower priority master that was waiting for control of the slave port.

The only other times the XBAR will have control of the slave port is when the XBAR is halting or when no masters are making access requests to the slave port and the XBAR is forced to either park the slave port on a specific master or put the slave port into low power park mode.

In most instances when the XBAR has control of the slave port it will indicate IDLE for the transfer type, negate all control signals and indicate ownership of the slave bus via the **hmaster** encoding of 4’b0000. One exception to this rule is when a master running locked cycles has left the slave port but continues to run locked cycles. In this case the XBAR will control the slave port and will indicate IDLE for the transfer type but it will not affect any other signals.

**NOTE**

When a master runs a locked cycle through the XBAR, the master will be guaranteed ownership of all slave ports it accesses while running locked cycles for one cycle beyond when the master finishes running locked cycles.

This page is intentionally left blank.

# Chapter 10
# Deserial Serial Peripheral Interface (DSPI)

## 10.1 Introduction

This chapter describes the deserial serial peripheral interface (DSPI), which provides a synchronous serial bus for communication between the MCU and an external peripheral device.

## 10.2 Block diagram

A block diagram of the DSPI is shown in Figure 10-1.

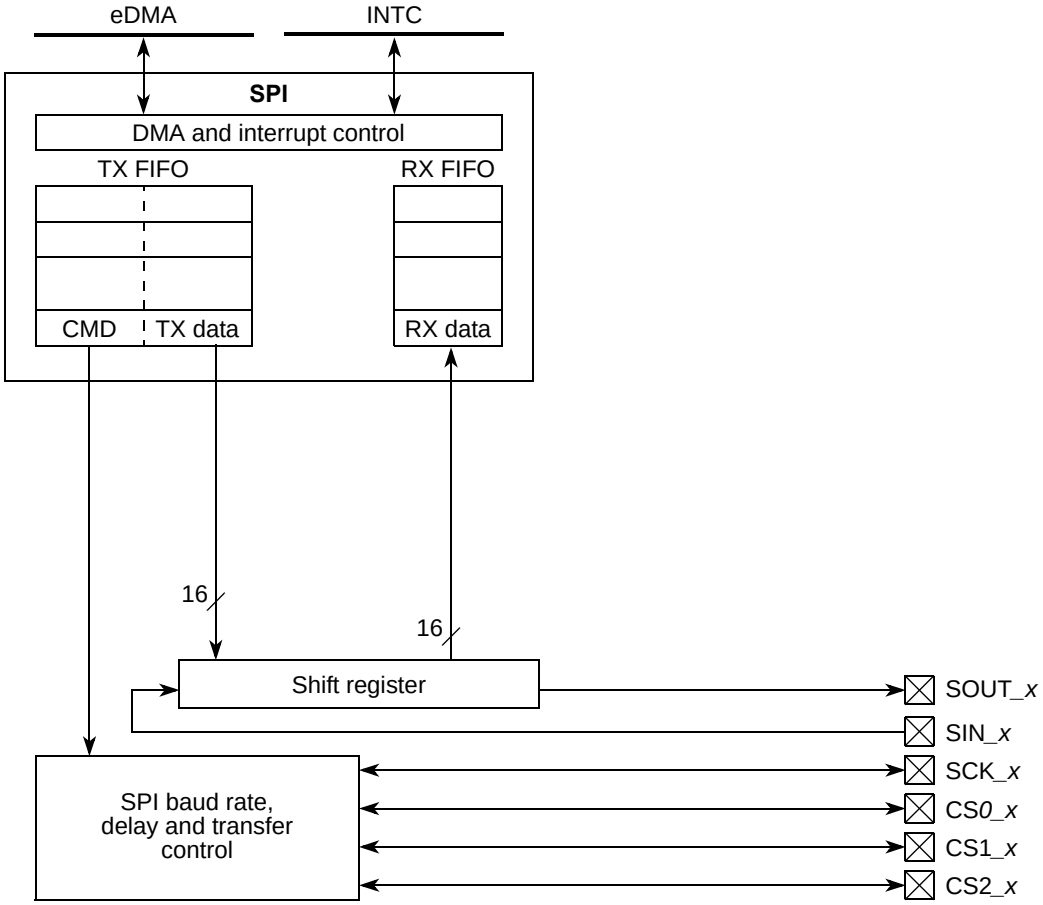



**Figure 10-1. DSPI block diagram**

## 10.3 Overview

The register content is transmitted using an SPI protocol.

For queued operations the SPI queues reside in internal SRAM which is external to the DSPI. Data transfers between the queues and the DSPI FIFOs are accomplished through the use of the eDMA controller or through host software.

Figure 10-2 shows a DSPI with external queues in internal SRAM.

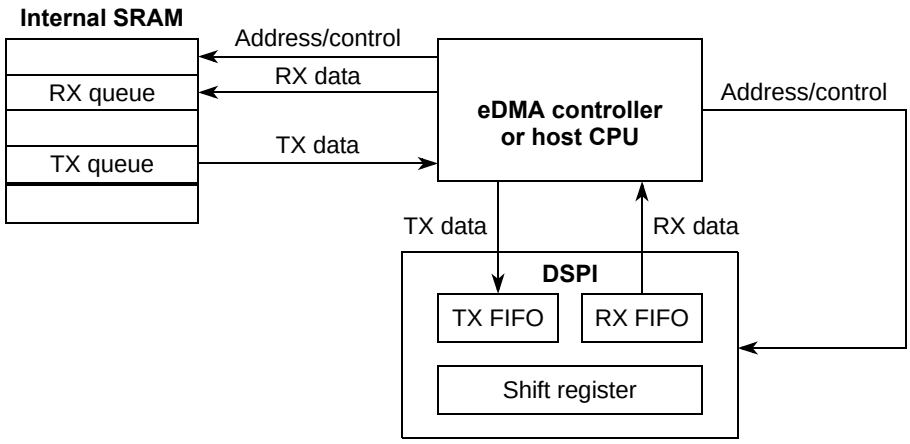



**Figure 10-2. DSPI with queues and eDMA**

## 10.4 Features

The DSPI supports these SPI features:

- Full-duplex, three-wire synchronous transfers
- Master and slave mode
- Buffered transmit and receive operation using the TX and RX FIFOs, with depths of five entries
- Visibility into TX and RX FIFOs for ease of debugging
- FIFO bypass mode for low-latency updates to SPI queues
- Programmable transfer attributes on a per-frame basis
  - 8 clock and transfer attribute registers
  - Serial clock with programmable polarity and phase
  - Programmable delays
    - CS to SCK delay
    - SCK to CS delay
    - Delay between frames
  - Programmable serial frame size of 4 to 16 bits, expandable with software control
  - Continuously held chip select capability
- 3 peripheral chip selects, expandable to 8 with external demultiplexer
- 2 DMA conditions for SPI queues residing in RAM or flash
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
- 6 interrupt conditions:
    - End of queue reached (EOQF)
    - TX FIFO is not full (TFFF)
    - Transfer of current frame complete (TCF)

- RX FIFO is not empty (RFDF)
- FIFO overrun (attempt to transmit with an empty TX FIFO or serial frame received while RX FIFO is full) (RFOF)
- FIFO under flow (slave only and SPI mode, the slave is asked to transfer data when the TX FIFO is empty) (TFUF)

- Modified SPI transfer formats for communication with slower peripheral devices
- Continuous serial communications clock (SCK)

## 10.5 Modes of operation

The DSPI has five modes of operation. These modes can be divided into two categories:

- Module-specific modes such as master, slave, and module disable modes
- MCU-specific modes such as external stop and debug modes

The module-specific modes are entered by host software writing to a register. The MCU-specific modes are controlled by signals external to the DSPI. The MCU-specific modes are modes that the entire device may enter, in parallel to the DSPI being in one of its module-specific modes.

### 10.5.1 Master mode

Master mode allows the DSPI to initiate and control serial communication. In this mode the SCK, and CS*n* signals are controlled by the DSPI and configured as outputs. (SOUT is always an output.)

For more information, refer to Section 10.9.1.1, “Master Mode.

### 10.5.2 Slave mode

Slave mode allows the DSPI to communicate with SPI bus masters. In this mode the DSPI responds to externally controlled serial transfers. The DSPI cannot initiate serial transfers in slave mode. In slave mode, the SCK signal and the CS0_*x* signal are configured as inputs and provided by a bus master. CS0_*x* must be configured as input and pulled high. If the internal pullup is being used then the appropriate bits in the relevant SIU_PCR must be set (SIU_PCR [WPE = 1], [WPS = 1]).

- All frames need to be masked when operating in Fast Continuous Mode.(User cannot switch to continuous mode (delays not masked) and come back to the Fast Continuous Mode when Fast Continuous Mode of operation is selected).
- CPHA is to be kept "1" when operating in Fast Continuous Mode.

For more information, refer to Section 10.9.1.2, “Slave Mode.

### 10.5.3 Module disable mode

The module disable mode is used for MCU power management. The clock to the non-memory mapped logic in the DSPI is stopped while in module disable mode. The DSPI enters the module disable mode when the MDIS bit in DSPI*x*_MCR is set.

For more information, refer to Section 10.9.1.3, “Module Disable Mode.

### 10.5.4 External stop mode

The external Stop Mode is used for MCU power management. The DSPI supports the IPI Green-Line Interface Stop Mode mechanism. When a request is made to enter External Stop Mode, the DSPI block acknowledges the request and completes the transfer in progress. When the DSPI reaches the frame boundary it signals that the system clocks to the DSPI block may be shut off.

### 10.5.5 Debug mode

Debug mode is used for system development and debugging. If the device enters debug mode while the FRZ bit in the DSPI*x*_MCR is set, the DSPI halts operation on the next frame boundary. If the device enters debug mode while the FRZ bit is cleared, the DSPI behavior is unaffected and remains dictated by the module-specific mode and configuration of the DSPI.

For more information, refer to Section 10.9.1.5, “Debug Mode.

## 10.6 Device-specific information

This device implements DSPI 0, DSPI 1, and DSPI 2. The "x" appended to signal names signifies the DSPI module to which the signal applies. Thus CS0_0 is the CS0 signal that applies to DSPI 0, CS0_1 is the CS0 signal that applies to DSPI 1, and so on.

## 10.7 External signal description

### 10.7.1 Signal overview

Table 10-1 lists off-chip DSPI signals.

**Table 10-1. Signal properties**

| Name | I/O Type | Function | |
|---|---|---|---|
| | | Master Mode | Slave Mode |
| CS0_*x* | Output / input | Peripheral chip select 0 | Slave select |
| CS1:2_x | Output | Peripheral chip select 1–2 | Unused[1] |
| SIN_x | Input | Serial data in | Serial data in |
| SOUT_x | Output | Serial data out | Serial data out |
| SCK_x | Output / input | Serial clock (output) | Serial clock (input) |

1 The SIU allows you to select alternate pin functions for the device.

### 10.7.2 Signal names and descriptions

#### 10.7.2.1 Peripheral Chip Select / Slave Select (CS_0)

In master mode, the CS_0 signal is a peripheral chip select output that selects the slave device to which the current transmission is intended.

In slave mode, the CS_0 signal is a slave select input signal that allows an SPI master to select the DSPI as the target for transmission. CS_0 must be configured as input and pulled high. If the internal pullup is being used then the appropriate bits in the relevant SIU_PCR must be set (SIU_PCR [WPE = 1], [WPS = 1]).

Set the IBE and OBE bits in the SIU_PCR for all CS_0 pins when the DSPI chip select or slave select primary function is selected for that pin. When the pin is used for DSPI master mode as a chip select output, set the OBE bit. When the pin is used in DSPI slave mode as a slave select input, set the IBE bit.

#### 10.7.2.2 Peripheral Chip Selects 1–2 (CS1:2)

CS1:2 are peripheral chip select output signals in master mode. In slave mode these signals are not used.

#### 10.7.2.3 Serial Input (SIN_*x*)

SIN_*x* is a serial data input signal.

#### 10.7.2.4 Serial Output (SOUT_*x*)

SOUT_*x* is a serial data output signal.

#### 10.7.2.5 Serial Clock (SCK_*x*)

SCK_*x* is a serial communication clock signal. In master mode, the DSPI generates the SCK. In slave mode, SCK_*x* is an input from an external bus master.

## 10.8 Memory map and register description

### 10.8.1 Memory map

Table 10-2 shows the DSPI memory map.

**Table 10-2. DSPI detailed memory map**

| Address | Register description | Location |
|---|---|---|
| Base:<br>0xFFF9_0000 (DSPI 0)<br>0xFFF9_4000 (DSPI 1)<br>0xFFF9_8000 (DSPI 2) | DSPI module configuration register (DSPI*x*_MCR) | on page 292 |
| Base + 0x0004 | Reserved | — |
| Base + 0x0008 | DSPI transfer count register (DSPI*x*_TCR) | on page 295 |

Table 10-2. DSPI detailed memory map (continued)

| Address | Register description | Location |
|---|---|---|
| Base + 0x000C | DSPI clock and transfer attributes register 0 (DSPI*x*_CTAR0) | on page 296 |
| Base + 0x0010 | DSPI clock and transfer attributes register 1 (DSPI*x*_CTAR1) | on page 296 |
| Base + 0x0014 | DSPI clock and transfer attributes register 2 (DSPI*x*_CTAR2) | on page 296 |
| Base + 0x0018 | DSPI clock and transfer attributes register 3 (DSPI*x*_CTAR3) | on page 296 |
| Base + 0x001C | DSPI clock and transfer attributes register 4 (DSPI*x*_CTAR4) | on page 296 |
| Base + 0x0020 | DSPI clock and transfer attributes register 5 (DSPI*x*_CTAR5) | on page 296 |
| Base + 0x0024 | DSPI clock and transfer attributes register 6 (DSPI*x*_CTAR6) | on page 296 |
| Base + 0x0028 | DSPI clock and transfer attributes register 7 (DSPI*x*_CTAR7) | on page 296 |
| Base + 0x002C | DSPI status register (DSPI*x*_SR) | on page 302 |
| Base + 0x0030 | DSPI DMA/interrupt request select and enable register (DSPI*x*_RSER) | on page 304 |
| Base + 0x0034 | DSPI push TX FIFO register (DSPI*x*_PUSHR) | on page 306 |
| Base + 0x0038 | DSPI pop RX FIFO register (DSPI*x*_POPR) | on page 308 |
| Base + 0x003C | DSPI transmit FIFO register 0 (DSPI*x*_TXFR0) | on page 309 |
| Base + 0x0040 | DSPI transmit FIFO register 1 (DSPI*x*_TXFR1) | on page 309 |
| Base + 0x0044 | DSPI transmit FIFO register 2 (DSPI*x*_TXFR2) | on page 309 |
| Base + 0x0048 | DSPI transmit FIFO register 3 (DSPI*x*_TXFR3) | on page 309 |
| Base + 0x004C | DSPI transmit FIFO register 4 (DSPI*x*_TXFR4) | on page 309 |
| Base + 0x0050–<br>Base + 0x0078 | Reserved | — |
| Base + 0x007C | DSPI receive FIFO register 0 (DSPI*x*_RXFR0) | on page 309 |
| Base + 0x0080 | DSPI receive FIFO register 1 (DSPI*x*_RXFR1) | on page 309 |
| Base + 0x0084 | DSPI receive FIFO register 2 (DSPI*x*_RXFR2) | on page 309 |
| Base + 0x0088 | DSPI receive FIFO register 3 (DSPI*x*_RXFR3) | on page 309 |
| Base + 0x008C | DSPI receive FIFO register 4 (DSPI*x*_RXFR4) | on page 309 |
| Base + 0x0090–<br>Base + 0x00CC | Reserved | — |

## 10.8.2 Register description

### 10.8.2.1 DSPI Module Configuration Register (DSPI*x*_MCR)

The DSPI*x*_MCR contains bits which configure attributes of the DSPI operation. The values of the HALT and MDIS bits can be changed at any time, but their effect begins on the next frame boundary. The HALT and MDIS bits in the DSPI*x*_MCR are the only bit values software can change while the DSPI is running.

Address: Base + 0x0000 Access: R/W

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MSTR | CONT_SCKE | DCONF | | FRZ | MTFE | 0 | ROOE | 0 | 0 | PCSIS5 | PCSIS4 | PCSIS3 | PCSIS2 | PCSIS1 | PCSIS0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | MDIS | DIS_TXF | DIS_RXF | CLR_TXF | CLR_RXF | SMPL_PT | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | HALT |
| W | | | | | w1c | w1c | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 10-3. DSPI Module Configuration Register (DSPI*x*_MCR)**

Table 10-3 describes the fields in the DSPI module configuration register.

**Table 10-3. DSPI*x*_MCR Field Descriptions**

| Field | Description |
|---|---|
| 0<br>MSTR | Master/slave mode select. Configures the DSPI for master mode or slave mode.<br><br>0 DSPI is in slave mode<br>1 DSPI is in master mode |
| 1<br>CONT_SCKE | Continuous SCK enable. Enables the serial communication clock (SCK) to run continuously. Refer to Section 10.9.6, “Continuous Serial Communications Clock, for details.<br><br>0 Continuous SCK disabled<br>1 Continuous SCK enabled |
| 2–3<br>DCONF<br>[0:1] | DSPI configuration. The following table lists the DCONF values for the various configurations.<br><br>**DCONF** / **Configuration**<br>00 — SPI<br>01 — Invalid value<br>10 — Invalid value<br>11 — Invalid value |
| 4<br>FRZ | Freeze. Enables the DSPI transfers to be stopped on the next frame boundary when the device enters debug mode.<br><br>0 Do not halt serial transfers<br>1 Halt serial transfers |

**Table 10-3. DSPI*x*_MCR Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>MTFE | Modified timing format enable. Enables a modified transfer format to be used. Refer to Section 10.9.5.4, "Modified SPI Transfer Format (MTFE = 1, CPHA = 1), for more information.<br><br>0 Modified SPI transfer format disabled<br>1 Modified SPI transfer format enabled |
| 6 | Reserved. This bit is writable, but has no effect. |
| 7<br>ROOE | Receive FIFO overflow overwrite enable. Enables an RX FIFO overflow condition to ignore the incoming serial data or to overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer that generated the overflow is ignored or put in the shift register.<br><br>If the ROOE bit is set, the incoming data is put in the shift register. If the ROOE bit is cleared, the incoming data is ignored. Refer to Section 10.9.7.6, "Receive FIFO Overflow Interrupt Request (RFOF), for more information.<br><br>0 Incoming data is ignored<br>1 Incoming data is put in the shift register |
| 8–9 | Reserved, but implemented. These bits are writable, but have no effect. |
| 10–15<br>PCSIS*n* | Peripheral chip select inactive state. Determines the inactive state of the CS0_*x* signal. CS0_*x* must be configured as inactive high for slave mode operation.<br><br>0 The inactive state of CS0_*x* is low<br>1 The inactive state of CS0_*x* is high |
| 16 | Reserved. |
| 17<br>MDIS | Module disable. Allows the clock to stop to the non-memory mapped logic in the DSPI, effectively putting the DSPI in a software controlled power-saving state. Refer to Section 10.9.8, "Power Saving Features," for more information. The reset value of the MDIS bit is parameterized, with a default reset value of 0.<br><br>0 Enable DSPI clocks<br>1 Allow external logic to disable DSPI clocks |
| 18<br>DIS_TXF | Disable transmit FIFO. Enables and disables the TX FIFO. When the TX FIFO is disabled, the transmit part of the DSPI operates as a simplified double-buffered SPI. Refer to Section 10.9.3.3, "FIFO Disable Operation," for details.<br><br>0 TX FIFO is enabled<br>1 TX FIFO is disabled |
| 19<br>DIS_RXF | Disable receive FIFO. Enables and disables the RX FIFO. When the RX FIFO is disabled, the receive part of the DSPI operates as a simplified double-buffered SPI. Refer to Section 10.9.3.3, "FIFO Disable Operation," for details.<br>0 RX FIFO is enabled<br>1 RX FIFO is disabled |
| 20<br>CLR_TXF | Clear TX FIFO. Flushes the TX FIFO. Write a 1 to the CLR_TXF bit to clear the TX FIFO counter. The CLR_TXF bit is always read as zero.<br>0 Do not clear the TX FIFO counter<br>1 Clear the TX FIFO counter |

**Table 10-3. DSPI*x*_MCR Field Descriptions (continued)**

| Field | Description |
|---|---|
| 21<br>CLR_RXF | Clear RX FIFO. Flushes the RX FIFO. Write a 1 to the CLR_RXF bit to clear the RX counter. The CLR_RXF bit is always read as zero.<br>0 Do not clear the RX FIFO counter<br>1 Clear the RX FIFO counter |
| 22–23<br>SMPL_<br>PT<br>[0:1] | Sample point. Allows the host software to select when the DSPI master samples SIN in modified transfer format. Figure 10-16 shows where the master can sample the SIN pin. The following table lists the delayed sample points.<br><br>**SMPL_PT** / **Number of system clock cycles between odd-numbered edge of SCK_*x* and sampling of SIN_*x*.**<br>00 / 0<br>01 / 1<br>10 / 2<br>11 / Invalid value |
| 24–30 | Reserved. |
| 31<br>HALT | Halt. Provides a mechanism for software to start and stop DSPI transfers. Refer to Section 10.9.2, "Start and Stop of DSPI Transfers, for details on the operation of this bit.<br>0 Start transfers<br>1 Stop transfers |

### 10.8.2.2 DSPI Transfer Count Register (DSPI*x*_TCR)

The DSPI*x*_TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. The user must not write to the DSPI*x*_TCR while the DSPI is running.

Address: Base + 0x0008 Access: R/W

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SPI_TCNT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 10-4. DSPI Transfer Count Register (DSPI*x*_TCR)**

Table 10-4 describes the field in the DSPI transfer count register.

**Table 10-4. DSPI*x*_TCR Field Descriptions**

| Field | Description |
|---|---|
| 0–15<br>SPI_TCNT<br>[0:15] | SPI transfer counter. Counts the number of SPI transfers the DSPI makes. The SPI_TCNT field is incremented every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The transfer counter 'wraps around,' incrementing the counter past 65535 resets the counter to zero. |
| 16–31 | Reserved. |

### 10.8.2.3 DSPI Clock and Transfer Attributes Registers 0–7 (DSPI*x*_CTAR*n*)

The DSPI modules each contain eight clock and transfer attribute registers (DSPI*x*_CTAR*n*) which are used to define different transfer attribute configurations. Each DSPI*x*_CTAR controls:

- Frame size
- Baud rate and transfer delay values
- Clock phase
- Clock polarity
- MSB or LSB first

At the initiation of an SPI transfer, control logic selects the DSPI*x*_CTAR that contains the transfer's attributes.Do not write to the DSPI*x*_CTARs while the DSPI is running.

In master mode, the DSPI*x*_CTAR*n* registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bit fields in the DSPI*x*_CTAR0 and DSPI*x*_CTAR1 registers are used to set the slave transfer attributes. Refer to the individual bit descriptions for details on which bits are used in slave modes.

When the DSPI is configured as an SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the DSPI*x*_CTAR registers is used on a per-frame basis. When the DSPI is configured as an SPI bus slave, the DSPI*x*_CTAR0 register is used.

Address:
Base + 0x000C (DSPI*x*_CTAR0)
Base + 0x0010 (DSPI*x*_CTAR1)
Base + 0x0014 (DSPI*x*_CTAR2)
Base + 0x0018 (DSPI*x*_CTAR3)
Base + 0x001C (DSPIx_CTAR4)
Base + 0x0020 (DSPI*x*_CTAR5)
Base + 0x0024 (DSPI*x*_CTAR6)
Base + 0x0028 (DSPI*x*_CTAR7)

Access: R/W

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DBR | FMSZ | | | | CPOL | CPHA | LSBFE | PCSSCK | | PASC | | PDT | | PBR | |
| Reset | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CSSCK | | | | ASC | | | | DT | | | | BR | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-5. DSPI Clock and Transfer Attributes Registers 0–7 (DSPI*x*_CTARn)**

**Table 10-5. DSPI*x*_CTAR*n* Field Descriptions**

| Field | Descriptions |
|---|---|
| 0<br>DBR | Double Baud Rate. The DBR bit doubles the effective baud rate of the Serial Communications Clock (SCK). This field is only used in Master Mode. It effectively halves the Baud Rate division ratio supporting faster frequencies and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in Table 10-6. See the BR field description for details on how to compute the baud rate. If the overall baud rate is divide by two or divide by three of the system clock then neither the Continuous SCK Enable or the Modified Timing Format Enable bits should be set.<br>0 The baud rate is computed normally with a 50/50 duty cycle<br>1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler |
| 1–4<br>FMSZ[0:3] | Frame Size. The FMSZ field selects the number of bits transferred per frame. The FMSZ field is used in Master Mode and Slave Mode. Table 10-7 lists the frame size encodings.<br>When operating in TSB confirmation, the FMSZ defines the point with in the 32-bit (maximum length) frame where control of the CS switches from the DSPI_DSICR to the DSPI_DSICR1 register. The cross over point must range between 4 bits and 16 bits and is encoded per Table 10-7. The remaining frame after the cross over point, regardless of how many bits are remaining, will be controlled by the DSPI_DSICR1 register. |
| 5<br>CPOL | Clock Polarity. The CPOL bit selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both Master and Slave Mode. For successful communication between serial devices, the devices must have identical clock polarities. When the continuous selection format (see Section 10.9.5.5, “Continuous Selection Format”) is selected, switching between clock polarities without stopping the DSPI can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge.<br>0 The inactive state value of SCK is low<br>1 The inactive state value of SCK is high |

**Table 10-5. DSPI*x*_CTAR*n* Field Descriptions (continued)**

<table>
<tr><th>Field</th><th>Descriptions</th></tr>
<tr><td>6<br>CPHA</td><td>Clock Phase. The CPHA bit selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both Master and Slave Mode. For successful communication between serial devices, the devices must have identical clock phase settings. Continuous SCK is only supported for CPHA=1.<br>0 Data is captured on the leading edge of SCK and changed on the following edge<br>1 Data is changed on the leading edge of SCK and captured on the following edge</td></tr>
<tr><td>7<br>LSBFE</td><td>LSB First. The LSBFE bit selects if the LSB or MSB of the frame is transferred first. This bit is only used in Master Mode. When operating in TSB configuration, this bit should be always 1.<br>0 Data is transferred MSB first<br>1 Data is transferred LSB first</td></tr>
<tr><td>8–9<br>PCSSCK[0:1]</td><td>PCS to SCK Delay Prescaler. The PCSSCK field selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. This field is only used in Master Mode. The table below lists the prescaler values. See the CSSCK[0:3] field description for details on how to compute the PCS to SCK Delay.
<table>
<tr><th>PCSSCK</th><th>PCS to SCK Delay Prescaler Value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>10–11<br>PASC[0:1]</td><td>After SCK Delay Prescaler. The PASC field selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. This field is only used in Master Mode. The table below lists the prescaler values. See the ASC[0:3] field description for details on how to compute the After SCK Delay.
<table>
<tr><th>PASC</th><th>After SCK Delay Prescaler Value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
</table>

**Table 10-5. DSPI*x*_CTAR*n* Field Descriptions (continued)**

<table>
<tr><th>Field</th><th>Descriptions</th></tr>
<tr><td>12–13<br>PDT[0:1]</td><td>Delay after Transfer Prescaler. The PDT field selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in Master Mode. The table below lists the prescaler values. See the DT[0:3] field description for details on how to compute the Delay after Transfer.
<table>
<tr><th>PDT</th><th>Delay after Transfer Prescaler Value</th></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>14–15<br>PBR[0:1]</td><td>Baud Rate Prescal. The PBR field selects the prescaler value for the baud rate. This field is only used in Master Mode. The Baud Rate is the frequency of the Serial Communications Clock (SCK). The system clock is divided by the prescaler value before the baud rate selection takes place. The Baud Rate Prescaler values are listed in the table below. See the BR[0:3] field description for details on how to compute the baud rate.
<table>
<tr><th>PBR</th><th>Baud Rate Prescaler Value</th></tr>
<tr><td>00</td><td>2</td></tr>
<tr><td>01</td><td>3</td></tr>
<tr><td>10</td><td>5</td></tr>
<tr><td>11</td><td>7</td></tr>
</table></td></tr>
<tr><td>16–19<br>CSSCK[0:3]</td><td>PCS to SCK Delay Scaler. The CSSCK field selects the scaler value for the PCS to SCK delay. This field is only used in Master Mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. Table 10-8 list the scaler values.The PCS to SCK Delay is a multiple of the system clock period and it is computed according to the following equation:<br>$$t_{CSC} = \frac{1}{f_{SYS}} \times PCSSCK \times CSSCK$$ Eqn. 10-1</td></tr>
<tr><td>20–23<br>ASC[0:3]</td><td>After SCK Delay Scaler. The ASC field selects the scaler value for the After SCK Delay. This field is only used in Master Mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. Table 10-9 list the scaler values.The After SCK Delay is a multiple of the system clock period, and it is computed according to the following equation:<br>$$t_{ASC} = \frac{1}{f_{SYS}} \times PASC \times ASC$$ Eqn. 10-2</td></tr>
</table>

**Table 10-5. DSPI*x*_CTAR*n* Field Descriptions (continued)**

| Field | Descriptions |
|---|---|
| 24–27<br>DT[0:3] | Delay after Transfer Scaler. The DT field selects the Delay after Transfer Scaler. This field is only used in Master Mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. Table 10-10 lists the scaler values. In the Continuous Serial Communications Clock operation the DT value is fixed to one TSCK, except when the TSBC bit from DSPI_DSICR register is enabled the TSB configuration. The Delay after Transfer is a multiple of the system clock period and it is computed according to the following equation:<br>$t_{DT} = \frac{1}{f_{SYS}} \times PDT \times DT$ ***Eqn. 10-3*** |
| 28–31<br>BR[0:3] | Baud Rate Scaler. The BR field selects the scaler value for the baud rate. This field is only used in Master Mode. The prescaled system clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. Table 10-11 lists the Baud Rate Scaler values.The baud rate is computed according to the following equation:<br>$\text{SCK baud rate} = \frac{f_{SYS}}{PBR} \times \frac{1 + DBR}{BR}$ ***Eqn. 10-4*** |

**Table 10-6. DSPI SCK Duty Cycle**

| DBR | CPHA | PBR | SCK Duty Cycle |
|---|---|---|---|
| 0 | any | any | 50/50 |
| 1 | 0 | 00 | 50/50 |
| 1 | 0 | 01 | 33/66 |
| 1 | 0 | 10 | 40/60 |
| 1 | 0 | 11 | 43/57 |
| 1 | 1 | 00 | 50/50 |
| 1 | 1 | 01 | 66/33 |
| 1 | 1 | 10 | 60/40 |
| 1 | 1 | 11 | 57/43 |

**Table 10-7. DSPI Transfer Frame Size**

| FMSZ | Framesize | FMSZ | Framesize |
|---|---|---|---|
| 0000 | Reserved | 1000 | 9 |
| 0001 | Reserved | 1001 | 10 |
| 0010 | Reserved | 1010 | 11 |
| 0011 | 4 | 1011 | 12 |
| 0100 | 5 | 1100 | 13 |
| 0101 | 6 | 1101 | 14 |
| 0110 | 7 | 1110 | 15 |
| 0111 | 8 | 1111 | 16 |

**Table 10-8. DSPI PCS to SCK Delay Scaler**

| CSSCK | PCS to SCK Delay Scaler Value | CSSCK | PCS to SCK Delay Scaler Value |
|---|---|---|---|
| 0000 | 2 | 1000 | 512 |
| 0001 | 4 | 1001 | 1024 |
| 0010 | 8 | 1010 | 2048 |
| 0011 | 16 | 1011 | 4096 |
| 0100 | 32 | 1100 | 8192 |
| 0101 | 64 | 1101 | 16384 |
| 0110 | 128 | 1110 | 32768 |
| 0111 | 256 | 1111 | 65536 |

**Table 10-9. DSPI After SCK Delay Scaler**

| ASC | After SCK Delay Scaler Value | ASC | After SCK Delay Scaler Value |
|---|---|---|---|
| 0000 | 2 | 1000 | 512 |
| 0001 | 4 | 1001 | 1024 |
| 0010 | 8 | 1010 | 2048 |
| 0011 | 16 | 1011 | 4096 |
| 0100 | 32 | 1100 | 8192 |
| 0101 | 64 | 1101 | 16384 |
| 0110 | 128 | 1110 | 32768 |
| 0111 | 256 | 1111 | 65536 |

**Table 10-10. DSPI Delay after Transfer Scaler**

| DT | Delay after Transfer Scaler Value | DT | Delay after Transfer Scaler Value |
|---|---|---|---|
| 0000 | 2 | 1000 | 512 |
| 0001 | 4 | 1001 | 1024 |
| 0010 | 8 | 1010 | 2048 |
| 0011 | 16 | 1011 | 4096 |
| 0100 | 32 | 1100 | 8192 |
| 0101 | 64 | 1101 | 16384 |
| 0110 | 128 | 1110 | 32768 |
| 0111 | 256 | 1111 | 65536 |

**Table 10-11. DSPI Baud Rate Scaler**

| BR | Baud Rate Scaler Value | BR | Baud Rate Scaler Value |
|---|---|---|---|
| 0000 | 2 | 1000 | 256 |
| 0001 | 4 | 1001 | 512 |
| 0010 | 6 | 1010 | 1024 |
| 0011 | 8 | 1011 | 2048 |
| 0100 | 16 | 1100 | 4096 |
| 0101 | 32 | 1101 | 8192 |
| 0110 | 64 | 1110 | 16384 |
| 0111 | 128 | 1111 | 32768 |

### 10.8.2.4 DSPI Status Register (DSPI*x*_SR)

The DSPI*x*_SR contains status and flag bits. The bits are set by the hardware and reflect the status of the DSPI and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the DSPI*x*_SR by writing a 1 to clear it (w1c). Writing a 0 to a flag bit has no effect.

Address: Base + 0x002C Access: R/W

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCF | TXRXS | 0 | EOQF | TFUF | 0 | TFFF | 0 | 0 | 0 | 0 | 0 | RFOF | 0 | RFDF | 0 |
| W | w1c | | | w1c | w1c | | w1c | | | | | | w1c | | w1c | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TXCTR | | | | TXNXTPTR | | | | RXCTR | | | | POPNXTPTR | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DSPI Status Register (DSPI*x*_SR)

Table 10-12 describes the fields in the DSPI status register.

**Table 10-12. DSPI*x*_SR Field Descriptions**

| Field | Description |
|---|---|
| 0<br>TCF | Transfer complete flag. Indicates that all bits in a frame have been shifted out. The TCF bit is set after the last incoming databit is sampled, but before the tASC delay starts. Refer to Section 10.9.5.1, “Classic SPI Transfer Format (CPHA = 0) for details. The TCF bit is cleared by writing 1 to it.<br><br>0 Transfer not complete<br>1 Transfer complete |
| 1<br>TXRXS | TX and RX status. Reflects the status of the DSPI. Refer to Section 10.9.2, “Start and Stop of DSPI Transfers for information on what clears and sets this bit.<br><br>0 TX and RX operations are disabled (DSPI is in STOPPED state)<br>1 TX and RX operations are enabled (DSPI is in RUNNING state) |
| 2 | Reserved. |
| 3<br>EOQF | End of queue flag. Indicates that transmission in progress is the last entry in a queue. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and after the last incoming databit is sampled, but before the tASC delay starts. Refer to Section 10.9.5.1, “Classic SPI Transfer Format (CPHA = 0) for details.<br><br>The EOQF bit is cleared by writing 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared.<br>0 EOQ is not set in the executing command<br>1 EOQ bit is set in the executing SPI command<br>**Note:** EOQF does not function in slave mode. |
| 4<br>TFUF | Transmit FIFO underflow flag. Indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for DSPI modules operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI operating in SPI slave mode is empty, and a transfer is initiated by an external SPI master. The TFUF bit is cleared by writing 1 to it.<br><br>0 TX FIFO underflow has not occurred<br>1 TX FIFO underflow has occurred |
| 5 | Reserved. |
| 6<br>TFFF | Transmit FIFO fill flag. Indicates that the TX FIFO can be filled. Provides a method for the DSPI to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing 1 to it, or an by acknowledgement from the eDMA controller when the TX FIFO is full.<br><br>0 TX FIFO is full<br>1 TX FIFO is not full |
| 7–11 | Reserved. |
| 12<br>RFOF | Receive FIFO overflow flag. Indicates that an overflow condition in the RX FIFO has occurred. The bit is set when the RX FIFO and shift register are full and a transfer is initiated. The bit is cleared by writing 1 to it.<br><br>0 RX FIFO overflow has not occurred<br>1 RX FIFO overflow has occurred |
| 13 | Reserved. |

**Table 10-12. DSPI*x*_SR Field Descriptions (continued)**

| Field | Description |
|---|---|
| 14<br>RFDF | Receive FIFO drain flag. Indicates that the RX FIFO can be drained. Provides a method for the DSPI to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it, or by acknowledgement from the eDMA controller when the RX FIFO is empty.<br><br>0 RX FIFO is empty<br>1 RX FIFO is not empty<br>**Note:** In the interrupt service routine, RFDF must be cleared only after the DSPIx_POPR register is read. |
| 15 | Reserved. |
| 16–19<br>TXCTR<br>[0:3] | TX FIFO counter. Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the DSPI _PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register. |
| 20–23<br>TXNXTPTR<br>[0:3] | Transmit next pointer. Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register. Refer to Section 10.9.3.4, “Transmit First In First Out (TX FIFO) Buffering Mechanism for more details. |
| 24–27<br>RXCTR<br>[0:3] | RX FIFO counter. Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the DSPI _POPR is read. The RXCTR is incremented after the last incoming databit is sampled, but before the tASC delay starts. Refer to Section 10.9.5.1, “Classic SPI Transfer Format (CPHA = 0) for details. |
| 28–31<br>POPNXTPTR<br>[0:3] | Pop next pointer. Contains a pointer to the RX FIFO entry that is returned when the DSPI*x*_POPR is read. The POPNXTPTR is updated when the DSPI*x*_POPR is read. Refer to Section 10.9.3.5, “Receive First In First Out (RX FIFO) Buffering Mechanism for more details. |

## 10.8.2.5 DSPI DMA / Interrupt Request Select and Enable Register (DSPI*x*_RSER)

The DSPI*x*_RSER serves two purposes: enables flag bits in the DSPI*x*_SR to generate DMA requests or interrupt requests, and selects the type of request to generate. Refer to the bit descriptions for the type of requests that are supported. Do not write to the DSPI*x*_RSER while the DSPI is running.

Address: Base + 0x0030 Access: R/W

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCF_RE | 0 | 0 | EOQF_RE | TFUF_RE | 0 | TFFF_RE | TFFF_DIRS | 0 | 0 | 0 | 0 | RFOF_RE | 0 | RFDF_RE | RFDF_DIRS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 10-6. DSPI DMA / Interrupt Request Select and Enable Register (DSPI*x*_RSER)**

Table 10-13 describes the fields in the DSPI DMA / interrupt request and enable register.

**Table 10-13. DSPI*x*_RSER Field Descriptions**

| Field | Description |
|---|---|
| 0<br>TCF_RE | Transmission complete request enable. Enables TCF flag in the DSPI*x*_SR to generate an interrupt request.<br><br>0 TCF interrupt requests are disabled<br>1 TCF interrupt requests are enabled |
| 1–2 | Reserved. |
| 3<br>EOQF_RE | DSPI finished request enable. Enables the EOQF flag in the DSPI*x*_SR to generate an interrupt request.<br><br>0 EOQF interrupt requests are disabled<br>1 EOQF interrupt requests are enabled |
| 4<br>TFUF_RE | Transmit FIFO underflow request enable. The TFUF_RE bit enables the TFUF flag in the DSPI*x*_SR to generate an interrupt request.<br><br>0 TFUF interrupt requests are disabled<br>1 TFUF interrupt requests are enabled |
| 5 | Reserved. |
| 6<br>TFFF_RE | Transmit FIFO fill request enable. Enables the TFFF flag in the DSPI*x*_SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA requests.<br><br>0 TFFF interrupt requests or DMA requests are disabled<br>1 TFFF interrupt requests or DMA requests are enabled |
| 7<br>TFFF_DIRS | Transmit FIFO fill DMA or interrupt request select. Selects between generating a DMA request or an interrupt request. When the TFFF flag bit in the DSPI*x*_SR is set, and the TFFF_RE bit in the DSPI*x*_RSER is set, this bit selects between generating an interrupt request or a DMA request.<br><br>0 Interrupt request is selected<br>1 DMA request is selected |
| 8–11 | Reserved. |
| 12<br>RFOF_RE | Receive FIFO overflow request enable. Enables the RFOF flag in the DSPI*x*_SR to generate an interrupt requests.<br><br>0 RFOF interrupt requests are disabled<br>1 RFOF interrupt requests are enabled |
| 13 | Reserved. |
| 14<br>RFDF_RE | Receive FIFO drain request enable. Enables the RFDF flag in the DSPI*x*_SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request.<br><br>0 RFDF interrupt requests or DMA requests are disabled<br>1 RFDF interrupt requests or DMA requests are enabled |

**Table 10-13. DSPI*x*_RSER Field Descriptions (continued)**

| Field | Description |
|---|---|
| 15<br>RFDF_DIRS | Receive FIFO drain DMA or interrupt request select. Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the DSPI*x*_SR is set, and the RFDF_RE bit in the DSPI*x*_RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request.<br><br>0 Interrupt request is selected<br>1 DMA request is selected |
| 16–31 | Reserved. |

### 10.8.2.6 DSPI PUSH TX FIFO Register (DSPI*x*_PUSHR)

The DSPI*x*_PUSHR provides a means to write to the TX FIFO. Data written to this register is transferred to the TX FIFO. Refer to Section 10.9.3.4, “Transmit First In First Out (TX FIFO) Buffering Mechanism, for more information. Write accesses of 8- or 16-bits to the DSPI*x*_PUSHR transfers 32 bits to the TX FIFO.

**NOTE**

TXDATA is used in master and slave modes.

Address: Base + 0x0034 Access: R/W

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CONT | CTAS | | | EOQ | CT CNT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PCS 2 | PCS 1 | PCS 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 10-7. DSPI PUSH TX FIFO Register (DSPI*x*_PUSHR)**

Table 10-14 describes the fields in the DSPI push transmit FIFO register.

**Table 10-14. DSPI*x*_PUSHR Field Descriptions**

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0<br>CONT</td><td>Continuous peripheral chip select enable. Selects a continuous selection format. The bit is used in SPI master mode. The bit enables the selected CS signals to remain asserted between transfers. Refer to Section 10.9.5.5, “Continuous Selection Format, for more information.<br><br>0 Return peripheral chip select signals to their inactive state between transfers<br>1 Keep peripheral chip select signals asserted between transfers</td></tr>
<tr><td>1–3<br>CTAS<br>[0:2]</td><td>Clock and transfer attributes select. Selects which of the DSPIx_CTARs is used to set the transfer attributes for the SPI frame. In SPI slave mode, DSPIx_CTAR0 is used. The following table shows how the CTAS values map to the DSPIx_CTARs. There are eight DSPIx_CTARs in the device DSPI implementation.<br><br>Note: Use in SPI master mode only.<br><br>
<table>
<tr><th>CTAS</th><th>Use Clock and Transfer Attributes from</th></tr>
<tr><td>000</td><td>DSPIx_CTAR0</td></tr>
<tr><td>001</td><td>DSPIx_CTAR1</td></tr>
<tr><td>010</td><td>DSPIx_CTAR2</td></tr>
<tr><td>011</td><td>DSPIx_CTAR3</td></tr>
<tr><td>100</td><td>DSPIx_CTAR4</td></tr>
<tr><td>101</td><td>DSPIx_CTAR5</td></tr>
<tr><td>110</td><td>DSPIx_CTAR6</td></tr>
<tr><td>111</td><td>DSPIx_CTAR7</td></tr>
</table></td></tr>
<tr><td>4<br>EOQ</td><td>End of queue. Provides a means for host software to signal to the DSPI that the current SPI transfer is the last in a queue. At the end of the transfer the EOQF bit in the DSPIx_SR is set.<br><br>0 The SPI data is not the last data to transfer<br>1 The SPI data is the last data to transfer<br>Note: Use in SPI master mode only.</td></tr>
<tr><td>5<br>CTCNT</td><td>Clear SPI_TCNT. Provides a means for host software to clear the SPI transfer counter. The CTCNT bit clears the SPI_TCNT field in the DSPIx_TCR. The SPI_TCNT field is cleared before transmission of the current SPI frame begins.<br><br>0 Do not clear SPI_TCNT field in the DSPIx_TCR<br>1 Clear SPI_TCNT field in the DSPIx_TCR<br>Note: Use in SPI master mode only.</td></tr>
<tr><td>6–7</td><td>Reserved.</td></tr>
<tr><td>8–9</td><td>Reserved, but implemented. These bits are writable, but have no effect.</td></tr>
<tr><td>10–12</td><td>Reserved.</td></tr>
</table>

**Table 10-14. DSPI*x*_PUSHR Field Descriptions (continued)**

| Field | Description |
|---|---|
| 13–15<br>PCS*x* | Peripheral chip select *x*. Selects which CS*x* signals are asserted for the transfer.<br><br>0 Negate the CS*x* signal<br>1 Assert the CS*x* signal<br>**Note:** Use in SPI master mode only. |
| 16–31<br>TXDATA<br>[0:15] | Transmit data. Holds SPI data for transfer according to the associated SPI command.<br><br>**Note:** Use TXDATA in master and slave modes. |

### 10.8.2.7 DSPI POP RX FIFO Register (DSPI*x*_POPR)

The DSPI*x*_POPR allows you to read the RX FIFO. Refer to Section 10.9.3.5, “Receive First In First Out (RX FIFO) Buffering Mechanism for a description of the RX FIFO operations. Eight or 16-bit read accesses to the DSPI*x*_POPR fetch the RX FIFO data, and update the counter and pointer.

**NOTE**

Reading the RX FIFO field fetches data from the RX FIFO. Once the RX FIFO is read, the read data pointer is moved to the next entry in the RX FIFO. Therefore, read DSPI*x*_POPR only when you need the data. For compatibility, configure the TLB (MMU table) entry for DSPI*x*_POPR as guarded.

Address: Base + 0x0038 Access: R/O

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

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

**Figure 10-8. DSPI POP RX FIFO Register (DSPI*x*_POPR)**

Table 10-15 describes the fields in the DSPI pop receive FIFO register.

**Table 10-15. DSPI*x*_POPR Field Descriptions**

| Field | Description |
|---|---|
| 0–15 | Reserved, must be cleared. |
| 16–31 RXDATA [0:15] | Received data. The RXDATA field contains the SPI data from the RX FIFO entry pointed to by the pop next data pointer (POPNXTPTR). |

### 10.8.2.8 DSPI Transmit FIFO Registers 0–4 (DSPI*x*_TXFR*n*)

The DSPI*x*_TXFR*n* registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the DSPI*x*_TXFR*n* registers does not alter the state of the TX FIFO.

Address:
Base + 0x003C (DSPI*x*_TXFR0)
Base + 0x0040 (DSPI*x*_TXFR1)
Base + 0x0044 (DSPI*x*_TXFR2)
Base + 0x0048 (DSPI*x*_TXFR3)
Base + 0x004C (DSPI*x*_TXFR4)

Access: R/O

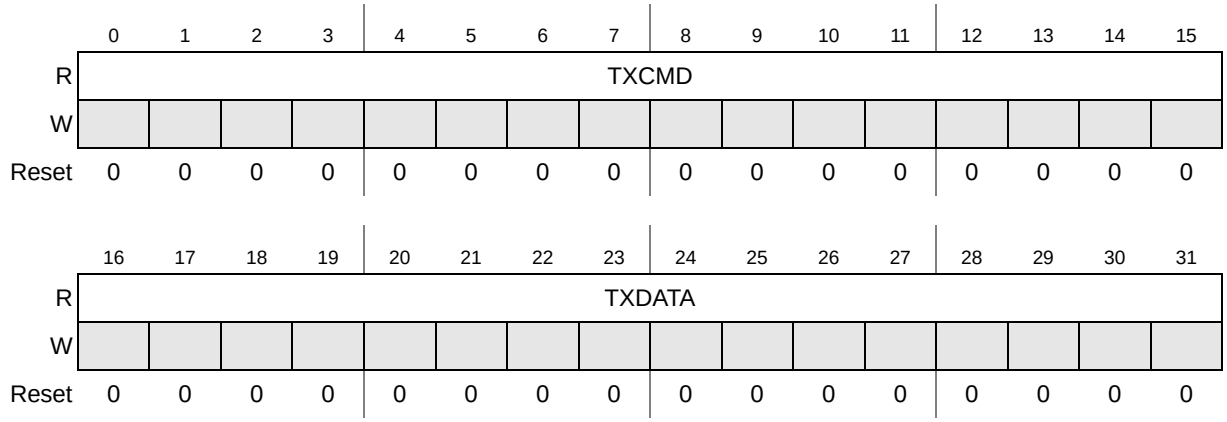



**Figure 10-9. DSPI Transmit FIFO Register 0–4 (DSPI*x*_TXFR*n*)**

Table 10-16 describes the fields in the DSPI transmit FIFO register.

**Table 10-16. DSPI*x*_TXFR*n* field descriptions**

| Field | Description |
|---|---|
| TXCMD | Transmit command. Contains the command that sets the transfer attributes for the SPI data. Refer to Section 10.8.2.6, “DSPI PUSH TX FIFO Register (DSPIx_PUSHR), for details on the command field. |
| TXDATA | Transmit data. Contains the SPI data to be shifted out. |

### 10.8.2.9 DSPI Receive FIFO Registers 0–4 (DSPI*x*_RXFR*n*)

The DSPI*x*_RXFR*n* registers provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The DSPI*x*_RXFR registers are read-only. Reading the DSPI*x*_RXFR*n* registers does not alter the state of the RX FIFO.

Address:
Base + 0x007C (DSPI*x*_RXFR0)
Base + 0x0080 (DSPI*x*_RXFR1)
Base + 0x0084 (DSPI*x*_RXFR2)
Base + 0x0088 (DSPI*x*_RXFR3)
Base + 0x008C (DSPI*x*_RXFR4)

Access: R/O

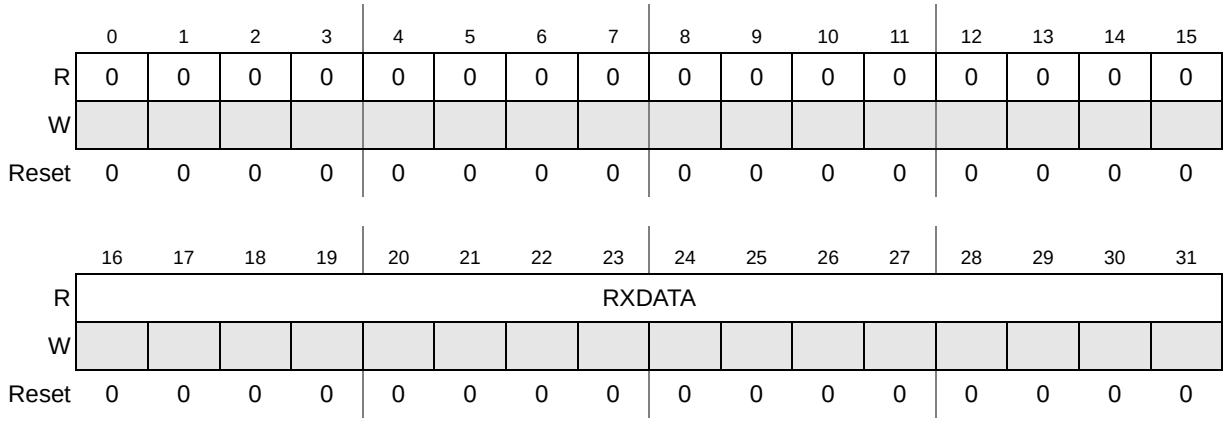



**Figure 10-10. DSPI Receive FIFO Registers 0–4 (DSPI*x*_RXFR*n*)**

Table 10-17 describes the field in the DSPI receive FIFO register.

**Table 10-17. DSPI*x*_RXFR*n* field descriptions**

| Field | Description |
|---|---|
| RXDATA | Receive data. Contains the received SPI data. |

## 10.9 Functional description

The DSPI supports full-duplex, synchronous serial communications between the MCU and peripheral devices. All communications are through an SPI-like protocol.

The DSPI has one configuration:

- Serial peripheral interface (SPI) configuration in which the DSPI operates as a basic SPI or a queued SPI.

The DCONF field in the DSPI*x*_MCR register determines the DSPI configuration. Refer to Table 10-3 for the DSPI configuration values.

The DSPI*x*_CTAR0–DSPI*x*_CTAR7 registers hold clock and transfer attributes.The SPI configuration can select which CTAR to use on a frame by frame basis by setting the CTAS field in the DSPI*x*_PUSHR.

The 16-bit shift register in the master and the 16-bit shift register in the slave are linked by the SOUT_*x* and SIN_*x* signals to form a distributed 32-bit register. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the registers are linked, data is exchanged between the master and the slave; the data that was in the master's shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the TCF bit in the DSPI*x*_SR is set to indicate a completed transfer. Figure 10-11 illustrates how master and slave data is exchanged.

**NOTE**

In Master and Slave standard DSPI transfer with FIFO enabled and using continuous PSC, if software does not provide correct number of frames in slave mode or does not program the CONT bit correctly, an underflow error is generated.

For correct operation:

In Master mode, frames that are to be transmitted under a continuous chip selection assertion (Continuous Selection Format) must have CONT=1, with the exception of last frame, where this bit will be set to 0.

In Slave mode, user software must ensure that the slave's Tx FIFO has the desired number of entries required in a continuous PCS transfer and when the last frame is sent out, the slave is de-selected to avoid an underflow error.

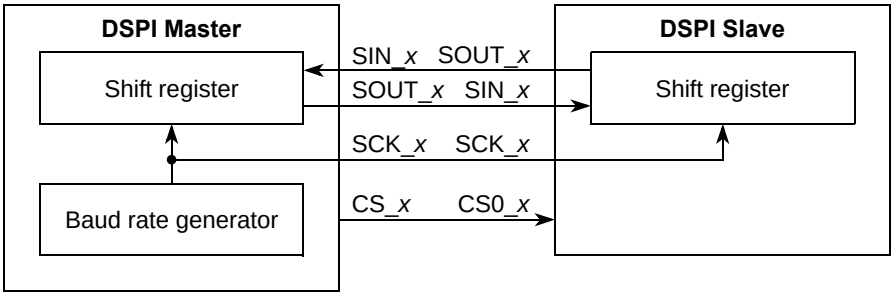



**Figure 10-11. SPI Serial Protocol Overview**

The DSPI has three peripheral chip select (CS*x*) signals that are be used to select which of the slaves to communicate with.

Transfer protocols and timing properties are shared by the three DSPI configurations; these properties are described independently of the configuration in Section 10.9.5, “Transfer Formats. The transfer rate and delay settings are described in section Section 10.9.4, “DSPI Baud Rate and Clock Delay Generation.

Refer to Section 10.9.8, “Power Saving Features for information on the power-saving features of the DSPI.

## 10.9.1 Modes of operation

The DSPI modules have five available distinct modes:

- Master mode
- Slave mode
- Module disable mode
- External stop mode
- Debug mode

Master, slave, and module disable modes are module-specific modes. The external stop and debug modes are device-specific modes.

The module-specific modes are determined by bits in the DSPI*x*_MCR. The device-specific modes are modes that the entire device can enter in parallel with the DSPI being configured in one of its module-specific modes.

#### 10.9.1.1 Master Mode

In master mode the DSPI can initiate communications with peripheral devices. The DSPI operates as bus master when the MSTR bit in the DSPI*x*_MCR is set. The serial communications clock (SCK) is controlled by the master DSPI. All three DSPI configurations are valid in master mode.

In SPI configuration, master mode transfer attributes are controlled by the SPI command in the current TX FIFO entry. The CTAS field in the SPI command selects which of the eight DSPI*x*_CTARs are used to set the transfer attributes. Transfer attribute control is on a frame by frame basis.

Refer to Section 10.9.3, “Serial Peripheral Interface (SPI) Configuration for more details.

#### 10.9.1.2 Slave Mode

In slave mode the DSPI responds to transfers initiated by an SPI master. The DSPI operates as bus slave when the MSTR bit in the DSPI*x*_MCR is negated. The DSPI slave is selected by a bus master by having the slave’s CS0_*x* asserted. In slave mode the SCK is provided by the bus master. All transfer attributes are controlled by the bus master, except the clock polarity, clock phase and the number of bits to transfer which must be configured in the DSPI slave to communicate correctly.

#### 10.9.1.3 Module Disable Mode

The module disable mode is used for MCU power management. The clock to the non-memory mapped logic in the DSPI is stopped while in module disable mode. The DSPI enters the module disable mode when the MDIS bit in DSPI*x*_MCR is set.

Refer to Section 10.9.8, “Power Saving Features, for more details on the module disable mode.

#### 10.9.1.4 External Stop Mode

For devices with low-power modes, the DSPI supports the Global Signal Stop Mode mechanism. The DSPI will not acknowledge the request to enter External Stop Mode until it has reached a frame boundary. When the DSPI has reached a frame boundary it will halt all operations and indicate that it is ready to have its clocks shut off. The DSPI exits External Stop Mode and resumes normal operation once the clocks are turned on. Serial communications or register accesses made while in External Stop Mode are ignored even if the clocks have not been shut off yet. See Section 10.9.8, “Power Saving Features,” for more details on the External Stop Mode.

### 10.9.1.5 Debug Mode

The debug mode is used for system development and debugging. If the MCU enters debug mode while the FRZ bit in the DSPI*x*_MCR is set, the DSPI stops all serial transfers and enters a stopped state. If the MCU enters debug mode while the FRZ bit is cleared, the DSPI behavior is unaffected and remains dictated by the module-specific mode and configuration of the DSPI. The DSPI enters debug mode when a debug request is asserted by an external controller.

Refer to Figure 10-12 for a state diagram.

## 10.9.2 Start and Stop of DSPI Transfers

The DSPI has two operating states: STOPPED and RUNNING. The states are independent of DSPI configuration. The default state of the DSPI is STOPPED. In the STOPPED state no serial transfers are initiated in master mode and no transfers are responded to in slave mode. The STOPPED state is also a safe state for writing the various configuration registers of the DSPI without causing undetermined results. The TXRXS bit in the DSPI*x*_SR is cleared in this state. In the RUNNING state, serial transfers take place. The TXRXS bit in the DSPI*x*_SR is set in the RUNNING state.

Figure 10-12 shows a state diagram of the start and stop mechanism.

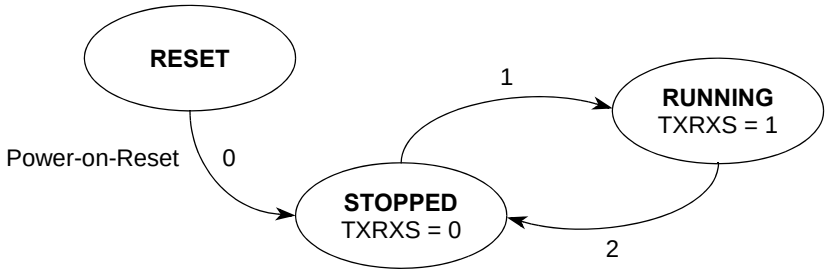



**Figure 10-12. DSPI Start and Stop State Diagram**

The transitions are described in Table 10-18.

**Table 10-18. State Transitions for Start and Stop of DSPI Transfers**

| Transition # | Current State | Next State | Description |
|---|---|---|---|
| 0 | RESET | STOPPED | Generic power-on-reset transition |
| 1 | STOPPED | RUNNING | The DSPI starts (transitions from STOPPED to RUNNING) when all of the following conditions are true:<br>• EOQF bit is clear<br>• Debug mode is unselected or the FRZ bit is clear<br>• HALT bit is clear |
| 2 | RUNNING | STOPPED | The DSPI stops (transitions from RUNNING to STOPPED) after the current frame for any one of the following conditions:<br>• EOQF bit is set<br>• Debug mode is selected and the FRZ bit is set<br>• HALT bit is set |

State transitions from RUNNING to STOPPED occur on the next frame boundary if a transfer is in progress, or on the next system clock cycle if no transfers are in progress.

### 10.9.3 Serial Peripheral Interface (SPI) Configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The DSPI is in SPI configuration when the DCONF field in the DSPI*x*_MCR is 0b00. The SPI frames can be from 4 to 16 bits long. The data to be transmitted can come from queues stored in RAM external to the DSPI. Host software or an eDMA controller can transfer the SPI data from the queues to a first-in first-out (FIFO) buffer. The received data is stored in entries in the receive FIFO (RX FIFO) buffer. Host software or an eDMA controller transfers the received data from the RX FIFO to memory external to the DSPI.

The FIFO buffer operations are described in Section 10.9.3.4, “Transmit First In First Out (TX FIFO) Buffering Mechanism, and Section 10.9.3.5, “Receive First In First Out (RX FIFO) Buffering Mechanism.

The interrupt and DMA request conditions are described in Section 10.9.7, “Interrupts/DMA Requests.

The SPI configuration supports two module-specific modes; master mode and slave mode. The FIFO operations are similar for the master mode and slave mode. The main difference is that in master mode the DSPI initiates and controls the transfer according to the fields in the SPI command field of the TX FIFO entry. In slave mode the DSPI only responds to transfers initiated by a bus master external to the DSPI and the SPI command field of the TX FIFO entry is ignored.

#### 10.9.3.1 SPI Master Mode

In SPI master mode the DSPI initiates the serial transfers by controlling the serial communications clock (SCK_*x*) and the peripheral chip select (CS*x*) signals. The SPI command field in the executing TX FIFO entry determines which CTARs are used to set the transfer attributes and which CS*x* signal to assert. The command field also contains various bits that help with queue management and transfer protocol. The data field in the executing TX FIFO entry is loaded into the shift register and shifted out on the serial out (SOUT_*x*) pin. In SPI master mode, each SPI frame to be transmitted has a command associated with it allowing for transfer attribute control on a frame by frame basis.

Refer to Section 10.8.2.6, “DSPI PUSH TX FIFO Register (DSPIx_PUSHR), for details on the SPI command fields.

#### 10.9.3.2 SPI Slave Mode

In SPI slave mode the DSPI responds to transfers initiated by an SPI bus master. The DSPI does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase and frame size must be set for successful communication with an SPI master. The SPI slave mode transfer attributes are set in the DSPI*x*_CTAR0.

#### 10.9.3.3 FIFO Disable Operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO or RX FIFO. The DSPI operates as a double-buffered simplified SPI when the FIFOs are disabled. The TX and RX FIFOs are

disabled separately. The TX FIFO is disabled by writing a 1 to the DIS_TXF bit in the DSPI*x*_MCR. The RX FIFO is disabled by writing a 1 to the DIS_RXF bit in the DSPI*x*_MCR.

The FIFO disable mechanisms are transparent to the user and to host software; transmit data and commands are written to the DSPI*x*_PUSHR and received data is read from the DSPI*x*_POPR. When the TX FIFO is disabled, the TFFF, TFUF, and TXCTR fields in DSPI*x*_SR behave as if there is a one-entry FIFO but the contents of the DSPI*x*_TXFRs and TXNXTPTR are undefined. When the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the DSPI*x*_SR behave as if there is a one-entry FIFO but the contents of the DSPI*x*_RXFRs and POPNXTPTR are undefined.

Disable the TX and RX FIFOs only if the FIFO must be disabled as a requirement of the application's operating mode. A FIFO must be disabled before it is accessed. Failure to disable a FIFO prior to a first FIFO access is not supported, and can result in incorrect results.

#### 10.9.3.4 Transmit First In First Out (TX FIFO) Buffering Mechanism

The TX FIFO functions as a buffer of SPI data and SPI commands for transmission. The TX FIFO holds five entries, each consisting of a command field and a data field. SPI commands and data are added to the TX FIFO by writing to the DSPI push TX FIFO register (DSPI*x*_PUSHR). For more information on DSPI*x*_PUSHR. TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

Refer to Section 10.8.2.6, “DSPI PUSH TX FIFO Register (DSPIx_PUSHR).

The TX FIFO counter field (TXCTR) in the DSPI status register (DSPI*x*_SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time the DSPI _PUSHR is written or SPI data is transferred into the shift register from the TX FIFO.

Refer to Section 10.8.2.4, “DSPI Status Register (DSPIx_SR) for more information on DSPI*x*_SR.

The TXNXTPTR field indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR contains the positive offset from DSPI*x*_TXFR0 in number of 32-bit registers. For example, TXNXTPTR equal to two means that the DSPI*x*_TXFR2 contains the SPI data and command for the next transfer. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register.

##### 10.9.3.4.1 Filling the TX FIFO

Host software or the eDMA controller can add (push) entries to the TX FIFO by writing to the DSPI*x*_PUSHR. When the TX FIFO is not full, the TX FIFO fill flag (TFFF) in the DSPI*x*_SR is set. The TFFF bit is cleared when the TX FIFO is full and the eDMA controller indicates that a write to DSPI*x*_PUSHR is complete or alternatively by host software writing a 1 to the TFFF in the DSPI*x*_SR. The TFFF can generate a DMA request or an interrupt request.

Refer to Section 10.9.7.2, “Transmit FIFO Fill Interrupt or DMA Request (TFFF), for details.

The DSPI ignores attempts to push data to a full TX FIFO; that is, the state of the TX FIFO is unchanged. No error condition is indicated.

#### 10.9.3.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO counter is decremented by one. At the end of a transfer, the TCF bit in the DSPI*x*_SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a 1 to the CLR_TXF bit in DSPI*x*_MCR.

If an external SPI bus master initiates a transfer with a DSPI slave while the slave's DSPI TX FIFO is empty, the transmit FIFO underflow flag (TFUF) in the slave's DSPI*x*_SR is set.

Refer to Section 10.9.7.4, "Transmit FIFO Underflow Interrupt Request (TFUF), for details.

### 10.9.3.5 Receive First In First Out (RX FIFO) Buffering Mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds four received SPI data frames. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data is removed (popped) from the RX FIFO by reading the DSPI*x*_POPR register. RX FIFO entries can only be removed from the RX FIFO by reading the DSPI*x*_POPR or by flushing the RX FIFO.

Refer to Section 10.8.2.7, "DSPI POP RX FIFO Register (DSPIx_POPR) for more information on the DSPI*x*_POPR.

The RX FIFO counter field (RXCTR) in the DSPI status register (DSPI*x*_SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the DSPI _POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the DSPI*x*_SR points to the RX FIFO entry that is returned when the DSPI*x*_POPR is read. The POPNXTPTR contains the positive, 32-bit word offset from DSPI*x*_RXFR0. For example, POPNXTPTR equal to two means that the DSPI*x*_RXFR2 contains the received SPI data that is returned when DSPI*x*_POPR is read. The POPNXTPTR field is incremented every time the DSPI*x*_POPR is read. POPNXTPTR rolls over every four frames on the MCU.

#### 10.9.3.5.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO the RX FIFO counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the DSPI*x*_SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the DSPI*x*_MCR, the data from the transfer that generated the overflow is ignored or put in the shift register. If the ROOE bit is set, the incoming data is put in the shift register. If the ROOE bit is cleared, the incoming data is ignored.

#### 10.9.3.5.2 Draining the RX FIFO

Host software or the eDMA can remove (pop) entries from the RX FIFO by reading the DSPI*x*_POPR. A read of the DSPI*x*_POPR decrements the RX FIFO counter by one. Attempts to pop data from an empty

RX FIFO are ignored, the RX FIFO counter remains unchanged. The data returned from reading an empty RX FIFO is undetermined.

Refer to Section 10.8.2.7, “DSPI POP RX FIFO Register (DSPIx_POPR) for more information on DSPI*x*_POPR.

When the RX FIFO is not empty, the RX FIFO drain flag (RFDF) in the DSPI*x*_SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the eDMA controller indicates that a read from DSPI*x*_POPR is complete; alternatively the RFDF bit can be cleared by the host writing a 1 to it.

## 10.9.4 DSPI Baud Rate and Clock Delay Generation

The SCK_*x* frequency and the delay values for serial transfer are generated by dividing the system clock frequency by a prescaler and a scaler with the option of doubling the baud rate.

Figure 10-13 shows conceptually how the SCK signal is generated.

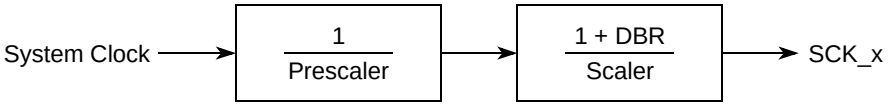



**Figure 10-13. Communications Clock Prescalers and Scalers**

### 10.9.4.1 Baud Rate Generator

The baud rate is the frequency of the serial communication clock (SCK_*x*). The system clock is divided by a baud rate prescaler (defined by DSPI*x*_CTAR[PBR]) and baud rate scaler (defined by DSPI*x*_CTAR[BR]) to produce SCK_*x* with the possibility of doubling the baud rate. The DBR, PBR, and BR fields in the DSPI*x*_CTARs select the frequency of SCK_*x* using the following formula:

$$\text{SCK baud rate} = \frac{f_{SYS}}{\text{PBRPrescalerValue}} \times \frac{1 + \text{DBR}}{\text{BRScalerValue}}$$

Table 10-19 shows an example of a computed baud rate.

**Table 10-19. Baud Rate Computation Example**

| $f_{SYS}$ | PBR | Prescaler Value | BR | Scaler Value | DBR Value | Baud Rate |
|---|---|---|---|---|---|---|
| 100 MHz | 0b00 | 2 | 0b0000 | 2 | 0 | 25 Mb/sec |
| 20 MHz | 0b00 | 2 | 0b0000 | 2 | 1 | 10 Mb/sec |

### 10.9.4.2 CS to SCK Delay ($t_{CSC}$)

The CS_*x* to SCK_*x* delay is the length of time from assertion of the CS_*x* signal to the first SCK_*x* edge. Refer to Figure 10-14 for an illustration of the CS_*x* to SCK_*x* delay. The PCSSCK and CSSCK fields in the DSPI*x*_CTAR*n* registers select the CS_*x* to SCK_*x* delay, and the relationship is expressed by the following formula:

$$t_{CSC} = \frac{1}{f_{SYS}} \times PCSSCK \times CSSCK$$

Table 10-20 shows an example of the computed CS to SCK_*x* delay.

**Table 10-20. CS to SCK Delay Computation Example**

| PCSSCK | Prescaler Value | CSSCK | Scaler Value | $f_{SYS}$ | CS to SCK Delay |
|---|---|---|---|---|---|
| 0b01 | 3 | 0b0100 | 32 | 100 MHz | 0.96 µs |

### 10.9.4.3 After SCK Delay ($t_{ASC}$)

The after SCK_*x* delay is the length of time between the last edge of SCK_*x* and the negation of CS_*x*. Refer to Figure 10-14 and Figure 10-15 for illustrations of the after SCK_*x* delay. The PASC and ASC fields in the DSPI*x*_CTAR*n* registers select the after SCK delay. The relationship between these variables is given in the following formula:

$$t_{ASC} = \frac{1}{f_{SYS}} \times PASC \times ASC$$

Table 10-21 shows an example of the computed after SCK delay.

**Table 10-21. After SCK Delay Computation Example**

| PASC | Prescaler Value | ASC | Scaler Value | $f_{SYS}$ | After SCK Delay |
|---|---|---|---|---|---|
| 0b01 | 3 | 0b0100 | 32 | 100 MHz | 0.96 µs |

### 10.9.4.4 Delay after Transfer ($t_{DT}$)

The delay after transfer is the length of time between negation of the CS*x* signal for a frame and the assertion of the CS*x* signal for the next frame. The PDT and DT fields in the DSPI*x*_CTAR*n* registers select the delay after transfer.

Refer to Figure 10-14 for an illustration of the delay after transfer.

The following formula expresses the PDT/DT/delay after transfer relationship:

$$t_{DT} = \frac{1}{f_{SYS}} \times PDT \times DT$$

Table 10-22 shows an example of the computed delay after transfer.

**Table 10-22. Delay after Transfer Computation Example**

| PDT | Prescaler Value | DT | Scaler Value | $f_{SYS}$ | Delay after Transfer |
|---|---|---|---|---|---|
| 0b01 | 3 | 0b1110 | 32768 | 100 MHz | 0.98 ms |

When in non-continuous clock mode the $t_{DT}$ delay is configurable as outlined in the DSPI_CTARx registers. When in continuous clock mode and TSB is not enabled the delay is fixed at 1 SCK period. When in TSB and continuous mode the delay is programmed as outlined in the DSPI_CTARx registers but in the event that the delay does not coincide with an SCK period in duration the delay is extended to the next SCK active edge. Table 10-23 shows an example of how to compute the Delay after Transfer with the clock period of SCK defined as $T_{SCK}$. The values calculated assume 1 TSCK period = 4 ipg_clk.

**Table 10-23. Delay after Transfer Computation Example in TSB Configuration**

| | | PDT field | | | |
|---|---|---|---|---|---|
| | $t_{DT}$[1] (Tsck) | 0 | 1 | 2 | 3 |
| DT field | 0[2] | 1 | 2 | 3 | 4 |
| | 1 | 1 | 3 | 5 | 7 |
| | 2 | 2 | 6 | 10 | 14 |
| | 3 | 4 | 12 | 20 | 28 |
| | 4 | 8 | 24 | 40 | 56 |
| | 5 | 16 | 48 | 80 | 112 |
| | 6 | 32 | 96 | 160 | 224 |
| | 7 | 64 | 192 | 320 | 448 |
| | 8 | 128 | 384 | 640 | 896 |
| | 9 | 256 | 768 | 1280 | 1792 |
| | 10 | 512 | 1536 | 2560 | 3584 |
| | 11 | 1024 | 3072 | 5120 | 7168 |
| | 12 | 2048 | 6144 | 10240 | 14336 |
| | 13 | 4096 | 12288 | 20480 | 28672 |
| | 14 | 8192 | 24576 | 40960 | 57344 |
| | 15 | 16384 | 49152 | 81920 | 114688 |

1 Some values are not reachable (i. e. 9, 11, 13, 15, 17, 18, 19...). To calculate these values, please see Equation 10-3.

2 The values in this row were rounded to the next integer value.

## 10.9.5 Transfer Formats

The SPI serial communication is controlled by the serial communications clock (SCK_*x*) signal and the CS*x* signals. The SCK_*x* signal provided by the master device synchronizes shifting and sampling of the data by the SIN_*x* and SOUT_*x* pins. The CS*x* signals serve as enable signals for the slave devices.

When the DSPI is the bus master, the CPOL and CPHA bits in the DSPI clock and transfer attributes registers (DSPI*x*_CTAR*n*) select the polarity and phase of the serial clock, SCK_*x*. The polarity bit selects the idle state of the SCK_*x*. The clock phase bit selects if the data on SOUT_*x* is valid before or on the first SCK_*x* edge.

When the DSPI is the bus slave, CPOL and CPHA bits in the DSPI*x*_CTAR0 (SPI slave mode) select the polarity and phase of the serial clock. Even though the bus slave does not control the SCK signal, clock polarity, clock phase and number of bits to transfer must be identical for the master device and the slave device to ensure proper transmission.

The DSPI supports four different transfer formats:

- Classic SPI with CPHA = 0
- Classic SPI with CPHA = 1
- Modified transfer format with CPHA = 0
- Modified transfer format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The DSPI can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the DSPI*x*_MCR selects between classic SPI format and modified transfer format. The classic SPI formats are described in Section 10.9.5.1, “Classic SPI Transfer Format (CPHA = 0) and Section 10.9.5.2, “Classic SPI Transfer Format (CPHA = 1). The modified transfer formats are described in Section 10.9.5.3, “Modified SPI Transfer Format (MTFE = 1, CPHA = 0) and Section 10.9.5.4, “Modified SPI Transfer Format (MTFE = 1, CPHA = 1).

In the SPI configuration, the DSPI provides the option of keeping the CS signals asserted between frames. Refer to Section 10.9.5.5, “Continuous Selection Format for details.

### 10.9.5.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in Figure 10-14 is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN_*x* pins on the odd-numbered SCK_*x* edges and change the data on their SOUT_*x* pins on the even-numbered SCK_*x* edges.

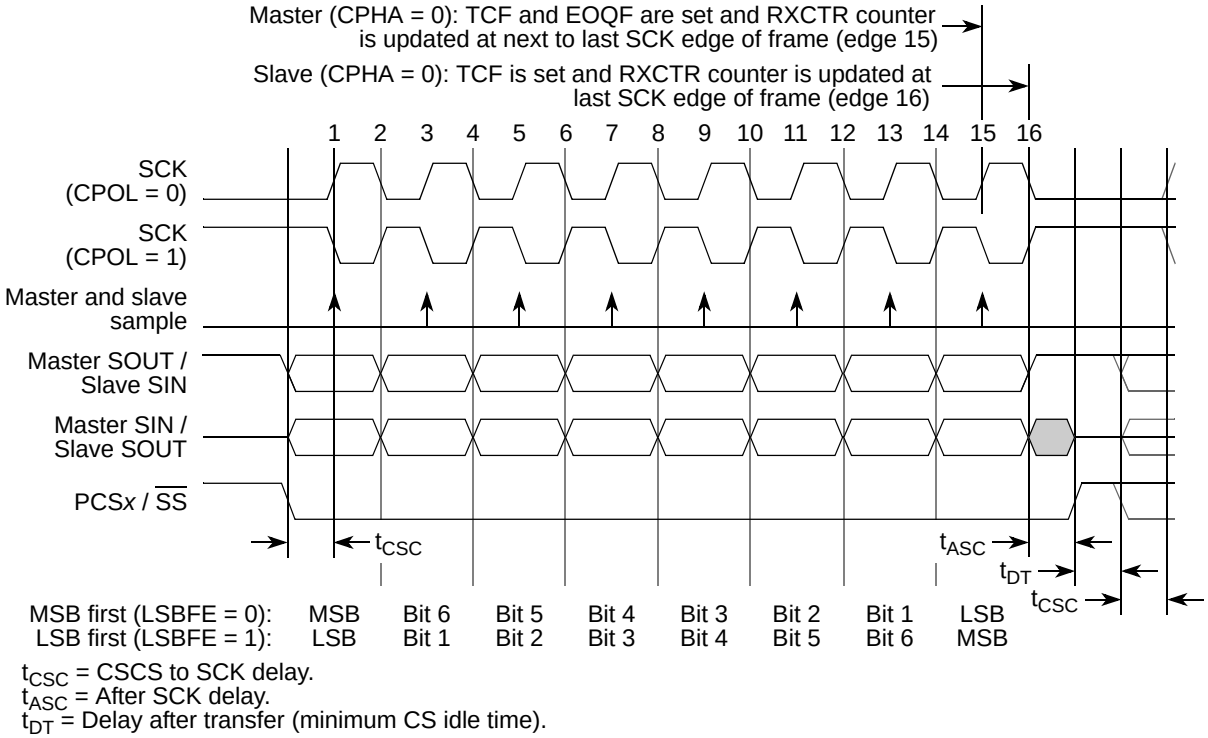



**Figure 10-14. DSPI Transfer Timing Diagram (MTFE = 0, CPHA = 0, FMSZ = 8)**

The master initiates the transfer by placing its first data bit on the SOUT_*x* pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT_*x* pin. After the $t_{CSC}$ delay has elapsed, the master outputs the first edge of SCK_*x*. This is the edge used by the master and slave devices to sample the first input data bit on their serial data input signals. At the second edge of the SCK_*x* the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN_*x* pins on the odd-numbered clock edges and changes the data on their SOUT_*x* pins on the even-numbered clock edges. After the last clock edge occurs a delay of $t_{ASC}$ is inserted before the master negates the CS signals. A delay of $t_{DT}$ is inserted before a new frame transfer can be initiated by the master.

For the CPHA = 0 condition of the master, TCF and EOQF are set and the RXCTR counter is updated at the next to last serial clock edge of the frame (edge 15) of Figure 10-14.

For the CPHA = 0 condition of the slave, TCF is set and the RXCTR counter is updated at the last serial clock edge of the frame (edge 16) of Figure 10-14.

### 10.9.5.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in Figure 10-15 is used to communicate with peripheral SPI slave devices that require the first SCK_*x* edge before the first data bit becomes available on the slave SOUT_*x* pin. In this format the master and slave devices change the data on their SOUT_*x* pins on the odd-numbered SCK_*x* edges and sample the data on their SIN_*x* pins on the even-numbered SCK_*x* edges.

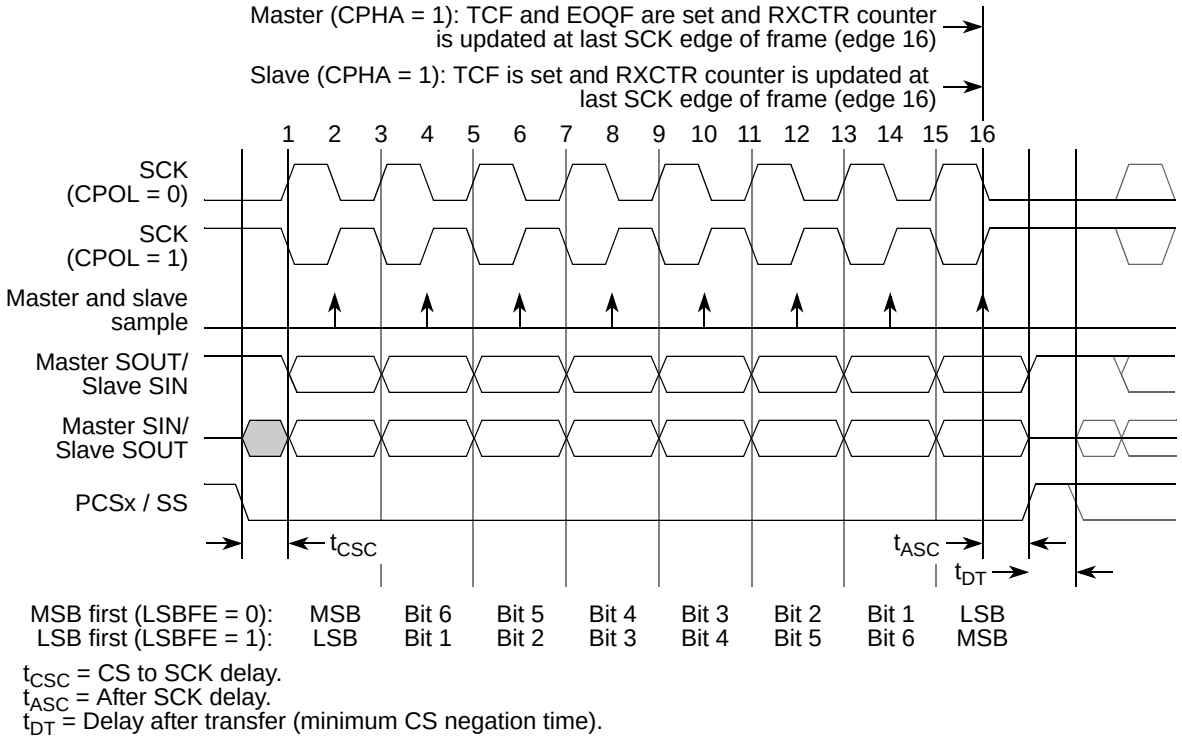



**Figure 10-15. DSPI Transfer Timing Diagram (MTFE = 0, CPHA = 1, FMSZ = 8)**

The master initiates the transfer by asserting the CS*x* signal to the slave. After the $t_{CSC}$ delay has elapsed, the master generates the first SCK_*x* edge and at the same time places valid data on the master SOUT_*x* pin. The slave responds to the first SCK_*x* edge by placing its first data bit on its slave SOUT_*x* pin.

At the second edge of the SCK_*x* the master and slave sample their SIN_*x* pins. For the rest of the frame the master and the slave change the data on their SOUT_*x* pins on the odd-numbered clock edges and sample their SIN_*x* pins on the even-numbered clock edges. After the last clock edge occurs a delay of $t_{ASC}$ is inserted before the master negates the CS*x* signal. A delay of $t_{DT}$ is inserted before a new frame transfer can be initiated by the master.

For CPHA = 1 the master EOQF and TCF and slave TCF are set at the last serial clock edge (edge 16) of Figure 10-15. For CPHA = 1 the master and slave RXCTR counters are updated on the same clock edge.

### 10.9.5.3 Modified SPI Transfer Format (MTFE = 1, CPHA = 0)

In this modified transfer format both the master and the slave sample later in the SCK period than in classic SPI mode to allow for delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

**NOTE**

For the modified transfer format to operate correctly, you must thoroughly analyze the SPI link timing budget.

The master and the slave place data on the SOUT_*x* pins at the assertion of the CS*x* signal. After the CS*x* to SCK_*x* delay has elapsed the first SCK_*x* edge is generated. The slave samples the master SOUT_*x*

signal on every odd numbered SCK_*x* edge. The slave also places new data on the slave SOUT_*x* on every odd numbered clock edge.

The master places its second data bit on the SOUT_*x* line one system clock after odd numbered SCK_*x* edge. The point where the master samples the slave SOUT_*x* is selected by writing to the SMPL_PT field in the DSPI*x*_MCR. Table 10-24 lists the number of system clock cycles between the active edge of SCK_*x* and the master sample point for different values of the SMPL_PT bit field. The master sample point can be delayed by one or two system clock cycles.

**Table 10-24. Delayed Master Sample Point**

| SMPL_PT | Number of System Clock Cycles between Odd-numbered Edge of SCK and Sampling of SIN |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 10 | 2 |
| 11 | Invalid value |

Figure 10-16 shows the modified transfer format for CPHA = 0. Only the condition where CPOL = 0 is illustrated. The delayed master sample points are indicated with a lighter shaded arrow.

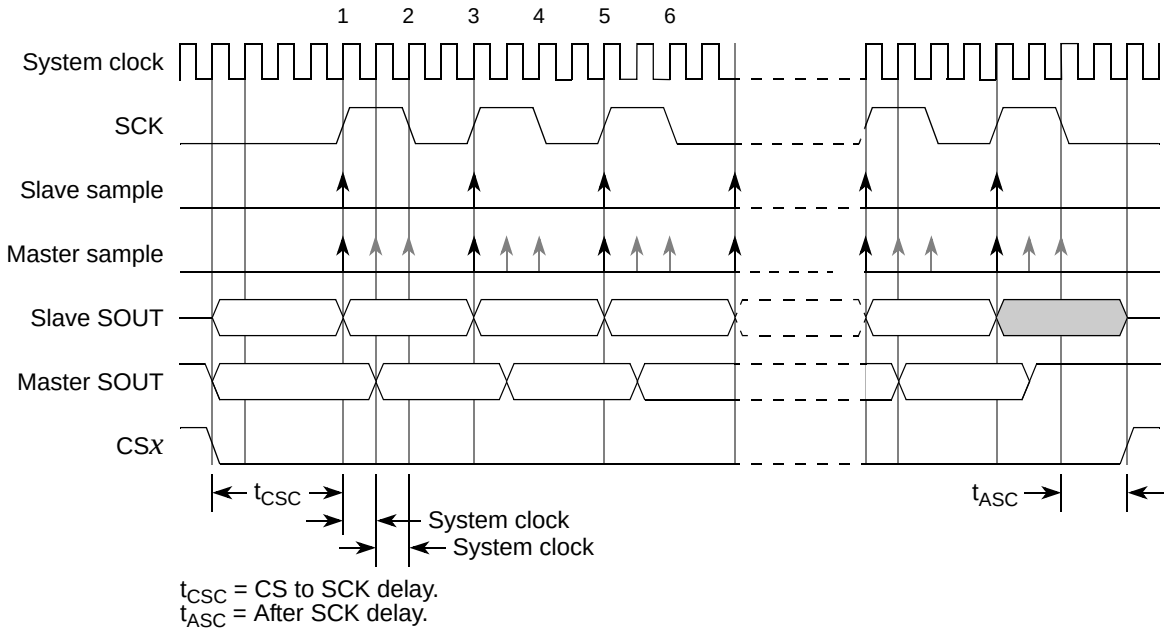



**Figure 10-16. DSPI Modified Transfer Format (MTFE = 1, CPHA = 0, $f_{SCK} = f_{SYS} / 4$)**

### 10.9.5.4 Modified SPI Transfer Format (MTFE = 1, CPHA = 1)

At the start of a transfer the DSPI asserts the CS signal to the slave device. After the CS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the

last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge is visible on the master SCK pin during the sampling of the last bit. The SCK to CS delay must be greater or equal to half of the SCK period.

**NOTE**

For the modified transfer format to operate correctly, analyze the SPI link timing budget thoroughly.

Figure 10-17 shows the modified transfer format for CPHA = 1. Only the condition where CPOL = 0 is described.

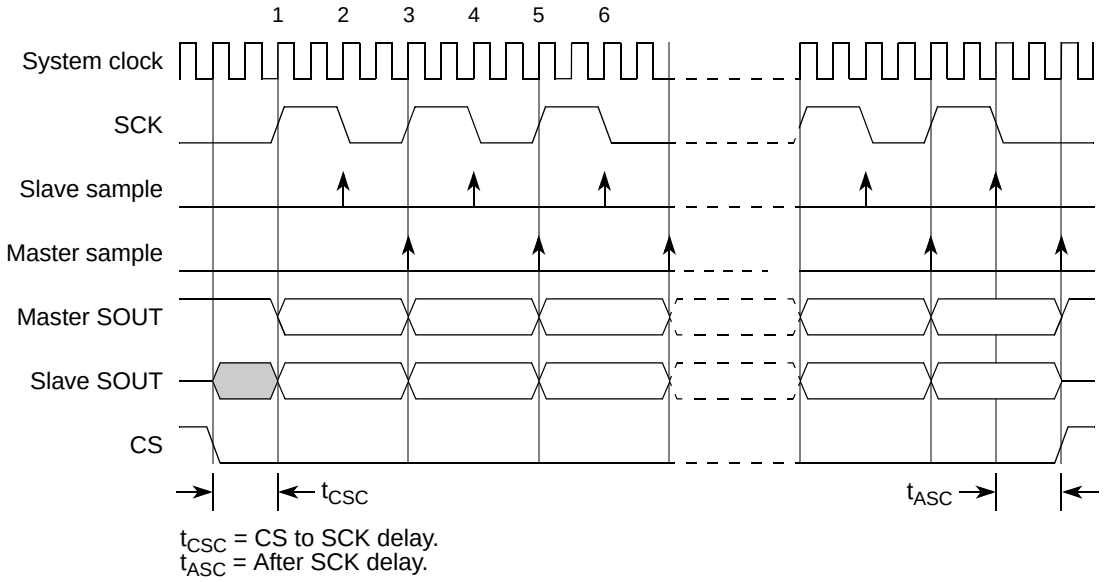



**Figure 10-17. DSPI Modified Transfer Format (MTFE = 1, CPHA = 1, $f_{SCK} = f_{SYS} / 4$)**

### 10.9.5.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The continuous selection format provides the flexibility to handle both cases. The continuous selection format is enabled for the SPI configuration by setting the CONT bit in the SPI command.

When the CONT bit = 0, the DSPI drives the asserted chip select signals to their idle states in between frames. The idle states of the chip select signals are selected by the PCSIS field in the DSPI*x*_MCR.

Figure 10-18 shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 0.

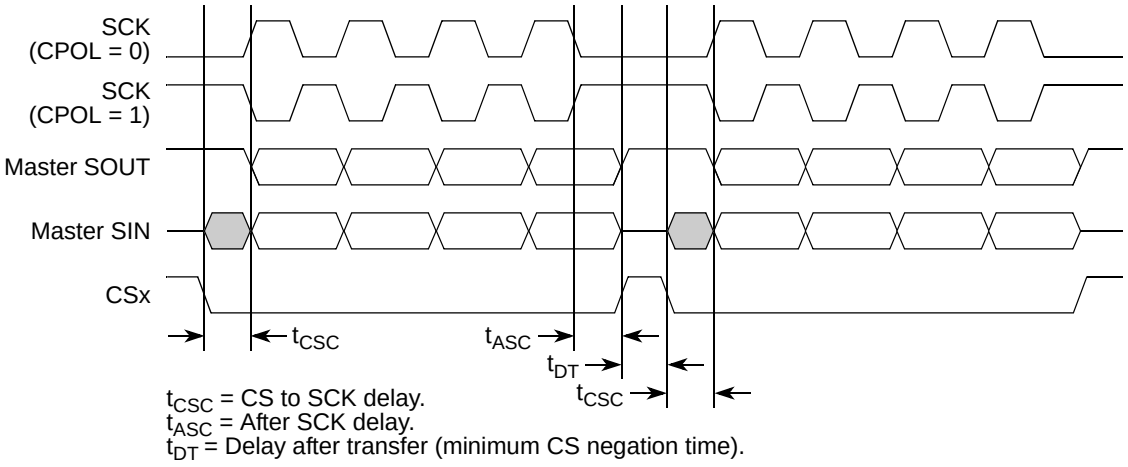



**Figure 10-18. Example of Non-Continuous Format (CPHA = 1, CONT = 0)**

When the CONT = 1 and the CS signal for the next transfer is the same as for the current transfer, the CS signal remains asserted for the duration of the two transfers. The delay between transfers ($t_{DT}$) is not inserted between the transfers.

Figure 10-19 shows the timing diagram for two 4-bit transfers with CPHA = 1 and CONT = 1.

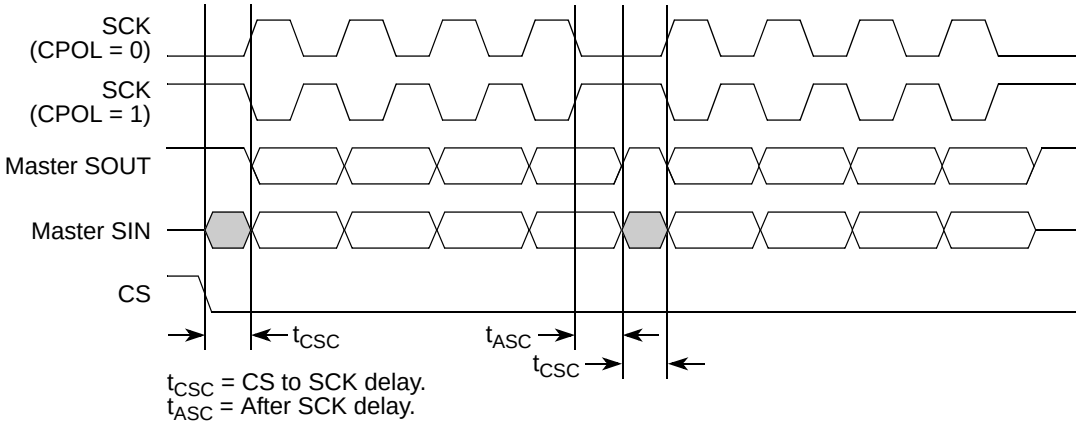



**Figure 10-19. Example of Continuous Transfer (CPHA = 1, CONT = 1)**

In Figure 10-19, the period length at the start of the next transfer is the sum of $t_{ASC}$ and $t_{CSC}$; i.e., it does not include a half-clock period. The default settings for these provide a total of four system clocks. In many situations, $t_{ASC}$ and $t_{CSC}$ must be increased if a full half-clock period is required.

When the CONT bit = 1 and the CS signals for the next transfer are different from the present transfer, the CS signals behave as if the CONT bit was not set.

Switching CTAR registers or changing which PCS signals are asserted between frames while using Continuous Selection can cause errors in the transfer. The PCS signal should be negated before CTAR is switched or different PCS signals are selected.

It is mandatory to fill the TXFIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TXFIFO becomes empty. For example, while

transmitting in master mode, it should be ensured that the last entry in the TXFIFO, after which TXFIFO becomes empty, must have the CONT bit in the command frame as deasserted (CONT bit = 0). While operating in slave mode, it should be ensured that when the last-entry in the TXFIFO is completely transmitted (i.e. the corresponding TCF flag is asserted and TXFIFO is empty) the slave is de-selected for any further serial communication; otherwise, an underflow error occurs.

#### 10.9.5.6 Clock Polarity Switching between DSPI Transfers

If it is desired to switch polarity between non-continuous DSPI frames, the edge generated by the change in the idle state of the clock occurs one system clock before the assertion of the chip select for the next frame.

Refer to Section 10.8.2.3, “DSPI Clock and Transfer Attributes Registers 0–7 (DSPIx_CTARn).

In Figure 10-20, time ‘A’ shows the one clock interval. Time ‘B’ is user programmable from a minimum of two system clocks.

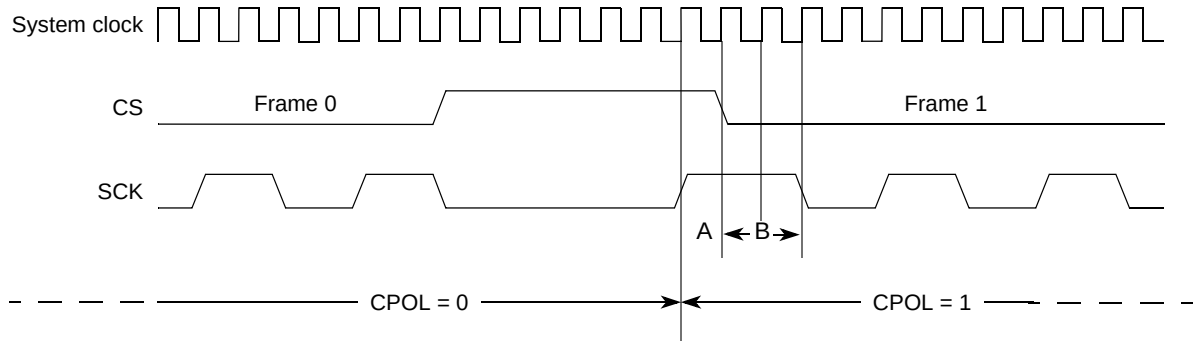



**Figure 10-20. Polarity Switching between Frames**

### 10.9.6 Continuous Serial Communications Clock

The DSPI provides the option of generating a continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the DSPI*x*_MCR. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA = 1. Setting CPHA = 0 is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for modified transfer format.

Clock and transfer attributes for the continuous SCK mode are set according to the following rules:

- When the DSPI is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the continuous SCK mode is terminated.

The device is designed to use the same baud rate for all transfers made while using the continuous SCK. Switching clock polarity between frames while using continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the DSPI is put into module disable mode.

Enabling continuous SCK disables the CS to SCK delay and the After SCK delay. The delay after transfer is fixed at one SCK cycle. Figure 10-21 shows timing diagram for continuous SCK format with continuous selection disabled.

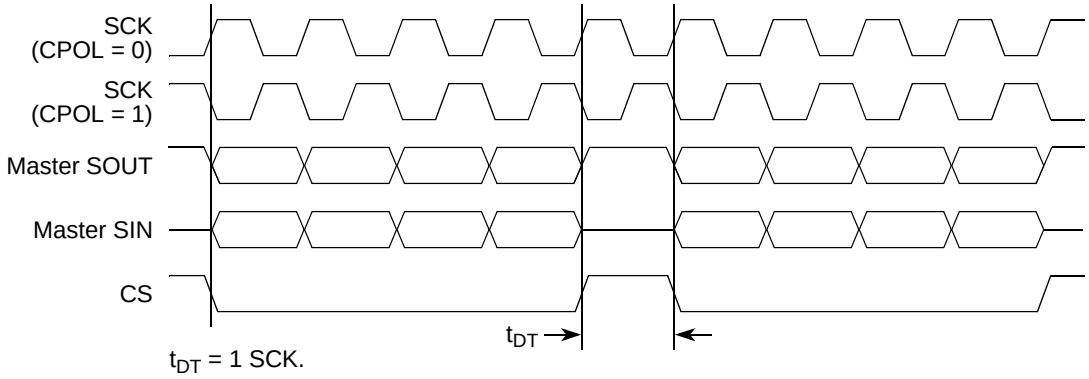



**Figure 10-21. Continuous SCK Timing Diagram (CONT= 0)**

If the CONT bit in the TX FIFO entry is set, CS remains asserted between the transfers when the CS signal for the next transfer is the same as for the current transfer. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT (SOUT pulled high). This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the transmit FIFO.
- Continuous SCK with CONT bit set and entering STOPPED state (refer to Section 10.9.2, “Start and Stop of DSPI Transfers”).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

Figure 10-22 shows timing diagram for continuous SCK format with continuous selection enabled.

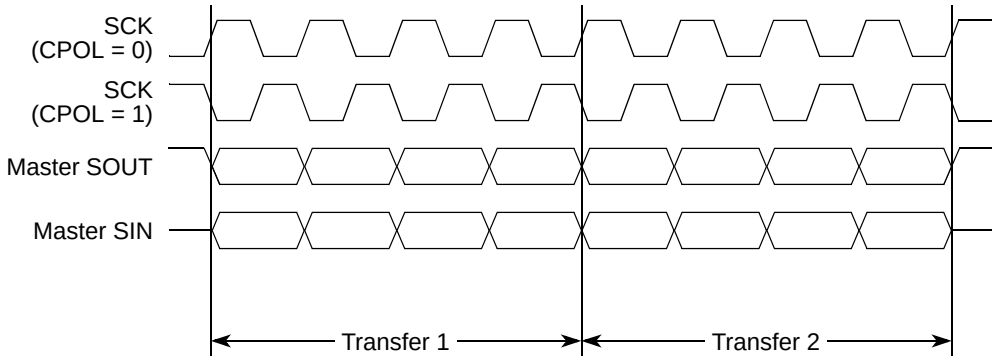



**Figure 10-22. Continuous SCK Timing Diagram (CONT=1)**

## 10.9.7 Interrupts/DMA Requests

The DSPI has five conditions that can generate interrupt requests only, and two conditions that can generate interrupt.

Table 10-25 lists the six conditions.

**Table 10-25. Interrupt and DMA Request Conditions**

| Condition | Flag | Interrupt | DMA |
|---|---|---|---|
| End of transfer queue has been reached (EOQ) | EOQF | X | |
| TX FIFO is not full | TFFF | X | X |
| Current frame transfer is complete | TCF | X | |
| TX FIFO underflow has occurred | TFUF | X | |
| RX FIFO is not empty | RFDF | X | X |
| RX FIFO overflow occurred | RFOF | X | |
| A FIFO overrun occurred[1] | TFUF ORed with RFOF | X | |

[1] The FIFO overrun condition is created by ORing the TFUF and RFOF flags together.

Each condition has a flag bit and a request enable bit. The flag bits are described in the Section 10.8.2.4, “DSPI Status Register (DSPIx_SR) and the request enable bits are described in the Section 10.8.2.5, “DSPI DMA / Interrupt Request Select and Enable Register (DSPIx_RSER). The TX FIFO fill flag (TFFF) and RX FIFO drain flag (RFDF) generate interrupt requests or DMA requests depending on the TFFF_DIRS and RFDF_DIRS bits in the DSPI*x*_RSER. Refer to Chapter 10, “Deserial Serial Peripheral Interface (DSPI) and Chapter 16, “eDMA Channel Mux (DMACHMUX) for more information on interrupt and DMA channel assignments.

### 10.9.7.1 End of Queue Interrupt Request (EOQF)

The end of queue equest indicates that the end of a transmit queue is reached. The end of queue request is generated when the EOQ bit in the executing SPI command is asserted and the EOQF_RE bit in the DSPI*x*_RSER is set. Refer to the EOQ bit description in Section 10.8.2.4, “DSPI Status Register (DSPIx_SR). Refer to Figure 10-14 and Figure 10-15 that illustrate when EOQF is set.

### 10.9.7.2 Transmit FIFO Fill Interrupt or DMA Request (TFFF)

The transmit FIFO fill request indicates that the TX FIFO is not full. The transmit FIFO fill request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the DSPI*x*_RSER is set. The TFFF_DIRS bit in the DSPI*x*_RSER selects whether a DMA request or an interrupt request is generated.

### 10.9.7.3 Transfer Complete Interrupt Request (TCF)

The transfer complete request indicates the end of the transfer of a serial frame. The transfer complete request is generated at the end of each frame transfer when the TCF_RE bit is set in the DSPI*x*_RSER. Refer to the TCF bit description in Section 10.8.2.4, “DSPI Status Register (DSPIx_SR). Refer to Figure 10-14 and Figure 10-15 that illustrate when TCF is set.

#### 10.9.7.4 Transmit FIFO Underflow Interrupt Request (TFUF)

The transmit FIFO underflow request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for DSPI modules operating in slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI operating in slave mode and SPI configuration is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the DSPI*x*_RSER is set, an interrupt request is generated.

#### 10.9.7.5 Receive FIFO Drain Interrupt or DMA Request (RFDF)

The receive FIFO drain request indicates that the RX FIFO is not empty. The receive FIFO drain request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the DSPI*x*_RSER is set. The RFDF_DIRS bit in the DSPI*x*_RSER selects whether a DMA request or an interrupt request is generated.

#### 10.9.7.6 Receive FIFO Overflow Interrupt Request (RFOF)

The receive FIFO overflow request indicates that an overflow condition in the RX FIFO has occurred. A receive FIFO overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the DSPI*x*_RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the DSPI*x*_MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is negated, the incoming data is ignored.

#### 10.9.7.7 FIFO Overrun Request (TFUF) or (RFOF)

The FIFO overrun request indicates that at least one of the FIFOs in the DSPI has exceeded its capacity. The FIFO overrun request is generated by logically OR'ing together the RX FIFO overflow and TX FIFO underflow signals.

### 10.9.8 Power Saving Features

The DSPI supports three power-saving strategies:

- External stop mode
- Module disable mode—clock gating of non-memory mapped logic
- Clock gating of slave interface signals and clock to memory-mapped logic

#### 10.9.8.1 External Stop Mode

The DSPI supports the Stop Mode protocol. When a request is made to enter External Stop Mode, the DSPI block acknowledges the request by negating ipg_stop_ack. When the DSPI is ready to have its clocks shut off the ipg_stop_ack signal is asserted. If a serial transfer is in progress, the DSPI waits until it reaches the frame boundary before it asserts ipg_stop_ack. While the clocks are shut off, the DSPI memory-mapped logic is not accessible. The states of the interrupt and DMA request signals cannot be changed while in External Stop Mode. Implementation of IPI Green Line Stop Mode in an SoC is optional.

#### 10.9.8.2 Module Disable Mode

Module disable mode is a module-specific mode that the DSPI can enter to save power. Host software can initiate the module disable mode by writing a 1 to the MDIS bit in the DSPI*x*_MCR. In module disable mode, the DSPI is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different affect when the DSPI is in the module disable mode. Reading the RX FIFO pop register does not change the state of the RX FIFO. Likewise, writing to the TX FIFO push register does not change the state of the TX FIFO. Clearing either of the FIFOs does not have any effect in the module disable mode. Changes to the DIS_TXF and DIS_RXF fields of the DSPI*x*_MCR does not have any affect in the module disable mode. In the module disable mode, all status bits and register flags in the DSPI return the correct values when read, but writing to them has no affect. Writing to the DSPI*x*_TCR during module disable mode does not have an effect. Interrupt and DMA request signals cannot be cleared while in the module disable mode.

#### 10.9.8.3 Slave Interface Signal Gating

The DSPI module enable signal is used to gate slave interface signals such as address, byte enable, read/write and data. This prevents toggling slave interface signals from consuming power unless the DSPI is accessed.

## 10.10 Initialization and Application Information

### 10.10.1 How to Change Queues

DSPI queues are not part of the DSPI module, but the DSPI includes features in support of queue management. Queues are primarily supported in SPI configuration. This section presents an example of how to change queues for the DSPI.

1. The last command word from a queue is executed. The EOQ bit in the command word is set to indicate to the DSPI that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the DSPI*x*_SR is set.
3. The setting of the EOQF flag disables both serial transmission, and serial reception of data, putting the DSPI in the STOPPED state. The TXRXS bit is negated to indicate the STOPPED state.
4. The eDMA continues to fill TX FIFO until it is full or step 5 occurs.
5. Disable DSPI DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the eDMA controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in DSPI*x*_SR or by checking RFDF in the DSPI*x*_SR after each read operation of the DSPI*x*_POPR.
7. Modify DMA descriptor of TX and RX channels for “new” queues.
8. Flush TX FIFO by writing a 1 to the CLR_TXF bit in the DSPI*x*_MCR register and flush the RX FIFO by writing a 1 to the CLR_RXF bit in the DSPI*x*_MCR register.

9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the DSPI*x*_TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the DSPI TX FIFO, and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

## 10.10.2 Baud Rate Settings

Table 10-26 shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the DSPI*x*_CTARs. The values are calculated at a 64 MHz system frequency with DBR = 0.

**Table 10-26. Baud Rate Values**

| | | Baud Rate Divider Prescaler Values (DSPI_CTAR[PBR]) | | | |
|---|---|---|---|---|---|
| | | **2** | **3** | **5** | **7** |
| **Baud Rate Scaler Values (DSPI_CTAR[BR])** | **2** | 16.0 MHz | 10.7 MHz | 6.4 MHz | 4.57 MHz |
| | **4** | 8 MHz | 5.33 MHz | 3.2 MHz | 2.28 MHz |
| | **6** | 5.33 MHz | 3.56 MHz | 2.13 MHz | 1.52 MHz |
| | **8** | 4 MHz | 2.67 MHz | 1.60 MHz | 1.15 MHz |
| | **16** | 2 MHz | 1.33 MHz | 800 kHz | 571 kHz |
| | **32** | 1 MHz | 670 kHz | 400 kHz | 285 kHz |
| | **64** | 500 kHz | 333 kHz | 200 kHz | 142 kHz |
| | **128** | 250 kHz | 166 kHz | 100 kHz | 71.7 kHz |
| | **256** | 125 kHz | 83.2 kHz | 50 kHz | 35.71 kHz |
| | **512** | 62.5 kHz | 41.6 kHz | 25 kHz | 17.86 kHz |
| | **1024** | 31.2 kHz | 20.8 kHz | 12.5 kHz | 8.96 kHz |
| | **2048** | 15.6 kHz | 10.4 kHz | 6.25 kHz | 4.47 kHz |
| | **4096** | 7.81 kHz | 5.21 kHz | 3.12 kHz | 2.23 kHz |
| | **8192** | 3.90 kHz | 2.60 kHz | 1.56 kHz | 1.11 kHz |
| | **16384** | 1.95 kHz | 1.31 kHz | 781 Hz | 558 Hz |
| | **32768** | 979 Hz | 653 Hz | 390 Hz | 279 Hz |

## 10.10.3 Delay Settings

Table 10-27 shows the values for the delay after transfer ($t_{DT}$) and CS to SCK delay ($t_{CSC}$) that can be generated based on the prescaler values and the scaler values set in the DSPI*x*_CTARs. The values calculated assume a 100 MHz system frequency.

**Table 10-27. Delay Values**

| | | Delay Prescaler Values (DSPI_CTAR[PBR]) | | | |
|---|---|---|---|---|---|
| | | 1 | 3 | 5 | 7 |
| Delay Scaler Values (DSPI_CTAR[DT]) | 2 | 20.0 ns | 60.0 ns | 100.0 ns | 140.0 ns |
| | 4 | 40.0 ns | 120.0 ns | 200.0 ns | 280.0 ns |
| | 8 | 80.0 ns | 240.0 ns | 400.0 ns | 560.0 ns |
| | 16 | 160.0 ns | 480.0 ns | 800.0 ns | 1.1 µs |
| | 32 | 320.0 ns | 960.0 ns | 1.6 µs | 2.2 µs |
| | 64 | 640.0 ns | 1.9 µs | 3.2 µs | 4.5 µs |
| | 128 | 1.3 µs | 3.8 µs | 6.4 µs | 9.0 µs |
| | 256 | 2.6 µs | 7.7 µs | 12.8 µs | 17.9 µs |
| | 512 | 5.1 µs | 15.4 µs | 25.6 µs | 35.8 µs |
| | 1024 | 10.2 µs | 30.7 µs | 51.2 µs | 71.7 µs |
| | 2048 | 20.5 µs | 61.4 µs | 102.4 µs | 143.4 µs |
| | 4096 | 41.0 µs | 122.9 µs | 204.8 µs | 286.7 µs |
| | 8192 | 81.9 µs | 245.8 µs | 409.6 µs | 573.4 µs |
| | 16384 | 163.8 µs | 491.5 µs | 819.2 µs | 1.1 ms |
| | 32768 | 327.7 µs | 983.0 µs | 1.6 ms | 2.3 ms |
| | 65536 | 655.4 µs | 2.0 ms | 3.3 ms | 4.6 ms |

## 10.10.4 Calculation of FIFO Pointer Addresses

The user has complete visibility of the TX and RX FIFO contents through the FIFO registers, and valid entries can be identified through a memory mapped pointer and a memory mapped counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the transmit next pointer (TXNXTPTR). For the RX FIFO the first-in pointer is the pop next pointer (POPNXTPTR).

Refer to Section 10.9.3.4, “Transmit First In First Out (TX FIFO) Buffering Mechanism, and Section 10.9.3.5, “Receive First In First Out (RX FIFO) Buffering Mechanism, for details on the FIFO operation. The TX FIFO is chosen for the illustration, but the concepts carry over to the RX FIFO.

Figure 10-23 illustrates the concept of first-in and last-in FIFO entries along with the FIFO counter.

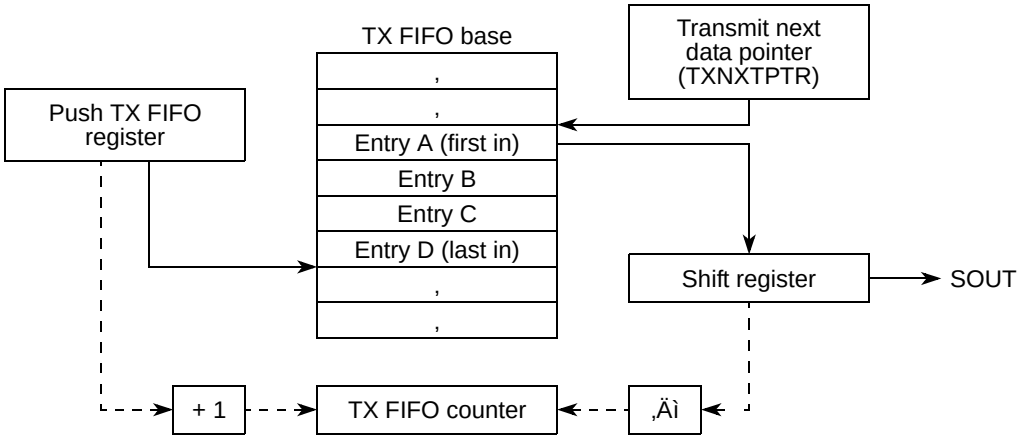



**Figure 10-23. TX FIFO Pointers and Counter**

### 10.10.4.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

First-in entry address = TXFIFO base + 4 (TXNXTPTR)

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

Last-in entry address = TXFIFO base + 4 x [(TXCTR + TXNXTPTR - 1) modulo TXFIFO depth]

where:

TXFIFO base = base address of transmit FIFO

TXCTR = transmit FIFO counter

TXNXTPTR = transmit next pointer

TX FIFO depth = transmit FIFO depth, implementation specific

### 10.10.4.2 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

First-in entry address = RXFIFO base + 4 x (POPNXTPTR)

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

Last-in entry address = RXFIFO base + 4 x [(RXCTR + POPNXTPTR - 1) modulo RXFIFO depth]

where:

RXFIFO base = base address of receive FIFO

RXCTR = receive FIFO counter

POPNXTPTR = pop next pointer

RX FIFO depth = receive FIFO depth, implementation specific

# Chapter 11
# Display Control Unit (DCU3)

## 11.1 Introduction

The Display Control Unit (DCU3) is a system master that fetches graphics stored in internal or external memory and displays them on a TFT LCD panel. A wide range of panel sizes is supported and the timing of the interface signals is highly configurable. Graphics are read directly from memory and then blended in real-time, which allows for dynamic content creation with minimal CPU intervention. Graphics may be encoded in a variety of formats to optimise memory usage. The DCU3 also has the capability of displaying real-time video from an external video source.

## 11.1.1 Overview

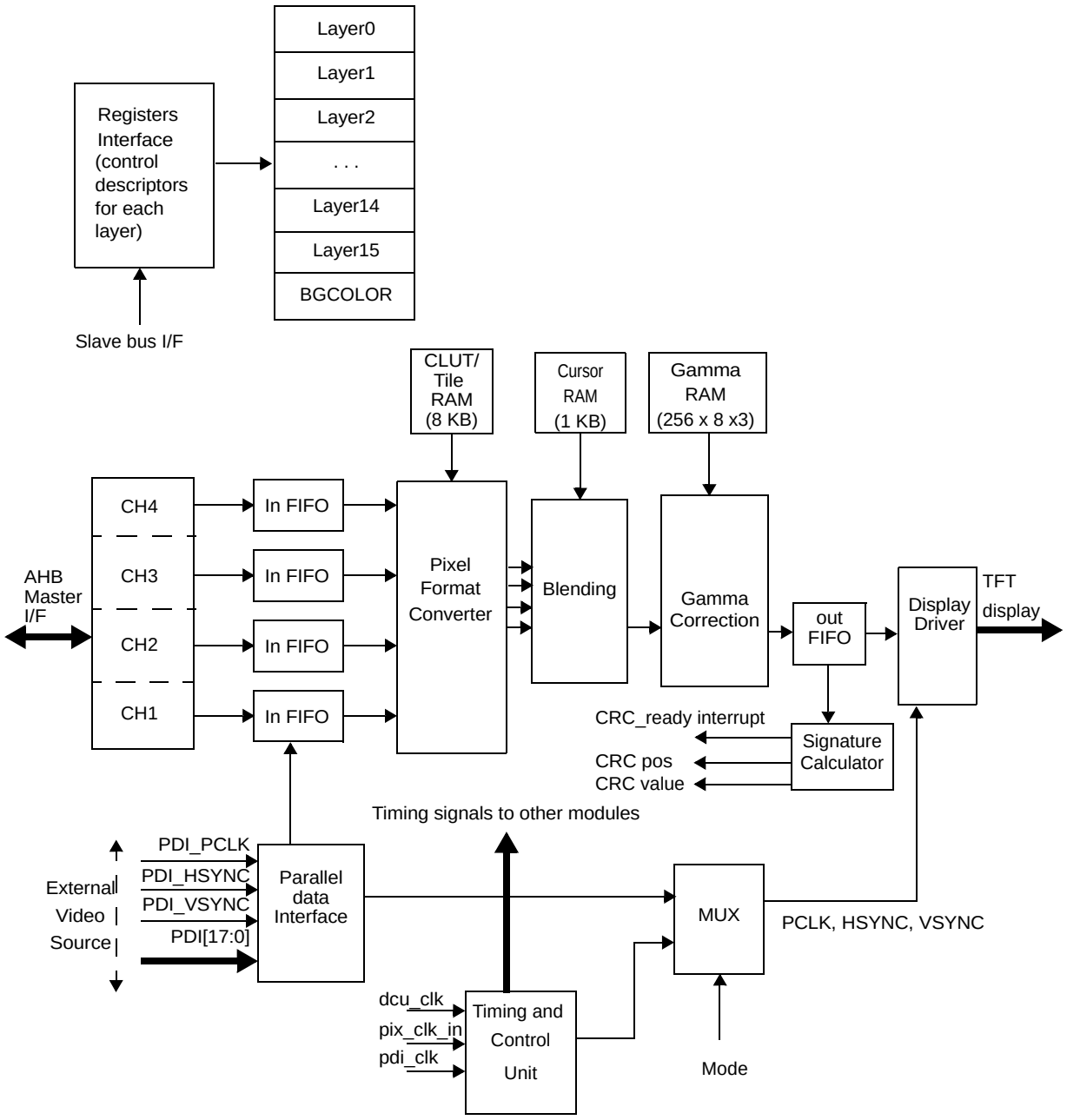



**Figure 11-1. DCU3 block diagram**

Figure 11-1 shows the DCU3 architecture. This comprises two distinct sections. The lower section shows the functional blocks of the DCU3 that fetch the graphic and video content and drive the TFT LCD panel. The upper section describes the user interface through which the user configures the graphical content of the TFT LCD panel.

The sections are analogous to the structure of communications modules, such as the FlexCAN, where one part of the module is configured to connect with the communications bus through bit-timing, parity, baud rate, etc., while a different part is used to store the data content and message identifiers.

The configuration of the lower section is dependent on the specific TFT LCD panel and optional real-time video hardware that are attached to the DCU3 inputs and outputs. In most cases, this is configured once for the hardware in use before the DCU3 is enabled. When active, this section automatically:

- Calculates the relevant graphical content for each pixel
- Fetches the source graphics from memory using its internal DMA channels (labelled CH1 to CH4)
- Converts the graphic value of each fetched pixel into full quality color format (if required)
- Calculates the required pixel value by blending the values of up to four separate graphics
- Performs a gamma correction on the pixel value (if required)
- Sends the pixel value to the TFT LCD display over its data bus
- Sets flags to indicate end of frame, buffer threshold, and other status changes

The upper section describes the characteristics of the graphics to be displayed on the panel and how they are blended together. The DCU3 manages the graphical content of the panel through sets of registers called layers. There are 16 layers available in the DCU3 and each contains the following information:

- Horizontal and vertical size of graphic
- Position of graphic on the panel
- Address of graphic in memory
- Color encoding format and color palettes (if required)
- Type and depth of blending
- Range of colors identified for chroma blending
- Tile size

The values in these registers may be changed at any time, and the panel content will be updated when the next full frame is ready to be displayed. The layers are set to a fixed priority, and this is used by the lower section to define which layers are blended, in which order, on the panel.

The upper section also contains configuration registers for a cursor graphic, the default background color, interrupt enables, test graphic, and simple register protection settings.

### 11.1.2 Features

The DCU3 has these features:

- Full RGB888 output to TFT LCD panel
- Optional support for panels without built in TCON features using the on-chip TCON module
- 16 graphics layers, a default background color layer and a  cursor layer with integrated blinking option
- Blending of each pixel using up to 4 source layers dependent on size of panel
- Programmable panel size up to a maximum of Wide VGA (800 x 480)
- Gamma correction with 8-bit resolution on each color component

- Safety mode for tagging pixels on highest priority layers
- Digital video input with and without sync extraction per ITU-R BT.656 supporting multiple video input formats including RGB666, RGB565, monochrome and YCbCr422
- Dedicated memory blocks to store a cursor and Color Look Up Tables (CLUTs)
- Temporal Dithering.

Each graphic layer has the following attributes:

- Can be placed with one pixel resolution in either axis
- Can also be placed in negative X and Y directions
- Supports multiple color-encoding formats including:
  - 1, 2, 4 and 8 bits per pixel indexed colors with alpha channel
  - APAL8 indexed colors with alpha channel
  - RGB565 and RGB888 direct colors
  - ARGB1555, ARGB4444, and BGRA8888 direct colors with an alpha channel
  - YCbCr422 format
- Alpha blending with 8-bit resolution
- Chroma-key blending for anti-mask encoding
- Multiple alpha and chroma-key blending modes
- Transparency modes for anti-aliased text and graphics
- Luminance mode for highlighting content
- Tile mode for efficient creation of textured background content
- Support for Run Length Encoding (RLE) compression
- Optimized mode for use with DDR memory

### 11.1.3 Modes of operation

The DCU3 has four modes of operation:

- **DCU_OFF**: When in this mode, the DCU3 is turned off. All the logic in the design is put in reset state to reduce power.
- **NORMAL_MODE**: The DCU3 displays and blends the graphics specified by the layer descriptors.
- **PDI_MODE**: A mode which fetches video from an external video source and combines that with the graphics configured on the layers.
- **COLBAR_MODE**: Color bar generation for testing purposes.

## 11.2 External signal description

### 11.2.1 Overview

The DCU3 has up to 22 input signals and up to 30 output signals. See Figure 11-2. The choice of signals used depends on the configuration of the DCU3. All active signals must be enabled by configuring the appropriate PCR registers in the SIUL module.

If required, the DCU3 output signals can be configured to drive a panel which does not have an embedded timing controller by enabling the on-chip TCON module.

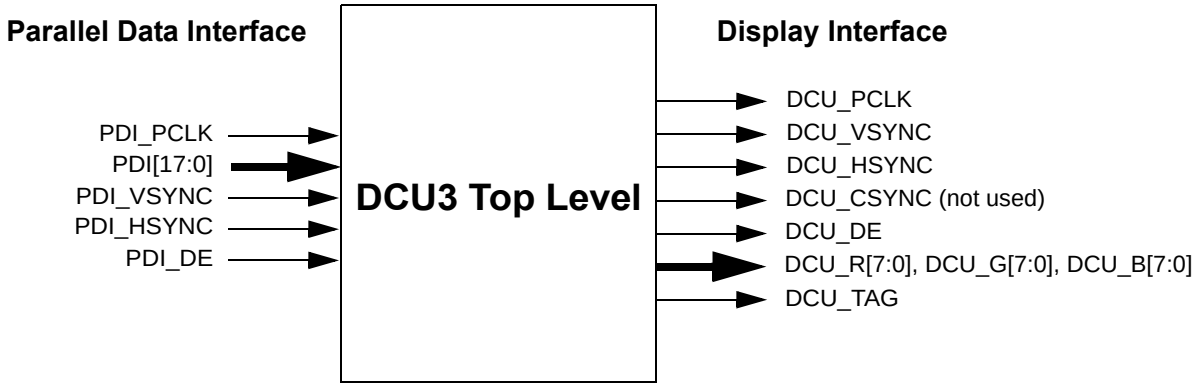



**Figure 11-2. External signals**

### 11.2.2 Detailed signal descriptions

**Table 11-1. Detailed signal descriptions**

| Signal | Direction | Description |
|---|---|---|
| Parallel Data Interface (Camera Interface) | | |
| PDI_PCLK | IN | Clock for the parallel data from the input video data |
| PDI_VSYNC | IN | Vertical sync to indicate the start of new frame for the display |
| PDI_HSYNC | IN | Horizontal sync to indicate the start of new line for the display |
| PDI_DE | IN | Data Enable for the camera data input |
| PDI[17:0] | IN | 18-bit parallel input data for the display |
| Display Interface | | |
| DCU_PCLK | OUT | Pixel clock used to drive the display panel |
| DCU_VSYNC | OUT | Vertical sync signal, indicating the beginning of a new frame |
| DCU_HSYNC | OUT | Horizontal sync signal, indicating the beginning of a new line |
| DCU_TAG | OUT | When high, this signal indicates that the pixel is tagged and an application can calculate CRC externally on this pixel. |

Table 11-1. Detailed signal descriptions (continued)

| Signal | Direction | Description |
|---|---|---|
| DCU_DE | OUT | Data Enable. Qualifies the data output |
| DCU_R[7:0], DCU_G[7:0], DCU_B[7:0] | OUT | Red, green and blue data output. |

## 11.3 Memory map and register definition

### 11.3.1 Memory map

Table 11-2 shows the memory map of the DCU3.

Table 11-2. DCU3 memory map

| Parameter | Address range |
|---|---|
| Register address space | 0x0000 – 0x03FF |
| Cursor address space | 0x0400 – 0x07FF |
| Gamma_R address space | 0x0800 – 0x0BFF |
| Gamma_G address space | 0x0C00 – 0x0FFF |
| Gamma_B address space | 0x1000 – 0x13FF |
| Empty space | 0x1400 – 0x1FFF |
| CLUT/tile address space | 0x2000 – 0x3FFF |

### 11.3.2 Register map

Table 11-3 provides the register map of the DCU3.

Only 32-bit writes and 32-bit aligned access are supported. Byte and half-word accesses are not supported.

Table 11-3. DCU3 register map

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x000 | CtrlDescL0_1 Register | R/W | 0x00000000 | on page 357 |
| 0x004 | CtrlDescL0_2 Register | R/W | 0x00000000 | on page 358 |
| 0x008 | CtrlDescL0_3 Register | R/W | 0x00000000 | on page 359 |
| 0x00C | CtrlDescL0_4 Register | R/W | 0x00000000 | on page 360 |
| 0x010 | CtrlDescL0_5 Register | R/W/ | 0x00000000 | on page 362 |
| 0x014 | CtrlDescL0_6 Register | R/W/ | 0x00000000 | on page 363 |
| 0x018 | CtrlDescL0_7 Register | R/W/ | 0x00000000 | on page 365 |
| 0x01C | CtrlDescL1_1 Register | R/W | 0x00000000 | on page 357 |
| 0x020 | CtrlDescL1_2 Register | R/W | 0x00000000 | on page 358 |

**Table 11-3. DCU3 register map (continued)**

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x024 | CtrlDescL1_3 Register | R/W | 0x00000000 | on page 359 |
| 0x028 | CtrlDescL1_4 Register | R/W | 0x00000000 | on page 360 |
| 0x02C | CtrlDescL1_5 Register | R/W | 0x00000000 | on page 362 |
| 0x030 | CtrlDescL1_6 Register | R/W | 0x00000000 | on page 363 |
| 0x034 | CtrlDescL1_7 Register | R/W | 0x00000000 | on page 365 |
| 0x038 | CtrlDescL2_1 Register | R/W | 0x00000000 | on page 357 |
| 0x03C | CtrlDescL2_2 Register | R/W | 0x00000000 | on page 358 |
| 0x040 | CtrlDescL2_3 Register | R/W | 0x00000000 | on page 359 |
| 0x044 | CtrlDescL2_4 Register | R/W | 0x00000000 | on page 360 |
| 0x048 | CtrlDescL2_5 Register | R/W | 0x00000000 | on page 362 |
| 0x04C | CtrlDescL2_6 Register | R/W | 0x00000000 | on page 363 |
| 0x050 | CtrlDescL2_7 Register | R/W | 0x00000000 | on page 365 |
| 0x054 | CtrlDescL3_1 Register | R/W | 0x00000000 | on page 357 |
| 0x058 | CtrlDescL3_2 Register | R/W | 0x00000000 | on page 358 |
| 0x05C | CtrlDescL3_3 Register | R/W | 0x00000000 | on page 359 |
| 0x060 | CtrlDescL3_4 Register | R/W | 0x00000000 | on page 360 |
| 0x064 | CtrlDescL3_5 Register | R/W | 0x00000000 | on page 362 |
| 0x068 | CtrlDescL3_6 Register | R/W | 0x00000000 | on page 363 |
| 0x06C | CtrlDescL3_7 Register | R/W | 0x00000000 | on page 365 |
| 0x070 | CtrlDescL4_1 Register | R/W | 0x00000000 | on page 357 |
| 0x074 | CtrlDescL4_2 Register | R/W | 0x00000000 | on page 358 |
| 0x078 | CtrlDescL4_3 Register | R/W | 0x00000000 | on page 359 |
| 0x07C | CtrlDescL4_4 Register | R/W | 0x00000000 | on page 360 |
| 0x080 | CtrlDescL4_5 Register | R/W | 0x00000000 | on page 362 |
| 0x084 | CtrlDescL4_6 Register | R/W | 0x00000000 | on page 363 |
| 0x088 | CtrlDescL4_7 Register | R/W | 0x00000000 | on page 365 |
| 0x08C | CtrlDescL5_1 Register | R/W | 0x00000000 | on page 357 |
| 0x090 | CtrlDescL5_2 Register | R/W | 0x00000000 | on page 358 |
| 0x094 | CtrlDescL5_3 Register | R/W | 0x00000000 | on page 359 |
| 0x098 | CtrlDescL5_4 Register | R/W | 0x00000000 | on page 360 |
| 0x09C | CtrlDescL5_5 Register | R/W | 0x00000000 | on page 362 |
| 0x0A0 | CtrlDescL5_6 Register | R/W | 0x00000000 | on page 363 |
| 0x0A4 | CtrlDescL5_7 Register | R/W | 0x00000000 | on page 365 |

Table 11-3. DCU3 register map (continued)

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x0A8 | CtrlDescL6_1 Register | R/W | 0x00000000 | on page 357 |
| 0x0AC | CtrlDescL6_2 Register | R/W | 0x00000000 | on page 358 |
| 0x0B0 | CtrlDescL6_3 Register | R/W | 0x00000000 | on page 359 |
| 0x0B4 | CtrlDescL6_4 Register | R/W | 0x00000000 | on page 360 |
| 0x0B8 | CtrlDescL6_5 Register | R/W | 0x00000000 | on page 362 |
| 0x0BC | CtrlDescL6_6 Register | R/W | 0x00000000 | on page 363 |
| 0x0C0 | CtrlDescL6_7 Register | R/W | 0x00000000 | on page 365 |
| 0x0C4 | CtrlDescL7_1 Register | R/W | 0x00000000 | on page 357 |
| 0x0C8 | CtrlDescL7_2 Register | R/W | 0x00000000 | on page 358 |
| 0x0CC | CtrlDescL7_3 Register | R/W | 0x00000000 | on page 359 |
| 0x0D0 | CtrlDescL7_4 Register | R/W | 0x00000000 | on page 360 |
| 0x0D4 | CtrlDescL7_5 Register | R/W | 0x00000000 | on page 362 |
| 0x0D8 | CtrlDescL7_6 Register | R/W | 0x00000000 | on page 363 |
| 0x0DC | CtrlDescL7_7 Register | R/W | 0x00000000 | on page 365 |
| 0x0E0 | CtrlDescL8_1 Register | R/W | 0x00000000 | on page 357 |
| 0x0E4 | CtrlDescL8_2 Register | R/W | 0x00000000 | on page 358 |
| 0x0E8 | CtrlDescL8_3 Register | R/W | 0x00000000 | on page 359 |
| 0x0EC | CtrlDescL8_4 Register | R/W | 0x00000000 | on page 360 |
| 0x0F0 | CtrlDescL8_5 Register | R/W | 0x00000000 | on page 362 |
| 0x0F4 | CtrlDescL8_6 Register | R/W | 0x00000000 | on page 363 |
| 0x0F8 | CtrlDescL8_7 Register | R/W | 0x00000000 | on page 365 |
| 0x0FC | CtrlDescL9_1 Register | R/W | 0x00000000 | on page 357 |
| 0x100 | CtrlDescL9_2 Register | R/W | 0x00000000 | on page 358 |
| 0x104 | CtrlDescL9_3 Register | R/W | 0x00000000 | on page 359 |
| 0x108 | CtrlDescL9_4 Register | R/W | 0x00000000 | on page 360 |
| 0x10C | CtrlDescL9_5 Register | R/W | 0x00000000 | on page 362 |
| 0x110 | CtrlDescL9_6 Register | R/W | 0x00000000 | on page 363 |
| 0x114 | CtrlDescL9_7 Register | R/W | 0x00000000 | on page 365 |
| 0x118 | CtrlDescL10_1 Register | R/W | 0x00000000 | on page 357 |
| 0x11C | CtrlDescL10_2 Register | R/W | 0x00000000 | on page 358 |
| 0x120 | CtrlDescL10_3 Register | R/W | 0x00000000 | on page 359 |
| 0x124 | CtrlDescL10_4 Register | R/W | 0x00000000 | on page 360 |
| 0x128 | CtrlDescL10_5 Register | R/W | 0x00000000 | on page 362 |

Table 11-3. DCU3 register map (continued)

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x12C | CtrlDescL10_6 Register | R/W | 0x00000000 | on page 363 |
| 0x130 | CtrlDescL10_7 Register | R/W | 0x00000000 | on page 365 |
| 0x134 | CtrlDescL11_1 Register | R/W | 0x00000000 | on page 357 |
| 0x138 | CtrlDescL11_2 Register | R/W | 0x00000000 | on page 358 |
| 0x13C | CtrlDescL11_3 Register | R/W | 0x00000000 | on page 359 |
| 0x140 | CtrlDescL11_4 Register | R/W | 0x00000000 | on page 360 |
| 0x144 | CtrlDescL11_5 Register | R/W | 0x00000000 | on page 362 |
| 0x148 | CtrlDescL11_6 Register | R/W | 0x00000000 | on page 363 |
| 0x14C | CtrlDescL11_7 Register | R/W | 0x00000000 | on page 365 |
| 0x150 | CtrlDescL12_1 Register | R/W | 0x00000000 | on page 357 |
| 0x154 | CtrlDescL12_2 Register | R/W | 0x00000000 | on page 358 |
| 0x158 | CtrlDescL12_3 Register | R/W | 0x00000000 | on page 359 |
| 0x15C | CtrlDescL12_4 Register | R/W | 0x00000000 | on page 360 |
| 0x160 | CtrlDescL12_5 Register | R/W | 0x00000000 | on page 362 |
| 0x164 | CtrlDescL12_6 Register | R/W | 0x00000000 | on page 363 |
| 0x168 | CtrlDescL12_7 Register | R/W | 0x00000000 | on page 365 |
| 0x16C | CtrlDescL13_1 Register | R/W | 0x00000000 | on page 357 |
| 0x170 | CtrlDescL13_2 Register | R/W | 0x00000000 | on page 358 |
| 0x174 | CtrlDescL13_3 Register | R/W | 0x00000000 | on page 359 |
| 0x178 | CtrlDescL13_4 Register | R/W | 0x00000000 | on page 360 |
| 0x17C | CtrlDescL13_5 Register | R/W | 0x00000000 | on page 362 |
| 0x180 | CtrlDescL13_6 Register | R/W | 0x00000000 | on page 363 |
| 0x184 | CtrlDescL13_7 Register | R/W | 0x00000000 | on page 365 |
| 0x188 | CtrlDescL14_1 Register | R/W | 0x00000000 | on page 357 |
| 0x18C | CtrlDescL14_2 Register | R/W | 0x00000000 | on page 358 |
| 0x190 | CtrlDescL14_3 Register | R/W | 0x00000000 | on page 359 |
| 0x194 | CtrlDescL14_4 Register | R/W | 0x00000000 | on page 360 |
| 0x198 | CtrlDescL14_5 Register | R/W | 0x00000000 | on page 362 |
| 0x19C | CtrlDescL14_6 Register | R/W | 0x00000000 | on page 363 |
| 0x1A0 | CtrlDescL14_7 Register | R/W | 0x00000000 | on page 365 |
| 0x1A4 | CtrlDescL15_1 Register | R/W | 0x00000000 | on page 357 |
| 0x1A8 | CtrlDescL15_2 Register | R/W | 0x00000000 | on page 358 |
| 0x1AC | CtrlDescL15_3 Register | R/W | 0x00000000 | on page 359 |

**Table 11-3. DCU3 register map (continued)**

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x1B0 | CtrlDescL15_4 Register | R/W | 0x00000000 | on page 360 |
| 0x1B4 | CtrlDescL15_5 Register | R/W | 0x00000000 | on page 362 |
| 0x1B8 | CtrlDescL15_6 Register | R/W | 0x00000000 | on page 363 |
| 0x1BC | CtrlDescL15_7 Register | R/W | 0x00000000 | on page 365 |
| 0x1C0 | CtrlDescCursor_1 Register | R/W | 0x00000000 | on page 365 |
| 0x1C4 | CtrlDescCursor_2 Register | R/W | 0x00000000 | on page 366 |
| 0x1C8 | CtrlDescCursor_3 Register | R/W | 0x00000000 | on page 367 |
| 0x1CC | CtrlDescCursor_4 Register | R/W | 0x00000000 | on page 367 |
| 0x1D0 | DCU_MODE Register | R/W | 0x00000000 | on page 368 |
| 0x1D4 | BGND Register | R/W | 0x00000000 | on page 370 |
| 0x1D8 | DISP_SIZE Register | R/W | 0x00000000 | on page 371 |
| 0x1DC | HSYN_PARA Register | R/W | 0x00C01803 | on page 371 |
| 0x1E0 | VSYN_PARA Register | R/W | 0x00C01803 | on page 372 |
| 0x1E4 | SYNPOL Register | R/W | 0x00000000 | on page 373 |
| 0x1E8 | THRESHOLD Register | R/W | 0x0000780A | on page 374 |
| 0x1EC | INT_STATUS Register | R | 0x00000000 | on page 375 |
| 0x1F0 | INT_MASK Register | R/W | 0x000F4FFF | on page 376 |
| 0x1F4 | COLBAR_1 Register | R/W | 0xFF000000 | on page 379 |
| 0x1F8 | COLBAR_2 Register | R/W | 0xFF0000FF | on page 379 |
| 0x1FC | COLBAR_3 Register | R/W | 0xFF00FFFF | on page 380 |
| 0x200 | COLBAR_4 Register | R/W | 0xFF00FF00 | on page 380 |
| 0x204 | COLBAR_5 Register | R/W | 0xFFFFFF00 | on page 381 |
| 0x208 | COLBAR_6 Register | R/W | 0xFFFF0000 | on page 381 |
| 0x20C | COLBAR_7 Register | R/W | 0xFFFF00FF | on page 382 |
| 0x210 | COLBAR_8 Register | R/W | 0xFFFFFFFF | on page 382 |
| 0x214 | DIV_RATIO Register | R/W | 0x0000001F | on page 382 |
| 0x218 | SIGN_CALC_1 Register | R/W | 0x00000000 | on page 383 |
| 0x21C | SIGN_CALC_2 Register | R/W | 0x00000000 | on page 384 |
| 0x220 | CRC_VAL Register | R/W | 0x00000000 | on page 384 |
| 0x224 | PDI_STATUS Register | R/W | 0x00000000 | on page 385 |
| 0x228 | MASK_PDI_STATUS Register | R/W | 0x000003FF | on page 386 |
| 0x22C | PARR_ERR_STATUS Register | R/W | 0x00000000 | on page 387 |
| 0x230 | MASK_PARR_ERR_STATUS Register | R/W | 0x000FFFFF | on page 390 |

**Table 11-3. DCU3 register map (continued)**

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x234 | THRESHOLD_INP_BUF_1 Register | R/W | 0x7F007F00 | on page 392 |
| 0x238 | THRESHOLD_INP_BUF_2 Register | R/W | 0x7F007F00 | on page 393 |
| 0x23C | LUMA_COMP Register | R/W | 0x9512A254 | on page 393 |
| 0x240 | CHROMA_RED Register | R/W | 0x03310000 | on page 394 |
| 0x244 | CHROMA_GREEN Register | R/W | 0x06600F38 | on page 394 |
| 0x248 | CHROMA_BLUE Register | R/W | 0x00000409 | on page 395 |
| 0x24C | CRC_POS Register | R/W | 0x00000000 | on page 396 |
| 0x250 | FG0_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x254 | FG0_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x258 | FG1_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x25C | FG1_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x260 | FG2_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x264 | FG2_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x268 | FG3_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x26C | FG3_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x270 | FG4_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x274 | FG4_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x278 | FG5_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x27C | FG5_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x280 | FG6_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x284 | FG6_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x288 | FG7_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x28C | FG7_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x290 | FG8_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x294 | FG8_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x298 | FG9_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x29C | FG9_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x2A0 | FG10_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x2A4 | FG10_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x2A8 | FG11_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x2AC | FG11_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x2B0 | FG12_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x2B4 | FG12_bcolor Register | R/W | 0x00000000 | on page 397 |

Table 11-3. DCU3 register map (continued)

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x2B8 | FG13_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x2BC | FG13_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x2C0 | FG14_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x2C4 | FG14_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x2C8 | FG15_fcolor Register | R/W | 0x00000000 | on page 396 |
| 0x2CC | FG15_bcolor Register | R/W | 0x00000000 | on page 397 |
| 0x2D0 | LYR_INTPOL_EN | R/W | 0x00000000 | on page 399 |
| 0x2D4 | LYR_LUMA_COMP Register | R/W | 0x9512A254 | on page 399 |
| 0x2D8 | LYR_CHROMA_RED Register | R/W | 0x03310000 | on page 400 |
| 0x2DC | LYR_CHROMA_GREEN Register | R/W | 0x06600F38 | on page 400 |
| 0x2E0 | LYR_CHROMA_BLUE Register | R/W | 0x00000409 | on page 401 |
| 0x2E4 | COMP_IMSIZE Register | R/W | 0x00000000 | on page 402 |
| 0x2E8–0x2FC | Reserved | | | |
| 0x300 | Global Protection Register | R/W | 0x00000000 | on page 402 |
| 0x304 | Soft Lock Bit Register L0 | R/W | 0x00000000 | on page 403 |
| 0x308 | Soft Lock Bit Register L1 | R/W | 0x00000000 | on page 405 |
| 0x30C | Soft Lock Bit Register DISP_SIZE | R/W | 0x00000000 | on page 406 |
| 0x310 | Soft Lock Bit Register VSYNC/HSYNC PARA | R/W | 0x00000000 | on page 407 |
| 0x314 | Soft Lock Bit Register POL | R/W | 0x00000000 | on page 408 |
| 0x318 | Soft Lock Bit Register L0_TRANSP | R/W | 0x00000000 | on page 408 |
| 0x31C | Soft Lock Bit Register L1_TRANSP | R/W | 0x00000000 | on page 410 |

## 11.3.3 Register summary

Figure 11-3 provides a key for register figures and tables and the register summary.

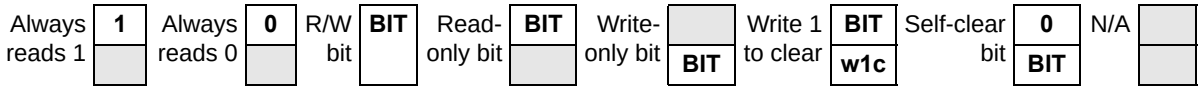



Figure 11-3. Key to register fields

The conventions in Table 11-4 serve as a key for the register summary and individual register diagrams.

**Table 11-4. Register conventions**

| Convention Des | cription |
|---|---|
| | Depending on its placement in the read or write row, indicates that the bit is not readable or not writable |
| FIELDNAME | Identifies the field. Its presence in the read or write row indicates that it can be read or written. |
| **Register Field Types** | |
| R | Read only. Writing this bit has no effect |
| W | Write only |
| R/W | Standard read/write bit. Only software can change the bit's value (other than a hardware reset). |
| rwm | A read/write bit that can be modified by hardware in some fashion other than by a reset |
| w1c | Write one to clear. A status bit that can be read, and is cleared by writing a one. |
| Self-clearing bit | Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr) |
| S | Set: Pattern on the data bus is ORed with and written into the register. |
| C | Clear: Pattern on the data bus is a mask. If a bit on the mask is set, then the corresponding register bit is cleared. |
| **Reset Values** | |
| 0 | Resets to zero |
| 1 | Resets to one |
| — | Undefined at reset |
| u | Unaffected by reset |
| [*signal_name*] | Reset value is determined by polarity of indicated signal. |

**Table 11-5. Register descriptions**

| Name Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| CtrlDescL0_1 0x000 | R | 0 | 0 | 0 | 0 | 0 | 0 | HEIGHT | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | WIDTH | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescL0_2 0x004 | R | 0 | 0 | 0 | 0 | 0 | POSY | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | POSX | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 11-5. Register descriptions (continued)**

| Name<br>Offset | | 0<br>16 | 1<br>17 | 2<br>18 | 3<br>19 | 4<br>20 | 5<br>21 | 6<br>22 | 7<br>23 | 8<br>24 | 9<br>25 | 10<br>26 | 11<br>27 | 12<br>28 | 13<br>29 | 14<br>30 | 15<br>31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CtrlDescL0_3<br>0x008 | R | ADDR | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescL0_4<br>0x00C | R | EN | TILE_EN | DATA_SEL | SAFETY_EN | TRANS | | | | | | | | BPP | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | RLE_EN | LUOFFS | | | | | | | | | | | 0 | BB | AB | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescL0_5<br>0x010 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CKMAX_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | CKMAX_G | | | | | | | | CKMAX_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescL0_6<br>0x014 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CKMIN_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | CKMIN_G | | | | | | | | CKMIN_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescL0_7<br>0x018 | R | 0 | 0 | 0 | 0 | 0 | TILE_VER_SIZE | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | TILE_HOR_SIZE | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescCursor_1<br>0x1C0 | R | 0 | 0 | 0 | 0 | 0 | HEIGHT | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | WIDTH | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescCursor_2<br>0x1C4 | R | 0 | 0 | 0 | 0 | 0 | POSY | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | POSX | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 11-5. Register descriptions (continued)**

| Name / Offset | | 0 / 16 | 1 / 17 | 2 / 18 | 3 / 19 | 4 / 20 | 5 / 21 | 6 / 22 | 7 / 23 | 8 / 24 | 9 / 25 | 10 / 26 | 11 / 27 | 12 / 28 | 13 / 29 | 14 / 30 | 15 / 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CtrlDescCursor_3 0x1C8 | R | CUR_EN | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DEFAULT_CURSOR_COLOR[0:7] | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | DEFAULT_CURSOR_COLOR[8:23] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescCursor_4 0x1CC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | HWC_BLINK_OFF | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | EN_BLINK | HWC_BLINK_ON | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| DCU_MODE 0x1D0 | R | DCU_SW_RESET | DITHER_EN | ADDB | | ADDG | | ADDR | | 0 | BLEND_ITER | | | PDI_SYNC_LOCK | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | PDI_INTERPOL_EN | RASTER_EN | PDI_EN | PDI_BYTE_REV | PDI_DE_MODE | PDI_NARROW_MODE | PDI_MODE | | PDI_SLAVE_MODE | TAG_EN | SIG_EN | PDI_SYNC | 0 | EN_GAMMA | DCU_MODE | |
| | W | | | | | | | | | | | | | | | | |
| BGND 0x1D4 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BGND_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | BGND_G | | | | | | | | BGND_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| DISP_SIZE 0x1D8 | R | 0 | 0 | 0 | 0 | 0 | DELTA_Y | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DELTA_X | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| HSYN_PARA 0x1DC | R | 0 | BP_H | | | | | | | | | 0 | 0 | PW_H[0:3] | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | PW_H[4:8] | | | | | 0 | 0 | FP_H | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 11-5. Register descriptions (continued)**

| Name<br>Offset | | 0<br>16 | 1<br>17 | 2<br>18 | 3<br>19 | 4<br>20 | 5<br>21 | 6<br>22 | 7<br>23 | 8<br>24 | 9<br>25 | 10<br>26 | 11<br>27 | 12<br>28 | 13<br>29 | 14<br>30 | 15<br>31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VSYN_PARA<br>0x1E0 | R | 0 | BP_V | | | | | | | | | 0 | 0 | PW_V[0:3] | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | PW_V[4:8] | | | | | 0 | 0 | FP_V | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| SYN_POL<br>0x1E4 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | INV_PDI_DE | INV_PDI_HS | INV_PDI_VS | INV_PDI_CLK | INV_PXCK | NEG | BP_VS | BP_HS | INV_CS | INV_VS | INV_HS |
| | W | | | | | | | | | | | | | | | | |
| THRESHOLD<br>0x1E8 | R | 0 | 0 | 0 | 0 | 0 | 0 | LS_BF_VS | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | OUT_BUF_HIGH | | | | | | | | OUT_BUF_LOW | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| INT_STATUS<br>0x1EC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | P4_FIFO_HI_FLAG | P4_FIFO_LO_FLAG | P3_FIFO_HI_FLAG | P3_FIFO_LO_FLAG |
| | W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| | R | 0 | DMA_TRANS_FINISH | 0 | 0 | IPM_ERROR | PROG_END | P2_FIFO_HI_FLAG | P2_FIFO_LO_FLAG | P1_FIFO_HI_FLAG | P1_FIFO_LO_FLAG | CRC_OVERFLOW | CRC_READY | VS_BLANK | LS_BF_VS | UNDRUN | VSYNC |
| | W | | w1c | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |

**Table 11-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| INT_MASK 0x1F0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | M_P4_FIFO_HI_FLAG | M_P4_FIFO_LO_FLAG | M_P3_FIFO_HI_FLAG | M_P3_FIFO_LO_FLAG |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | M_DMA_TRANS_FINISH | 0 | 0 | M_IPM_ERROR | M_PROG_END | M_P2_FIFO_HI_FLAG | M_P2_FIFO_LO_FLAG | M_P1_FIFO_HI_FLAG | M_P1_FIFO_LO_FLAG | M_CRC_OVERFLOW | M_CRC_READY | M_VS_BLANK | M_LS_BF_VS | M_UNDRUN | M_VSYNC |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_1 0x1F4 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_1_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_1_G | | | | | | | | COLBAR_1_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_2 0x1F8 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_2_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_2_G | | | | | | | | COLBAR_2_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_3 0x1FC | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_3_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_3_G | | | | | | | | COLBAR_3_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_4 0x200 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_4_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_4_G | | | | | | | | COLBAR_4_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_5 0x204 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_5_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_5_G | | | | | | | | COLBAR_5_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 11-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| COLBAR_6 0x208 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_6_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_6_G | | | | | | | | COLBAR_6_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_7 0x20C | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_7_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_7_G | | | | | | | | COLBAR_7_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_8 0x210 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_8_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_8_G | | | | | | | | COLBAR_8_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| DIV_RATIO 0x214 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DIV_RATIO | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| SIGN_CALC_1 0x218 | R | 0 | 0 | 0 | 0 | 0 | SIG_VER_SIZE | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | SIG_HOR_SIZE | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| SIGN_CALC_2 0x21C | R | 0 | 0 | 0 | 0 | 0 | SIG_VER_POS | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | SIG_HOR_POS | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CRC_VAL 0x220 | R | CRC_VAL | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 11-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| PDI_STATUS 0x224 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | PDI_BLANKING_ERR | PDI_ECC_ERR2 | PDI_ECC_ERR1 | PDI_LOCK_LOST | PDI_LOCK_DET | PDI_VSYNC_DET | PDI_HSYNC_DET | PDI_DE_DET | PDI_CLK_LOST | PDI_CLK_DET |
| | W | | | | | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| MASK_PDI_ STATUS 0x228 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | M_PDI_BLANKING_ERR | M_PDI_ECC_ERR2 | M_PDI_ECC_ERR1 | M_PDI_LOCK_LOST | M_PDI_LOCK_DET | M_PDI_VSYNC_DET | M_PDI_HSYNC_DET | M_PDI_DE_DET | M_PDI_CLK_LOST | M_PDI_CLK_DET |
| | W | | | | | | | | | | | | | | | | |
| PARR_ERR_ STATUS 0x22C | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RLE_ERR | HWC_ERR | SIG_ERR | DISP_ERR |
| | W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| | R | L15_PARR_ERR | L14_PARR_ERR | L13_PARR_ERR | L12_PARR_ERR | L11_PARR_ERR | L10_PARR_ERR | L9_PARR_ERR | L8_PARR_ERR | L7_PARR_ERR | L6_PARR_ERR | L5_PARR_ERR | L4_PARR_ERR | L3_PARR_ERR | L2_PARR_ERR | L1_PARR_ERR | L0_PARR_ERR |
| | W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |

**Table 11-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| MASK_PARR_ERR_STATUS 0x230 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | M_RLE_ERR | M_HWC_ERR | M_SIG_ERR | M_DISP_ERR |
| | W | | | | | | | | | | | | | | | | |
| | R | M_L15_PARR_ERR | M_L14_PARR_ERR | M_L13_PARR_ERR | M_L12_PARR_ERR | M_L11_PARR_ERR | M_L10_PARR_ERR | M_L9_PARR_ERR | M_L8_PARR_ERR | M_L7_PARR_ERR | M_L6_PARR_ERR | M_L5_PARR_ERR | M_L4_PARR_ERR | M_L3_PARR_ERR | M_L2_PARR_ERR | M_L1_PARR_ERR | M_L0_PARR_ERR |
| | W | | | | | | | | | | | | | | | | |
| THRESHOLD_INP_BUF 0x234 | R | 0 | INP_BUF_P2_HI | | | | | | | 0 | INP_BUF_P2_LO | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | INP_BUF_P1_HI | | | | | | | 0 | INP_BUF_P1_LO | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| THRESHOLD_INP_BUF 0x238 | R | 0 | INP_BUF_P4_HI | | | | | | | 0 | INP_BUF_P4_LO | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | INP_BUF_P3_HI | | | | | | | 0 | INP_BUF_P3_LO | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| LUMA_COMP 0x23C | R | Y_RED | | | | | | | | | | | Y_GREEN[0:4] | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | Y_GREEN[5:9] | | | | | | Y_BLUE | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CHROMA_RED 0x240 | R | 0 | 0 | 0 | 0 | 0 | CR_RED | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_RED | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CHROMA_GREEN 0x244 | R | 0 | 0 | 0 | 0 | 0 | CR_GREEN | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_GREEN | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CHROMA_BLUE 0x248 | R | 0 | 0 | 0 | 0 | 0 | CR_BLUE | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_BLUE | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 11-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| CRC_POS 0x24c | R | CRC_POS | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| FGX_ FCOLOR 0x250 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | FGX_FCOLOR[0:7] | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | FGX_FCOLOR[8:23] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| FGX_ BCOLOR 0x254 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | FGX_BCOLOR[0:7] | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | FGX_BCOLOR[8:23] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| LYR_INTPOL _EN 0x2D0 | R | EN | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| LYR_LUMA_ COMP 0x2D4 | R | Y_RED | | | | | | | | | | | Y_GREEN[0:4] | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | Y_GREEN[5:9] | | | | | | Y_BLUE | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| LYR_CHROM A_RED 0x2D8 | R | 0 | 0 | 0 | 0 | 0 | CR_RED | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_RED | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| LYR_CHROM A_GREEN 0x2Dc | R | 0 | 0 | 0 | 0 | 0 | CR_GREEN | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_GREEN | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| LYR_CHROM A_BLUE 0x2E0 | R | 0 | 0 | 0 | 0 | 0 | CR_BLUE | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_BLUE | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 11-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| COMP_IMSIZE 0x2E4 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | COMP_IMSIZE[10:15] | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COMP_IMSIZE[16:31] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| GLOBAL_PROTECTION 0x300 | R | HLB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock_Bit L0 0x304 | R | 0 | 0 | 0 | 0 | | | | | 0 | 0 | 0 | 0 | | | | 0 |
| | W | WEN_LO_1 | WEN_LO_2 | WEN_LO_3 | WEN_LO_4 | SLB_L0_1 | SLB_L0_2 | SLB_L0_3 | SLB_L0_4 | WEN_LO_5 | WEN_LO_6 | WEN_LO_7 | | SLB_L0_5 | SLB_L0_6 | SLB_L0_7 | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock_Bit L1 0x308 | R | 0 | 0 | 0 | 0 | | | | | 0 | 0 | 0 | 0 | | | | 0 |
| | W | WEN_L1_1 | WEN_L1_2 | WEN_L1_3 | WEN_L1_4 | SLB_L1_1 | SLB_L1_2 | SLB_L1_3 | SLB_L1_4 | WEN_L1_5 | WEN_L1_6 | WEN_L1_7 | | SLB_L1_5 | SLB_L1_6 | SLB_L1_7 | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock_DISP_SIZE 0x30c | R | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | WEN_DISP | | | | SLB_DISP | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock_HSYNC/VSYNC PARA 0x310 | R | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | WEN_HSYNC | WEN_VSYNC | | | SLB_HSYNC | SLB_VSYNC | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |

**Table 11-5. Register descriptions (continued)**

| Name<br>Offset | | 0<br>16 | 1<br>17 | 2<br>18 | 3<br>19 | 4<br>20 | 5<br>21 | 6<br>22 | 7<br>23 | 8<br>24 | 9<br>25 | 10<br>26 | 11<br>27 | 12<br>28 | 13<br>29 | 14<br>30 | 15<br>31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Soft_Lock_POL<br>0x314 | R | 0 | 0 | 0 | 0 | SLB_POL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | WEN_POL | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock<br>L0_TRANSP<br>0x318 | R | 0 | 0 | 0 | 0 | SLB_FCOLOR | SLB_BCOLOE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | WEN_FCOLOR | WEN_BCOLOR | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock<br>L1_TRANSP<br>0x31C | R | 0 | 0 | 0 | 0 | SLB_FCOLOR | SLB_BCOLOE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | WEN_FCOLOR | WEN_BCOLOR | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |

## 11.3.4 Register descriptions

This section describes the DCU3 registers.

### 11.3.4.1 Control Descriptor L0_1 Register (CtrlDescL0_1)

Figure 11-4 represents the control descriptor L0_1 register. This register sets the height and width of the layer associated with the register.

Offsets:
0x000 (CtrlDescL0_1)
0x01C (CtrlDescL1_1)
0x038 (CtrlDescL2_1)
0x054 (CtrlDescL3_1)
0x070 (CtrlDescL4_1)
0x08C (CtrlDescL5_1)
0x098 (CtrlDescL6_1)
0x0C4 (CtrlDescL7_1)
0x0E0 (CtrlDescL8_1)
0x0FC (CtrlDescL9_1)
0x118 (CtrlDescL10_1)
0x134 (CtrlDescL11_1)
0x150 (CtrlDescL12_1)
0x16C (CtrlDescL13_1)
0x188 (CtrlDescL14_1)
0x194 (CtrlDescL15_1)

Access: User read/write

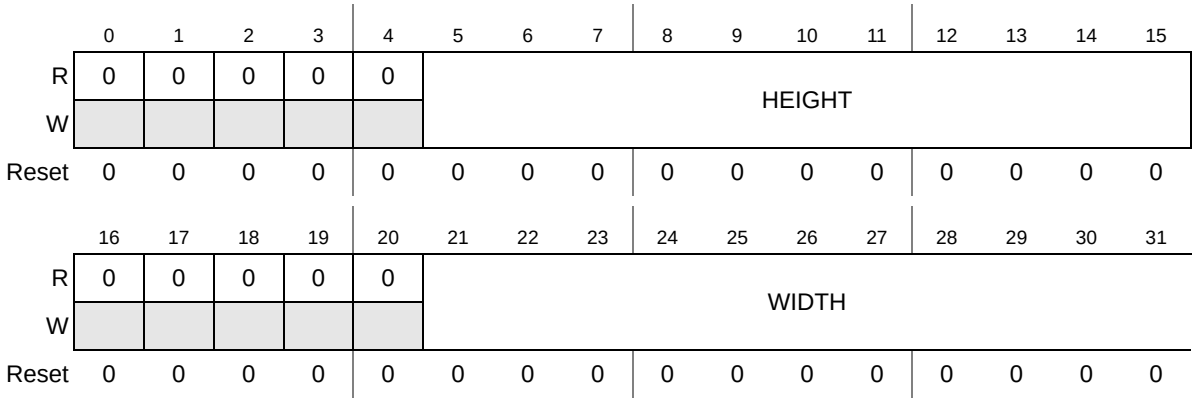



**Figure 11-4. CtrlDescL0_1 Register**

**Table 11-6. CtrlDescL0_1 Field Descriptions**

| Field | Description |
|---|---|
| 6–15<br>HEIGHT | Height of the layer in pixels |
| 20–31<br>WIDTH | Width of the layer (in pixels). The layer width must be in multiples of the number of pixels that can be stored in 32 bits except for the special case of 1 bit per pixel, and therefore differs depending on color encoding. For example, if 2 bits per pixel format is used, then the layer width must be configured in multiples of 16. See Section 11.4.4.3, “Layer size and positioning”.” |

### 11.3.4.2 Control Descriptor L0_2 Register

Figure 11-5 represents the control descriptor L0_2 register. This register sets the origin (top/left) of the layer associated with the register.

Offsets:
0x004 (CtrlDescL0_2)
0x020 (CtrlDescL1_2)
0x03C (CtrlDescL2_2)
0x058 (CtrlDescL3_2)
0x074 (CtrlDescL4_2)
0x090 (CtrlDescL5_2)
0x0AC (CtrlDescL6_2)
0x0C8 (CtrlDescL7_2)
0x0E4 (CtrlDescL8_2)
0x100 (CtrlDescL9_2)
0x11C (CtrlDescL10_2)
0x138 (CtrlDescL11_2)
0x154 (CtrlDescL12_2)
0x170 (CtrlDescL13_2)
0x18C (CtrlDescL14_2)
0x198 (CtrlDescL15_2)

Access: User read/write

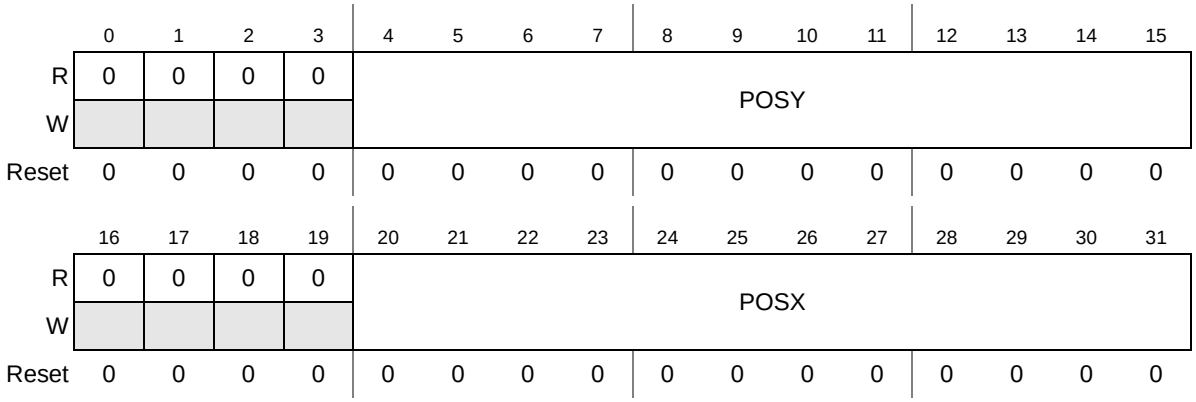



**Figure 11-5. CtrlDescL0_2 Register**

**Table 11-7. CtrlDescL0_2 Field Descriptions**

| Field | Description |
|---|---|
| POSY | Two's complement signed value setting the vertical position of top row of the layer, where 0 is the top row of the panel. Positive values are below and negative values are above the top row of the panel. |
| POSX | Two's complement signed value setting the horizontal position of left hand column of the layer, where 0 is the left-hand column of the panel. Positive values are to the right and negative values are to the left the left-hand column of the panel. |

### 11.3.4.3 Control Descriptor L0_3 Register

Figure 11-6 represents the control descriptor L0_3 register. This register sets the beginning address of layer data.

Offsets:
0x008 (CtrlDescL0_3)
0x024 (CtrlDescL1_3)
0x040 (CtrlDescL2_3)
0x05C (CtrlDescL3_3)
0x078 (CtrlDescL4_3)
0x094 (CtrlDescL5_3)
0x0b0 (CtrlDescL6_3)
0x0CC (CtrlDescL7_3)
0x0E8 (CtrlDescL8_3)
0x104 (CtrlDescL9_3)
0x120 (CtrlDescL10_3)
0x13C (CtrlDescL11_3)
0x158 (CtrlDescL12_3)
0x174 (CtrlDescL13_3)
0x18C (CtrlDescL14_3)
0x1AC (CtrlDescL15_3)

Access: User read/write

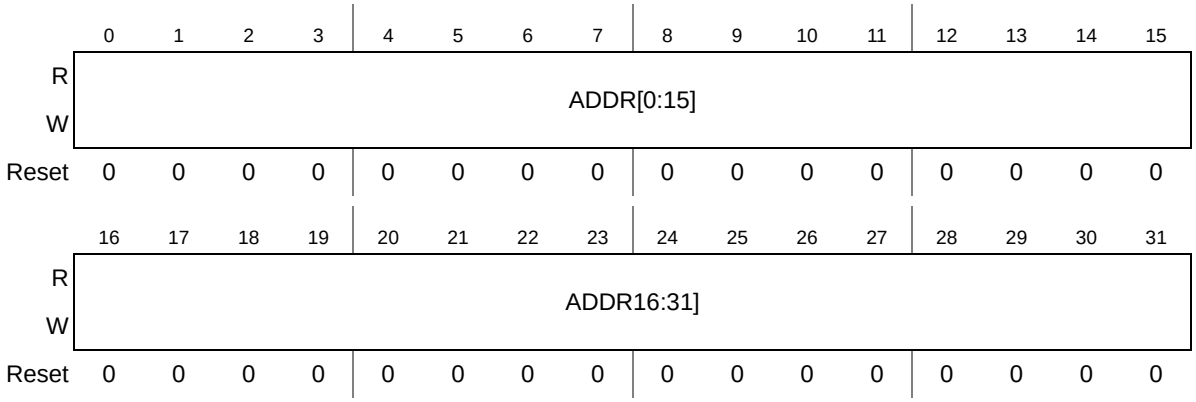



**Figure 11-6. CtrlDescL0_3 Register**

**Table 11-8. CtrlDescL0_3 Field Descriptions**

| Field | Description |
|---|---|
| ADDR | Address of layer data in the memory. The address programmed should be 64-bit aligned. |

### 11.3.4.4 Control Descriptor L0_4 Register

Figure 11-7 represents the control descriptor L0_4 register. This register controls various graphics options and whether the layer is enabled.

Offsets:
0x00C (CtrlDescL0_4)
0x028 (CtrlDescL1_4)
0x044 (CtrlDescL2_4)
0x060 (CtrlDescL3_4)
0x07C (CtrlDescL4_4)
0x098 (CtrlDescL5_4)
0x0B4 (CtrlDescL6_4)
0x0D0 (CtrlDescL7_4)
0x0EC (CtrlDescL8_4)
0x108 (CtrlDescL9_4)
0x124 (CtrlDescL10_4)
0x140 (CtrlDescL11_4)
0x15C (CtrlDescL12_4)
0x178 (CtrlDescL13_4)
0x190 (CtrlDescL14_4)
0x1B0 (CtrlDescL15_4)

Access: User read/write

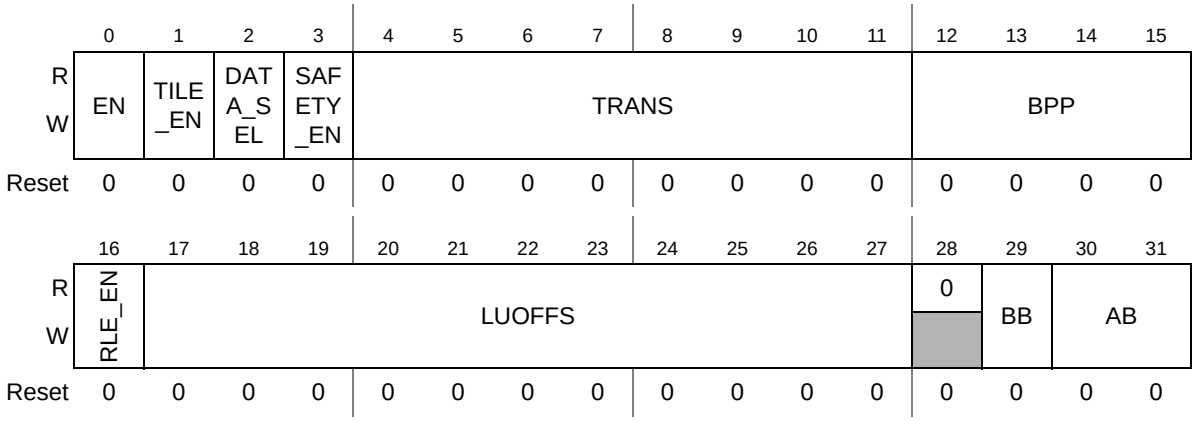



**Figure 11-7. CtrlDescL0_4 register**

**Table 11-9. CtrlDescL0_4 field descriptions**

| Field | Description |
|---|---|
| 0<br>EN | Enable the layer<br>1'b1: ON<br>1'b0: OFF |
| 1<br>TILE_EN | Enable the Tile Mode<br>1'b1: ON<br>1'b0: OFF |
| 2<br>DATA_SEL | Selects the Tile data either from MCU memory or CLUT<br>1'b0: Tile Mode data resides in the MCU memory<br>1'b1: Tile mode data resides in the CLUT |
| 3<br>SAFETY_EN | Safety Mode Enable Bit. Valid only for layer 0 and layer 1. For registers of all other layers, this should be set to 0.<br>1'b1: Safety Mode is enabled for this layer<br>1'b0: Safety Mode is disabled |
| 4–11<br>TRANS | Transparency Level. Specifies the alpha value for the layer. This value may be used by the blending engine to blend pixels on this layer. Value can vary between 0-255 where 0 is completely transparent and 255 is completely opaque. |

Table 11-9. CtrlDescL0_4 field descriptions (continued)

| Field | Description |
|---|---|
| 12–15<br>BPP | Bits Per Pixel<br>4'b0000 = 1 bpp<br>4'b0001 = 2 bpp<br>4'b0010 = 4 bpp<br>4'b0011 = 8 bpp<br>4'b0100 = 16 bpp (RGB565)<br>4'b0101 = 24 bpp<br>4'b0110 = 32 bpp (BGRA8888)<br>4'b0111 = Transparency mode 4 bpp<br>4'b1000 = Transparency mode 8bpp<br>4'b1001 = Luminance offset mode 4 bpp<br>4'b1010 = Luminance offset mode 8 bpp<br>4'b1011 = 16 bpp (ARGB1555)<br>4'b1100 = 16 bpp (ARGB4444)<br>4'b1101 = 16 bpp (APAL8)<br>4'b1110 = YCbCr422 (the blend engine allows only a single YCbCr layer in any blend operation)<br>4'b1111 = Reserved |
| 16<br>RLE_EN | Enable RLE mode for layer.<br>1'b1:Enabled<br>1'b0:Disabled |
| 17–27<br>LUOFFS | Look Up Table offset. Value gives the offset to the start address of the CLUT or tile (when used in internal tile mode) in the CLUT/TILE RAM. |
| 29<br>BB | Chroma Keying<br>1'b1: ON<br>1'b0: OFF |
| 30–31<br>AB | Alpha Blending<br>2'b00: No alpha Blending<br>2'b01: Blend only the pixels selected by chroma keying in case BB=1'b1<br>2'b10: Blend the whole frame<br>2'b11: Same functionality as 2'b00. |

### 11.3.4.5 Control Descriptor L0_5 Register

Figure 11-8 represents the control Descriptor L0_5 register. This register sets the maximum Chroma Keying values for RGB.

Refer to Section 11.4.4.5, "Alpha and chroma-key blending"," for a description of Chroma Keying.

Offset:
0x010 (CtrlDescL0_5)
0x02C (CtrlDescL1_5)
0x048 (CtrlDescL2_5)
0x064 (CtrlDescL3_5)
0x080 (CtrlDescL4_5)
0x09C (CtrlDescL5_5)
0x0B8 (CtrlDescL6_5)
0x0D4 (CtrlDescL7_5)
0x0F0 (CtrlDescL8_5)
0x10C (CtrlDescL9_5)
0x128 (CtrlDescL10_5)
0x144 (CtrlDescL11_5)
0x160 (CtrlDescL12_5)
0x17C (CtrlDescL13_5)
0x198 (CtrlDescL14_5)
0x1B4 (CtrlDescL15_5)

Access: User read/write

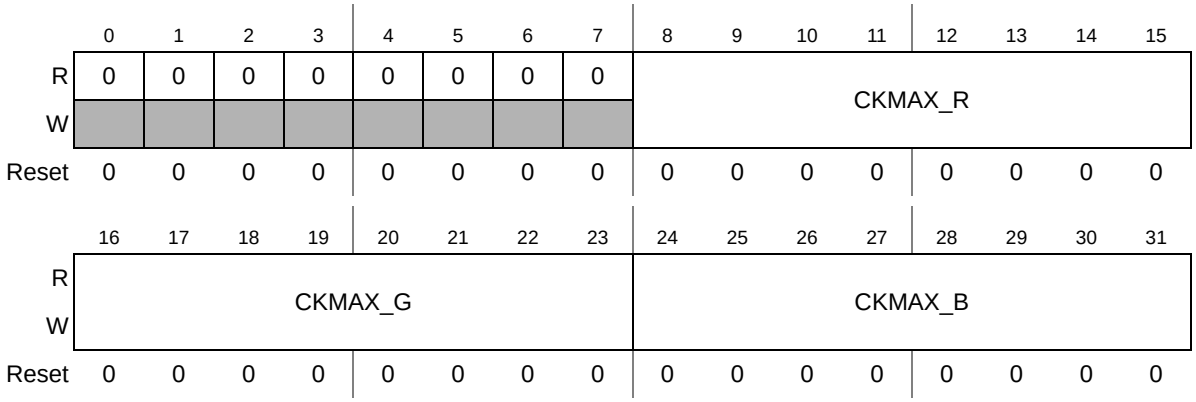



**Figure 11-8. CtrlDescL0_5 Register**

**Table 11-10. CtrlDescL0_5 Field Descriptions**

| Field | Description |
|---|---|
| 8–15<br>CKMAX_R | Chroma Keying Max Red Component |
| 16–23<br>CKMAX_G | Chroma Keying Max Green Component |
| 24–31<br>CKMAX_B | Chroma Keying Max Blue Component |

### 11.3.4.6 Control Descriptor L0_6 Register

Figure 11-9 represents the control descriptor L0_6 register. This register sets the minimum Chroma Keying values for RGB. Refer to Section 11.4.4.5, “Alpha and chroma-key blending”,” for a description of Chroma Keying.

Offset:
0x014 (CtrlDescL0_6)
0x030 (CtrlDescL1_6)
0x04C (CtrlDescL2_6)
0x068 (CtrlDescL3_6)
0x084 (CtrlDescL4_6)
0x0A0 (CtrlDescL5_6)
0x0BC (CtrlDescL6_6)
0x0D8 (CtrlDescL7_6)
0x0F4 (CtrlDescL8_6)
0x110 (CtrlDescL9_6)
0x12C (CtrlDescL10_6)
0x148 (CtrlDescL11_6)
0x164 (CtrlDescL12_6)
0x180 (CtrlDescL13_6)
0x198 (CtrlDescL14_6)
0x1B8 (CtrlDescL15_6)

Access: User read/write

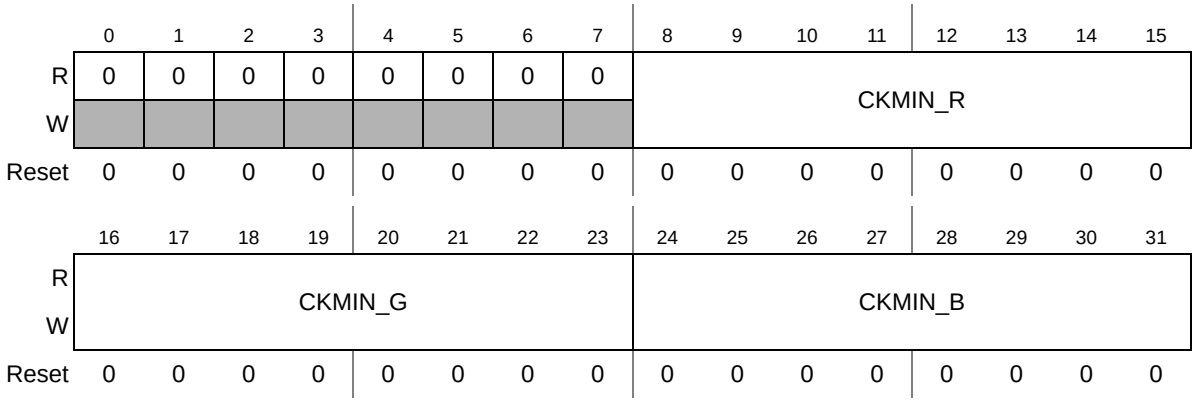



**Figure 11-9. CtrlDescL0_6 Register**

**Table 11-11. CtrlDescL0_6 Field Descriptions**

| Field | Description |
|---|---|
| 8–15<br>CKMIN_R | Chroma Keying Minimum Red Component |
| 16–23<br>CKMIN_G | Chroma Keying Minimum Green Component |
| 24–31<br>CKMIN_B | Chroma Keying Minimum Blue Component |

### 11.3.4.7 Control Descriptor L0_7 Register

Figure 11-10 represents the Control Descriptor L0_7 Register.

Offset:
0x018 (CtrlDescL0_7)
0x034 (CtrlDescL1_7)
0x050 (CtrlDescL2_7)
0x06C (CtrlDescL3_7)
0x088 (CtrlDescL4_7)
0x094 (CtrlDescL5_7)
0x0C0 (CtrlDescL6_7)
0x0DC (CtrlDescL7_7)
0x0F8 (CtrlDescL8_7)
0x114 (CtrlDescL9_7
0x130 (CtrlDescL10_7)
0x14C (CtrlDescL11_7)
0x168 (CtrlDescL12_7)
0x184 (CtrlDescL13_7)
0x19C (CtrlDescL14_7)
0x1BC (CtrlDescL15_7)

Access: User read/write

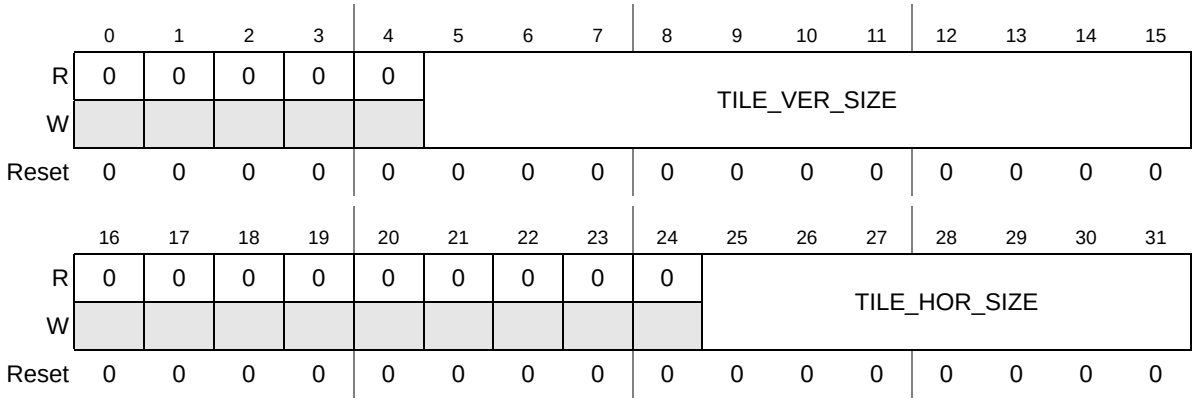



**Figure 11-10. Control Descriptor L0_7 Register**

**Table 11-12. Control Descriptor L0_7 Register**

| Field | Description |
|---|---|
| 6–15<br>TILE_VER_SIZE | Height of the TILE (in pixels) |
| 24–31<br>TILE_HOR_SIZE | Width of the TILE (in multiples of 16 pixels) |

For the other 16 layers, the Control Descriptor Register set is identical.

### 11.3.4.8 Control Descriptor Cursor 1 Register (CtrlDescCursor_1)

Figure 11-11 represents the Control Descriptor Cursor 1 register.

Offset: 0x1C0 Access: User read/write

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

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

**Figure 11-11. Control Descriptor Cursor 1 Register (CtrlDescCursor_1)**

**Table 11-13. CtrlDescCursor_1 Field Descriptions**

| Field | Description |
|---|---|
| 6–15<br>HEIGHT | Height of the cursor in pixels |
| 22–31<br>WIDTH | Width of the cursor in pixels |

### 11.3.4.9 Control Descriptor Cursor 2 Register (CtrlDescCursor_2)

Figure 11-12 represents the Control descriptor Cursor 2 register.

Offset: 0x1C4 Access: User read/write

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

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

**Figure 11-12. Control Descriptor Cursor 2 Register (CtrlDescCursor_2)**

**Table 11-14. CtrlDescCursor_2 Field Descriptions**

| Field | Description |
|---|---|
| POSY | Y position of the cursor in pixels |
| POSX | X position of the cursor in pixels |

### 11.3.4.10 Control Descriptor Cursor 3 Register (CtrlDescCursor_3)

Figure 11-13 represents the Control Descriptor Cursor 3 register.

Offset: 0x1C8 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CUR_EN | | | | | | | | DEFAULT_CURSOR_COLOR[0:8] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEFAULT_CURSOR_COLOR[9:24] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-13. Control Descriptor Cursor 3 Register (CtrlDescCursor_3)**

**Table 11-15. Control Descriptor Cursor_3 Field Descriptions**

| Field | Description |
|---|---|
| CUR_EN | Cursor Enable signal<br>1'b1: Enable the cursor<br>1'b0: Cursor is disabled |
| DEFAULT_CURSOR_COLOR | Default pixel color value for the cursor. In the DCU3, the pixel value for the cursor is fixed for a particular frame. |

### 11.3.4.11 Control Descriptor Cursor 4 Register (CtrlDescCursor_4)

Figure 11-14 represents the Control Descriptor Cursor 4 register.

Offset: 0x1CC Access: User read/write

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

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

**Figure 11-14. Control Descriptor Cursor 4 Register (CtrlDescCursor_4)**

**Table 11-16. CtrlDescCursor_4 Field Descriptions**

| Field | Description |
|---|---|
| HWC_BLINK_OFF | HWC blink register. Loads the counter value (number of frames) for which the cursor will remain turned OFF. |
| EN_BLINK | Enable the cursor blink mode.<br>1'b1:Enable the blink mode<br>1'b0:Disable the blink mode |
| HWC_BLINK_ON | HWC blink register. Loads the counter value (number of frames) for which the cursor will remain turned ON. |

### 11.3.4.12 DCU3 Mode Register (DCU_MODE)

Figure 11-15 represents the DCU_MODE register. This register sets the mode in which DCU3 is operating.

Offset: 0x1D0 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DCU_SW_RESET | DITHER_EN | ADDB | | ADDG | | ADDR | | DDR_MODE | BLEND_ITER | | | PDI_SYNC_LOCK | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDI_INTERPOL_EN | RASTER_EN | PDI_EN | PDI_BYTE_REV | PDI_DE_MODE | PDI_NARROW_MODE | PDI_MODE | | PDI_SLAVE_MODE | TAG_EN | SIG_EN | PDI_SYNC | 0 | EN_GAMMA | DCU_MODE | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-15. DCU3 Mode Register (DCU_MODE)**

**Table 11-17. DCU_MODE field descriptions**

| Field | Description |
|---|---|
| DCU_SW_RESET | Used to clear all the registers to reset state<br>1'b1:All the DCU3 registers are put in reset state |
| DITHER_EN | Enable dithering mode<br>1'b1:Enabled.<br>1'b0:Disabled |

**Table 11-17. DCU_MODE field descriptions (continued)**

| Field | Description |
|---|---|
| ADDB | Two-bit value to be added to pixel blue component for dithering |
| ADDG | Two-bit value to be added to pixel green component for dithering |
| ADDR | Two-bit value to be added to pixel red component for dithering |
| DDR_MODE | Enables Special DDR Mode (see Section 11.4.9, “Special DDR mode””). Select BLEND_ITER = 4 when enabled. |
| BLEND_ITER | Defines the number of planes used for blending.<br>3’d4:Four plane blending<br>3’d3:Three plane blending<br>3’d2:Two plane blending<br>default: Two plane blending |
| PDI_SYNC_LOCK | Defines the number of frames which should be received by the PDI validation state machine before it locks and sets the PDI_LOCK_DET bit in the PDI Status Register (see Section 11.3.4.26, “PDI Status Register””) |
| PDI_INTERPOL_EN | Control Bit to decide whether the conversion from YCbCr 4:2:2 to 4:4:4 needs to be done using interpolation or Chroma value is same for two pixels<br>1’b1:Interpolation is enabled.<br>1’b0:Chroma value is same for two pixels |
| RASTER_EN | Enables raster scanning of pixel data including the VSYNC and HSYNC signals and the pixel data. Changes to this bit take effect after the completion of the current frame.<br>1’b1: Enabled<br>1’b0:Disabled |
| PDI_EN | Enables the PDI<br>1’b1: Enabled<br>1’b0:Disabled |
| PDI_BYTE_REV | Controls the byte ordering in Narrow Mode<br>1’b0:LSB is followed by MSB data<br>1’b1:MSB is followed by LSB data |
| PDI_DE_MODE | Enables the PDI data Enable Mode. Here Data Enable is treated as an input.<br>1’b0: Value on data Enable signal is ignored<br>1’b1: Data enable signal must be present in incoming stream |
| PDI_NARROW_MODE | Enables the PDI Narrow Mode (refer to Section 11.8.1.5, “Normal and Narrow Mode””)<br>1’b0: Narrow Mode is Disabled<br>1’b1: Narrow Mode is Enabled |
| PDI_MODE | Defines the different modes in which PDI is operating<br>2’b00: 8 bit monochrome data input<br>2’b01: 16 bit RGB 565 format<br>2’b10:18 bit RGB 666 data format.<br>2’b11:YCbCr data in 4:2:2 format. |
| PDI_SLAVE_MODE | Enables PDI slave Mode<br>1’b0:Disabled<br>1’b1:Enabled |

**Table 11-17. DCU_MODE field descriptions (continued)**

| Field | Description |
|---|---|
| TAG_EN | Enables the calculation of CRC only on the safety layers<br>1'b0: CRC calculated over the whole area of interest (area of interest given by SIG_DESC registers)<br>1'b1: calculates CRC only on safety enabled layers |
| SIG_EN | Enables the signature calculator block<br>1'b0: signature calculator is disabled<br>1'b1: signature calculator is enabled |
| PDI_SYNC | Decides whether the PDI uses external or internal synchronization.<br>1'b0: External Synchronization. The PDI receives the SYNC (HSYNC, VSYNC) signals from external source.<br>1'b1: Internal Synchronization. PDI extracts the SYNC information from the digital data.<br>**Note:** YCbCr Mode supports Internal Sync only. Therefore, when PDI_MODE = 3, PDI_SYNC must be set to 0. |
| EN_GAMMA | Enables/Disables the Gamma Correction<br>1'b0: Gamma correction is disabled<br>1'b1: Gamma Correction is enabled |
| DCU_MODE | DCU3 operating mode<br>2'b00: DCU3 off (pixel clock active if enabled by I/O)<br>2'b01: Normal mode. Panel content controlled by layer configuration.<br>2'b10: Test mode. DCU3 disables all DMA fetches and all the pixels of an enabled layer take the value in the CLUT RAM selected by the respective LUOFFS field of control descriptor 4.<br>2'b11: Color Bar Generation. Panel content controlled by color bar registers. |

### 11.3.4.13 BGND Register

Figure 11-16 represents the BGND register.

Offset: 0x1D4 Access: User read/write

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

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

**Figure 11-16. BGND Register**

**Table 11-18. BGND Field Descriptions**

| Field | Description |
|---|---|
| 8–15<br>BGND_R | Red component of the default color displayed in the sectors where no layer is active |
| 16–23<br>BGND_G | Green component of the default color displayed in the sectors where no layer is active |
| 24–31<br>BGND_B | Blue component of the default color displayed in the sectors where no layer is active |

### 11.3.4.14 DISP_SIZE Register

Figure 11-17 represents the DISP_SIZE register

Offset: 0x1D8 Access: User read/write

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

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

**Figure 11-17. DISP_SIZE Register**

**Table 11-19. DISP_SIZE Field Descriptions**

| Field | Description |
|---|---|
| 6–15<br>DELTA_Y | Sets the display size vertical resolution (in pixels) |
| 24–31<br>DELTA_X | Sets the display size horizontal resolution (in multiples of 16 pixels) |

### 11.3.4.15 HSYN_PARA Register

Figure 11-18 represents the HSYN_PARA register. HSYN_PARA register sets timing parameters related to the horizontal synchronization signal generation. The fields FP_H, BP_H, and PW_H stand for HSYNC signal front-porch, back-porch, and active pulse width, respectively.

Offset: 0x1DC Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | BP_H | | | | | | | | | 0 | 0 | PW_H[0:3] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PW_H[4:8] | | | | | 0 | 0 | FP_H | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**Figure 11-18. HSYN_PARA Register**

**Table 11-20. HSYN_PARA Field Descriptions**

| Field | Description |
|---|---|
| 1–9<br>BP_H | HSYNC back-porch pulse width (in pixel clock cycles). Pulse width has a minimum value of 1. |
| 12–20<br>PW_H | HSYNC active pulse width (in pixel clock cycles). |
| 23–31<br>FP_H | HSYNC front-porch pulse width (in pixel clock cycles). Pulse width has a minimum value of 1. |

### 11.3.4.16 VSYN_PARA Register

Figure 11-19 represents the VSYN_PARA register. VSYN_PARA register sets timing parameters related to the vertical synchronization signal generation. The fields FP_V, BP_V, and PW_V stand for VSYNC signal front-porch, back-porch, and active pulse width, respectively.

Offset: 0x1E0 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | BP_V | | | | | | | | | 0 | 0 | PW_V[0:3] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PW_V[4:8] | | | | | 0 | 0 | FP_V | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**Figure 11-19. VSYN_PARA Register**

**Table 11-21. VSYN_PARA Field Descriptions**

| Field | Description |
|---|---|
| 1–9<br>BP_V | VSYNC back-porch pulse width (in horizontal line cycles). Pulse width has a minimum value of 1. |
| 12–20<br>PW_V | VSYNC active pulse width (in horizontal line cycles). |
| 23–31<br>FP_V | VSYNC front-porch pulse width (in horizontal line cycles). Pulse width has a minimum value of 1. |

### 11.3.4.17 SYN_POL Register

Figure 11-20 represents the SYN_POL register. SYN_POL register selects polarity for corresponding synchronize signals (HSYNC, VSYNC, CSYNC), and controls the bypass of HSYNC or VSYNC with CSYNC signal.

Offset: 0x1E4 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | |
| W | | | | | | INV_PDI_DE | INV_PDI_HS | INV_PDI_VS | INV_PDI_CLK | INV_PXCK | NEG | BP_VS | BP_HS | INV_CS | INV_VS | INV_HS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-20. SYN_POL Register**

**Table 11-22. SYN_POL Field Descriptions**

| Field | Description |
|---|---|
| INV_PDI_DE | Polarity change of PDI input data Enable.<br>1’b0: DE is active high<br>1’b1: DE is active low |
| INV_PDI_HS | Polarity change of PDI input HSYNC.<br>1’b0: HSYNC is active high<br>1’b1: HSYNC is active low |
| INV_PDI_VS | Polarity change of PDI input VSYNC.<br>1’b0: VSYNC is active high<br>1’b1: VSYNC is active low |

**Table 11-22. SYN_POL Field Descriptions (continued)**

| Field | Description |
|---|---|
| INV_PDI_CLK | Polarity change of PDI input Clock.<br>1'b0: DCU3 samples data on the rising edge<br>1'b1: DCU3 samples data on the falling edge |
| INV_PXCK | Polarity change of Pixel Clock.<br>1'b0: Display samples data on the falling edge<br>1'b1: Display samples data on the rising edge |
| NEG | Indicates if value at the output (pixel data output) needs to be negated.<br>1'b0: Output is to remain same<br>1'b1: Output to be negated |
| BP_VS | Bypass Vertical Synchronize Signal (internal pin muxing).<br>1'b0: Do not bypass VSYNC signal output<br>1 'b1: CSYNC bypass VSYNC signal, output CSYNC instead of VSYNC |
| BP_HS | Bypass Horizontal Synchronize Signal (internal pin muxing).<br>1'b0: Do not bypass HSYNC signal output<br>1'b1: CSYNC bypass HSYNC signal, output CSYNC instead of HSYNC |
| INV_CS | Invert Composite Synchronize Signal.<br>1'b0: Not invert CSYNC signal, active HIGH<br>1 'b1: Invert CSYNC signal, active LOW |
| INV_VS | Invert Vertical Synchronize Signal<br>1'b0: Not invert VSYNC signal, active HIGH<br>1 'b1: Invert VSYNC signal, active LOW |
| INV_HS | Invert Horizontal Synchronize Signal.<br>1'b0: Not invert HSYNC signal, active HIGH<br>1'b1: Invert HSYNC signal, active LOW |

### 11.3.4.18 Threshold Register

Figure 11-21 represents the Threshold Register.

Offset: 0x1E8 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OUT_BUF_HIGH | | | | | | | | OUT_BUF_LOW | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

**Figure 11-21. Threshold Register**

**Table 11-23. Threshold Register Field Descriptions**

| Field | Description |
|---|---|
| 6–15<br>LS_BF_VS | Lines before VSYNC threshold value. The LS_BF_VS status flag (in INT_STATUS) is set this number of lines before the VSYNC signal is asserted. |
| 16–23<br>OUT_BUF_HIGH | Output buffer high threshold (in pixels). When the output buffer exceeds this value the datapath clock is suspended. |
| 24–31<br>OUT_BUF_LOW | Output buffer filling low Threshold (in pixels).This value is used to generate the underrun exception (UNDRUN in INT_STATUS). |

### 11.3.4.19 Interrupt Status Register (INT_STATUS)

Figure 11-22 indicates the interrupt status register. See Section 11.5.4, “Interrupt generation”,” for a description of how the DCU3 collects interrupt events into different source groups.

Offset: 0x1EC Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | P4_FIFO_HI_FLAG | P4_FIFO_LO_FLAG | P3_FIFO_HI_FLAG | P3_FIFO_LO_FLAG |
| W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMA_TRANS_FINISH | 0 | 0 | IPM_ERROR | PROG_END | P2_FIFO_HI_FLAG | P2_FIFO_LO_FLAG | P1_FIFO_HI_FLAG | P1_FIFO_LO_FLAG | CRC_OVERFLOW | CRC_READY | VS_BLANK | LS_BF_VS | UNDRUN | VSYNC |
| W | | w1c | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-22. Interrupt Status Register (INT_STATUS)**

**Table 11-24. INT_STATUS Field Descriptions**

| Field | Description |
|---|---|
| 12<br>P4_FIFO_HI_FLAG | Interrupt signal to indicate that High threshold has been reached for plane 4(FG2plane) input buffer |
| 13<br>P4_FIFO_LO_FLAG | Interrupt signal to indicate that Low threshold has been reached for plane 4(FG2plane) input buffer |
| 14<br>P3_FIFO_HI_FLAG | Interrupt signal to indicate that High threshold has been reached for plane 3(FG1plane) input buffer |
| 15<br>P3_FIFO_LO_FLAG | Interrupt signal to indicate that Low threshold has been reached for plane 3(FG1plane) input buffer |
| 17<br>DMA_TRANS_FINISH | Interrupt signal which indicates that the DCU3 DMA has fetched the last pixel of data from the memory |
| 20<br>IPM_ERROR | Interrupt flag, which indicates that an error response has been received by the IP from the AHB bus |
| 21<br>PROG_END | Interrupt signal which indicates that the duration for programming of DCU3 registers and internal memories is finished |
| 22<br>P2_FIFO_HI_FLAG | Interrupt signal to indicate that High threshold has been reached for plane 2 (FGplane) input buffer |
| 23<br>P2_FIFO_LO_FLAG | Interrupt signal to indicate that Low threshold has been reached for plane 2 (FGplane) input buffer |
| 24<br>P1_FIFO_HI_FLAG | Interrupt signal to indicate that High threshold has been reached for plane 1 (BGplane) input buffer |
| 25<br>P1_FIFO_LO_FLAG | Interrupt signal to indicate that Low threshold has been reached for plane 1 (BGplane) input buffer |
| 26<br>CRC_OVERFLOW | Interrupt signal to indicate that CRC_ready has not been serviced and CRC has been calculated for the next frame |
| 27<br>CRC_READY | Interrupt signal to indicate CRC calculation is done and ready to be compared with precomputed CRC value by the software |
| 28<br>VS_BLANK | Interrupt signal to indicate vertical blanking period. This is the period in which all the registers that affect the visible state of the layers need to be latched. This is needed so that CPU writes to the register while the display is being updated does not cause any errors. |
| 29<br>LS_BF_VS | Lines Before Vsync interrupt. It is generated threshold LS_BF_VS number of lines ahead of the vertical front porch (FP_V) if enabled. The CPU can program the registers after LS_BF_VS interrupt. |
| 30<br>UNDRUN | Under Run Exception Interrupt. Asserted when display needs data and output buffer filling is lower than or equal to the OUT_BUF_LOW threshold. Interrupt is cleared when the data in the output buffer is greater than threshold and CPU writes 1 to this bit. |
| 31<br>VSYNC | Vertical Synchronize Interrupt. If enabled, an interrupt is generated at the beginning of a frame. |

### 11.3.4.20 Interrupt Mask Register (INT_MASK)

Figure 11-23 represents the interrupt mask register.This register enables or masks corresponding interrupt.

Offset: 0x1F0 Access: User read/write

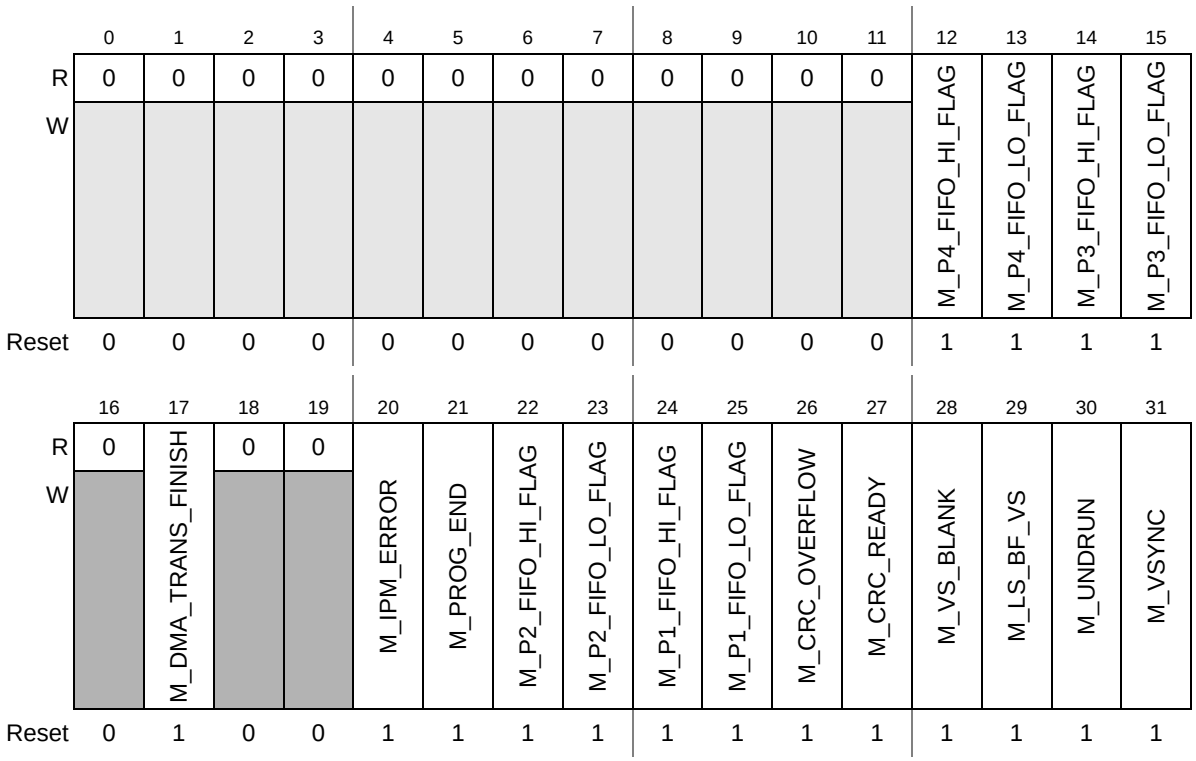



**Figure 11-23. Interrupt Mask Register (INT_MASK)**

**Table 11-25. INT_MASK Field Descriptions**

| Field | Description |
|---|---|
| 12<br>M_P4_FIFO_HI_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0: Not masked |
| 13<br>M_P4_FIFO_LO_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0: Not masked |
| 14<br>M_P3_FIFO_HI_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0: Not masked |
| 15<br>M_P3_FIFO_LO_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0: Not masked |
| 17<br>M_DMA_TRANS_FINISH | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 20<br>M_IPM_ERROR | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |

Table 11-25. INT_MASK Field Descriptions (continued)

| Field | Description |
|---|---|
| 21<br>M_PROG_END | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 22<br>M_P2_FIFO_HI_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 23<br>M_P2_FIFO_LO_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 24<br>M_P1_FIFO_HI_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 25<br>M_P1_FIFO_LO_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 26<br>M_CRC_OVERFLOW | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 27<br>M_CRC_READY | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 28<br>M_VS_BLANK | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 29<br>M_LS_BF_VS | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 30<br>M_UNDRUN | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| 31<br>M_VSYNC | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |

### 11.3.4.21 COLBAR Registers

The COLBAR registers are used to generate color bars in functional test mode. Eight different pixel values are taken as input data, to display 8 color bars on the display.

Table 11-26. COLBAR_*n* register field descriptions

| Field Name | Description |
|---|---|
| COLBAR_*n*_R | Red component value |

**Table 11-26. COLBAR_*n* register field descriptions**

| Field Name | Description |
|---|---|
| COLBAR_*n*_G | Green component value |
| COLBAR_*n*_B | Blue component value |

### 11.3.4.21.1 COLBAR_1 register

Offset: 0x1F4 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_1_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_1_G | | | | | | | | COLBAR_1_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-24. COLBAR_1 Register (Black)**

### 11.3.4.21.2 COLBAR_2 Register

Offset: 0x1F8 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_2_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_2_G | | | | | | | | COLBAR_2_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 11-25. COLBAR_2 Register (Blue)**

### 11.3.4.21.3 COLBAR_3 Register

Offset: 0x1FC

Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_3_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_3_G | | | | | | | | COLBAR_3_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 11-26. COLBAR_3 Register (Cyan)**

### 11.3.4.21.4 COLBAR_4 Register

Offset: 0x200

Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_4_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_4_G | | | | | | | | COLBAR_4_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-27. COLBAR_4 Register (Green)**

### 11.3.4.21.5 COLBAR_5 Register

Offset: 0x204 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_5_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_5_G | | | | | | | | COLBAR_5_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-28. COLBAR_5 Register (Yellow)**

### 11.3.4.21.6 COLBAR_6 Register

Offset: 0x208 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_6_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_6_G | | | | | | | | COLBAR_6_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-29. COLBAR_6 Register (Red)**

### 11.3.4.21.7 COLBAR_7 Register

Offset: 0x20C

Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_7_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_7_G | | | | | | | | COLBAR_7_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 11-30. COLBAR_7 Register (Purple)**

### 11.3.4.21.8 COLBAR_8 Register

Offset: 0x210

Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_8_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_8_G | | | | | | | | COLBAR_8_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 11-31. COLBAR_8 Register (White)**

## 11.3.4.22 Divide Ratio Register (DIV_RATIO)

Figure 11-32 shows the Divide Ratio Register.

Offset: 0x214 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DIV_RATIO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**Figure 11-32. Divide Ratio Register (DIV_RATIO)**

**Table 11-27. DIV_RATIO Field Descriptions**

| Field | Description |
|---|---|
| 24–31<br>DIV_RATIO | Specifies the divide value for the input clock. Used to generate the pixel clock to support different types of displays. To divide by N, set the DIV_RATIO to (N-1). |

### 11.3.4.23 SIGN_CALC_1 Register

Figure 11-33 presents the register for vertical/horizontal size of the area for CRC calculation.

Offset: 0x218 Access: User read/write

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

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

**Figure 11-33. SIGN_CALC_1 Register**

**Table 11-28. SIGN_CALC_1 Field Descriptions**

| Field | Description |
|---|---|
| 6–15<br>SIG_VER_SIZE | Vertical size of the window of interest of pixels for CRC calculation (in pixels) |
| 22–31<br>SIG_HOR_SIZE | Horizontal size of window of interest of pixels for CRC calculations (in pixels) |

### 11.3.4.24 SIGN_CALC_2 Register

Figure 11-34 represents the register for position of the window of interest for CRC calculation.

Offset: 0x21C Access: User read/write

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

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

**Figure 11-34. SIGN_CALC_2 Register**

**Table 11-29. SIGN_CALC_2 Field Descriptions**

| Field | Description |
|---|---|
| 6–15<br>SIG_VER_POS | Vertical position of the window of interest of pixels for CRC calculation (in pixels) |
| 22–31<br>SIG_HOR_POS | Horizontal position of window of interest of pixels for CRC calculation (in pixels) |

### 11.3.4.25 CRC_VAL Register

Figure 11-35 represents the register presenting the CRC value to the software for comparison.

Offset: 0x220 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_VAL[0:15] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_VAL[16:31] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-35. CRC_VAL Register**

**Table 11-30. CRC_VAL Field Descriptions**

| Field | Description |
|---|---|
| 0–31<br>CRC_VAL | CRC value calculated for safety enabled layers to be presented to the software for comparison. |

### 11.3.4.26 PDI Status Register

Figure 11-36 represents the PDI status register.

Offset: 0x224 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | PDI_BLANKING_ERR | PDI_ECC_ERR2 | PDI_ECC_ERR1 | PDI_LOCK_LOST | PDI_LOCK_DET | PDI_VSYNC_DET | PDI_HSYNC_DET | PDI_DE_DET | PDI_CLK_LOST | PDI_CLK_DET |
| W | | | | | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-36. PDI Status Register (PDI_STATUS)**

**Table 11-31. PDI_STATUS field descriptions**

| Field | Description |
|---|---|
| 22<br>PDI_BLANKING_ERR | Status bit to inform the software that 80h,10h sequence is not present during the blanking period in internal sync mode.<br>1'b1:Correct data sequence not present in blanking period<br>1'b0:Correct data sequence present in blanking period |
| 23<br>PDI_ECC_ERR2 | Status bit to inform the software about multibit bit error that is detected.<br>1'b1: Multibit ECC error detected<br>1'b0: Multibit ECC error is not detected |
| 24<br>PDI_ECC_ERR1 | Status bit to inform the software about one bit error is detected.<br>1'b1: One bit ECC error detected<br>1'b0: One bit ECC error is not detected |
| 25<br>PDI_LOCK_LOST | Status bit to inform the software that frame lock is lost.<br>1'b1: Frame lock is lost<br>1'b0: Frame is locked |

**Table 11-31. PDI_STATUS field descriptions (continued)**

| Field | Description |
|---|---|
| 26<br>PDI_LOCK_DET | Status bit to inform the software PDI is frame locked to the camera interface.<br>1'b1: Frame lock is detected<br>1'b0: Waiting for frame to lock |
| 27<br>PDI_VSYNC_DET | Status bit to inform the software that vsync for the camera data has been detected.<br>1'b1: pdi_vsync is detected<br>1'b0: pdi_vsync not detected |
| 28<br>PDI_HSYNC_DET | Status bit to inform the software that hsync for the camera data has been detected.<br>1'b1: pdi_hsync is detected<br>1'b0: pdi_hsync not detected |
| 29<br>PDI_DE_DET | Status bit to inform the software that data Enable for the camera data has been detected.<br>1'b1: pdi_de is detected<br>1'b0: pdi_de not detected |
| 30<br>PDI_CLK_LOST | Status bit to inform the software that pdi_clk is lost<br>1'b1: pdi_clk is lost<br>1'b0: pdi_clk is present |
| 31<br>PDI_CLK_DET | Status bit to inform the software that clock for the camera data has been detected.<br>1'b1: pdi_clk is detected<br>1'b0: pdi_clk not detected |

## 11.3.4.27 PDI Status Mask Register

Figure 11-37 represents the Mask PDI status register

Offset: 0x228 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | M_PDI_BLANKING_ERR | M_PDI_ECC_ERR2 | M_PDI_ECC_ERR1 | M_PDI_LOCK_LOST | M_PDI_LOCK_DET | M_PDI_VSYNC_DET | M_PDI_HSYNC_DET | M_PDI_DE_DET | M_PDI_CLK_LOST | M_PDI_CLK_DET |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 11-37. PDI status mask register**

Table 11-32. PDI Status Mask Register Field Descriptions

| Field | Description |
|---|---|
| M_PDI_BLANKING_ERR | Mask the PDI_BLANKING_ERR bit<br>1'b1: Mask the PDI_BLANKING_ERR interrupt<br>1'b0: Do not mask the PDI_BLANKING_ERR interrupt |
| M_PDI_ECC_ERR2 | Mask the PDI_ECC_ERR2 bit<br>1'b1: Mask the PDI_ECC_ERR2 interrupt<br>1'b0: Do not mask the PDI_ECC_ERR2 interrupt |
| M_PDI_ECC_ERR1 | Mask the PDI_ECC_ERR1 bit<br>1'b1: Mask the PDI_ECC_ERR1 interrupt<br>1'b0: Do not mask the PDI_ECC_ERR1 interrupt |
| M_PDI_LOCK_LOST | Mask the PDI_LOCK_LOST bit<br>1'b1: Mask the PDI_LOCK_LOST interrupt<br>1'b0: Do not mask the PDI_LOCK_LOST interrupt |
| M_PDI_LOCK_DET | Mask the PDI_LOCK_DET bit<br>1'b1: Mask the PDI_LOCK_DET interrupt<br>1'b0: Do not mask the PDI_LOCK_DET interrupt |
| M_PDI_VSYNC_DET | Mask the PDI_VSYNC_DET bit<br>1'b1: Mask the PDI_VSYNC_DET interrupt<br>1'b0: Do not mask the PDI_VSYNC_DET interrupt |
| M_PDI_HSYNC_DET | Mask the PDI_HSYNC_DET bit<br>1'b1: Mask the PDI_HSYNC_DET interrupt<br>1'b0: Do not mask the PDI_HSYNC_DET interrupt |
| M_PDI_DE_DET | Mask the PDI_DE_DET bit<br>1'b1: Mask the PDI_DE_DET interrupt<br>1'b0: Do not mask the PDI_DE_DET interrupt |
| M_PDI_CLK_LOST | Mask the PDI_CLK_LOST bit<br>1'b1: Mask the PDI_CLK_LOST interrupt<br>1'b0: Do not mask the PDI_CLK_LOST interrupt |
| M_PDI_CLK_DET | Mask the PDI_CLK_DET bit<br>1'b1: Mask the PDI_CLK interrupt<br>1'b0: Do not mask the PDI_CLK interrupt |

### 11.3.4.28 PARR_ERR_STATUS Register

Figure 11-38 shows the parameter error status register.

An error in a layer can occur under the following conditions:

a) Number of pixels in a tile > maximum tile memory size in case of Tile bandwidth optimized mode (when in internal memory mode)

b) There is an automatic error checking mechanism when a layer is enabled that detects a non-valid horizontal size and color format combination. See Section 11.4.4.3, "Layer size and positioning"," for details.

These errors are grouped into a single bit error for each layer. The parameter error specific to each layer is signalled only when the layer is enabled.

RLE_ERR occurs when more than one layer has its RLE_EN bit set (Control Descriptor 4).

DISP_ERR occurs when the size of display (height or width) is set to zero or when the pulse width of HSYNC/VSYNC is programmed as zero.

SIG_ERR occurs when the area of interest for calculating CRC value is programmed with values which are outside the display.

HWC_ERR occurs if size of cursor programmed is greater than memory size(256x32).See Section 11.4.5, “Hardware cursor”,” for further details on how cursor can be programmed.

Offset: 0x22C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RLE_ERR | HWC_ERR | SIG_ERR | DISP_ERR |
| W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | L15_PARR_ERR | L14_PARR_ERR | L13_PARR_ERR | L12_PARR_ERR | L11_PARR_ERR | L10_PARR_ERR | L9_PARR_ERR | L8_PARR_ERR | L7_PARR_ERR | L6_PARR_ERR | L5_PARR_ERR | L4_PARR_ERR | L3_PARR_ERR | L2_PARR_ERR | L1_PARR_ERR | L0_PARR_ERR |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-38. Parameter Error Status Register (PARR_ERR_STATUS)**

**Table 11-33. PARR_ERR_STATUS field descriptions**

| Field | Description |
|---|---|
| 13<br>RLE_ERR | Error signal to indicate that more than one layer has RLE mode enabled. |
| 13<br>HWC_ERR | Interrupt signal to indicate HWC error. This can occur if HWC position is out of display area or cursor memory is bigger than the HWC size. When this occurs, the HWC is disabled. |
| 14<br>SIG_ERR | Interrupt occurs whenever the area of interest specified by SIG_CALC register is outside the display size.<br>1’b0: SIG_ERR is not set<br>1’b1: SIG_ERR is set |
| 15<br>DISP_ERR | Interrupt occurs whenever width and height of display, pulse width (both vertical and horizontal sync) value is 0.<br>1’b0: DISP_ERR is not set<br>1’b1: DISP_ERR is set |

**Table 11-33. PARR_ERR_STATUS field descriptions (continued)**

| Field | Description |
|---|---|
| 13<br>RLE_ERR | Error signal to indicate that more than one layer has RLE mode enabled. |
| 16<br>L15_PARR_ERR | Interrupt occurs whenever there is an error in layer 15.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 17<br>L14_PARR_ERR | Interrupt occurs whenever there is an error in layer 14.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 18<br>L13_PARR_ERR | Interrupt occurs whenever there is an error in layer 13.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 19<br>L12_PARR_ERR | Interrupt occurs whenever there is an error in layer 12.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 20<br>L11_PARR_ERR | Interrupt occurs whenever there is an error in layer 11.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 21<br>L10_PARR_ERR | Interrupt occurs whenever there is an error in layer 10.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 22<br>L9_PARR_ERR | Interrupt occurs whenever there is an error in layer 9<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 23<br>L08_PARR_ERR | Interrupt occurs whenever there is an error in layer 8.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 24<br>L7_PARR_ERR | Interrupt occurs whenever there is an error in layer 7<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 25<br>L6_PARR_ERR | Interrupt occurs whenever there is an error in layer 6.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 26<br>L5_PARR_ERR | Interrupt occurs whenever there is an error in layer 5.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 27<br>L4_PARR_ERR | Interrupt occurs whenever there is an error in layer 4<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 28<br>L3_PARR_ERR | Interrupt occurs whenever there is an error in layer 3.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 29<br>L2_PARR_ERR | Interrupt occurs whenever there is an error in layer 2.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |

**Table 11-33. PARR_ERR_STATUS field descriptions (continued)**

| Field | Description |
|---|---|
| 13<br>RLE_ERR | Error signal to indicate that more than one layer has RLE mode enabled. |
| 30<br>L1_PARR_ERR | Interrupt occurs whenever there is an error in layer 1.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| 31<br>L0_PARR_ERR | Interrupt occurs whenever there is an error in layer 0.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |

### 11.3.4.29 Mask PARR_ERR status register

Figure 11-39 shows the mask register for parameter error status register.

Offset: 0x230 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | M_RLE_ERR | M_HWC_ERR | M_SIG_ERR | M_DISP_ERR |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | M_L15_parr_err | M_L14_parr_err | M_L13_parr_err | M_L12_parr_err | M_L11_parr_err | M_L10_parr_err | M_L9_parr_err | M_L8_parr_err | M_L7_parr_err | M_L6_parr_err | M_L5_parr_err | M_L4_parr_err | M_L3_parr_err | M_L2_parr_err | M_L1_parr_err | M_L0_parr_err |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 11-39. Mask parameter error status register**

**Table 11-34. Mask parameter error status register field descriptions**

| Field | Description |
|---|---|
| 13<br>M_RLE_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 13<br>M_HWC_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 14<br>M_SIG_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |

**Table 11-34. Mask parameter error status register field descriptions (continued)**

| Field | Description |
|---|---|
| 13<br>M_RLE_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 15<br>M_DISP_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 16<br>M_L15_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 17<br>M_L14_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 18<br>M_L13_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 19<br>M_L12_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 20<br>M_L11_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 21<br>M_L10_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 22<br>M_L9_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 23<br>M_L8_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 24<br>M_L7_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 25<br>M_L6_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 26<br>M_L5_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 27<br>M_L4_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |

**Table 11-34. Mask parameter error status register field descriptions (continued)**

| Field | Description |
|---|---|
| 13<br>M_RLE_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 28<br>M_L3_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 29<br>M_L2_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 30<br>M_L1_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| 31<br>M_L0_parr_err | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |

## 11.3.4.30 THRESHOLD_INP_BUF_1 Register

Figure 11-40 shows the threshold register for input buffer.

Offset: 0x234 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | INP_BUF_P2_HI | | | | | | | 0 | INP_BUF_P2_LO | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | INP_BUF_P1_HI | | | | | | | 0 | INP_BUF_P1_LO | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-40. Threshold input buffer 1 Register (THRESHOLD_INP_BUF_1)**

**Table 11-35. THRESHOLD_INP_BUF_1 field descriptions**

| Field | Description |
|---|---|
| INP_BUF_P2_HI | High threshold for input buffer for blend stage 2. |
| INP_BUF_P2_LO | Low threshold for input buffer for blend stage 2. |
| INP_BUF_P1_HI | High threshold for input buffer for blend stage 1 (background). |
| INP_BUF_P1_LO | Low threshold for input buffer for blend stage 1 (background plane). |

### 11.3.4.31 THRESHOLD_INP_BUF_2 Register

Figure 11-41 represents the threshold register for input buffer for plane 3 and plane 4.

Offset: 0x238 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | INP_BUF_P4_HI | | | | | | | 0 | INP_BUF_P4_LO | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | INP_BUF_P3_HI | | | | | | | 0 | INP_BUF_P3_LO | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-41. THRESHOLD_INP_BUF_2 Register**

**Table 11-36. THRESHOLD_INP_BUF_2 field descriptions**

| Field | Description |
|---|---|
| INP_BUF_P4_HI | High threshold for input buffer for blend stage 4. |
| INP_BUF_P4_LO | Low threshold for input buffer for blend stage 4. |
| INP_BUF_P3_HI | High threshold for input buffer for blend stage 3. |
| INP_BUF_P3_LO | Low threshold for input buffer for blend stage 3. |

### 11.3.4.32 LUMA Component Register

Figure 11-42 represents the LUMA component register.

Offset: 0x23C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Y_RED | | | | | | | | | | 0 | Y_GREEN[0:4] | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Y_GREEN[5:9] | | | | | 0 | Y_BLUE | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

**Figure 11-42. LUMA Component Register**

**Table 11-37. LUMA Component Register Field Descriptions**

| Field | Description |
|---|---|
| 0–9<br>Y_RED | Luminance Coefficient for Red Matrix |
| 11–20<br>Y_GREEN | Luminance Coefficient for Green Matrix |
| 22–31<br>Y_BLUE | Luminance Coefficient for Blue Matrix |

### 11.3.4.33 Red Chroma Components

Figure 11-43 represents the Red Chroma component register.

Offset: 0x240 Access: User read/write

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

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

**Figure 11-43. RED Chroma Component Register**

**Table 11-38. RED Chroma Component Register Field Descriptions**

| Field | Description |
|---|---|
| 5–15<br>Cr_RED | Cr Coefficient for Red Matrix |
| 20–31<br>Cb_GREEN | Cb Coefficient for Red Matrix |

### 11.3.4.34 Green Chroma Component Register

Figure 11-44 represents the Green Chroma component register

Offset: 0x244 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | Cb_GREEN | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

**Figure 11-44. GREEN Chroma Component Register**

**Table 11-39. GREEN chroma Component Register Field Descriptions**

| Field | Description |
|---|---|
| 5–15<br>Cr_GREEN | Cr Coefficient for Green Matrix |
| 20–31<br>Cb_GREEN | Cb Coefficient for Green Matrix |

## 11.3.4.35 BLUE chroma Component Register

Figure 11-45 represents the Blue Chroma component register.

Offset: 0x248 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | Cb_BLUE | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 11-45. BLUE chroma component Register**

**Table 11-40. BLUE chroma component Register Field Descriptions**

| Field | Description |
|---|---|
| 5–15<br>Cr_BLUE | Cr Coefficient for Blue Matrix |
| 20–31<br>Cb_BLUE | Cb Coefficient for Blue Matrix |

### 11.3.4.36 CRC_POS Register

Figure 11-46 represents the CRC_POS register.

Offset: 0x24C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_POS[0:15] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CRC_POS[16:31] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-46. CRC_POS Register**

**Table 11-41. CRC_POS Field Descriptions**

| Field | Description |
|---|---|
| 0–31<br>CRC_POS | CRC position value calculated for safety enabled layers to be presented to the software for comparison |

### 11.3.4.37 FG0_FCOLOR Register

Figure 11-47 represents the FG0_fcolor register.

Offset:
0x250 (FG0_FCOLOR)
0x258 (FG1_FCOLOR)
0x260 (FG2_FCOLOR)
0x268 (FG3_FCOLOR)
0x270 (FG4_FCOLOR)
0x278 (FG5_FCOLOR))
0x280 (FG6_FCOLOR)
0x288 (FG7_FCOLOR)
0x290 (FG8_FCOLOR)
0x298 (FG9_FCOLOR)
0x2A0 (FG10_FCOLOR)
0x2A8 (FG11_FCOLOR)
0x2B0 (FG12_FCOLOR)
0x2B8 (FG13_FCOLOR)
0x2C0 (FG14_FCOLOR)
0x2C8 (FG15_FCOLOR)

Access: User read/write

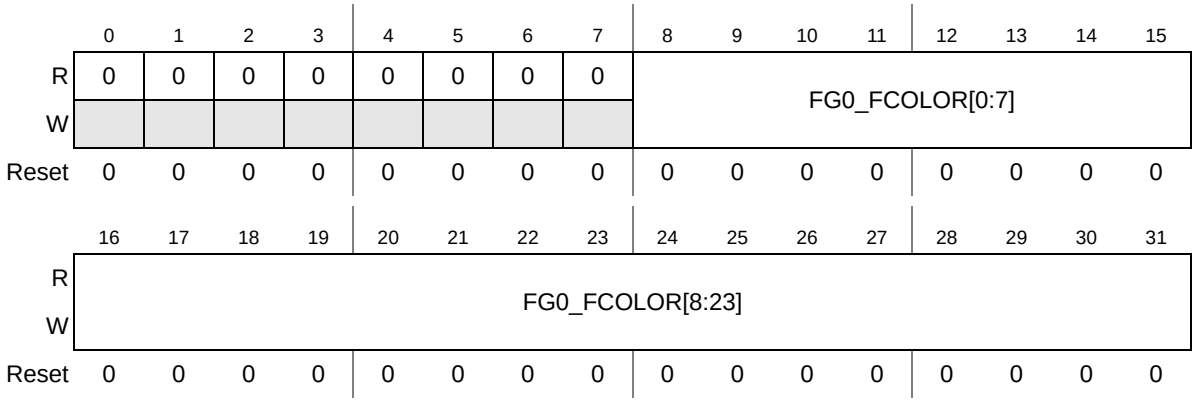



**Figure 11-47. FG0_fcolor Register**

**Table 11-42. FG0_fcolor Field Descriptions**

| Field | Description |
|---|---|
| 8–31<br>FG0_FCOLOR | Foreground color for layer FG0 for pre-blending engine |

## 11.3.4.38 FG0_bcolor

Figure 11-48 represents the FG0_bcolor register.

Offset:
0x254 (FG0_BCOLOR)
0x25C (FG1_BCOLOR)
0x264 (FG2_BCOLOR)
0x26C (FG3_BCOLOR)
0x274 (FG4_BCOLOR)
0x27C (FG5_BCOLOR))
0x284 (FG6_BCOLOR)
0x28C (FG7_BCOLOR)
0x294 (FG8_BCOLOR)
0x29C (FG9_BCOLOR)
0x2A4 (FG10_BCOLOR)
0x2AC (FG11_BCOLOR)
0x2B4 (FG12_BCOLOR)
0x2BC (FG13_BCOLOR)
0x2C4 (FG14_BCOLOR)
0x2CC (FG15_BCOLOR)

Access: User read/write

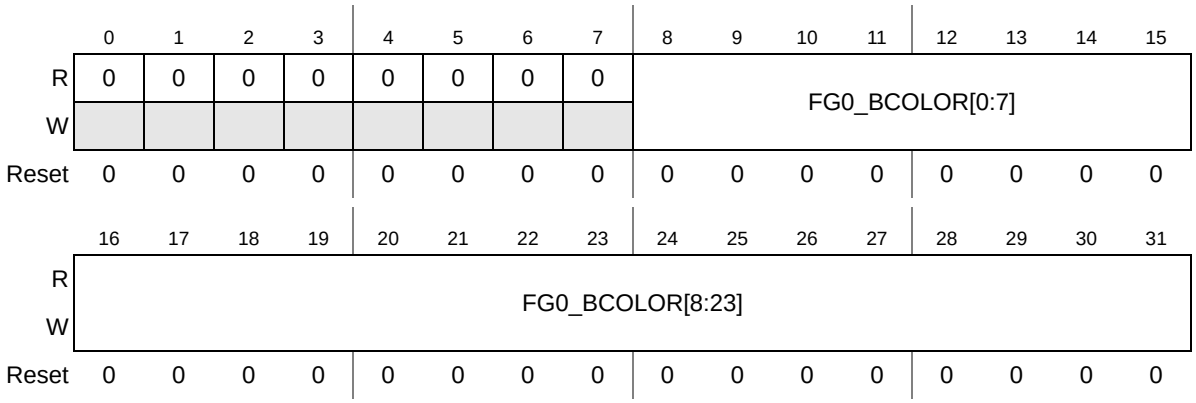



**Figure 11-48. FG0_bcolor Register**

**Table 11-43. FG0_color Register Field Description**

| Field | Description |
|---|---|
| 8–31<br>FG0_BCOLOR | Background color for layer FG0 for pre-blending engine |

### 11.3.4.39 LYR_INTPOL_EN

Figure 11-49 represents LYR_INTPOL_EN register.

Offset: 0x2D0 Access: User read/write

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

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

**Figure 11-49. LYR_INTPOL_EN register**

**Table 11-44. LYR_INTPOL_EN field descriptions**

| Field | Description |
|---|---|
| EN | Interpolation Enable bit for DCU3 Layer coded in YCbCr422 format.<br>This bit controls whether the chroma value for each pixel in the conversion from YCbCr 4:2:2 to 4:4:4 should use interpolation or the same value for both pixels.<br>0 Chroma value is same for two pixels<br>1 Interpolation is enabled. |

### 11.3.4.40 LYR_LUMA_COMPONENT

Figure 11-50 represents the Layer LUMA component register.

Offset: 0x2D4 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Y_RED | | | | | | | | | | 0 | Y_GREEN[0:4] | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Y_GREEN[5:9] | | | | | 0 | Y_BLUE | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

**Figure 11-50. Layer LUMA Component Register**

**Table 11-45. Layer LUMA Component Register Field Descriptions**

| Field | Description |
|---|---|
| 0–9<br>Y_RED | Luminance Coefficient for Red Matrix |
| 11–20<br>Y_GREEN | Luminance Coefficient for Green Matrix |
| 22–31<br>Y_BLUE | Luminance Coefficient for Blue Matrix |

## 11.3.4.41 LYR_CHROMA_RED

Figure 11-51 represents the Layer Red Chroma component register.

Offset: 0x2D8 Access: User read/write

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

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

**Figure 11-51. Layer RED Chroma Component Register**

**Table 11-46. Layer RED Chroma Component Register Field Descriptions**

| Field | Description |
|---|---|
| 5–15<br>Cr_RED | Cr Coefficient for Red Matrix |
| 20–31<br>Cb_GREEN | Cb Coefficient for Red Matrix |

## 11.3.4.42 LYR_CHROMA_GREEN

Figure 11-52 represents the Layer Green Chroma component register

Offset: 0x2DC Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | Cb_GREEN | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

**Figure 11-52. Layer GREEN Chroma Component Register**

**Table 11-47. Layer GREEN chroma Component Register Field Descriptions**

| Field | Description |
|---|---|
| 5–15<br>Cr_GREEN | Cr Coefficient for Green Matrix |
| 20–31<br>Cb_GREEN | Cb Coefficient for Green Matrix |

## 11.3.4.43 LYR_CHROMA_BLUE

Figure 11-53 represents the Layer Blue Chroma component register.

Offset: 0x2E0 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | Cb_BLUE | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 11-53. Layer BLUE chroma component Register**

**Table 11-48. Layer BLUE chroma component Register Field Descriptions**

| Field | Description |
|---|---|
| 5–15<br>Cr_BLUE | Cr Coefficient for Blue Matrix |
| 20–31<br>Cb_BLUE | Cb Coefficient for Blue Matrix |

### 11.3.4.44 COMP_IMSIZE

Figure 11-54 represents the Compression Image Size Register.

Offset: 0x2E4 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | COMP_IMSIZE[21:16] | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COMP_IMSIZE[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 11-54. COMP_IMSIZE Register**

**Table 11-49. COMP_IMSIZE Register Field Descriptions**

| Field | Description |
|---|---|
| COMP_IMSIZE | Compressed Image size in bytes for RLE coded layer |

### 11.3.4.45 Global Protection Register

Figure 11-55 represents the Global Protection register.

Offset: 0x300 Access: User read/write

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

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

**Figure 11-55. Global Protection Register**

**Table 11-50. Global Protection Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>HLB | Hard Lock Bit. This bit cannot be cleared once it is set by software. It can only be cleared by a system reset.<br>1'b1:All SLB's are write protected & cannot be modified<br>1'b0:All SLB's are accessible & can be modified |

## 11.3.4.46 Soft Lock Bit Register L0

Figure 11-56 represents the Soft Lock Bit Register for Layer0. This is used to protect the 7 control descriptor layer registers for Layer0.

Offset: 0x304 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | SLB_L0_1 | SLB_L0_2 | SLB_L0_3 | SLB_L0_4 | 0 | 0 | 0 | 0 | SLB_L0_5 | SLB_L0_6 | SLB_L0_7 | 0 |
| W | WEN_LO_1 | WEN_LO_2 | WEN_LO_3 | WEN_LO_4 | | | | | WEN_LO_5 | WEN_LO_6 | WEN_LO_7 | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 11-56. Soft Lock Register L0**

**Table 11-51. Soft Lock Register L0 Field Descriptions**

| Field | Description |
|---|---|
| 0<br>WEN_L0_1 | Write Enable for Soft Lock Bit SLB_L0_1<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 1<br>WEN_L0_2 | Write Enable for Soft Lock Bit SLB_L0_2<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 2<br>WEN_L0_3 | Write Enable for Soft Lock Bit SLB_L0_3<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 3<br>WEN_L0_4 | Write Enable for Soft Lock Bit SLB_L0_4<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 4<br>SLB_L0_1 | Soft Lock Bit for Control Desc L0_1 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 5<br>SLB_L0_2 | Soft Lock Bit for Control Desc L0_2 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 6<br>SLB_L0_3 | Soft Lock Bit for Control Desc L0_3 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 7<br>SLB_L0_4 | Soft Lock Bit for Control Desc L0_4 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 8<br>WEN_L0_5 | Write Enable for Soft Lock Bit SLB_L0_5<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 9<br>WEN_L0_6 | Write Enable for Soft Lock Bit SLB_L0_6<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 10<br>WEN_L0_7 | Write Enable for Soft Lock Bit SLB_L0_7<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 12<br>SLB_L0_5 | Soft Lock Bit for Control Desc L0_5 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 13<br>SLB_L0_6 | Soft Lock Bit for Control Desc L0_6 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 14<br>SLB_L0_7 | Soft Lock Bit for Control Desc L0_7 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

### 11.3.4.47 Soft Lock Bit Register L1

Figure 11-57 represents the Soft Lock Bit Register for Layer1. This is used to protect the 7 control descriptor layer registers for Layer1.

Offset: 0x308 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | SLB_L1_1 | SLB_L1_2 | SLB_L1_3 | SLB_L1_4 | 0 | 0 | 0 | 0 | SLB_L1_5 | SLB_L1_6 | SLB_L1_7 | 0 |
| W | WEN_L1_1 | WEN_L1_2 | WEN_L1_3 | WEN_L1_4 | | | | | WEN_L1_5 | WEN_L1_6 | WEN_L1_7 | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 11-57. Soft Lock Register L1**

**Table 11-52. Soft Lock Register L1 Field Descriptions**

| Field | Description |
|---|---|
| 0<br>WEN_L1_1 | Write Enable for Soft Lock Bit SLB_L1_1<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 1<br>WEN_L1_2 | Write Enable for Soft Lock Bit SLB_L1_2<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 2<br>WEN_L1_3 | Write Enable for Soft Lock Bit SLB_L1_3<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 3<br>WEN_L1_4 | Write Enable for Soft Lock Bit SLB_L1_4<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 4<br>SLB_L1_1 | Soft Lock Bit for Control Desc L1_1 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 5<br>SLB_L1_2 | Soft Lock Bit for Control Desc L1_2 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 6<br>SLB_L1_3 | Soft Lock Bit for Control Desc L1_3 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

**Table 11-52. Soft Lock Register L1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 7<br>SLB_L1_4 | Soft Lock Bit for Control Desc L1_4 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 8<br>WEN_L1_5 | Write Enable for Soft Lock Bit SLB_L1_5<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 9<br>WEN_L1_6 | Write Enable for Soft Lock Bit SLB_L1_6<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 10<br>WEN_L1_7 | Write Enable for Soft Lock Bit SLB_L1_7<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 12<br>SLB_L1_5 | Soft Lock Bit for Control Desc L1_5 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 13<br>SLB_L1_6 | Soft Lock Bit for Control Desc L1_6 Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 14<br>SLB_L1_7 | Soft Lock Bit for Control Desc L1_7 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

## 11.3.4.48 Soft Lock DISP_SIZE Register

Figure 11-58 represents the Soft Lock DISP_SIZE register.

Offset: 0x30C — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | SLB_DISP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | WEN_DISP | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 11-58. Soft Lock DISP_SIZE Register**

**Table 11-53. Soft Lock DISP_SIZE Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>WEN_DISP | Write Enable for Soft Lock Bit SLB_DISP<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 4<br>SLB_DISP | Soft Lock Bit for DISP_SIZE Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

## 11.3.4.49 Soft Lock HSYNC/VSYNC PARA Register

Figure 11-59 represents the Soft Lock HSYNC/VSYNC register.

Offset: 0x310 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | WEN_HSYNC | WEN_VSYNC | | | SLB_HSYNC | SLB_VSYNC | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 11-59. Soft Lock HSYNC/VSYNC PARA Register**

**Table 11-54. Soft Lock HSYNC/VSYNC PARA Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>WEN_HSYNC | Write Enable for Soft Lock Bit SLB_HSYNC<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 1<br>WEN_VSYNC | Write Enable for Soft Lock Bit SLB_VSYNC<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 4<br>SLB_HSYNC | Soft Lock Bit for HSYNC Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 5<br>SLB_VSYNC | Soft Lock Bit for VSYNC Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

### 11.3.4.50 Soft Lock POL Register

Figure 11-60 represents the Soft Lock POL Register.

Offset: 0x314 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | WEN_POL | | | | SLB_POL | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 11-60. Soft Lock POL Register**

**Table 11-55. Soft Lock POL Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>WEN_POL | Write Enable for Soft Lock Bit SLB_POL<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 4<br>SLB_POL | Soft Lock Bit for SYN_POL Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

### 11.3.4.51 Soft Lock L0_TRANSP Register

Figure 11-61 represents the Soft Lock L0_TRANSP register.

Offset: 0x318 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | WEN_L0_FCOLOR | WEN_L0_BCOLOR | | | SLB_L0_FCOLOR | SLB_L0_BCOLOR | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 11-61. Soft Lock L0_TRANSP Register**

**Table 11-56. Soft Lock L0_TRANSP Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>WEN_L0_FCOLOR | Write Enable for Soft Lock Bit SLB_L0_FCOLOR<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 1<br>WEN_L0_BCOLOR | Write Enable for Soft Lock Bit SLB_L0_BCOLOR<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| 4<br>SLB_L0_FCOLOR | Soft Lock Bit for L0_FCOLOR Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 5<br>SLB_L0_BCOLOR | Soft Lock Bit for L0_BCOLOR Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

### 11.3.4.52 Soft Lock L1_TRANSP Register

Figure 11-62 represents the Soft Lock L1_TRANSP register.

**Figure 11-62. Soft Lock L1_TRANSP Register**

Offset: 0x31C — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | WEN_L1_FCOLOR | WEN_L1_BCOLOR | | | SLB_L1_FCOLOR | SLB_L1_BCOLOR | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Table 11-57. Soft Lock L0_TRANSP Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>WEN_L1_FCOLOR | Write Enable for Soft Lock Bit SLB_L1_FCOLOR<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified. |
| 1<br>WEN_L1_BCOLOR | Write Enable for Soft Lock Bit SLB_L1_BCOLOR<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified. |
| 4<br>SLB_L1_FCOLOR | Soft Lock Bit for L1_FCOLOR Register.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| 5<br>SLB_L1_BCOLOR | Soft Lock Bit for L1_BCOLOR Register.<br>1'b1:Associated protected register is locked for write access<br>1'b0:Associated protected register is not locked & writeable |

## 11.4 Functional description

The DCU3 is a master on the crossbar switch; it fetches graphic source information directly from memory and dynamically performs blending and bit-blitting operations before delivering data to a TFT LCD panel.

### 11.4.1 Graphic sources

As the DCU3 is a master on the crossbar switch, it can access directly any memory or device connected to the crossbar switch as a slave. This includes all on-chip flash, all on-chip RAM, and any slave capable of providing high enough data rates, such as, for example an expanded bus interface or a QuadSPI module. Therefore, any compatible graphic stored anywhere on-chip or in an accessible interface can be displayed on the connected TFT LCD panel with no further intervention from the CPU, except to program the DCU3 to fetch and place it. The DCU3 also includes a dedicated memory to store the graphic for its cursor layer.

### 11.4.2 TFT LCD panel configuration

The nature and timing of the signals required by TFT LCD panels vary greatly between manufacturers. Therefore, the DCU3 allows highly flexible and detailed configuration of these signals. Refer to the TCON module chapter for additional options when configuring timing signal outputs for panels without an embedded timing controller.

Timing diagrams for TFT LCD panels are typically divided into a horizontal timing chart and a vertical timing chart. See Figure 11-63 for details.

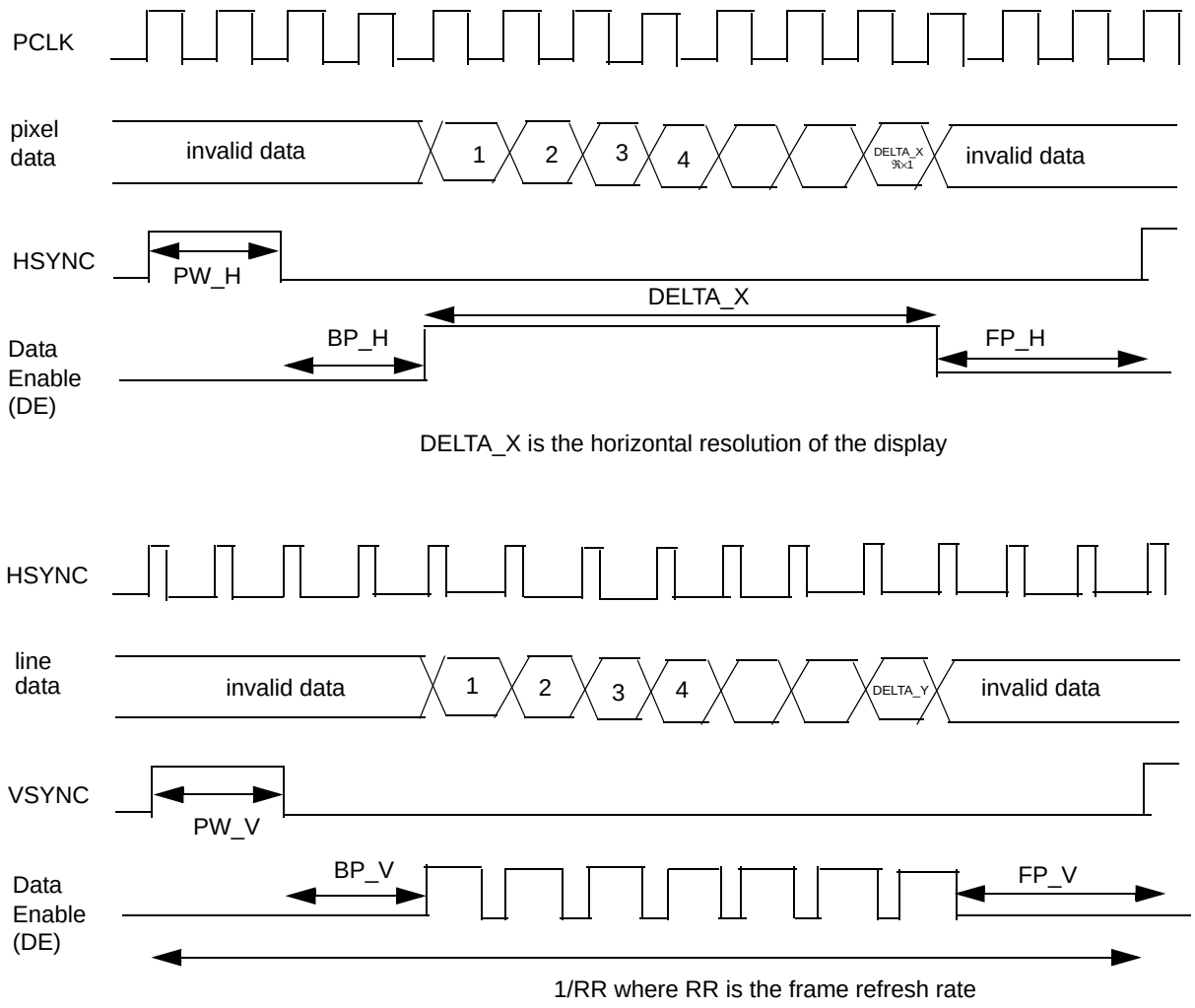



**Figure 11-63. HSYNC and VSYNC timing diagram**

The number of pixel data slots in the horizontal timing diagram is defined by the width of the panel. The number of line data slots is defined by the height of the panel. Both of these values are defined in the DISP_SIZE register (DELTA_X, DELTA_Y). The width of the panel must always be defined as a multiple of 16.

The timing of the pixel clock is defined by the DIV_RATIO register and the frequency of the clock supplied to the DCU3.

In addition to defining the number and timing of pixels in each line and the number of lines, it is normal for TFT LCD panel manufacturers to define other timing signals in terms of pixel clock periods or of the number of horizontal lines. The DCU3 also follows this convention.

If the TFT LCD panel requires a horizontal synchronizing signal (HSYNC) and/or a data enable signal, then these can be configured using the fields in the HSYN_PARA register. HSYNC provides a pulse to give the panel notice that the next line of pixel data is about to start, and the data enable signal indicates when that data is present. The PW_H bit field indicates the width of the HSYNC pulse, in pixel data clock

periods. The BP_H bit field defines the delay between the end of the HSYNC pulse and the start of the data enable signal (and pixel data delivery), in pixel clock periods. The FP_H bit field defines the delay between the end of the data enable signal (and pixel data delivery) and the next HSYNC pulse, in pixel clock periods. FP_H and BP_H have minimum values of 1.

If the TFT LCD panel requires a vertical synchronizing signal (VSYNC), then this can be configured using the fields in the VSYN_PARA register. VSYNC provides a pulse to give the panel notice that the next frame of pixel data lines is about to start, and the panel defines delays before and after this pulse, in terms of pixel clock periods. The PW_V bit field indicates the width of the VSYNC pulse in horizontal line periods. The BP_V bit field defines the delay between the end of the VSYNC pulse and the start of the next pixel data (data enable signal), in horizontal line periods. The FP_V bit field defines the delay between the end of the last pixel data (data enable signal) and the next VSYNC pulse, in horizontal line periods. FP_V and BP_V have minimum values of 1.

The polarity of all these signals, including the pixel data itself, may be inverted by using the control bits in the SYN_POL register.

The refresh rate for the panel can be calculated using Equation 11-1 and Equation 11-2 below.

$$RR = \frac{pix_clk}{(DELTA_X + FP_H + PW_H + BP_H) \times (DELTA_Y + FP_V + PW_V + BP_V)} \quad \textit{Eqn. 11-1}$$

where: pix_clk is the pixel clock

DELTA_X is the horizontal resolution (in pixels)

DELTA_Y is the vertical resolution (in pixels)

FP_H is the HSYNC front porch pulse width (in pixel clock cycles)

BP_H is the HSYNC back porch pulse width (in pixel clock cycles)

PW_H is the HSYNC active pulse width (in pixel clock cycles)

FP_V is the VSYNC front porch pulse width (in pixel clock cycles)

BP_V is the VSYNC back porch pulse width (in pixel clock cycles)

PW_V is the VSYNC active pulse width (in pixel clock cycles)

**Pixel Clock = (DCU3 Clock) / PRESCALE VALUE** *Eqn. 11-2*

where PRESCALE VALUE is an integer value that can range from 2–32.

### 11.4.3 DCU3 mode selection and background color

Once the DCU3 is configured for use with a particular TFT LCD panel, it can be enabled for use. There are five modes to choose from, as shown in Table 11-58.

**Table 11-58. List of DCU3 operating modes**

| Mode | DCU_MODE[1:0] | PDI_EN | Description |
|---|---|---|---|
| Off | 00 | X | DCU3 disabled; the TFT LCD panel is not driven. |
| Color bar | 11 | X | DCU3 displays a test pattern consisting of vertical bands of programmable color. |

Table 11-58. List of DCU3 operating modes

| Mode | DCU_MODE[1:0] | PDI_EN | Description |
|---|---|---|---|
| Normal | 01 | 0 | DCU3 blends layers and displays result on TFT LCD panel. |
| PDI normal | 01 | 1 | As normal mode, except that the panel timing is defined by the input on the PDI interface, and the background color is replaced by the content provided on the PDI interface. |
| PDI slave | 01 | 0 | The DCU3 synchronizes its timing to an external signal when PDI_SLAVE_MODE is enabled. |

The DCU_MODE, PDI_EN and PDI_SLAVE_MODE control bits are in the DCU_MODE register. The DCU3 has an interface enable bit for the TFT LCD panel interface called RASTER_EN, also in the DCU_MODE register. When RASTER_EN is 0 the raster scanning of pixels to the panel is disabled but the pixel clock continues to run as long if enabled on the I/O pin.

Color bar mode is intended for testing the interface between the DCU3 and the TFT LCD panel. In this mode, the panel is divided into eight vertical strips of equal width, and the strips display a single color whose RGB value is specified in the COLBAR_1 to COLBAR_8 registers. At reset, the colors are set to black, blue, cyan, green, yellow, red, magenta, and white, where positive logic for the RGB values is assumed. The mode can be used to verify correct connection of the interface to the DCU3 and correct timing configuration of the interface. In this mode, any layer configuration settings are ignored.

In Normal mode, the DCU3 operates according to the timings specified in Section 11.4.2, “TFT LCD panel configuration”” and displays graphics according to the configuration of its layers. The BGND register sets the RGB color of the background shown when no other layers are present. This background color is included in the layer blending process but, since it is always the background, it does not include any layer blending settings.

In PDI normal mode, the DCU3 adopts the timing provided on the PDI interface and replaces the background color by the pixel data coming from the PDI interface. The timing values set in the DCU3 are ignored in this mode, and the pixel clock and synchronization signals are taken from the PDI interface and passed to the TFT LCD panel. The content of the panel is a combination of the incoming pixel stream and layers generated by the DCU3.

PDI slave mode allows the DCU3 to synchronize with the external timing signals on the PDI input.

### 11.4.4 Layer configuration and blending

Users control the graphical content of the TFT panel by manipulating the configuration of elements in the DCU3 called layers. Each layer has a control descriptor that defines the size, position, memory encoding, blending, and memory location of the graphic to be displayed. The DCU3 provides 16 independent layers that are identical except that they have a fixed priority with respect to each other, and this affects how individual pixels are blended when layers overlap. The blending setting on each layer allows the pixels on that layer to be opaque, partially transparent, or fully transparent, which allows them to combine with pixels on other layers that they overlap.

### 11.4.4.1 Blending priority of layers

The 16 layers available in the DCU3 are each fixed in priority order, with layer 0 being the highest priority, layer 1 being the second highest priority, and so on until layer 15, which is the lowest priority. The priority is used by the DCU3 to define how to blend individual pixels within the layers. For example, if layer 0 is defined as not being blended with other layers and a pixel on layer 0 overlaps a pixel on layer 1 then the pixel on layer 0 will be visible on the panel unchanged by the pixel on layer 1. However, if layer 0 is defined as being partially transparent, then the DCU3 will blend the overlapping pixel such that the result is a combination of the pixel on layer 0 and the pixel on layer 1. It is possible to blend up to four layers at each pixel position.

As there is a maximum number of layers that can blended together, then any pixel on a layer that is lower than the threshold priority will not be included in any blend. If a pixel is on a layer that has the lowest priority in any blending scheme, then the blending settings for that pixel are ignored and the pixel is treated as a background pixel. This means that a lower priority layer may have some pixels completely obscured by those on higher priority layers on one part of the panel, and some other pixels visible or blended on other parts of the panel.

Figure 11-64 shows how the pixel blend takes place inside the DCU3. The priority of the layers determines at which stage of the blend the pixel enters. Any pixels lower than the threshold priority are ignored and, as can be seen, the blend settings for the lowest priority pixel is also ignored. The maximum number of pixels in the blend is configured by the BLEND_ITER bit field in the DCU_MODE register. As can be seen in the figure, the blending process is iterative so that four-pixel blending takes more DCU3 clock cycles than three-pixel blending, and three-pixel blending takes more DCU3 clock cycles than two-pixel blending.

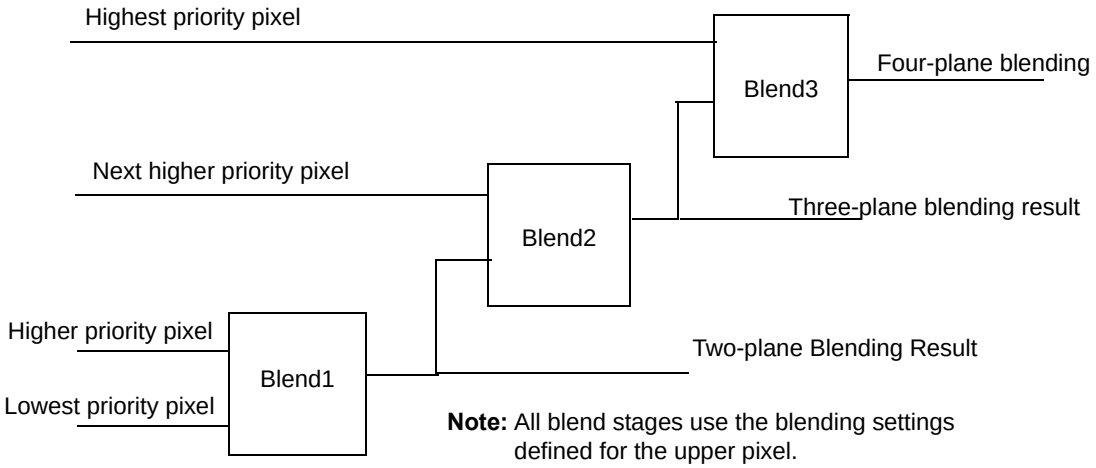



**Figure 11-64. Pixel blending stack**

This priority concept is illustrated in Figure 11-65 and Figure 11-66. In this case, there are five layers enabled, and each contains a graphic that is a solid rectangular block of a single color. The size and shape of each layer is different. The background color of the panel is set to grey and layers have been placed such that they overlap each other.

Figure 11-65 shows the individual source graphics and the case where no layer has any blending enabled. Here, the highest priority layer (in this case layer 0) is fully visible. Layer 1 is visible where layer 0 does not overlap it. Layer 2 is visible where layer 1 does not overlap it. Layer 3 is overlapped by layers 0 and 1 and so is only partially visible. Layer 4 is partially obscured by all of the other layers. Note that layer 4 is higher priority than the background color.

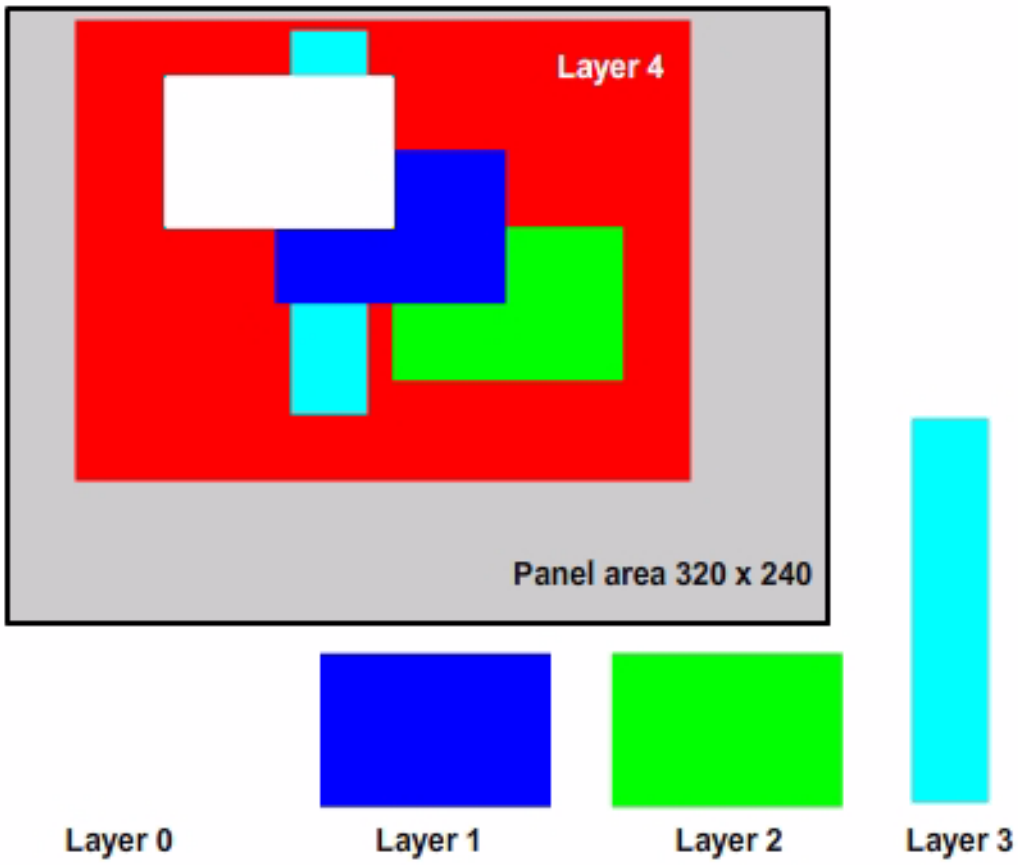



**Figure 11-65. Example of layer placement with no blending**

Figure 11-66 shows the same layer configuration except that, in this case, the layers have been made 50% transparent and the depth of the pixel blend is set to 3. The pixels in layer 0 are now blended with pixels in the underlying layers. In particular, note region A where layer 0 is blended with layer 4 and the background color. This blending effect is repeated across all of the layers; however, note the pixels in region B. In this region the pixels from layer 0 are blended with those on layer 1 and layer 2; however, the pixels from layer 4 and the background color are not included in this blend. This is because the DCU3 is configured to blend three layers only, and so the blend setting for layer 2 is ignored for those pixels in region B.

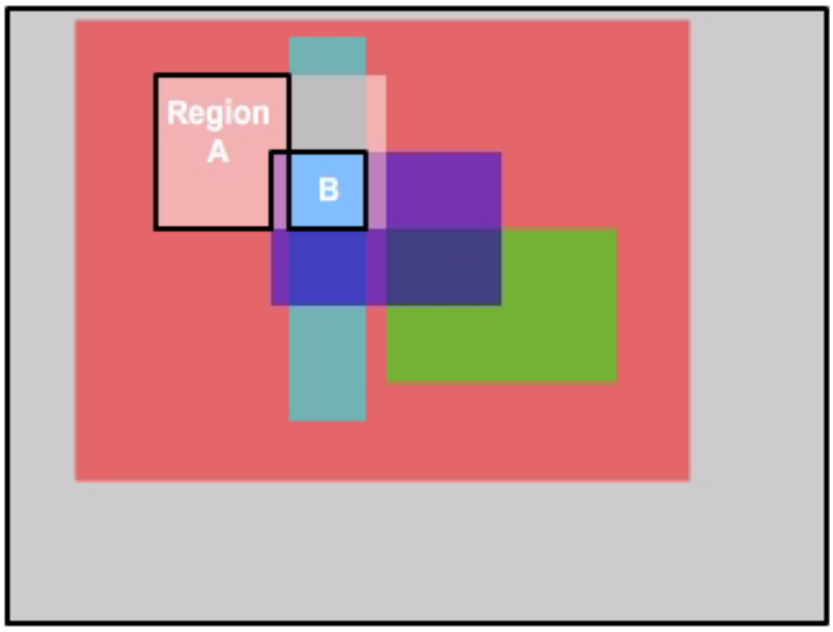



**Figure 11-66. Example of layer placement with 3-layer blending**

All blending is performed using full 8-bits-per-component colors. The DCU3 automatically performs a color promotion on source data that is stored in less than RGB888 color.

### 11.4.4.2 Control Descriptors

The control descriptor for each layer consists of seven registers, and all 16 control descriptors are identical except the two highest priority layers, which have additional control bits for the safety mode.

The control descriptors may be written to at any time, and the value present in the registers at the start of the next frame refresh cycle defines the content of the panel for that frame. To avoid coherency issues, ensure all control descriptor changes are made before the PROG_END bit in the INT_STATUS register is asserted.

### 11.4.4.3 Layer size and positioning

The size of each layer is defined by register 1 in the control descriptor for the layer (CTRLDESCLn_1, where n is the layer number). The register contains two bit fields, HEIGHT and WIDTH, which determine the size and shape of the layer. Both fields are expressed in terms of the number of pixels in each dimension.

The HEIGHT bit field may take any value; however, it may not be useful to define a value larger than the height of the panel.

The WIDTH field has a restriction on the value it can take, depending on the data format of the graphic specified by the layer. This field must always be an integer multiple of the number of pixels that are represented by a 32-bit word except in the special case of 1 bit per pixel where the multiple is 16. The data format can range from 1 bit per pixel to 32 bits per pixel and so there is a range of multiples from 1 to 32. Figure 11-59 shows the multiples for the WIDTH bit field and some correct values.

**Table 11-59. Example of WIDTH multiples for different graphic data formats**

| Data format | WIDTH multiples | Example values |
|---|---|---|
| 1 bpp | 16 | 16, 32, 48, 64, … |
| 2 bpp | 16 | 16, 32, 48, 64, … |
| 4 bpp | 8 | 8, 16, 24, 32, … |
| 8 bpp | 4 | 4, 8, 12, 16, … |
| 16 bpp | 2 | 2, 4, 6, 8, … |
| 24 bpp | 4 (= 3 whole 32-bit words) | 4, 8, 12, 16 |
| 32 bpp | 1 | 1, 2, 3, 4, … |
| YCbCr422 | 4 | 4, 8, 12, 16, … |

If the WIDTH bit field is set to an invalid multiple, then the layer configuration is invalid, the layer cannot be made visible, and an error flag is set in the layer parameter error register (PARR_ERR).

The position of each layer on the panel is defined by register 2 in the control descriptor for the layer (CTRLDESCLn_2, where n is the layer number). The register contains two bit fields, POSY and POSX, which determine the location of the upper left pixel of the layer in the x and y axes. Both fields are expressed in terms of the number of pixels in each axis.

There are no restrictions on layer placement. Any layer can be placed and moved to any panel position. If a layer is placed so that pixels would appear beyond the dimensions of the panel, then the DCU3 displays the pixels on the panel and ignores the pixels off the panel.

### 11.4.4.4 Graphics and data format

The memory location of the graphic that is displayed on the layer is defined by register 3 in the control descriptor for the layer (CTRLDESCLn_3, where n is the layer number). This 32-bit value can contain the address of any 64-bit aligned memory location in the memory map of the MCU.

The format of the data that describes the graphic is defined by the BPP bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). This value also influences the range of values for the width of the layer (see Section 11.4.4.3, “Layer size and positioning””). By choosing an appropriate format, it is possible to optimize the memory required by the graphics in use.

There are five formats where the RGB values of the pixels are stored directly in the graphic. In these formats, the DCU3 treats the data as describing a true RGB color. The formats are:

- BGRA8888, where the data defines 8-bit values for the red, green, blue, and alpha components of the image. This blends as ARGB, however, in this format the order of the bytes is reversed compared to other formats.
- RGB888, where the data defines 8-bit values for the red, green, and blue components of the image.
- RGB565 where the data defines 5-bit values for the red and blue components, and 6-bit values for the green component of the image.

- ARGB1555 where the data defines 5-bit values for the red, green, and blue components, and a 1-bit value for the alpha channel of the image.
- ARGB4444, where the data defines 4-bit values for the red, green, blue, and alpha components of the image.

The three 16-bit formats (RGB565, ARGB1555, and ARGB4444) are promoted to full 8 bit per component format by shifting the bits left so that the MSB of the component in the 16-bit format becomes the MSB of the 24/32 bpp (bit per pixel) format, and the LSB is filled with the value of the MSBs. For example, an RGB565 value of 10000:010000:11011 becomes 10000100:01000001:11011110. An RGB4444 value of 1010:0011:1100:0101 becomes 10101010:00110011:11001100:01010101. An RGB1555 value of 1:10100:01000:11011 becomes 11111111:10100101:01000010:11011110.

There are five indexed color formats (1/2/4/8 bpp & APAL8) where the data in the graphic does not define the RGB color to display. Instead, the data defines the entry in a color look-up table (CLUT) that contains a palette of ARGB colors. The maximum number of colors in the CLUT is defined by the size of the data stored in the graphic. For 1 bpp graphics, there is a maximum of two colors in the CLUT. For 2 bpp, there is a maximum of four colors. For 4 bpp and 8 bpp data, the maximums are 16 and 256 colors, respectively. In APAL8 mode(16 bpp), the upper 8 bits define the alpha component of the pixel and the lower 8 bits define the offset in the CLUT (the alpha component in the CLUT color is ignored).

The address of the first value in the CLUT is defined in the LUOFFS bit field of register 4 and the CLUT is the RAM block dedicated to the DCU3 which is described in Section 11.4.6, “CLUT/tile RAM””. Since the ARGB values stored in the CLUT are 32-bit ARGB, there is no need for further adjustment before blending.

The DCU3 also supports graphics encoded using luminance and chrominance format. This format is generically known as YUV and stores the luminance (brightness, Y) of a pixel separate from its chrominance (color information, U and V). This format is widely used by cameras and is supported by the PDI for direct video in as well as the DCU3 when stored in memory for display on a layer. The specific implementation used by the DCU3 is more accurately described as YCbCr422 which uses twice as many bits to describe the luminance as to describe the blue (Cb) and red (Cr) difference of the chrominance.

The DCU3 takes these pixels and converts them to RGB format using equations configured using its LYR_LUMA_COMP, LYR_CHROMA_RED, LYR_CHROMA_GREEN, and LYR_CHROMA_BLUE registers. The YCbCr format specifies a common chroma setting for two pixels; however, it is possible to interpolate the chroma for the pixels rather than setting both to the same value. This feature is enabled by the LYR_INTPOL_EN[EN] field. Due to the additional conversion step required, the DCU3 is able to blend a maximum of one layer encoded in YCbCr for each pixel. See Section 11.8.1.6.3, “PDI YCbCr mode and DCU3 YCbCr color format””.

There are four additional formats defined by the BPP bit field. These configure the graphic in transparency mode and luminance mode (see Section 11.4.4.6, “Transparency mode and blending”” and Section 11.4.4.7, “Luminance mode”” respectively).

There is a set storage format for each data format provided by the DCU3. These formats can be seen in Table 11-60 to Table 11-70.

**Table 11-60. Data layout for BGRA8888**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | B0 | G0 | R0 | A0 | B1 | G1 | R1 | A1 |
| 0x08 | B2 | G2 | R2 | A2 | B3 | G3 | R3 | A3 |

**Table 11-61. Data layout for YCbCr422 format[1]**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | Cb0 | Y0 | Cr0 | Y1 | Cb2 | Y2 | Cr2 | Y3 |
| 0x08 | Cb4 | Y4 | Cr4 | Y5 | Cb6 | Y6 | Cr6 | Y7 |

1 The YCbCr422 format encodes chroma information across two pixels. Therefore, the chroma values apply to the even pixel denoted in the table and its adjacent odd pixel.

**Table 11-62. Data layout for 24 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | B0 | G0 | R0 | B1 | G1 | R1 | B2 | G2 |
| 0x08 | R2 | B3 | G3 | R3 | B4 | G4 | R4 | B5 |

For 16 bpp, data expected is in the form of RGB565, ARGB1555, ARGB4444, or APAL8.

**Table 11-63. Generic data layout for 16 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | pixel0[15:8] | pixel0[7:0] | pixel1[15:8] | pixel1[7:0] | pixel2[15:8] | pixel2[7:0] | pixel3[15:8] | pixel3[7:0] |
| 0x08 | pixel4[15:8] | pixel4[7:0] | pixel5[15:8] | pixel5[7:0] | pixel6[15:8] | pixel6[7:0] | pixel7[15:8] | pixel7[7:0] |

**Table 11-64. Data layout for RGB565 format**

| Address offset | [0:4] | [5:10] | [11:15] | [16:20] | [21:26] | [27:31] |
|---|---|---|---|---|---|---|
| 0x00 | R0 | G0 | B0 | R1 | G1 | B1 |
| 0x04 | R2 | G2 | B2 | R3 | G3 | B3 |

**Table 11-65. Data layout for ARGB1555 format**

| Address offset | [0] | [1:5] | [6:10] | [11:15] | [16] | [17:21] | [22:26] | [27:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | A0 | R0 | G0 | B0 | A1 | R1 | G1 | B1 |
| 0x04 | A2 | R2 | G2 | B2 | A3 | R3 | G3 | B3 |

**Table 11-66. Data layout for APAL8 format**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | A0 | Offset 0 | A1 | Offset 1 | A2 | Offset 2 | A3 | Offset 3 |
| 0x08 | A4 | Offset 4 | A5 | Offset 5 | A6 | Offset 6 | A7 | Offset 7 |

**Table 11-67. Data layout for 8 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | pixel0[0:7] | pixel1[0:7] | pixel2[0:7] | pixel3[0:7] | pixel4[0:7] | pixel5[0:7] | pixel6[0:7] | pixel7[0:7] |
| 0x08 | pixel8[0:7] | pixel9[0:7] | pixel10[0:7] | pixel11[0:7] | pixel12[0:7] | pixel13[0:7] | pixel14[0:7] | pixel15[0:7] |

**Table 11-68. Data layout for 4 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | pixel1-pixel0 | pixel3-pixel2 | pixel5-pixel4 | pixel7-pixel6 | pixel9-pixel8 | pixel11-pixel10 | pixel13-pixel12 | pixel15-pixel14 |
| 0x08 | pixel17-pixel16 | pixel19-pixel18 | pixel21-pixel20 | pixel23-pixel22 | pixel25-pixel24 | pixel27-pixel26 | pixel29-pixel28 | pixel31-pixel30 |

**Table 11-69. Data layout for 2 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | pixel3-pixel0 | pixel7-pixel4 | pixel11-pixel8 | pixel15-pixel12 | pixel19-pixel16 | pixel23-pixel20 | pixel27-pixel24 | pixel31-pixel28 |
| 0x08 | pixel35-pixel32 | pixel39-pixel36 | pixel43-pixel40 | pixel47-pixel44 | pixel51-pixel48 | pixel55-pixel52 | pixel59-pixel56 | pixel63-pixel60 |

**Table 11-70. Data layout for 1 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | pixel7-pixel0 | pixel15-pixel8 | pixel23-pixel16 | pixel31-pixel24 | pixel39-pixel32 | pixel47-pixel40 | pixel55-pixel48 | pixel63-pixel56 |
| 0x08 | pixel71-pixel64 | pixel79-pixel72 | pixel87-pixel80 | pixel95-pixel88 | pixel103-pixel96 | pixel11-pixel104 | pixel119-pixel112 | pixel127-pixel120 |

The DCU3 includes a flag that indicates when it has completed fetching graphics from memory for the current frame refresh. If required, this flag (DMA_TRANS_FINISH in the INT_STATUS register) can be used to determine when changes can be made to the source graphic content.

### 11.4.4.5 Alpha and chroma-key blending

The blending configuration of each layer is defined by the BB, AB, and TRANS bit fields in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). The pixels affected by the blending configuration can be further selected by registers 5 and 6 in the control descriptor (CTRLDESCLn_4 and CTRLDESCLn_5). Depending on the priority and placement of the layer (see Section 11.4.4.1, “Blending priority of layers””), these bit fields and registers define how pixels from different layers are blended together.

The AB and BB bit fields define whether blending is active and whether the whole graphic or a selected portion is blended. Registers 5 and 6 define the range of RGB colors that define the selected pixels. The TRANS bit field defines the transparency of the selected pixels.

The BB bit field defines whether the whole graphic, or only certain pixels, should be blended. When this bit is set, pixels that have an RGB value that falls into the range defined by registers 5 and 6 are considered to be selected and treated differently to the non-selected pixels in the graphic. This is a process known as chroma-keying since it is the color of the pixel that defines the selection. The selected pixels must be within the range defined by each color component of registers 5 and 6. See Table 11-71 for examples of pixels that are selected and not selected when the given range is defined as 0x0080C0 to 0x0FB0FF.

**Table 11-71. Example of how chroma-key range selects pixels**

| Source pixel | Red 00–0F | Green 80–B0 | Blue C0–FF | Comment |
|---|---|---|---|---|
| 0x000000 | P | X | X | Not selected |
| 0x08C0C0 | P | X | P | Not selected |
| 0x08A0C0 | P | P | P | Pixel is selected |

The AB bit field defines how any selected and non-selected pixels are blended. By combining this control with the BB bit field it is possible to define 11 unique ways of blending the pixels on a layer dependent on the type of layer. Depending on the configuration defined by the AB and BB bit fields, the TRANS bit field combines the two pixels in every blend stage using the alpha value of the upper pixel (which has the effect of making this pixel more or less transparent and revealing more or less of the lower pixel).

The result of each blend stage is calculated for all three color components as shown in Equation 11-3.

$$A = (BGPixel*(255 - alpha)) + (FGPixel * alpha) \qquad \textit{Eqn. 11-3}$$

The result of the calculation must then be divided by 255 to normalize the result. This calculation is performed as follows:

```
//First Division
output_val = A + (A >> 8)
//Rounding off first addition & division
if (((A>>7) & 0x1) == 0x1)
output_val ++
//Second Division with rounding
```

output_val = output_val >>7;

if ((output_val & 0x1) == 0x1))

output_val = output_val + 0x2;

output_val = output_val >> 1;

The blend can apply to pixels with no alpha channel (RGB) or with an alpha channel (ARGB) in different ways.

Table 11-72 defines how the settings of the BB and AB bit fields affect the pixels in the layer; RGB formats are RGB565, and RGB888; ARGB formats are 1 bpp, 2 bpp, 4 bpp, 8 bpp, APAL8, ARGB1555, ARGB4444, and BGRA8888. Pixels in YCbCr format are treated as RGB pixels for the purposes of the blend.

**Table 11-72. Blend options for BB and AB configurations**

| Case | BB | AB | Format | Function |
|---|---|---|---|---|
| 1 | 0 | 00 | RGB | No blending, underlying pixels are obscured |
| 2 | 1 | 00 | RGB | Selected pixels are completely removed |
| 3 | 0 | 01 | RGB | The value in TRANS becomes the alpha channel of all pixels on the layer |
| 4 | 1 | 01 | RGB | The value in TRANS becomes the alpha channel of the selected pixels on the layer |
| 5 | 0 | 10 | RGB | Same as case 3 |
| 6 | 1 | 10 | RGB | Selected pixels are completely removed and the value in TRANS becomes the alpha channel of the non-selected pixels on the layer |
| 7 | 0 | 11 | RGB | Reserved |
| 8 | 1 | 11 | RGB | Reserved |
| 9 | 0 | 00 | ARGB | No blending, pixel alpha is ignored and underlying pixels are obscured |
| 10 | 1 | 00 | ARGB | Selected pixels are completely removed, pixel alpha is ignored |
| 11 | 0 | 01 | ARGB | Pixel alpha is used to blend layer with underlying pixels. Value in TRANS is ignored. |
| 12 | 1 | 01 | ARGB | Uses the pixel alpha of the selected pixels only to blend layer with underlying pixels. Value in TRANS is ignored. |
| 13 | 0 | 10 | ARGB | The value in TRANS is multiplied with the pixel alpha value and the resultant alpha is used to blend all the pixels |
| 14 | 1 | 10 | ARGB | Selected pixels are completely removed, the value in TRANS is multiplied with the pixel alpha value and the resultant alpha is used to blend the non-selected pixels on the layer |
| 15 | 0 | 11 | ARGB | Reserved |
| 16 | 1 | 11 | ARGB | Reserved |

Figure 11-67 to Figure 11-75 illustrate the effect of the cases identified in Table 11-72. In all cases there is a single active layer and a white background color.

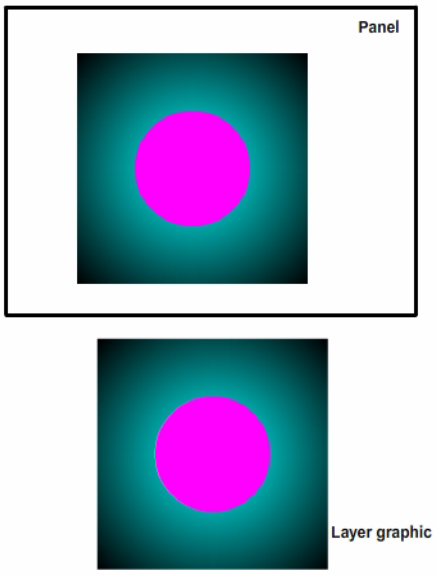



**Figure 11-67. Case 1 example (no blend)**

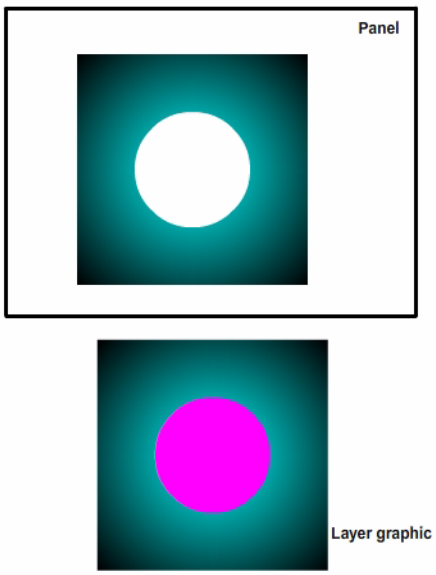



**Figure 11-68. Case 2 example (remove selected pixels)**

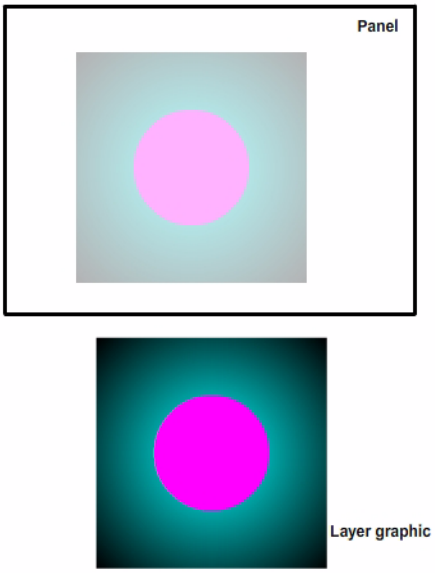



**Figure 11-69. Case 3 example (all pixels transparent)**

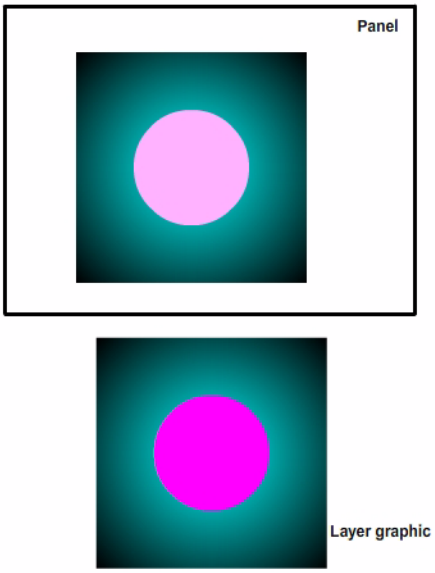



**Figure 11-70. Case 4 example (selected pixels transparent)**

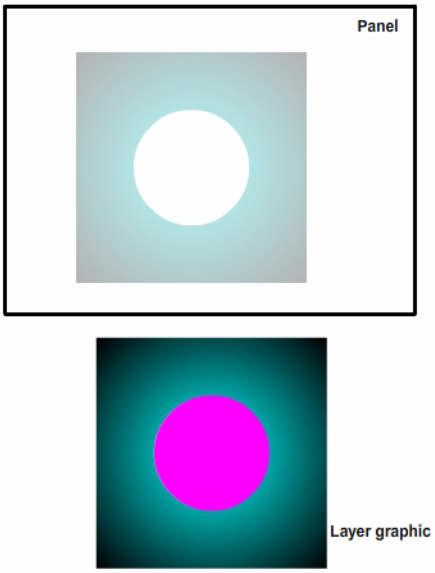



**Figure 11-71. Case 6 example (selected pixels removed, others transparent)**

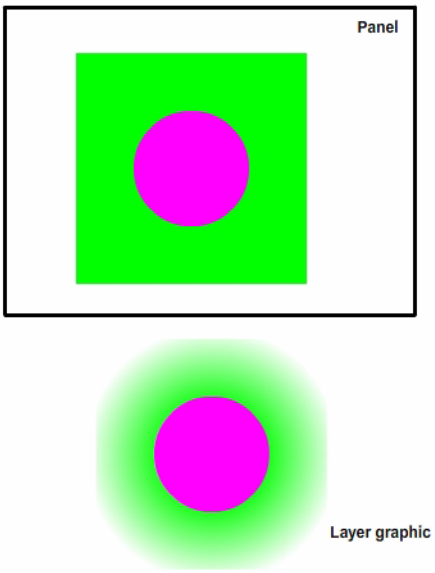



**Figure 11-72. Case 9 example (no blend, pixel alpha ignored)**

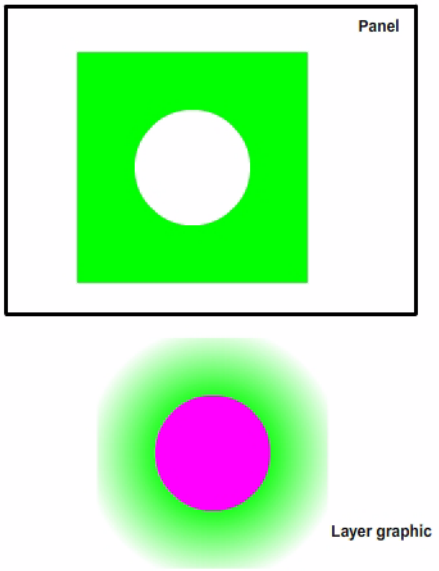



**Figure 11-73. Case 10 example (selected pixels removed, pixel alpha ignored)**

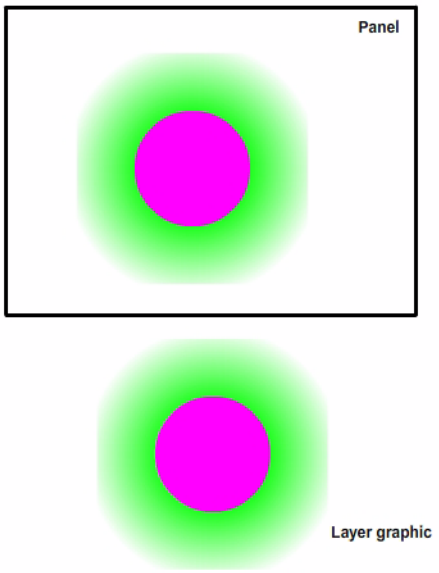



**Figure 11-74. Case 13 example (pixel and layer alpha used in blend)**

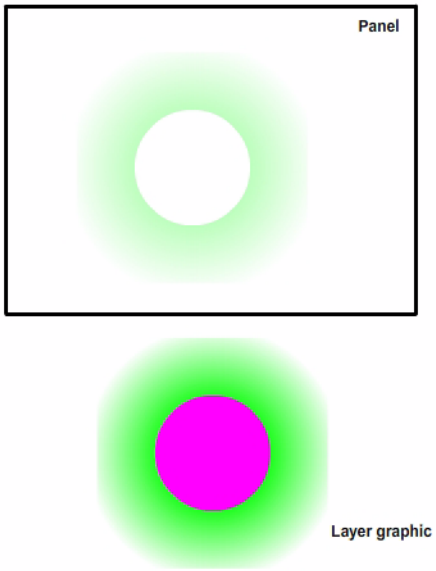



**Figure 11-75. Case 14 example (selected pixels removed, pixel and layer alpha used in blend)**

### 11.4.4.6 Transparency mode and blending

Transparency mode is a special case for the graphic data format and is defined by the BPP bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). This value also influences the range of values for the width of the layer (see Section 11.4.4.3, “Layer size and positioning””). By choosing an appropriate format, it is possible to optimize the memory required by the graphics in use.

In transparency mode, the source graphic does not contain any direct or indexed color information. Instead, the graphic data represents the alpha channel of the graphic. The DCU3 creates the final graphic by pre-blending a foreground color and background color using the alpha value of each pixel. The result of this pre-blend can then be blended with pixels on other layers using the normal blending process. Each layer has dedicated registers to contain the foreground and background colors for this mode. These are FGn_fcolor and FGn_bcolor, where n is the layer number. See Figure 11-76.

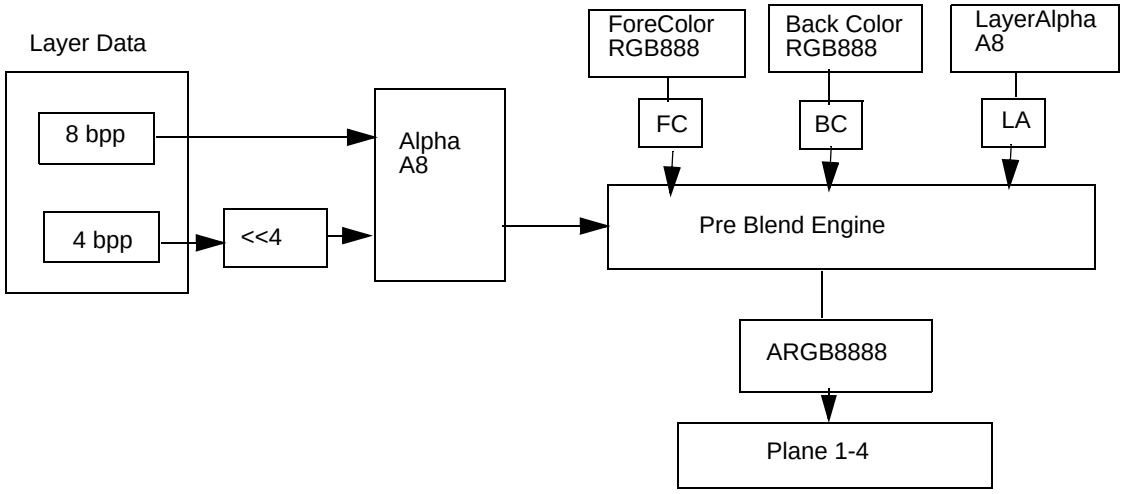



**Figure 11-76. Transparency mode description**

Transparency mode is typically used when a graphic must blend smoothly into the underlying layers, but where a rich color palette is not required. Examples include text where this mode allows the text to blend smoothly with any background — this is known as anti-aliasing.

There are two transparency modes available: 4 bpp and 8 bpp. The result of the pre-blend can be treated as an RGB888 graphic and blended in a similar way to previously described, or it can be treated as a special case of ARGB with only the foreground color visible in the final blend. Table 11-73 describes the blend options for transparency mode.

**Table 11-73. Blend options for transparency mode**

| Case | BB | AB[1:0] | Mode | Function |
|---|---|---|---|---|
| 1 | 0 | 00 | Transparency | No blending, underlying pixels are obscured |
| 2 | 1 | 00 | Transparency | Reserved |
| 3 | 0 | 01 | Transparency | The value in TRANS becomes the alpha channel of all pixels on the layer |
| 4 | 1 | 01 | Transparency | The value in TRANS becomes the alpha channel of the selected pixels on the layer |
| 5 | 0 | 10 | Transparency | Same as case 3 |
| 6 | 1 | 10 | Transparency | Background color is ignored, selected pixels are completely removed, the value in TRANS is multiplied with the graphic data value (alpha) and the resultant alpha is used to blend the non-selected pixels on the layer |
| 7 | 0 | 11 | Transparency | Reserved |
| 8 | 1 | 11 | Transparency | Reserved |

Figure 11-77–Figure 11-80 illustrate the effect of the cases identified in Table 11-73. In all cases there is a single active transparency layer and a white background color.

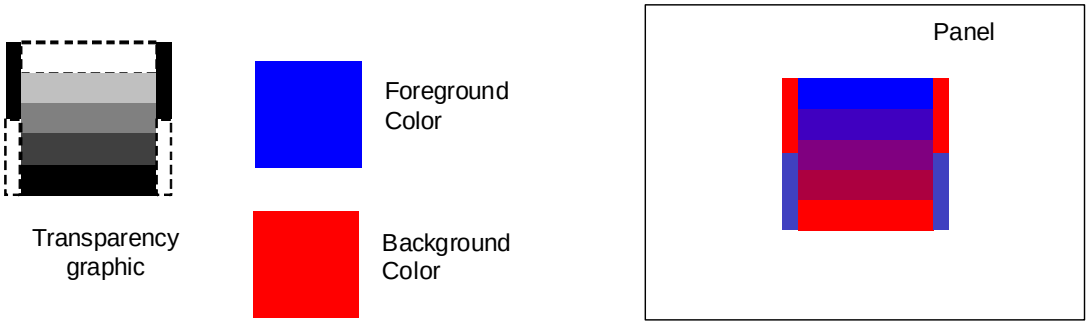



**Figure 11-77. Case 1 example (no blend)**

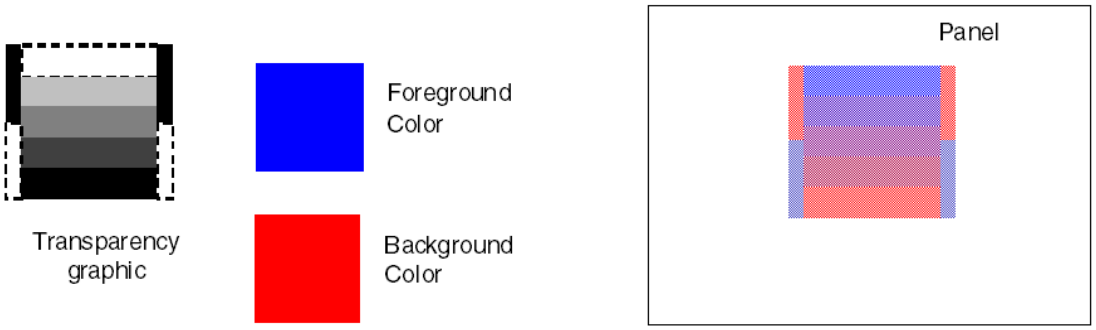



**Figure 11-78. Case 3 example (all pixels transparent)**

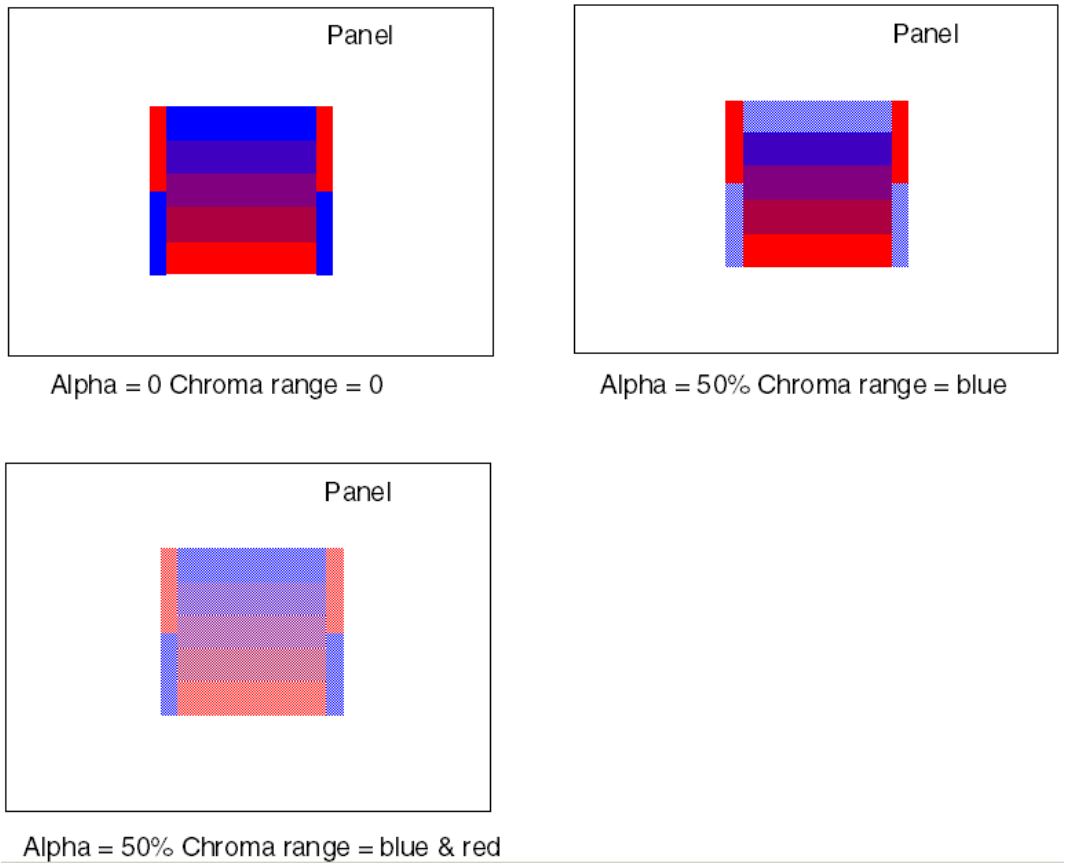



**Figure 11-79. Case 4 example (selected pixels transparent)**

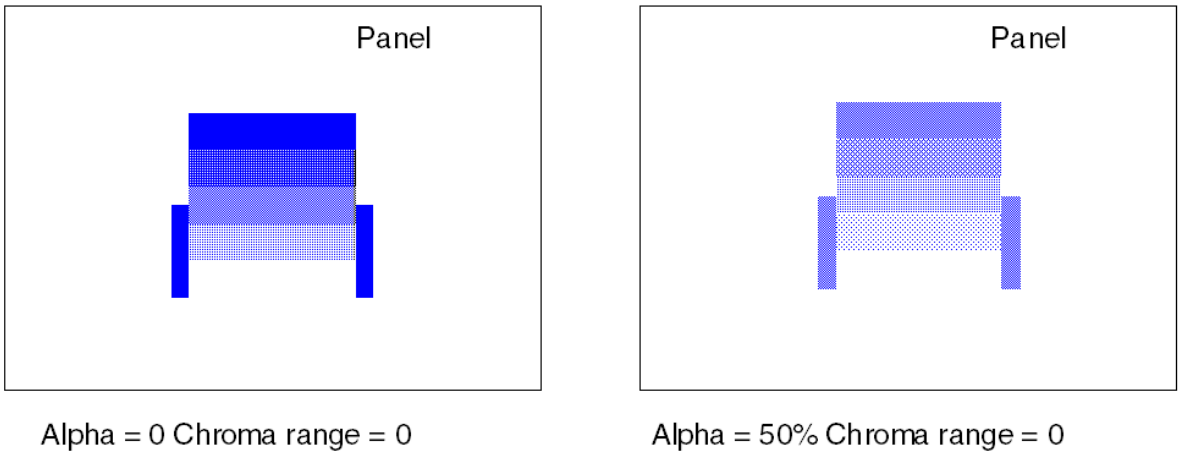



**Figure 11-80. Case 6 example (only foreground color blended)**

### 11.4.4.7 Luminance mode

Luminance mode is a special case for the graphic data format and is defined by the BPP bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). This value also influences the range of values for the width of the layer (see Section 11.4.4.3, “Layer size and

positioning””). By choosing an appropriate format, it is possible to optimize the memory required by the graphics in use.

In luminance mode, the data in the source graphic does not contain any direct or indexed color information or alpha information. The data values in a layer in luminance mode modify the values of the pixels on underlying layers only. There are two luminance modes available: 4 bpp and 8 bpp. In both cases, the data values behave as signed integers that are added to each component of the underlying pixel. The 4 bpp mode is left-shifted to form a signed 8 bpp integer. The results of the addition are prevented from overflowing, so that any result greater than 0xFF is set to 0xFF and any result less than 0x00 is set to 0x00.

The result of a blend with a luminance layer is that the intensity of the underlying pixel's color will be increased or decreased. In this way, luminance mode can be used to highlight or dim pixels on the panel without having to modify the source graphic data. Table 11-74 describes the effect of luminance blends on an underlying pixel.

**Table 11-74. Example of a blend with a luminance mode layer**

| Pixel value | Luminance value | Resultant pixel |
|---|---|---|
| 0xFF8040 | 0x40 | 0xFFC080 |
| 0xFF8040 | 0xC0 | 0x3F0000 |

### 11.4.4.8 Tile mode

Tile mode is a special case for the layer and is enabled by the TILE_EN bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number).

In this mode the layer size register (CTRLDESCLn_1, where n is the layer number) defines the size of the layer; however, the size of the graphic is defined in control register 7 (CTRLDESCLn_7, where n is the layer number). The size of the graphic must be less than or equal to the size of the layer. When tile mode is enabled, the graphic is repeated horizontally and vertically until it fills the whole layer. The horizontal size of the tile is defined by the TILE_HOR_SIZE bit field and is restricted to be a multiple of 16 pixels. The vertical size of the tile is defined by the TILE_VER_SIZE bit field.

The graphic data for the Tile Mode can be fetched either from the system memory or from the internal CLUT/Tile memory. This is defined by the DATA_SEL bit field in register 4 in the control descriptor for the layer. If the graphic is fetched from CLUT/tile memory then it must be in the CLUT/tile RAM direct color format. Otherwise the graphic can be in any previously described data format. See Figure 11-81 for an example of a layer in tile mode.

When DATA_SEL is set (to use CLUT/TILE RAM) the LUOFFS bit-field defines the start address of the tile graphic. Tile mode is not permitted when RLE mode is active on a layer.

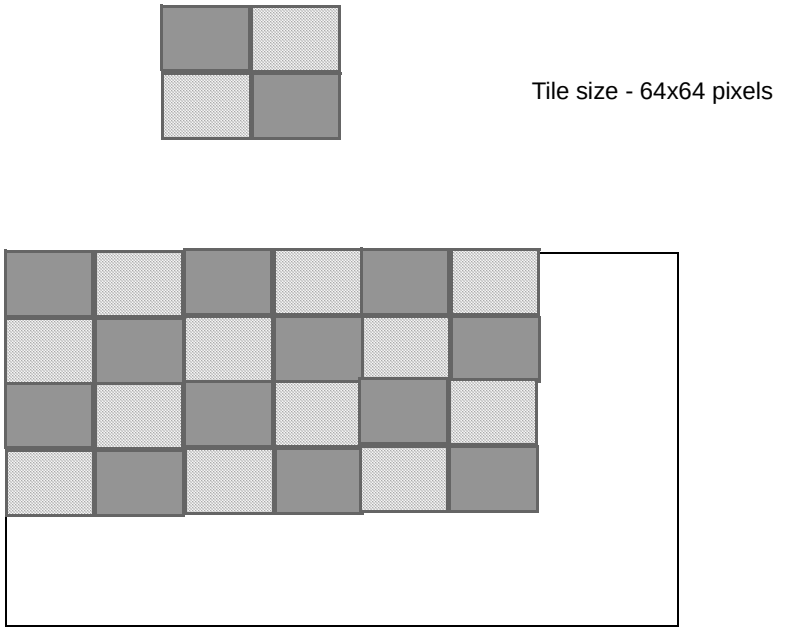



**Figure 11-81. Tile Mode**

## 11.4.5 Hardware cursor

In addition to the 16 layers, the DCU3 also provides a special layer intended for use as a cursor. This cursor operates in 1 bpp mode and includes its own RAM area to store the graphic. The cursor may be placed at any location on the panel and includes an automatic blink option. The hardware cursor is configured using a dedicated control descriptor.

The size of the cursor is defined by register 1 in the control descriptor for the cursor (CTRLDESCCURSOR_1). The register contains two bit fields, HEIGHT and WIDTH, which determine the size and shape of the layer. Both fields are expressed in terms of the number of pixels in each dimension. The HEIGHT is limited to a maximum of 256 pixels, and the total number of pixels cannot exceed the number of bits in the cursor RAM (8192 bits).

Bits in the cursor RAM that are 0 become transparent on the panel. Bits that are 1 become fully opaque in the color defined in register 3 in the control descriptor for the cursor (CTRLDESCCURSOR_3). The DEFAULT_CURSOR_COLOR bit field is in RGB888 format.

There are restrictions on the arrangement of bits in the cursor RAM depending on how the HEIGHT and WIDTH bit fields are configured.

- The rightmost bit in the cursor RAM (bit 31) represents the leftmost pixel on the display.
- When the cursor size is less than 32 bits, each row of the cursor is contained in a single 32-bit word of cursor RAM. The other bits in each row must be filled with zeros.

- When the cursor width is an integer multiple of 32 bits, the pixels in each row roll from one word in the RAM to the next one. The rightmost bit in the first word in the RAM is the top leftmost pixel on the display. The leftmost bit in the word represents a pixel that is adjacent to the rightmost bit in the next word (in the same row). The leftmost pixel on the next row is the rightmost bit in the first word after n words that describe the first row.
- When the cursor is greater than 32 bits but not an integer multiple of 32, the pixels in each row roll from one word into the next one such that the rightmost bit in the first word of the row is the leftmost bit on the display. In the final word of the row there are unused bits.

The position of the cursor on the panel is defined by register 2 in the control descriptor for the cursor (CTRLDESCCURSOR _2). The register contains two bit fields, POSY and POSX, which determine the location of the upper left pixel of the cursor in the x and y axes. Both fields are expressed in terms of the number of pixels in each axis. Placing the cursor beyond the panel area is not allowed.

The cursor can be configured to blink at a particular rate when it is enabled. The EN_BLINK, HWC_BLINK_ON, and HWC_BLINK_OFF bit fields define the blink behavior. These are in register 4 in the control descriptor for the cursor (CTRLDESCCURSOR_4). EN_BLINK enables blinking. The blinking time is based on the frame rate, and the on and off times are independently configurable. HWC_BLINK_ON configures the number of frame refresh cycles for which the cursor is visible. HWC_BLINK_OFF configures the number of frame refresh cycles for which the cursor is not visible. For a frame refresh rate of 64 Hz, the HWC_BLINK_ON and HWC_BLINK_OFF counters give a range of on/off times up to 4 seconds.

The cursor is enabled by setting the CUR_EN bit field in register 3 in the control descriptor for the cursor (CTRLDESCCURSOR_3).

If the DCU3 detects an invalid configuration in the cursor control descriptor, then the cursor configuration is invalid and it cannot be made visible. In addition, the error flag HWC_ERR is set in the layer parameter error register (PARR_ERR).

The cursor RAM may be written at any time when the TFT LCD panel is not being driven with data. This means that the RAM can be modified when the DCU3 is not enabled and during the vertical blanking period.

### 11.4.6 CLUT/tile RAM

The internal tile memory and color look up table (CLUT) memory share a common block of RAM internal to the DCU3. Color information in this RAM is always stored as aligned 32-bit words where the most-significant byte is the alpha component, the next byte contains the red component, the next the green component and the least significant byte the blue component (0xAARRGGBB).

This memory block can be used to store either color look-up tables or graphics for use as a tile on a layer. The content of the RAM at a specific address is defined by the control descriptor of a layer. The LUOFFS bit field in the layer control descriptor defines the starting address of the area, and the BPP and TILE_EN bit fields define what type of use the RAM area has.

In Figure 11-82 three areas of the RAM are defined for different purposes. Area A is used by layer 1 as a CLUT for its 4 bpp graphic. Area B is use by layer 5 as a store for its tile graphic. Area C is used by layers 2, 7, and 9 as a CLUT for their 8 bpp graphics.

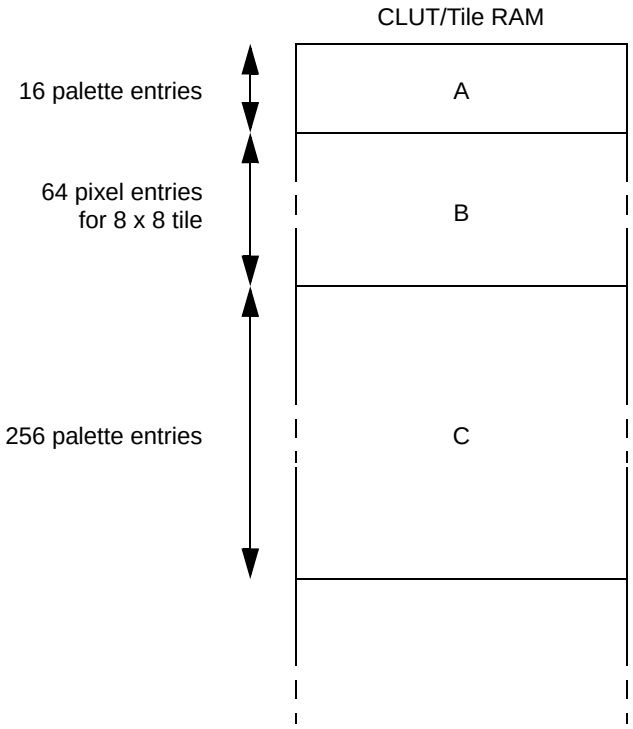



**Figure 11-82. An example of use for the CLUT/Tile RAM**

The CLUT/Tile RAM is mapped in the DCU3 16K memory space from address 0x2000 to 0x3FFF. This gives 2048 entries, which provides up to eight full CLUTs for 8 bpp layers.

The CLUT/Tile RAM may be written at any time when the TFT LCD panel is not being driven with data. This means that the RAM can be modified when the DCU3 is not enabled and during the vertical blanking period.

## 11.4.7 Gamma correction

The gamma table allows the user to define an arbitrary transfer function at the output of each color component. The function (Equation 11-4) is applied to each pixel after all blending is complete and before the data is driven to the TFT LCD panel. Gamma correction is optional and can be used to adjust the color output values to match the gamut of a particular TFT LCD panel, or to perform data inversion or data length reduction on each component.

**output_color_component = gamma_table[input_color_component]** ***Eqn. 11-4***

The table is arranged as three separate memory blocks within the DCU3 memory map; one for each of the three color components. Each memory block has one entry for every possible 8-bit value and the entries are stored at 32-bit aligned addresses. This means that the upper 24 bits are not used while reading/writing the gamma memories. See Figure 11-83 for details of the memory arrangement.

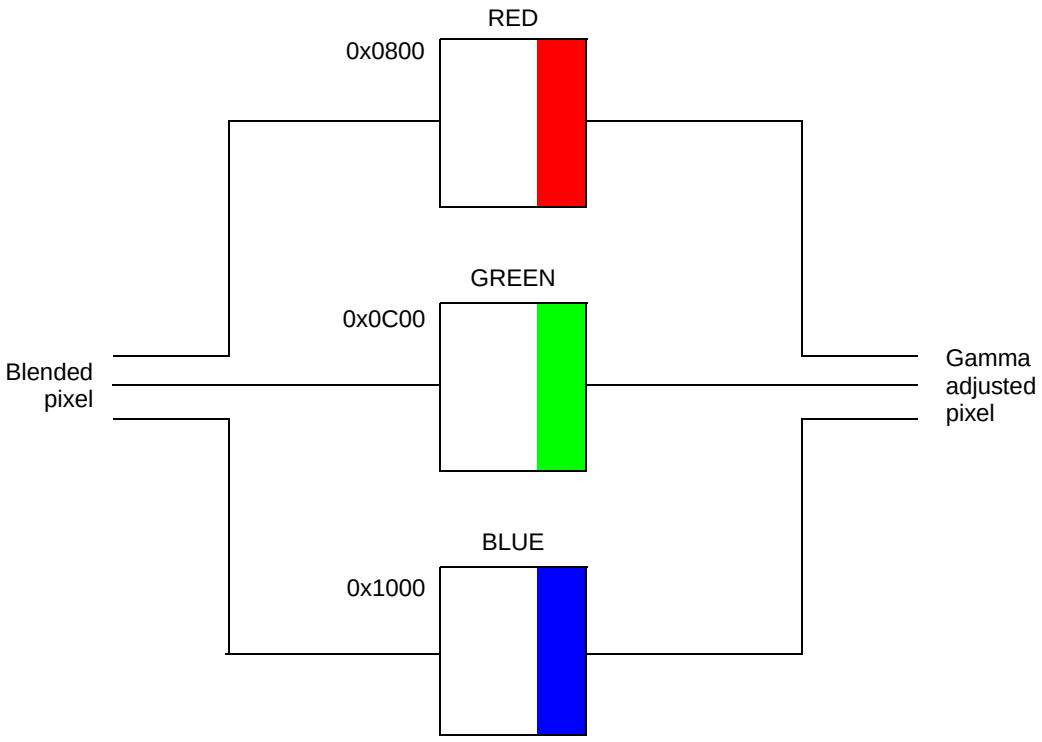



**Figure 11-83. Gamma Correction Table Organization**

The gamma table can only be read or written when the DCU3 is not enabled or during the vertical blanking period.

## 11.4.8 Temporal dithering

This is a technique that allows the emulation of a color resolution higher than the resolution supported by the display. It is done by changing the intensity values over time sent to the display. The averaging done by the human eye gives the impression of the intensity of such alternating pixels as an interim value between the two supported intensity values. Temporal dithering is enabled by the DCU_MODE[DITHER_EN] bit.

The key features of the dithering block are

- Temporal dithering increases the optically perceived depth of a limited TFT display
- Supports display with 5-8 bits resolution per color component
- Independent dither control parameters per color component
- Support for safety mode

Temporal dithering is enabled and controlled by the EN_DITHER, ADDB, ADDG, and ADDR bits in the DCU_MODE register. The ADDx fields are each 2 bits wide and select how many bits to add to each color component. The typical setting is 8 minus the number of bits in each component required by the display.

The Random Number Generator (RNG) provides a random number of up to 3 bits. The number of bits provided is selected by the values of each component's ADDx field.

When pixels from a safety layer are encountered the RNG output is forced to 0 which effectively disables the temporal dithering block and these pixels are passed to the display unmodified.

The Add & Clamp block adds the eight bit pixel value to the three bit number generated by the RNG. The result is then clamped to the range of 0..255.

### 11.4.9 Special DDR mode

Special DDR mode is a special configuration that optimizes the use of an SDRAM memory by the DCU3 by forcing the DCU3 to fetch data in optimal chunks. In this special mode only the four highest priority layers (0:3) are available in the DCU3. This mode is enabled using the DCU_MODE[DDR_MODE] bit.

When this mode is enabled the DCU3 will fetch data in 32-byte chunks if the layer is encoded in 8, 16 or 32 bpp formats, thus optimizing the SDRAM throughput. Any layers in 1, 2, 4 or 24 bpp formats will be fetched using normal access thus they will not benefit from any optimization and may disrupt optimal access for any 8-, 16- and 32-bit formatted layers if both are in the SDRAM. Therefore it is highly recommended to store any 1, 2, 4 or 24 bpp layers in non-SDRAM memory such as on-chip SRAM or flash.

Depending on the layer configuration in use this mode may also benefit other synchronous memory interfaces such as QuadSPI.

This mode only permits operation in four-layer blend mode, therefore, the DCU_MODE[BLEND_ITER] field must be set to 4.

### 11.4.10 Run Length Encoding (RLE) mode

RLE mode is available on the two highest priority layers (layer 0 and layer 1) and allows the DCU3 to load RLE compressed data from memory and directly decode it for use on the panel. The mode is enabled using the RLE_EN bit in the control descriptor 4 register CTRLDESCLn_4.

This mode is only available when Special DDR mode is enabled and can only be used on a single layer at a time. In addition the mode only supports 8 bpp, 16 bpp (RGB565, ARGB1555, ARGB4444, APAL8), 24 bpp and 32 bpp BGRA8888 formats.

When enabled, the DCU3 fetches data in 3-byte chunks and decompresses it before sending the decoded data to the normal DCU blend process.

The dataflow for the RLE decoded data is as shown in Figure 11-84.

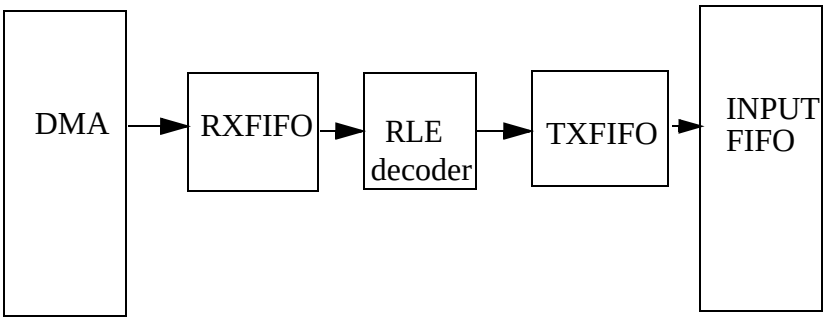



**Figure 11-84. RLE Decoding in DCU3**

The decoded data is read by the input FIFO once at least 8 bytes are available in the TXFIFO.The size of RXFIFO is 64x8 bits while the size of TXFIFO is 16x8 bits.

If both layer 0 and layer 1 have RLE_EN set, then the error flag RLE_ERR is asserted.

### 11.4.10.1 RLE decoding scheme

Before enabling an RLE encoded layer configure the COMP_IMSIZE register with the size of the compressed image. The decoder expects to read COMP_IMSIZE bytes from the image and produce from that the number of pixels specified in the layer Control Descriptor 1 register.

The format of RLE encoded layers is as follows:

- The first data byte read at the image address (Layer Control Descriptor 3) is a command byte (CMD[7:0])
- The ms bit (CMD[7]) indicates if the following bytes are raw or compressed pixels. One pixel can be 8-bit, 16-bit, 24-bit or 32-bit wide, depending on the BPP bitfield in the Layer Control Descriptor 4 register.
- The remaining 7 command bits (CMD[6:0]) specify the number of raw or compressed pixels that follow the command byte. This count is offset by 1 such that a value of 0 means one pixel follows.
- For compressed pixels (CMD[7] = 1), only one pixel follows the command byte. This pixel is repeated count+1 times on the layer. The pixel size may be 8, 16, 24 and 32 bits.
- For raw pixels (CMD[7] = 0) count+1 pixels follow the command byte and these are included on the layer as is. The pixel size may be 8, 16, 24 and 32 bits.
- If there is more data to decode then a new command follows after CMD+(1*{Pixel width}) bytes. This encoding is repeated until the whole image is decoded.

## 11.5 Timing, Error and Interrupt Management

The DCU3 can detect and raise status and error flags when the status of the system changes and when configuration or operational errors are detected.

### 11.5.1 Synchronizing to panel frame rate

Since the DCU3 fetches data directly from memory independently of the CPU, there is the possibility that changes to the DCU3 layer configuration or content can create incoherent content on the panel. To help avoid this situation there are five timing control flags that define when the DCU3 recognizes and locks changes to its configuration. These can be used to manage changes to control descriptors, CLUT or tile memory, or source graphics and so avoid coherency problems on the panel. All the timing flags are in the INT_STATUS register and can be used to generate interrupts from the DCU3.

The DCU3 configuration is completely open during the vertical blanking period and control descriptors and some other registers may also be programmed at any time. The configuration that is present one HSYNC before the end of the vertical blanking period is the configuration used by the DCU3 for the panel refresh phase.

The VS_BLANK and LS_BF_VS flags give indication of the start of the vertical blanking period. The VS_BLANK flag is set at the beginning of the vertical blanking period. The LS_BF_VS flag is set a given number of horizontal lines before the start of the vertical blanking period; the given number of lines is defined by the LS_BF_VS bit field in the THRESHOLD register.

The PROG_END flag indicates that the DCU3 has locked the contents of its configuration registers for the new panel refresh period. No further changes are accepted to the DCU3 configuration after this flag is set (until the next vertical blanking period).

The DMA_TRANS_FINISH flag indicates that the DCU3 has completed fetching all data from memory in the current panel refresh cycle. This normally precedes the vertical blanking period and indicates that it is possible to change the contents of a memory that contains graphics used by the DCU3.

The VSYNC flag indicates that the DCU3 has begun the next panel refresh period.

### 11.5.2 Managing the DCU3 FIFOs and DMA activity

The DCU3 fetches graphic data directly from internal and external memory using a dedicated DMA system and manages the output of data to the TFT LCD panel such that the panel always receives the pixel information when expected. Since the panel is sharing access to memory with the system DMA and CPU it cannot depend on the required data always being available at all times. It therefore uses input and output FIFOs to temporarily store incoming and outgoing data until required and thus reduces the opportunity for the panel to be starved of pixel data.

The DCU3 manages the supply of graphic data to its format conversion and blending stages using input FIFOs that are 256×64 bits in size. The data that is driven to panel is managed using an output FIFO that is 128 pixels in size. See Figure 11-1 for a diagram of the input FIFO and output FIFO operation in the DCU3.

The input FIFOs are not accessible to the user but it is possible to set thresholds that control the DCU3 behavior when the FIFOs are becoming full or empty and observe when the lower and higher thresholds are reached. This can help detect and avert situations where the DCU3 is running out of data to send to the panel.

The FIFO thresholds are set in the THRESHOLD_INPUT_BUF_1/2 registers. The upper thresholds are set by the INP_BUF_Pm_HI bit fields (where m is the position of the pixel in the blend stack) and these set the point at which the DCU3 pauses fetching data from memory. The maximum size of any DMA burst is fixed to 16 pixels and so is dependent on the graphic encoding. The lower thresholds are set by the INP_BUF_Pm_LO bit fields.

Each of the four input FIFOs has two flags that indicate whether the FIFO has reached its upper or lower threshold. The Pm_FIFO_HI_FLAG flags (where m is the position of the pixel in the blend stack) indicates that the input FIFO has reached the upper threshold. The Pm_FIFO_LO_FLAG indicates that the input FIFO has less data than its low threshold. Depending on when the low threshold is reached this may indicate a number of scenarios

- The expected graphical data is not available for the DCU3 to load
- The DCU3 is reaching the end of a frame and does not need to load any more data
- The blend stack does not need pixels of this priority

In the situation where the data is not available to the DCU3 then there may or may not be an impact to the data visible on the panel. In the situation where the output FIFO is full then it is possible for the DCU3 to accept a delay before it requires to use the incoming data.

The output FIFO is not accessible to the user but it is possible to set thresholds that control the DCU3 behavior when the FIFO is becoming full or empty and observe the lower threshold. This can help detect and avert situations where the DCU3 is running out of data to send to the panel.

The buffer thresholds are set in the THRESHOLD register. The upper threshold is set by the OUT_BUF_HIGH bit field and this indicates that sufficient data exists in the output buffer and processing should stop until the DCU3 uses some of the values in the FIFO. If this value is set too low then the possibility of the DCU3 running out of data to drive the panel is increased. The lower threshold is set by the OUT_BUF_LOW bit field.

When the output FIFO has emptied below its low threshold (OUT_BUF_LOW bit field) it sets the UNDRUN bit. In an under run situation there may or may not be an impact to the data visible on the panel. The impact depends on whether the DCU3 is reaching the end of a frame and how close to running out the threshold is set.

The best guide to indicate whether the DCU3 is able to supply the required pixel information to the panel is the output buffer. If the output is indicating that it is running out of data then the input FIFOs may help identify the areas of memory that are restricting the supply of data. Using these indicators can help to set the DCU3 thresholds and ensure that the data throughput on the MCU is balanced correctly for all master devices.

Finally, note that the number of DCU3 clock cycles to fetch and blend each pixel increases with the depth of the blend stack. However, the time taken to process the pixel data is fixed by the timing requirements of the panel. Therefore, for full performance across all color encodings the ratio between the DCU3 clock and the pixel clock must increase as the blend stack depth increases:

- For two-pixel blending, the minimum DCU clock is the same as the TFT pixel clock.
- For three-pixel blending, the minimum DCU3 clock must be twice the TFT pixel clock.
- For four-pixel blending, the minimum DCU3 clock must be thrice the TFT pixel clock.

### 11.5.3 Error detection

The DCU3 asserts error flags when errors are detected in its configuration or when the user attempts to modify the configuration at an invalid point in the panel refresh period or when it is unable to access the required source data. The error flags may raise an interrupt if enabled to do so by the related mask bit in the corresponding mask register.

Error flags are stored in the PARR_ERR_STATUS and INT_STATUS registers.

Errors in the DCU3 configuration are collected in the PARR_ERR_STATUS register. The flags Ln_PARR_ERR (where n is the layer number) indicate an error in the configuration of the layer which can be either an invalid tile mode size or a layer with a horizontal dimension that is smaller than the minimum size defined by the layer encoding (see Section 11.4.4.3, “Layer size and positioning””). The DISP_ERR flag indicates that the VSYNC and HSYNC pulse widths are configured to the invalid value of 0. The HWC_ERR flag indicates that the hardware cursor is either larger than the available memory or is placed in an off-panel position. The SIG_ERR indicates that the signature calculation specifies an area that extends beyond the panel size. The RLE_ERR indicates that more than one layer has RLE enabled.

Reads of CLUT/Tile RAM during the period when the TFT LCD panel is being updated do not return the CLUT/Tile RAM content.

Errors caused when the DCU3 is unable to access its required source data are collected in the INT_STATUS register. These errors are indicated by the UNDRUN flag and the Pm_FIFO_LO_FLAG flags (where m is the position in the blend stack)

### 11.5.4 Interrupt generation

The DCU3 generates interrupt through four lines that are controlled by the contents of six registers:

- INT_STATUS
- INT_MASK
- PDI_STATUS
- MASK_PDI_STATUS
- PARR_ERR STATUS
- MASK_PARR_ERR STATUS

There are four interrupt status lines defined.These lines are grouped as follows

- Timing based interrupts:
  - VSYNC
  - LS_BF_VS
  - VS_BLANK
  - PROG_END
  - DMA_TRANS_FINISH
- Functional interrupts:
  - UNDRUN
  - CRC_READY

- CRC_OVERFLOW
- P1_FIFO_HI_FLAG
- P1_FIFO_LO_FLAG
- P2_FIFO_LO_FLAG
- P2_FIFO_HI_FLAG
- P3_FIFO_HI_FLAG
- P3_FIFO_LOW_FLAG
- P4_FIFO_HI_FLAG
- P4_FIFO_LOW_FLAG
- IPM_ERROR

- Parameter error interrupts
  - Layer Error
  - Signature Calculator Error
  - Display Error
  - HWC_error
  - RLE error
- PDI-related interrupts (pdi_int)
  - This includes PDI related interrupts. See Section 11.8.1.8, “PDI-related Interrupts”,” for a description.

When any interrupt occurs, the host can identify which type of interrupt has occurred by reading the interrupt status register/PDI status register/PARR_ERR status register.

## 11.6 Register protection

There is a customized register protection scheme on the DCU3 that is different to the protection scheme implemented elsewhere on the MCU. The scheme provides a mechanism to protect certain registers in the DCU3 from being written.

### 11.6.1 Operation of scheme

The register protection scheme provides a two-step protection scheme for the protected register.

Firstly, each register has an associated soft lock bit (SLB) that prevents further writes to the register when it is set. Each SLB has a corresponding write enable (WEN) bit that must be set in the same write operation as the SLB. The SLB can be set or cleared by writing a '1' or '0' to it while its WEN bit is set. The SLB bits are in the Soft Lock Registers L0 and L1, DISP_SIZE, HSYNC/VSYNC_PARA, POL, L0_TRANSP and L1_TRANSP registers.

Secondly, there is a hard lock bit (HLB) in the Global Protection Register which prevents all changes to soft lock bits. The HLB can only be cleared by a system reset.

If a write is made to a register whose SLB is set then a transfer error occurs that generates an IVOR1 exception on the CPU. Similarly if the HLB is set then any write to the SLB registers causes a transfer error.

### 11.6.2 List of protected registers

The register protection scheme applies to the following registers:

- All Layer 0 control descriptors CTRLDESCL0_1 to CTRLDESCL0_7
- All Layer 1 control descriptors CTRLDESCL1_1 to CTRLDESCL1_7
- Layer 0 foreground and background registers for transparency mode FG0_FCOLOR and FG0_BCOLOR
- Layer 1 foreground and background registers for transparency mode FG1_FCOLOR and FG1_BCOLOR
- All Control Descriptors & Transparency Registers for Layer1
- DISP_SIZE
- HSYNC_PARA
- VSYNC_PARA
- SYN_POL

## 11.7 Safety Mode

Safety layers are used in a multi-layer DCU3 environment for the purpose of guaranteeing that the content is driven to the display regardless of the setting of remaining layers and the pixel manipulation algorithms of the DCU3. Features such as this are a requirement from qualification institutes to be able to reach a safety level of SIL2 or ASILB. The DCU3 has two safety layers (Layer 0 and Layer 1) which also have the highest priority. When Safety Mode is active the safety layers can use chroma keying for complex area description, however alpha blending for the layer is always ignored. Additionally, if a layer has safety mode enabled then a layer format of 32 bpp or luminance is not allowed. Using these formats causes the layer to be disabled.

Safety Mode is implemented using a signature calculator module implemented inside the DCU3 that calculates two signatures (pixel value and pixel position) for a predefined area of the frame. The user makes layer 0 and/or layer 1 active as a safety layer, defines the window/area of the pixels for which the signature is to be calculated, and enables safety mode. When enabled, the signature calculator starts to calculate the signature after the first pixel in the selected area is available and after the start of the next frame (VSYNC). It is also possible to calculate the signature value for all pixels if DCU_MODE[TAG_EN] = 0.

As the pixels in the selected area become available they are "tagged" by the DCU3, except for those removed by chroma-keying. These tags identify the pixels to be included in the signature calculation. The signature calculation itself is an industry-standard CRC.

The DCU3 asserts the CRC_READY flag at the end of any frame which has Safety Mode enabled. This can be used to indicate the completed signature calculations for each full frame of pixels after the mode is enabled. The completed signature can then be compared against a pre-calculated value with any difference

indicating that the pixels displayed did not match what was expected. The signature calculator then continues to calculate the CRC for the next frame. If the CRC_READY flag is not processed within one frame time period, then the CRC_OVERFLOW interrupt is issued and the latest calculation overwrites the previous value. Since the CRC_READY flag is set at the end of any frame with Safety Mode enabled, it is possible that a full frame has not yet been completed and therefore no signature calculation exists for the frame that set the flag.

If the user has set the NEG bit for the DCU3 which indicates that the pixels fed to the display are inverted, then the value CRC is calculated on non-inverted values. The position CRC, however, remains as is.

Normal arbitration takes place only when a pixel has content on layer 0 and layer 1 but where Safety Mode is enabled in layer 1.

The polynomial used for CRC calculation is

$$(x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1) \quad \textit{Eqn. 11-5}$$

For the value CRC, the 24-bit value of each output pixel (after decoding) is sent to the polynomial. For the position CRC, the value sent is

$$\text{(pixel_delta_y * display.delta_x) + (pixel_delta_x + 1)} \quad \textit{Eqn. 11-6}$$

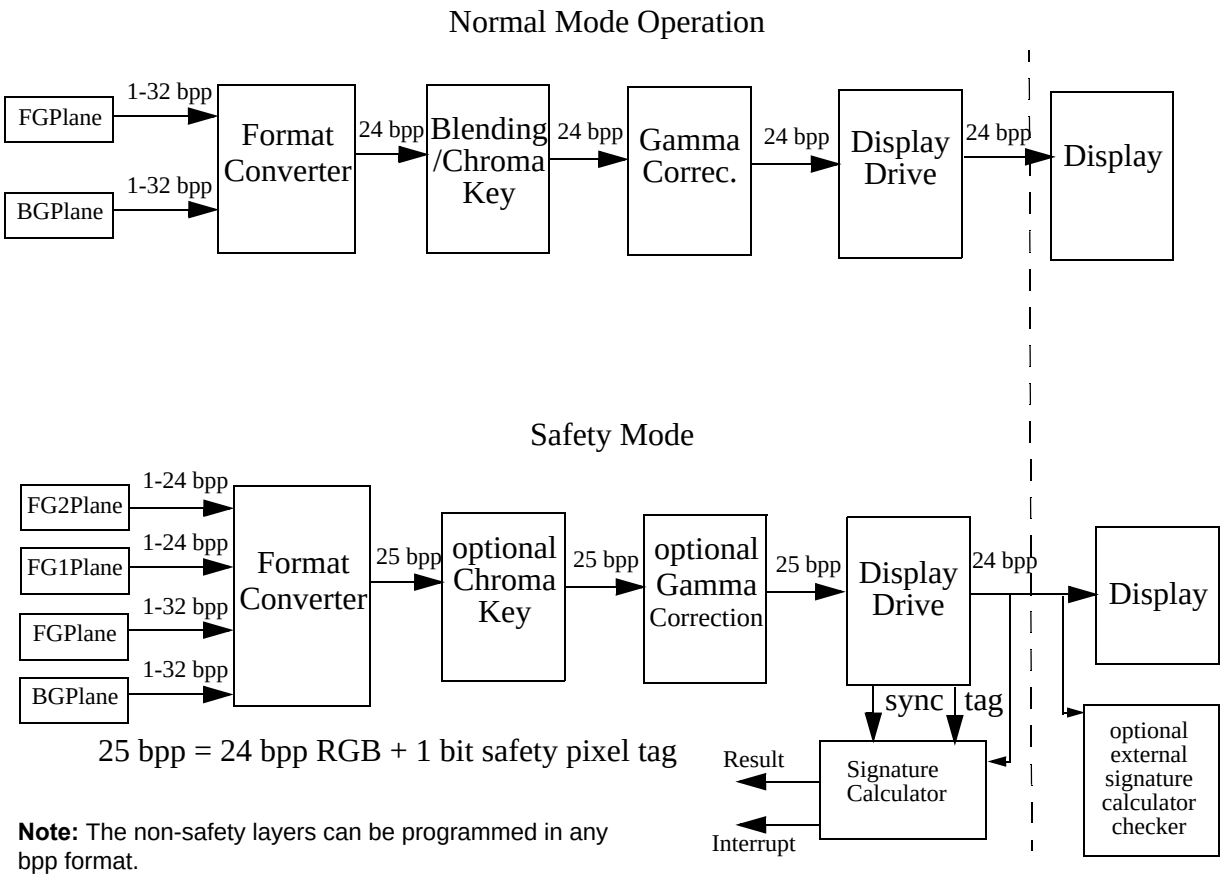



**Figure 11-85. Safety Mode block diagram**

## 11.7.1 CRC area description

### 11.7.1.1 Configuring the CRC calculation

1. CRC_VAL and CRC_POS are calculated when Safety Mode is enabled using DCU_MODE[SIG_EN].
2. The CRC can be calculated for part of the panel as shown in Figure 11-86. The green portion on the panel is identified using the SIGN_CALC_1 and SIGN_CALC_2 registers which defined the size of the area and the location of the area respectively. If SIGN_CALC_1 and SIGN_CALC_2 are configured appropriately this calculation will cover the whole panel.

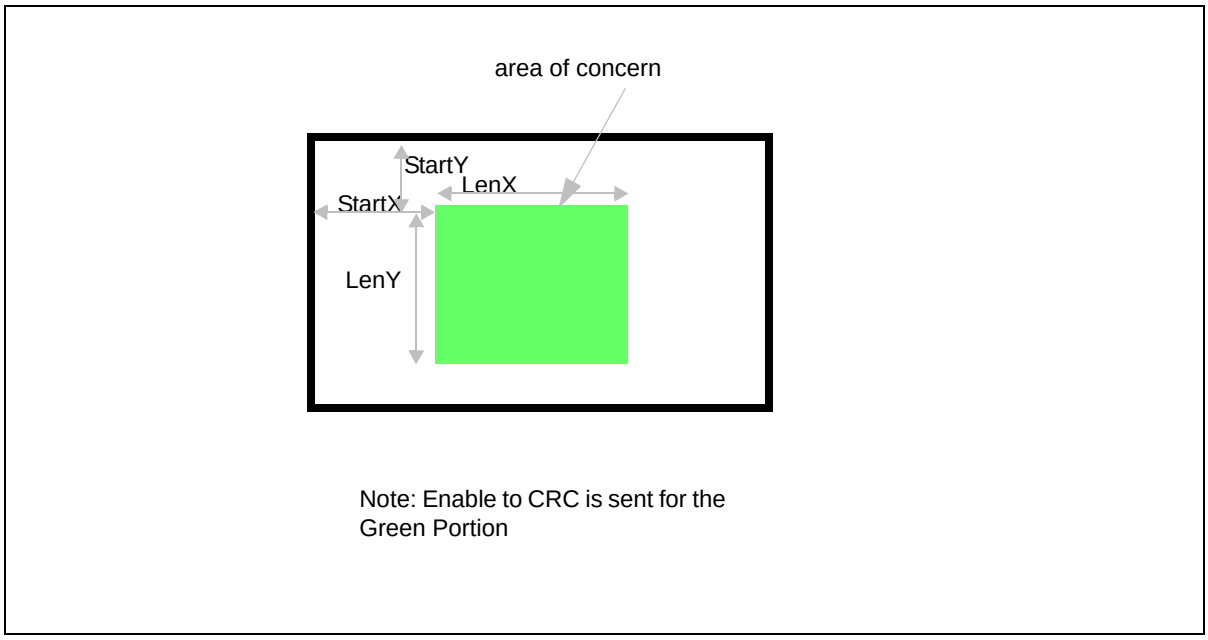



**Figure 11-86. Safety Mode enabled for part of the screen**

3. The CRC can be calculated exclusively for layers 0 and 1 by setting DCU_MODE[TAG_EN] and enabling the SAFETY_EN bit in control descriptor 4 of each of the layers. In this configuration the CRC is calculated using values from the layers only where they intersect the area of interest defined in SIGN_CALC_1 and SIG_CALC_2. An example is shown (in dark pink) in Figure 11-87.

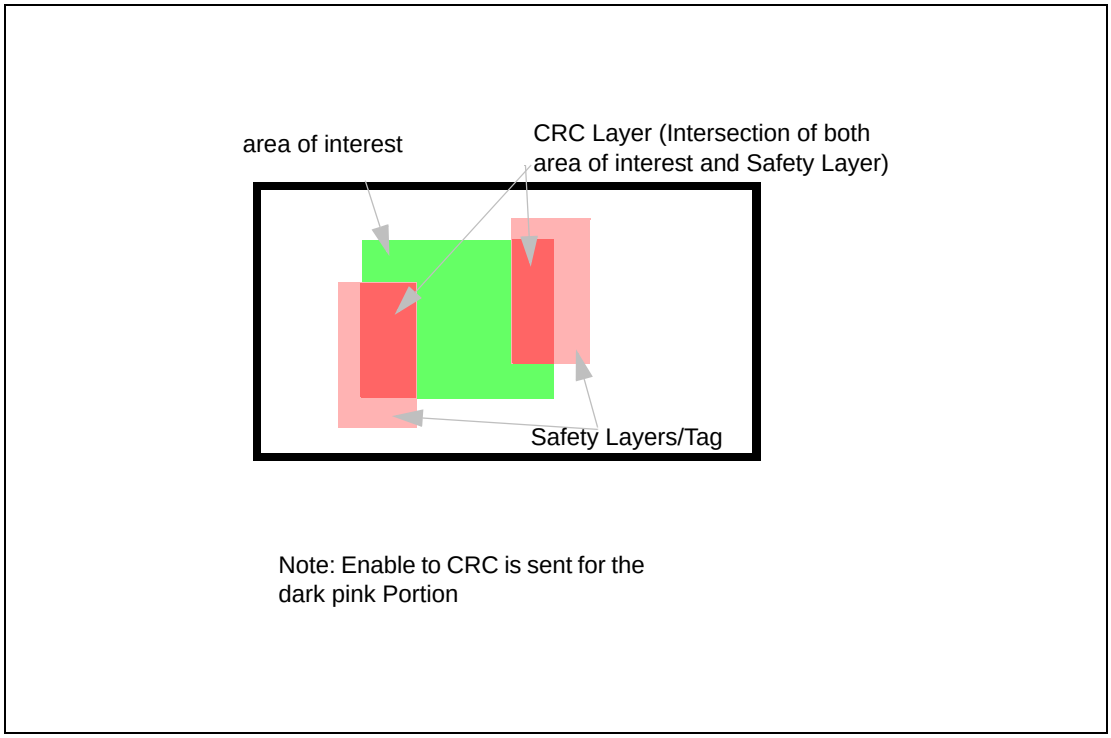



**Figure 11-87. Safety Mode with DCU_MODE[TAG_EN] = 1**

## 11.7.2 Summary of operation

The area included in the CRC calculation is summarized in Table 11-75. The initial value on all CRC calculations is 0x00000000.

**Table 11-75. Supported area**

| Area DCU_ | MODE[TAG_EN] | Note |
|---|---|---|
| Full | 0 | SIGN_CALC_2[SIG_HOR_POS] = 0<br>SIGN_CALC_2[SIG_VER_POS] = 0<br>SIGN_CALC_1[SIG_HOR_SIZE] = Panel width<br>SIGN_CALC_1[SIG_VER_SIZE] = Panel height |
| Part | 0 | The SIGN_CALC parameters have values other than those mentioned above — see Figure 11-86 |
| Safety Layer<br>(Layer 0 and 1 only) | 1 | The included portion of the safety layers depends on:<br>• The portion lying within the area defined by SIGN_CALC_1 and SIGN_CALC_2<br>• The pixels removed by chroma keying functionality |

## 11.8 Parallel Data Interface (camera interface)

### 11.8.1 PDI interface description

#### 11.8.1.1 Introduction

This block extracts the timing and pixel information from an external video source and passes it to the DCU3 block to synchronize with the timing and display the pixel information on the TFT LCD screen. The BGND layer in the DCU3 is replaced by the incoming video stream.

The PDI requires that the incoming video stream match the resolution and timing at which the DCU3 is driving the TFT LCD panel and the incoming stream must not be interlaced.

The PDI can also be configured in slave mode in which case it ignores the pixel information from the external video source and only passes the timing information to the DCU3 to synchronize with. In this instance the DCU3 will continue to operate as normal (the BGND layer will not be altered) but will use the timing from the PDI.

The PDI shares configuration registers with the DCU3.

#### 11.8.1.2 PDI interaction with other modules

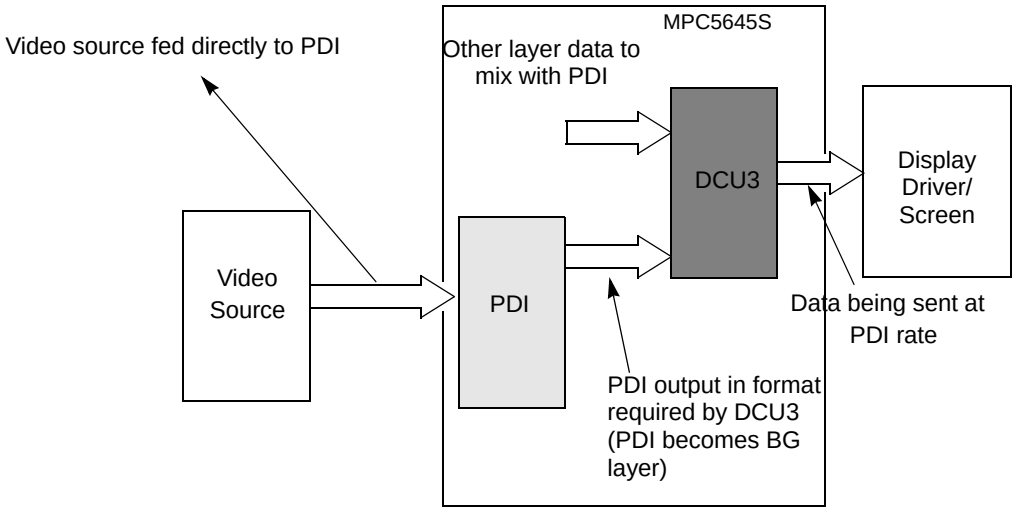



**Figure 11-88. PDI interacting directly with the external sensor**

In Figure 11-88, the PDI accepts data and timing directly from the external video source. The external device must output the video in a digital format supported by the PDI.

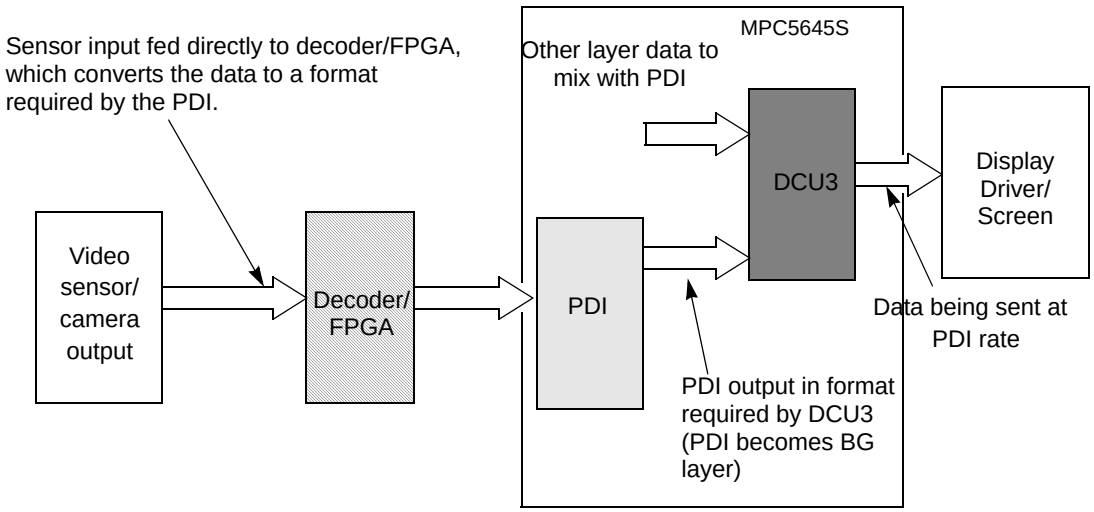



**Figure 11-89. PDI interacting with FPGA in between**

In the case shown in Figure 11-89, the video stream is sent to a decoder or an FPGA which alters the incoming stream to a format which is compatible with the PDI. A decoder/FPGA is required if a video source with an analog output format (e.g. NTSC/PAL) is used. The decoder should perform analog-to-digital conversion on the stream and ensure the timing match that of the DCU3. The incoming stream may also need to be de-interlaced.

The PDI is compatible with various input formats:

- Normal Mode: the PDI clock frequency must be equal to pixel clock frequency required by the TFT display driver
- Narrow Mode: the PDI clock frequency is double the desired pixel clock frequency.
- External Synchronization: The timing signals (HSYNC, VSYNC, and DE) each have a dedicated input pin. (DE is optional)
- Internal Synchronization: The timing signals (HSYNC, VSYNC and DE) are embedded in the data stream, as such only the DATA and PCLK inputs are required. (See Section 11.8.1.4, “ITU-R BT.656 sync information extraction””.)

Before the DCU3 locks onto the PDI timing signals it will run on the internal DCU3 clock. After lock has been achieved, the DCU3 will switch to the clock from the PDI stream and the this is then used to send data and timing signal to TFT/LCD display driver.

In all cases, the resolution of the incoming stream and the HSYNC and VSYNC frequency must be the same as that for TFT screen. All the horizontal parameters (Front Porch width, Back Porch width, Pulse width) and vertical parameters (Front Porch width, Back Porch width, Pulse width) must be the same as that of TFT screen.

When PDI is the background layer, no other layer can be a background layer for that particular frame. Only one background layer is possible i.e. PDI Layer when PDI is enabled.

### 11.8.1.3 Features

The PDI supports the following:

- RGB565, RGB666, 8 bit monochrome format,YCbCr422 mode
- Max input frequency of 32 MHz in 8/16/18 normal mode input
- Max input frequency of 64 MHz in 8 bit muxed (narrow) mode
- External Synchronization using PDI_HSYNC, PDI_VSYNC, and PDI_CLK
- External Synchronization using PDI_HSYNC, PDI_VSYNC, PDI_PCLK, and PDI_DE
- Internal synchronization using PDI[17:0] and PDI_PCLK. It is supported for RGB565 and YCbCr422 muxed mode only

**Table 11-76. Supported RGB format and Sync Format**

| RGB Format | Data Input Bus |
|---|---|
| 8-bit monochrome | 8 bit |
| RGB565 | 16 bit |
| RGB666 | 18 bit |
| RGB565 muxed (uses narrow mode) | 8 bit |
| YCbCr (uses narrow mode) | 8 bit |
| **Sync Format** | **Pin Used** |
| Internal Sync (Valid only for RGB565and YCbCr in narrow mode) | Pclk |
| External Sync | PDI_HSYNC, PDI_VSYNC, PDI_PCLK |
| External sync (with data En) | PDI_HSYNC, PDI_VSYNC, PDI_PCLK, PDI_DE |

### 11.8.1.4 ITU-R BT.656 sync information extraction

According to ITU-R BT.656 recommendation, the incoming digital video will have a pdi_clk signal and 8 data bits. The data bits can contain both the video data and the timing reference signals (VSYNC and HSYNC).

The timing signals are encoded at the start and end of each line by timing reference codes known as Start of Active Video (SAV) and End of Active Video (EAV). The SAV and EAV codes are identified by their preamble of three bytes (0xFF,0x00,0x00). Due to this, neither 0x00 or 0xFF can be used during the Active Video Data. The preamble is followed by the XY status word which contains a Field Bit (F), a Vertical blanking bit (V) and Horizontal blanking bit (H) and four protection bits for single bit error correction and detection.

The H bit is set to 1 to denote an EAV — that is the end of a line, and the beginning of the horizontal blanking period. The V bit is set to 1 to denote the beginning of the vertical blanking period. The F bit is used for interlaced video to denote if the forthcoming line is odd or even.

The remaining 4 bits contains make up the protection bits for single bit error correction and detection. It should be noted that F and V fields are only allowed to change as part of EAV sequences i.e during transitions from H=0 to H=1.

An entire line of video comprises Active Video + Horizontal Blanking (from the start of the EAV code until the end of the SAV code) and Vertical Blanking (the space where V = 1).

**NOTE**

This device supports only 8-bit, non-interlaced, video. The Field (F) value is ignored.

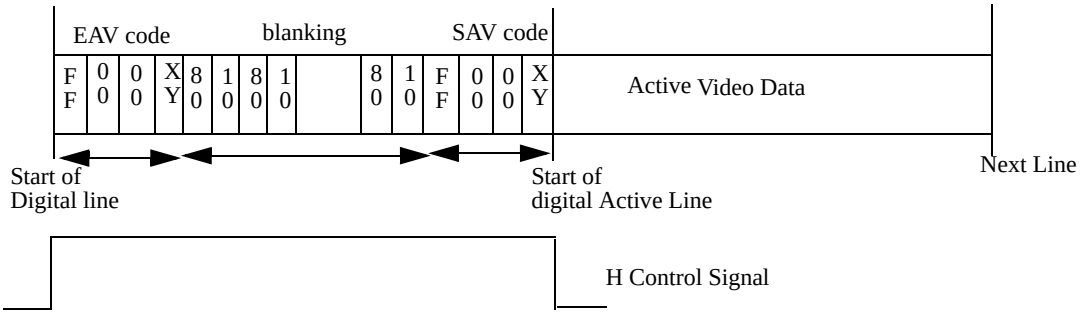



**Figure 11-90. ITU-R BT.656 8 bit parallel data format for 525 video system**

| Data Bit | FirstWord (FF) | SecondWord (00) | ThirdWord (00) | FourthWord (XY) |
|---|---|---|---|---|
| D7(MSB) | 1 | 0 | 0 | 1 |
| D6 | 1 | 0 | 0 | F |
| D5 | 1 | 0 | 0 | V |
| D4 | 1 | 0 | 0 | H |
| D3 | 1 | 0 | 0 | P3 |
| D2 | 1 | 0 | 0 | P2 |
| D1 | 1 | 0 | 0 | P1 |
| D0 | 1 | 0 | 0 | P0 |

**Figure 11-91. Control Byte Sequence for 8-bit/10-bit video**

The bit definitions for the status word XY are shown in Table 11-77.

**Table 11-77. Status word definitions**

| Bit | Definition |
|---|---|
| F | 0 for field 0<br>1 for field 1 |
| V | 1 during vertical blanking period<br>0 when not in vertical blanking |
| H | 0 at SAV<br>1 at EAV |

Table 11-77. Status word definitions (continued)

| Bit | Definition |
|---|---|
| P3 | V XOR H |
| P2 | F XOR H |
| P1 | F XOR V |
| P0 | F XOR V XOR H |

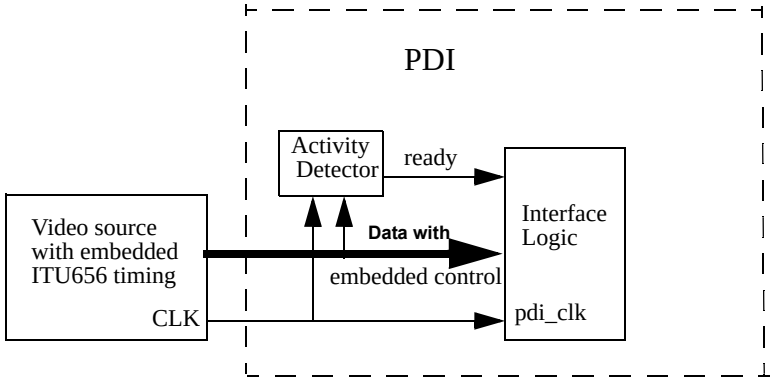



Figure 11-92. PDI Input data mode

Figure 11-92 represents the scenario in which data from an ITU-R BT.656 compliant video source is fed into the PDI interface. The incoming data includes codes that trigger the start and end of the active video and blanking fields. An activity detector checks for the transitions on the PDI bus. It samples the values on the PDI bus and once it has detected valid activity, sets a flag in the status register and can optionally trigger an interrupt. The PDI interface has a state machine which extracts the control information from the video data. The machine checks the video data for the Preamble Field (0xFF,0x00,0x00) and then depending on the status bits XY decides if it has received a valid control signal.

### 11.8.1.5 Normal and Narrow Mode

In normal mode, the PDI supports a maximum input frequency of 32 MHz. In narrow mode, the PDI supports a maximum input frequency of 64 MHz.

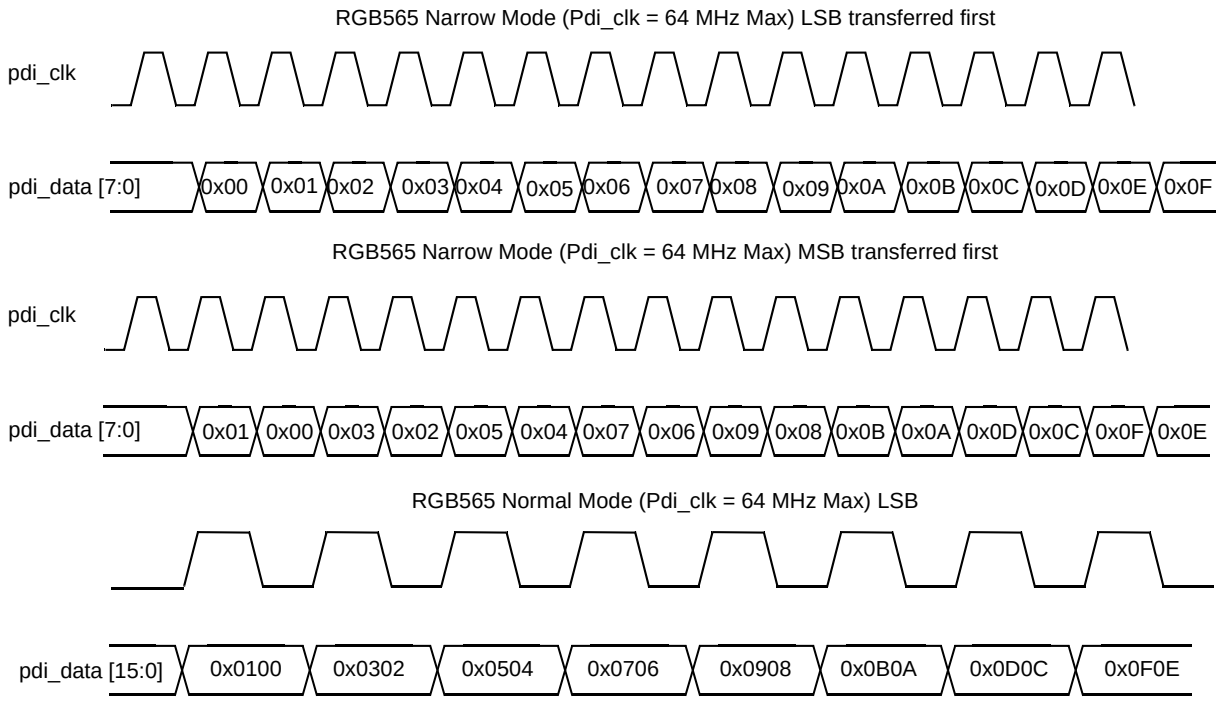



**Figure 11-93. Data Transfer in Normal and Narrow Mode**

The byte transferred first (MSB or LSB) depends on the configuration register as shown in Figure 11-93.This does not affect the sync preamble sequence in case internal sync mode.

On this device, the incoming RGB data is mapped onto the PDI pins as described in Table 11-78.

**Table 11-78. Mapping of RGB data onto PDI pins**

| Mode | Mapping |
|---|---|
| Normal (full 18-bit PDI interface) | PDI[17:12] = R[5:0]<br>PDI[11:6] = G[5:0]<br>PDI[5:0] = B[5:0] |
| Normal (RGB565 16-bit PDI interface) | PDI[15:11] = R[4:0]<br>PDI[[10:5] = G[5:0]<br>PDI[4:0] = B[4:0] |
| Narrow (8-bit PDI interface) | RGB565:<br>In first clock cycle, PDI[7:0] = { R[4:0]}<br>In second clock cycle, PDI[7:0] = { G[2:0]}<br>YCbCr:<br>In first clock cycle, PDI[7:0] = { Cb[7:0] }<br>In second clock cycle, PDI[7:0] = { Y0[7:0] }<br>In third clock cycle, PDI[7:0] = { Cr[7:0] }<br>In forth clock cycle, PDI[7:0] = { Y1[7:0] } |

### 11.8.1.6 Modes of operation based on sync extraction

#### 11.8.1.6.1 PDI input data (external sync mode)

In external sync mode the timing signals (HSYNC, VSYNC and, optionally, Date Enable) are provided to the PDI input timing pins by the external video source.

External sync mode can be used in both normal mode and 8-bit narrow mode, but cannot be used in conjunction with the YCbCr data format. In the instance that external sync and narrow mode is selected, the external signals are used, and any timing information (EAV/SAV) embedded in the data stream is ignored.

As in Figure 11-95, PDI data enable (PDI_DE) should be low during VSYNC and HSYNC pulse, VSYNC front porch (FP_V) and back porch (BP_V), HSYNC front porch (FP_H) and back porch (BP_H). This is valid for Data Enable Mode when the PDI_DE_EN bit is set in the DCU_MODE register (i.e. mode with HSYNC, VSYNC, PDI_DE and PDI_PCLK as pin signals).

Pulse width, Front and back porch values should be picked from those programmed in DCU3 registers. In order to achieve lock, it must have same value as that of TFT screen. Front porch and back porch value can be zero. Pulse width and TFT screen size parameters cannot be zero. In case they are programmed as zero, it might lead to malfunctioning of the validation state machine.

As in Figure 11-94 HSYNC must occur during the VSYNC and vertical blanking period. The time between 2 HSYNC should be the same during VSYNC and vertical blanking as during the active line period. As in Figure 11-94 the positive edge of HSYNC and VSYNC should be aligned. As in Figure 11-94 the positive edge of the HSYNC and start of the vertical front/back porch should be aligned. The polarity of HSYNC and VSYNC is selectable.

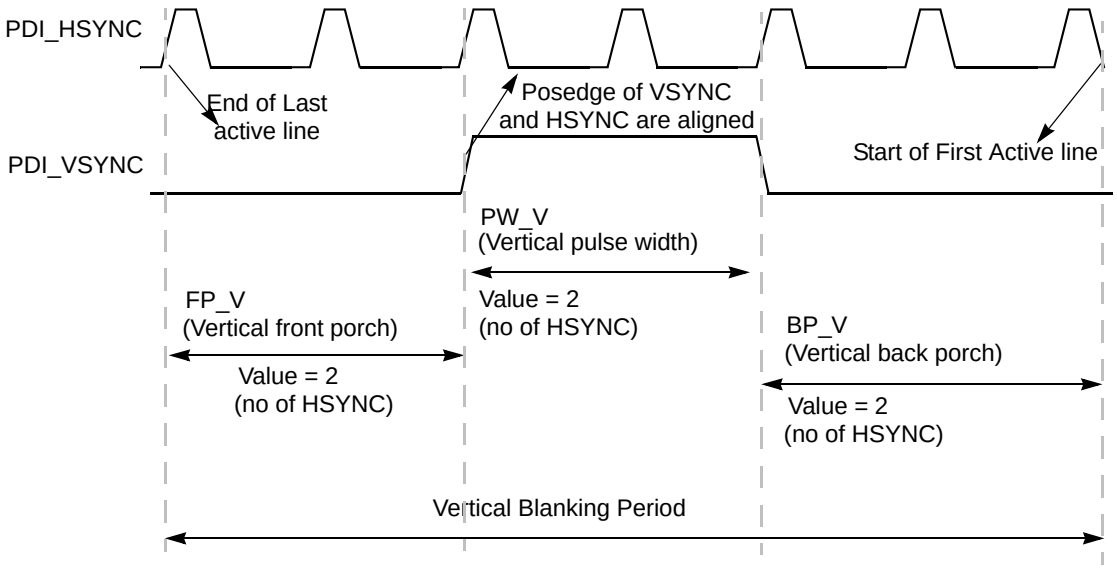



**Figure 11-94. Relationship between HSYNC and VSYNC in external synchronization**

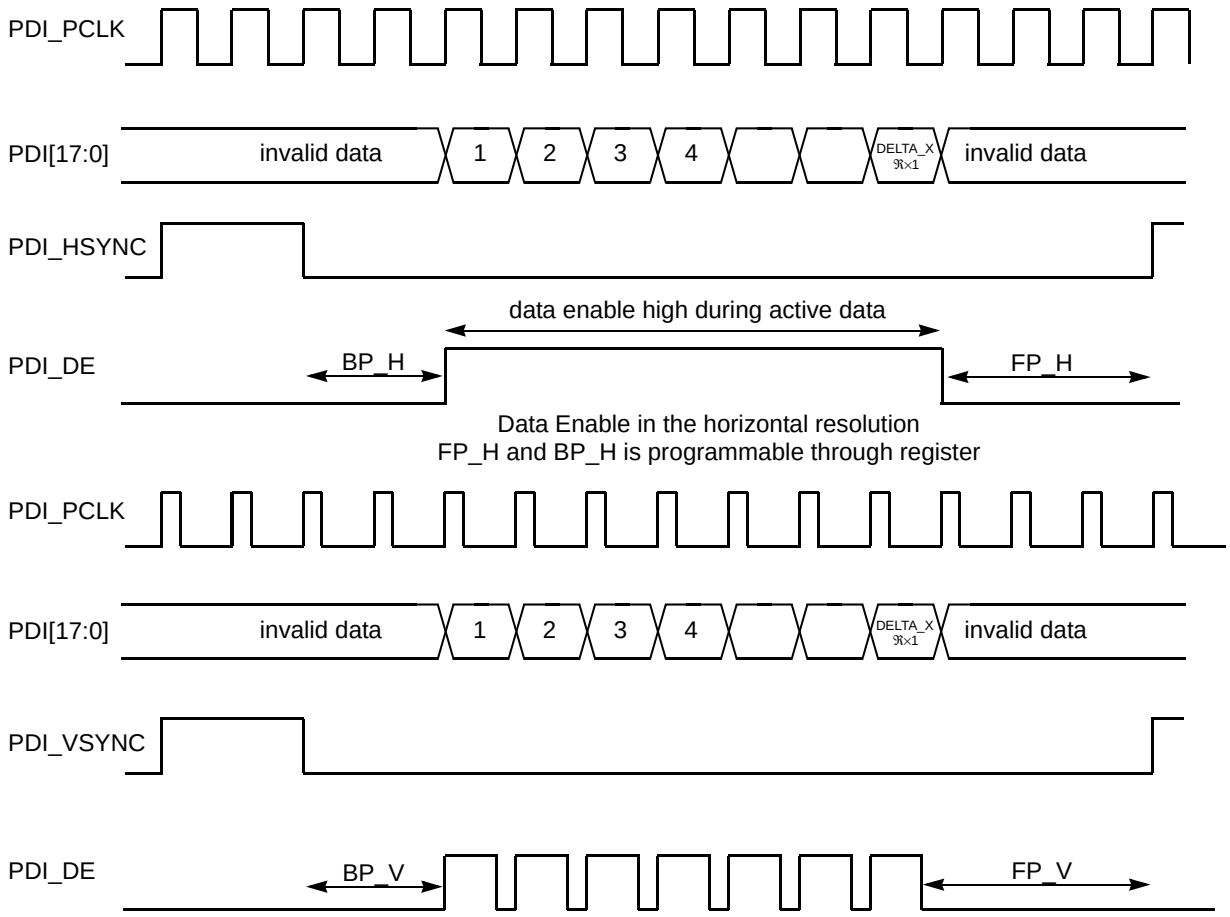



**Figure 11-95. Occurrence of HSYNC and Vsync and PDI_DE for the entire frame**

### 11.8.1.6.2 PDI input data (internal sync extraction mode)

In internal sync mode the timing parameters (horizontal and vertical blanking) are encoded into the data stream.

Internal sync mode can only be used in 8-bit narrow mode.

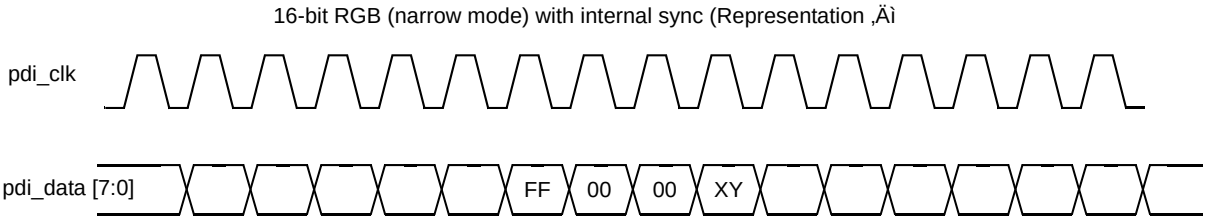
In Figure 11-96, XY is used to decode the vertical and horizontal blanking period.

**Table 11-79. XYh Value**

| Bit | Value | Description |
|---|---|---|
| 7 | 1'b1 | Always 1'b1. This is checked while decoding sync preamble |
| 6 | F | Not considered in the state machine logic |
| 5 | V | 1'b1 during vertical blanking<br>1'b0 elsewhere |
| 4 | H | 1'b0 for start of active video<br>1'b1 for end of active video |

**Table 11-79. XYh Value (continued)**

| Bit | Value | Description |
|---|---|---|
| 3<br>2<br>1<br>0 | P3<br>P2<br>P1<br>P0 | Protection bits (used to detect ECC errors). It would not be used for bit correction. |




**Note:** Preamble sequence is independent of data sequence

**Figure 11-96. Location of Sync Preamble in Narrow Mode**

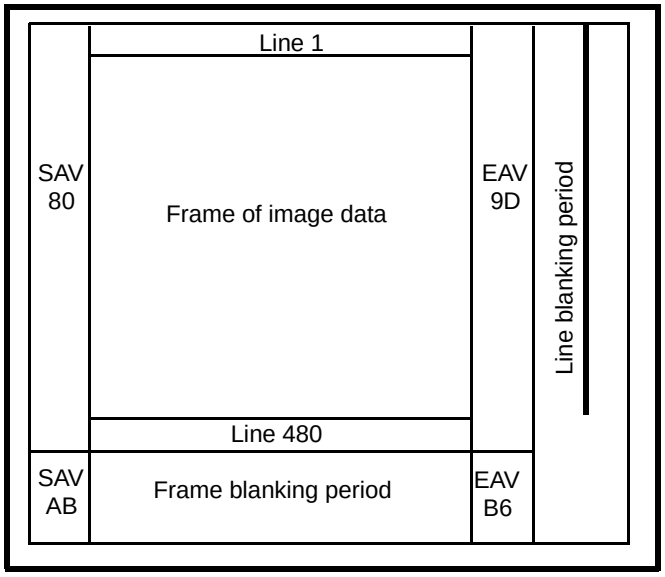
Sync Preamble would come continuously for 4 clock cycles as shown in Figure 11-96. It does not depend upon which byte is coming first in data (MSB or LSB). Sync extraction is done using PDI[7:0] to identify the horizontal and vertical blanking period using H and V field of the 'XYh' data as mentioned in Table 11-79.

ITU 656 Sync preamble pattern (FFh 00h 00h) has to be masked out in the RGB and YCbCr data. The data stream must not include FFh 00h 00h as the valid pixel data to avoid malfunction by the validation state machine.

Horizontal blanking period must continue during the Vertical blanking period. The gap between 2 horizontal blanking periods should be the same during vertical blanking period as during line active. All Vertical and horizontal parameter values are validated against the DCU3 registers programmed by the user. Polarity of HSYNC and VSYNC are selectable. Horizontal blanking and vertical blanking must be aligned as shown in Figure 11-97. During blanking period the input stream must be a 80h 10h 80h 10h sequence. This sequence must be present during both horizontal (line) blanking and vertical (frame) blanking.

As the PDI does not support interlaced video, the Framing Bit (F field in XYh) will be ignored during timing extraction. Narrow mode is compatible with RGB565 and YCbCr422 muxed modes. Each header contains an ECC value, which the PDI will check. The PDI is capable of detecting an error but not correcting it.

As with External sync mode, the value of front and back porch can be zero but the pulse width and TFT screen parameter cannot be zero.




NOTE: The SAV and EAV bytes are included as part of the blanking period.

**Figure 11-97. Relationship Between Hblank and Vblank in Internal Sync**

#### 11.8.1.6.3 PDI YCbCr mode and DCU3 YCbCr color format

The DCU3 can process incoming data from the PDI and from memory in YCbCr422 format. Both sources use the same RGB conversion and interpolation equations, however, the coefficients for the equations and enable for the interpolation are independently controlled.

In YCbCr mode, the PDI extracts the ITU656 sync (FF-00-00) and sends the video to the processing functions. The first processing function converts the 422 stream to a 444 stream, by providing interpolation on the chroma components of the stream depending on PDI_INTERPOL_EN bit. The second processing function converts the stream to RGB888/RGB565.

The DCU3 can also select YCbCr format in the BPP field of layer descriptor 4, which allows the same process to be applied to values stored in memory rather than brought in from the PDI. Interpolation is controlled for the layers using the LYR_INTPOL_EN register.

The RGB pixel value is computed using following equations:

$$\text{Red} = \frac{(Y-16)\cdot yred}{512} + \frac{(Cr-128)Crred}{512} + \frac{(Cb-128)Cbred}{512} \qquad \textit{Eqn. 11-7}$$

$$\text{Green} = \frac{(Y-16)\cdot Ygreen}{512} + \frac{(Cr-128)Crgreen}{512} + \frac{(Cb-128)Cbgreen}{512} \qquad \textit{Eqn. 11-8}$$

$$\text{Blue} = \frac{(Y-16)\cdot yblue}{512} + \frac{(Cr-128)Crblue}{512} + \frac{(Cb-128)Cbblue}{512} \qquad \textit{Eqn. 11-9}$$

Note that the first multiplication (i.e (y-16)*ycoeff) is unsigned, the two others are signed.

The register values after reset are as follows:

Yred = 10’h254 (596/512 = 1.16)

Crred = 11’h331 (817/512 = 1.6)

Cbred = 12’h000

Ygreen = 10’h254(596/512 = 1.16)

Crgreen = 11’h660(-416/512 = -0.812)

Cbgreen = 12’hf38(-200/512 = -0.39)

Yblue = 10’h254(596/512 = 1.16)

Crblue = 11’h000

Cbblue = 12’h409(1033/512 = 2.017)

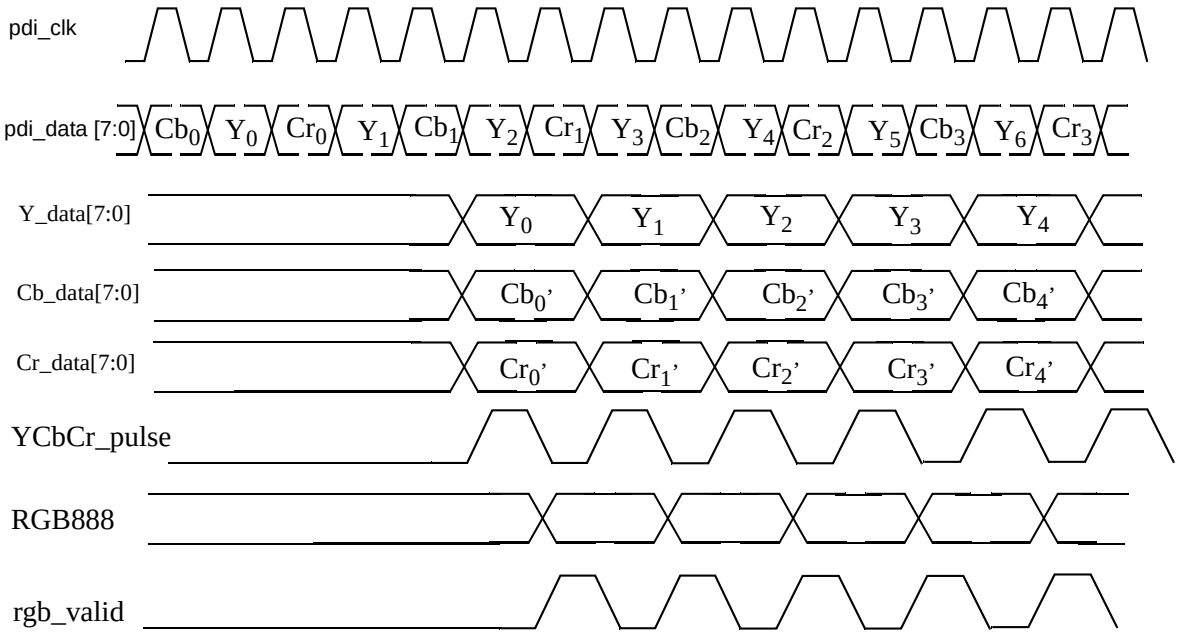



**Figure 11-98. YCbCr timing diagram**

$$Cb_n' = \begin{cases} Cb_{n/2} & \text{for even } n \\ (Cb_{(n-1)/2}' + Cb_{(n+1)/2})/2 & \text{for odd } n \end{cases} \qquad Cr_n' = \begin{cases} Cr_{n/2} & \text{for even } n \\ (Cr_{(n-1)/2}' + Cr_{(n+1)/2})/2 & \text{for odd } n \end{cases}$$

**Figure 11-99. YCbCr interpolation calculation**

### 11.8.1.7 Mode of operation depending on PDI[17:0]

PDI supports following modes (other than the Slave Mode):

- 8-bit monochrome (8-bit input data, each pixel info is coming in 1 clocks)
- 16-bit — RGB565 (16-bit input data, each pixel info is coming in 1 clocks)
- 18-bit — RGB666 (18-bit input data, each pixel info is coming in 1 clocks)
- 16-bit — RGB565 (8-bit input data, each pixel info is coming in 2 clocks)
- 16-bit — YCbCr422 (8-bit input data, info for 2 co-sited pixels coming in 4 clocks)

Data info extraction is given in Table 11-80.

**Table 11-80. Data extraction in all possible modes**

| PDI mode | Narrow mode | Pins | Data | Notes |
|---|---|---|---|---|
| 8-bit monochrome mode | 1'b0 | 8 bit | PDI[7:0] | — |
| RGB565 | 1'b0 | 16 bit | PDI[15:0] | — |
| RGB666 | 1'b0 | 18 bit | PDI[17:0] | — |
| RGB565 muxed | 1'b1 | 8 bit | PDI[7:0] | Data from two clocks are combined. |
| YCbCr422 | 1'b1 | 8 bit | PDI[7:0] | Data from four clocks are combined for 2 pixels. |

The 8-bit monochrome image is equivalent to 8-bit grayscale images. For converting 8-bit monochrome data to RGB data, each of the R/G/B components will have a value equal to the 8-bit monochrome value.

RGB extraction starts when PDI is enabled (from the next falling edge of validated vertical blanking period)

### 11.8.1.8 PDI-related Interrupts

PDI can be configured to trigger an interrupt when synchronization is achieved i.e. it receives the prespecified numbers of frames without error. PDI can also give an interrupt when synchronization is lost i.e. it receives any error in frame there after. This interrupt is raised when HSYNC/VSYNC is lost.

The PDI can also trigger an interrupt if there is either a one bit or a multiple bit error in the ECC value during the extraction of the preamble in internal synchronization mode.

Blanking sequence error interrupt can be triggered in case 80h 10h is not found in vertical and line blanking period during internal synchronization.

Activity detection interrupt for CLK detection.

HSYNC, VSYNC and DE detection interrupts can be set to generate upon transitions on the PDI_HSYNC, PDI_VSYNC and PDI_DE pins.

Activity lost interrupt for PDI_CLK — for the PDI_CLK lost interrupt, is triggered when the PDI_CLK is less than DCU3 module clock frequency divided by 32. (i.e. if DCU3 module clock freq. max = 64 MHz, then the PDI_CLK_LOST flag will be set if pdi clk freq. min < 2 MHz).

All interrupt flags are "write-one-to-clear" and all interrupts are maskable.

PDI must reset to show the latest status of the clock activity detect interrupt.

## 11.9 Switch between DCU mode and PDI mode (top-level description)

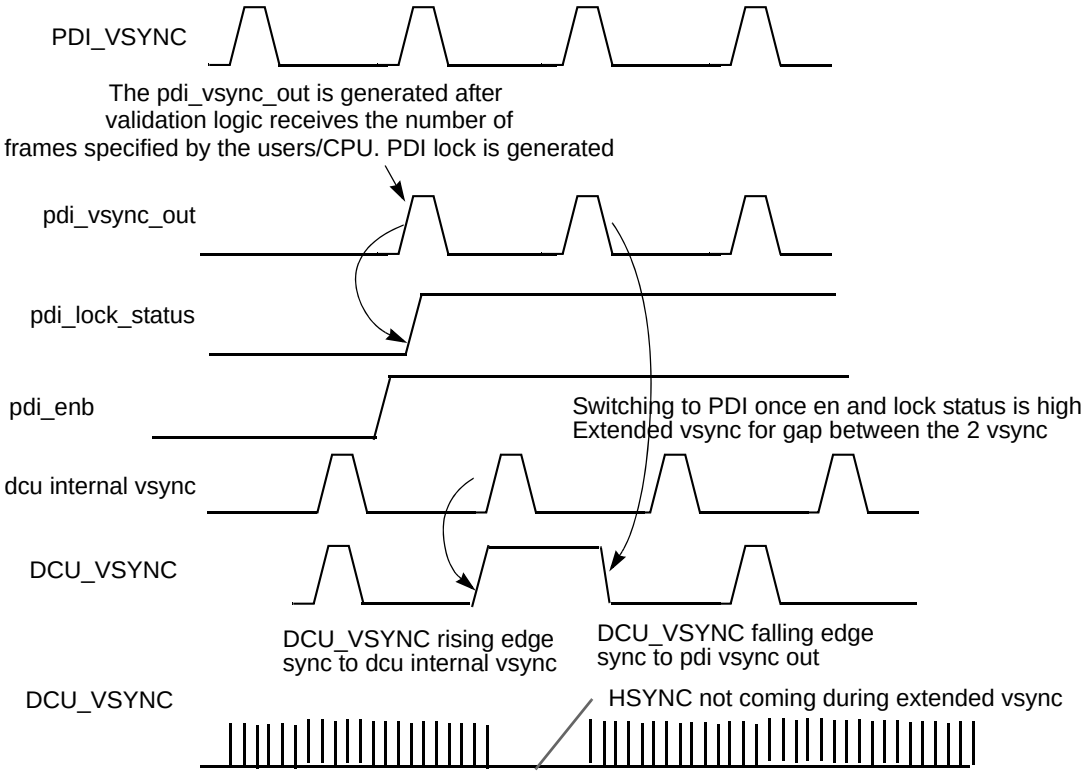



**Figure 11-100. Switch to PDI on receiving the interrupt**

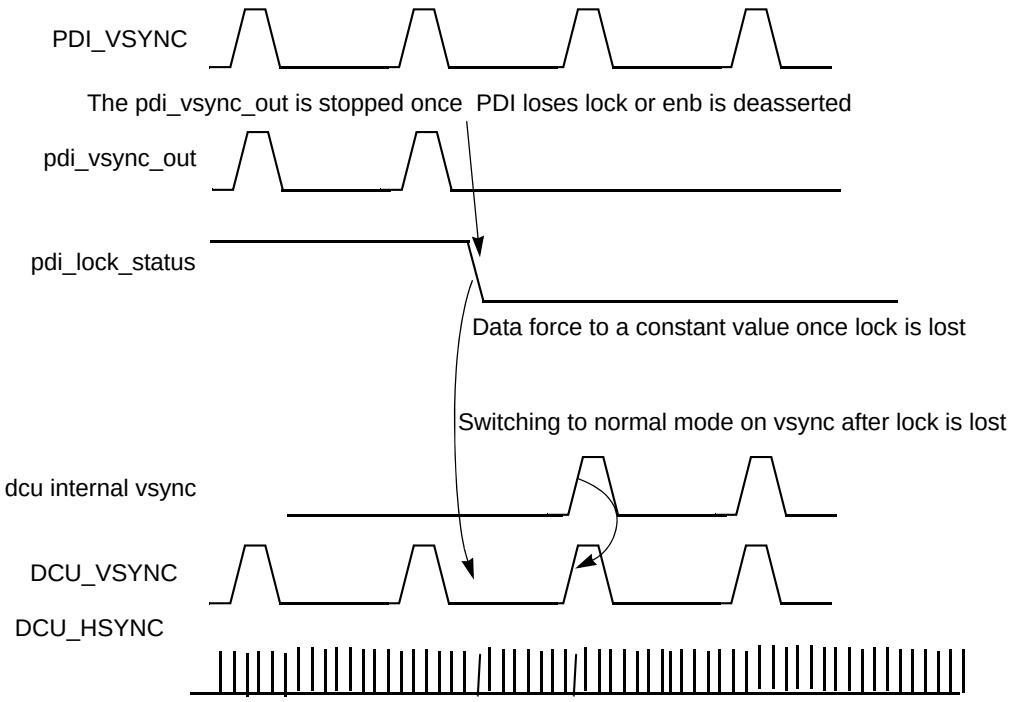



**Figure 11-101. Switch to Normal Mode from PDI mode**

### 11.9.1 Changes in the configuration

Any changes in the RGB format or the synchronization mode configuration requires the PDI system to be in disable mode i.e. PDI_EN should be 0 during this time. The resolution of the screen/layer (PDI or TFT) cannot change on the fly.

#### 11.9.1.1 PDI slave mode

The VSYNC generated by the DCU3 block is synchronized to the PDI input VSYNC; the PDI VSYNC resets the internal timing generation unit. HSYNC and VSYNC are generated internally corresponding to the external TFT screen display parameters programmed in the DCU3 registers.

In slave mode the PDI and DCU3 introduce a fixed latency to the VSYNC and HSYNC timing.

#### 11.9.1.2 PDI sync detection/validation

PDI declares a lock when it has correctly received the continuous number of frames programmed by the application.

PDI declares the sync lost:

- When it receives data enable during the vertical and horizontal blanking period (in case of data enable mode)
- When the incoming HSYNC and VSYNC timing does not match the programmed parameter values in the DCU3. This also means that if any of control signal info lost, then it also declares sync lost.

After PDI_PCLK is lost, the PDI requires 64 DCU3 module clock cycles to detect PDI_PCLK again.

PDI does not declare lost sync in case of blanking and ECC errors.

Writes to the input FIFO are stopped as soon as PDI sync is lost aborting transfer of the current frame data. Sync detection works continuously, independent of PDI enable. PDI fires an interrupt on:

- Sync lock is achieved
- Sync is lost
- Activity is detected on hsync
- Activity is detected on vsync
- PCLK activity detection (it is not generated from the state machine)
- DE Activity detection (it is not generated from the state machine)
- PCLK activity lost (it is not generated from the state machine)

On receiving a wrong sync pulse, the DCU3 stops the HSYNC and VSYNC activity detection and gives an interrupt when it finds sync pulse again. The state machine works for zero values of front and back porches but not for pulse width and screen parameters.

#### 11.9.1.3 Other assumptions

The reset to the PDI clock is synchronized to the peripheral clock. Reset synchronization is done with respect to the PDI clock internally.

The PDI clock should be available at least 10 clock after the last valid data. This corresponds to the delay of the PDI block.

## 11.10 DCU3 initialization

The following steps describe a typical approach to initializing the DCU3 for use in an application.

1. After reset configure the DCU3 peripheral to be active using the mode entry module and configure the DCU3 clock source in the MC_CGM.
2. If using a panel with an integrated TCON module, disable the TCON signals by setting the TCON_BYPASS bit in the TCON CTRL1 register. Due to the configuration of the TCON module, the DCU3 pixel clock signal will be output as soon as it is selected by the SIUL PCR. This is independent of the DCU3 operating mode.
3. Configure the output ports in the SIUL as required.
4. Configure the timing registers to match the TFT LCD panel in use (Section 11.4.2, “TFT LCD panel configuration””).
5. Set the background color as required.
6. Load the initial tile or palette colors into the CLUT/Tile memory
7. Configure the control descriptors for the layers and cursor that are to be used initially
8. Enable the DCU3 in the appropriate mode (DCU_MODE and RASTER_EN bit fields).

## 11.11 Glossary

**Table 11-81. Glossary**

| Term | Description |
|---|---|
| ARGB (also BGRA) | A data format where the pixel values are stored using four components: Alpha, Red, Green and Blue. DCU3 supports different variations of this format where different numbers of bits can be used to represent each of the components |
| Component | Part of a pixel that contains a single color (red, green or blue) |
| CLUT | Color Look-up table. The table that contains the palette used by an indexed-color graphic |
| Direct color | The full 24-bit value actually written to a pixel to create a color |
| Frame | The collection of all pixels on a panel |
| Gamut | The set of colors that a panel can display. In most cases a panel cannot display the full gamut of colors visible to the human eye. |
| Indexed color | An index into a table containing direct-colors. Usually smaller in size than the direct color; the DCU3 provides 1, 2, 4, and 8 bits per pixel options |
| Palette | The list of colors used by a graphic when an indexed colors format is used. The palette is stored in a color look up table and can be from one color up to the maximum of the size of the CLUT. |
| Panel | A TFT LCD containing an array of colored pixels. |
| Pixel | The basic graphical element on a TFT LCD panel. Can display a range of colors depending on the value of the red, green and blue values written to it. Normally arranged in a rectangular array. |
| RGB | A data format where the pixel values are stored using three components: Red, Green and Blue. DCU3 supports different variations of this format where different numbers of bits can be used to represent each of the components |
| Vertical blanking period | A time during the TFT LCD panel refresh cycle when no data is being written to the panel |

# Chapter 12
# Display Control Unit Lite (DCULite)

## 12.1 Introduction

The Display Control Unit Lite (DCULite) module is a system master that fetches graphics stored in internal or external memory and displays them on a TFT LCD panel. A wide range of panel sizes is supported and the timing of the interface signals is highly configurable. Graphics are read directly from memory and then blended in real time, which allows for dynamic content creation with minimal CPU intervention. Graphics may be encoded in a variety of formats to optimize memory usage. The DCULite also has the capability of displaying real-time video from an external video source.

### 12.1.1 Overview

Figure 12-1 shows the block diagram for DCULite.

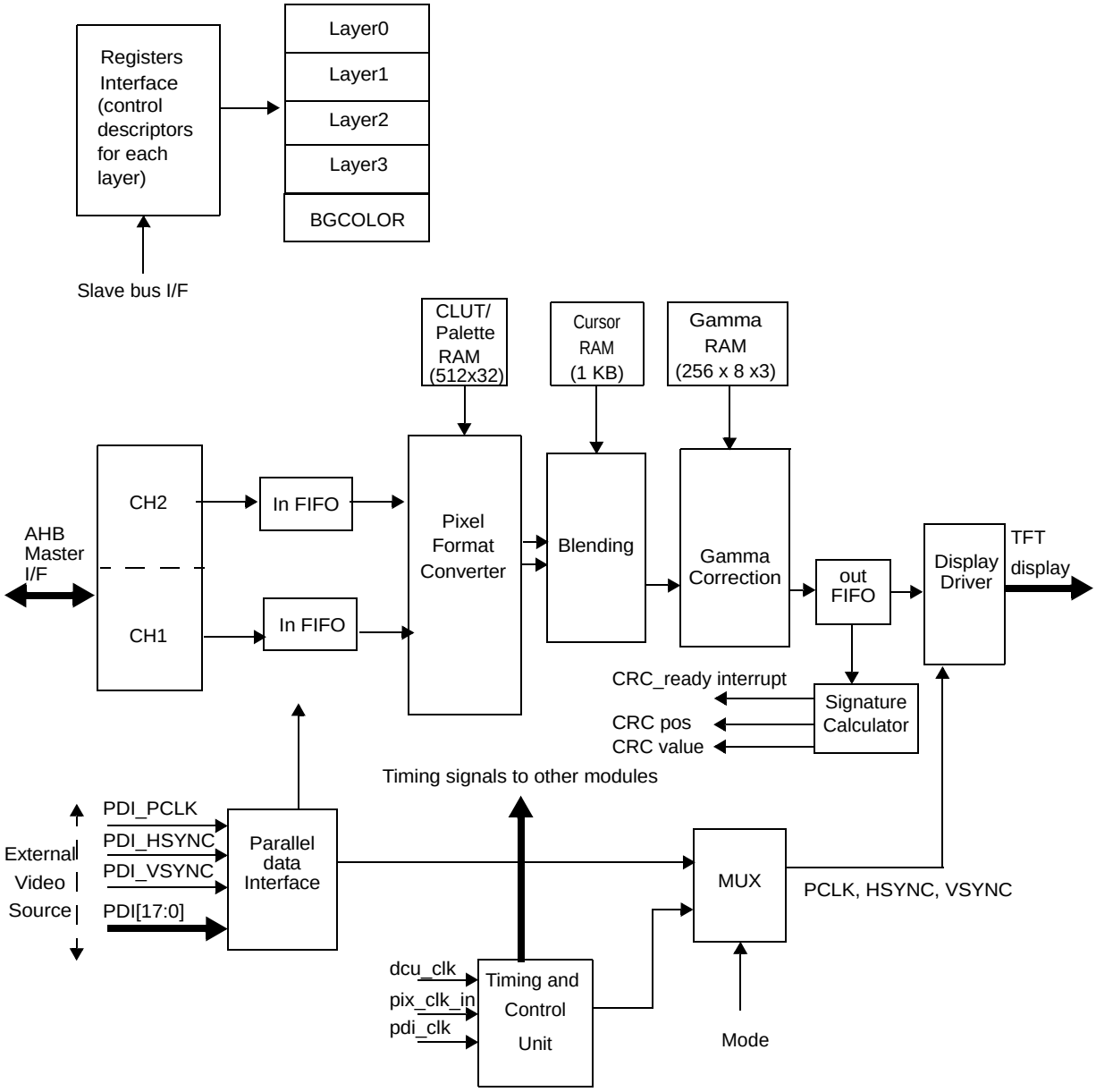



**Figure 12-1. DCULite block diagram**

The block diagram comprises two distinct sections. The lower section shows the functional blocks of the DCULite that fetch the graphic and video content and drive the TFT LCD panel. The upper section describes the user interface through which the user configures the graphical content of the TFT LCD panel.

The sections are analogous to the structure of communications modules, such as the FlexCAN, where one part of the module is configured to connect with the communications bus through bit-timing, parity, baud rate, etc., while a different part is used to store the data content and message identifiers.

The configuration of the lower section is dependent on the specific TFT LCD panel and optional real-time video hardware that are attached to the DCULite inputs and outputs. In most cases, this is configured once for the hardware in use before the DCULite is enabled. When active, this section automatically:

- Calculates the relevant graphical content for each pixel
- Fetches the source graphics from memory using its internal DMA channels (labelled CH1 and CH2)
- Converts the graphic value of each fetched pixel into full quality color format (if required)
- Calculates the required pixel value by blending the values of up to two separate graphics
- Performs a gamma correction on the pixel value (if required)
- Sends the pixel value to the TFT LCD display over its data bus
- Sets flags to indicate end of frame, buffer threshold, and other status changes

The upper section describes the characteristics of the graphics to be displayed on the panel and how they are blended together. The DCULite manages the graphical content of the panel through sets of registers called layers. There are 4 layers available in the DCULite and each contains the following information:

- Horizontal and vertical size of graphic
- Position of graphic on the panel
- Address of graphic in memory
- Color encoding format and color palettes (if required)
- Type and depth of blending
- Range of colors identified for chroma blending
- Tile size

The values in these registers may be changed at any time, and the panel content will be updated when the next full frame is ready to be displayed. The layers are set to a fixed priority, and this is used by the lower section to define which layers are blended, in which order, on the panel.

The upper section also contains configuration registers for a cursor graphic, the default background color, interrupt enables, test graphic, and simple register protection settings.

### 12.1.2 Features

The DCULite has these features:

- Full RGB888 output to TFT LCD panel
- 4 graphics layers, a default background color layer and a cursor layer with integrated blinking option
- Blending of each pixel using up to 2 source layers dependent on size of panel
- Programmable panel size up to a maximum of XGA (1024×768)
- Gamma correction with 8-bit resolution on each color component
- Safety mode for tagging pixels on highest priority layers
- Digital video input with and without sync extraction per ITU-R BT.656 supporting multiple video input formats including RGB666, RGB565, monochrome and YCbCr422

- Dedicated memory blocks to store a cursor and Color Look Up Tables (CLUTs)
- Temporal Dithering.

Each graphic layer has the following attributes:

- Can be placed with one pixel resolution in either axis
- Can also be placed in negative X & Y directions
- Supports multiple color-encoding formats including:
  - 1, 2, 4 and 8 bits per pixel indexed colors with alpha channel
  - APAL8 indexed colors with alpha channel
  - RGB565 and RGB888 direct colors
  - ARGB1555, ARGB4444, and BRGA 8888 direct colors with an alpha channel
  - YCbCr422 format
- Alpha blending with 8-bit resolution
- Chroma-key blending for anti-mask encoding
- Multiple alpha and chroma-key blending modes
- Transparency modes for anti-aliased text and graphics
- Luminance mode for highlighting content
- Tile mode for efficient creation of textured background content
- Support for Run length Encoding (RLE) compression
- Optimized mode for use with DDR memory

### 12.1.3 Modes of operation

The DCULite has four modes of operation:

- **DCU_OFF**: When in this mode, the DCULite is turned off. All the logic in the design is put in reset state to reduce power.
- **NORMAL_MODE**: The DCULite displays and blends the graphics specified by the layer descriptors.
- **PDI_MODE**: A mode which fetches video from an external video source and combines that with the graphics configured on the layers.
- **COLBAR_MODE**: Color bar generation for testing purposes. It is important for us to generate the color bars within the DCULite itself. This is useful for the user to verify if the DCULite is operational without interaction with the system memory.

## 12.2 External signal description

The DCULite has up to 22 input signals and up to 30 output signals. See Figure 12-2. The choice of signals used depends on the configuration of the DCULite. All active signals must be enabled by configuring the appropriate PCR registers in the SIUL module.

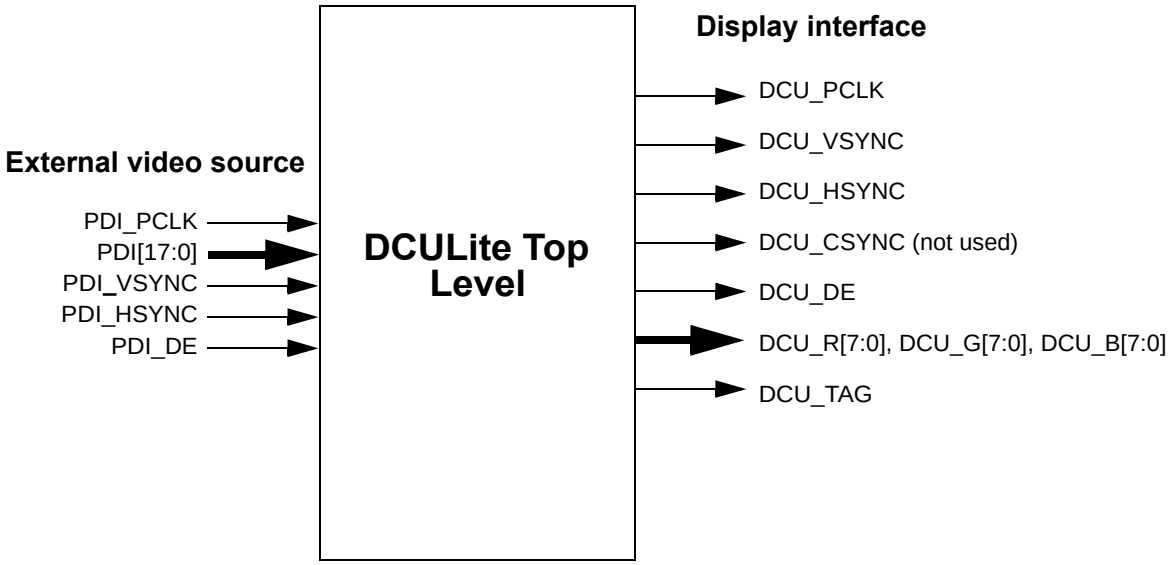



**Figure 12-2. External signals**

## 12.2.1 Detailed signal descriptions

**Table 12-1. Detailed signal descriptions**

| Signal | Direction | Description |
|---|---|---|
| Parallel Data Interface (Camera Interface) | | |
| PDI_PCLK | IN | Clock for the parallel data from the input video data |
| PDI_VSYNC | IN | Vertical sync to indicate the start of new frame for the display |
| PDI_HSYNC | IN | Horizontal sync to indicate the start of new line for the display |
| PDI_DE | IN | Data Enable for the camera data input |
| PDI[17:0] | IN | 18-bit parallel input data for the display |
| Display Interface | | |
| DCU_PCLK | OUT | Pixel clock used to drive the display panel |
| DCU_VSYNC | OUT | Vertical sync signal, indicating the beginning of a new frame |
| DCU_HSYNC | OUT | Horizontal sync signal, indicating the beginning of a new line |
| DCU_TAG | OUT | When high, this signal indicates that the pixel is tagged and an application can calculate CRC externally on this pixel. |
| DCU_DE | OUT | Data Enable. Qualifies the data output |
| DCU_R[7:0], DCU_G[7:0], DCU_B[7:0] | OUT | Red, green and blue data output. |

## 12.3 Memory map and register definition

### 12.3.1 Memory map

Table 12-2 shows the memory map of the DCULite.

**Table 12-2. DCULite memory map**

| Parameter | Address range |
|---|---|
| Register address space | 0x0000 – 0x03FF |
| Cursor address space | 0x0400 – 0x07FF |
| Gamma_R address space | 0x0800 – 0x0BFF |
| Gamma_G address space | 0x0C00 – 0x0FFF |
| Gamma_B address space | 0x1000 – 0x13FF |
| Empty space | 0x1400 – 0x1FFF |
| CLUT address space (2 KB repeated) | 0x2000 – 0x3FFF |

### 12.3.2 Register map

Table 12-3 provides the register map of the DCULite.

Only 32-bit writes and 32-bit aligned access are supported. Byte half-word access is not supported.

**Table 12-3. DCULite register map**

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x000 | CtrlDescL0_1 register | R/W | 0x00000000 | on page 481 |
| 0x004 | CtrlDescL0_2 register | R/W | 0x00000000 | on page 482 |
| 0x008 | CtrlDescL0_3 register | R/W | 0x00000000 | on page 483 |
| 0x00C | CtrlDescL0_4 register | R/W | 0x00000000 | on page 484 |
| 0x010 | CtrlDescL0_5 register | R/W/ | 0x00000000 | on page 485 |
| 0x014 | CtrlDescL0_6 register | R/W/ | 0x00000000 | on page 486 |
| 0x018 | CtrlDescL0_7 register | R/W/ | 0x00000000 | on page 487 |
| 0x01C | CtrlDescL1_1 register | R/W | 0x00000000 | on page 481 |
| 0x020 | CtrlDescL1_2 register | R/W | 0x00000000 | on page 482 |
| 0x024 | CtrlDescL1_3 register | R/W | 0x00000000 | on page 483 |
| 0x028 | CtrlDescL1_4 register | R/W | 0x00000000 | on page 484 |
| 0x02C | CtrlDescL1_5 register | R/W | 0x00000000 | on page 485 |
| 0x030 | CtrlDescL1_6 register | R/W | 0x00000000 | on page 486 |
| 0x034 | CtrlDescL1_7 register | R/W | 0x00000000 | on page 487 |
| 0x038 | CtrlDescL2_1 register | R/W | 0x00000000 | on page 481 |

**Table 12-3. DCULite register map (continued)**

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x03C | CtrlDescL2_2 register | R/W | 0x00000000 | on page 482 |
| 0x040 | CtrlDescL2_3 register | R/W | 0x00000000 | on page 483 |
| 0x044 | CtrlDescL2_4 register | R/W | 0x00000000 | on page 484 |
| 0x048 | CtrlDescL2_5 register | R/W | 0x00000000 | on page 485 |
| 0x04C | CtrlDescL2_6 register | R/W | 0x00000000 | on page 486 |
| 0x050 | CtrlDescL2_7 register | R/W | 0x00000000 | on page 487 |
| 0x054 | CtrlDescL3_1 register | R/W | 0x00000000 | on page 481 |
| 0x058 | CtrlDescL3_2 register | R/W | 0x00000000 | on page 482 |
| 0x05C | CtrlDescL3_3 register | R/W | 0x00000000 | on page 483 |
| 0x060 | CtrlDescL3_4 register | R/W | 0x00000000 | on page 484 |
| 0x064 | CtrlDescL3_5 register | R/W | 0x00000000 | on page 485 |
| 0x068 | CtrlDescL3_6 register | R/W | 0x00000000 | on page 486 |
| 0x06C | CtrlDescL3_7 register | R/W | 0x00000000 | on page 487 |
| 0x070–0x1BC | Reserved | | | |
| 0x1C0 | CtrlDescCursor_1 register | R/W | 0x00000000 | on page 488 |
| 0x1C4 | CtrlDescCursor_2 register | R/W | 0x00000000 | on page 488 |
| 0x1C8 | CtrlDescCursor_3 register | R/W | 0x00000000 | on page 489 |
| 0x1CC | CtrlDescCursor_4 register | R/W | 0x00000000 | on page 490 |
| 0x1D0 | DCU_MODE register | R/W | 0x00008000 | on page 490 |
| 0x1D4 | BGND register | R/W | 0x00000000 | on page 492 |
| 0x1D8 | DISP_SIZE register | R/W | 0x00000000 | on page 493 |
| 0x1DC | HSYN_PARA register | R/W | 0x00C01803 | on page 494 |
| 0x1E0 | VSYN_PARA register | R/W | 0x00C01803 | on page 494 |
| 0x1E4 | SYNPOL register | R/W | 0x00000000 | on page 495 |
| 0x1E8 | THRESHOLD register | R/W | 0x0000780A | on page 496 |
| 0x1EC | INT_STATUS register | R | 0x00000000 | on page 497 |
| 0x1F0 | INT_MASK register | R/W | 0x00004FFF | on page 498 |
| 0x1F4 | COLBAR_1 register | R/W | 0xFF000000 | on page 500 |
| 0x1F8 | COLBAR_2 register | R/W | 0xFF0000FF | on page 500 |
| 0x1FC | COLBAR_3 register | R/W | 0xFF00FFFF | on page 501 |
| 0x200 | COLBAR_4 register | R/W | 0xFF00FF00 | on page 501 |
| 0x204 | COLBAR_5 register | R/W | 0xFFFFFF00 | on page 502 |
| 0x208 | COLBAR_6 register | R/W | 0xFFFF0000 | on page 502 |

**Table 12-3. DCULite register map (continued)**

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x20C | COLBAR_7 register | R/W | 0xFFFF00FF | on page 503 |
| 0x210 | COLBAR_8 register | R/W | 0xFFFFFFFF | on page 503 |
| 0x214 | DIV_RATIO register | R/W | 0x0000001F | on page 503 |
| 0x218 | SIGN_CALC_1 register | R/W | 0x00000000 | on page 504 |
| 0x21C | SIGN_CALC_2 register | R/W | 0x00000000 | on page 505 |
| 0x220 | CRC_VAL register | R/W | 0x00000000 | on page 505 |
| 0x224 | PDI_STATUS register | R | 0x00000000 | on page 506 |
| 0x228 | MASK_PDI_STATUS register | R/W | 0x000003FF | on page 507 |
| 0x22C | PARR_ERR_STATUS register | R | 0x00000000 | on page 508 |
| 0x230 | MASK_PARR_ERR_STATUS register | R/W | 0x000F000F | on page 510 |
| 0x234 | THRESHOLD_INP_BUF_1 Register | R/W | 0x7F007F00 | on page 511 |
| 0x238 | Reserved | | | |
| 0x23C | LUMA_COMP register | R/W | 0x9512A254 | on page 512 |
| 0x240 | RED register | R/W | 0x03310000 | on page 512 |
| 0x244 | GREEN register | R/W | 0x06600F38 | on page 513 |
| 0x248 | BLUE register | R/W | 0x00000409 | on page 513 |
| 0x24C | CRC_POS register | R/W | 0x00000000 | on page 514 |
| 0x250 | FG0_fcolor register | R/W | 0x00000000 | on page 515 |
| 0x254 | FG0_bcolor register | R/W | 0x00000000 | on page 515 |
| 0x258 | FG1_fcolor register | R/W | 0x00000000 | on page 515 |
| 0x25C | FG1_bcolor register | R/W | 0x00000000 | on page 515 |
| 0x260 | FG2_fcolor register | R/W | 0x00000000 | on page 515 |
| 0x264 | FG2_bcolor register | R/W | 0x00000000 | on page 515 |
| 0x268 | FG3_fcolor register | R/W | 0x00000000 | on page 515 |
| 0x26c | FG3_bcolor register | R/W | 0x00000000 | on page 515 |
| 0x270–0x2CC | Reserved | | | |
| 0x2D0 | LYR_INTPOL_EN | R/W | 0x00000000 | on page 516 |
| 0x2D4 | LYR_LUMA_COMP register | R/W | 0x9512A254 | on page 516 |
| 0x2D8 | LYR_CHROMA_RED register | R/W | 0x03310000 | on page 517 |
| 0x2Dc | LYR_CHROMA_GREEN register | R/W | 0x06600F38 | on page 518 |
| 0x2E0 | LYR_CHROMA_BLUE register | R/W | 0x00000409 | on page 518 |
| 0x2E4 | COMP_IMSIZE register | R/W | 0x00000000 | on page 519 |
| 0x2E8–0x2FC | Reserved | | | |

**Table 12-3. DCULite register map (continued)**

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x300 | Global protection register | R/W | 0x00000000 | on page 519 |
| 0x304 | Soft lock bit register L0 | R/W | 0x00000000 | on page 520 |
| 0x308 | Soft lock bit register L1 | R/W | 0x00000000 | on page 521 |
| 0x30C | Soft lock bit register DISP_SIZE | R/W | 0x00000000 | on page 523 |
| 0x310 | Soft lock bit register VSYNC/HSYNC PARA | R/W | 0x00000000 | on page 524 |
| 0x314 | Soft lock bit register POL | R/W | 0x00000000 | on page 525 |
| 0x318 | Soft lock bit register L0_TRANSP | R/W | 0x00000000 | on page 525 |
| 0x31C | Soft lock bit register L1_TRANSP | R/W | 0x00000000 | on page 526 |

## 12.3.3 Register summary

Figure 12-3 provides a key for register figures and tables and the register summary.

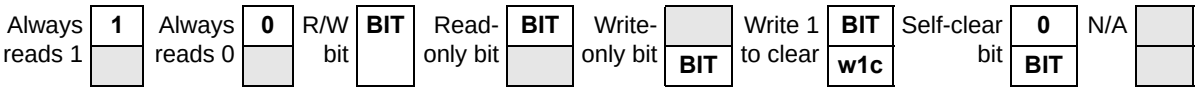



**Figure 12-3. Key to register fields**

The conventions in Table 12-4 serve as a key for the register summary and individual register diagrams.

**Table 12-4. Register conventions**

| Convention | Description |
|---|---|
| | Depending on its placement in the read or write row, indicates that the bit is not readable or not writable |
| FIELDNAME | Identifies the field. Its presence in the read or write row indicates that it can be read or written. |
| **Register field types** | |
| R | Read only. Writing this bit has no effect |
| W | Write only |
| R/W | Standard read/write bit. Only software can change the bit's value (other than a hardware reset). |
| rwm | A read/write bit that can be modified by hardware in some fashion other than by a reset |
| w1c | Write one to clear. A status bit that can be read, and is cleared by writing a one. |
| Self-clearing bit | Writing a one has some effect on the module, but it always reads as zero. (Previously designated slfclr) |
| S | Set: Pattern on the data bus is ORed with and written into the register. |
| C | Clear: Pattern on the data bus is a mask. If a bit on the mask is set, then the corresponding register bit is cleared. |
| **Reset values** | |
| 0 | Resets to zero |

**Table 12-4. Register conventions (continued)**

| Convention Description | |
|---|---|
| 1 | Resets to one |
| — | Undefined at reset |
| u | Unaffected by reset |
| [*signal_name*] | Reset value is determined by polarity of indicated signal. |

**Table 12-5. Register descriptions**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| CtrlDescL0_1 0x000 | R | 0 | 0 | 0 | 0 | 0 | HEIGHT | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | WIDTH | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescL0_2 0x004 | R | 0 | 0 | 0 | 0 | POSY | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | POSX | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescL0_3 0x008 | R | ADDR | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescL0_4 0x00C | R | EN | TILE_EN | 0 | SAFETY_EN | TRANS | | | | | | | | BPP | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | RLE_EN | 0 | 0 | LUOFFS | | | | | | | | | 0 | BB | AB | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescL0_5 0x010 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CKMAX_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | CKMAX_G | | | | | | | | CKMAX_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescL0_6 0x014 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CKMIN_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | CKMIN_G | | | | | | | | CKMIN_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 12-5. Register descriptions (continued)**

| Name / Offset | | 0 / 16 | 1 / 17 | 2 / 18 | 3 / 19 | 4 / 20 | 5 / 21 | 6 / 22 | 7 / 23 | 8 / 24 | 9 / 25 | 10 / 26 | 11 / 27 | 12 / 28 | 13 / 29 | 14 / 30 | 15 / 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CtrlDescL0_7 0x018 | R | 0 | 0 | 0 | 0 | 0 | TILE_VERT_SIZE | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | TILE_HOR_SIZE | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescCursor_1 0x1C0 | R | 0 | 0 | 0 | 0 | 0 | HEIGHT | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | WIDTH | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescCursor_2 0x1C4 | R | 0 | 0 | 0 | 0 | 0 | POSY | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | POSX | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescCursor_3 0x1C8 | R | CUR_EN | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DEFAULT_CURSOR_COLOR[0:7] | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | DEFAULT_CURSOR_COLOR[8:23] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CtrlDescCursor_4 0x1CC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | HWC_BLINK_OFF | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | EN_BLINK | HWC_BLINK_ON | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| DCU_MODE 0x1D0 | R | DCU_SW_RESET | DITHER_EN | ADDB | | ADDG | | ADDR | | DDR_MODE | 0 | 0 | 0 | PDI_SYNC_LOCK | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | PDI_INTERPOL_EN | RASTER_EN | PDI_EN | PDI_BYTE_REV | PDI_DE_MODE | PDI_NARROW_MODE | PDI_MODE | | PDI_SLAVE_MODE | TAG_EN | SIG_EN | PDI_SYNC | 0 | EN_GAMMA | DCU_MODE | |
| | W | | | | | | | | | | | | | | | | |

**Table 12-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| BGND 0x1D4 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BGND_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | BGND_G | | | | | | | | BGND_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| DISP_SIZE 0x1D8 | R | 0 | 0 | 0 | 0 | 0 | DELTA_Y | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DELTA_X | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| HSYN_PARA 0x1DC | R | 0 | BP_H | | | | | | | | | 0 | 0 | PW_H[0:3] | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | PW_H[4:8] | | | | | 0 | 0 | FP_H | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| VSYN_PARA 0x1E0 | R | 0 | BP_V | | | | | | | | | 0 | 0 | PW_V[0:3] | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | PW_V[4:8] | | | | | 0 | 0 | FP_V | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| SYN_POL 0x1E4 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | |
| | W | | | | | | INV_PDI_DE | INV_PDI_HS | INV_PDI_VS | INV_PDI_CLK | INV_PXCK | NEG | BP_VS | BP_HS | INV_CS | INV_VS | INV_HS |
| THRESHOLD 0x1E8 | R | 0 | 0 | 0 | 0 | 0 | LS_BF_VS | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | OUT_BUF_HIGH | | | | | | | | OUT_BUF_LOW | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 12-5. Register descriptions (continued)**

| Name / Offset | | 0 / 16 | 1 / 17 | 2 / 18 | 3 / 19 | 4 / 20 | 5 / 21 | 6 / 22 | 7 / 23 | 8 / 24 | 9 / 25 | 10 / 26 | 11 / 27 | 12 / 28 | 13 / 29 | 14 / 30 | 15 / 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INT_STATUS 0x1EC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | DMA_TRANS_FINISH | 0 | 0 | IPM_ERROR | PROG_END | P2_FIFO_HI_FLAG | P2_FIFO_LO_FLAG | P1_FIFO_HI_FLAG | P1_FIFO_LO_FLAG | CRC_OVERFLOW | CRC_READY | VS_BLANK | LS_BF_VS | UNDRUN | VSYNC |
| | W | | w1c | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| INT_MASK 0x1F0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | M_DMA_TRANS_FINISH | 0 | 0 | M_IPM_ERROR | M_PROG_END | M_P2_FIFO_HI_FLAG | M_P2_FIFO_LO_FLAG | M_P1_FIFO_HI_FLAG | M_P1_FIFO_LO_FLAG | M_CRC_OVERFLOW | M_CRC_READY | M_VS_BLANK | M_LS_BF_VS | M_UNDRUN | M_VSYNC |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_1 0x1F4 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_1_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_1_G | | | | | | | | COLBAR_1_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_2 0x1F8 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_2_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_2_G | | | | | | | | COLBAR_2_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_3 0x1FC | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_3_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_3_G | | | | | | | | COLBAR_3_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_4 0x200 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_4_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_4_G | | | | | | | | COLBAR_4_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 12-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| COLBAR_5 0x204 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_5_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_5_G | | | | | | | | COLBAR_5_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_6 0x208 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_6_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_6_G | | | | | | | | COLBAR_6_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_7 0x20C | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_7_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_7_G | | | | | | | | COLBAR_7_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| COLBAR_8 0x210 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_8_R | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COLBAR_8_G | | | | | | | | COLBAR_8_B | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| DIV_RATIO 0x214 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DIV_RATIO | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| SIGN_CALC_1 0x218 | R | 0 | 0 | 0 | 0 | 0 | SIG_VER_SIZE | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | SIG_HOR_SIZE | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| SIGN_CALC_2 0x21C | R | 0 | 0 | 0 | 0 | 0 | SIG_VER_POS | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | SIG_HOR_POS | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CRC_VAL 0x220 | R | CRC_VAL | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 12-5. Register descriptions (continued)**

| Name<br>Offset | | 0<br>16 | 1<br>17 | 2<br>18 | 3<br>19 | 4<br>20 | 5<br>21 | 6<br>22 | 7<br>23 | 8<br>24 | 9<br>25 | 10<br>26 | 11<br>27 | 12<br>28 | 13<br>29 | 14<br>30 | 15<br>31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PDI_STATUS<br>0x224 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | PDI_BLANKING_ERR | PDI_ECC_ERR2 | PDI_ECC_ERR1 | PDI_LOCK_LOST | PDI_LOCK_DET | PDI_VSYNC_DET | PDI_HSYNC_DET | PDI_DE_DET | PDI_CLK_LOST | PDI_CLK_DET |
| | W | | | | | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| MASK_PDI_<br>STATUS<br>0x228 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | M_PDI_BLANKING_ERR | M_PDI_ECC_ERR2 | M_PDI_ECC_ERR1 | M_PDI_LOCK_LOST | M_PDI_LOCK_DET | M_PDI_VSYNC_DET | M_PDI_HSYNC_DET | M_PDI_DE_DET | M_PDI_CLK_LOST | M_PDI_CLK_DET |
| | W | | | | | | | | | | | | | | | | |
| PARR_ERR_<br>STATUS<br>0x22C | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RLE_ERR | HWC_ERR | SIG_ERR | DISP_ERR |
| | W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | L3_PARR_ERR | L2_PARR_ERR | L1_PARR_ERR | L0_PARR_ERR |
| | W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |

**Table 12-5. Register descriptions (continued)**

| Name / Offset | | 0 / 16 | 1 / 17 | 2 / 18 | 3 / 19 | 4 / 20 | 5 / 21 | 6 / 22 | 7 / 23 | 8 / 24 | 9 / 25 | 10 / 26 | 11 / 27 | 12 / 28 | 13 / 29 | 14 / 30 | 15 / 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MASK_PARR_ERR_STATUS 0x230 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | M_RLE_ERR | M_HWC_ERR | M_SIG_ERR | M_DISP_ERR |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | M_L3_PARR_ERR | M_L2_PARR_ERR | M_L1_PARR_ERR | M_L0_PARR_ERR |
| | W | | | | | | | | | | | | | | | | |
| THRESHOLD_INP_BUF 0x234 | R | 0 | INP_BUF_P2_HI | | | | | | | 0 | INP_BUF_P2_LO | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | INP_BUF_P1_HI | | | | | | | 0 | INP_BUF_P1_LO | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| LUMA_COMP 0x23C | R | Y_RED | | | | | | | | | | | Y_GREEN[0:4] | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | Y_GREEN[5:9] | | | | | | Y_BLUE | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CHROMA_RED 0x240 | R | 0 | 0 | 0 | 0 | 0 | CR_RED | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_RED | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CHROMA_GREEN 0x244 | R | 0 | 0 | 0 | 0 | 0 | CR_GREEN | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_GREEN | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CHROMA_BLUE 0x248 | R | 0 | 0 | 0 | 0 | 0 | CR_BLUE | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_BLUE | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| CRC_POS 0x24c | R | CRC_POS | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 12-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| FGx_fcolor 0x250 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | FGX_FCOLOR[0:7] | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | FGX_FCOLOR[8:23] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| FGx_bcolor 0x254 | R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | FGX_BCOLOR[0:7] | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | FGX_BCOLOR[8:23] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| LYR_INTPOL_EN 0x2D0 | R | LYR_INTPOL_EN | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | | |
| LYR_LUMA_COMP 0x2D4 | R | Y_RED | | | | | | | | | | | Y_GREEN[0:4] | | | | | |
| | W | | | | | | | | | | | | | | | | | |
| | R | Y_GREEN[5:9] | | | | | | | | Y_BLUE | | | | | | | | |
| | W | | | | | | | | | | | | | | | | | |
| LYR_CHROMA_RED 0x2D8 | R | 0 | 0 | 0 | 0 | 0 | CR_RED | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_RED | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | | |
| LYR_CHROMA_GREEN 0x2Dc | R | 0 | 0 | 0 | 0 | 0 | CR_GREEN | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_GREEN | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | | |
| LYR_CHROMA_BLUE 0x2E0 | R | 0 | 0 | 0 | 0 | 0 | CR_BLUE | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | CB_BLUE | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | | |

**Table 12-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| COMP_IMSIZE 0x2E4 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | COMP_IMSIZE[21:16] | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COMP_IMSIZE[15:0] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| Global_protection 0x300 | R | HLB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock_Bit L0 0x304 | R | 0 | 0 | 0 | 0 | | | | | 0 | 0 | 0 | 0 | | | | 0 |
| | W | WEN_LO_1 | WEN_LO_2 | WEN_LO_3 | WEN_LO_4 | SLB_L0_1 | SLB_L0_2 | SLB_L0_3 | SLB_L0_4 | WEN_LO_5 | WEN_LO_6 | WEN_LO_7 | | SLB_L0_5 | SLB_L0_6 | SLB_L0_7 | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock_Bit L1 0x308 | R | 0 | 0 | 0 | 0 | | | | | 0 | 0 | 0 | 0 | | | | 0 |
| | W | WEN_L1_1 | WEN_L1_2 | WEN_L1_3 | WEN_L1_4 | SLB_L1_1 | SLB_L1_2 | SLB_L1_3 | SLB_L1_4 | WEN_L1_5 | WEN_L1_6 | WEN_L1_7 | | SLB_L1_5 | SLB_L1_6 | SLB_L1_7 | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock_DISP_SIZE 0x30c | R | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | WEN_DISP | | | | SLB_DISP | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock_HSYNC/VSYNC PARA 0x310 | R | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | WEN_HSYNC | WEN_VSYNC | | | SLB_HSYNC | SLB_VSYNC | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |

**Table 12-5. Register descriptions (continued)**

| Name / Offset | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Soft_Lock_POL 0x314 | R | 0 | 0 | 0 | 0 | SLB_POL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | WEN_POL | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock L0_TRANSP 0x318 | R | 0 | 0 | 0 | 0 | SLB_FCOLOR | SLB_BCOLOE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | WEN_FCOLOR | WEN_BCOLOR | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| Soft_Lock L1_TRANSP 0x31C | R | 0 | 0 | 0 | 0 | SLB_FCOLOR | SLB_BCOLOE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | WEN_FCOLOR | WEN_BCOLOR | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |

## 12.3.4 Register descriptions

This section describes the DCULite registers.

### 12.3.4.1 Control Descriptor L0_1 Register (CtrlDescL0_1)

Figure 12-4 represents the control descriptor L0_1 register. This register sets the height and width of the layer associated with the register.

Offsets:
0x000 (CtrlDescL0_1)
0x01C (CtrlDescL1_1)
0x038 (CtrlDescL2_1)
0x054 (CtrlDescL3_1)

Access: User read/write

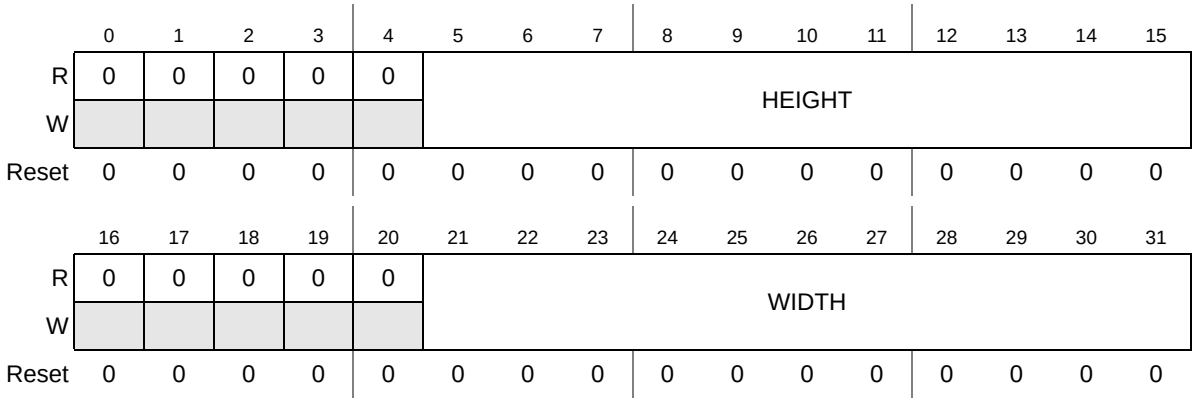



**Figure 12-4. CtrlDescL0_1 Register**

**Table 12-6. CtrlDescL0_1 field descriptions**

| Field | Description |
|---|---|
| HEIGHT | Height of the layer in pixels |
| WIDTH | Width of the layer (in pixels). The layer width must be in multiples of the number of pixels that can be stored in 32 bits except for the special case of 1 bit per pixel, and therefore differs depending on color encoding. For example, if 2 bits per pixel format is used, then the layer width must be configured in multiples of 16. |

### 12.3.4.2 Control Descriptor L0_2 Register

Figure 12-5 represents the control descriptor L0_2 register.This register sets the origin (top/left) of the layer associated with the register

Offsets:
0x004 (CtrlDescL0_2)
0x020 (CtrlDescL1_2)
0x03C (CtrlDescL2_2)
0x058 (CtrlDescL3_2)

Access: User read/write

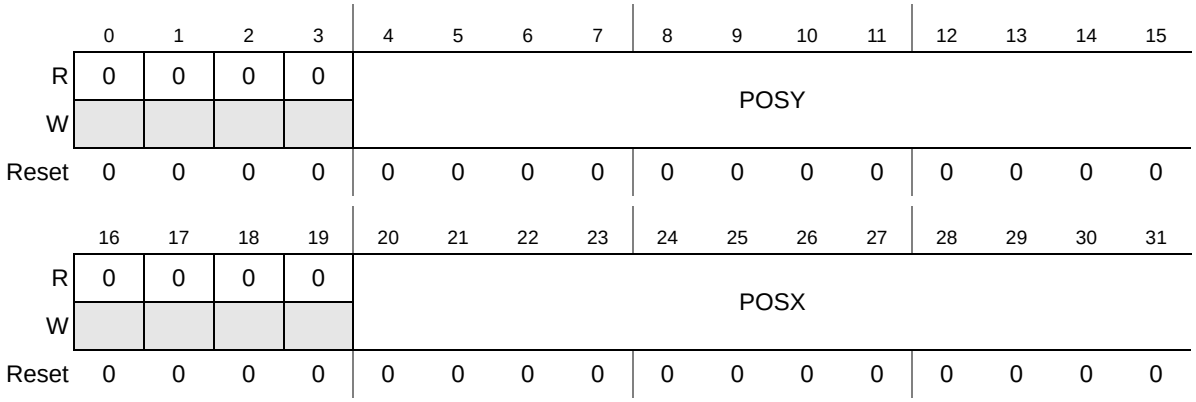



**Figure 12-5. CtrlDescL0_2 Register**

**Table 12-7. CtrlDescL0_2 field descriptions**

| Field | Description |
|---|---|
| POSY | Two's complement signed value setting the vertical position of top row of the layer, where 0 is the top row of the panel. Positive values are below and negative values are above the top row of the panel. |
| POSX | Two's complement signed value setting the horizontal position of left hand column of the layer, where 0 is the left hand column of the panel. Positive values are to the right and negative values are to the left the left hand column of the panel. |

### 12.3.4.3 Control Descriptor L0_3 Register

Figure 12-8 represents the control descriptor L0_3 register.This register sets the beginning address of layer data.

Offsets:
0x008 (CtrlDescL0_3)
0x024 (CtrlDescL1_3)
0x040 (CtrlDescL2_3)
0x05C (CtrlDescL3_3)

Access: User read/write

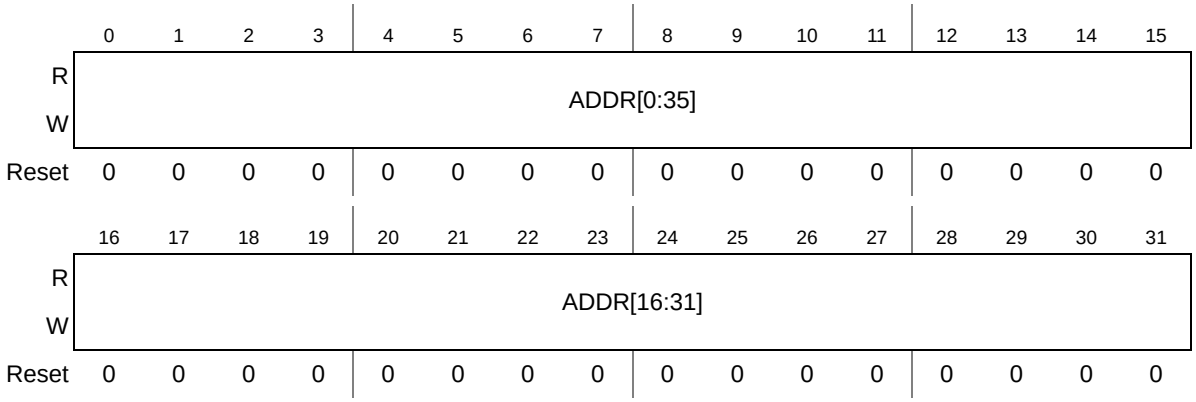



**Figure 12-6. CtrlDescL0_3 Register**

**Table 12-8. CtrlDescL0_3 field descriptions**

| Field | Description |
|---|---|
| ADDR | Address of layer data in the memory. The address programmed should be 64-bit aligned. |

### 12.3.4.4 Control Descriptor L0_4 Register

Figure 12.3.4.5 represents the control descriptor L0_4 register.his register controls various graphics options and whether the layer is enabled.

Offsets:
0x00C (CtrlDescL0_4)
0x028 (CtrlDescL1_4)
0x044 (CtrlDescL2_4)
0x060 (CtrlDescL3_4)

Access: User read/write

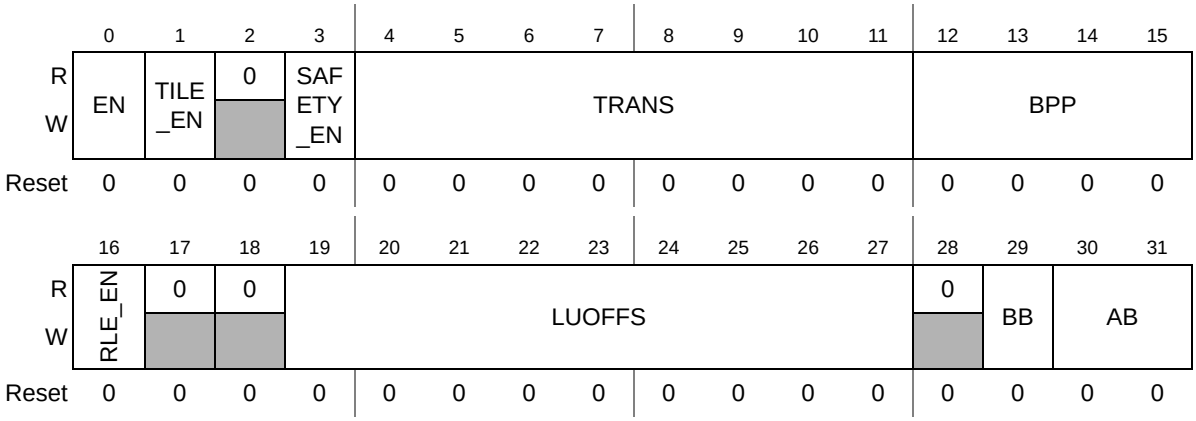



**Figure 12-7. CtrlDescL0_4 Register**

**Table 12-9. CtrlDescL0_4 field descriptions**

| Field | Description |
|---|---|
| EN | Enable the layer<br>1'b1: ON<br>1'b0: OFF |
| TILE_EN | Enable the Tile Mode<br>1'b1: ON<br>1'b0: OFF |
| SAFETY_EN | Safety Mode Enable Bit. Valid only for layer 0 and layer 1. For registers of all other layers, this should be set to 0.<br>1'b1: Safety Mode is enabled for this layer<br>1'b0: Safety Mode is disabled |
| TRANS | Transparency Level. Specifies the alpha value for the layer. This value may be used by the blending engine to blend pixels on this layer. Value can vary between 0–255 where 0 is completely transparent and 255 is completely opaque. |
| BPP | Bits Per Pixel<br>4'b0000 = 1 bpp<br>4'b0001 = 2 bpp<br>4'b0010 = 4 bpp<br>4'b0011 = 8 bpp<br>4'b0100 = 16 bpp (RGB565)<br>4'b0101 = 24 bpp<br>4'b0110 = 32 bpp (BGRA8888)<br>4'b0111 = Transparency mode 4 bpp<br>4'b1000 = Transparency mode 8bpp<br>4'b1001 = Luminance offset mode 4 bpp<br>4'b1010 = Luminance offset mode 8 bpp<br>4'b1011 = 16 bpp (ARGB1555)<br>4'b1100 = 16 bpp (ARGB4444)<br>4'b1101 = 16 bpp (APAL8)<br>4'b1110 = YCbCr422 (the blend engine allows only a single YCbCr layer in any blend operation)<br>4'b1111 = Reserved |
| RLE_EN | Enable RLE mode for layer.<br>1'b1:Enabled<br>1'b0:Disabled |
| LUOFFS | Look Up Table offset. Value gives the offset to the start address of the CLUT (maximum value = 0x1FF for 512x32-bit entries). |
| BB | Chroma Keying<br>1'b1: ON<br>1'b0: OFF |
| AB | Alpha Blending<br>2'b00: No alpha Blending<br>2'b01: Blend only the pixels selected by chroma keying in case BB=1'b1<br>2'b10: Blend the whole frame<br>2'b11: Same functionality as 2'b00. |

### 12.3.4.5 Control Descriptor L0_5 Register

Figure  represents the control Descriptor L0_5 register. This register sets the maximum Chroma Keying values for RGB.

See Section 12.4.4.5, “Alpha and Chroma-key blending,” for a description of Chroma Keying.

Offsets:
0x010 (CtrlDescL0_5)
0x02C (CtrlDescL1_5)
0x048 (CtrlDescL2_5)
0x064 (CtrlDescL3_5)

Access: User read/write

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

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

**Figure 12-8. CtrlDescL0_5 Register**

**Table 12-10. CtrlDescL0_5 field descriptions**

| Field | Description |
|---|---|
| CKMAX_R | Chroma Keying Max Red Component |
| CKMAX_G | Chroma Keying Max Green Component |
| CKMAX_B | Chroma Keying Max Blue Component |

### 12.3.4.6 Control Descriptor L0_6 Register

Figure 12-9 represents the control descriptor L0_6 register. This register sets the minimum Chroma Keying values for RGB. See Section Section 12.4.4.5, “Alpha and Chroma-key blending,” for a description of Chroma Keying.

Offsets:
0x014 (CtrlDescL0_6)
0x030 (CtrlDescL1_6)
0x04C (CtrlDescL2_6)
0x068 (CtrlDescL3_6)

Access: User read/write

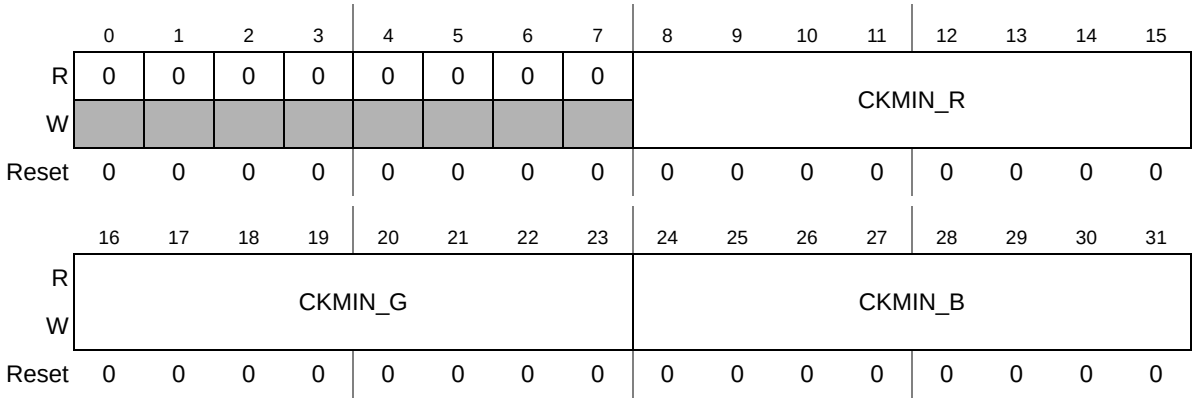



**Figure 12-9. CtrlDescL0_6 Register**

**Table 12-11. CtrlDescL0_6 field descriptions**

| Field | Description |
|---|---|
| CKMIN_R | Chroma Keying Minimum Red Component |
| CKMIN_G | Chroma Keying Minimum Green Component |
| CKMIN_B | Chroma Keying Minimum Blue Component |

### 12.3.4.7 Control Descriptor L0_7 Register

Figure 12-10 represents the Control Descriptor L0_7 Register.

Offsets:
0x018 (CtrlDescL0_7)
0x034 (CtrlDescL1_7)
0x050 (CtrlDescL2_7)
0x06C (CtrlDescL3_7)

Access: User read/write

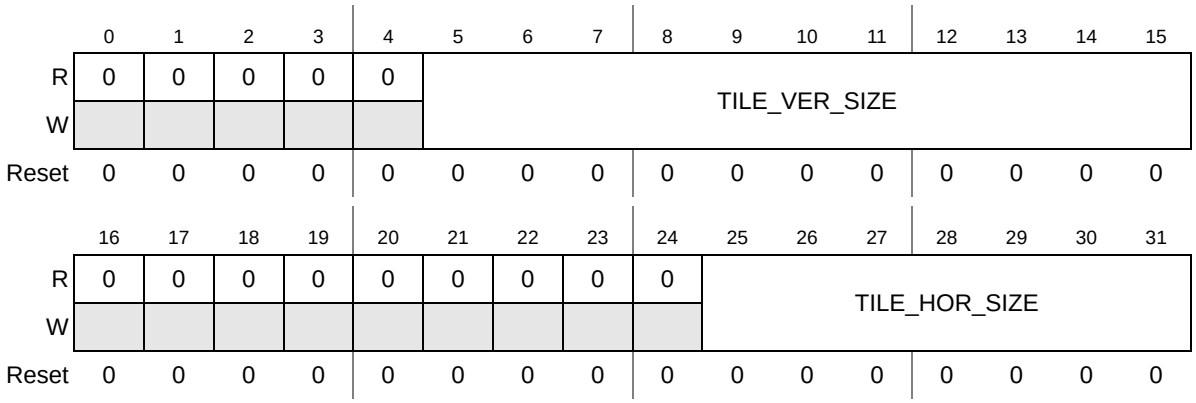



**Figure 12-10. Control Descriptor L0_7 Register**

**Table 12-12. CtrlDescL0_7 field descriptions**

| Field | Description |
|---|---|
| TILE_VER_SIZE | Height of the TILE (in pixels) |
| TILE_HOR_SIZE | Width of the TILE (in multiples of 16 pixels) |

For the other 3 layers, the Control Descriptor Register set is identical.

### 12.3.4.8 Control Descriptor Cursor 1 Register (CtrlDescCursor_1)

Figure 12-11 represents the Control Descriptor Cursor 1 register.

Offset 0x1C0 Access: User read/write

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

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

**Figure 12-11. Control Descriptor Cursor_1 Register (CtrlDescCursor_1)**

**Table 12-13. CtrlDescCursor_1 field descriptions**

| Field | Description |
|---|---|
| HEIGHT | Height of the cursor in pixels |
| WIDTH | Width of the cursor in pixels |

### 12.3.4.9 Control Descriptor Cursor 2 Register (CtrlDescCursor_2)

Figure 12-12 represents the Control descriptor Cursor 2 register.

Offset 0x1C4 Access: User read/write

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

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

**Figure 12-12. Control Descriptor Cursor 2 Register (CtrlDescCursor_2)**

**Table 12-14. CtrlDescCursor_2 field descriptions**

| Field | Description |
|---|---|
| POSY | Y position of the cursor in pixels |
| POSX | X position of the cursor in pixels |

## 12.3.4.10 Control Descriptor Cursor 3 Register (CtrlDescCursor_3)

Figure 12-13 represents the Control Descriptor Cursor 3 register.

Offset 0x1C8 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CUR_EN | | | | | | | | DEFAULT_CURSOR_COLOR[0:8] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DEFAULT_CURSOR_COLOR[9:24] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-13. Control Descriptor Cursor_3 Register (CtrlDescCursor_3)**

**Table 12-15. Control Descriptor Cursor_3 field descriptions**

| Field | Description |
|---|---|
| CUR_EN | Cursor Enable signal<br>1'b1: Enable the cursor<br>1'b0: Cursor is disabled |
| DEFAULT_CURSOR_COLOR | Default pixel color value for the cursor. In the DCULite, the pixel value for the cursor is fixed for a particular frame. |

### 12.3.4.11 Control Descriptor Cursor_4 Register (CtrlDescCursor_4)

Figure 12-14 represents the Control Descriptor Cursor_4 register.

Offset 0x1CC — Access: User read/write

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

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

**Figure 12-14. Control Descriptor Cursor_4 Register (CtrlDescCursor_4)**

**Table 12-16. CtrlDescCursor_4 field descriptions**

| Field | Description |
|---|---|
| HWC_BLINK_OFF | HWC blink register. Loads the counter value (number of frames) for which the cursor will remain turned OFF. |
| EN_BLINK | Enable the cursor blink mode.<br>1'b1:Enable the blink mode<br>1'b0:Disable the blink mode |
| HWC_BLINK_ON | HWC blink register. Loads the counter value (number of frames) for which the cursor will remain turned ON. |

### 12.3.4.12 DCULite Mode Register (DCU_MODE)

Figure 12-15 represents the DCU_MODE register.This register sets the mode in which DCULite is operating.

Offset 0x1D0 Access: User read/write

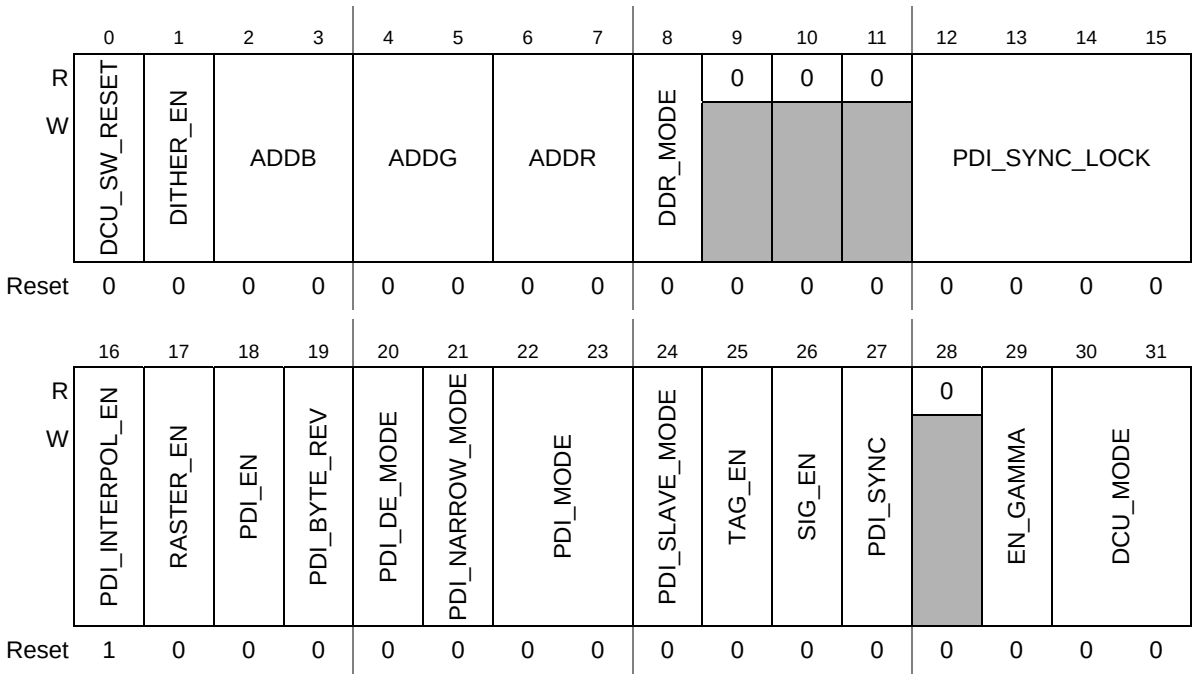



**Figure 12-15. DCULite Mode Register (DCU_MODE)**

**Table 12-17. DCU_MODE field descriptions**

| Field | Description |
|---|---|
| DCU_SW_RESET | Used to clear all the registers to reset state<br>1'b1:All the DCULite registers are put in reset state |
| DITHER_EN | Enable dithering mode<br>1'b1:Enabled.<br>1'b0:Disabled |
| ADDB | Two-bit value to be added to pixel blue component for dithering |
| ADDG | Two-bit value to be added to pixel green component for dithering |
| ADDR | Two-bit value to be added to pixel red component for dithering |
| DDR_MODE | Enables Special DDR Mode (see Section 12.4.9, "Special DDR mode") |
| PDI_SYNC_LOCK | Defines the number of frames which should be received by the PDI validation state machine before it locks and sets the PDI_LOCK_DET bit in the PDI Status Register (see Section 12.3.4.26, "PDI Status Register (PDI_STATUS)") |
| PDI_INTERPOL_EN | Control Bit to decide whether the conversion from YCbCr 4:2:2 to 4:4:4 needs to be done using interpolation or Chroma value is same for two pixels<br>1'b1:Interpolation is enabled.<br>1'b0:Chroma value is same for two pixels |
| RASTER_EN | Enables raster scanning of pixel data including the VSYNC and HSYNC signals and the pixel data. Changes to this bit take effect after the completion of the current frame.<br>1'b1: Enabled<br>1'b0: Disabled |

**Table 12-17. DCU_MODE field descriptions (continued)**

| Field | Description |
|---|---|
| PDI_EN | Enables the PDI<br>1'b1: Enabled<br>1'b0: Disabled |
| PDI_BYTE_REV | Controls the byte ordering in Narrow Mode<br>1'b0:LSB is followed by MSB data<br>1'b1:MSB is followed by LSB data |
| PDI_DE_MODE | Enables the PDI data Enable Mode. Here Data Enable is treated as an input.<br>1'b0: Value on data Enable signal is ignored<br>1'b1: Data enable signal must be present in incoming stream |
| PDI_NARROW_MODE | Enables the PDI Narrow Mode (refer to Section 12.8.1.5, "Normal and Narrow Mode")<br>1'b0: Narrow Mode is Disabled<br>1'b1: Narrow Mode is Enabled |
| PDI_MODE | Defines the different modes in which PDI is operating<br>2'b00: 8 bit monochrome data input<br>2'b01: 16 bit RGB 565 format<br>2'b10:18 bit RGB 666 data format.<br>2'b11:YCbCr data in 4:2:2 format. |
| PDI_SLAVE_MODE | Enables PDI slave Mode<br>1'b0:Disabled<br>1'b1:Enabled |
| TAG_EN | Enables the calculation of CRC only on the safety layers<br>1'b0: CRC calculated over the whole area of interest (area of interest given by SIG_DESC registers)<br>1'b1: calculates CRC only on safety enabled layers |
| SIG_EN | Enables the signature calculator block<br>1'b0: signature calculator is disabled<br>1'b1: signature calculator is enabled |
| PDI_SYNC | Decides whether the camera data needs external or internal synchronization.<br>1'b0: External Synchronization. The PDI receives the SYNC (HSYNC, VSYNC) signals from external source.<br>1'b1: Internal Synchronization. PDI extracts the SYNC information from the digital data.<br>**Note:** YCbCr Mode supports Internal Sync only. Therefore, when PDI_MODE = 3, PDI_SYNC must be set to 0. |
| EN_GAMMA | Enables/Disables the Gamma Correction<br>1'b0: Gamma correction is disabled<br>1'b1: Gamma Correction is enabled |
| DCU_MODE | DCULite operating mode<br>2'b00: DCULite off (pixel clock active if enabled by I/O)<br>2'b01: Normal mode. Panel content controlled by layer configuration.<br>2'b10: Test mode. DCULite disables all DMA fetches and all the pixels of an enabled layer take the value in the CLUT RAM selected by the respective LUOFFS field of control descriptor 4.<br>2'b11: Color bar generation. Panel content controlled by color bar registers. |

### 12.3.4.13 BGND Register

Figure 12-16 represents the BGND register.

Offset 0x1D4 Access: User read/write

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

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

**Figure 12-16. BGND Register**

**Table 12-18. BGND field descriptions**

| Field | Description |
|---|---|
| BGND_R | Red component of the default color displayed in the sectors where no layer is active |
| BGND_G | Green component of the default color displayed in the sectors where no layer is active |
| BGND_B | Blue component of the default color displayed in the sectors where no layer is active |

### 12.3.4.14 DISP_SIZE Register

Figure 12-17 represents the DISP_SIZE register

Offset 0x1D8 Access: User read/write

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

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

**Figure 12-17. DISP_SIZE Register**

**Table 12-19. DISP_SIZE field descriptions**

| Field | Description |
|---|---|
| DELTA_Y | Sets the display size vertical resolution (in pixels) |
| DELTA_X | Sets the display size horizontal resolution (in pixels/16) |

### 12.3.4.15 HSYN_PARA Register

Figure 12-18 represents the HSYN_PARA register. HSYN_PARA register sets timing parameters related to the horizontal synchronization signal generation. The fields FP_H, BP_H, and PW_H stand for HSYNC signal front-porch, back-porch, and active pulse width, respectively.

Offset 0x1DC Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | BP_H | | | | | | | | | 0 | 0 | PW_H[0:4] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PW_H[5:8] | | | | | 0 | 0 | FP_H | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**Figure 12-18. HSYN_PARA Register**

**Table 12-20. HSYN_PARA field descriptions**

| Field | Description |
|---|---|
| BP_H | HSYNC back-porch pulse width (in pixel clock cycles) |
| PW_H | HSYNC active pulse width (in pixel clock cycles) |
| FP_H | HSYNC front-porch pulse width (in pixel clock cycles) |

### 12.3.4.16 VSYN_PARA Register

Figure 12-19 represents the VSYN_PARA register. VSYN_PARA register sets timing parameters related to the vertical synchronization signal generation. The fields FP_V, BP_V, and PW_V stand for VSYNC signal front-porch, back-porch, and active pulse width, respectively.

Offset 0x1E0 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | BP_V | | | | | | | | | 0 | 0 | PW_V[0:4] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PW_V[4:05:8] | | | | | 0 | 0 | FP_V | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**Figure 12-19. VSYN_PARA Register**

**Table 12-21. VSYN_PARA field descriptions**

| Field | Description |
|---|---|
| BP_V | VSYNC back-porch pulse width (in pixel clock cycles) |
| PW_V | VSYNC active pulse width (in pixel clock cycles). |
| FP_V | VSYNC front-porch pulse width (in pixel clock cycles) |

### 12.3.4.17 SYN_POL Register

Figure 12-20 represents the SYN_POL register. SYN_POL register selects polarity for corresponding synchronize signals (HSYNC, VSYNC, CSYNC), and controls the bypass of HSYNC or VSYNC with CSYNC signal.

Offset 0x1E4 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | |
| W | | | | | | INV_PDI_DE | INV_PDI_HS | INV_PDI_VS | INV_PDI_CLK | INV_PXCK | NEG | BP_VS | BP_HS | INV_CS | INV_VS | INV_HS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-20. SYN_POL Register**

**Table 12-22. SYN_POL field descriptions**

| Field | Description |
|---|---|
| INV_PDI_DE | Polarity change of PDI input data Enable.<br>1'b0: DE is active high<br>1'b1: DE is active low |
| INV_PDI_HS | Polarity change of PDI input HSYNC.<br>1'b0: HSYNC is active high<br>1'b1: HSYNC is active low |
| INV_PDI_VS | Polarity change of PDI input VSYNC.<br>1'b0: VSYNC is active high<br>1'b1: VSYNC is active low |
| INV_PDI_CLK | Polarity change of PDI input Clock.<br>1'b0: DCULite samples data on the rising edge<br>1'b1: DCULite samples data on the falling edge |

**Table 12-22. SYN_POL field descriptions (continued)**

| Field | Description |
|---|---|
| INV_PXCK | Polarity change of Pixel Clock.<br>1'b0: Display samples data on the falling edge<br>1'b1: Display samples data on the rising edge |
| NEG | Indicates if value at the output (pixel data output) needs to be negated.<br>1'b0: Output is to remain same<br>1'b1: Output to be negated |
| BP_VS | Bypass Vertical Synchronize Signal (internal pin muxing).<br>1'b0: Do not bypass VSYNC signal output<br>1 'b1: CSYNC bypass VSYNC signal, output CSYNC instead of VSYNC |
| BP_HS | Bypass Horizontal Synchronize Signal (internal pin muxing).<br>1'b0: Do not bypass HSYNC signal output<br>1'b1: CSYNC bypass HSYNC signal, output CSYNC instead of HSYNC |
| INV_CS | Invert Composite Synchronize Signal.<br>1'b0: Not invert CSYNC signal, active HIGH<br>1 'b1: Invert CSYNC signal, active LOW |
| INV_VS | Invert Vertical Synchronize Signal<br>1'b0: Not invert VSYNC signal, active HIGH<br>1 'b1: Invert VSYNC signal, active LOW |
| INV_HS | Invert Horizontal Synchronize Signal.<br>1'b0: Not invert HSYNC signal, active HIGH<br>1'b1: Invert HSYNC signal, active LOW |

## 12.3.4.18 Threshold Register (THRESHOLD)

Figure 12-21 represents the Threshold Register.

Offset 0x1E8 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OUT_BUF_HIGH | | | | | | | | OUT_BUF_LOW | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

**Figure 12-21. Threshold Register (THRESHOLD)**

**Table 12-23. THRESHOLD field descriptions**

| Field | Description |
|---|---|
| LS_BF_VS | Lines before VSYNC threshold value. It is a threshold value used to generate the ls_bf_vs interrupt status. Sets the number of lines before VSYNC that the interrupt would be generated. |
| OUT_BUF_HIGH | Output buffer filling high threshold (in pixels). It is used to generate the datapath clock enable signal. Gates the datapath when output buffer filling is higher than OUT_BUF_HIGH. |
| OUT_BUF_LOW | Output buffer filling low Threshold (in pixels).This value is used to generate the underrun exception. |

### 12.3.4.19 Interrupt Status Register (INT_STATUS)

Figure 12-22 indicates the interrupt status register. The DCULite has only one interrupt signal, and the CPU reads the INT_STATUS register to decide which exception occurs when an interrupt is detected.

Offset 0x1EC — Access: User read

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMA_TRANS_FINISH | 0 | 0 | IPM_ERROR | PROG_END | P2_FIFO_HI_FLAG | P2_FIFO_LO_FLAG | P1_FIFO_HI_FLAG | P1_FIFO_LO_FLAG | CRC_OVERFLOW | CRC_READY | VS_BLANK | LS_BF_VS | UNDRUN | VSYNC |
| W | | w1c | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-22. Interrupt Status Register (INT_STATUS)**

**Table 12-24. INT_STATUS field descriptions**

| Field | Description |
|---|---|
| DMA_TRANS_FINISH | Interrupt signal which indicates that the DCULite DMA has fetched the last pixel of data from the memory |
| IPM_ERROR | Interrupt flag, which indicates that an error response has been received by the IP from the AHB bus |
| PROG_END | Interrupt signal which indicates that the duration for programming of DCULite registers and internal memories is finished |
| P2_FIFO_HI_FLAG | Interrupt signal to indicate that High threshold has been reached for plane 2 (FGplane) input buffer |

**Table 12-24. INT_STATUS field descriptions (continued)**

| Field | Description |
|---|---|
| P2_FIFO_LO_FLAG | Interrupt signal to indicate that Low threshold has been reached for plane 2 (FGplane) input buffer |
| P1_FIFO_HI_FLAG | Interrupt signal to indicate that High threshold has been reached for plane 1 (BGplane) input buffer |
| P1_FIFO_LO_FLAG | Interrupt signal to indicate that Low threshold has been reached for plane 1 (BGplane) input buffer |
| CRC_OVERFLOW | Interrupt signal to indicate that CRC_ready has not been serviced and CRC has been calculated for the next frame |
| CRC_READY | Interrupt signal to indicate CRC calculation is done and ready to be compared with precomputed CRC value by the software |
| VS_BLANK | Interrupt signal to indicate vertical blanking period. This is the period in which all the registers that affect the visible state of the layers need to be latched. This is needed so that CPU writes to the register while the display is being updated does not cause any errors. |
| LS_BF_VS | Lines Before Vsync interrupt. It is generated threshold LS_BF_VS number of lines ahead of the vertical front porch (FP_V) if enabled. The CPU can program the registers after LS_BF_VS interrupt. |
| UNDRUN | Under Run Exception Interrupt. Asserted when display needs data and output buffer filling is lower than or equal to the OUT_BUF_LOW threshold. Interrupt is cleared when the data in the output buffer is greater than threshold and CPU writes 1 to this bit. |
| VSYNC | Vertical Synchronize Interrupt. If enabled, an interrupt is generated at the beginning of a frame.. |

### 12.3.4.20 Interrupt Mask Register (INT_MASK)

Figure 12-23 represents the interrupt mask register.This register enables or masks corresponding interrupt.

Offset 0x1F0 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | M_DMA_TRANS_FINISH | 0 | 0 | M_IPM_ERROR | M_PROG_END | M_P2_FIFO_HI_FLAG | M_P2_FIFO_LO_FLAG | M_P1_FIFO_HI_FLAG | M_P1_FIFO_LO_FLAG | M_CRC_OVERFLOW | M_CRC_READY | M_VS_BLANK | M_LS_BF_VS | M_UNDRUN | M_VSYNC |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 12-23. Interrupt Mask Register (INT_MASK)**

Table 12-25. INT_MASK field descriptions

| Field | Description |
|---|---|
| M_DMA_TRANS_FINISH | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_IPM_ERROR | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_PROG_END | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_P2_FIFO_HI_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_P2_FIFO_LO_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_P1_FIFO_HI_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_P1_FIFO_LO_FLAG | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_CRC_OVERFLOW | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_CRC_READY | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_VS_BLANK | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_LS_BF_VS | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_UNDRUN | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |
| M_VSYNC | Mask the interrupt<br>1'b1: interrupt is masked<br>1'b0:Not masked |

### 12.3.4.21 COLBAR registers

The COLBAR registers are used to generate color bars in functional test mode. Eight different pixel values are taken as input data, to display 8 color bars on the display.

### 12.3.4.21.1 COLBAR_1 register

Offset 0x1F4 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_1_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_1_G | | | | | | | | COLBAR_1_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-24. COLBAR_1 register**

### 12.3.4.21.2 COLBAR_2 register

Offset 0x1F8 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_2_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_2_G | | | | | | | | COLBAR_2_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 12-25. COLBAR_2 register (blue)**

### 12.3.4.21.3 COLBAR_3 register

Offset 0x1FC — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_3_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_3_G | | | | | | | | COLBAR_3_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 12-26. COLBAR_3 register (cyan)**

### 12.3.4.21.4 COLBAR_4 register

Offset 0x200 — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_4_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_4_G | | | | | | | | COLBAR_4_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-27. COLBAR_4 register (green)**

### 12.3.4.21.5 COLBAR_5 register

Offset 0x204 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_5_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_5_G | | | | | | | | COLBAR_5_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-28. COLBAR_5 register (yellow)**

### 12.3.4.21.6 COLBAR_6 Register

Offset 0x208 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_6_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_6_G | | | | | | | | COLBAR_6_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-29. COLBAR_6 register (red)**

### 12.3.4.21.7 COLBAR_7 register

Offset 0x20C

Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_7_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_7_G | | | | | | | | COLBAR_7_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 12-30. COLBAR_7 register (purple)**

### 12.3.4.21.8 COLBAR_8 register

Offset 0x210

Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | COLBAR_8_R | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COLBAR_8_G | | | | | | | | COLBAR_8_B | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 12-31. COLBAR_8 register (white)**

## 12.3.4.22 Clock Divider Ratio (DIV_RATIO) register

Figure 12-32 shows Clock Divider Ratio (DIV_RATIO) register.

Offset 0x214 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DIV_RATIO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**Figure 12-32. Clock Divider Register (DIV_RATIO)**

**Table 12-26. DIV_RATIO field descriptions**

| Field | Description |
|---|---|
| DIV_RATIO | Specifies the divide value for the input clock. Used to generate the pixel clock to support different types of displays. To divide by N, set the DIV_RATIO to (N-1). |

## 12.3.4.23 SIGN_CALC_1 register

Figure 12-33 presents the register for vertical/horizontal size of the area for CRC calculation.

Offset 0x218 Access: User read/write

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

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

**Figure 12-33. SIGN_CALC_1 register**

**Table 12-27. SIGN_CALC_1 field descriptions**

| Field | Description |
|---|---|
| SIG_VER_SIZE | Vertical size of the window of interest of pixels for CRC calculation (in pixels) |
| SIG_HOR_SIZE | Horizontal size of window of interest of pixels for CRC calculations (in pixels) |

### 12.3.4.24 SIGN_CALC_2 register

Figure 12-34 represents the register for position of the window of interest for CRC calculation.

Offset 0x21C — Access: User read/write

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

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

**Figure 12-34. SIGN_CALC_2 register**

**Table 12-28. SIGN_CALC_2 field descriptions**

| Field | Description |
|---|---|
| 6–15<br>SIG_VER_POS | Vertical position of the window of interest of pixels for CRC calculation (in pixels) |
| 22–31<br>SIG_HOR_POS | Horizontal position of window of interest of pixels for CRC calculation (in pixels) |

### 12.3.4.25 CRC_VAL register

Figure 12-35 represents the register presenting the CRC value to the software for comparison.

Offset 0x220 — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_VAL[0:15] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_VAL[16:31] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-35. CRC_VAL register**

**Table 12-29. CRC_VAL field descriptions**

| Field | Description |
|---|---|
| CRC_VAL | CRC value calculated for safety enabled layers to be presented to the software for comparison. |

### 12.3.4.26 PDI Status Register (PDI_STATUS)

Figure 12-36 represents the PDI status register.

Offset 0x224 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | PDI_BLANKING_ERR | PDI_ECC_ERR2 | PDI_ECC_ERR1 | PDI_LOCK_LOST | PDI_LOCK_DET | PDI_VSYNC_DET | PDI_HSYNC_DET | PDI_DE_DET | PDI_CLK_LOST | PDI_CLK_DET |
| W | | | | | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-36. PDI Status Register (PDI_STATUS)**

**Table 12-30. PDI_STATUS field descriptions**

| Field | Description |
|---|---|
| PDI_BLANKING_ERR | Status bit to inform the software that 80h,10h sequence is not present during the blanking period in internal sync mode.<br>1'b1:Correct data sequence not present in blanking period<br>1'b0:Correct data sequence present in blanking period |
| PDI_ECC_ERR2 | Status bit to inform the software about multibit bit error that is detected.<br>1'b1: Multibit ECC error detected<br>1'b0: Multibit ECC error is not detected |
| PDI_ECC_ERR1 | Status bit to inform the software about one bit error is detected.<br>1'b1: One bit ECC error detected<br>1'b0: One bit ECC error is not detected |
| PDI_LOCK_LOST | Status bit to inform the software that frame lock is lost.<br>1'b1: Frame lock is lost<br>1'b0: Frame is locked |
| PDI_LOCK_DET | Status bit to inform the software PDI is frame locked to the camera interface.<br>1'b1: Frame lock is detected<br>1'b0: Waiting for frame to lock |

**Table 12-30. PDI_STATUS field descriptions (continued)**

| Field | Description |
|---|---|
| PDI_VSYNC_DET | Status bit to inform the software that vsync for the camera data has been detected.<br>1'b1: pdi_vsync is detected<br>1'b0: pdi_vsync not detected |
| PDI_HSYNC_DET | Status bit to inform the software that hsync for the camera data has been detected.<br>1'b1: pdi_hsync is detected<br>1'b0: pdi_hsync not detected |
| PDI_DE_DET | Status bit to inform the software that data Enable for the camera data has been detected.<br>1'b1: pdi_de is detected<br>1'b0: pdi_de not detected |
| PDI_CLK_LOST | Status bit to inform the software that pdi_clk is lost<br>1'b1: pdi_clk is lost<br>1'b0: pdi_clk is present |
| PDI_CLK_DET | Status bit to inform the software that clock for the camera data has been detected.<br>1'b1: pdi_clk is detected<br>1'b0: pdi_clk not detected |

### 12.3.4.27 PDI Status Mask Register (MASK_PDI_STATUS)

Figure 12-37 represents the Mask PDI status register

Offset 0x228 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | M_PDI_BLANKING_ERR | M_PDI_ECC_ERR2 | M_PDI_ECC_ERR1 | M_PDI_LOCK_LOST | M_PDI_LOCK_DET | M_PDI_VSYNC_DET | M_PDI_HSYNC_DET | M_PDI_DE_DET | M_PDI_CLK_LOST | M_PDI_CLK_DET |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 12-37. PDI status mask register (MASK_PDI_STATUS)**

**Table 12-31. MASK_PDI_STATUS field descriptions**

| Field | Description |
|---|---|
| M_PDI_BLANKING_ERR | Mask the PDI_BLANKING_ERR bit<br>1'b1: Mask the PDI_BLANK_ERR interrupt<br>1'b0: Do not mask the PDI_BLANK_ERR interrupt |
| M_PDI_ECC_ERR2 | Mask the PDI_ECC_ERR2 bit<br>1'b1: Mask the PDI_ECC_ERR2 interrupt<br>1'b0: Do not mask the PDI_ECC_ERR2 interrupt |
| M_PDI_ECC_ERR1 | Mask the PDI_ECC_ERR1 bit<br>1'b1: Mask the PDI_ECC_ERR1 interrupt<br>1'b0: Do not mask the PDI_ECC_ERR1 interrupt |
| M_PDI_LOCK_LOST | Mask the PDI_LOCK_LOST bit<br>1'b1: Mask the PDI_LOCK_LOST interrupt<br>1'b0: Do not mask the PDI_LOCK_LOST interrupt |
| M_PDI_LOCK_DET | Mask the PDI_LOCK_DET bit<br>1'b1: Mask the PDI_LOCK_DET interrupt<br>1'b0: Do not mask the PDI_LOCK_DET interrupt |
| M_PDI_VSYNC_DET | Mask the PDI_VSYNC_DET bit<br>1'b1: Mask the PDI_VSYNC_DET interrupt<br>1'b0: Do not mask the PDI_VSYNC_DET interrupt |
| M_PDI_HSYNC_DET | Mask the PDI_HSYNC_DET bit<br>1'b1: Mask the PDI_HSYNC_DET interrupt<br>1'b0: Do not mask the PDI_HSYNC_DET interrupt |
| M_PDI_DE_DET | Mask the PDI_DE_DET bit<br>1'b1: Mask the PDI_DE_DET interrupt<br>1'b0: Do not mask the PDI_DE_DET interrupt |
| M_PDI_CLK_LOST | Mask the PDI_CLK_LOST bit<br>1'b1: Mask the PDI_CLK_LOST interrupt<br>1'b0: Do not mask the PDI_CLK_LOST interrupt |
| M_PDI_CLK_DET | Mask the PDI_CLK_DET bit<br>1'b1: Mask the PDI_CLK_DET interrupt<br>1'b0: Do not mask the PDI_CLK_DET interrupt |

### 12.3.4.28 PARR_ERR Status Register (PARR_ERR_STATUS)

Figure 12-38 shows the parameter error status register.

An error in a layer can occur when there is an automatic error checking mechanism when a layer is enabled that detects a non-valid horizontal size and color format combination. See Section 12.4.4.3, "Layer size and positioning," for details.

These errors are grouped into a single bit error for each layer. The parameter error specific to each layer is signalled only when the layer is enabled.

RLE_ERR occurs when more than one layer has its RLE_EN bit set (Control Descriptor 4).

DISP_ERR occurs when the size of display (height or width) is set to zero or when the pulse width of HSYNC/VSYNC is programmed as zero.

SIG_ERR occurs when the area of interest for calculating CRC value is programmed with values which are outside the display.

HWC_ERR occurs if size of cursor programmed is greater than memory size (256x32). See Section 12.4.5, “Hardware cursor,” for further details on how cursor can be programmed.

Offset 0x22C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RLE_ERR | HWC_ERR | SIG_ERR | DISP_ERR |
| W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | L3_PARR_ERR | L2_PARR_ERR | L1_PARR_ERR | L0_PARR_ERR |
| W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-38. Parameter Error Status Register (PARR_ERR_STATUS)**

**Table 12-32. PARR_ERR_STATUS field descriptions**

| Field | Description |
|---|---|
| RLE_ERR | Error signal to indicate that more than one layer has RLE mode enabled. |
| HWC_ERR | Interrupt signal to indicate HWC error. This can occur if HWC position is out of display area or cursor memory is bigger than the HWC size. When this occurs, the HWC is disabled. |
| SIG_ERR | Interrupt occurs whenever the area of interest specified by SIG_CALC register is outside the display size.<br>1’b0: SIG_ERR is not set<br>1’b1: SIG_ERR is set |
| DISP_ERR | Interrupt occurs whenever width and height of display, pulse width (both vertical and horizontal sync) value is 0.<br>1’b0: DISP_ERR is not set<br>1’b1: DISP_ERR is set |
| L3_PARR_ERR | Interrupt occurs whenever there is an error in layer 3.<br>1’b0: Parameter error is not set<br>1’b1: Parameter error is set |
| L2_PARR_ERR | Interrupt occurs whenever there is an error in layer 2.<br>1’b0: Parameter error is not set<br>1’b1: Parameter error is set |

**Table 12-32. PARR_ERR_STATUS field descriptions (continued)**

| Field | Description |
|---|---|
| RLE_ERR | Error signal to indicate that more than one layer has RLE mode enabled. |
| L1_PARR_ERR | Interrupt occurs whenever there is an error in layer 1.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |
| L0_PARR_ERR | Interrupt occurs whenever there is an error in layer 0.<br>1'b0: Parameter error is not set<br>1'b1: Parameter error is set |

### 12.3.4.29 MASK_PARR_ERR_STATUS register

Figure 12-39 shows the mask register for parameter error status register.

Offset 0x230 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | M_RLE_ERR | M_HWC_ERR | M_SIG_ERR | M_DISP_ERR |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | M_L3_PARR_ERR | M_L2_PARR_ERR | M_L1_PARR_ERR | M_L0_PARR_ERR |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**Figure 12-39. Mask parameter error status register (MASK_PARR_ERR_STATUS)**

**Table 12-33. MASK_PARR_ERR_STATUS field descriptions**

| Field | Description |
|---|---|
| M_RLE_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| M_HWC_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| M_SIG_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |

Table 12-33. MASK_PARR_ERR_STATUS field descriptions (continued)

| Field | Description |
|---|---|
| M_DISP_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| M_L3_PARR_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| M_L2_PARR_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| M_L1_PARR_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |
| M_L0_PARR_ERR | Mask the interrupt<br>1'b1: mask the interrupt<br>1'b0: Do not mask interrupt |

## 12.3.4.30 THRESHOLD_INPUT BUF_1 Register

Figure 12-40 shows the threshold register for input buffer.

Offset 0x234 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | INP_BUF_P2_HI | | | | | | | 0 | INP_BUF_P2_LO | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | INP_BUF_P1_HI | | | | | | | 0 | INP_BUF_P1_LO | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 12-40. Threshold input buffer 1 register (THRESHOLD_INPUT BUF_1)

Table 12-34. THRESHOLD_INPUT BUF_1 field descriptions

| Field | Description |
|---|---|
| INP_BUF_p2_hi | High Threshold for input buffer for plane 2 (FGPlane) |
| INP_BUF_p2_lo | Low Threshold for input buffer for plane 2 (FGPlane) |
| INP_BUF_p1_hi | High Threshold for input buffer for plane 1 (BGPlane) |
| INP_BUF_p1_lo | Low Threshold for input buffer for plane 1 (BGPlane) |

### 12.3.4.31 Luma Component Register (LUMA)

Figure 12-41 represents the LUMA component register.

Offset 0x23C — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Y_RED | | | | | | | | | | 0 | Y_GREEN[0:4] | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Y_GREEN[5:9] | | | | | 0 | Y_BLUE | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

**Figure 12-41. LUMA Component Register (LUMA)**

**Table 12-35. LUMA field descriptions**

| Field | Description |
|---|---|
| Y_RED | Luminance Coefficient for Red Matrix |
| Y_GREEN | Luminance Coefficient for Green Matrix |
| Y_BLUE | Luminance Coefficient for Blue Matrix |

### 12.3.4.32 Red Chroma Components (RED)

Figure 12-42 represents the Red Chroma component register.

Offset 0x240 — Access: User read/write

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

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

**Figure 12-42. RED Chroma Component Register (RED)**

**Table 12-36. RED field descriptions**

| Field | Description |
|---|---|
| CR_RED | Cr Coefficient for Red Matrix |
| CB_RED | Cb Coefficient for Red Matrix |

### 12.3.4.33 Green Chroma Component Register (GREEN)

Figure 12-43 represents the Green Chroma component register

Offset 0x244 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | CB_GREEN | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

**Figure 12-43. Green Chroma Component Register (GREEN)**

**Table 12-37. GREEN field descriptions**

| Field | Description |
|---|---|
| CR_GREEN | Cr Coefficient for Green Matrix |
| CB_GREEN | Cb Coefficient for Green Matrix |

### 12.3.4.34 Blue Chroma Component Register (BLUE)

Figure 12-44 represents the Blue Chroma component register.

Offset 0x248 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | CB_BLUE | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 12-44. Blue Chroma Component Register (BLUE)**

**Table 12-38. BLUE field descriptions**

| Field | Description |
|---|---|
| CR_BLUE | Cr Coefficient for Blue Matrix |
| CB_BLUE | Cb Coefficient for Blue Matrix |

## 12.3.4.35 CRC_POS Register

Figure 12-45 represents the CRC_POS register.

Offset 0x24C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_POS[0:15] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CRC_POS[16:31] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-45. CRC_POS Register**

**Table 12-39. CRC_POS field descriptions**

| Field | Description |
|---|---|
| CRC_POS | CRC position value calculated for safety enabled layers to be presented to the software for comparison |

### 12.3.4.36 FG0_fcolor Register

Figure 12-46 represents the FG0_fcolor register.

Offsets:
0x250 (FG0_fcolor)
0x258 (FG1_fcolor)
0x260 (FG2_fcolor)
0x268 (FG3_fcolor)

Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FG0_fcolor[0:7] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FG0_fcolor[8:23] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-46. FG0_fcolor Register**

**Table 12-40. FG0_fcolor field descriptions**

| Field | Description |
|---|---|
| FG0_fcolor | Foreground color for layer FG0 for pre-blending engine |

### 12.3.4.37 FG0_bcolor

Figure 12-47 represents the FG0_bcolor register.

Offsets:
0x254 (FG0_bcolor)
0x25C (FG1_bcolor)
0x264 (FG2_bcolor)
0x26C (FG3_bcolor)

Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FG0_bcolor[0:7] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FG0_bcolor[8:23] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-47. FG0_bcolor Register**

**Table 12-41. FG0_bcolor Register field descriptions**

| Field | Description |
|---|---|
| FG0_bcolor | Background color for layer FG0 for pre-blending engine |

## 12.3.4.38 LYR_INTPOL_EN

Figure 12-48 represents the LYR_INTPOL_EN register.

Offset: 0x2D0 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 12-48. LYR_INTPOL_EN register**

**Table 12-42. LYR_INTPOL_EN field descriptions**

| Field | Description |
|---|---|
| EN | Interpolation Enable bit for DCULite Layer coded in YCbCr422 format.<br>This bit controls whether the chroma value for each pixel in the co.nversion from YCbCr 4:2:2 to 4:4:4 should use interpolation or the same value for both pixels<br>0 Chroma value is same for two pixels<br>1 Interpolation is enabled. |

## 12.3.4.39 LYR_LUMA_COMPONENT

Figure 12-49 represents the Layer LUMA component register.

Offset: 0x2D4 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Y_RED | | | | | | | | | | 0 | Y_GREEN[0:4] | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Y_GREEN[5:9] | | | | | 0 | Y_BLUE | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |

**Figure 12-49. Layer LUMA Component Register (LYR_LUMA_COMPONENT)**

**Table 12-43. LYR_LUMA_COMPONENT field descriptions**

| Field | Description |
|---|---|
| Y_RED | Luminance Coefficient for Red Matrix |
| Y_GREEN | Luminance Coefficient for Green Matrix |
| Y_BLUE | Luminance Coefficient for Blue Matrix |

## 12.3.4.40 LYR_CHROMA_RED

Figure 12-50 represents the Layer Red Chroma component register.

Offset: 0x2D8 Access: User read/write

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

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

**Figure 12-50. Layer RED Chroma Component Register (LYR_CHROMA_RED)**

**Table 12-44. LYR_CHROMA_RED field descriptions**

| Field | Description |
|---|---|
| CR_RED | Cr Coefficient for Red Matrix |
| CB_RED | Cb Coefficient for Red Matrix |

### 12.3.4.41 LYR_CHROMA_GREEN

Figure 12-51 represents the Layer Green Chroma component register.

Offset: 0x2DC Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | CB_GREEN | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

**Figure 12-51. Layer GREEN Chroma Component Register (LYR_CHROMA_GREEN)**

**Table 12-45. LYR_CHROMA_GREEN field descriptions**

| Field | Description |
|---|---|
| CR_GREEN | Cr Coefficient for Green Matrix |
| CB_GREEN | Cb Coefficient for Green Matrix |

### 12.3.4.42 LYR_CHROMA_BLUE

Figure 12-52 represents the Layer Blue Chroma component register.

Offset: 0x2E0 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | CB_BLUE | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 12-52. Layer BLUE chroma component Register (LYR_CHROMA_BLUE)**

**Table 12-46. LYR_CHROMA_BLUE field descriptions**

| Field | Description |
|---|---|
| CR_BLUE | Cr Coefficient for Blue Matrix |
| CB_BLUE | Cb Coefficient for Blue Matrix |

### 12.3.4.43 COMP_IMSIZE

Figure 12-53 represents the Compression Image Size Register.

Offset: 0x2E4 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | COMP_IMSIZE[21:16] | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | COMP_IMSIZE[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-53. COMP_IMSIZE Register**

**Table 12-47. COMP_IMSIZE field descriptions**

| Field | Description |
|---|---|
| COMP_IMSIZE | Compressed Image size in bytes for RLE coded layer |

### 12.3.4.44 Global Protection Register

Figure 12-54 represents the Global Protection register.

Offset: 0x300 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 12-54. Global Protection Register**

**Table 12-48. Global Protection Register Field Descriptions**

| Field | Description |
|---|---|
| HLB | Hard Lock Bit. This bit cannot be cleared once it is set by software. It can only be cleared by a system reset.<br>1'b1:All SLB's are write protected & cannot be modified<br>1'b0:All SLB's are accessible & can be modified |

## 12.3.4.45 Soft Lock Bit Register L0

Figure 12-55 represents the Soft Lock Bit Register for Layer0. This is used to protect the 7 control descriptor layer registers for Layer0.

Offset: 0x304 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | SLB_L0_1 | SLB_L0_2 | SLB_L0_3 | SLB_L0_4 | 0 | 0 | 0 | 0 | SLB_L0_5 | SLB_L0_6 | SLB_L0_7 | 0 |
| W | WEN_LO_1 | WEN_LO_2 | WEN_LO_3 | WEN_LO_4 | | | | | WEN_LO_5 | WEN_LO_6 | WEN_LO_7 | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 12-55. Soft Lock Register L0**

**Table 12-49. Soft Lock Register L0 field descriptions**

| Field | Description |
|---|---|
| WEN_L0_1 | Write Enable for Soft Lock Bit SLB_L0_1<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L0_2 | Write Enable for Soft Lock Bit SLB_L0_2<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L0_3 | Write Enable for Soft Lock Bit SLB_L0_3<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L0_4 | Write Enable for Soft Lock Bit SLB_L0_4<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |

Table 12-49. Soft Lock Register L0 field descriptions (continued)

| Field | Description |
|---|---|
| SLB_L0_1 | Soft Lock Bit for Control Desc L0_1 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L0_2 | Soft Lock Bit for Control Desc L0_2 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L0_3 | Soft Lock Bit for Control Desc L0_3 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L0_4 | Soft Lock Bit for Control Desc L0_4 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| WEN_L0_5 | Write Enable for Soft Lock Bit SLB_L0_5<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L0_6 | Write Enable for Soft Lock Bit SLB_L0_6<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L0_7 | Write Enable for Soft Lock Bit SLB_L0_7<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| SLB_L0_5 | Soft Lock Bit for Control Desc L0_5 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L0_6 | Soft Lock Bit for Control Desc L0_6 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L0_7 | Soft Lock Bit for Control Desc L0_7 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

### 12.3.4.46 Soft Lock Bit Register L1

Figure 12-56 represents the Soft Lock Bit Register for Layer1. This is used to protect the 7 control descriptor layer registers for Layer1.

Offset: 0x308 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | SLB_L1_1 | SLB_L1_2 | SLB_L1_3 | SLB_L1_4 | 0 | 0 | 0 | 0 | SLB_L1_5 | SLB_L1_6 | SLB_L1_7 | 0 |
| W | WEN_L1_1 | WEN_L1_2 | WEN_L1_3 | WEN_L1_4 | | | | | WEN_L1_5 | WEN_L1_6 | WEN_L1_7 | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 12-56. Soft Lock Register L1**

**Table 12-50. Soft Lock Register L1 field descriptions**

| Field | Description |
|---|---|
| WEN_L1_1 | Write Enable for Soft Lock Bit SLB_L1_1<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L1_2 | Write Enable for Soft Lock Bit SLB_L1_2<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L1_3 | Write Enable for Soft Lock Bit SLB_L1_3<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L1_4 | Write Enable for Soft Lock Bit SLB_L1_4<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| SLB_L1_1 | Soft Lock Bit for Control Desc L1_1 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L1_2 | Soft Lock Bit for Control Desc L1_2 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L1_3 | Soft Lock Bit for Control Desc L1_3 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L1_4 | Soft Lock Bit for Control Desc L1_4 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

**Table 12-50. Soft Lock Register L1 field descriptions (continued)**

| Field | Description |
|---|---|
| WEN_L1_5 | Write Enable for Soft Lock Bit SLB_L1_5<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L1_6 | Write Enable for Soft Lock Bit SLB_L1_6<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L1_7 | Write Enable for Soft Lock Bit SLB_L1_7<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| SLB_L1_5 | Soft Lock Bit for Control Desc L1_5 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L1_6 | Soft Lock Bit for Control Desc L1_6 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L1_7 | Soft Lock Bit for Control Desc L1_7 Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

## 12.3.4.47 Soft Lock DISP_SIZE Register

Figure 12-57 represents the Soft Lock DISP_SIZE register.

Offset: 0x30C — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | WEN_DISP | | | | SLB_DISP | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 12-57. Soft Lock DISP_SIZE Register**

**Table 12-51. Soft Lock DISP_SIZE Register field descriptions**

| Field | Description |
|---|---|
| WEN_DISP | Write Enable for Soft Lock Bit SLB_DISP<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| SLB_DISP | Soft Lock Bit for DISP_SIZE Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

## 12.3.4.48 Soft Lock HSYNC/VSYNC PARA Register

Figure 12-58 represents the Soft Lock HSYNC/VSYNC register.

Offset: 0x310 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | WEN_HSYNC | WEN_VSYNC | | | SLB_HSYNC | SLB_VSYNC | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 12-58. Soft Lock HSYNC/VSYNC PARA Register**

**Table 12-52. Soft Lock HSYNC/VSYNC PARA Register field descriptions**

| Field | Description |
|---|---|
| WEN_HSYNC | Write Enable for Soft Lock Bit SLB_HSYNC<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_VSYNC | Write Enable for Soft Lock Bit SLB_VSYNC<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |

Table 12-52. Soft Lock HSYNC/VSYNC PARA Register field descriptions (continued)

| Field | Description |
|---|---|
| SLB_HSYNC | Soft Lock Bit for HSYNC Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_VSYNC | Soft Lock Bit for VSYNC Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

## 12.3.4.49 Soft Lock POL Register

Figure 12-59 represents the Soft Lock POL Register.

Offset: 0x314 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | WEN_POL | | | | SLB_POL | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

Figure 12-59. Soft Lock POL Register

Table 12-53. Soft Lock POL Register field descriptions

| Field | Description |
|---|---|
| WEN_POL | Write Enable for Soft Lock Bit SLB_POL<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| SLB_POL | Soft Lock Bit for SYN_POL Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

## 12.3.4.50 Soft Lock L0_TRANSP Register

Figure 12-60 represents the Soft Lock L0_TRANSP register.

Offset: 0x318 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | WEN_L0_FCOLOR | WEN_L0_BCOLOR | | | SLB_L0_FCOLOR | SLB_L0_BCOLOR | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 12-60. Soft Lock L0_TRANSP Register**

**Table 12-54. Soft Lock L0_TRANSP Register field descriptions**

| Field | Description |
|---|---|
| WEN_L0_FCOLOR | Write Enable for Soft Lock Bit SLB_L0_FCOLOR<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| WEN_L0_BCOLOR | Write Enable for Soft Lock Bit SLB_L0_BCOLOR<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified |
| SLB_L0_FCOLOR | Soft Lock Bit for L0_FCOLOR Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L0_BCOLOR | Soft Lock Bit for L0_BCOLOR Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |

## 12.3.4.51 Soft Lock L1_TRANSP Register

Figure 12-61 represents the Soft Lock L1_TRANSP register

Offset: 0x31C

Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | WEN_L1_FCOLOR | WEN_L1_BCOLOR | | | SLB_L1_FCOLOR | SLB_L1_BCOLOR | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

**Figure 12-61. Soft Lock L1_TRANSP Register**

**Table 12-55. Soft Lock L0_TRANSP Register field descriptions**

| Field | Description |
|---|---|
| WEN_L1_FCOLOR | Write Enable for Soft Lock Bit SLB_L1_FCOLOR<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified. |
| WEN_L1_BCOLOR | Write Enable for Soft Lock Bit SLB_L1_BCOLOR<br>1'b1: Value is written to SLB<br>1'b0: SLB is not modified. |
| SLB_L1_FCOLOR | Soft Lock Bit for L1_FCOLOR Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1: Associated protected register is locked for write access<br>1'b0: Associated protected register is not locked & writeable |
| SLB_L1_BCOLOR | Soft Lock Bit for L1_BCOLOR Register. This bit cannot be cleared once set by software. Can only be cleared by system reset.<br>1'b1:Associated protected register is locked for write access<br>1'b0:Associated protected register is not locked & writeable |

## 12.4 Functional description

The DCULite is a master on the crossbar switch. It fetches graphic source information directly from memory and dynamically performs blending and bit-blitting operations before delivering data to a TFT LCD panel.

### 12.4.1 Graphic sources

As the DCULite is a master on the crossbar switch, it can access directly any memory or device connected to the crossbar switch as a slave. This includes all on-chip flash, all on-chip RAM, and any slave capable of providing high enough data rates, such as, for example an expanded bus interface or a QuadSPI module. Therefore, any compatible graphic stored anywhere on-chip or in an accessible interface can be displayed on the connected TFT LCD panel with no further intervention from the CPU, except to program the DCULite to fetch and place it. The DCULite also includes a dedicated memory to store the graphic for its cursor layer.

### 12.4.2 TFT LCD panel configuration

The nature and timing of the signals required by TFT LCD panels vary greatly between manufacturers. Therefore, the DCULite allows highly flexible and detailed configuration of these signals.

Timing diagrams for TFT LCD panels are typically divided into a horizontal timing chart and a vertical timing chart. See Figure 12-62 for details.

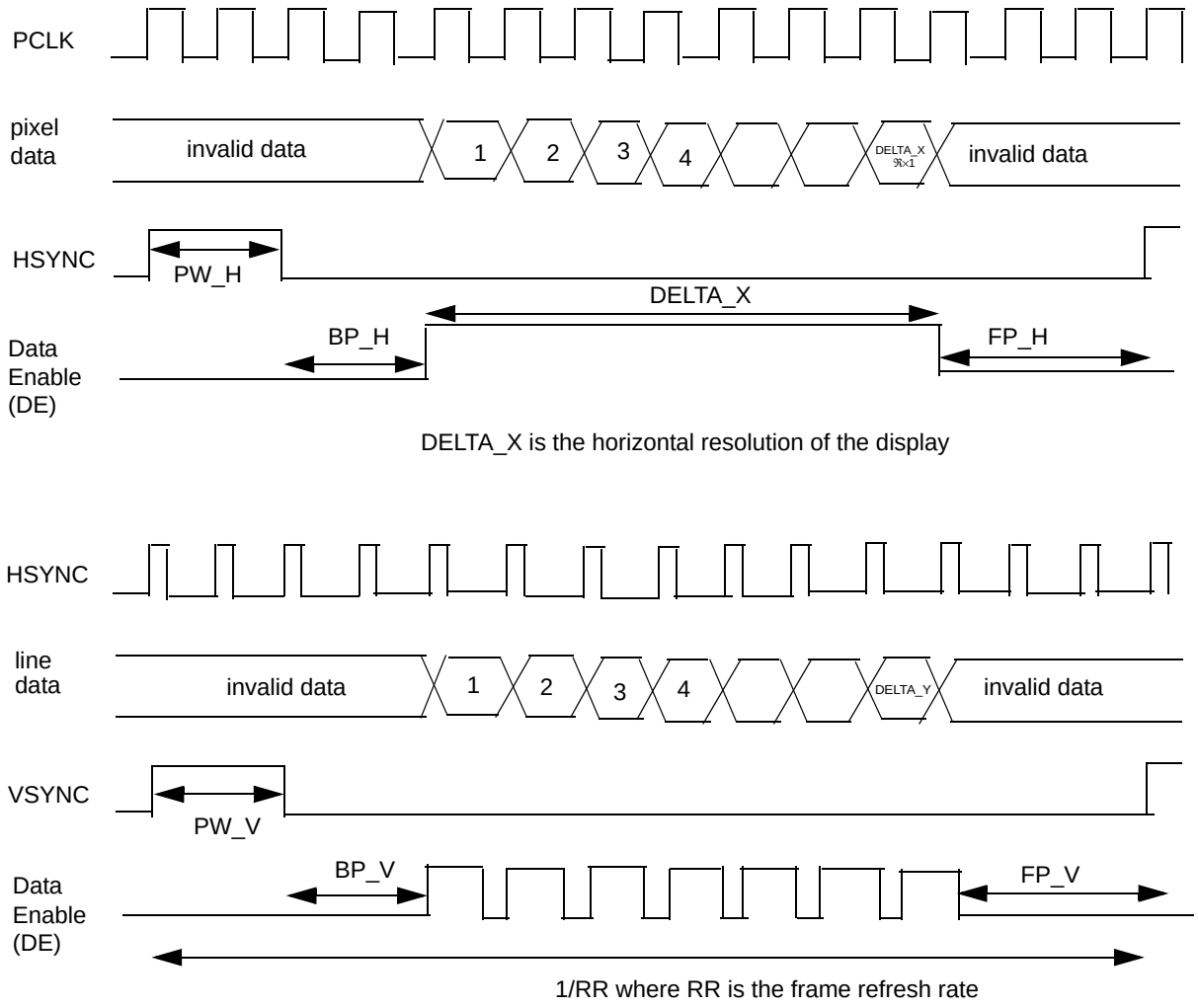



**Figure 12-62. HSYNC and VSYNC timing diagram**

The number of pixel data slots in the horizontal timing diagram is defined by the width of the panel. The number of line data slots is defined by the height of the panel. Both of these values are defined in the DISP_SIZE register (DELTA_X, DELTA_Y). The width of the panel must always be defined as a multiple of 16.

The timing of the pixel clock is defined by the DIV_RATIO register and the frequency of the clock supplied to the DCULite.

In addition to defining the number and timing of pixels in each line and the number of lines, it is normal for TFT LCD panel manufacturers to define other timing signals in terms of pixel clock periods or of the number of horizontal lines. The DCULite also follows this convention.

If the TFT LCD panel requires a horizontal synchronizing signal (HSYNC) and/or a data enable signal, then these can be configured using the fields in the HSYN_PARA register. HSYNC provides a pulse to give the panel notice that the next line of pixel data is about to start, and the data enable signal indicates when that data is present. The PW_H bit field indicates the width of the HSYNC pulse, in pixel data clock

periods. The BP_H bit field defines the delay between the end of the HSYNC pulse and the start of the data enable signal (and pixel data delivery), in pixel clock periods. The FP_H bit field defines the delay between the end of the data enable signal (and pixel data delivery) and the next HSYNC pulse, in pixel clock periods. FP_H and BP_H have minimum values of 1.

If the TFT LCD panel requires a vertical synchronizing signal (VSYNC), then this can be configured using the fields in the VSYN_PARA register. VSYNC provides a pulse to give the panel notice that the next frame of pixel data lines is about to start, and the panel defines delays before and after this pulse, in terms of pixel clock periods. The PW_V bit field indicates the width of the VSYNC pulse in horizontal line periods. The BP_V bit field defines the delay between the end of the VSYNC pulse and the start of the next pixel data (data enable signal), in horizontal line periods. The FP_V bit field defines the delay between the end of the last pixel data (data enable signal) and the next VSYNC pulse, in horizontal line periods. FP_V and BP_V have minimum values of 1.

The polarity of all these signals, including the pixel data itself, may be inverted by using the control bits in the SYN_POL register.

The refresh rate for the panel can be calculated using Equation 12-1 and Equation 12-2 below.

$$RR = \frac{pix_clk}{(DELTA_X + FP_H + PW_H + BP_H) \times (DELTA_Y + FP_V + PW_V + BP_V)} \quad \textit{Eqn. 12-1}$$

where: pix_clk is the pixel clock

DELTA_X is the horizontal resolution (in pixels)

DELTA_Y is the vertical resolution (in pixels)

FP_H is the HSYNC front porch pulse width (in pixel clock cycles)

BP_H is the HSYNC back porch pulse width (in pixel clock cycles)

PW_H is the HSYNC active pulse width (in pixel clock cycles)

FP_V is the VSYNC front porch pulse width (in pixel clock cycles)

BP_V is the VSYNC back porch pulse width (in pixel clock cycles)

PW_V is the VSYNC active pulse width (in pixel clock cycles)

**Pixel Clock = (DCULite Clock) / PRESCALE VALUE** *Eqn. 12-2*

where PRESCALE VALUE is an integer value that can range from 2–32.

### 12.4.3 DCULite mode selection and background color

Once the DCULite is configured for use with a particular TFT LCD panel, it can be enabled for use. There are five modes to choose from, as shown in Table 12-56.

**Table 12-56. List of DCULite operating modes**

| Mode | DCU_MODE[1:0] | PDI_EN | Description |
|---|---|---|---|
| Off | 00 | X | DCULite disabled; the TFT LCD panel is not driven. |
| Color bar | 11 | X | DCULite displays a test pattern consisting of vertical bands of programmable color. |

Table 12-56. List of DCULite operating modes

| Mode | DCU_MODE[1:0] | PDI_EN | Description |
|---|---|---|---|
| Normal | 01 | 0 | DCULite blends layers and displays result on TFT LCD panel. |
| PDI normal | 01 | 1 | As normal mode, except that the panel timing is defined by the input on the PDI interface, and the background color is replaced by the content provided on the PDI interface. |
| PDI slave | 01 | 0 | The DCULite synchronizes its timing to an external signal when PDI_SLAVE_MODE is enabled. |

The DCU_MODE, PDI_EN and PDI_SLAVE_MODE control bits are in the DCU_MODE register. The DCULite has an interface enable bit for the TFT LCD panel interface called RASTER_EN, also in the DCU_MODE register. When RASTER_EN is 0 the raster scanning of pixels to the panel is disabled but the pixel clock continues if enabled on the I/O pin.

Color bar mode is intended for testing the interface between the DCULite and the TFT LCD panel. In this mode, the panel is divided into eight vertical strips of equal width, and the strips display a single color whose RGB value is specified in the COLBAR_1 to COLBAR_8 registers. At reset, the colors are set to black, blue, cyan, green, yellow, red, magenta, and white, where positive logic for the RGB values is assumed. The mode can be used to verify correct connection of the interface to the DCULite and correct timing configuration of the interface. In this mode, any layer configuration settings are ignored.

In Normal mode, the DCULite operates according to the timings specified in Section 12.4.2, “TFT LCD panel configuration” and displays graphics according to the configuration of its layers. The BGND register sets the RGB color of the background shown when no other layers are present. This background color is included in the layer blending process but, since it is always the background, it does not include any layer blending settings.

In PDI normal mode, the DCULite adopts the timing provided on the PDI interface and replaces the background color by the pixel data coming from the PDI interface. The timing values set in the DCULite are ignored in this mode, and the pixel clock and synchronization signals are taken from the PDI interface and passed to the TFT LCD panel. The content of the panel is a combination of the incoming pixel stream and layers generated by the DCULite.

PDI slave mode allows the DCULite to synchronize with the external timing signals on the PDI input.

### 12.4.4 Layer configuration and blending

Users control the graphical content of the TFT panel by manipulating the configuration of elements in the DCULite called layers. Each layer has a control descriptor that defines the size, position, memory encoding, blending, and memory location of the graphic to be displayed. The DCULite provides 16 independent layers that are identical except that they have a fixed priority with respect to each other, and this affects how individual pixels are blended when layers overlap. The blending setting on each layer allows the pixels on that layer to be opaque, partially transparent, or fully transparent, which allows them to combine with pixels on other layers that they overlap.

### 12.4.4.1 Blending priority of layers

The 4 layers available in the DCULite are each fixed in priority order, with layer 0 being the highest priority, layer 1 being the second highest priority, and so on until layer 3, which is the lowest priority. The priority is used by the DCULite to define how to blend individual pixels within the layers. For example, if layer 0 is defined as not being blended with other layers and a pixel on layer 0 overlaps a pixel on layer 1 then the pixel on layer 0 will be visible on the panel unchanged by the pixel on layer 1. However, if layer 0 is defined as being partially transparent, then the DCULite will blend the overlapping pixel such that the result is a combination of the pixel on layer 0 and the pixel on layer 1. It is possible to blend up to four layers at each pixel position.

As there is a maximum number of layers that can blended together, then any pixel on a layer that is lower than the threshold priority will not be included in any blend. If a pixel is on a layer that has the lowest priority in any blending scheme, then the blending settings for that pixel are ignored and the pixel is treated as a background pixel. This means that a lower priority layer may have some pixels completely obscured by those on higher priority layers on one part of the panel, and some other pixels visible or blended on other parts of the panel.

Figure 12-63 shows how the pixel blend takes place inside the DCULite. The priority of the layers determines at which stage of the blend the pixel enters. Any pixels lower than the threshold priority are ignored and, as can be seen, the blend settings for the lowest priority pixel is also ignored.

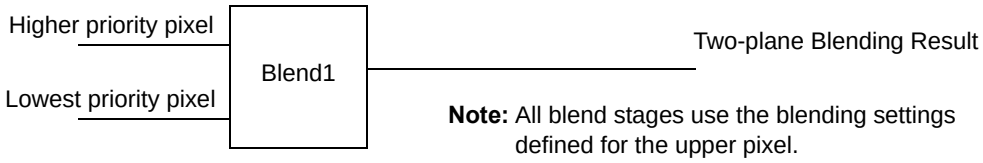



**Figure 12-63. Pixel blending stack**

This priority concept is illustrated in Figure 12-64 and Figure 12-65. In this case, there are five layers enabled, and each contains a graphic that is a solid rectangular block of a single color. The size and shape of each layer is different. The background color of the panel is set to grey and layers have been placed such that they overlap each other.

Figure 12-64 shows the individual source graphics and the case where no layer has any blending enabled. Here, the highest priority layer (in this case layer 0) is fully visible. Layer 1 is visible where layer 0 does not overlap it. Layer 2 is visible where layer 1 does not overlap it. Layer 3 is overlapped by layers 0 and 1 and so is only partially visible. Layer 4 is partially obscured by all of the other layers. Note that layer 4 is higher priority than the background color.

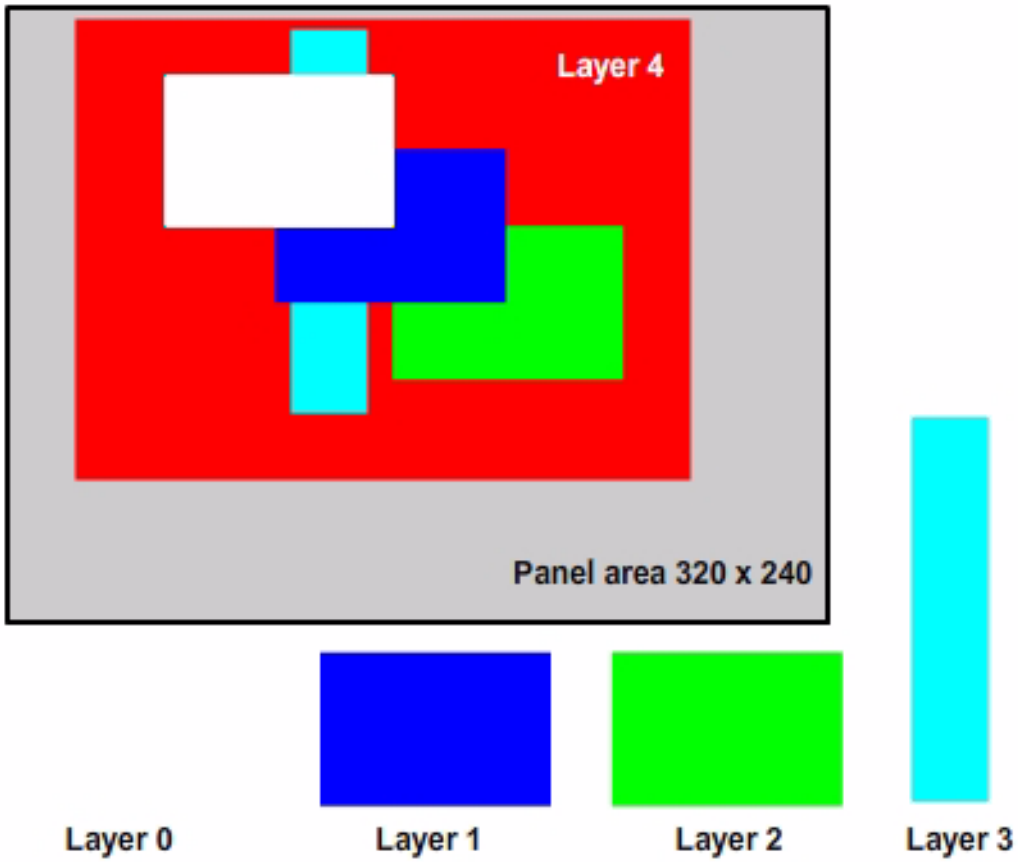



**Figure 12-64. Example of layer placement with no blending**

Figure 12-65 shows the same layer configuration except that, in this case, layer 0 and layer 3 have been made 50% transparent. The pixels in layer 0 are blended with those on the layers below including layer 3. However, notice that in the area where layer 0 overlaps layer 3 there is no further blending with the underlying layer 4 or the background. This is because the DCULite can only blend two layers.

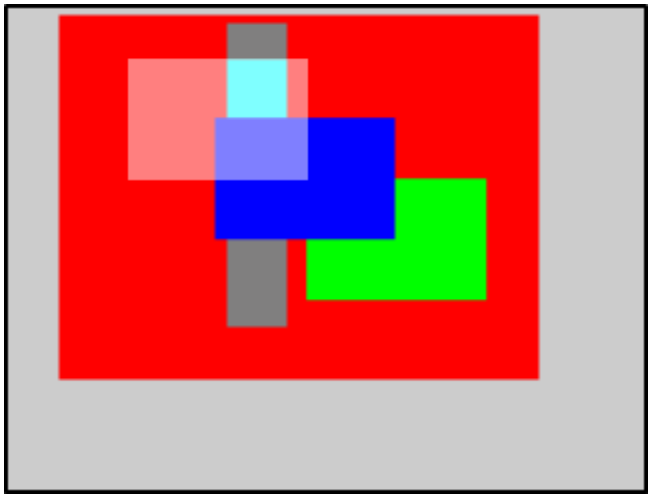


**Figure 12-65. Example of layer placement with 3-layer blending**

All blending is performed using full 8-bits-per-component colors. The DCULite automatically performs a color promotion on source data that is stored in less than RGB888 color.

#### 12.4.4.2 Control Descriptors

The control descriptor for each layer consists of seven registers, and all 4 control descriptors are identical except the two highest priority layers, which have additional control bits for the safety mode.

The control descriptors may be written to at any time, and the value present in the registers at the start of the next frame refresh cycle defines the content of the panel for that frame. To avoid coherency issues, ensure all control descriptor changes are made before the PROG_END bit in the INT_STATUS register is asserted.

#### 12.4.4.3 Layer size and positioning

The size of each layer is defined by register 1 in the control descriptor for the layer (CTRLDESCLn_1, where n is the layer number). The register contains two bit fields, HEIGHT and WIDTH, which determine the size and shape of the layer. Both fields are expressed in terms of the number of pixels in each dimension.

The HEIGHT bit field may take any value; however, it may not be useful to define a value larger than the height of the panel.

The WIDTH field has a restriction on the value it can take, depending on the data format of the graphic specified by the layer. This field must always be an integer multiple of the number of pixels that are represented by a 32-bit word except in the special case of 1 bit per pixel where the multiple is 16. The data format can range from 1 bit per pixel to 32 bits per pixel and so there is a range of multiples from 1 to 32. Figure 12-57 shows the multiples for the WIDTH bit field and some correct values.

**Table 12-57. Example of WIDTH multiples for different graphic data formats**

| Data format | WIDTH multiples | Example values |
|---|---|---|
| 1 bpp | 16 | 16, 32, 48, 64, … |
| 2 bpp | 16 | 16, 32, 48, 64, … |
| 4 bpp | 8 | 8, 16, 24, 32, … |
| 8 bpp | 4 | 4, 8, 12, 16, … |
| 16 bpp | 2 | 2, 4, 6, 8, … |
| 24 bpp | 4 (= 3 whole 32-bit words) | 4, 8, 12, 16 |
| 32 bpp | 1 | 1, 2, 3, 4, … |
| YCbCr422 | 4 | 4, 8, 12, 16, … |

If the WIDTH bit field is set to an invalid multiple, then the layer configuration is invalid, the layer cannot be made visible, and an error flag is set in the layer parameter error register (PARR_ERR).

The position of each layer on the panel is defined by register 2 in the control descriptor for the layer (CTRLDESCLn_2, where n is the layer number). The register contains two bit fields, POSY and POSX,

which determine the location of the upper left pixel of the layer in the x and y axes. Both fields are expressed in terms of the number of pixels in each axis.

There are no restrictions on layer placement. Any layer can be placed and moved to any panel position. If a layer is placed so that pixels would appear beyond the dimensions of the panel, then the DCULite displays the pixels on the panel and ignores the pixels off the panel.

#### 12.4.4.4 Graphics and data format

The memory location of the graphic that is displayed on the layer is defined by register 3 in the control descriptor for the layer (CTRLDESCLn_3, where n is the layer number). This 32-bit value can contain the address of any 64-bit aligned memory location in the memory map of the MCU.

The format of the data that describes the graphic is defined by the BPP bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). This value also influences the range of values for the width of the layer (see Section 12.4.4.3, “Layer size and positioning”). By choosing an appropriate format, it is possible to optimize the memory required by the graphics in use.

There are five formats where the RGB values of the pixels are stored directly in the graphic. In these formats, the DCULite treats the data as describing a true RGB color. The formats are:

- BGRA8888, where the data defines 8-bit values for the red, green, blue, and alpha components of the image. This blends as ARGB, however, in this format the order of the bytes is reversed compared to other formats.
- RGB888, where the data defines 8-bit values for the red, green, and blue components of the image.
- RGB565 where the data defines 5-bit values for the red and blue components, and 6-bit values for the green component of the image.
- ARGB1555 where the data defines 5-bit values for the red, green, and blue components, and a 1-bit value for the alpha channel of the image.
- ARGB4444, where the data defines 4-bit values for the red, green, blue, and alpha components of the image.

The three 16-bit formats (RGB565, ARGB1555, and ARGB4444) are promoted to full 8 bit per component format by shifting the bits left so that the MSB of the component in the 16-bit format becomes the MSB of the 24/32 bpp (bit per pixel) format, and the LSB is filled with the value of the MSBs. For example, an RGB565 value of 10000:010000:11011 becomes 10000100:01000001:11011110. An RGB4444 value of 1010:0011:1100:0101 becomes 10101010:00110011:11001100:01010101. An RGB1555 value of 1:10100:01000:11011 becomes 11111111:10100101:01000010:11011110.

There are five indexed color formats (1/2/4/8 bpp & APAL8) where the data in the graphic does not define the RGB color to display. Instead, the data defines the entry in a color look-up table (CLUT) that contains a palette of ARGB colors. The maximum number of colors in the CLUT is defined by the size of the data stored in the graphic. For 1 bpp graphics, there is a maximum of two colors in the CLUT. For 2 bpp, there is a maximum of four colors. For 4 bpp and 8 bpp data, the maximums are 16 and 256 colors, respectively. In APAL8 mode(16 bpp), the upper 8 bits define the alpha component of the pixel and the lower 8 bits define the offset in the CLUT (the alpha component in the CLUT color is ignored).

The address of the first value in the CLUT is defined in the LUOFFS bit field of register 4 and the CLUT is the RAM block dedicated to the DCULite which is described in Section 12.4.6, “CLUT RAM”. Since the ARGB values stored in the CLUT are 32-bit ARGB, there is no need for further adjustment before blending.

The DCULite also supports graphics encoded using luminance and chrominance format. This format is generically known as YUV and stores the luminance (brightness, Y) of a pixel separate from its chrominance (color information, U and V). This format is widely used by cameras and is supported by the PDI for direct video in as well as the DCULite when stored in memory for display on a layer. The specific implementation used by the DCULite is more accurately described as YCbCr422 which uses twice as many bits to describe the luminance as to describe the blue (Cb) and red (Cr) difference of the chrominance.

The DCULite takes these pixels and converts them to RGB format using equations configured using its LYR_LUMA_COMP, LYR_CHROMA_RED, LYR_CHROMA_GREEN, and LYR_CHROMA_BLUE registers. The YCbCr format specifies a common chroma setting for two pixels; however, it is possible to interpolate the chroma for the pixels rather than setting both to the same value. Due to the additional conversion step required, the DCULite is able to blend a maximum of one layer encoded in YCbCr for each pixel. This feature is enabled by the LYR_INTPOL_EN[EN] field (see Section 12.8.1.6.3, “YCbCr mode”).

There are four additional formats defined by the BPP bit field. These configure the graphic in transparency mode and luminance mode (see Section 12.4.4.6, “Transparency mode and blending” and Section 12.4.4.7, “Luminance mode” respectively).

There is a set storage format for each data format provided by the DCULite. These formats can be seen in Table 12-58 to Table 12-67.

**Table 12-58. Data layout for BGRA8888**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | B0 | G0 | R0 | A0 | B1 | G1 | R1 | A1 |
| 0x08 | B2 | G2 | R2 | A2 | B3 | G3 | R3 | A3 |

**Table 12-59. Data layout for YCbCr422 format[1]**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | Cb0 | Y0 | Cr0 | Y1 | Cb2 | Y2 | Cr2 | Y3 |
| 0x08 | Cb4 | Y4 | Cr4 | Y5 | Cb6 | Y6 | Cr6 | Y7 |

1 The YCbCr422 format encodes chroma information across two pixels. Therefore, the chroma values apply to the even pixel denoted in the table and its adjacent odd pixel.

**Table 12-60. Data layout for 24 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | B0 | G0 | R0 | B1 | G1 | R1 | B2 | G2 |
| 0x08 | R2 | B3 | G3 | R3 | B4 | G4 | R4 | B5 |

For 16 bpp, data expected is in the form of RGB565, ARGB1555, ARGB4444, or APAL8.

**Table 12-61. Generic data layout for 16 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | pixel0[15:8] | pixel0[7:0] | pixel1[15:8] | pixel1[7:0] | pixel2[15:8] | pixel2[7:0] | pixel3[15:8] | pixel3[7:0] |
| 0x08 | pixel4[15:8] | pixel4[7:0] | pixel5[15:8] | pixel5[7:0] | pixel6[15:8] | pixel6[7:0] | pixel7[15:8] | pixel7[7:0] |

**Table 12-62. Data layout for ARGB1555 format**

| Address offset | [0] | [1:5] | [6:10] | [11:15] | [16] | [17:21] | [22:26] | [27:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | A0 | R0 | G0 | B0 | A1 | R1 | G1 | B1 |
| 0x04 | A2 | R2 | G2 | B2 | A3 | R3 | G3 | B3 |

**Table 12-63. Data layout for APAL8 format**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | A0 | Offset 0 | A1 | Offset 1 | A2 | Offset 2 | A3 | Offset 3 |
| 0x08 | A4 | Offset 4 | A5 | Offset 5 | A6 | Offset 6 | A7 | Offset 7 |

**Table 12-64. Data layout for 8 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | pixel0[0:7] | pixel1[0:7] | pixel2[0:7] | pixel3[0:7] | pixel4[0:7] | pixel5[0:7] | pixel6[0:7] | pixel7[0:7] |
| 0x08 | pixel8[0:7] | pixel9[0:7] | pixel10[0:7] | pixel11[0:7] | pixel12[0:7] | pixel13[0:7] | pixel14[0:7] | pixel15[0:7] |

**Table 12-65. Data layout for 4 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | pixel1-pixel0 | pixel3-pixel2 | pixel5-pixel4 | pixel7-pixel6 | pixel9-pixel8 | pixel11-pixel10 | pixel13-pixel12 | pixel15-pixel14 |
| 0x08 | pixel17-pixel16 | pixel19-pixel18 | pixel21-pixel20 | pixel23-pixel22 | pixel25-pixel24 | pixel27-pixel26 | pixel29-pixel28 | pixel31-pixel30 |

**Table 12-66. Data layout for 2 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | pixel3-pixel0 | pixel7-pixel4 | pixel11-pixel8 | pixel15-pixel12 | pixel19-pixel16 | pixel23-pixel20 | pixel27-pixel24 | pixel31-pixel28 |
| 0x08 | pixel35-pixel32 | pixel39-pixel36 | pixel43-pixel40 | pixel47-pixel44 | pixel51-pixel48 | pixel55-pixel52 | pixel59-pixel56 | pixel63-pixel60 |

**Table 12-67. Data layout for 1 bpp**

| Address offset | [0:7] | [8:15] | [16:23] | [24:31] | [0:7] | [8:15] | [16:23] | [24:31] |
|---|---|---|---|---|---|---|---|---|
| 0x00 | pixel7-pixel0 | pixel15-pixel8 | pixel23-pixel16 | pixel31-pixel24 | pixel39-pixel32 | pixel47-pixel40 | pixel55-pixel48 | pixel63-pixel56 |
| 0x08 | pixel71-pixel64 | pixel79-pixel72 | pixel87-pixel80 | pixel95-pixel88 | pixel103-pixel96 | pixel11-pixel104 | pixel119-pixel112 | pixel127-pixel120 |

The DCULite includes a flag that indicates when it has completed fetching graphics from memory for the current frame refresh. If required, this flag (DMA_TRANS_FINISH in the INT_STATUS register) can be used to determine when changes can be made to the source graphic content.

### 12.4.4.5 Alpha and Chroma-key blending

The blending configuration of each layer is defined by the BB, AB, and TRANS bit fields in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). The pixels affected by the blending configuration can be further selected by registers 5 and 6 in the control descriptor (CTRLDESCLn_4 and CTRLDESCLn_5). Depending on the priority and placement of the layer (see Section 12.4.4.1, “Blending priority of layers”), these bit fields and registers define how pixels from different layers are blended together.

The AB and BB bit fields define whether blending is active and whether the whole graphic or a selected portion is blended. Registers 5 and 6 define the range of RGB colors that define the selected pixels. The TRANS bit field defines the transparency of the selected pixels.

The BB bit field defines whether the whole graphic, or only certain pixels, should be blended. When this bit is set, pixels that have an RGB value that falls into the range defined by registers 5 and 6 are considered to be selected and treated differently to the non-selected pixels in the graphic. This is a process known as chroma-keying since it is the color of the pixel that defines the selection. The selected pixels must be within the range defined by each color component of registers 5 and 6. See Table 12-68 for examples of pixels that are selected and not selected when the given range is defined as 0x0080C0 to 0x0FB0FF.

**Table 12-68. Example of how chroma-key range selects pixels**

| Source pixel | Red 00–0F | Green 80–B0 | Blue C0–FF | Comment |
|---|---|---|---|---|
| 0x000000 | P | X | X | Not selected |

**Table 12-68. Example of how chroma-key range selects pixels**

| Source pixel | Red 00–0F | Green 80–B0 | Blue C0–FF | Comment |
|---|---|---|---|---|
| 0x08C0C0 | P | X | P | Not selected |
| 0x08A0C0 | P | P | P | Pixel is selected |

The AB bit field defines how any selected and non-selected pixels are blended. By combining this control with the BB bit field it is possible to define 11 unique ways of blending the pixels on a layer dependent on the type of layer. Depending on the configuration defined by the AB and BB bit fields, the TRANS bit field combines the two pixels in every blend stage using the alpha value of the upper pixel (which has the effect of making this pixel more or less transparent and revealing more or less of the lower pixel).

The result of each blend stage is calculated for all three color components as shown in Equation 12-3.

$$A = (BGPixel*(255 - alpha)) + (FGPixel * alpha) \qquad \textit{Eqn. 12-3}$$

The result of the calculation must then be divided by 255 to normalize the result. This calculation is performed as follows:

```
//First Division
output_val = A + (A >> 8)
//Rounding off first addition & division
if (((A>>7) & 0x1) == 0x1)
output_val ++
//Second Division with rounding
output_val = output_val >>7;
if ((output_val & 0x1) == 0x1))
output_val = output_val + 0x2;
output_val = output_val >> 1;
```

The blend can apply to pixels with no alpha channel (RGB) or with an alpha channel (ARGB) in different ways.

Table 12-69 defines how the settings of the BB and AB bit fields affect the pixels in the layer; RGB formats are RGB565, and RGB888; ARGB formats are 1 bpp, 2 bpp, 4 bpp, 8 bpp, APAL8, ARGB1555, ARGB4444, and BGRA8888. Pixels in YCbCr format are treated as RGB pixels for the purposes of the blend.

**Table 12-69. Blend options for BB and AB configurations**

| Case | BB | AB | Format | Function |
|---|---|---|---|---|
| 1 | 0 | 00 | RGB | No blending, underlying pixels are obscured |
| 2 | 1 | 00 | RGB | Selected pixels are completely removed |

**Table 12-69. Blend options for BB and AB configurations (continued)**

| Case | BB | AB | Format | Function |
|---|---|---|---|---|
| 3 | 0 | 01 | RGB | The value in TRANS becomes the alpha channel of all pixels on the layer |
| 4 | 1 | 01 | RGB | The value in TRANS becomes the alpha channel of the selected pixels on the layer |
| 5 | 0 | 10 | RGB | Same as case 3 |
| 6 | 1 | 10 | RGB | Selected pixels are completely removed and the value in TRANS becomes the alpha channel of the non-selected pixels on the layer |
| 7 | 0 | 11 | RGB | Reserved |
| 8 | 1 | 11 | RGB | Reserved |
| 9 | 0 | 00 | ARGB | No blending, pixel alpha is ignored and underlying pixels are obscured |
| 10 | 1 | 00 | ARGB | Selected pixels are completely removed, pixel alpha is ignored |
| 11 | 0 | 01 | ARGB | Pixel alpha is used to blend layer with underlying pixels. Value in TRANS is ignored. |
| 12 | 1 | 01 | ARGB | Uses the pixel alpha of the selected pixels only to blend layer with underlying pixels. Value in TRANS is ignored. |
| 13 | 0 | 10 | ARGB | The value in TRANS is multiplied with the pixel alpha value and the resultant alpha is used to blend all the pixels |
| 14 | 1 | 10 | ARGB | Selected pixels are completely removed, the value in TRANS is multiplied with the pixel alpha value and the resultant alpha is used to blend the non-selected pixels on the layer |
| 15 | 0 | 11 | ARGB | Reserved |
| 16 | 1 | 11 | ARGB | Reserved |

Figure 12-66 to Figure 12-74 illustrate the effect of the cases identified in Table 12-69. In all cases there is a single active layer and a white background color.

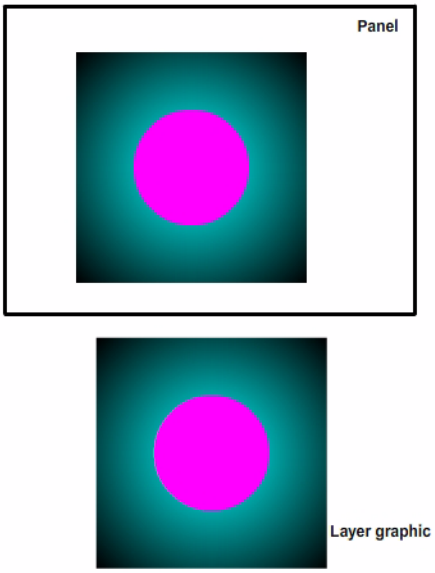



**Figure 12-66. Case 1 example (no blend)**

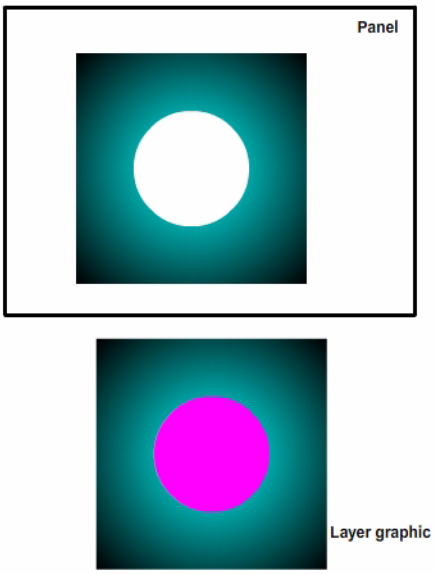



**Figure 12-67. Case 2 example (remove selected pixels)**

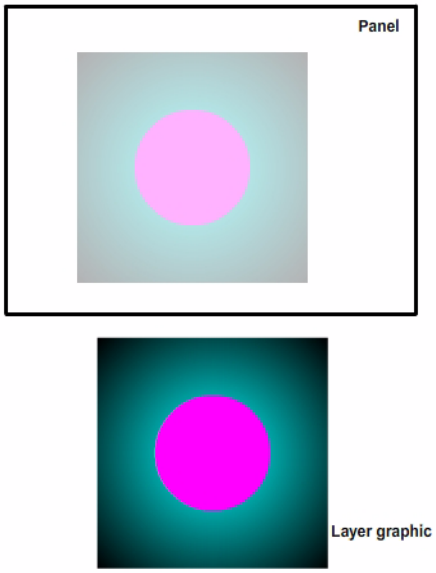



**Figure 12-68. Case 3 example (all pixels transparent)**

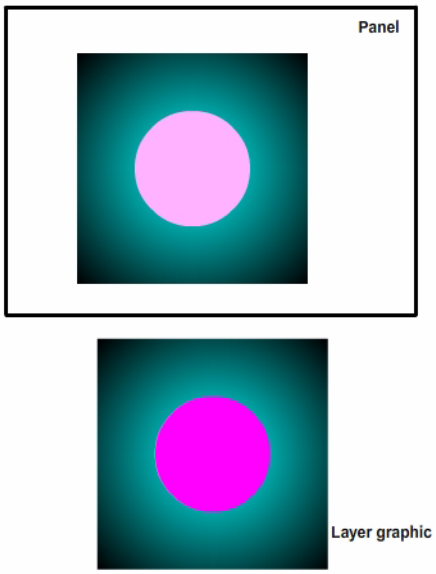



**Figure 12-69. Case 4 example (selected pixels transparent)**

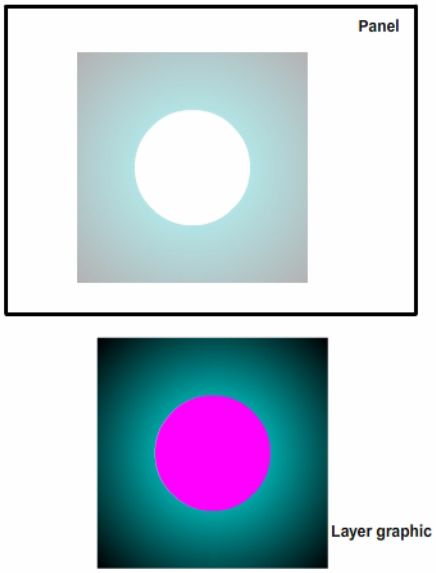



**Figure 12-70. Case 6 example (selected pixels removed, others transparent)**

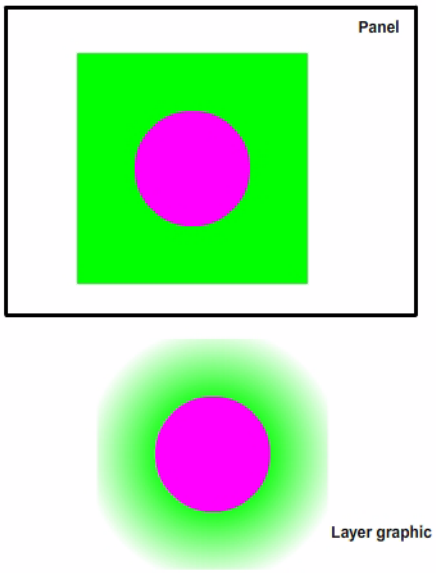



**Figure 12-71. Case 9 example (no blend, pixel alpha ignored)**

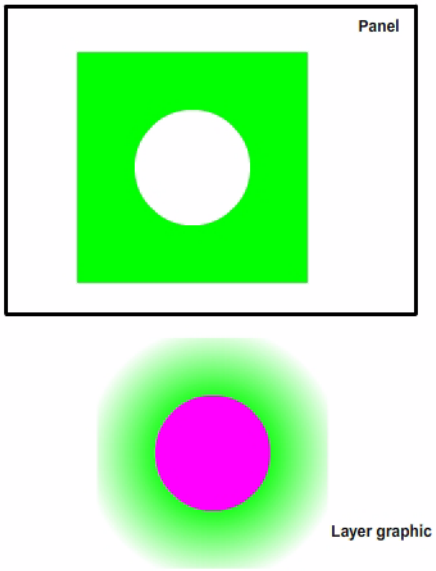



**Figure 12-72. Case 10 example (selected pixels removed, pixel alpha ignored)**

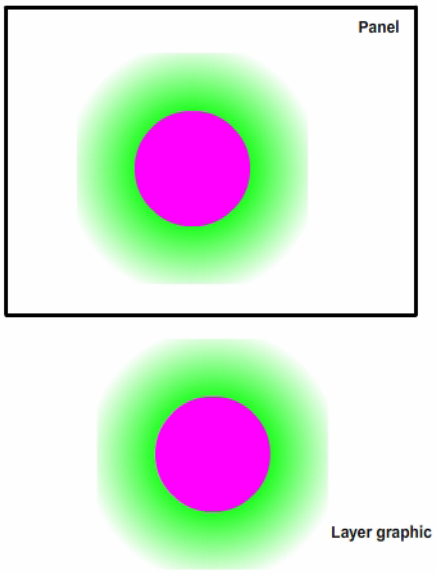



**Figure 12-73. Case 13 example (pixel and layer alpha used in blend)**

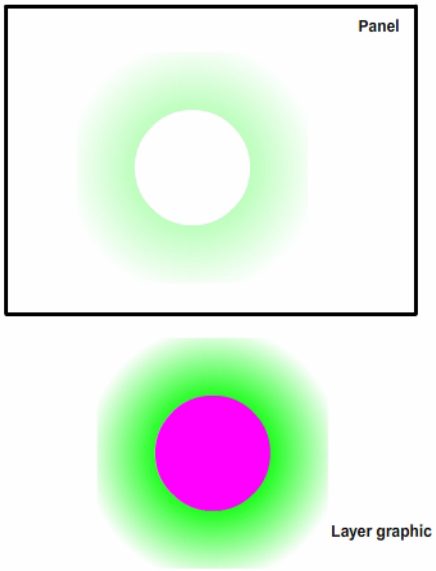



**Figure 12-74. Case 14 example (selected pixels removed, pixel and layer alpha used in blend)**

### 12.4.4.6 Transparency mode and blending

Transparency mode is a special case for the graphic data format and is defined by the BPP bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). This value also influences the range of values for the width of the layer (see Section 12.4.4.3, “Layer size and positioning”). By choosing an appropriate format, it is possible to optimize the memory required by the graphics in use.

In transparency mode, the source graphic does not contain any direct or indexed color information. Instead, the graphic data represents the alpha channel of the graphic. The DCULite creates the final graphic by pre-blending a foreground color and background color using the alpha value of each pixel. The result of this pre-blend can then be blended with pixels on other layers using the normal blending process. Each layer has dedicated registers to contain the foreground and background colors for this mode. These are FGn_fcolor and FGn_bcolor, where n is the layer number. See Figure 12-75.

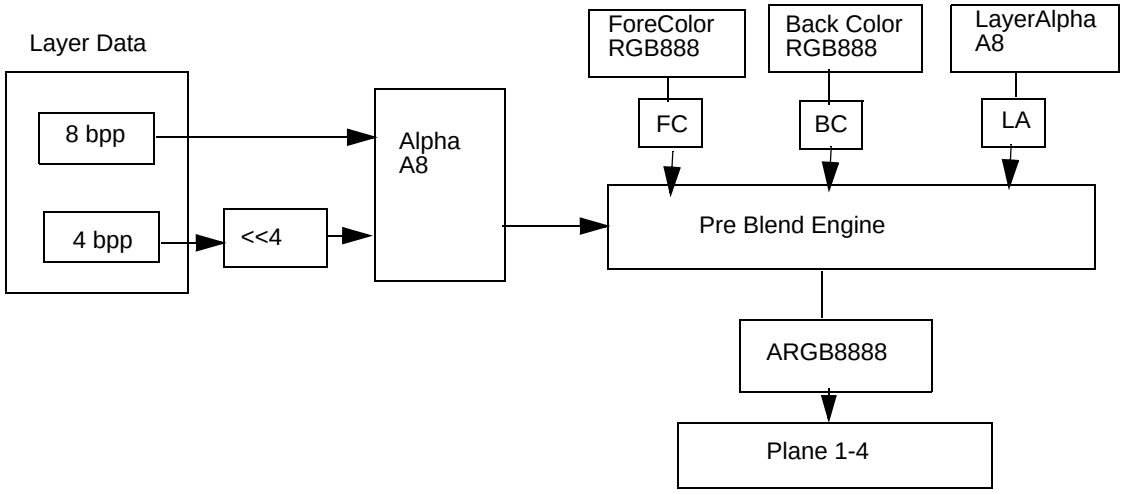



**Figure 12-75. Transparency mode description**

Transparency mode is typically used when a graphic must blend smoothly into the underlying layers, but where a rich color palette is not required. Examples include text where this mode allows the text to blend smoothly with any background — this is known as anti-aliasing.

There are two transparency modes available: 4 bpp and 8 bpp. The result of the pre-blend can be treated as an RGB888 graphic and blended in a similar way to previously described, or it can be treated as a special case of ARGB with only the foreground color visible in the final blend. Table 12-70 describes the blend options for transparency mode.

**Table 12-70. Blend options for transparency mode**

| Case | BB | AB[1:0] | Mode | Function |
|---|---|---|---|---|
| 1 | 0 | 00 | Transparency | No blending, underlying pixels are obscured |
| 2 | 1 | 00 | Transparency | Reserved |
| 3 | 0 | 01 | Transparency | The value in TRANS becomes the alpha channel of all pixels on the layer |
| 4 | 1 | 01 | Transparency | The value in TRANS becomes the alpha channel of the selected pixels on the layer |
| 5 | 0 | 10 | Transparency | Same as case 3 |
| 6 | 1 | 10 | Transparency | Background color is ignored, selected pixels are completely removed, the value in TRANS is multiplied with the graphic data value (alpha) and the resultant alpha is used to blend the non-selected pixels on the layer |
| 7 | 0 | 11 | Transparency | Reserved |
| 8 | 1 | 11 | Transparency | Reserved |

Figure 12-76–Figure 12-79 illustrate the effect of the cases identified in Table 12-70. In all cases there is a single active transparency layer and a white background color.

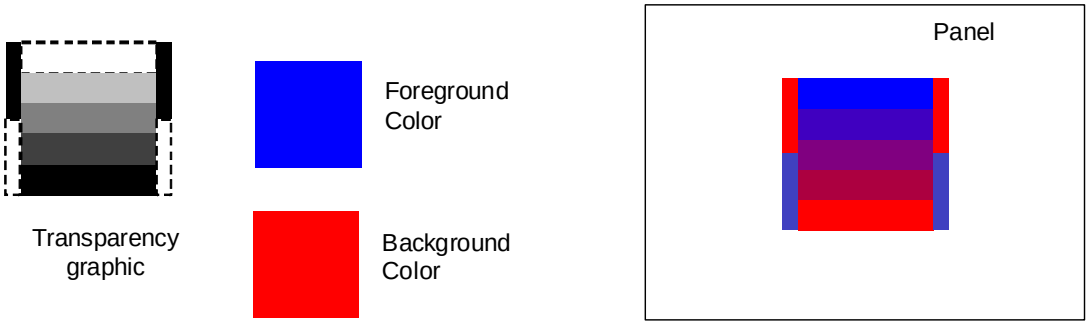



**Figure 12-76. Case 1 example (no blend)**

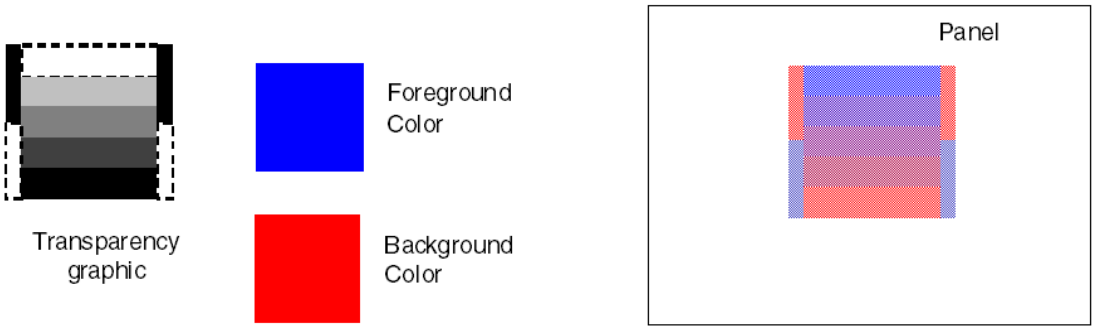



**Figure 12-77. Case 3 example (all pixels transparent)**

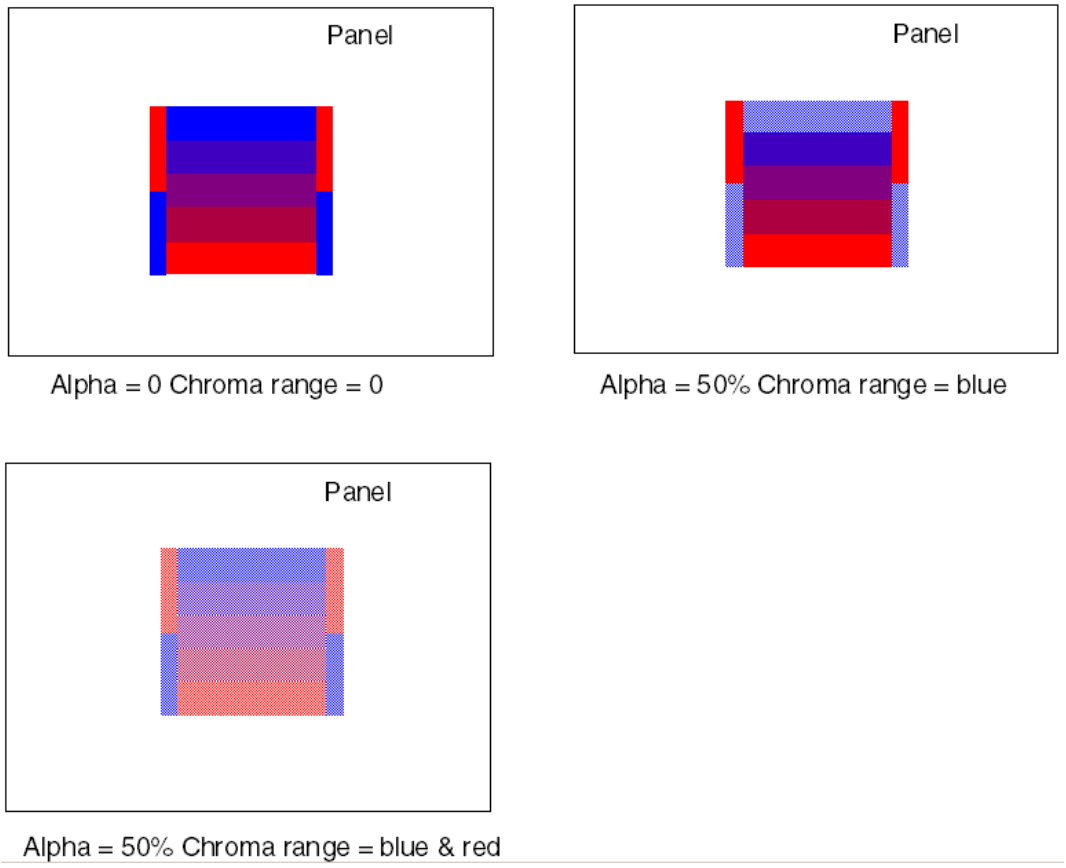



**Figure 12-78. Case 4 example (selected pixels transparent)**

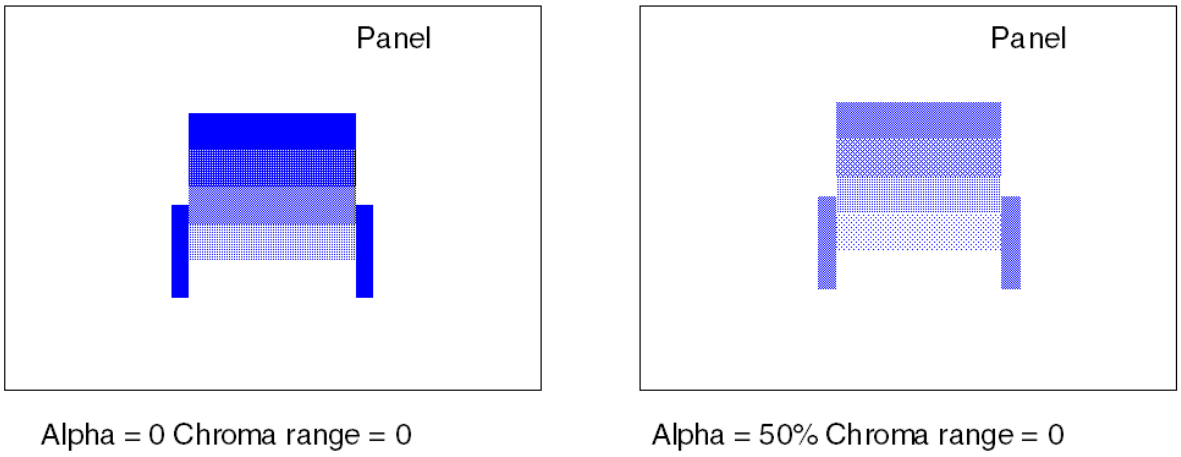



**Figure 12-79. Case 6 example (only foreground color blended)**

### 12.4.4.7 Luminance mode

Luminance mode is a special case for the graphic data format and is defined by the BPP bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number). This value also influences the range of values for the width of the layer (see Section 12.4.4.3, “Layer size and

positioning”). By choosing an appropriate format, it is possible to optimize the memory required by the graphics in use.

In luminance mode, the data in the source graphic does not contain any direct or indexed color information or alpha information. The data values in a layer in luminance mode modify the values of the pixels on underlying layers only. There are two luminance modes available: 4 bpp and 8 bpp. In both cases, the data values behave as signed integers that are added to each component of the underlying pixel. The 4 bpp mode is left-shifted to form a signed 8 bpp integer. The results of the addition are prevented from overflowing, so that any result greater than 0xFF is set to 0xFF and any result less than 0x00 is set to 0x00.

The result of a blend with a luminance layer is that the intensity of the underlying pixel's color will be increased or decreased. In this way, luminance mode can be used to highlight or dim pixels on the panel without having to modify the source graphic data. Table 12-71 describes the effect of luminance blends on an underlying pixel.

**Table 12-71. Example of a blend with a luminance mode layer**

| Pixel value | Luminance value | Resultant pixel |
|---|---|---|
| 0xFF8040 | 0x40 | 0xFFC080 |
| 0xFF8040 | 0xC0 | 0x3F0000 |

### 12.4.4.8 Tile mode

Tile mode is a special case for the layer and is enabled by the TILE_EN bit field in register 4 in the control descriptor for the layer (CTRLDESCLn_4, where n is the layer number).

In this mode the layer size register (CTRLDESCLn_1, where n is the layer number) defines the size of the layer; however, the size of the graphic is defined in control register 7 (CTRLDESCLn_7, where n is the layer number). The size of the graphic must be less than or equal to the size of the layer. When tile mode is enabled, the graphic is repeated horizontally and vertically until it fills the whole layer. The horizontal size of the tile is defined by the TILE_HOR_SIZE bit field and is restricted to be a multiple of 16 pixels. The vertical size of the tile is defined by the TILE_VER_SIZE bit field. See Figure 12-80 for an example of a layer in tile mode.

The graphic data for the Tile Mode is fetched from the MCU memory according to the address in Control Descriptor 3 and can be in any previously-described data format.

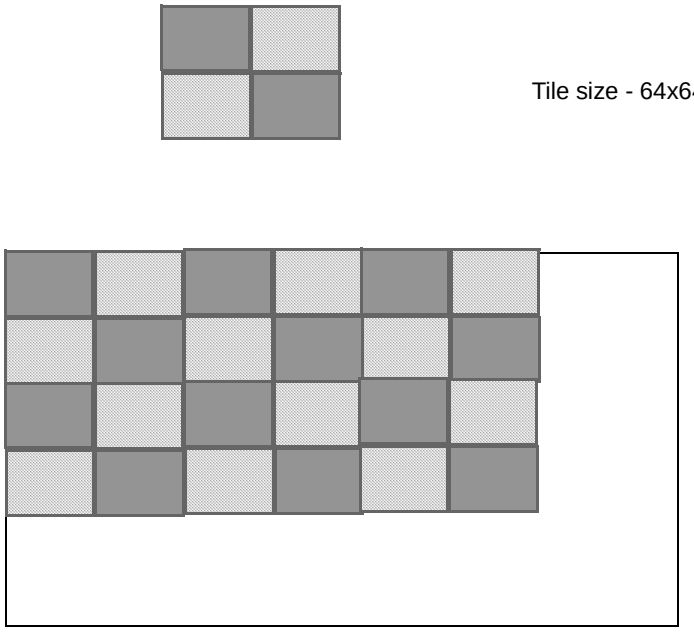



**Figure 12-80. Tile Mode**

## 12.4.5 Hardware cursor

In addition to the 4 layers, the DCULite also provides a special layer intended for use as a cursor. This cursor operates in 1 bpp mode and includes its own RAM area to store the graphic. The cursor may be placed at any location on the panel and includes an automatic blink option. The hardware cursor is configured using a dedicated control descriptor.

The size of the cursor is defined by register 1 in the control descriptor for the cursor (CTRLDESCCURSOR_1). The register contains two bit fields, HEIGHT and WIDTH, which determine the size and shape of the layer. Both fields are expressed in terms of the number of pixels in each dimension. The HEIGHT is limited to a maximum of 256 pixels, and the total number of pixels cannot exceed the number of bits in the cursor RAM (8192 bits).

Bits in the cursor RAM that are 0 become transparent on the panel. Bits that are 1 become fully opaque in the color defined in register 3 in the control descriptor for the cursor (CTRLDESCCURSOR_3). The DEFAULT_CURSOR_COLOR bit field is in RGB888 format.

There are restrictions on the arrangement of bits in the cursor RAM depending on how the HEIGHT and WIDTH bit fields are configured.

- The rightmost bit in the cursor RAM (bit 31) represents the leftmost pixel on the display.
- When the cursor size is less than 32 bits, each row of the cursor is contained in a single 32-bit word of cursor RAM. The other bits in each row must be filled with zeros.

- When the cursor width is an integer multiple of 32 bits, the pixels in each row roll from one word in the RAM to the next one. The rightmost bit in the first word in the RAM is the top leftmost pixel on the display. The leftmost bit in the word represents a pixel that is adjacent to the rightmost bit in the next word (in the same row). The leftmost pixel on the next row is the rightmost bit in the first word after n words that describe the first row.
- When the cursor is greater than 32 bits but not an integer multiple of 32, the pixels in each row roll from one word into the next one such that the rightmost bit in the first word of the row is the leftmost bit on the display. In the final word of the row there are unused bits.

The position of the cursor on the panel is defined by register 2 in the control descriptor for the cursor (CTRLDESCCURSOR _2). The register contains two bit fields, POSY and POSX, which determine the location of the upper left pixel of the cursor in the x and y axes. Both fields are expressed in terms of the number of pixels in each axis. Placing the cursor beyond the panel area is not allowed.

The cursor can be configured to blink at a particular rate when it is enabled. The EN_BLINK, HWC_BLINK_ON, and HWC_BLINK_OFF bit fields define the blink behavior. These are in register 4 in the control descriptor for the cursor (CTRLDESCCURSOR_4). EN_BLINK enables blinking. The blinking time is based on the frame rate, and the on and off times are independently configurable. HWC_BLINK_ON configures the number of frame refresh cycles for which the cursor is visible. HWC_BLINK_OFF configures the number of frame refresh cycles for which the cursor is not visible. For a frame refresh rate of 64 Hz, the HWC_BLINK_ON and HWC_BLINK_OFF counters give a range of on/off times up to 4 seconds.

The cursor is enabled by setting the CUR_EN bit field in register 3 in the control descriptor for the cursor (CTRLDESCCURSOR_3).

If the DCULite detects an invalid configuration in the cursor control descriptor, then the cursor configuration is invalid and it cannot be made visible. In addition, the error flag HWC_ERR is set in the layer parameter error register (PARR_ERR).

The cursor RAM may be written at any time when the TFT LCD panel is not being driven with data. This means that the RAM can be modified when the DCULite is not enabled and during the vertical blanking period.

### 12.4.6 CLUT RAM

The color look up table (CLUT) RAM is a dedicated memory in the DCULite used to store palettes for indexed colors. Color information in this RAM is always stored as aligned 32-bit words where the most-significant byte is always the alpha component, the next byte contains the red component, the next the green component and the least significant byte the blue component (0xAARRGGBB).

The content of the RAM at a specific address is defined by the control descriptor of a layer. The LUOFFS bit field in the layer control descriptor defines the starting address of the area, and the BPP bit field defines the maximum size of the palette for that layer.

In Figure 12-81 three areas of the RAM are defined for different purposes. Area A is used by layer 1 as a CLUT for its 4 bpp graphic. Area B is use by layer 5 as a store for its APAL8 graphic. Area C is used by

layers 2, 7, and 9 as a CLUT for their 8 bpp graphics. In this example, the APAL8 palette in area B uses a reduced palette size of 64 entries instead of its maximum of 256 entries.

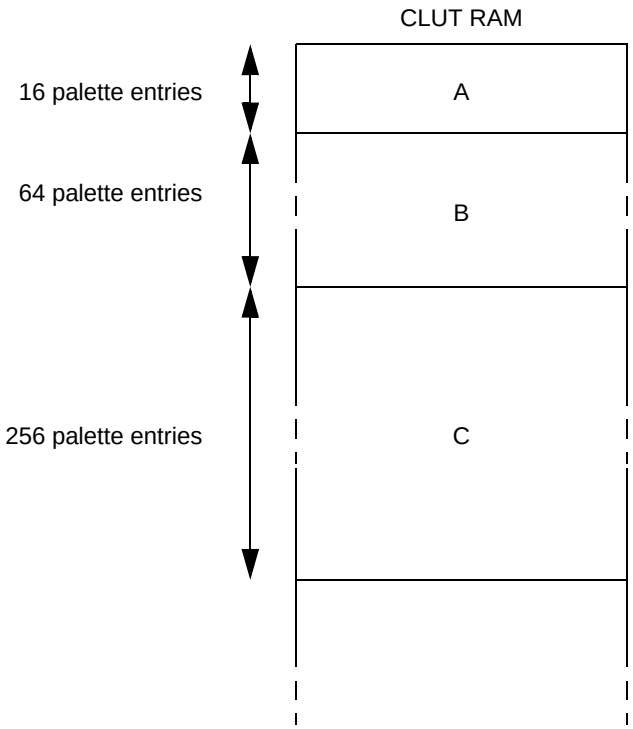



**Figure 12-81. An example of use for the CLUT RAM**

The CLUT RAM is 512 entries×32 bits in size and is mapped into the DCULite 16 KB memory space from address 0x2000 to 0x3FFF. For compatibility with the DCU3, the CLUT repeats at every 2 KB boundary in this space (0x2000, 0x2800, 0x3000, 0x3800). The 512 entries provides up to two full CLUTs for 8 bpp layers.

The CLUT RAM may be written at any time when the TFT LCD panel is not being driven with data. This means that the RAM can be modified when the DCULite is not enabled and during the vertical blanking period.

## 12.4.7 Gamma correction

The gamma table allows the user to define an arbitrary transfer function at the output of each color component. The function (Equation 12-4) is applied to each pixel after all blending is complete and before the data is driven to the TFT LCD panel. Gamma correction is optional and can be used to adjust the color output values to match the gamut of a particular TFT LCD panel, or to perform data inversion or data length reduction on each component.

**output_color_component = gamma_table[input_color_component]** ***Eqn. 12-4***

The table is arranged as three separate memory blocks within the DCULite memory map; one for each of the three color components. Each memory block has one entry for every possible 8-bit value and the entries

are stored at 32-bit aligned addresses. This means that the upper 24 bits are not used while reading/writing the gamma memories. See Figure 12-82 for details of the memory arrangement.

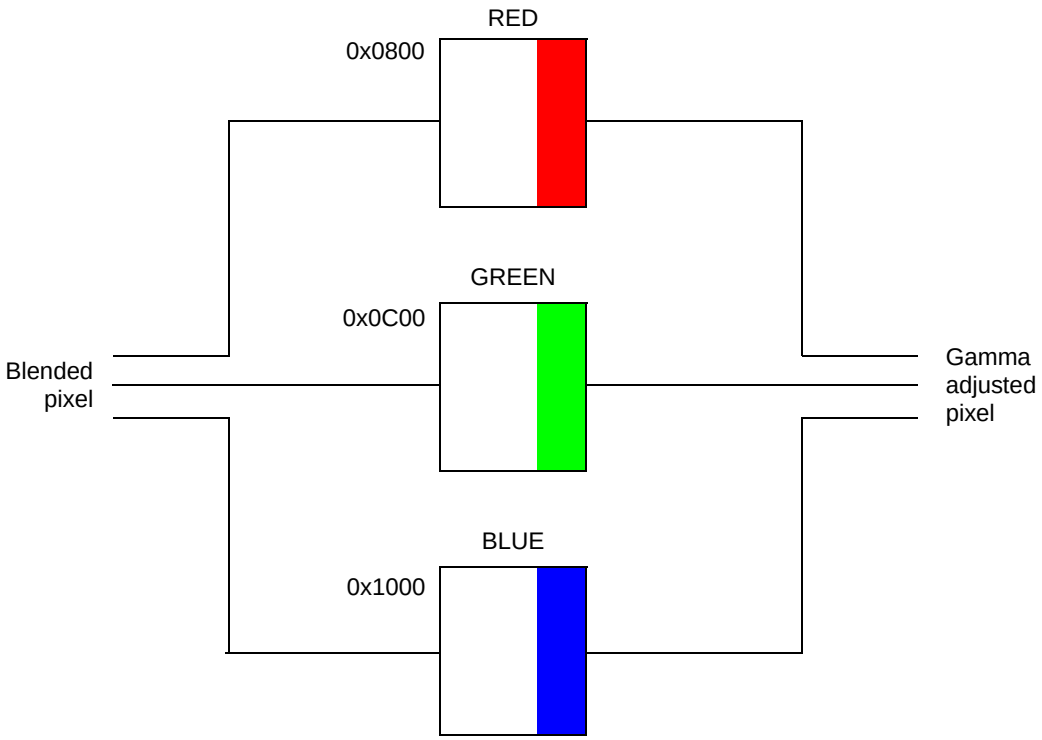



**Figure 12-82. Gamma Correction Table Organization**

The gamma table can only be read or written when the DCULite is not enabled or during the vertical blanking period.

## 12.4.8 Temporal dithering

This is a technique that allows the emulation of a color resolution higher than the resolution supported by the display. It is done by changing the intensity values over time sent to the display. The averaging done by the human eye gives the impression of the intensity of such alternating pixels as an interim value between the two supported intensity values. Temporal dithering is enabled by the DCU_MODE[DITHER_EN] bit.

The key features of the dithering block are

- Temporal dithering increases the optically perceived depth of a limited TFT display
- Supports display with 5-8 bits resolution per color component
- Independent dither control parameters per color component
- Support for safety mode

Temporal dithering is enabled and controlled by the EN_DITHER, ADDB, ADDG, and ADDR bits in the DCU_MODE register. The ADDx fields are each 2 bits wide and select how many bits to add to each color component. The typical setting is 8 minus the number of bits in each component required by the display.

The Random Number Generator (RNG) provides a random number of up to 3 bits. The number of bits provided is selected by the values of each component's ADDx field.

When pixels from a safety layer are encountered the RNG output is forced to 0 which effectively disables the temporal dithering block and these pixels are passed to the display unmodified.

The Add & Clamp block adds the eight bit pixel value to the three bit number generated by the RNG. The result is then clamped to the range of 0..255.

### 12.4.9 Special DDR mode

Special DDR mode is a special configuration that optimizes the use of an SDRAM memory by the DCULite by forcing the DCULite to fetch data in optimal chunks. In this special mode, only the two highest priority layers (0:1) are available in the DCULite. This mode is enabled using the DCU_MODE[DDR_MODE] bit.

When this mode is enabled the DCULite will fetch data in 32-byte chunks if the layer is encoded in 8, 16 or 32bpp formats thus optimizing the SDRAM throughput. Any layers in 1, 2, 4 or 24 bpp formats will be fetched using normal access thus they will not benefit from any optimization and may disrupt optimal access for any 8, 16 and 32 bit formatted layers if both are in the SDRAM. Therefore it is highly recommended to store any 1, 2, 4 or 24bpp layers in non-SDRAM memory such as on-chip SRAM or flash.

Depending on the layer configuration in use this mode may also benefit other synchronous memory interfaces such as QuadSPI.

### 12.4.10 Run Length Encoding (RLE) mode

RLE mode is available on the two highest priority layers (layer 0 and layer 1) and allows the DCULite to load RLE compressed data from memory and directly decode it for use on the panel. The mode is enabled using the RLE_EN bit in the control descriptor 4 register CTRLDESCLn_4.

This mode is only available when Special DDR mode is enabled and can only be used on a single layer at a time. In addition the mode only supports 8 bpp, 16 bpp (RGB565, ARGB1555, ARGB4444, APAL8), 24 bpp and 32 bpp BGRA8888 formats.

When enabled, the DCULite fetches data in 3-byte chunks and decompresses it before sending the decoded data to the normal DCULite blend process.

The dataflow for the RLE decoded data is as shown in Figure 12-83.

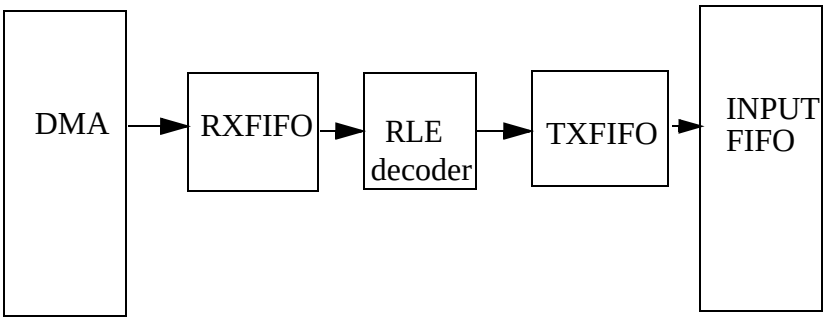



**Figure 12-83. RLE Decoding in DCULite**

The decoded data is read by the input FIFO once at least 8 bytes are available in the TXFIFO.The size of RXFIFO is 64x8 bits while the size of TXFIFO is 16x8 bits.

If both layer 0 and layer 1 have RLE_EN set, then the error flag RLE_ERR is asserted.

### 12.4.10.1 RLE decoding scheme

Before enabling an RLE encoded layer configure the COMP_IMSIZE register with the size of the compressed image. The decoder expects to read COMP_IMSIZE bytes from the image and produce from that the number of pixels specified in the layer Control Descriptor 1 register.

The format of RLE encoded layers is as follows:

- The first data byte read at the image address (Layer Control Descriptor 3) is a command byte (CMD[7:0])
- The ms bit (CMD[7]) indicates if the following bytes are raw or compressed pixels. One pixel can be 8-bit, 16-bit, 24-bit or 32-bit wide, depending on the BPP bitfield in the Layer Control Descriptor 4 register.
- The remaining 7 command bits (CMD[6:0]) specify the number of raw or compressed pixels that follow the command byte. This count is offset by 1 such that a value of 0 means one pixel follows.
- For compressed pixels (CMD[7] = 1), only one pixel follows the command byte. This pixel is repeated count+1 times on the layer. The pixel size may be 8, 16, 24 and 32 bits.
- For raw pixels (CMD[7] = 0) count+1 pixels follow the command byte and these are included on the layer as is. The pixel size may be 8, 16, 24 and 32 bits.
- If there is more data to decode then a new command follows after CMD+(1*{Pixel width}) bytes. This encoding is repeated until the whole image is decoded.

## 12.5 Timing, error and interrupt management

The DCULite can detect and raise status and error flags when the status of the system changes and when configuration or operational errors are detected.

### 12.5.1 Synchronizing to panel frame rate

Since the DCULite fetches data directly from memory independently of the CPU, there is the possibility that changes to the DCULite layer configuration or content can create incoherent content on the panel. To help avoid this situation there are five timing control flags that define when the DCULite recognizes and locks changes to its configuration. These can be used to manage changes to control descriptors, CLUT memory, or source graphics and so avoid coherency problems on the panel. All the timing flags are in the INT_STATUS register and can be used to generate interrupts from the DCULite.

The DCULite configuration is completely open during the vertical blanking period and control descriptors and some other registers may also be programmed at any time. The configuration that is present one HSYNC before the end of the vertical blanking period is the configuration used by the DCULite for the panel refresh phase.

The VS_BLANK and LS_BF_VS flags give indication of the start of the vertical blanking period. The VS_BLANK flag is set at the beginning of the vertical blanking period. The LS_BF_VS flag is set a given number of horizontal lines before the start of the vertical blanking period; the given number of lines is defined by the LS_BF_VS bit field in the THRESHOLD register.

The PROG_END flag indicates that the DCULite has locked the contents of its configuration registers for the new panel refresh period. No further changes are accepted to the DCULite configuration after this flag is set (until the next vertical blanking period).

The DMA_TRANS_FINISH flag indicates that the DCULite has completed fetching all data from memory in the current panel refresh cycle. This normally precedes the vertical blanking period and indicates that it is possible to change the contents of a memory that contains graphics used by the DCULite.

The VSYNC flag indicates that the DCULite has begun the next panel refresh period.

### 12.5.2 Managing the DCULite FIFOs and DMA activity

The DCULite fetches graphic data directly from internal and external memory using a dedicated DMA system and manages the output of data to the TFT LCD panel such that the panel always receives the pixel information when expected. Since the panel is sharing access to memory with the system DMA and CPU it cannot depend on the required data always being available at all times. It therefore uses input and output FIFOs to temporarily store incoming and outgoing data until required and thus reduces the opportunity for the panel to be starved of pixel data.

The DCULite manages the supply of graphic data to its format conversion and blending stages using input FIFOs that are 256×64 bits in size. The data that is driven to panel is managed using an output FIFO that is 128 pixels in size. See Figure 12-1 for a diagram of the input FIFO and output FIFO operation in the DCULite.

The input FIFOs are not accessible to the user but it is possible to set thresholds that control the DCULite behavior when the FIFOs are becoming full or empty and observe when the lower and higher thresholds are reached. This can help detect and avert situations where the DCULite is running out of data to send to the panel.

The FIFO thresholds are set in the THRESHOLD_INPUT_BUF_1/2 registers. The upper thresholds are set by the INP_BUF_Pm_HI bit fields (where m is the position of the pixel in the blend stack) and these set the point at which the DCULite pauses fetching data from memory. The maximum size of any DMA burst is fixed to 16 pixels and so is dependent on the graphic encoding. The lower thresholds are set by the INP_BUF_Pm_LO bit fields.

Each of the four input FIFOs has two flags that indicate whether the FIFO has reached its upper or lower threshold. The Pm_FIFO_HI_FLAG flags (where m is the position of the pixel in the blend stack) indicates that the input FIFO has reached the upper threshold. The Pm_FIFO_LO_FLAG indicates that the input FIFO has less data than its low threshold. Depending on when the low threshold is reached this may indicate a number of scenarios

- The expected graphical data is not available for the DCULite to load
- The DCULite is reaching the end of a frame and does not need to load any more data
- The blend stack does not need pixels of this priority

In the situation where the data is not available to the DCULite then there may or may not be an impact to the data visible on the panel. In the situation where the output FIFO is full then it is possible for the DCULite to accept a delay before it requires to use the incoming data.

The output FIFO is not accessible to the user but it is possible to set thresholds that control the DCULite behavior when the FIFO is becoming full or empty and observe the lower threshold. This can help detect and avert situations where the DCULite is running out of data to send to the panel.

The buffer thresholds are set in the THRESHOLD register. The upper threshold is set by the OUT_BUF_HIGH bit field and this indicates that sufficient data exists in the output buffer and processing should stop until the DCULite uses some of the values in the FIFO. If this value is set too low then the possibility of the DCULite running out of data to drive the panel is increased. The lower threshold is set by the OUT_BUF_LOW bit field.

When the output FIFO has emptied below its low threshold (OUT_BUF_LOW bit field) it sets the UNDRUN bit. In an under run situation there may or may not be an impact to the data visible on the panel. The impact depends on whether the DCULite is reaching the end of a frame and how close to running out the threshold is set.

The best guide to indicate whether the DCULite is able to supply the required pixel information to the panel is the output buffer. If the output is indicating that it is running out of data then the input FIFOs may help identify the areas of memory that are restricting the supply of data. Using these indicators can help to set the DCULite thresholds and ensure that the data throughput on the MCU is balanced correctly for all master devices.

Finally, note that the number of DCULite clock cycles to fetch and blend each pixel increases with the depth of the blend stack. However, the time taken to process the pixel data is fixed by the timing requirements of the panel. Therefore, for full performance across all color encodings the ratio between the DCULite clock and the pixel clock must increase as the blend stack depth increases:

- For two-pixel blending, the minimum DCULite clock must be the same as the TFT pixel clock.
- For three-pixel blending, the minimum DCULite clock must be twice the TFT pixel clock.
- For four-pixel blending, the minimum DCULite clock must be twice the TFT pixel clock.

### 12.5.3 Error detection

The DCULite asserts error flags when errors are detected in its configuration or when the user attempts to modify the configuration at an invalid point in the panel refresh period or when it is unable to access the required source data. The error flags may raise an interrupt if enabled to do so by the related mask bit in the corresponding mask register.

Error flags are stored in the PARR_ERR_STATUS and INT_STATUS registers.

Errors in the DCULite configuration are collected in the PARR_ERR_STATUS register. The flags Ln_PARR_ERR (where n is the layer number) indicate an error in the configuration of the layer which indicates a layer with a horizontal dimension that is smaller than the minimum size defined by the layer encoding (see Section 12.4.4.3, “Layer size and positioning”). The DISP_ERR flag indicates that the VSYNC and HSYNC pulse widths are configured to the invalid value of 0. The HWC_ERR flag indicates that the hardware cursor is either larger than the available memory or is placed in an off-panel position. The SIG_ERR indicates that the signature calculation specifies an area that extends beyond the panel size. The RLE_ERR indicates that more than one layer has RLE enabled.

Reads of CLUT RAM during the period when the TFT LCD panel is being updated do not return the CLUT RAM content.

Errors caused when the DCULite is unable to access its required source data are collected in the INT_STATUS register. These errors are indicated by the UNDRUN flag and the Pm_FIFO_LO_FLAG flags (where m is the position in the blend stack)

### 12.5.4 Interrupt generation

The DCULite generates interrupt through four lines that are controlled by the contents of six registers:

- INT_STATUS
- INT_MASK
- PDI_STATUS
- MASK_PDI_STATUS
- PARR_ERR STATUS
- MASK_PARR_ERR STATUS

There are four interrupt status lines defined.These lines are grouped as follows

- Timing based interrupts:
  - VSYNC
  - LS_BF_VS
  - VS_BLANK
  - PROG_END
  - DMA_TRANS_FINISH
- Functional interrupts:
  - UNDRUN
  - CRC_READY

- CRC_OVERFLOW
- P1_FIFO_HI_FLAG
- P1_FIFO_LO_FLAG
- P2_FIFO_LO_FLAG
- P2_FIFO_HI_FLAG
- P3_FIFO_HI_FLAG
- P3_FIFO_LOW_FLAG
- P4_FIFO_HI_FLAG
- P4_FIFO_LOW_FLAG
- IPM_ERROR

- Parameter error interrupts
  - Layer Error
  - Signature Calculator Error
  - Display Error
  - HWC_error
  - RLE error
- PDI-related interrupts (pdi_int)
  - This includes PDI related interrupts. See Section 12.8.1.8, “PDI-related Interrupts,” for a description.

When any interrupt occurs, the host can identify which type of interrupt has occurred by reading the interrupt status register/PDI status register/PARR_ERR status register.

## 12.6 Register protection

There is a customized register protection scheme on the DCULite that is different to the protection scheme implemented elsewhere on the MCU. The scheme provides a mechanism to protect certain registers in the DCULite from being written.

### 12.6.1 Operation of scheme

The register protection scheme provides a two-step protection scheme for the protected register.

Firstly, each register has an associated soft lock bit (SLB) that prevents further writes to the register when it is set. Each SLB has a corresponding write enable (WEN) bit that must be set in the same write operation as the SLB. The SLB can be set or cleared by writing a '1' or '0' to it while its WEN bit is set. The SLB bits are in the Soft Lock Registers L0 and L1, DISP_SIZE, HSYNC/VSYNC_PARA, POL, L0_TRANSP and L1_TRANSP registers.

Secondly, there is a hard lock bit (HLB) in the Global Protection Register which prevents all changes to soft lock bits. The HLB can only be cleared by a system reset.

If a write is made to a register whose SLB is set then a transfer error occurs that generates an IVOR1 exception on the CPU. Similarly if the HLB is set then any write to the SLB registers causes a transfer error.

### 12.6.2 List of protected registers

The register protection scheme applies to the following registers:

- All Layer 0 control descriptors CTRLDESCL0_1 to CTRLDESCL0_7
- All Layer 1 control descriptors CTRLDESCL1_1 to CTRLDESCL1_7
- Layer 0 foreground and background registers for transparency mode FG0_FCOLOR and FG0_BCOLOR
- Layer 1 foreground and background registers for transparency mode FG1_FCOLOR and FG1_BCOLOR
- All Control Descriptors & Transparency Registers for Layer1
- DISP_SIZE
- HSYNC_PARA
- VSYNC_PARA
- SYN_POL

## 12.7 Safety mode

Safety layers are used in a multi-layer DCULite environment for the purpose of guaranteeing that the content is driven to the display regardless of the setting of remaining layers and the pixel manipulation algorithms of the DCULite. Features such as this are a requirement from qualification institutes to be able to reach a safety level of SIL2 or ASILB. The DCULite has two safety layers (Layer 0 and Layer 1) which also have the highest priority. When Safety Mode is active the safety layers can use chroma keying for complex area description; however, alpha blending for the layer is always ignored. Additionally, if a layer has safety mode enabled then a layer format of 32 bpp or luminance is not allowed. Using these formats causes the layer to be disabled.

Safety Mode is implemented using a signature calculator module implemented inside the DCULite that calculates two signatures (pixel value and pixel position) for a predefined area of the frame. The user makes layer 0 and/or layer 1 active as a safety layer, defines the window/area of the pixels for which the signature is to be calculated, and enables safety mode. When enabled, the signature calculator starts to calculate the signature after the first pixel in the selected area is available and after the start of the next frame (VSYNC). It is also possible to calculate the signature value for all pixels if the DCU_MODE[TAG_EN] = 0.

As the pixels in the selected area become available they are "tagged" by the DCULite, except for those removed by chroma-keying. These tags identify the pixels to be included in the signature calculation. The signature calculation itself is an industry-standard CRC.

The DCULite asserts the CRC_READY flag at the end of any frame which has Safety Mode enabled. This can be used to indicate the completed signature calculations for each full frame of pixels after the mode is enabled. The completed signature can then be compared against a pre-calculated value with any difference

indicating that the pixels displayed did not match what was expected. The signature calculator then continues to calculate the CRC for the next frame. If the CRC_READY flag is not processed within one frame time period, then the CRC_OVERFLOW interrupt is issued and the latest calculation overwrites the previous value. Since the CRC_READY flag is set at the end of any frame with Safety Mode enabled, it is possible that a full frame has not yet been completed and therefore no signature calculation exists for the frame that set the flag.

If the user has set the NEG bit for the DCULite, which indicates that the pixels fed to the display are inverted, then the value CRC is calculated on non-inverted values. The position CRC, however, remains as is.

Normal arbitration takes place only when a pixel has content on layer 0 and layer 1 but where safety mode is enabled in layer 1.

The polynomial used for CRC calculation is

$$(x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1) \quad \textit{Eqn. 12-5}$$

For the value CRC, the 24-bit value of each output pixel (after decoding) is sent to the polynomial. For the position CRC, the value sent is

$$\text{(pixel_delta_y * display.delta_x) + (pixel_delta_x + 1)} \quad \textit{Eqn. 12-6}$$

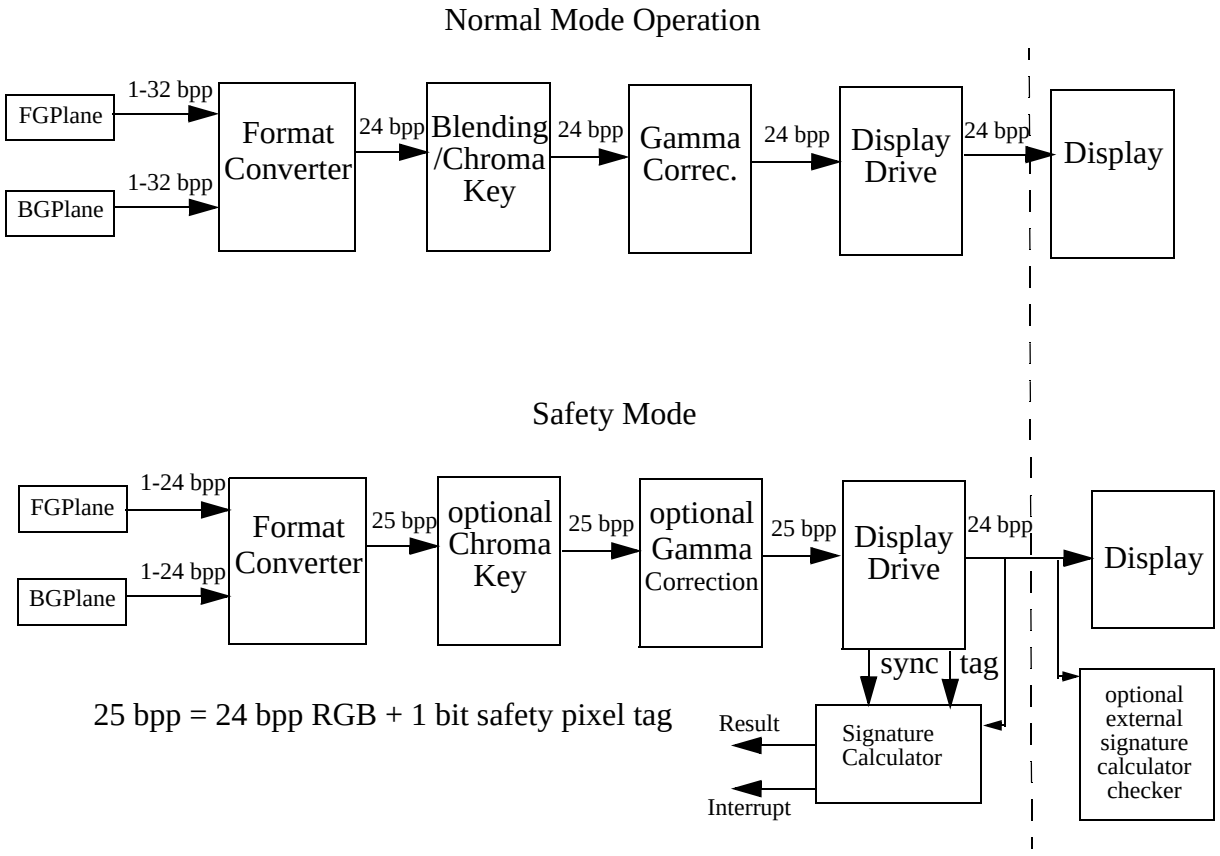



**Figure 12-84. Safety Mode block diagram**

## 12.7.1 CRC area description

### 12.7.1.1 Configuring the CRC calculation

1. CRC_VAL and CRC_POS are calculated when Safety Mode is enabled using DCU_MODE[SIG_EN].
2. The CRC can be calculated for part of the panel as shown in Figure 12-85. The green portion on the panel is identified using the SIGN_CALC_1 and SIGN_CALC_2 registers which defined the size of the area and the location of the area respectively. If SIGN_CALC_1 and SIGN_CALC_2 are configured appropriately this calculation will cover the whole panel.

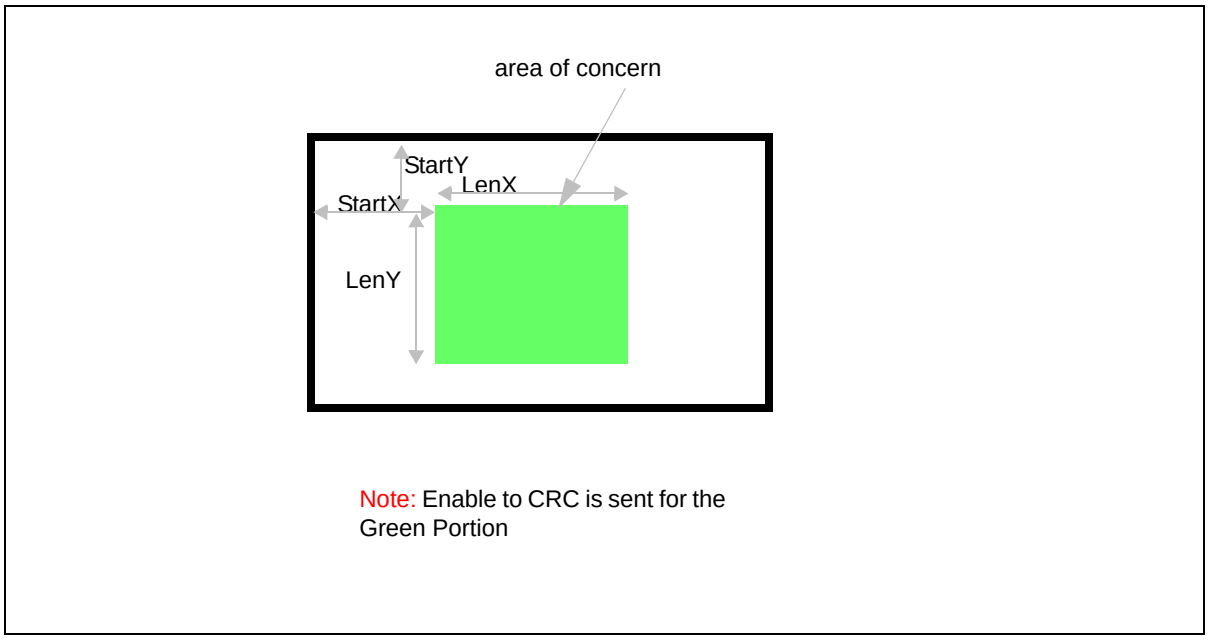



**Figure 12-85. Safety Mode enabled for part of the screen**

3. The CRC can be calculated exclusively for layers 0 and 1 by setting DCU_MODE[TAG_EN] and enabling the SAFETY_EN bit in control descriptor 4 of each of the layers. In this configuration the CRC is calculated using values from the layers only where they intersect the area of interest defined in SIGN_CALC_1 and SIG_CALC_2. An example is shown (in dark pink) in Figure 12-86.

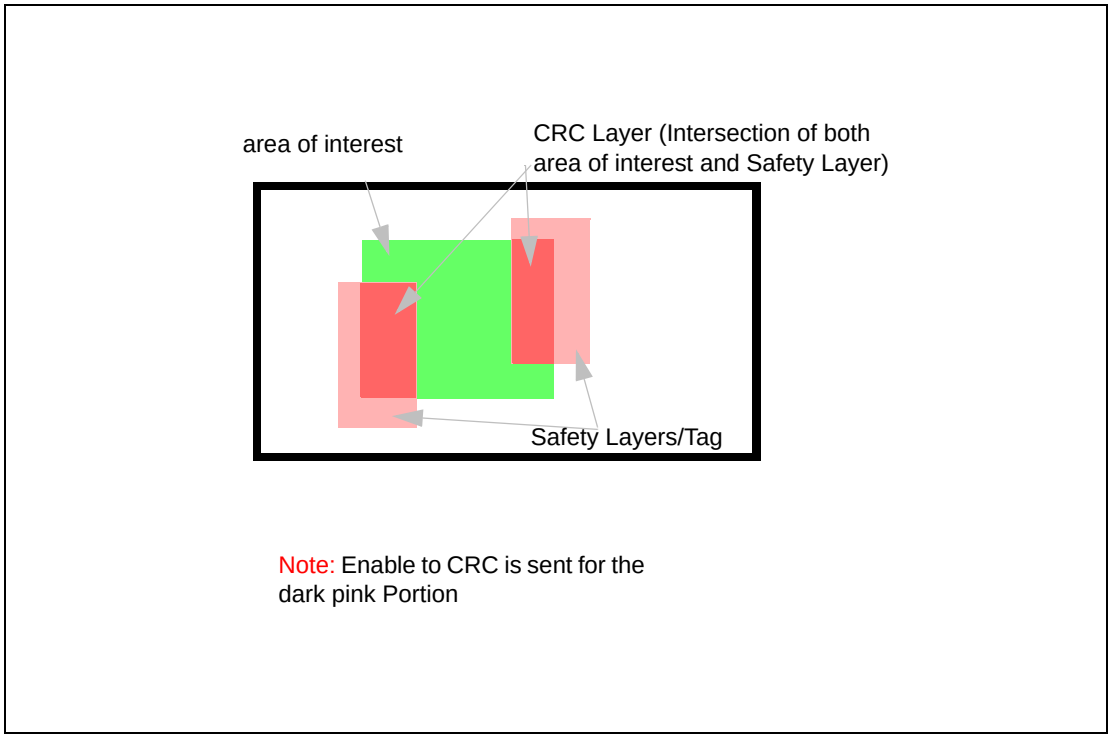



**Figure 12-86. Safety Mode with DCU_MODE[TAG_EN] = 1**

## 12.7.2 Summary of operation

The area included in the CRC calculation is summarized in Table 12-72. The initial value on all CRC calculations is 0x00000000.

**Table 12-72. Supported area**

| Area DCU_MODE[TAG_EN] | | Note |
|---|---|---|
| Full | 1'b0 | SIGN_CALC_2[SIG_HOR_POS] = 0<br>SIGN_CALC_2[SIG_VER_POS] = 0<br>SIGN_CALC_1[SIG_HOR_SIZE] = Panel width<br>SIGN_CALC_1[SIG_VER_SIZE] = Panel height |
| Part | 1'b0 | The SIGN_CALC parameters have values other than those mentioned above — see Figure 12-85 |
| Safety Layer<br>(Layer 0 and 1 only) | 1'b1 | The included portion of the safety layers depends on<br>• The portion lying within the area defined by SIGN_CALC_1 and SIGN_CALC_2<br>• The pixels removed by chroma keying functionality |

## 12.8 Parallel Data Interface (camera interface)

### 12.8.1 PDI interface description

#### 12.8.1.1 Introduction

This block extracts the timing and pixel information from an external video source and passes it to the DCULite block to synchronize with the timing and display the pixel information on the TFT LCD screen. The BGND layer in the DCULite is replaced by the incoming video stream.

The PDI requires that the incoming video stream match the resolution and timing at which the DCULite is driving the TFT LCD panel and the incoming stream must not be interlaced.

The PDI can also be configured in slave mode in which case it ignores the pixel information from the external video source and only passes the timing information to the DCULite to synchronize with. In this instance the DCULite will continue to operate as normal (the BGND layer will not be altered) but will use the timing from the PDI.

The PDI shares configuration registers with the DCULite.

#### 12.8.1.2 PDI interaction with other modules

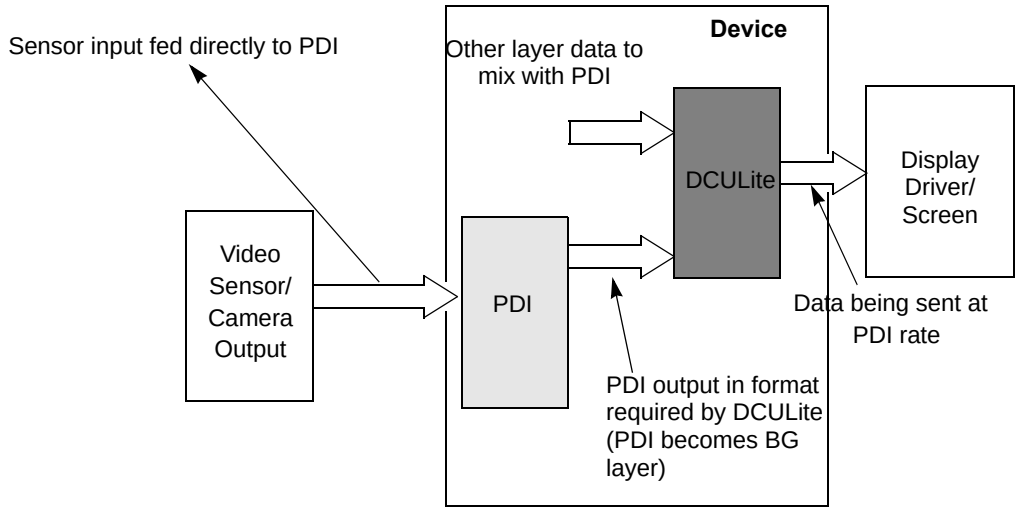



**Figure 12-87. PDI interacting directly with the external sensor**

In Figure 12-87, the PDI accepts data and timing directly from the external video source. The external device must output the video in a digital format supported by the PDI.

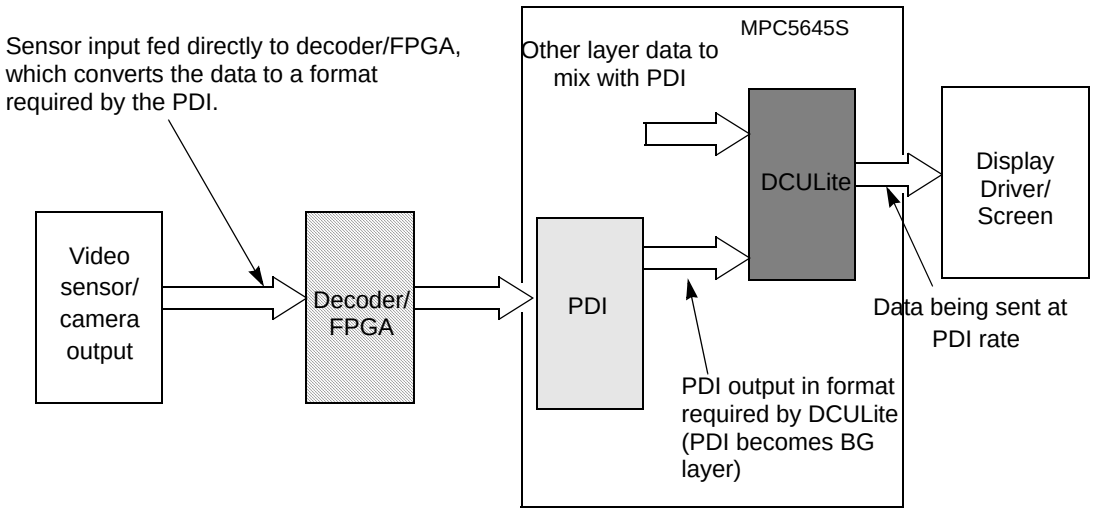



**Figure 12-88. PDI interacting with FPGA in between**

In the case shown in Figure 12-88, the video stream is sent to a decoder or an FPGA which alters the incoming stream to a format which is compatible with the PDI. A decoder/FPGA is required if a video source with an analog output format (e.g. NTSC/PAL) is used. The decoder should perform analog-to-digital conversion on the stream and ensure the timing match that of the DCULite. The incoming stream may also need to be de-interlaced.

The PDI is compatible with various input formats:

- Normal Mode: the PDI clock frequency must be equal to pixel clock frequency required by the TFT display driver
- Narrow Mode: the PDI clock frequency is double the desired pixel clock frequency.
- External Synchronization: The timing signals (HSYNC, VSYNC, and DE) each have a dedicated input pin. (DE is optional)
- Internal Synchronization: The timing signals (HSYNC, VSYNC and DE) are embedded in the data stream, as such only the DATA and PCLK inputs are required. (See Section 12.8.1.4, “ITU-R BT.656 sync information extraction”.)

Before the DCULite locks onto the PDI timing signals it will run on the internal DCULite clock. After lock has been achieved, the DCULite will switch to the clock from the PDI stream and the this is then used to send data and timing signal to TFT/LCD display driver.

In all cases, the resolution of the incoming stream and the HSYNC and VSYNC frequency must be the same as that for TFT screen. All the horizontal parameters (Front Porch width, Back Porch width, Pulse width) and vertical parameters (Front Porch width, Back Porch width, Pulse width) must be same as that of TFT screen.

When PDI is the background layer, no other layer can be a background layer for that particular frame. Only one background layer is possible i.e. PDI Layer when PDI is enabled.

### 12.8.1.3 Features

The PDI supports the following:

- RGB565, RGB666, 8 bit monochrome format,YCbCr422 mode
- Max input frequency of 32 MHz in 8/16/18 normal mode input
- Max input frequency of 64 MHz in 8 bit muxed (narrow) mode
- External synchronization using PDI_HSYNC, PDI_VSYNC and PDI_PCLK
- External synchronization using PDI_HSYNC, PDI_VSYNC, PDI_PCLK, and PDI_DE
- Internal synchronization using PDI[17:0] and PDI_PCLK is supported for RGB565 and YCbCr422 muxed mode only

**Table 12-73. Supported RGB format and Sync Format**

| RGB Format | Data Input Bus |
|---|---|
| 8-bit monochrome | 8 bit |
| RGB565 | 16 bit |
| RGB666 | 18 bit |
| RGB565 muxed (uses narrow mode) | 8 bit |
| **Sync Format** | **Pin Used** |
| Internal Sync (Valid only for RGB565 and YCbCr in narrow mode) | Pclk |
| External Sync | PDI_HSYNC, PDI_VSYNC, PDI_PCLK |
| External sync (with data En) | PDI_HSYNC, PDI_VSYNC, PDI_PCLK, PDI_DE |

### 12.8.1.4 ITU-R BT.656 sync information extraction

According to ITU-R BT.656 recommendation, the incoming digital video will have a pdi_clk signal and 8 data bits. The data bits can contain both the video data and the timing reference signals (VSYNC and HSYNC).

The timing signals are encoded at the start and end of each line by timing reference codes known as Start of Active Video (SAV) and End of Active Video (EAV). The SAV and EAV codes are identified by their preamble of three bytes (0xFF,0x00,0x00). Due to this, neither 0x00 or 0xFF can be used during the Active Video Data. The preamble is followed by the XY status word which contains a Field Bit (F), a Vertical blanking bit (V) and Horizontal blanking bit (H) and four protection bits for single bit error correction and detection.

The H bit is set to 1 to denote an EAV — that is the end of a line, and the beginning of the horizontal blanking period. The V bit is set to 1 to denote the beginning of the vertical blanking period. The F bit is used for interlaced video to denote if the forthcoming line is odd or even.

The remaining 4 bits contains make up the protection bits for single bit error correction and detection. It should be noted that F and V fields are only allowed to change as part of EAV sequences i.e during transitions from H=0 to H=1.

An entire line of video comprises Active Video + Horizontal Blanking (from the start of the EAV code until the end of the SAV code) and Vertical Blanking (the space where V = 1).

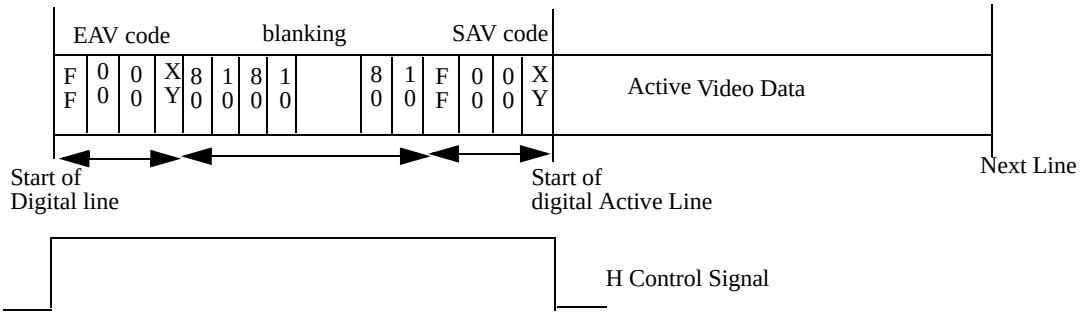



**Figure 12-89. ITU-R BT.656 8 bit parallel data format for 525 video system**

The SAV and EAV codes have a defined preamble of three bytes (0xFF,0x00,0x00) followed by XY status word which aside from the Field (F), Vertical blanking (V) and Horizontal blanking bits contains four protection bits for single bit error correction and detection. Also, F and V fields are only allowed to change as part of EAV sequences i.e transition from H=0 to H=1.

| Data Bit | FirstWord (FF) | SecondWord (00) | ThirdWord (00) | FourthWord (XY) |
|---|---|---|---|---|
| D7(MSB) | 1 | 0 | 0 | 1 |
| D6 | 1 | 0 | 0 | F |
| D5 | 1 | 0 | 0 | V |
| D4 | 1 | 0 | 0 | H |
| D3 | 1 | 0 | 0 | P3 |
| D2 | 1 | 0 | 0 | P2 |
| D1 | 1 | 0 | 0 | P1 |
| D0 | 1 | 0 | 0 | P0 |

**Figure 12-90. Control Byte Sequence for 8 bit/10 bit video**

The bit definitions for the status word XY are shown in Table 12-74.

**Table 12-74. Status word definitions**

| Bit | Definition |
|---|---|
| F | 0 for field 0<br>1 for field 1 |
| V | 1 during vertical blanking period<br>0 when not in vertical blanking |
| H | 0 at SAV<br>1 at EAV |
| P3 | V XOR H |
| P2 | F XOR H |

**Table 12-74. Status word definitions (continued)**

| Bit | Definition |
|---|---|
| P1 | F XOR V |
| P0 | F XOR V XOR H |

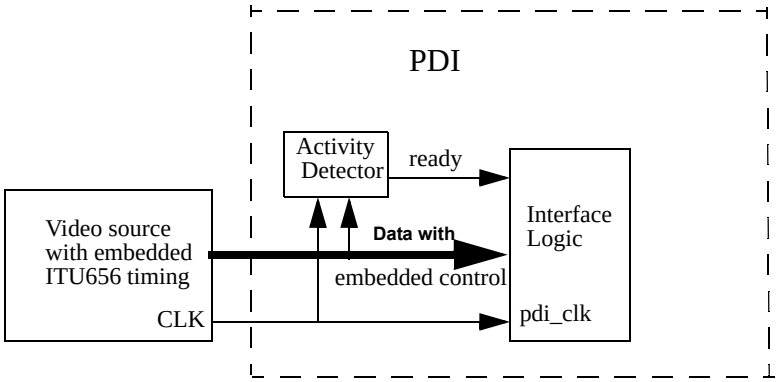



**Figure 12-91. PDI Input data mode**

Figure 12-91 represents the scenario in which data from an ITU-R BT.656 compliant video source is fed into the PDI interface. The data has embedded control information (HSYNC and VSYNC) in it. An activity detector checks for the transitions on the PDI bus. It samples the values on the PDI bus and once it has detected valid activity, sets a flag in the status register and can optionally trigger an interrupt. The PDI interface has a state machine which extracts the control information from the video data. The machine checks the video data for the Preamble Field (0xFF,0x00,0x00) and then depending on the status bits XY decides if it has received a valid control signal.

### 12.8.1.5 Normal and Narrow Mode

In normal mode, the PDI supports a maximum input frequency of 32 MHz. In narrow mode, the PDI supports a maximum input frequency of 64 MHz.

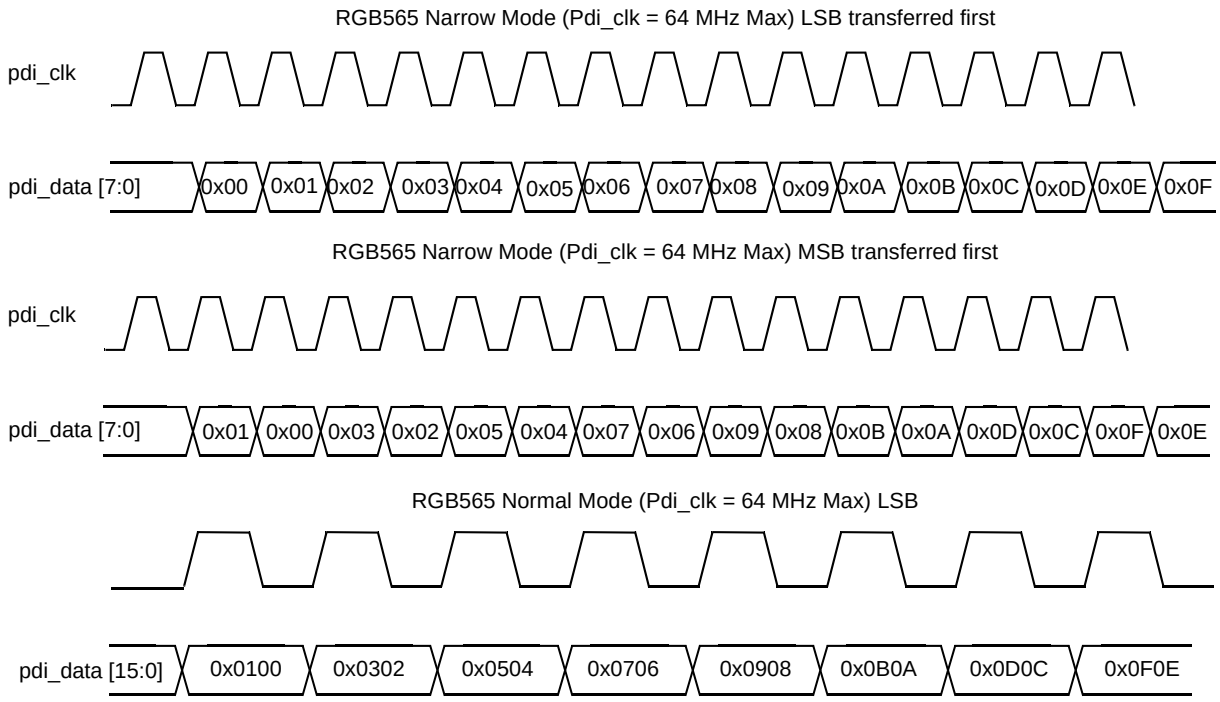



**Figure 12-92. Data Transfer in Normal and Narrow Mode**

The byte transferred first (MSB or LSB) depends on the configuration register as shown in Figure 12-92. This does not affect the sync preamble sequence in case internal sync mode.

On this device, the incoming RGB data is mapped onto the PDI pins as described in Table 12-75.

**Table 12-75. Mapping of RGB data onto PDI pins**

| Mode | Mapping |
|---|---|
| Normal (full 18-bit PDI interface) | PDI[17:12] = R[5:0]<br>PDI[11:6] = G[5:0]<br>PDI[5:0] = B[5:0] |
| Normal (RGB565 16-bit PDI interface) | PDI[15:11] = R[4:0]<br>PDI[[10:5] = G[5:0]<br>PDI[4:0] = B[4:0] |
| Narrow (8-bit PDI interface) | RGB565:<br>In first clock cycle, PDI[7:0] = { R[4:0]}<br>In second clock cycle, PDI[7:0] = { G[2:0]}<br>YCbCr:<br>In first clock cycle, PDI[7:0] = { Cb[7:0] }<br>In second clock cycle, PDI[7:0] = { Y0[7:0] }<br>In third clock cycle, PDI[7:0] = { Cr[7:0] }<br>In forth clock cycle, PDI[7:0] = { Y1[7:0] } |

### 12.8.1.6 Modes of Operation Based on Sync Extraction

#### 12.8.1.6.1 PDI input data (external sync mode)

In external sync mode the timing signals (HSYNC, VSYNC and, optionally, Date Enable) are provided to the PDI input timing pins by the external video source.

External sync mode can be used in both normal mode and 8-bit narrow mode, but cannot be used in conjunction with the YCbCr data format. In the instance that external sync and narrow mode is selected, the external signals are used, and any timing information (EAV/SAV) embedded in the data stream is ignored.

As in Figure 12-94, PDI data enable (PDI_DE) should be low during VSYNC and HSYNC pulse, VSYNC front porch (FP_V) and back porch (BP_V), HSYNC front porch (FP_H) and back porch (BP_H). This is valid for Data Enable Mode when the PDI_DE_EN bit is set in the DCU_MODE register (i.e. mode with HSYNC, VSYNC, PDI_DE and PDI_PCLK as pin signals).

Pulse width, Front and back porch values should be picked from those programmed in DCULite registers. In order to achieve lock, it must have same value as that of TFT screen. Front porch and back porch value can be zero. Pulse width and TFT screen size parameters cannot be zero. In case they are programmed as zero, it might lead to malfunctioning of the validation state machine.

As in Figure 12-93 HSYNC must occur during the VSYNC and vertical blanking period. The time between 2 HSYNC should be same during VSYNC and vertical blanking as during the active line period. As in Figure 12-93 the positive edge of HSYNC and VSYNC should be aligned. As in Figure 12-93 the positive edge of the HSYNC and start of the vertical front/back porch should be aligned. The polarity of HSYNC and VSYNC is selectable.

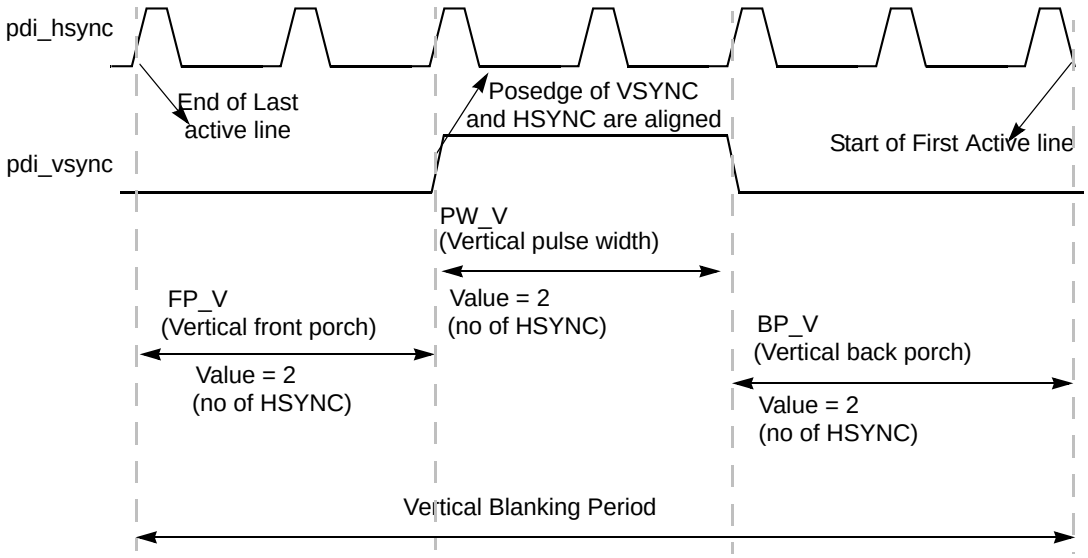



**Figure 12-93. Relationship between HSYNC and VSYNC in external synchronization**

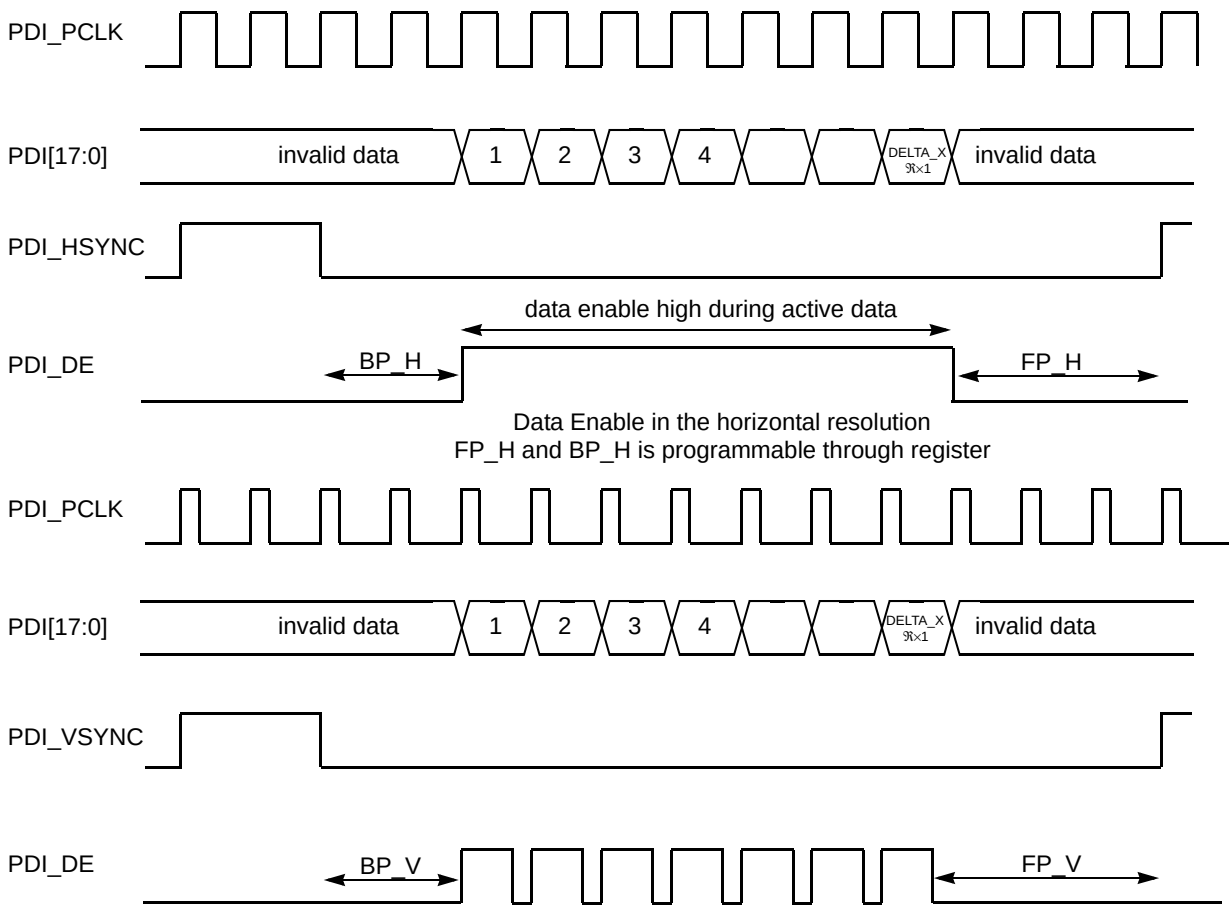



**Figure 12-94. Occurrence of HSYNC, VSYNC, and PDI_DE for the entire frame**

### 12.8.1.6.2 PDI input data (internal sync extraction mode)

In internal sync mode the timing parameters (horizontal and vertical blanking) are encoded into the data stream.

Internal sync mode can only be used in 8-bit narrow mode.

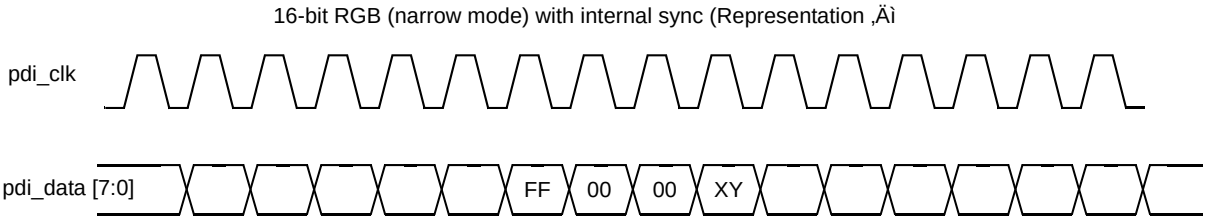
In Figure 12-95, XY is used to decode the vertical and horizontal blanking period.

**Table 12-76. XYh Value**

| Bit | Value | Description |
|---|---|---|
| 7 | 1'b1 | Always 1'b1. This is checked while decoding sync preamble |
| 6 | F | Not considered in the state machine logic |
| 5 | V | 1'b1 during vertical blanking<br>1'b0 elsewhere |
| 4 | H | 1'b0 for start of active video<br>1'b1 for end of active video |

**Table 12-76. XYh Value (continued)**

| Bit | Value | Description |
|---|---|---|
| 3<br>2<br>1<br>0 | P3<br>P2<br>P1<br>P0 | Protection bits (used to detect ECC errors). |




**Note:** Preamble sequence is independent of data sequence

**Figure 12-95. Location of Sync Preamble in Narrow Mode**

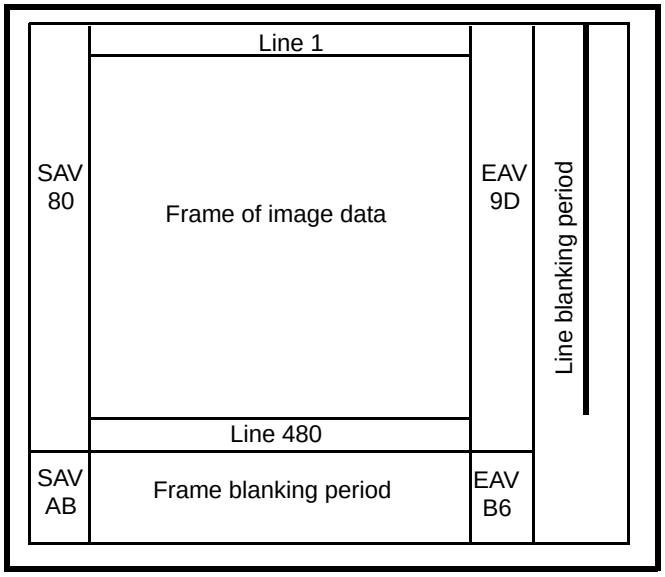
Sync Preamble would come continuously for 4 clock cycles as shown in Figure 12-95. It does not depend on which byte is coming first in data (MSB or LSB). Sync extraction is done using PDI[7:0] to identify the horizontal and vertical blanking period using H and V field of the 'XYh' data as mentioned in Table 12-73.

ITU 656 Sync preamble pattern (FFh 00h 00h) has to be masked out in the RGB data. The data stream must not include FFh 00h 00h as the valid pixel data to avoid malfunction by the validation state machine.

Horizontal blanking period must continue during the Vertical blanking period. The gap between 2 horizontal blanking periods should be the same during vertical blanking period as during line active. All vertical and horizontal parameter values are validated against the DCULite registers programmed by the user. Polarity of HSYNC and VSYNC are selectable. Horizontal blanking and vertical blanking must be aligned as shown in Figure 12-96. During blanking period the input stream must be a 80h 10h 80h 10h sequence. This sequence must be present during both horizontal (line) blanking and vertical (frame) blanking.

As the PDI does not support interlaced video, the Framing Bit (F field in XYh) will be ignored during timing extraction. Narrow mode is compatible with RGB565 and YCbCr422 muxed modes. Each header contains an ECC value, which the PDI will check. The PDI is capable of detecting an error but not correcting it.

As with External sync mode, the value of front and back porch can be zero but the pulse width and TFT screen parameter cannot be zero.




NOTE: The SAV and EAV bytes are included as part of the blanking period.

**Figure 12-96. Relationship Between Hblank and Vblank in Internal Sync**

#### 12.8.1.6.3 YCbCr mode

The DCULite can process incoming data from the PDI and from memory in YCbCr422 format. Both sources use the same RGB conversion and interpolation equations, however, the coefficients for the equations and enable for the interpolation are independently controlled.

In YCbCr mode, the PDI extracts the ITU656 sync (FF-00-00) and sends the video to the processing functions. The first processing function converts the 422 stream to a 444 stream, by providing interpolation on the chroma components of the stream depending on PDI_INTERPOL_EN bit. The second processing function converts the stream to RGB888/RGB565.

The DCULite can also select YCbCr format in the BPP field of layer descriptor 4 which allows the same process to be applied to values stored in memory rather than brought in from the PDI. Interpolation is controlled for the layers using the LYR_INTPOL_EN register.

The RGB pixel value is computed using following equations:

$$\text{Red} = \frac{(Y-16)\cdot yred}{512} + \frac{(Cr-128)Crred}{512} + \frac{(Cb-128)Cbred}{512} \qquad \textit{Eqn. 12-7}$$

$$\text{Green} = \frac{(Y-16)\cdot Ygreen}{512} + \frac{(Cr-128)Crgreen}{512} + \frac{(Cb-128)Cbgreen}{512} \qquad \textit{Eqn. 12-8}$$

$$\text{Blue} = \frac{(Y-16)\cdot yblue}{512} + \frac{(Cr-128)Crblue}{512} + \frac{(Cb-128)Cbblue}{512} \qquad \textit{Eqn. 12-9}$$

Note that the first multiplication (i.e (y-16)*ycoeff) is unsigned, the 2 others are signed.

The register values after reset are as follows:

Yred = 10’h254 (596/512 = 1.16)

Crred = 11’h331 (817/512 = 1.6)

Cbred = 12’h000

Ygreen = 10’h254(596/512 = 1.16)

Crgreen = 11’h660(-416/512 = -0.812)

Cbgreen = 12’hf38(-200/512 = -0.39)

Yblue = 10’h254(596/512 = 1.16)

Crblue = 11’h000

Cbblue = 12’h409(1033/512 = 2.017)

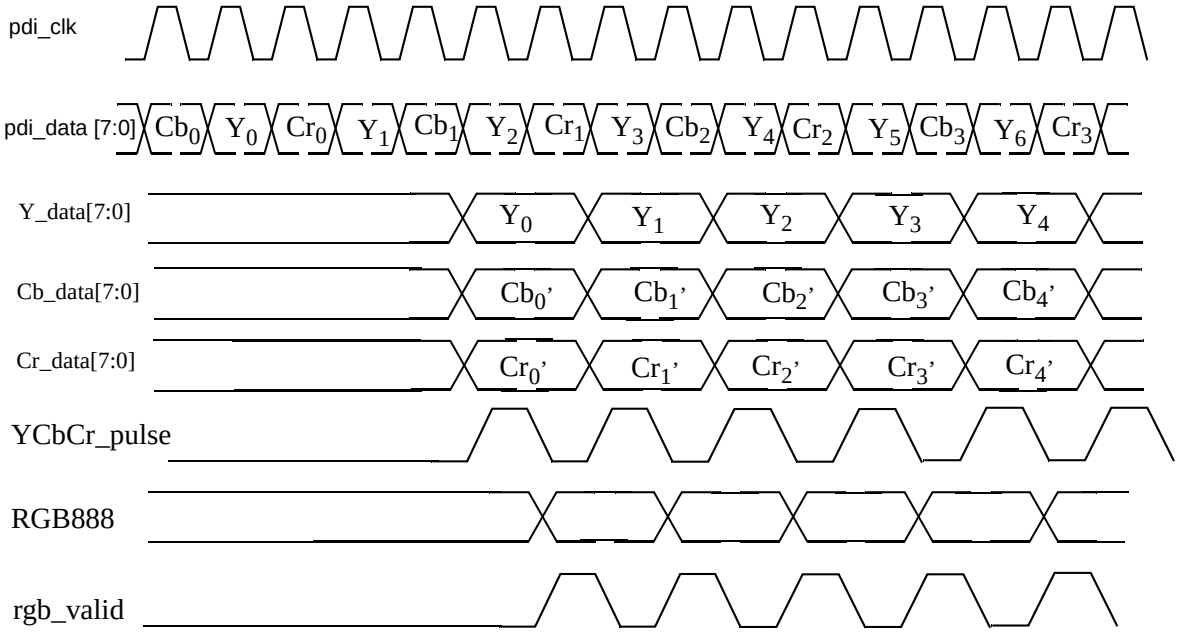



**Figure 12-97. YCbCr timing diagram**

$$Cb_n{}^{,\ddot{A}\hat{o}} = \begin{cases} Cb_{n/} & \text{for even n} \\ (Cb_{(n-1)/2}{}^{,\ddot{A}\hat{o}} + Cb_{(n+1)/2})/ & \text{for odd n} \end{cases} \qquad Cr_n{}^{,\ddot{A}\hat{o}} = \begin{cases} Cr_{n/} & \text{for even n} \\ (Cr_{(n-1)/2}{}^{,\ddot{A}\hat{o}} + Cr_{(n+1)/2})/ & \text{for odd n} \end{cases}$$

**Figure 12-98. YCbCr interpolation calculation**

### 12.8.1.7 Mode of operation depending on PDI[17:0]

PDI would support following modes (other than the Slave Mode):

- 8-bit monochrome (8 bit input data, each pixel info is coming in 1 clocks)
- 16 bit - RGB565 (16 bit input data, each pixel info is coming in 1 clocks)
- 18 bit - RGB666 (18 bit input data, each pixel info is coming in 1 clocks)
- 16 bit - RGB565 (8 bit input data, each pixel info is coming in 2 clocks)
- 16-bit - YCbCr422 (8-bit input data, info for 2 co-sited pixels coming in 4 clocks)

Data info extraction is given in Table 12-76.

**Table 12-77. Data extraction in all possible modes**

| PDI mode | Narrow mode | Pins | Data | Notes |
|---|---|---|---|---|
| 8-bit monochrome mode | 1'b0 | 8 bit | PDI[7:0] | — |
| RGB565 | 1'b0 | 16 bit | PDI[15:0] | — |
| RGB666 | 1'b0 | 18 bit | PDI[17:0] | — |
| RGB565 muxed | 1'b1 | 8 bit | PDI[7:0] | Data from two clocks are combined. |
| YCbCr422 | 1'b1 | 8 bit | PDI[7:0] | Data from four clocks are combined for 2 pixels. |

The 8-bit monochrome image is equivalent to 8 bit grayscale images. Conversion of 8 bit monochrome to RGB. All RGB components have value equal to 8 bit monochrome value.

RGB extraction starts when PDI is enabled (from the next falling edge of validated vertical blanking period)

#### 12.8.1.8 PDI-related Interrupts

PDI can be configured to trigger an interrupt when synchronization is achieved i.e. it receives the pre-specified numbers of frames without error. PDI can also give an interrupt when synchronization is lost i.e. it receives any error in frame there after. This interrupt is raised when HSYNC/VSYNC is lost.

The PDI can also trigger an interrupt if there is either a one bit or a multiple bit error in the ECC value during the extraction of the preamble in internal synchronization mode.

Blanking sequence error interrupt can be triggered in case 80h 10h is not found in vertical and line blanking period during internal synchronization.

Activity detection interrupt for CLK detection.

HSYNC, VSYNC and DE detection interrupts can be set to generate upon transitions on the PDI_HSYNC, PDI_VSYNC and PDI_DE pins.

Activity lost interrupt for PDI_CLK — for the PDI_CLK lost interrupt, is triggered when the PDI_CLK is less than DCULite module clock frequency divided by 32. (i.e. if DCULite module clock freq. max = 64 MHz, then the PDI_CLK_LOST flag will be set if pdi clk freq. min < 2 MHz).

All interrupt flags are "write-one-to-clear" and all interrupts are maskable.

PDI must reset to show the latest status of the clock activity detect interrupt.

## 12.9 Switch between DCU mode and PDI mode (top-level description)

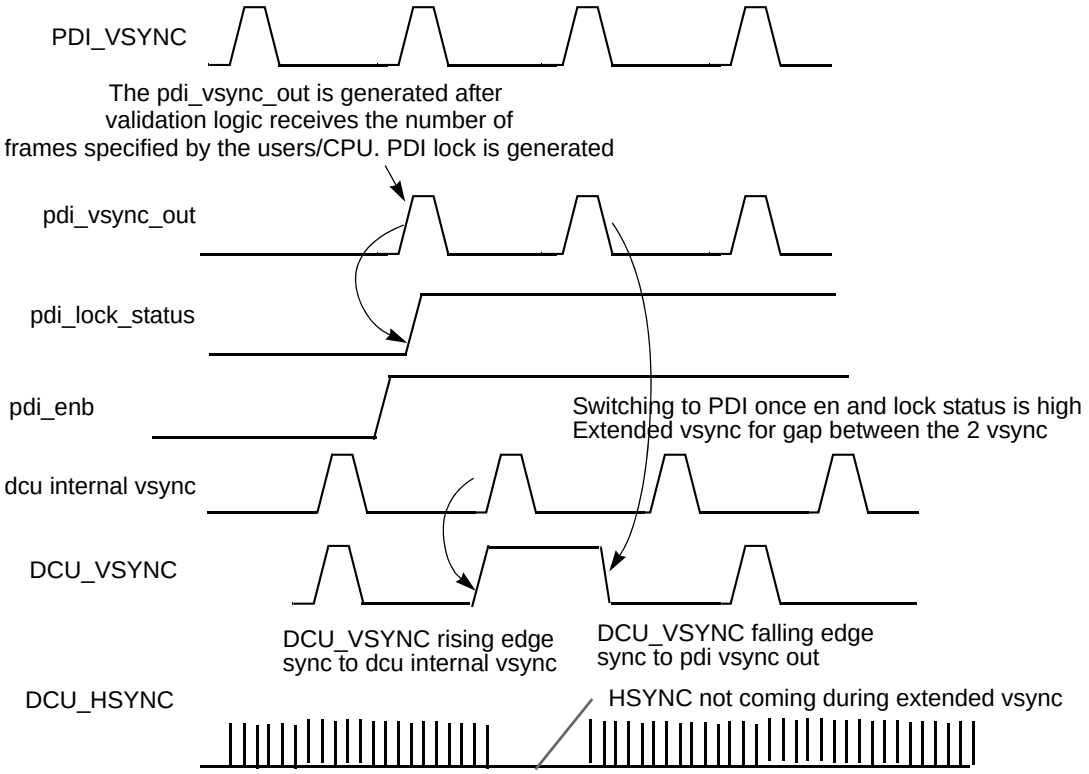



**Figure 12-99. Switch to PDI on receiving the interrupt**

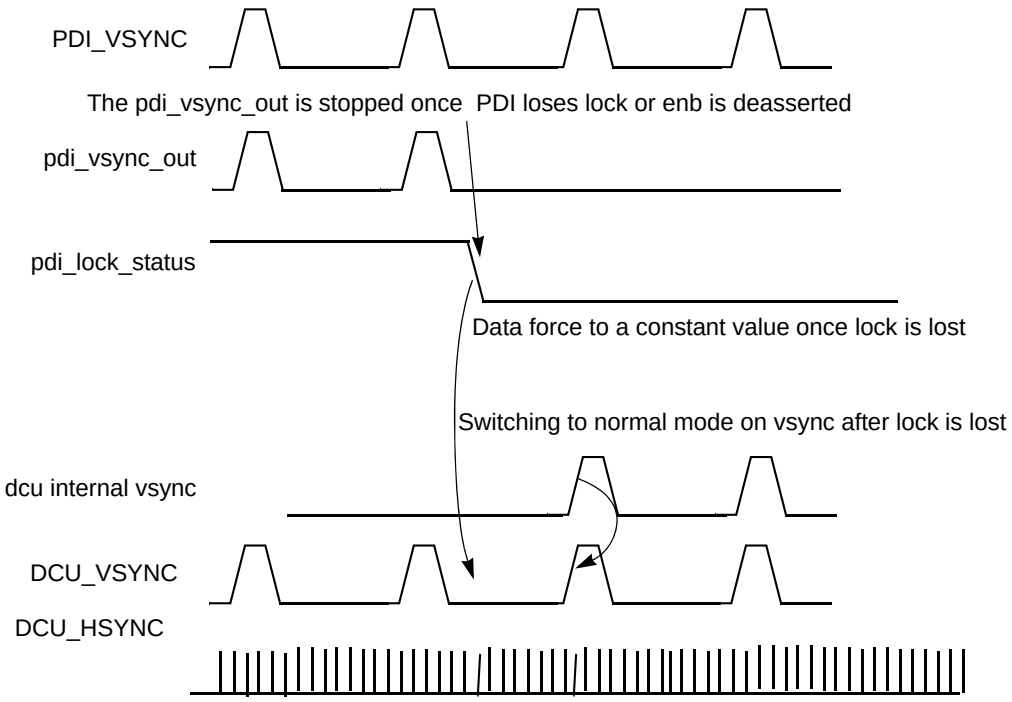



**Figure 12-100. Switch to Normal Mode from PDI mode**

### 12.9.1 Changes in the configuration

Any changes in the RGB format or the synchronization mode configuration requires the PDI system to be in disable mode i.e. PDI_EN should be 0 during this time. The resolution of the screen/layer (PDI or TFT) cannot change on the fly.

#### 12.9.1.1 PDI slave mode

The VSYNC generated by the DCULite block is synchronized to the PDI input VSYNC; the PDI VSYNC resets the internal timing generation unit. HSYNC and VSYNC are generated internally corresponding to the external TFT screen display parameters programmed in the DCULite registers.

In slave mode the PDI and DCULite introduce a fixed latency to the VSYNC and HSYNC timing.

#### 12.9.1.2 PDI sync detection/validation

PDI declares a lock when it has correctly received the continuous number of frames programmed by the application.

PDI declares the sync lost:

- When it receives data enable during the vertical and horizontal blanking period (in case of data enable mode)
- When the incoming HSYNC and VSYNC timing does not match the programmed parameter values in the DCULite. This also means that if any of control signal info lost, then it also declares sync lost.

After PDI_PCLK is lost, the PDI requires 64 DCULite module clock cycles to detect PDI_PCLK again.

PDI does not declare lost sync in case of blanking and ECC errors.

Writes to the input FIFO are stopped as soon as PDI sync is lost aborting transfer of the current frame data. Sync detection works continuously, independent of PDI enable. PDI fires an interrupt on:

- Sync lock is achieved
- Sync is lost
- Activity is detected on hsync
- Activity is detected on vsync
- PCLK Activity detection (it is not generated from the state machine)
- DE Activity detection (it is not generated from the state machine)
- PCLK Activity lost (it is not generated from the state machine)

On receiving a wrong sync pulse, the DCULite stops the HSYNC and VSYNC activity detection and gives an interrupt when it finds sync pulse again. The state machine works for zero values of front and back porches but not for pulse width and screen parameters.

#### 12.9.1.3 Other assumptions

The reset to the PDI clock is synchronized to the peripheral clock. Reset synchronization is done with respect to the PDI clock internally.

The PDI clock should be available at least 10 clock after the last valid data. This corresponds to the delay of the PDI block.

## 12.10 DCULite initialization

The following steps describe a typical approach to initializing the DCULite for use in an application.

1. After reset configure the DCULite peripheral to be active using the mode entry module and configure the DCULite clock source in the MC_CGM. Configure the output ports in the SIUL as required.
2. Configure the timing registers to match the TFT LCD panel in use (see Section 12.4.2, “TFT LCD panel configuration”).
3. Set the background color as required.
4. Load the palette colors into the CLUT memory.
5. Configure the control descriptors for the layers and cursor that are to be used initially.
6. Enable the DCULite in the appropriate mode (DCU_MODE and RASTER_EN bit fields).

## 12.11 Glossary

**Table 12-78. Glossary**

| Term | Description |
|---|---|
| ARGB (also BGRA) | A data format where the pixel values are stored using four components: Alpha, Red, Green and Blue. DCULite supports different variations of this format where different numbers of bits can be used to represent each of the components |
| Component | Part of a pixel that contains a single color (red, green or blue) |
| CLUT | Color Look-up table. The table that contains the palette used by an indexed-color graphic |
| Direct color | The full 24-bit value actually written to a pixel to create a color |
| Frame | The collection of all pixels on a panel |
| Gamut | The set of colors that a panel can display. In most cases a panel cannot display the full gamut of colors visible to the human eye. |
| Indexed color | An index into a table containing direct-colors. Usually smaller in size than the direct color; the DCULite provides 1, 2, 4, and 8 bits per pixel options |
| Palette | The list of colors used by a graphic when an indexed colors format is used. The palette is stored in a color look up table and can be from one color up to the maximum of the size of the CLUT. |
| Panel | A TFT LCD containing an array of colored pixels. |
| Pixel | The basic graphical element on a TFT LCD panel. Can display a range of colors depending on the value of the red, green and blue values written to it. Normally arranged in a rectangular array. |

**Table 12-78. Glossary (continued)**

| Term | Description |
|---|---|
| RGB | A data format where the pixel values are stored using three components: Red, Green and Blue. DCULite supports different variations of this format where different numbers of bits can be used to represent each of the components |
| Vertical blanking period | A time during the TFT LCD panel refresh cycle when no data is being written to the panel |

This page is intentionally left blank.

# Chapter 13
# DRAM Controller (DRAMC)

## 13.1 Introduction

The DDR DRAM controller (DRAMC) is a multi-port DRAM controller (five ports). It supports Mobile-DDR[1], DDR-1, and DDR-2 memories.A block diagram of the DRAMC is given in Figure 13-1.

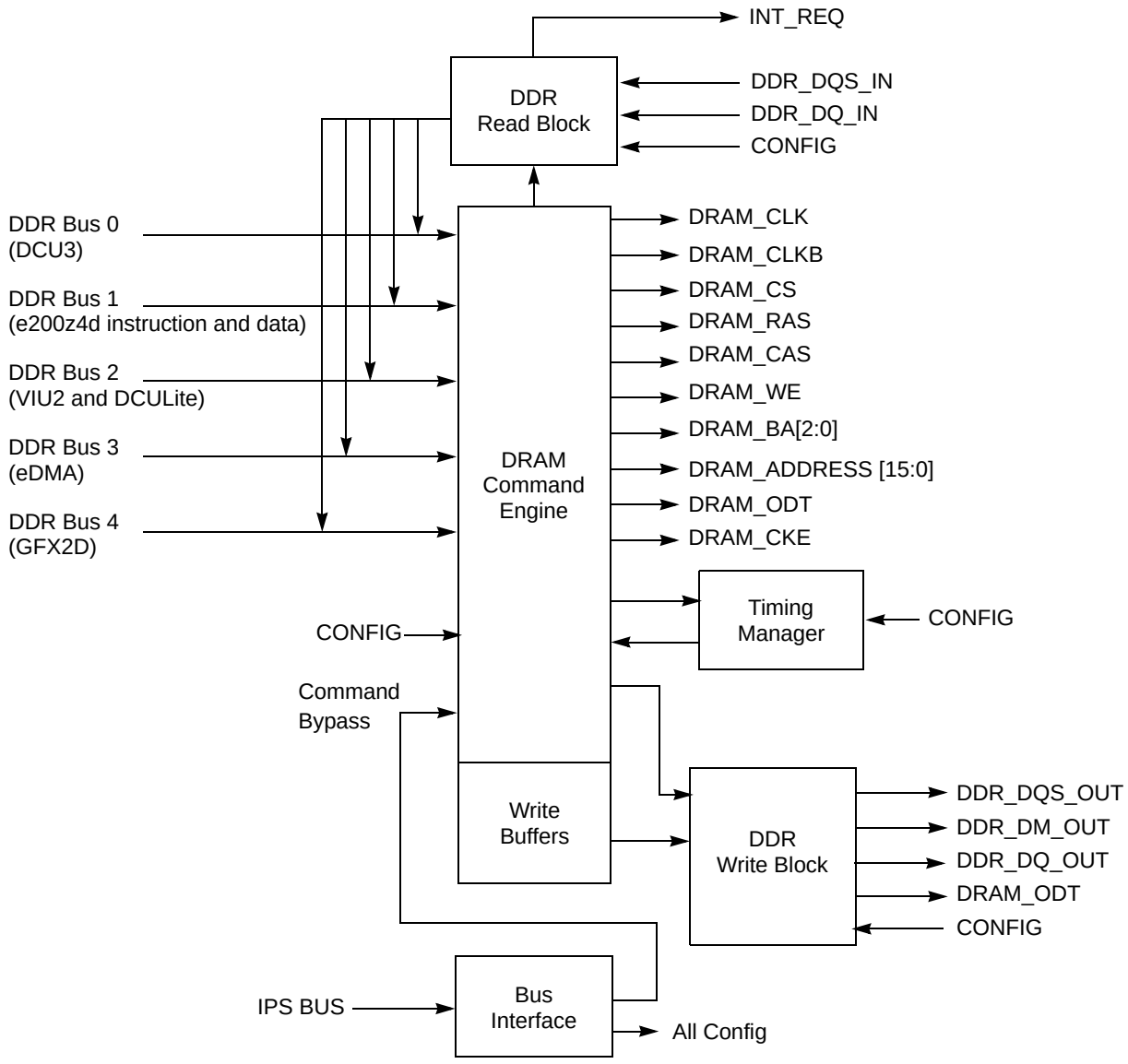



**Figure 13-1. DRAMC block diagram**

1. The JEDEC standard calls these LPDDR. Most DRAM vendors call them Mobile-DDR.

### 13.1.1 Overview

The DRAMC is a multi-port controller that listens to incoming requests on the five incoming buses and decides on each rising clock edge what command needs to be sent to the DRAM. Each incoming bus is a 64-bit bus.The block supports connection of two DRAM ranks (two chip selects) and supports the four major classes of DRAM:

- Mobile-DDR (LPDDR)
- DDR1
- DDR2

It supports these memories in 16-bit or 32-bit wide configurations.

The DRAMC listens to the incoming requests to all the buses in parallel and then sends commands to the DRAM from the highest priority bus at the current time, while the DRAM is ready to receive the command from this particular bus. If the DRAM is blocked because it needs to meet a timing requirement, the controller sends a command from a bus where there is no blockage.

For example, suppose bus one has an incoming request on priority five, and it hits in bank 1 and the page is not open (the bank needs a precharge+activate command before the request can be serviced). Bus two has an incoming request on priority four, it hits in bank two and the correct page is already open. In this case, the DRAMC accepts the bus two request first. While it is reading from the appropriate bank, it issues the active+precharge command for the bus one request. Because the DRAMC sees it can’t issue the read for the bus one request (the bank needs precharge + activate), it takes the bus two request first. Because it can issue the read, the correct page is open. During this, it issues the precharge + activate for the bus 1 request in the background. This request does not suffer from the bus two request being serviced first.

The embedded priority manager determines the relative priority of each bus, and this is used by the DRAMC to determine which requests are more urgent.

## 13.2 Features

- Supports CAS latency of 2, 3, or 4 clock cycles.
- Master buses
  - — Seven incoming masters shared on five master busses
  - — Supports 16-byte and 32-byte bursts
  - — Supports byte enables
  - — Supports 4-bit priority signal for each bus.
- Arbitration protocol
  - — Inside the arbiter block, there are a total of six different arbiters that each take out the highest priority request in a certain class. All the arbiters are DRAM state aware, meaning they disregard requests that cannot be sent to the DRAM because of DRAM timing limitations.
    - Arbiter 1: Looks for highest priority read command
    - Arbiter 2: Looks for highest priority write command
    - Arbiter 3: Looks for highest priority activate-for-read command
    - Arbiter 4: Looks for highest priority activate-for-write command
    - Arbiter 5: Looks for highest priority precharge-for-read command

- Arbiter 6: Looks for highest priority precharge-for-write command

— After the first prioritization, the next round of arbitrating between the different arbiters is done. A fixed-priority schema is followed:
  - Read and write commands have highest priority
  - Activate has next-highest priority
  - Precharge has lowest priority
  - The DRAM is in read or write mode. In read mode, reads have priority over writes. In write mode, writes have priority over read.
  - DRAM only switches from read to write mode or vice-versa if:
    A high-priority write is found, and the write buffer is full.

    A high-priority read is found.

    The device is in read mode, but no more reads pending

    The device is in write mode, but no more writes pending.

- Write buffer contains five 32-byte entries.
- Supports 16- and 32-bit-wide Mobile-DDR/DDR1/DDR2 and DRAM devices
- Controller supports two chip selects, 8 banks per chip select (16 banks total).
- Supports dynamic on-die termination in the host device and in the DRAM.

## 13.3 Memory map and register definition

### 13.3.1 Memory map

**Table 13-1. DRAMC memory map**

| Offset from DRAM_BASE | Register | Access | Reset Value | Location |
|---|---|---|---|---|
| 0x000 | DRAMC_SCR—DRAMC System Configuration Register | R/W | 0x1000_0000 | on page 584 |
| 0x004 | DRAMC_TC0—DRAMC Time Config0 Register | R/W | 0x0000_0000 | on page 588 |
| 0x008 | DRAMC_TC1—DRAMC Time Config1 Register | R/W | 0x0000_0000 | on page 589 |
| 0x00C | DRAMC_TC2—DRAMC Time Config2 Register | R/W | 0x0000_0000 | on page 589 |
| 0x010 | DRAMC_CMD—DRAMC Command Register | R/W | 0x0000_0000 | on page 593 |
| 0x014 | DRAMC_CCMD—DRAMC Compact Command Register | R/W | 0x0000_0000 | on page 593 |
| 0x018–0x034 | Reserved | | | |
| 0x038 | DRAMC_DQS_OC—DQS Config Offset Count Register | R/W | 0x0000_0000 | on page 595 |
| 0x03C | DRAMC_DQS_OT—DQS Config Offset Time Register | R/W | 0x0000_0000 | on page 596 |
| 0x040 | DRAMC_DQS_DS—DQS Delay Status Register | R | 0x0000_0000 | on page 596 |
| 0x044–0x05F | Reserved | | | |
| 0x060 | DRAMC_EXTRA—DRAMC extra attributes | R/W | 0x0000_0000 | on page 597 |
| 0x064–0xFFF | Reserved | | | |

## 13.3.2 Register descriptions

### 13.3.2.1 DRAMC System Configuration Register (DRAMC_SCR)

Address: Base + 0x0000 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RST | CKE | CLK ON | CMD | ROW_SEL | | | | BK_SEL | | | 0 | 0 | 0 | B16 | RDLY[3] |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RDLY[2:0] | | | H_DQSDLY | Q_DQSDLY | WDLY[2:0] | | | EODT | ODT | FIFO_OVP | FIFO_UVP | 0 | 0 | FIFO_OVEN | FIFO_UVEN |
| W | | | | | | | | | | | w1c | w1c | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-2. DRAMC System Configuration Register (DRAMC_SCR)**

**Table 13-2. DRAMC_SCR field descriptions**

| Field | Description |
|---|---|
| RST | DRAMC soft reset. When this bit is 0, the DRAMC is in the reset state. When this bit is 1, the DRAMC is out of reset. The bit controls the reset to the internal state machines. The configuration registers are reset by the resets from the hardware reset block, not by this bit. |
| CKE | Value on the DRAM CKE pin. For functional operation, this needs to be high. During power-down, value can be low. |
| CLKON | When this bit is 1, the DRAM clock is running. When this bit is 0, the DRAM clock is stopped |
| CMD | When this bit is 0, the DRAMC is in normal operation. When this bit is 1, the DRAMC is in command mode and does not respond to requests on the incoming buses. Command mode is used for DRAM initialization and to switch the DRAM into and out of the different power-down and self-refresh modes. |
| ROWSEL<br>BK_SEL | These fields control the multiplexing of the bus address to the DRAM bank and row address. Table 13-5 gives the details. DRAM column address depends on 16-bit mode bit, and relationship is given in Table 13-5. |
| B16 | When this bit is set, the DRAMC assumes a 16-bit wide memory is used. When this bit is cleared, a 32-bit wide memory is assumed.<br>**Note:** This does not configure the pins for 16-bit mode. That must be done in the pin configuration. |
| RDLY[3:0] | This field controls the expected delay between sending a read command to the DRAM and receiving the read data from the DRAM.<br>RDLY, HALF DQS DLY, and QUART DQS delay together to code for $t_{DQSEN}$.<br>The $t_{DQSEN}$ is the delay between the read command and when the internal DQS enable goes high. See Figure 13-3. Timing is internally compensated, and is referred to timing at the device pins. $t_{DQSEN}$ should be selected so the L-H transition of DQS enable is always in the preamble of the DQS input of the READ command. Required $t_{DQSEN}$ value depends on the CAS latency (CL), the distance between the DRAM and the device, and the type of DRAM used. Table 13-3 gives the detail on programming $t_{DQSEN}$. |

**Table 13-2. DRAMC_SCR field descriptions (continued)**

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>H_DQSDLY</td><td>This field is an extra field to control the expected read delay between issuing the read command and getting read data from the DRAM. This field offers 1/2 system bus clock granularity when programming the delay. See description of field RDLY for details.</td></tr>
<tr><td>Q_DQSDLY</td><td>This field is an extra field to control the expected read delay between issuing the read command, and getting read data from the DRAM. This field offers 1/4 system bus clock granularity when programming the delay. See the description of the RDLY bitfield.</td></tr>
<tr><td>WDLY[2:0]</td><td>This field controls the write latency (WL) for write commands.
<table>
<tr><th>WDLY[2:0]</th><th>Write Latency (CSB Clocks)</th></tr>
<tr><td>0b001</td><td>1</td></tr>
<tr><td>0b010</td><td>2</td></tr>
<tr><td>0b011</td><td>3</td></tr>
<tr><td>0b100</td><td>4</td></tr>
</table>
Note: For SDR mode, WDLY must be 001.</td></tr>
<tr><td>EODT</td><td>This bit controls the assertion of the MODT signal when on-die termination (ODT) is used with DDR2 DRAM.<br>• If set to 0, the MODT is asserted two clocks before the write data.<br>• If set to 1, the MODT is asserted three clocks before the write data.<br>It may be necessary, depending on the time required to turn on termination resistors in the memory device, to set EARLY_ODT=1. Doing so will insert an extra clock cycle between non-consecutive back-to-back write cycles but has no other performance impact.</td></tr>
<tr><td>ODT</td><td>This bit controls on-die termination (ODT) in the controller. If this bit is 1, the internal pads generate ODT during read. If the bit is 0, no ODT is provided. The ODT in the DRAM is controlled via the DRAM internal configuration registers. Please consult DRAM data sheet for it.</td></tr>
<tr><td>FIFO_OVP<br>FIFO_UVP<br>FIFO_OVEN<br>FIFO_UVEN</td><td>These bits indicate timing errors and allow the generation of interrupts when these errors occur. The DRAMC has two interrupts: FIFO OV pending and FIFO UV pending. These interrupts are set on overflow or underflow of the FIFO in the read block. When a read command is sent to the DRAM, it is entered into a FIFO. The DRAM is expected to answer by sending back the read data with some up and down edges on the DQS lines (the DQS strobes) used to clock the data. The DRAMC clocks the read data with the DQS strobes supplied by the DRAM and retrieves the read command from the FIFO after receiving the correct number of read strobes. When the read data strobes returned by the DRAM do not match the expectations of the controller, the FIFO may underflow (if too many DQS strobes are coming back from the DRAM) or overflow (if not enough DQS strobes are coming back). These underflows and overflows are the result of problems with the DRAM interface or incorrect parameter settings in the controller or the DRAM. Care has been taken during the design of the DRAMC not to enter a hang-up state when this occurs. However, read data is corrupt and CPU is informed via the FIFO overflow and FIFO underflow interrupts. The issue is also discussed in Section 13.4.7, “Bus Interface.”<br>• FIFO_OV_PENDING and FIFO_UV_PENDING signal to the CPU if an overflow or underflow interrupt is pending. Write “1“ to these bits clears the corresponding interrupt.<br>• FIFO_OV_EN and FIFO_UV_EN bits are interrupt enable bits. If the pending + enable bit is set at the same time, the interrupt is sent to the CPU.</td></tr>
</table>

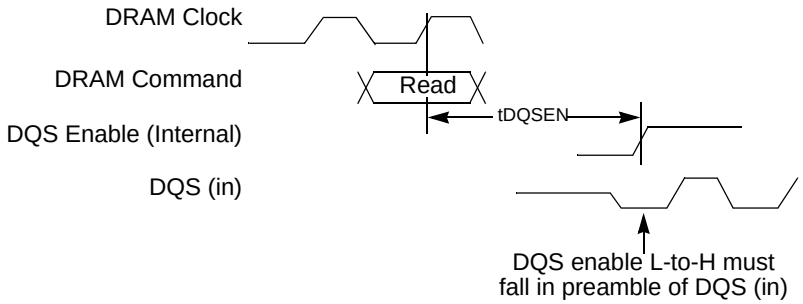



**Figure 13-3. $t_{DQSEN}$**

**Table 13-3. Programming $t_{DQSEN}$**

| {RDLY, H_DQSDLY, Q_DQSDLY} | $t_{DQSEN}$ (system bus clock periods) |
|---|---|
| 1000 0 0 | 0.5 |
| 1000 0 1 | 0.75 |
| 1000 1 0 | 1.0 |
| 1000 1 1 | 1.25 |
| 0100 0 0 | 1.5 |
| 0100 0 1 | 1.75 |
| 0100 1 0 | 2.0 |
| 0100 1 1 | 2.25 |
| 0010 0 0 | 2.5 |
| 0010 0 1 | 2.75 |
| 0010 1 0 | 3.0 |
| 0010 1 1 | 3.25 |
| 0001 0 0 | 3.5 |
| 0001 0 1 | 3.75 |
| 0001 1 0 | 4.0 |
| 0001 1 1 | 4.25 |
| 0000 0 0 | 4.5 |
| 0000 0 1 | 4.75 |
| 0000 1 0 | 5.0 |
| 0000 1 1 | 5.25 |

**Table 13-4. Number of DRAM banks addressed and mapping of address to DRAM bank address**

| BKSEL | Number of banks | DRAM bank address |
|---|---|---|
| 0 | 4 | DRAM_BANK[1:0] = address[11:10] |
| 1 | 4 | DRAM_BANK[1:0] = address[12:11] |

**Table 13-4. Number of DRAM banks addressed and mapping of address to DRAM bank address (continued)**

| BKSEL | Number of banks | DRAM bank address |
|---|---|---|
| 2 | 8 | DRAM_BANK[2:0] = address[13:11] |
| 3 | 4 | DRAM_BANK[1:0] = address[13:12] |
| 4 | 8 | DRAM_BANK[2:0] = address[14:12] |
| 5 | 4 | DRAM_BANK[1:0] = address[14:13] |
| 6 | 8 | DRAM_BANK[2:0] = address[15:13] |
| 7 | 4 | DRAM_BANK[1:0] = address[15:14] |
| 8 | 8 | DRAM_BANK[2:0] = address[16:14] |
| 9 | 4 | DRAM_BANK[1:0] = address[25:24] |
| 10 | 8 | DRAM_BANK[2:0] = address[26:24] |
| 11 | 4 | DRAM_BANK[1:0] = address[26:25] |
| 12 | 8 | DRAM_BANK[2:0] = address[27:25] |
| 13 | 8 | DRAM_BANK[2:0] = address[28:26] |
| 14 | 8 | DRAM_BANK[2:0] = address[29:27] |
| 15 | 8 | DRAM_BANK[2:0] = address[30:28] |

**Table 13-5. Mapping of address to DRAM column address**

| B16 | DRAM column address |
|---|---|
| 0 | DRAM_COLUMN = {address[13:3], 1'b0} |
| 1 | DRAM_COLUMN = {address[12:3], 2'b0} |

## NOTE

In 16-bit mode (16BITMODE = 1), DDR memories with column address line 0 to 7 are not supported.

**Table 13-6. Mapping of address to DRAM row address**

| ROWSEL | DRAM row address |
|---|---|
| 0 | DRAM_ROW[15:0] = address[25:10] |
| 1 | DRAM_ROW[15:0] = address[26:11] |
| 2 | DRAM_ROW[15:0] = address[27:12] |
| 3 | DRAM_ROW[15:0] = address[28:13] |
| 4 | DRAM_ROW[15:0] = address[29:14] |
| 5 | DRAM_ROW[15:0] = address[30:15] |
| 6 | DRAM_ROW[14:0] = address[30:16] |
| 7 | DRAM_ROW[13:0] = address[24: 9] |

**Table 13-7. Mapping of Address to DRAM Chip Select**

| CS_SELECT[1] | DRAM Chip Select |
|---|---|
| 0 | dram_cs_select[2] = 0 |
| 1 | dram_cs_select = address[12] |
| 2 | dram_cs_select = address[13] |
| 3 | dram_cs_select = address[14] |
| 4 | dram_cs_select = address[15] |
| 5 | dram_cs_select = address[16] |
| 6 | dram_cs_select = address[27] |
| 7 | dram_cs_select = address[28] |
| 8 | dram_cs_select = address[29] |
| 9 | dram_cs_select = address[30] |

1 For the field of register 0x60, see Table 13-16.

2 if DRAM_CS_SELECT = 0 → use CS0.
if DRAM_CS_SELECT = 1 → use CS1.

## 13.3.2.2 Timing Configuration

### 13.3.2.2.1 DRAMC Time Configuration Register 0 (DRAMC_TC0)

Address: Base + 0x0004 Access: User read/write

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

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

**Figure 13-4. DRAMC Time Configuration Register 0 (DRAMC_TC0)**

**Table 13-8. DRAMC_TC0 field descriptions**

| Field | Description |
|---|---|
| REFRESH | Refresh interval of the DRAM. Program in this register the number of system bus clocks between any two refresh requests. |
| CMD | Time-out after sending a command to the DRAM in bypass mode. For command sent to the DRAM using the DDR_COMMAND and DDR_COMPACT_COMMAND register, the normal checking of the timing parameters is not done. Instead, any new command to the DRAM is disabled for DRAM_COMMAND_TIME[7:0] dram clock periods. This parameter needs to be programmed for the worst-case time-out. |

**Table 13-8. DRAMC_TC0 field descriptions (continued)**

| Field | Description |
|---|---|
| PRE | Time-out. Any active bank, that has no outstanding requests, is automatically precharged by the DRAMC after this time-out has elapsed since the last access to the bank. This time can be set short, which results in open banks being precharged quite fast to long, which results in open banks left open for a long time. The value is a time count in DRAM clock periods. |

#### 13.3.2.2.2 DRAMC Time Configuration Register 1 (DRAMC_TC1)

Address: Base + 0x0008 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RFC | | | | | | WR1 | | | | | WTR1 | | | | RRD |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RRD | | | | | RC | | | | | | RAS | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-5. DRAMC Time Configuration Register 1 (DRAMC_TC1)**

#### 13.3.2.2.3 DRAMC Time Configuration Register 2 (DRAMC_TC2)

Address: Base + 0x000C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RCD | | | | FAW | | | | RTW1 | | | | CCD | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CCD | RTP | | | | RP | | | | | | RPA | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-6. DRAMC Time Configuration Register 2 (DRAMC_TC2)**

The DDR_TIME_CONFIG1 and DDR_TIME_CONFIG2 registers need to be programmed with the DDR1/DDR2 timing parameters. All times are given in clock cycles.

The timing parameters are conceived so the controller system bus clock cycles match with the JEDEC DDR2 specification. To interface with DDR1 or Mobile-DDR (LPDDR), some timing parameters need not be enforced, or are calculated differently. Refer to the DRAM datasheet to determine their value. The timing parameters need to be programmed in function of this DRAM requirement. Table 13-9 gives the details.

**Table 13-9. Timing parameters**

| Timing parameter | Controls JEDEC parameter (JEDEC spec) | Formulae (all times in system bus clock periods) | Description |
|---|---|---|---|
| RFC | $t_{RFC}$ | RFC = $t_{RFC}$ | REFRESH to ACTIVE or REFRESH to REFRESH command interval. |
| RRD | $t_{RRD}$ | RRD = $t_{RRD}$ | ACTIVE bank A to ACTIVE bank B command. |
| RC | $t_{RC}$ | RC = $t_{RC}$ | ACTIVE to ACTIVE (same bank) command. |
| RAS | $t_{RAS}$ | RAS = $t_{RAS}$ | ACTIVE to PRECHARGE command. |
| RCD | $t_{RCD}$ | RCD = $t_{RCD}$ | ACTIVE to READ or WRITE delay. |
| FAW | $t_{FAW}$ | FAW[1] = $t_{FAW}$ | 4-bank activate period. |
| CCD | $t_{CCD}$ | CCD[2] = max($t_{CCD}$,2) (32-bit mode) max($t_{CCD}$,4)(16-bit mode) | CAS to CAS delay Because time is needed for data to be sent over, this time is minimum two clocks in 32-bit mode and four clocks in 16-bit mode. |
| RTP | $t_{RTP}$ | RTP[3] = max ($t_{RTP}$, 2) (32-bit mode, DDR2) max ($t_{RTP}$, 2) + (16-bit mode, DDR2) | Read to precharge delay. RTP is the read-to-precharge delay and $t_{RTP}$ is the *internal* read-to-precharge delay, hence, the difference for 16-bit mode. Figure 13-7 gives the details. |
| RP | $t_{RP}$ | RP = $t_{RP}$ | Precharge command period. |
| RPA | $t_{RP}$ | RPA[4] = $t_{RP}$ + 1 (8 bank device) RPA = $t_{RP}$ (4 bank device) | Precharge all command period. |
| WR1 | $t_{WR}$ | WR1 = WL + $t_{WR}$ + 2 (32-bit mode) WL + $t_{WR}$ + 4 (16-bit mode) | Write recovery time, measured in clocks between write command and precharge command. For this reason, WL (the write latency) and the length of the actual write (2 or 4) need to be added to $t_{WR}$. Figure 13-8 gives the details. |
| WTR1 | $t_{WTR}$ | WTR1 = WL + $t_{WTR}$ + 2 (32-bit mode) WL + $t_{WTR}$ + 4 (16-bit mode) | Write to read time, measured in clocks between write command and read command. For this reason, WL (the write latency) and the length of the actual write (2 or 4) need to be added to $t_{WTR}$. Figure 13-9 gives the details. |
| RTW1 | — | RTW1 = CL - WL + 2 + $t_{BTA}$ (32-bit) CL - WL + 4 + $t_{BTA}$ (16-bit) | Read-to-write time, measured in clocks between the read and write command. There is no limitation on the DRAM on how to set this parameter. The parameter should be set such that there is no contention on the DQ data bus when switching from read to write. Equation given at left tries to come up with a formulae that defines the minimum value of DRAM_TIME_RTW1 to avoid contention. CL is the cas latency, WL is the write latency, and $t_{BTA}$ is the bus turn-around time. $t_{BTA}$ is the minimum dead time that needs to be put on the bus between the driving the bus and the DRAM driving the bus to take into account the transit delay on the PCB, the pad delay, the DRAM skew, and the on-chip delay. |

**Table 13-9. Timing parameters (continued)**

| Timing parameter | Controls JEDEC parameter (JEDEC spec) | Formulae (all times in system bus clock periods) | Description |
|---|---|---|---|
| CCD_OTHER[5] | — | CCD_OTHER = max($t_{CCD}$,2) (32-bit mode) +1 max($t_{CCD}$,4)(16-bit mode) +1 | CAS to CAS delay from one chip select to the other Because time is needed for data to be sent over, this time is minimum 2 clocks in 32-bit mode, 4 clocks in 16-bit mode |
| WTR1_OTHER[5] | | WTR1_OTHER = WL - RL + 2 + 2(32-bit mode) WL - RL + 4 + 2(16-bit mode) | Write to read time for write and read happening on different chip selects, *measured in clocks between write command and read command*. For this reason, WL (the write latency) and the length of the actual write (2 or 4) need to be added to $t_{WTR}$. Figure 13-9 gives the details. |

1 For DRAMs that do not need this check, set equal to 4 × $t_{RRD}$

2 For DDR1 and Mobile-DDR $t_{CCD}$ is 2 for 32-bit operation, 4 for 16-bit operation. For SDR $t_{CCD}$ is 4 for 32-bit operation, 8 for 16-bit operation.

3 For DDR1 and Mobile-DDR mode, $t_{RTP}$ is not explicitly given in SDR, DDR1 and Mobile-DDR mode. For DDR1 and Mobile-DDR, it is equal to 4 for 16-bit mode, equal to 2 for 32-bit mode. For SDR, it is equal to 8 for 16-bit mode, equal to 4 for 32-bit mode.

4 This timing parameter controls precharge all command period duration. The equations shown are the JEDEC definition of the $t_{RPA}$. Some DRAM vendors do not follow JEDEC on this, and list $t_{RPA}$ directly. In this case, set *DRAM_TIME_RPA* = $t_{RPA}$.

5 Field is part of register 0x60, dram_extra_attributes

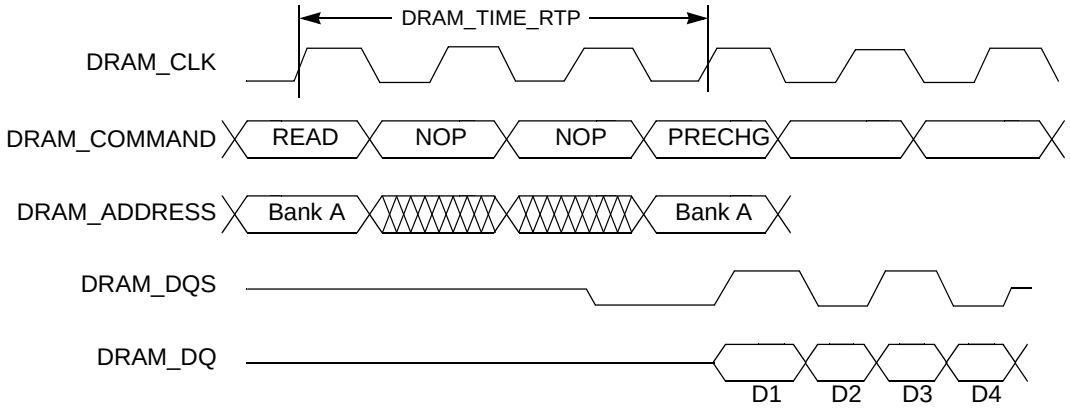



**Figure 13-7. Read to Precharge Timing Diagram**

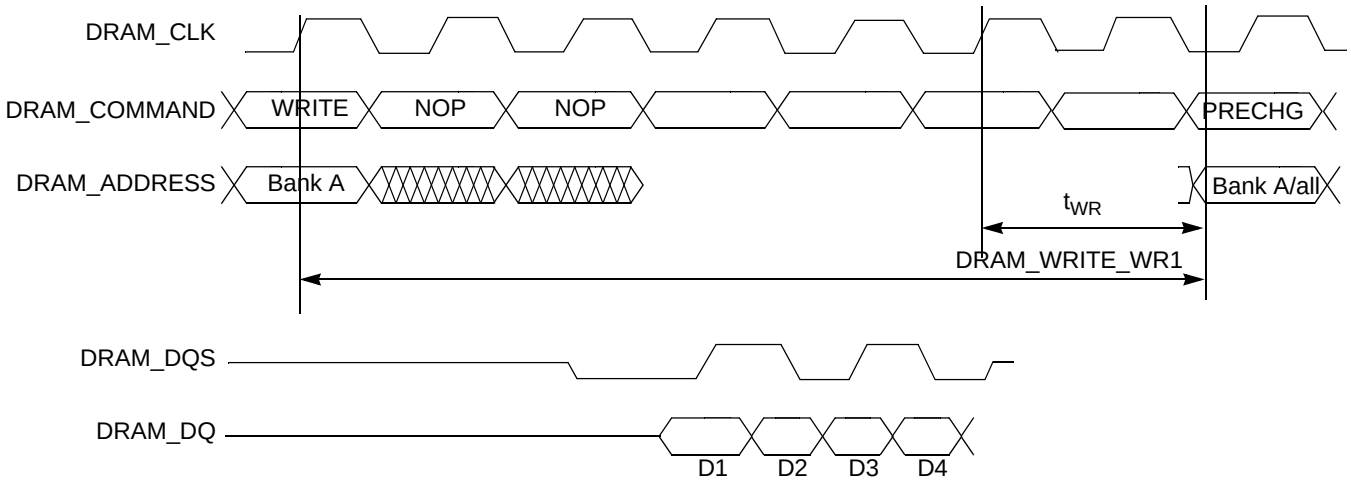



**Figure 13-8. Write to precharge timing diagram**

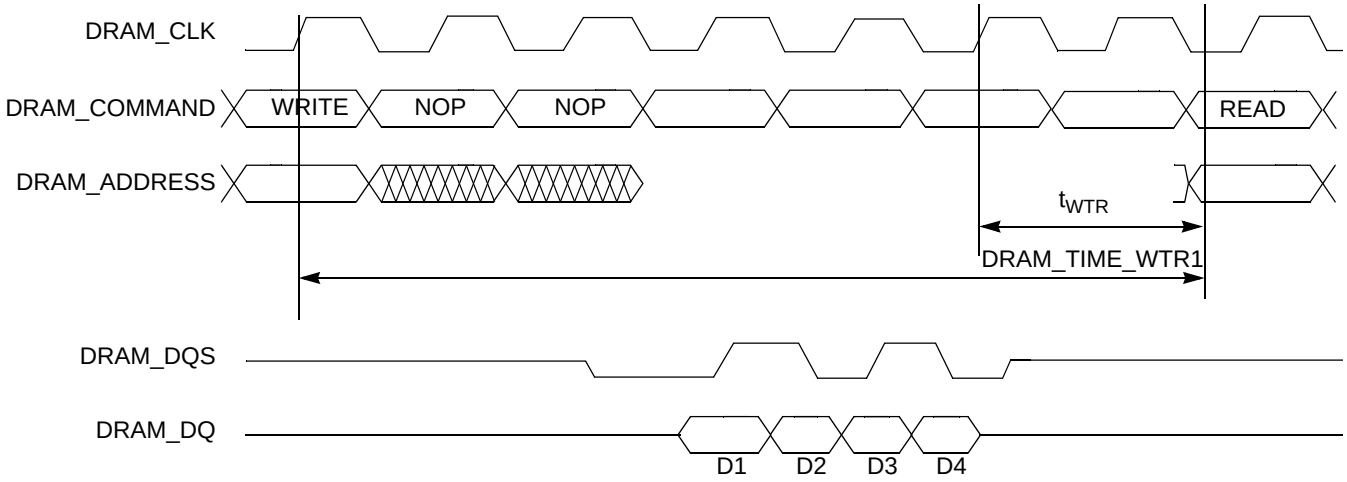



**Figure 13-9. Write to read timing diagram**

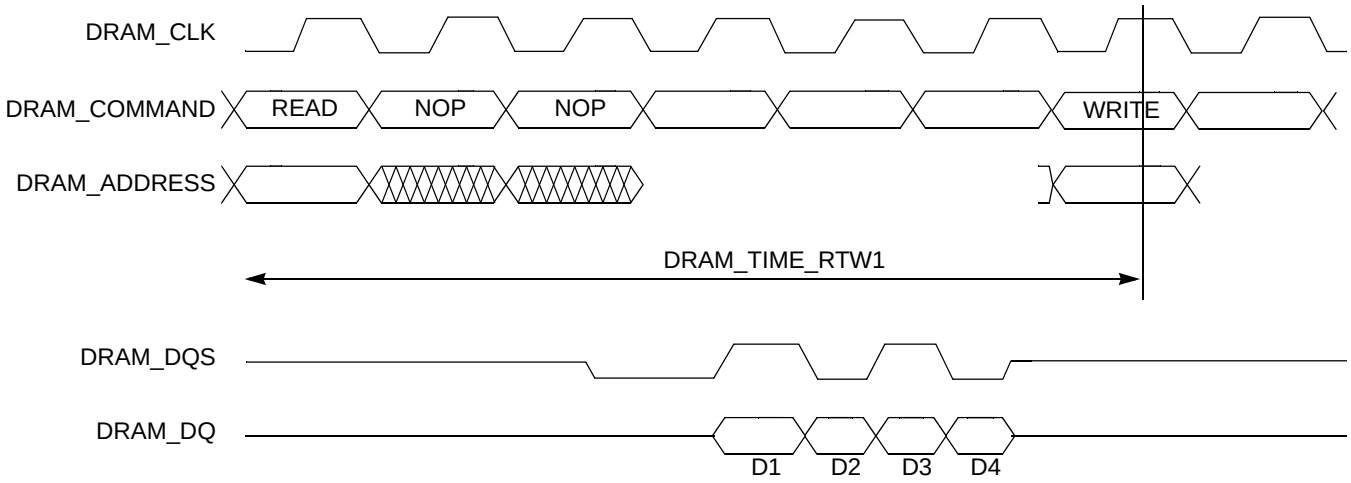



**Figure 13-10. Read to write timing diagram**

### 13.3.2.3 DRAMC Command Register (DRAMC_CMD)

The DRAM command register (DRAMC_CMD) gives the option to send commands directly to the DRAM. This register only operates when the command mode bit (CMD, bit 3) is set in the DRAMC_SCR register.

Address: Base + 0x0010 Access: User read/write

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

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

**Figure 13-11. DRAMC Command Register (DRAMC_CMD)**

**Table 13-10. DRAMC_CMD field descriptions**

| Field | Description |
|---|---|
| CMD[23:0] | When the DRAMC_SCR[CMD] bit is set, the value written to bits [23:0] of this register is output on the DRAM address group with following mapping:<br>• DRAM_ADDRESS[14:0] = CMD[14:0]<br>• DRAM_ADDRESS[15] = 0<br>• if(DRAM_COMMAND[15] == 1) turn off CKE DRAM attribute bit[1]<br>• DRAM_BA[2:0] = CMD[18:16]<br>• DRAM_WEB = CMD[19]<br>• DRAM_CAS = CMD[20]<br>• DRAM_RAS = CMD[21]<br>• DRAM_CS0 = CMD[22]<br>• DRAM_CS1 = CMD[23]<br>**Note:** The intended use of the command interface is to initialize the DRAM and to put the DRAM into or out of the self-refresh and power-down modes. |

1 CKE is turned off on the same clock cycle as when the requested command is being sent to the DRAM.

### 13.3.2.4 DRAMC Compact Command Register (DRAMC_CCMD)

The DRAMC compact command register (DRAMC_CCMD) gives the option to send commands to the DRAM using only a 16-bit interface.

Address: Base + 0x0014 Access: User write-only

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

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

**Figure 13-12. Compact Command Register (DRAMC_CCMD)**

**Table 13-11. DRAMC_CCMD field descriptions**

| Field | Description |
|---|---|
| CMD | The compact command register gives the option to send commands to the DRAM using 16-bit writes. See Table 13-12. |

**Table 13-12. DRAMC_CCMD register options**

| CMD[15:14] | Description |
|---|---|
| 00 | Write DRAM attributes and wait (wait time = wait time till next command) Wait is executed after writing attributes.<br>• CKE = CMD[13]<br>• Self Ref En = CMD[12]<br>• CLK ON = CMD[11]<br>• CMD MODE = CMD[10]<br>• If (CMD[7] == 1'b1)<br>• Wait time = (CMD[6:0] × 512) dram clock periods<br>Or<br>• Wait time = (CMD[6:0] × 32) dram clock periods |
| 01 | DRAM command<br>• DRAM_CS = CMD[12]<br>• DRAM_RAS = CMD[11]<br>• DRAM_CAS = CMD[10]<br>• DRAM_WEB = CMD[9]<br>• DRAM_BA[2:0] = CMD[8:6]<br>• DRAM_ADDRESS[10] = CMD[5]<br>• if(CMD[4] == 1'b1) turn off CKE DRAM attribute bit[1] |
| 1x | DRAM set mode registers<br>• DRAM_CS = 0<br>• DRAM_RAS = 0<br>• DRAM_CAS = 0<br>• DRAM_WEB = 0<br>• DRAM_ADDRESS[13] = 0<br>• DRAM_BA[2] = 0<br>• DRAM_ADDRESS[12:0] = CMD[12:0]<br>• DRAM_ADDRESS[14:13] = CMD[14:13] |

[1] CKE is turned off the clock cycle when sending the requested command to the DRAM.

The DRAMC_CCMD register's main purpose is to be written during enter/exit of self-refresh (the auto-sequencer).

The compact command register allows three types of actions to be executed:

- Write DRAM attributes and wait. Wait is executed after updating the DRAM attributes. It is possible to update the CKE bit, the self-refresh enable, the CLK configuration (on/off), and the CMD mode setting.

  If, during the time the wait is executed, another command is written to the Compact Command register, this write is delayed until the wait is over.

  During this time, the peripheral bus and all buses connected to it block and are not able to process any other read or write.
- Write a command to DRAM without controlling the address. In this mode, it is possible to send refresh, activate, and precharge commands to the DRAM
- Write a DRAM mode register.

### 13.3.2.5 DQS Config Offset Count Register (DRAMC_DQS_OC)

Address: Base + 0x0038 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | OFF3 | | | | | | | | OFF2 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | OFF1 | | | | | | | | OFF0 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-13. DQS Config Offset Count Register (DRAMC_DQS_OC)**

**Table 13-13. DRAMC_DQS_OC Register field descriptions**

| Field | Description |
|---|---|
| OFF[3:0] | There is a separate field for each DQS input to the controller. These fields code for an offset counted in elemental gate delay increments applied to each DQS slave. The number is a two-complement number that can be positive and negative.<br>This register can be used to compensate systematic delay shift in the DRAMC due to processing. Leave this register all-zero, unless Freescale issues a report giving a different value. |

### 13.3.2.6 DQS Config Offset Time Register (DRAMC_DQS_OT)

Address: Base + 0x003C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | OFF3 | | | | | | 0 | 0 | OFF2 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | OFF1 | | | | | | 0 | 0 | OFF0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-14. DQS Config Offset Time Register (DRAMC_DQS_OT)**

**Table 13-14. DRAMC_DQS_OT Register field descriptions**

| Field | Description |
|---|---|
| OFF[3:0] | There is a separate field for each DQS input to the controller. These fields code for an offset counted in time units.<br>This register can be used to advance or delay the read strobe. Negative values advance the read strobe, positive values retard the read strobe.<br>Time delay coded = [field value (two's complement)] × Tdram-clock/128.<br>The applied offset range for a 200 MHz clock is approximately ± 290 ps. |

### 13.3.2.7 DQS Delay Status (DRAMC_DQS_DS)

Address: Base + 0x0040 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | MC2 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | MC1 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-15. DQS Delay Status (DRAMC_DQS_DS)**

**Table 13-15. DRAMC_DQS_DS Register field descriptions**

| Field | Description |
|---|---|
| MC2 | Delay count output by the controller for the first DQS master to code for 1/4 CSB clock delay. |
| MC1 | Delay count output by the controller for the second DQS master to code for 1/4 CSB clock delay. |

### 13.3.2.8 DRAMC Extra Attributes (DRAMC_EXTRA)

Address: Base + 0x0060 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CCD_OTHER | | | | WTR1_OTHER | | | | CONFIG_SDR | CONFIG_CAS3 | CONFIG_A15 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 13-16. DRAMC Extra Attributes (DRAMC_EXTRA)**

**Table 13-16. DRAMC_EXTRA Register field descriptions**

| Field | Description |
|---|---|
| CCD_OTHER | For description, refer to Table 13-9. |
| WTR1_OTHER | For description, refer to Table 13-9. |
| CONFIG_SDR | 0 DDR mode.<br>1 Reserved. |
| CONFIG_CAS3 | Reserved. |
| CONFIG_A15 | 0 A15 pin acts as A15.<br>1 A15 pin acts as Chip Select 1. |
| CS_SELECT | For description, refer to Table 13-7. |

## 13.4 Functional description

The DRAMC is a multi-port DRAM controller. It listens to incoming requests on multiple buses and decides on each rising clock edge what command needs to be sent to the DRAM.

A block diagram is given in Figure 13-1. The major blocks of the DRAMC are described in the following sections.

### 13.4.1 Interfacing with the DRAM

#### 13.4.1.1 Connecting the DRAM

- 32-bit DRAM systems need to be connected to all DQ, DM, DQS lines.
- 16-bit DRAM systems need to connected to the low order bits of the data bus. (DQ[15:0], DQS[1:0], and DM[1:0])
- Row/column address pins need to be connected starting with bit [0] and ending with the highest order DRAM bit. Leave MSB's unconnected if the DRAM has fewer address pins than the controller.

- DRAM bank address pins need to be connected starting with bit [0] and ending with the highest order bank address bit. Leave MSB unconnected if the DRAM has fewer bank address pins than the controller.
- Electrical characteristics for the DRAM interface signals are configured using the Pad Configuration Registers (PCRs) described in Chapter 43, “System Integration Unit Lite (SIUL).”

### 13.4.2 Programming DRAM Device Internal Configuration Register

- Set burst type to sequential.
- Burst length is always 16-byte. Means 4-beat bursts in a 32-bit system, 8-beat burst in a 16-bit system.
- Set CAS latency to lowest value DRAM can tolerate at intended speed, and then set write latency and read latency accordingly.
- Set posted CAS additive latency to 0.
- Controller never uses auto-precharge on read or write.
- Configure DQS operation for single-ended operation.
- RTT and output drive strength configuration depends on electrical characteristics.

### 13.4.3 DRAM Command Engine

This block decides what command to send to the DRAMC next. There are four different commands that can be sent to the DRAM to service incoming requests from the five incoming buses.

- Precharge
- Activate
- Read
- Write

On every rising clock edge, the DRAM command engine first determines using parallel logic what is highest priority pending precharge, activate, read and write command. Next, it decides which of these commands to send to the DRAM.

The arbiters that make the decisions about what command to send next to the DRAM are aware of the current state the DRAM is in. When arbitrating a command on the DRAM bus, the following information is processed:

- For each bank, if it is precharged or not
- For each incoming request, if it hits in an already active bank or not
- For each bank, if the DRAM currently can accept a precharge command to it
- For each bank, if the DRAM currently can accept an activate command to it
- For each bank, if the DRAM currently can accept a read command to it
- For each bank, if the DRAM currently can accept a write command to it

The logic keeping track of what is currently possible on each of the banks is not in the DRAM command engine. It is part of the timing manager, whose task is to signal to the DRAM command engine that commands are currently possible.

### 13.4.4 Write Buffer

All incoming writes are sent first to the write buffer, part of the command engine. Writes are sent to the DRAM in background, whenever possible. The DRAM tries to postpone the writes until there are no further outstanding read requests. However, when the write buffer is full, or when there is a new request for an address already inside the write buffer, the DRAMC writes the content of the write buffer to the DRAM.

### 13.4.5 Timing Manager

The timing manager consists of a bank of counters. These counters keeps track of all DRAM timing parameters and signals to the DRAM command engine when a precharge, activate, read or write command is possible. This information is supplied to the DRAM command engine for each bank separately.

All timing parameters are programmable in software.

### 13.4.6 DRAM Read Block and DRAM Write Block

Sending a read or write command to the DRAM is a two-step process. First, the command is sent, which is done by the command engine. After some clock cycles, the data must follow.

Manipulating the read data is done by the read block. For every read command sent to the DRAM, the command engine informs the read block. Upon receiving the read command, the read block delays this to account for DRAM pipelining. Then, it receives the correct amount of data from the DRAM DQ inputs and forwards this data to the correct bus.

Manipulating the write data is done by the write block. It works the same way as the read block. The command engine informs the write block of a pending write. Upon receiving the command, the write block delays this to account for DRAM pipelining. Then, it receives the relevant data from the write buffer and transmits this to the DRAM.

### 13.4.7 Bus Interface

The bus interface accepts a slave peripheral bus. The bus interface fulfills several functions:

- It contains all configuration registers
- It contains logic to send an error interrupt to the processor. The error interrupt is active when the FIFO overflow or FIFO underflow error condition and corresponding interrupt enable in register DRAMC_SCR is set. The register summary is given in Table 13-1.

  The FIFO overflow and underflow flags are tied to a FIFO that keeps track of the number of DQS strobes the DRAM is expected to produce. If a read command is sent to the DRAM, the DRAM is expected to answer after producing the read data on its DQ outputs, with some edges on its DQS output used by the controller to clock the read data. If the DRAMC produces the read strobes at an

incorrect time, or produces not enough or too many read strobes, the DRAMC may detect some error conditions because they result in an overflow or underflow of the FIFO that keeps track of the number of outstanding DQS pulses. These bits do not detect timing configuration errors. Underflows and overflows signaled by the read FIFO point to following possible error sources:

— Incorrect configuration of the DRAM. Burst length set incorrectly

— Incorrect configuration of the DRAMC.

  – Incorrect RDLY

  – Incorrect H_DQSDLY

  – Incorrect Q_DQSDLY

  – Incorrect DRAM timing parameters or mis-match between various settings.

— Problems with the electrical connections between the DRAMC and the DRAM

- It contains a bypass path to send commands to the DRAM. This is because the DRAMC contains no logic to take care of DRAM initialization, programming the mode registers, or putting the DRAM into or out of the sleep and standby modes like self-refresh. Essentially, these functions are made available over the peripheral bus. To program the mode registers, the DRAMC needs to be put in a bypass mode, where incoming requests are not serviced. In this bypass mode, commands are sent from the peripheral interface directly to the DRAM to program the mode registers or to put the DRAM into or out of sleep mode.
- During bypass mode, all reads and writes are blocked. Refresh keeps running, but can be separately disabled.

# Chapter 14
# DRAMC Priority Manager

## 14.1 Introduction

The multi-port DRAM controller (DRAMC) has built-in arbiters that use a 4-bit priority signal. The DRAMC will try to service the request with the highest priority first. As a result of this, the DRAMC itself has no built-in way to fairly split the DRAM bandwidth over the different masters, taking into account their requirements.

The task of setting the priorities is done by this block, the priority manager. It will in a dynamic way set the priorities of the 7 DRAM busses in such a way that bandwidth is divided fairly, and that each master gets its fair share of the bandwidth.

The priority manager does this in a dynamic way. If a channel is not serviced during some time, it will increase its priority. If a channel is serviced, it will decrease the priority.

Operation of the priority manager can be configured under software control.

A block diagram of the priority manager is given in Figure 14-1. It accepts the request and ACK (acknowledge) signals for all 7 DRAM busses, and produces the priority signals for the 7 busses.

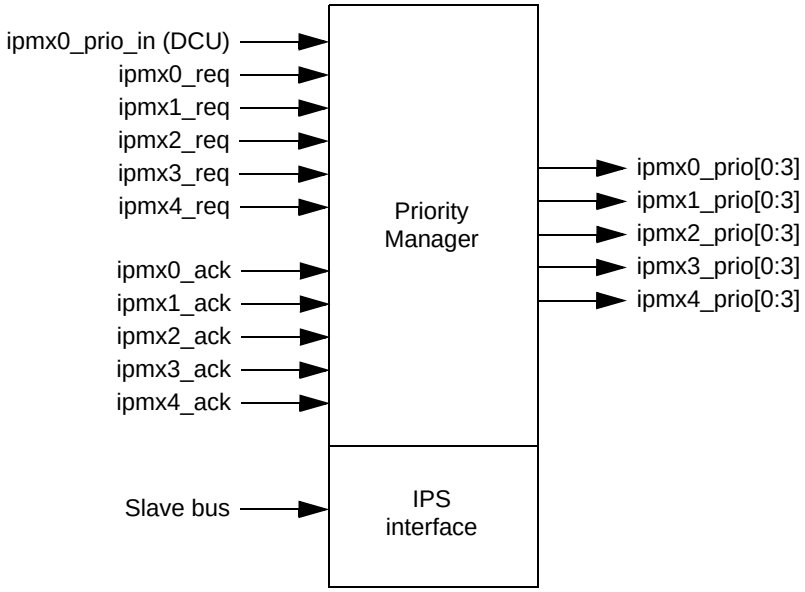



**Figure 14-1. Priority manager block diagram**

The priority manager uses an ACK-based schema, meaning that the priority is dependent on how many times for the last N requests accepted by the DRAMC, the current own bus won the request. The algorithm is given in Figure 14-2. A running average counter keeps track of how many ACKs on the memory busses out of the last N ACKs have been for the own bus, this number is then put in a look-up table that is configurable by writing some config registers, and the output of the look-up table then is the priority for the next request on this bus.

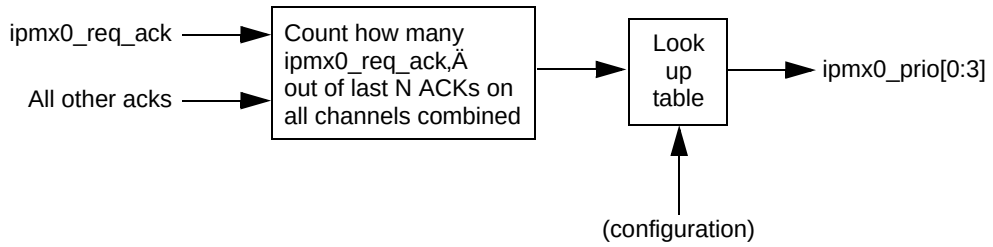



**Figure 14-2. Priority manager algorithm**

This priority schema is versatile, as programming the look-up table allows controlling relative priority to other channels, and the average share of the bandwidth the current master will get. The priority schema introduces fairness, as the look-up table can be programmed to reduce the priority of a bus that has "won" a lot of requests, and increase the priority of a bus that lost a lot of requests.

## 14.2 Features

- Dynamic priority calculation based on ACK-in history
  - — Fully programmable using look-up table.
    - – Can be configured for high or low latency and high or low bandwidth
    - – Separate control over average latency and average bandwidth
    - – Versatile so it can mimic the CSB arbitration schema.
  - — Fairness guaranteed by reducing priority of channels that receive a lot of grants, and increasing priority of channels that are denied the bus often.
  - — Repeat transfer built into the DRAMC. Priority manager can set the maximum repeat count by controlling when lowest priority occurs.
- Feed-through mode where DCU priority is controlled directly by the DCU

## 14.3 Detailed signal description

**Table 14-1. Detailed Signal Description**

| Signal | I/O | Description |
|---|---|---|
| ipg_clk | I | system clock |
| ipg_hard_reset_b | I | system reset |
| ips_clk_en | I | clock enable to signal edges on ips_clk |
| ipmx0_req | I | DRAMC request for bus 0 |
| ipmx1_req | I | DRAMC request for bus 1 |
| ipmx2_req | I | DRAMC request for bus 2 |
| ipmx3_req | I | DRAMC request for bus 3 |
| ipmx4_req | I | DRAMC request for bus 4 |
| ipmx0_req_ack | I | DRAMC request acknowledge for bus 0 |
| ipmx1_req_ack | I | DRAMC request acknowledge for bus 1 |
| ipmx2_req_ack | I | DRAMC request acknowledge for bus 2 |
| ipmx3_req_ack | I | DRAMC request acknowledge for bus 3 |
| ipmx4_req_ack | I | DRAMC request acknowledge for bus 4 |
| ipmx0_prio_in[0:3] | I | Incoming priority signal for DCU (bus 0) |
| ipmx0_prio[0:3] | O | Outgoing priority signal to DRAMC for bus 0 |
| ipmx1_prio[0:3] | O | Outgoing priority signal to DRAMC for bus 1 |
| ipmx2_prio[0:3] | O | Outgoing priority signal to DRAMC for bus 2 |
| ipmx3_prio[0:3] | O | Outgoing priority signal to DRAMC for bus 3 |
| ipmx4_prio[0:3] | O | Outgoing priority signal to DRAMC for bus 4 |
| ips_module_en | I | Slave bus module enable |
| ips_addr[0:8] | I | Slave bus address |
| ips_rwb | I | Slave bus read/$\overline{\text{write}}$ signal |
| ips_byte_en[0:3] | I | Slave bus byte enables |
| ips_wdata[0:31] | I | Slave bus write data bus |
| ips_rdata[0:31] | O | Slave bus read data bus |

## 14.4 Memory map and register definition

### 14.4.1 Memory map

**Table 14-2. Priority manager memory map**

| Offset or Address | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0x80 | prioman_config1 (CFG1) | R/W | 0x0007_7777 | on page 606 |
| 0x84 | prioman_config2 (CFG2) | R/W | 0x0000_0011 | on page 606 |
| 0x88 | hiprio_config (HPCFG) | R/W | 0x0 | on page 608 |
| 0x8C | LUT 0 main upper (MLUTU0) | R/W | 0x01111_1222 | on page 609 |
| 0x90 | LUT 1 main upper (MLUTU1) | R/W | 0x01111_1222 | on page 609 |
| 0x94 | LUT 2 main upper (MLUTU2) | R/W | 0x01111_1222 | on page 609 |
| 0x98 | LUT 3 main upper (MLUTU3) | R/W | 0x01111_1222 | on page 609 |
| 0x9C | LUT 4 main upper (MLUTU4) | R/W | 0x01111_1222 | on page 609 |
| 0xA0 | LUT 0 main lower (MLUTL0) | R/W | 0x2334_567A | on page 610 |
| 0xA4 | LUT 1 main lower (MLUTL1) | R/W | 0x2334_567A | on page 610 |
| 0xA8 | LUT 2 main lower (MLUTL2) | R/W | 0x2334_567A | on page 610 |
| 0xAC | LUT 3 main lower (MLUTL3) | R/W | 0x2334_567A | on page 610 |
| 0xB0 | LUT 4 main lower (MLUTL4) | R/W | 0x2334_567A | on page 610 |
| 0xB4 | LUT 0 alternate upper (ALUTU0) | R/W | 0x0 | on page 611 |
| 0xB8 | LUT 1 alternate upper (ALUTU1) | R/W | 0x0 | on page 611 |
| 0xBC | LUT 2 alternate upper (ALUTU2) | R/W | 0x0 | on page 611 |
| 0xC0 | LUT 3 alternate upper (ALUTU3) | R/W | 0x0 | on page 611 |
| 0xC4 | LUT 4 alternate upper (ALUTU4) | R/W | 0x0 | on page 611 |
| 0xC8 | LUT 0 alternate lower (ALUTL0) | R/W | 0x0 | on page 612 |
| 0xCC | LUT 1 alternate lower (ALUTL1) | R/W | 0x0 | on page 612 |
| 0xD0 | LUT 2 alternate lower (ALUTL2) | R/W | 0x0 | on page 612 |
| 0xD4 | LUT 3 alternate lower (ALUTL3) | R/W | 0x0 | on page 612 |
| 0xD8 | LUT 4 alternate lower (ALUTL4) | R/W | 0x0 | on page 612 |
| 0xDC | Performance monitor config (PMCFG) | R/W | 0x0 | on page 613 |
| 0xE0 | Event time counter (EVTMR) | R/W | 0x0 | on page 614 |
| 0xE4 | Event time preset (EVPRST) | R/W | 0x0 | on page 614 |
| 0xE8 | Performance monitor 1 address low (PM1L) | R/W | 0x0 | on page 615 |
| 0xEC | Performance monitor 2 address low (PM2L) | R/W | 0x0 | on page 615 |
| 0xF0 | Performance monitor 1 address high (PM1H) | R/W | 0x0 | on page 615 |

**Table 14-2. Priority manager memory map (continued)**

| Offset or Address | Register | Access | Reset value | Location |
|---|---|---|---|---|
| 0xF4 | Performance monitor 2 address high (PM2H) | R/W | 0x0 | on page 615 |
| 0xF8–0xFC | Reserved | | | |
| 0x100 | Performance monitor 1 read counter (PM1CNTR) | R | 0x0 | on page 616 |
| 0x104 | Performance monitor 2 read counter (PM2CNTR) | R | 0x0 | on page 616 |
| 0x108 | Performance monitor 1 write counter (PM1CNTW) | R | 0x0 | on page 616 |
| 0x10C | Performance monitor 2 write counter (PM2CNTW) | R | 0x0 | on page 616 |
| 0x110 | Granted ACK counter 0 (GACKCTR0) | R | | on page 616 |
| 0x114 | Granted ACK counter 1 (GACKCTR1) | R | | on page 616 |
| 0x118 | Granted ACK counter 2 (GACKCTR2) | R | | on page 616 |
| 0x11C | Granted ACK counter 3 (GACKCTR3) | R | | on page 616 |
| 0x120 | Granted ACK counter 4 (GACKCTR4) | R | | on page 616 |
| 0x124 | Cumulative wait counter 0 (CUMWCTR0) | R | | on page 616 |
| 0x128 | Cumulative wait counter 1 (CUMWCTR1) | R | | on page 616 |
| 0x12C | Cumulative wait counter 2 (CUMWCTR2) | R | | on page 616 |
| 0x130 | Cumulative wait counter 3 (CUMWCTR3) | R | | on page 616 |
| 0x134 | Cumulative wait counter 4 (CUMWCTR4) | R | | on page 616 |
| 0x138 | Summed priority counter 0 (SPRIOCTR0) | R | | on page 616 |
| 0x13C | Summed priority counter 1 (SPRIOCTR1) | R | | on page 616 |
| 0x140 | Summed priority counter 2 (SPRIOCTR2) | R | | on page 616 |
| 0x144 | Summed priority counter 3 (SPRIOCTR3) | R | | on page 616 |
| 0x148 | Summed priority counter 4 (SPRIOCTR4) | R | | on page 616 |

## 14.4.2 Register descriptions

### 14.4.2.1 prioman_config1, prioman_config2 (CFG1, CFG2)

Offset: 0x80 read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | LUTSEL4 | | LUTSEL3 | | LUTSEL2 | | LUTSEL1 | | LUTSEL0 | | ACKCNT4 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ACKCNT3 | | | | ACKCNT2 | | | | ACKCNT1 | | | | ACKCNT0 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

= Unimplemented or Reserved

**Figure 14-3. prioman_config1 register (CFG1)**

Offset: 0x84 read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | CONG | DCUOVR | ACKSEL4 | ACKSEL3 | ACKSEL2 | ACKSEL1 | ACKSEL0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

= Unimplemented or Reserved

**Figure 14-4. prioman_config2 register (CFG2)**

**Table 14-3. CFG1 and CFG2 field descriptions**

| Field | Description |
|---|---|
| CONG | 1: Congested flag is set<br>0: Congested flag is cleared |
| DCUOVR | The effect of this bit would only take place if DCU is configured to work in DDR Mode, that is, DDR_MODE bit in the DCU_MODE register is set to 1.<br>1: Priority for channel 0 taken from DCU directly.<br>0: DCU priority follows normal schema. |

**Table 14-3. CFG1 and CFG2 field descriptions (continued)**

| Field | Description |
|---|---|
| ACKSEL4<br>ACKSEL2<br>ACKSEL2<br>ACKSEL1<br>ACKSEL0 | There is one of these bits for each priority manager channel. They determine what happens if the current channel is not requesting<br>1: If current channel is not requesting, every ACK for other channel is treated like a ACK for the current channel. Will regulate default priority to quite low value<br>0: No special overrule |
| LUTSEL4<br>LUTSEL3<br>LUTSEL2<br>LUTSEL1<br>LUTSEL0 | Selectors between primary and secondary Look-Up table configuration register<br>0: Select main Look-up table configuration register<br>1: Select alternate Look-up table configuration register<br>2: Select alternate Look-Up table configuration register if congested flag is set[1]<br>3: Select alternate look-up table configuration register if DCU incoming priority bit 3 is high. |
| ACKCNT4<br>ACKCNT3<br>ACKCNT2<br>ACKCNT1<br>ACKCNT0 | configuration fields, one for every channel, that determines over how many requests the number of ACKs for the self channel is counted.[2]<br>0 : 1<br>1 : 2<br>2 : 3<br>3 : 4<br>4 : 6<br>5 : 8<br>6 : 12<br>7 : 16<br>8 : 24<br>9 : 32<br>10 : 48<br>11 : 63 |

1 Congested flag is explained in Section 14.4.2.2, “HPCFG

2 Look-up table input is running average of the number of ACKs for the self channel counted over the grant total of the last N ACKs. ack_count[2:0] controls the parameter N.

### 14.4.2.2 HPCFG

Offset: 0x88 read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SETCGLVL | | | | | | | | CLRCGLVL | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

= Unimplemented or Reserved

**Figure 14-5. HPCFG register**

This register controls the hiprio detection logic. The hiprio detection logic detect what percentage of the requests ACKed by the DRAMC are ACK-ed with a priority larger than 8.

**Table 14-4. HPCFG field descriptions**

| Field | Description |
|---|---|
| AVGHIPR | Average number of hi priority requests to DRAM, coded between values 0x1000 and 0x0000<br>0x1000: 100% high-priority requests<br>0x0000: 0% high-priority requests |
| FLTRBAND | This setting controls the averaging time of the filter used for average_hipriority[12:0]<br>0: time constant W0 = 8 ACKs, K = 0.125<br>1: time constant W0 = 16 ACKs, K = 0.0625<br>2: time constant W0 = 32 ACKs, K = 0.0312<br>3: time constant W0 = 64 ACKs, K = 0.0156<br>4: time constant W0 = 128 ACKs, K = 0.0078<br>5: time constant W0 = 256 ACKs, K = 0.0039<br>6: time constant W0 = 512 ACKs, K = 0.0020<br>7: time constant W0 = 1024 ACKs, K = 0.0010 |
| SETCGLVL | if(average_hipriority[4:12] > SETCGLVL) -> set the congested flag |
| CLRCGLVL | if(average_hipriority[4:12] < CLRCGLVL) -> clear the congested flag |

### 14.4.2.3 Lookup table main upper registers (MLUTU0–MLUTU4)

Offsets: 0x8C (MLUTO0)
0x90 (MLUTU1)
0x94 (MLUTU2)
0x98 (MLUTU3)
0x9C (MLUTU4)

read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRIO15 | | | | PRIO14 | | | | PRIO13 | | | | PRIO12 | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRIO11 | | | | PRIO10 | | | | PRIO9 | | | | PRIO8 | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

**Figure 14-6. MLUTU registers**

These registers contain the upper 8 entries of the look-up tables for channels 0 to 4, main table. All registers contain identical fields. The content of these registers if the “main” table is enabled for the particular LUT.

**Table 14-5. MLUTU field descriptions**

| Field | Description |
|---|---|
| PRIO15 | Priority setting if 15 or more ACK’s for own channel counted |
| PRIO14 | Priority setting if 14 ACK’s for own channel counted |
| PRIO13 | Priority setting if 13 ACK’s for own channel counted |
| PRIO12 | Priority setting if 12 ACK’s for own channel counted |
| PRIO11 | Priority setting if 11 ACK’s for own channel counted |
| PRIO10 | Priority setting if 10 ACK’s for own channel counted |
| PRIO9 | Priority setting if 9 ACK’s for own channel counted |
| PRIO8 | Priority setting if 8 ACK’s for own channel counted |

### 14.4.2.4 Lookup table main lower registers (MLUTL0–MLUTL4)

Offsets: 0xA0 (MLUTL0)
0xA4 (MLUTL1)
0xA8 (MLUTL2)
0xAC (MLUTL3)
0xB0 (MLUTL4)

read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRIO7 | | | | PRIO6 | | | | PRIO5 | | | | PRIO4 | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRIO3 | | | | PRIO2 | | | | PRIO1 | | | | PRIO0 | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

**Figure 14-7. MLUTL registers**

These registers contain the upper 8 entries of the look-up tables for channels 0 to 4, main table. All registers contain identical fields. The content of these registers if the “main” table is enabled for the particular LUT.

**Table 14-6. MLUTL field descriptions**

| Field | Description |
|---|---|
| PRIO7 | Priority setting if 7 ACK’s for own channel counted |
| PRIO6 | Priority setting if 6 ACK’s for own channel counted |
| PRIO5 | Priority setting if 5 ACK’s for own channel counted |
| PRIO4 | Priority setting if 4 ACK’s for own channel counted |
| PRIO3 | Priority setting if 3 ACK’s for own channel counted |
| PRIO2 | Priority setting if 2 ACK’s for own channel counted |
| PRIO1 | Priority setting if 1 ACK for own channel counted |
| PRIO0 | Priority setting if 0 ACK’s for own channel counted |

### 14.4.2.5 Lookup table alternate upper registers (ALUTU0–ALUTU4)

Offsets: 0xB4 (ALUTU0)
0xB8 (ALUTU1)
0xBC (ALUTU2)
0xC0 (ALUTU3)
0xC4 (ALUTU4)

read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRIO15 | | | | PRIO14 | | | | PRIO13 | | | | PRIO12 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRIO11 | | | | PRIO10 | | | | PRIO9 | | | | PRIO8 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-8. ALUTU registers**

These registers contain the upper 8 entries of the look-up tables for channels 0 to 4, main table. All registers contain identical fields. The content of these registers if the “alternate” table is enabled for the particular LUT.

**Table 14-7. ALUTU field descriptions**

| Field | Description |
|---|---|
| PRIO15 | Priority setting if 15 or more ACK’s for own channel counted |
| PRIO14 | Priority setting if 14 ACK’s for own channel counted |
| PRIO13 | Priority setting if 13 ACK’s for own channel counted |
| PRIO12 | Priority setting if 12 ACK’s for own channel counted |
| PRIO11 | Priority setting if 11 ACK’s for own channel counted |
| PRIO10 | Priority setting if 10 ACK’s for own channel counted |
| PRIO9 | Priority setting if 9 ACK’s for own channel counted |
| PRIO8 | Priority setting if 8 ACK’s for own channel counted |

### 14.4.2.6 Lookup table alternate lower registers (ALUTL0–ALUTL4)

Offsets: 0xC8 (ALUTL0)
0xCC (ALUTL1)
0xD0 (ALUTL2)
0xD4 (ALUTL3)
0xD8 (ALUTL4)

read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRIO7 | | | | PRIO6 | | | | PRIO5 | | | | PRIO4 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRIO3 | | | | PRIO2 | | | | PRIO1 | | | | PRIO0 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-9. ALUTL registers**

These registers contain the upper 8 entries of the look-up tables for channels 0 to 4, alternate table. All registers contain identical fields. The content of these registers if the “alternate” table is enabled for the particular LUT.

**Table 14-8. ALUTL field descriptions**

| Field | Description |
|---|---|
| PRIO7 | Priority setting if 7 ACK’s for own channel counted |
| PRIO6 | Priority setting if 6 ACK’s for own channel counted |
| PRIO5 | Priority setting if 5 ACK’s for own channel counted |
| PRIO4 | Priority setting if 4 ACK’s for own channel counted |
| PRIO3 | Priority setting if 3 ACK’s for own channel counted |
| PRIO2 | Priority setting if 2 ACK’s for own channel counted |
| PRIO1 | Priority setting if 1 ACK for own channel counted |
| PRIO0 | Priority setting if 0 ACK’s for own channel counted |

### 14.4.2.7 Performance monitor config register (PMCFG)

Offset: 0xDC read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INT | | INTEN | DMAREQ | DMARSTP | EVTTIM | | | | | | | | | | |
| W | | CLRINT | | | | | EVTTIMTRG | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | LUTSEL4 | | LUTSEL3 | | LUTSEL2 | | LUTSEL1 | | LUTSEL0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-10. Performance monitor config register (PMCFG)**

**Table 14-9. PMCFG field descriptions**

| Field | Description |
|---|---|
| INT | (Read-only)(sticky bit) - Interrupt pending register - Set when interrupt is made pending |
| CLRINT | (write only) - Writing this bit ‘1’ clears bit ‘int’ to zero. |
| INTEN | Interrupt enable - When this bit is ‘1’ and bit “int” is ‘1’, the processor gets an interrupt request |
| DMAREQ | (read-only) - DMA request - Set when “event counter time” reaches terminal count.<br>Cleared when first counter register is read. |
| DMARSTP[1] | (read/write) - When this bit is ‘1’, the DMA request is cleared, and cannot get set.<br>When this bit is ‘0’, the DMAreq functions as expected. |
| EVTTIM | 1: Event counter free run - After reaching terminal count, the event time counter is reloaded from *event time preset*f, and a new cycle starts.<br>0 : Event counter single-shot - After reaching terminal count, the event time counter stays at 0, and is not reloaded. |
| EVTTIMTRG | (write-only bit) - Writing to this bit causes all count registers to be transferred to the buffer registers, and subsequent be cleared. It causes the Event Counter to be reloaded from the *event time preset* register. No interrupt or DMA request is generated on writing this register, but both will be generated when the *event time counter* register reaches terminal count. |
| LUTSEL4<br>LUTSEL3<br>LUTSEL2<br>LUTSEL1<br>LUTSEL0 | Selectors between primary and secondary Look-Up table configuration register<br>These selectors determine which LUT is used for the “summed priority counters”. The priorities entered into these counters may depend on a different LUT than the priorities sent to the DRAMC.<br>0: Select main Look-up table configuration register<br>1: Select alternate Look-up table configuration register<br>2: Select alternate Look-Up table configuration register if congested flag is set[2]<br>3: Select alternate look-up table configuration register if DCU incoming priority bit 3 is high |

1 This bit should be set as long as the DMA channel is not configured to handle the request. After configuring the DMA, clear the bit, and data will start to be transferred on every time tick.

2 Congested flag is explained in section “congestion detector” on page 626.

### 14.4.2.8 Event Time Timer (EVTMR)

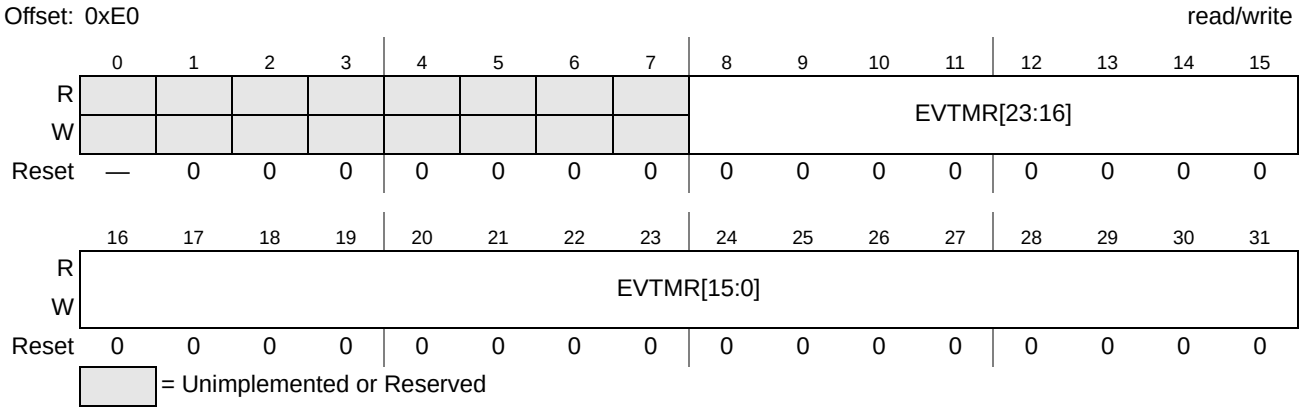



**Figure 14-11. Event Time Counter Register (EVTMR)**

This 32-bit register has only one field - the 24-bit event time counter. The counter decrements to 0, on reaching 0, the interrupt and the DMA request are made pending. On reaching 0, the counter will re-load from *event count preset* register if the bit *eventCountFreeRun* is set in the perfmon_config register.

On reaching terminal count, all performance monitor count registers are loaded in the performance monitor buffer registers, and cleared.

The register is read/write.

### 14.4.2.9 Event Time Preset (EVPRST)

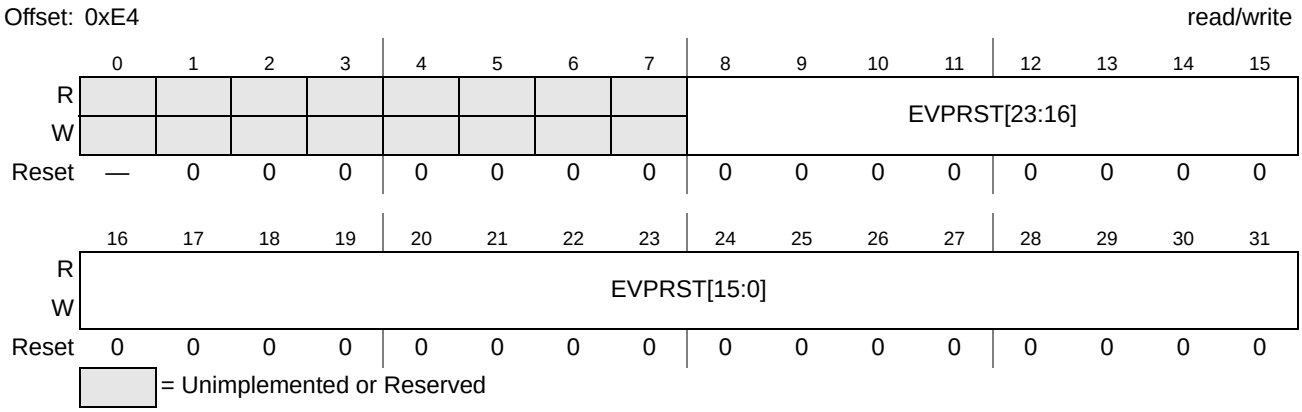



**Figure 14-12. Event Time Preset Register (EVPRST)**

The event_time_preset register contains the 24-bit preset value to be loaded into the event time counter register in case this preloads.

### 14.4.2.10 Performance monitor address registers

Offsets: 0xE8 (PM1L)
0xEC (PM2L)
0xF0 (PM1H)
0xF4 (PM2H)

read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ADDR[31:16] | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ADDR [15:5] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-13. Performance Monitor Address Registers**

These registers determine if a CPU (e200) access hits in the performance monitor 1 or performance monitor 2 address space.

1. If ((e200 CPU address >= performance monitor 1 address low) &&
   (e200 CPU address < performance monitor 1 address hi))
   - Increment *performance monitor 1 read counter* on reads
   - Increment *performance monitor 1 write counter* on writes.
2. If ((e200 CPU address >= performance monitor 2address low) &&
   (e200 CPU address < performance monitor 3address hi))
   - Increment *performance monitor 2 read counter* on reads
   - Increment *performance monitor 2 write counter* on writes.

### 14.4.2.11 Counter registers

Offset: 0x100 (PM1CNTR) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-14. Performance Monitor 1 Read Counter (PM1CNTR)**

Offset: 0x104 (PM2CNTR) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-15. Performance Monitor 2 Read Counter (PN2CNTR)**

Offset: 0x108 (PM1CNTW) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-16. Performance Monitor 1 Write Counter (PM1CNTW)**

Offset: 0x10C (PM2CNTW) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-17. Performance Monitor 2 Write Counter (PM2CNTW)**

Offset: 0x110 (GACKCTR0) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPRIOCTR2[7:0] | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-18. Granted ACK Counter 0 (GACKCTR0)**

Offset: 0x114 (GACKCTR1) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPRIOCTRL2[15:8] | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-19. Granted ACK Counter 1 (GACKCTR1)**

Offset: 0x118 (GACKCTR2) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPRIOCTR2[23:16] | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-20. Granted ACK Counter 2 (GACKCTR2)**

Offset: 0x11C (GACKCTR3) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPRIOCTR3[7:0] | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-21. Granted ACK Counter 3 (GACKCTR3)**

Offset: 0x120 (GACKCTR4) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPRIOCTR3[15:8] | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-22. Granted ACK Counter 4 (GACKCTR4)**

Offset: 0x124 (CUMWCTR0) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPRIOCTR3[23:16] | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-23. Cumulative Wait Counter 0 (CUMWCTR0)**

Offset: 0x128 (CUMWCTR1) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPRIOCTR4[7:0] | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-24. Cumulative Wait Counter 1 (CUMWCTR1)**

Offset: 0x12C (CUMWCTR2) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPRIOCTR4[15:8] | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-25. Cumulative Wait Counter 2 (CUMWCTR2)**

Offset: 0x130 (CUMWCTR3) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPRIOCTR4[23:16] | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-26. Cumulative Wait Counter 3 (CUMWCTR3)**

Offset: 0x134 (CUMWCTR4) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-27. Cumulative Wait Counter 4 (CUMWCTR4)**

Offset: 0x138 (SPRIOCTR0) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-28. Summed Priority Counter 0 (SPRIOCTR0)**

Offset: 0x13C (SPRIOCTR1) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-29. Summed Priority Counter 1 (SPRIOCTR1)**

Offset: 0x140 (SPRIOCTR2) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-30. Summed Priority Counter 2 (SPRIOCTR2)**

Offset: 0x144 (SPRIOCTR3) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-31. Summed Priority Counter 3 (SPRIOCTR3)**

Offset: 0x148 (SPRIOCTR4) read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | CNTR[23:16] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTR[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-32. Summed Priority Counter 4 (SPRIOCTR4)**

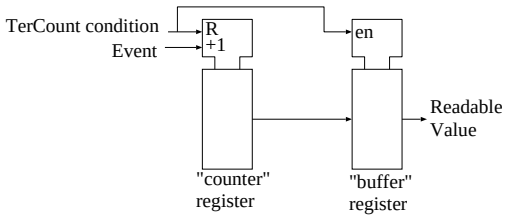
The counter registers contain 19 different 24-bit counter values. All these counter values count certain events. XXX gives the details on the nature of the event. Counter values *Summed Priority Counter 2*, *Summed Priority Counter 3* and *Summed Priority Counter 4* are available in 2 sets of registers. Firstly, they are available in the registers with the same name, but they are also available in a set of 3 other registers (*granted ACK counter* or *cumulative wait counter* registers). The multiple-mapping of the 3 upper *Summed Priority Counter* registers is done to allow easy and compact DMA transfer to memory. Because of the multiple mapping, all 19 count values can be transferred to memory with a 64-byte DMA transfer starting at address 0x100. The multiple mapping allows the DMA to get all information with a 64-byte transfer, but some decompression is needed on decoding the data, while the CPU can read the 19 registers, and just mask out the upper 8 bits to get relevant information.

**Table 14-10. Monitor Counter Descriptions**

| Field | Description |
|---|---|
| performance monitor 1 read counter | Every time the PowerPC performs a read access, whose address is higher or equal than the *performance monitor 1 address low*, and lower than the *performance monitor 1 address high*, this counter is incremented[1]. |
| performance monitor 2 read counter | Every time the PowerPC performs a read access, whose address is higher or equal than the *performance monitor 2 address low*, and lower than the *performance monitor 2 address high*, this counter is incremented[1]. |
| performance monitor 1 write counter | Every time the PowerPC performs a write access, whose address is higher or equal than the *performance monitor 1 address low*, and lower than the *performance monitor 1 address high*, this counter is incremented[1]. |
| performance monitor 2 write counter | Every time the PowerPC performs a write access, whose address is higher or equal than the *performance monitor 2 address low*, and lower than the *performance monitor 2 address high*, this counter is incremented[1]. |
| granted ACK counter 0 | Every time the DRAMC grants a request for channel 0, this counter is incremented[1]. |
| granted ACK counter 1 | Every time the DRAMC grants a request for channel 1, this counter is incremented[1]. |
| granted ACK counter 2 | Every time the DRAMC grants a request for channel 2, this counter is incremented[1]. |
| granted ACK counter 3 | Every time the DRAMC grants a request for channel 3, this counter is incremented[1]. |
| granted ACK counter 4 | Every time the DRAMC grants a request for channel 4, this counter is incremented[1]. |
| cumulative wait counter 0 | Every time there is a request pending to the DRAMC for channel 0, and its not granted in the current cycle, this counter is incremented[1]. |

**Table 14-10. Monitor Counter Descriptions (continued)**

| Field | Description |
|---|---|
| cumulative wait counter 1 | Every time there is a request pending to the DRAMC for channel 1, and its not granted in the current cycle, this counter is incremented[1]. |
| cumulative wait counter 2 | Every time there is a request pending to the DRAMC for channel 2, and its not granted in the current cycle, this counter is incremented[1]. |
| cumulative wait counter 3 | Every time there is a request pending to the DRAMC for channel 3, and its not granted in the current cycle, this counter is incremented[1]. |
| cumulative wait counter 4 | Every time there is a request pending to the DRAMC for channel 4, and its not granted in the current cycle, this counter is incremented[1]. |
| summed priority counter 0 | Every time a request is granted by the DRAMC for channel 0, a priority code[2] is added to this counter[1]. |
| summed priority counter 1 | Every time a request is granted by the DRAMC for channel 1, a priority code[2] is added to this counter[1]. |
| summed priority counter 2 | Every time a request is granted by the DRAMC for channel 2, a priority code[2] is added to this counter[1]. |
| summed priority counter 3 (SPRIOCTR3) | Every time a request is granted by the DRAMC for channel 3, a priority code[2] is added to this counter[1]. This counter shares a register location with granted ACK counters 3 and 4 and cumulative wait counter 0. |
| summed priority counter 4 (SPRIOCTR4) | Every time a request is granted by the DRAMC for channel 4, a priority code[2] is added to this counter[1]. This counter shares a register location with cumulative wait counters 1, 2 and 3. |




[1]All counters in this table are double-buffered. Figure at the left gives the details. There are always 2 registers associated with every counter. The first register is the “counter”. It counts the events mentioned in the table. When the trigger condition occurs- the *time event counter* reaching terminal count -, the “counter” register is transferred to the “buffer” register, and the “counter” register is cleared. When accessing the register, its always the buffer register value that is returned.

[2] The priority code added may or may not be the same as the priority code the request used on the DRAMC. The codes will be equal if the *LUT sel* field for the channel is the same in registers *prioman_config* and *perfmon_config*. If the *LUT sel* fields differ, the field in *prioman_config* is used to calculate the channel priority code on the DRAM, and the field in *perfmon_config* is used to calculate the priority code added to this register.
The possibility to use unequal LUT sel fields, makes it possible to use the “main” look-up tables for DRAM priority programming, and the “alternate” look-up tables for performance monitoring. Making the look-up tables independent, opens a wide possibility in what can be monitored.

## 14.5 Functional description

The priority manager calculates the outgoing priority for all 5 channels of DRAMC. The priority of any channel at a given time, is a function of the request granting history of the DRAMC. A granted request is called an ACK, so this schema is called an ACK-based schema, because the priority is determined by the history of which channels have been ACK-ed in the past and when.

The block diagram in Figure 14-2 makes this dependency clear: Priority manager determines priority on the basis of the outstanding requests, and the requests granted (the ACK’s) by the DRAMC.

The priority manager calculates the priorities in a *dynamic* way. This means, a priority is never constant, but changes over time, even when the request is not serviced. As a request ages while its not being serviced, its priority will escalate to higher level, and as the level increases, it will eventually be serviced.

The DRAMC has a built-in preference to offer repeat for any incoming read request. The repeat goes on as long as the requesting channel keeps requesting, and its priority is greater than 0. When the outgoing priority for any channel is 0, the DRAMC will no longer service or repeat the request. This feature allows the priority manager to control the maximum repeat count for any incoming channel.

### 14.5.1 Description of operation — overview

Priority calculation for all channels is independent. So, there is no direct cross-dependency of the priority of one channel on the priority of another channel. The algorithm looks at the last N arbitration cycles on the bus. N is a programmable number, set by fields *ack_count* in register *prio_man*, described in *Figure 14-3*. For the last N arbitration cycles, the number of times the own channel won the bus, is summed up, and saturated to a maximum of 15. This number of 0 to 15 is then input into the applicable look-up table, and the value for the particular number, is the priority code going to the DRAMC. If e.g. N is set to 16, and the own channel was granted the bus 4 times in the last 16 bus grant, the index into the look-up table is 4, and the field *prio4* of the relevant look-up table will be the priority going to the DRAMC.

There are 2 look-up tables for every channel. The “main” look-up table, and the “alternate” look-up table. The algorithm may switch between both, depending on some settings.

- The “default” look-up table is the “main”. However, the “alternate” will be used if
- The particular channel has been configured to look at the DCU incoming priority, and the DCU incoming priority is 8 or higher.
- The particular channel has been configured to look at the congestion monitor, and this block indicates the multi-port DRAM is congested.

### 14.5.2 Description of operation — block diagram

A block diagram of the block is given in Figure 14-33.

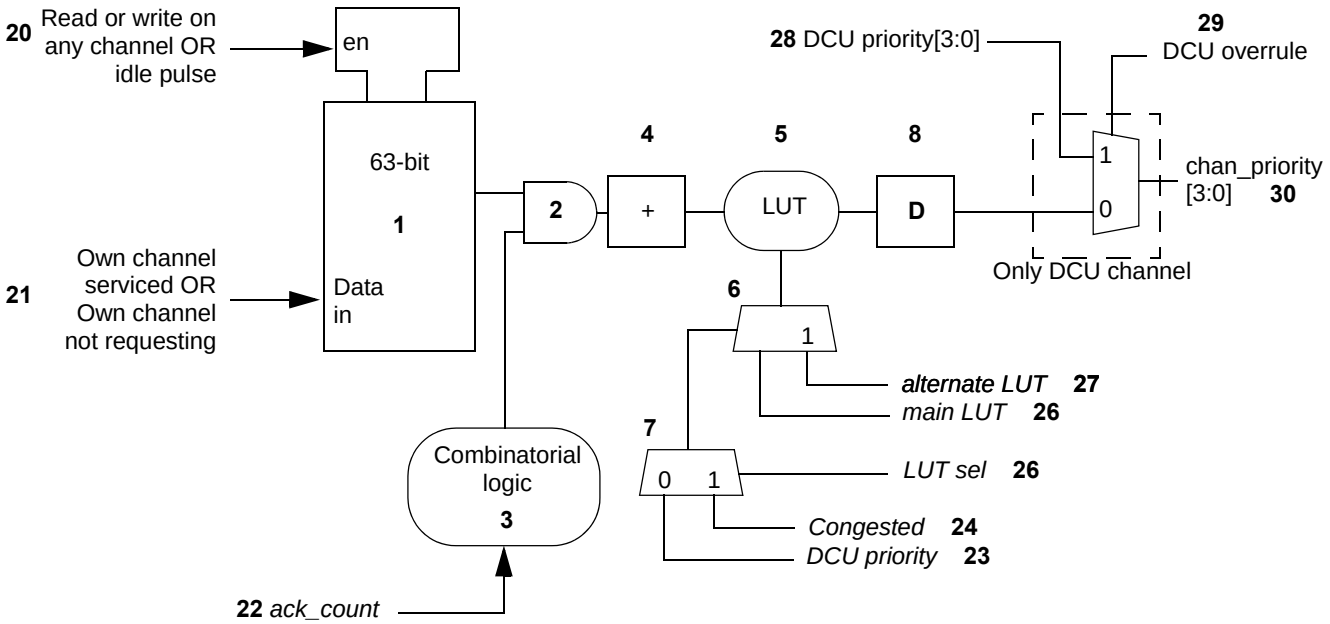



**Figure 14-33. Priority channel block diagram**

Shift register **1** shifts in information of the recent ACK's. Its a 63-stage shift register, so it contains information on the last 63 bus cycles of the DRAMC.

The shift register is shifted any time a read or write request has been granted to the DRAM. (An ACK to the requesting bus), or when there is an idle_pulse. An idle pulse is generated every time the DRAM is idle for 4 consecutive clock cycles. Idle means none of the 5 incoming busses is making a request.

The shift data in is the corrected ACK for the self channel. If the shift register shifts because the current cycle is granted to the "self" channel, a '1' is shifted in, if not a '0' is shifted in. It always occurs like this when the own channel is requesting access. However, if the own channel is not requesting access, depending on control bit *ack_sel*, a '1' or a '0' is shifted in. If ack_sel is '1', a '1' will be shifted in all the time when the self channel is not requesting, and there is an ACK on any other channel, or there is an idle pulse. If ack_sel is '0', zeros are shifted in.

The correction for the non-requesting channel allows the user to steer the "default" priority, i.e., the priority that the channel will get, when it has not been requesting for some time. If ack_sel is set '1', the default priority will be low. This setting is appropriate for peripherals with (large) FIFO's. When they are not requesting, the FIFO is quite full, and when they do get on the bus, its OK for them to start with low priority, and escalate to higher after some time.

Setting ack_sel to '0' is appropriate for peripherals that desire high priority. The e200z4d and GFX2D GPU are in this case. When they are not on the bus, its because they find the instruction or data that they need in the processor caches, so they don't request. When the cache misses, the request comes on the bus, and need to be serviced fast. Therefore, ack_sel is set '0', the default priority will be high, and servicing fast. If e200z4d and/or GFX2D get on the bus a lot (e.g. due to a lot of cache swapping), the priority manager will detect this, and degrade their priorities over time, so the other masters still get their fair share of bus bandwidth.

The output of the shift register ANDED in **2** to look at only the last N ACK's. Combinatorial logic **3** decodes the AND-ing code from register field *ack_count*. The number of ones after the AND-ing is added up in adder **4**, and saturated, so that the result out of adder **4** is a number in the range of 0 to 15. This number is input in the look-up table **5**. Table look-up content is taken for e.g. channel 1 from register *LUT 1 main[63:0]* or *LUT 1 alternate[63:0]*. Because of the 64-bit nature of the registers, actually 4 32-bit registers are involved, whose description is given in *Figure 14-6*, *Figure 14-7*, *Figure 14-8* and *Figure 14-9*.

Selection on whether to use the "main" or the "alternate" register, is done via MUX **6**. The MUX condition has 2 possible sources again, selected by MUX **7**, by means of control bit *LUT sel*, described in register *prioman_config*, with details in *Figure 14-3*.

If *LUT sel* is '1', alternate table is selected when the multi-port controller is congested. If *LUT sel* is '0', alternate table is selected when DCU incoming priority is higher than 8.

Pipeline register **8** is present purely for implementation reasons. It has no algorithmic function.

For the DCU, an additional bypass mux **9** is present. It overrules the prioman logic, and inserts the incoming DCU priority in the output, if control bit *DCU overrule* is set. This bit is present in register *prioman_config*, with details in *Figure 14-3*.

## 14.5.3 Congestion detector

The congestion detectors purpose is to detect when the DRAMC is congested. Congestion is assumed if the share of the requests with priorities equal or greater than 8 is more than a certain percentage. If congestion occurs, the priority manager may react by exchanging the look-up tables with the alternate look-up tables, and in this way, reduce the average priority of the incoming requests. The reduced priorities mean that on average, every incoming channel gets a lower priority, and the DRAMC will try harder to optimize on bandwidth, and less to optimize to service the high-priority requests first. The switch-over is driven by the congestion state. If many requests come in on "high priority", so they all need to be serviced "first", the "congested" flag goes high, and the controller reacts to this by reducing the request priorities (by switching in the alternate tables), so it can concentrate on the ones that are "really" important, and get room again to optimized bandwidth.

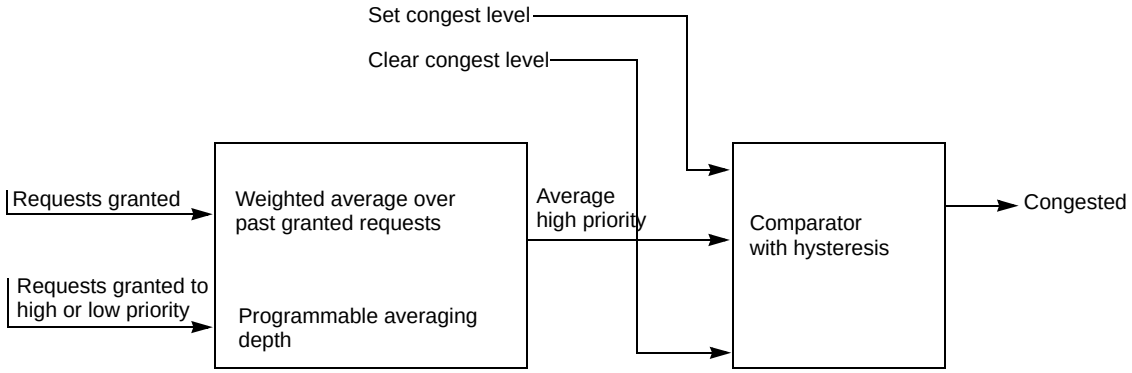



**Figure 14-34. Congestion Detector - Simplified block diagram**

A block diagram of the congestion detector is given in Figure 14-34. The congestion detector consists of an averaging block, followed by a comparator with hysteresis. The averaging block calculates the weighted average of the percentage of high-priority requests, like given below.

$$\text{averagepriority} = \sum \text{weigth(k)} \cdot \text{val(k)}$$

The weighted average uses an exponential weighting when looking at the past granted requests. Requests that have been granted in a more distant past, get a lower weighting coefficient. The weighting coefficient uses an exponential back-off, following the formulae

$$\text{weigth(k)} = \frac{1}{W0} \cdot \exp\left(-\frac{k}{W0}\right)$$

In this formulae, *weight(k)* is the weighting coefficient that is used for the request that was granted *k* acknowledges ago, meaning *k* other requests have been granted after this one. The coefficient *W0* is programmable, dependent on the control field *filter bandwidth* in register hiprio_config, detailed in *Figure 14-5*.

The value input in the weighting block, *val(k)* is dependent on the priority of the request granted. It is 0 if the priority was 7 or lower, it is 0x10000 if the priority was 8 or lower.

The result of the weighted average is a number between 0 and 0x10000, that is input in the comparator with hysteresis. This result, *average_hipriority*, can be monitored in register *hiprio_config*.

The weighted averaging block is followed by a comparator with hysteresis, whose high and low threshold are programmable.

- If *average_hipriority* is greater than *set_congest_level*, the “congested” flag is set.
- If *average_hipriority* is lower than *clear_congest_level*, the “congested” flag is cleared.

This page is intentionally left blank.

# Chapter 15
# e200z4d Core

This chapter provides an overview of the e200z4d microprocessor core present in this device. It includes the following:

- An overview of the core, including the block diagram (Figure 15-1)
- A summary of the feature set for this core (see Section 15.2, “Features”)
  - A description of the execution units (see Section 15.2.1, “Execution Unit Features”)
  - A description of the memory management architecture (see Section 15.2.3, “Memory Management Unit Features”)
  - High-level details of the core memory and coherency model (see Section 15.2.4, “System Bus (Core Complex Interface) Features”)
  - High-level details of the Nexus 3+ features (see Section 15.2.5, “Nexus 3+ Features”)
- A summary of the programming model for this core (see Section 15.3, “Programming model”)
  - An overview of the register set (see Section 15.3.1, “Register set”)
  - An overview of the instruction set (see Section 15.3.2, “Instruction set”)
  - An overview of interrupts and exception handling (see Section 15.3.3, “Interrupts and exception handling”)
- A summary of instruction pipeline and flow (see Section 15.4, “Microarchitecture summary”)

## 15.1 Overview

The e200z4d processor family is a set of CPU cores that implement low-cost versions of Power Architecture technology. The e200z4d core is a dual-issue, 32-bit design with 64-bit general-purpose registers (GPRs). The e200z4d integrates a CPU core, a memory management unit (MMU), a 4-KB instruction cache, and a Nexus Class 3+ real-time debug unit. Separate instruction and data AHB 2.v6 system interfaces are provided.

The e200z4d is compliant with the PowerPC instruction set architecture (ISA). It does not support Power ISA floating-point instructions in hardware, but traps them so they can be emulated by software.

Instructions of the embedded floating-point category are provided to support real-time single-precision embedded numerics operations using the general-purpose registers.

Instructions of the signal processing extension (SPE) category are provided to support real-time SIMD fixed-point and single-precision embedded numerics operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the general-purpose registers (GPRs). The GPRs have been extended to 64-bits in order to support vector instructions defined by the SPE category. These instructions operate on a vector pair of 16-bit or 32-bit data types and deliver vector and scalar results.

In addition to the base Power ISA embedded category instruction set, the core also implements the variable-length encoding category (VLE), which provides improved code density. See the *EREF* and

supplementary VLE Programming Environments Manual (*VLEPEM)* for more information about the VLE extension.

The processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit and load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

Throughout the remainder of this document, the core is referred to as the “e200z4d” when speaking of e200z4d-specific implementations, the “e200z4xx” when speaking of a specific variety of e200z4 core, or “e200” when referring to the whole e200 family.

Figure 15-1 shows the block diagram for the device.

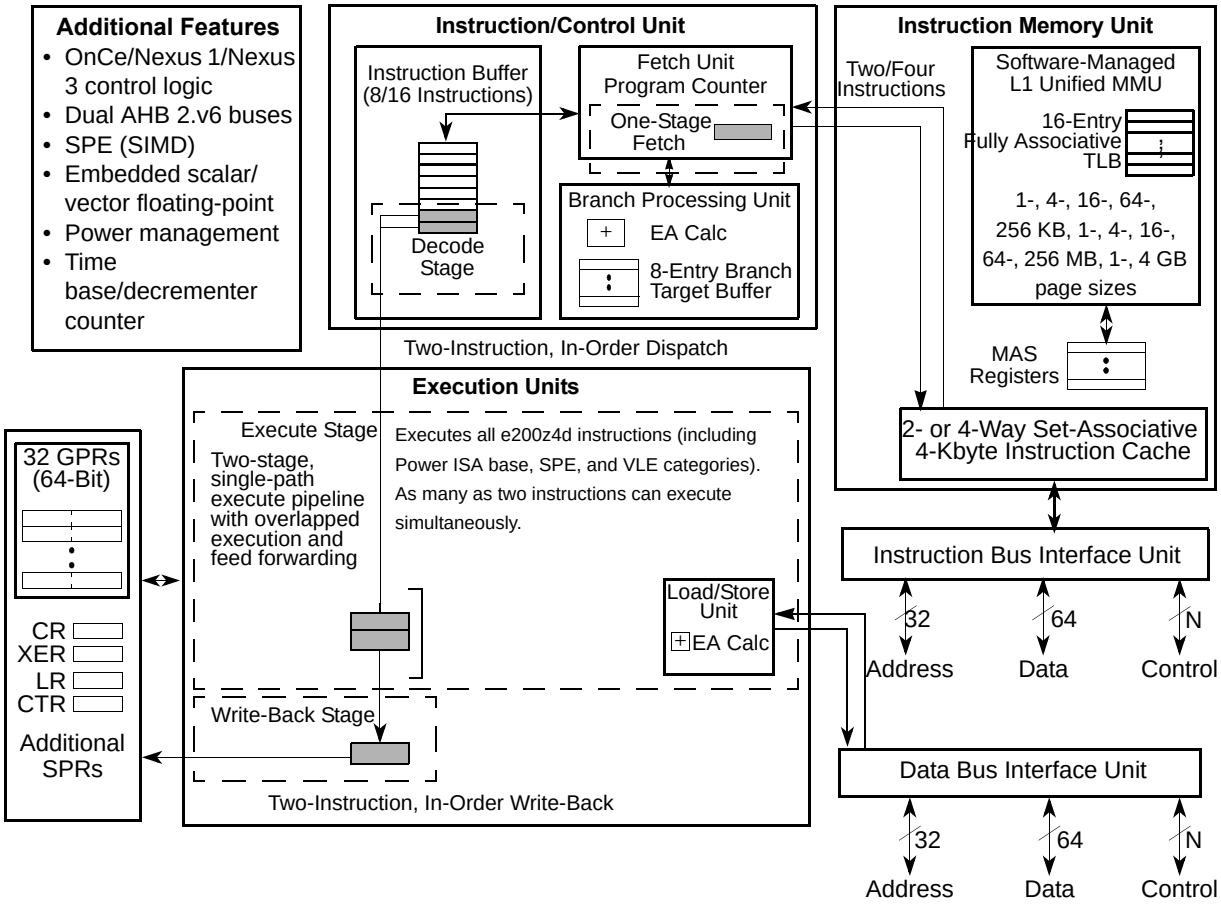



**Figure 15-1. e200z4d block diagram**

## 15.2 Features

Key features of the e200z4d are summarized as follows:

- Dual-issue, 32-bit Power ISA-compliant core
- Implementation of the VLE category for reduced code footprint
- In-order execution and retirement

- Precise exception handling
- Branch processing unit (BPU)
  - Dedicated branch address calculation adder
  - Branch target prefetching using an 8-entry branch target buffer (BTB)
- Supports independent instruction and data accesses to different memory subsystems, such as SRAM and flash memory by means of independent instruction and data bus interface units.
- Load/store unit
- 64-bit general-purpose register file
- Dual advanced high-performance (AHB) 2.v6 64-bit system buses
- Memory management unit (MMU) with 16-entry fully associative TLB and multiple page-size support
- 4 KB, 2/4-way set-associative instruction cache
- Signal processing extension unit, version 1.1 supporting SIMD fixed-point operations using the 64-bit general-purpose register file
- Embedded floating-point (FPU) unit, version 2 supporting scalar and vector SIMD single-precision floating-point operations using the 64-bit general-purpose register file
- Nexus Class 3+ real-time development unit
- Power management
  - Low power design—extensive clock gating
  - Dedicated power saving state: wait
  - Dynamic power management of execution units, cache, and MMU

See the following sections for more details about specific units.

### 15.2.1 Execution Unit Features

The following subsections describes the execution units' main features.

#### 15.2.1.1 Instruction Unit Features

The instruction unit features the following:

- 64-bit path to cache supports fetching of two 32-bit Power ISA instructions or four 16-bit VLE instructions per clock cycle.
- Instruction buffer holds up to eight 32-bit Power ISA instructions or sixteen 16-bit VLE instructions.
- Dedicated program counter (PC) incrementer supports instruction prefetches.
- Branch unit with dedicated branch address adder and branch target buffer supports single-cycle execution of successfully predicted branches.

#### 15.2.1.2 Integer Unit Features

The integer units feature support for single-cycle execution of most integer instructions, as follows:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for count-leading-zeros function
- 32-bit single-cycle barrel shifter for static shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in ≤ 14 clock cycles with minimized execution timing (integer unit 1 only)
- Pipelined 32 × 32 hardware multiplier array supports 32 × 32→32 multiply with 2 clock latency, 1 clock throughput

#### 15.2.1.3 Load/Store Unit Features

The load/store unit supports load, store, and load multiple/store multiple instructions by means of the following:

- 32-bit effective address adder for data memory address calculations
- Pipelined operation supports throughput of one load or store operation per cycle
- Dedicated 64-bit interface to memory supports saving and restoring of up to two registers per cycle for load multiple and store multiple word instructions
- Two-cycle load latency
- Big- and little-endian support
- Misaligned access support

### 15.2.2 L1 Cache Features

The L1 cache features the following:

- 4 KB, 2- or 4-way configurable set-associative instruction cache
- 64-bit data, 32-bit address bus plus attributes and control
- 32-byte line size
- Cache line locking
- Way allocation
- Tag and data parity or multi-bit EDC protection with correction/auto-invalidation capability
- Virtually indexed, physically tagged
- Pseudo round-robin replacement algorithm
- Line-fill buffer
- Hit under fill

### 15.2.3 Memory Management Unit Features

The memory management unit features the following:

- Virtual memory support
- 32-bit virtual and physical addresses
- 8-bit process identifier
- 16-entry fully associative TLB
- Hardware assist for TLB miss exceptions
- Per-entry multiple page size support from 1 Kbyte to 4 Gbyte
- Entry flush protection
- Software managed by **tlbre**, **tlbwe**, **tlbsx**, **tlbsync**, and **tlbivax** instructions
- Freescale EIS MMU architecture compliant
- Support for external control of entry matching for a subset of TID values to support non-intrusive runtime mapping modifications

### 15.2.4 System Bus (Core Complex Interface) Features

The core complex interface features the following:

- Independent instruction and data buses
- Advanced microcontroller bus architecture (AMBA) AHB 2.v6 protocol
- 32-bit address bus, 64-bit data bus, plus attributes and control
- Separate unidirectional 64-bit read and write data buses
- Support for HCLK running at a slower rate than CPU clock

### 15.2.5 Nexus 3+ Features

The Nexus 3+ module provides real-time development capabilities for e200z4d processors in compliance with the IEEE-ISTO 5001™-2008 standard. The ‘3+’ suffix indicates that some Nexus Class 4 features are available. A portion of the pin interface (the JTAG port) is also shared with the OnCE/Nexus 1 unit.

The following features are implemented:

- Program trace by means of branch trace messaging.
  - Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus, static code may be traced.
- Data trace by means of data write messaging and data read messaging.
  - Provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership trace by means of ownership trace messaging (OTM).

- OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated.An ownership trace message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Allows enhanced download/upload capabilities.

- Data acquisition messaging
  - Allows code to be instrumented to export customized information to the Nexus auxiliary output port.
- Watchpoint messaging by means of the auxiliary interface
- Watchpoint trigger enable of program and/or data trace messaging
- Run-time access to the processor memory map by means of the JTAG port

All features are controllable and configurable by means of the JTAG port.

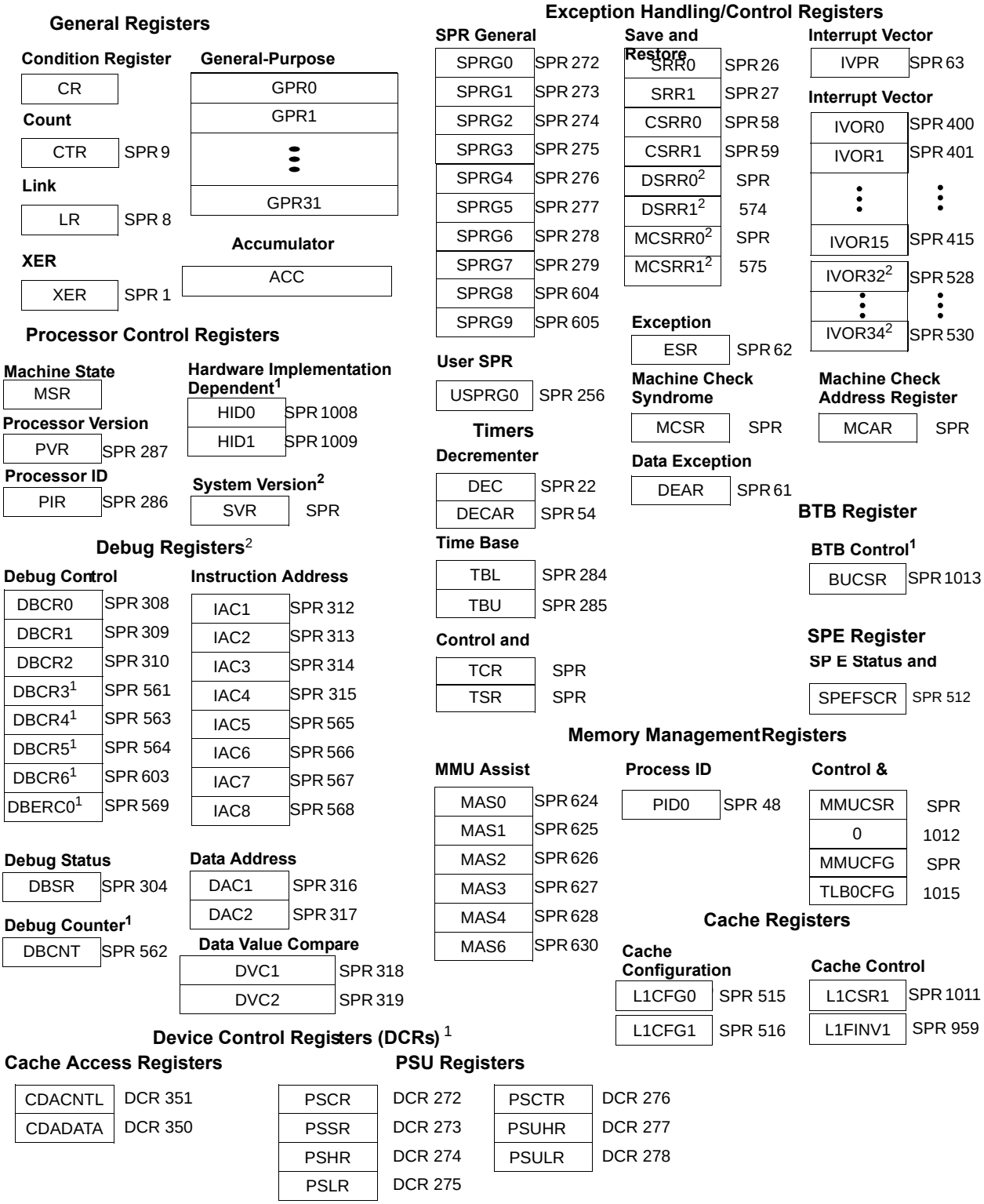
## 15.3 Programming model

This section describes the register model, instruction model, and the interrupt model as they are defined by the Power ISA, Freescale EIS, and the e200z4d implementation.

### 15.3.1 Register set

Figure 15-2 and Figure 15-3 show the complete e200z4d register set, including the sets of the registers that are accessible in supervisor mode and the set of registers that are accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register. For example, the integer exception register (XER) is SPR 1.

Figure 15-2 shows the registers that can be accessed by supervisor-level software. User-level software can access only those registers listed in Figure 15-3.




1 - These e200-specific registers may not be supported by other processors built on Power Architecture technology

2 - Optional registers defined by the Power ISA embedded architecture

**Figure 15-2. e200z4d Supervisor Mode Programmer's Model**

Figure 15-3 shows the user-mode special-purpose registers.

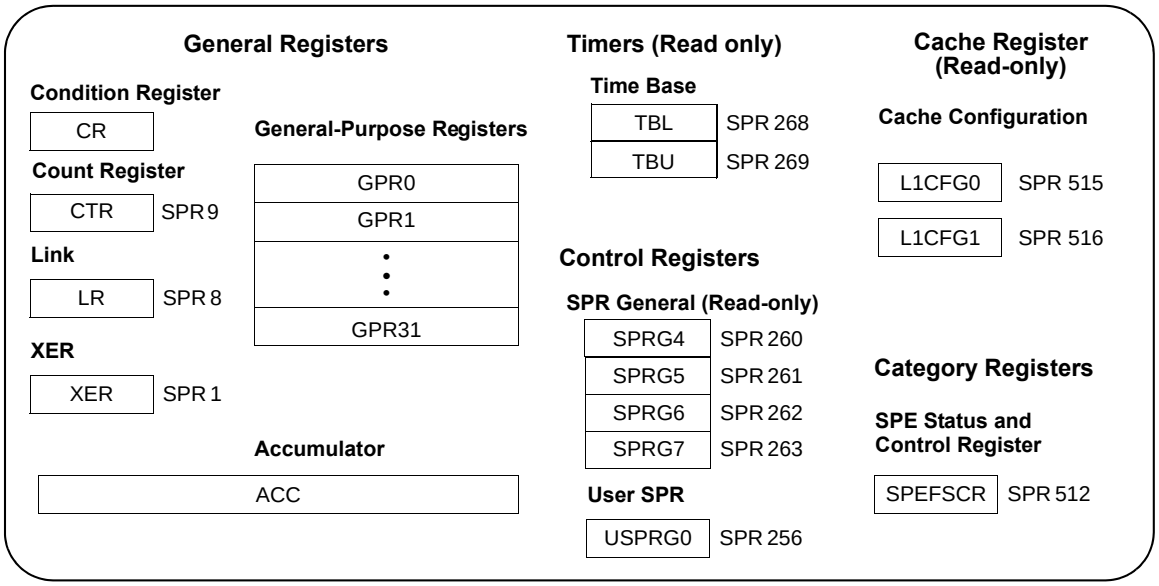



**Figure 15-3. e200z4d User Mode Programmer's Model SPRs**

The GPRs are accessed through instruction operands. Access to other registers can be explicit, by using instructions for that purpose such as the Move to Special-Purpose Register (**mtspr**) and Move from Special-Purpose Register (**mfspr**) instructions. Access to other registers can also be implicit, as part of the execution of an instruction. Some registers are accessed both explicitly and implicitly.

## 15.3.2 Instruction set

The e200z4d supports the Power ISA instruction set for 32-bit embedded implementations. This is composed primarily of the user-level instructions defined by the user instruction set architecture (UISA). The e200z4d does not include the Power ISA floating-point, load string, or store string instructions.

The e200z4d core implements the following architectural extensions:

- The VLE category
- The integer select category (ISEL)
- Enhanced debug and the debug notify halt instruction categories
- The machine check category
- The WAIT category
- The volatile context save/restore category
- The embedded floating-point unit, version 2
- The signal processing extension unit, version 1.1
- The cache line locking category
- The enhanced reservations category

### 15.3.3 Interrupts and exception handling

The e200z4d core supports an extended exception handling model with nested interrupt capability and extensive interrupt vector programmability. In general, interrupt processing begins with an exception that occurs due to external conditions, errors, or program execution problems. When an exception occurs, the processor checks whether interrupt processing is enabled for that particular exception. If enabled, the interrupt causes the state of the processor to be saved in the appropriate registers and begins execution of the handler located at the associated vector address for that particular exception.

Once the handler is executing, the implementation may need to check bits in the exception syndrome register (ESR), the machine check syndrome register (MCSR), or the signal processing and embedded floating-point status and control register (SPEFSCR) to verify the specific cause of the exception and take appropriate action.

The core complex supports the interrupts described in Table 15-1.

**Table 15-1. Interrupt registers**

| Register Desc | ription |
|---|---|
| **Noncritical Interrupt Registers** | |
| SRR0 | Save/restore register 0—On noncritical interrupts, stores either the address of the instruction causing the exception or the address of the instruction that executes after the **rfi** instruction. |
| SRR1 | Save/restore register 1—Saves machine state on noncritical interrupts and restores machine state after an **rfi** instruction is executed. |
| **Critical Interrupt Registers** | |
| CSRR0 | Critical save/restore register 0—On critical interrupts, stores either the address of the instruction causing the exception or the address of the instruction that executes after the **rfci** instruction. |
| CSRR1 | Critical save/restore register 1—Saves machine state on critical interrupts and restores machine state after an **rfci** instruction is executed. |
| **Debug Interrupt Registers** | |
| DSRR0 | Debug save/restore register 0—On debug interrupts, stores either the address of the instruction causing the exception or the address of the instruction that executes after the **rfdi** instruction. |
| DSRR1 | Debug save/restore register 1—Saves machine state on debug interrupts and restores machine state after an **rfdi** instruction is executed. |
| Machine Check Interrupts | |
| MCSRR0 | Machine check save/restore register 0—On machine check interrupts, stores either the address of the instruction causing the exception or the address of the instruction that executes after the **rfmci** instruction. |
| MCSRR1 | Machine check save/restore register 1—Saves machine state on machine check interrupts and restores those values when an **rfmci** instruction is executed |
| **Syndrome Registers** | |
| MCSR | Machine check syndrome register—Saves machine check syndrome information on machine check interrupts. |

**Table 15-1. Interrupt registers (continued)**

| Register Description | |
|---|---|
| ESR | Exception syndrome register—Provides a syndrome to differentiate among the different kinds of exceptions that generate the same interrupt type. Upon generation of a specific exception type, the associated bits are set and all other bits are cleared. |
| **SPE Interrupt Registers** | |
| SPEFSCR | Signal processing and embedded floating-point status and control register—Provides interrupt control and status as well as various condition bits associated with the operations performed by the SPE. See Table 15-2 for a list of the associated IVORs. |
| **Other Interrupt Registers** | |
| DEAR | Data exception address register—Contains the address that was referenced by a load, store, or cache management instruction that caused an alignment, data TLB miss, or data storage interrupt. |
| IVPR<br>IVORs | Together, IVPR[32–47] \|\| IVOR*n* [48–59] \|\| 0b0000 define the address of an interrupt-processing routine. See Table 15-2 for more information. |
| MSR | Machine state register—Defines the state of the processor. When an interrupt occurs, it is updated to preclude unrecoverable interrupts from occurring during the initial portion of the interrupt handler |

Each interrupt has an associated interrupt vector address, obtained by concatenating IVPR[32–47] with the address index in the associated IVOR (that is, IVPR[32–47] || IVOR*n*[48–59] || 0b0000). The resulting address is that of the instruction to be executed when that interrupt occurs. IVPR and IVOR values are indeterminate on reset and must be initialized by the system software using **mtspr**.

Table 15-2 lists IVOR registers implemented on the e200z4d and the associated interrupts.

**Table 15-2. Exceptions and conditions**

| IVOR*n* Interrupt Type | | | IVOR*n* Interrupt Type | |
|---|---|---|---|---|
| None[1] | System reset (not an interrupt) | | 9 | AP unavailable (not used by this core) |
| 0[2] | Critical input | | 10 | Decrementer |
| 1 | Machine check | | 11 | Fixed-interval timer |
| | Machine check (non-maskable interrupt) | | 12 | Watchdog timer |
| 2 | Data storage | | 13 | Data TLB error |
| 3 | Instruction storage | | 14 | Instruction TLB error |
| 4[2] | External input | | 15 | Debug |
| 5 | Alignment | | 16–31 | Reserved |
| 6 | Program | | 32 | SPE unavailable |
| 7 | Floating-point unavailable | | 33 | SPE data exception |
| 8 | System call | | 34 | SPE round exception |

1 Vector to [*p_rstbase[0:29]*] || 0xFFC.

2 The CPU supports external vector override options on these IVORs when they are provided on the device. For example an INTC module may provide a separate vector for each of its sources when in hardware mode.

## 15.4 Microarchitecture summary

The e200z4d processor utilizes a five-stage pipeline for instruction execution. These stages operate in an overlapped fashion, allowing single clock-cycle instruction execution for most instructions. The stages are as follows:

1. Instruction fetch
2. Instruction decode/register file read/effective address calculation
3. Execute 0/memory access 0
4. Execute 1/memory access 1
5. Register write-back

The integer execution units consist of a 32-bit arithmetic unit, a logic unit, a 32-bit barrel shifter, a mask-insertion unit, a condition register manipulation unit, a count-leading-zeros unit, a $32 \times 32$ hardware multiplier array, and result feed-forward hardware. Integer unit 1 also supports hardware division.

Most arithmetic and logical operations are executed in a single cycle with the exception of multiply, which is implemented with a 2-cycle pipelined hardware array, and the divide instructions. A count-leading-zeros unit operates in a single clock cycle.

The instruction unit contains a program counter incrementer and dedicated branch address adder to minimize delays during change-of-flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching using the BTB is performed to accelerate taken branches. Prefetched instructions are placed into an 8-entry instruction buffer, with each entry capable of holding a single 32-bit instruction or a pair of 16-bit instructions.

Branch target addresses are calculated in parallel with branch instruction decode. Conditional branches that are not taken execute in a single clock cycle. Branches with successful BTB target prefetching have an effective execution time of one clock cycle if correctly predicted. All other taken branches have an execution time of two clock cycles.

Memory load and store operations are provided for byte, half-word, word (32-bit), and double-word data with automatic zero or sign extension of byte and half-word load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single-cycle throughput. Load and store multiple word instructions allow low-overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized. There is a single load-to-use bubble for load instructions.

The condition register unit supports the condition register (CR) and condition register operations defined by the architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions. It also provides a mechanism for testing and branching.

Vectored and autovectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

The SPE or SPE2 category supports vector instructions operating on 8-, 16-, and 32-bit fixed-point data types, as well as 32-bit IEEE Std. 754™ single-precision floating-point formats. It supports single-precision floating-point operations in a pipelined fashion.

The 64-bit general-purpose register file is used for source and destination operands, and there is a unified storage model for single-precision floating-point data types of 32-bits and the normal integer type. Low latency fixed-point and floating-point add, subtract, multiply, multiply-add, multiply-sub, divide, compare, and conversion operations are provided. Most operations can be pipelined.

## 15.5 Availability of detailed documentation

Detailed documentation of the e200z4d core will be provided in a separate core reference manual (CRM). When the CRM is available, a link to it will be added to it in this document.

# Chapter 16
# eDMA Channel Mux (DMACHMUX)

## 16.1 Introduction

### 16.1.1 Overview

The DMACHMUX controls the routing of multiple DMA peripheral sources (called slots) to 16 eDMA channels. This is illustrated in Figure 16-1.

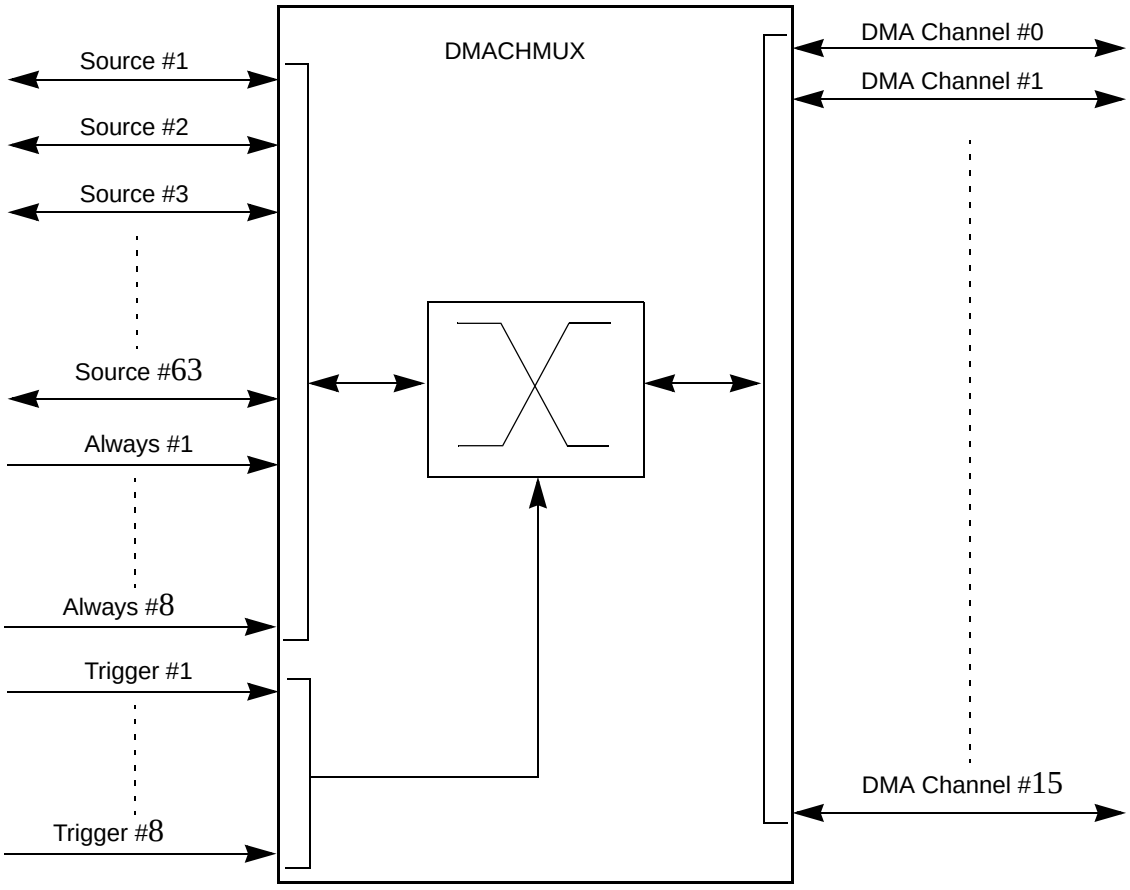



**Figure 16-1. DMACHMUX block diagram**

### 16.1.2 Features

The DMA Channel Mux provides these features:

- 41 peripheral slots + 8 always-on slots can be routed to 16 channels
- 16 independently selectable DMA channels routers
  - — the first 8 channels additionally provide a trigger functionality

- Each channel router can be assigned to one of 41 possible peripheral DMA slots or to one of the 8 always-on slots.

### 16.1.3 Modes of operation

The following operation modes are available:

- Disabled Mode

  In this mode, the DMA channel is disabled. Since disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA Channel Mux. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place (e.g. changing the period of a DMA trigger).
- Normal Mode

  In this mode, a DMA source (such as DSPI transmit or DSPI receive for example) is routed directly to the specified DMA channel. The operation of the DMA Mux in this mode is completely transparent to the system.
- Periodic Trigger Mode

  In this mode, a DMA source may only request a DMA transfer (such as when a transmit buffer becomes empty or a receive buffer becomes full) periodically. Configuration of the period is done in the registers of the Periodic Interrupt Timer (PIT). This mode is only available for channels 0-8.

## 16.2 External signal description

### 16.2.1 Overview

The DMA Channel Mux has no external pins.

## 16.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the DMA Channel Mux.

Table 16-1 shows the memory map for the DMA Channel Mux. Note that all addresses are offsets; the absolute address may be computed by adding the specified offset to the base address of the DMA Channel Mux.

**Table 16-1. Module memory map**

| Address Use | | Access | Location |
|---|---|---|---|
| Base + 0x00 | Channel #0 Configuration (CHCONFIG0) | R/W | on page 643 |
| Base + 0x01 | Channel #1 Configuration (CHCONFIG1) | R/W | on page 643 |
| .. | .. | .. | .. |
| Base + 0x*#n-1* | Channel *#n* Configuration (CHCONFIG*#n-1*)[1] | R/W | on page 643 |

[1] In the table *n* refers to the *number of channels - 1*

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, CHCONFIG0 through CHCONFIG3 are accessible by a 32-bit READ/WRITE to address ‘Base + 0x00’, but performing a 32-bit access to address ‘Base + 0x01’ is illegal.

## 16.3.1 Register descriptions

The following memory-mapped registers are available in the DMA Channel Mux.

### 16.3.1.1 Channel configuration registers

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

Address: Base + *#n* Access: User read/write

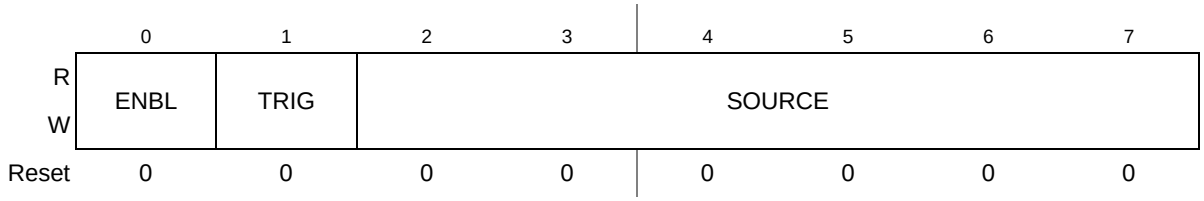



**Figure 16-2. Channel Configuration Registers (CHCONFIG*#n*)**

**Table 16-2. CHCONFIG*xx* Field Descriptions**

| Field D | escription |
|---|---|
| ENBL | DMA Channel Enable. ENBL enables the DMA Channel<br>0 DMA channel is disabled. This mode is primarily used during configuration of the DMA Mux. The DMA has separate channel enables/disables, which should be used to disable or re-configure a DMA channel.<br>1 DMA channel is enabled |
| TRIG | DMA Channel Trigger Enable (for triggered channels only). TRIG enables the periodic trigger capability for the DMA Channel<br>0 Triggering is disabled. If triggering is disabled, and the ENBL bit is set, the DMA Channel will simply route the specified source to the DMA channel.<br>1 Triggering is enabled |
| SOURCE | DMA Channel Source (slot). SOURCE specifies which DMA source, if any, is routed to a particular DMA channel. Please check your SoC-Guide for further details about the peripherals and their slot numbers. |

**Table 16-3. Channel and trigger enabling**

| ENBL TRI G | Function | Mode |
|---|---|---|
| 0 | X | DMA Channel is disabled | Disabled Mode |
| 1 | 0 | DMA Channel is enabled with no triggering (transparent) | Normal Mode |
| 1 | 1 | DMA Channel is enabled with triggering | Periodic Trigger Mode |

**NOTE**

Setting multiple CHCONFIG registers with the same Source value will result in unpredictable behavior.

**NOTE**

Before changing the trigger or source settings a DMA channel must be disabled via the CHCONFIG[*#n*].ENBL bit.

**Table 16-4. DMACHMUX request assignments**

| DMA requesting module | DMACHMUX source number (ipd_ref_peripher[$N_{periphs}$:1]) |
|---|---|
| Channel Disable[1] | DMA MUX Source #0 |
| DSPI 0 TX | DMA MUX Source #1 |
| DSPI 0 RX | DMA MUX Source #2 |
| DSPI 1 TX | DMA MUX Source #3 |
| DSPI 1 RX | DMA MUX Source #4 |
| DSPI 2 TX | DMA MUX Source #5 |
| DSPI 2 RX | DMA MUX Source #6 |
| I2C_0 TX | DMA MUX Source #7 |
| I2C_0 RX | DMA MUX Source #8 |
| I2C_1 TX | DMA MUX Source #9 |
| I2C_1 RX | DMA MUX Source #10 |
| I2C_2 TX | DMA MUX Source #11 |
| I2C_2 RX | DMA MUX Source #12 |
| I2C_3 TX | DMA MUX Source #13 |
| I2C_3 RX | DMA MUX Source #14 |
| eMIOS0_FLAG_F9 | DMA MUX Source #15 |
| eMIOS0_FLAG_F11 | DMA MUX Source #16 |
| eMIOS0_FLAG_F13 | DMA MUX Source #17 |
| eMIOS0_FLAG_F15 | DMA MUX Source #18 |
| eMIOS0_FLAG_F17 | DMA MUX Source #19 |
| eMIOS0_FLAG_F19 | DMA MUX Source #20 |
| eMIOS0_FLAG_F21 | DMA MUX Source #21 |
| eMIOS1_FLAG_F9 | DMA MUX Source #22 |
| eMIOS1_FLAG_F11 | DMA MUX Source #23 |
| eMIOS1_FLAG_F13 | DMA MUX Source #24 |
| eMIOS1_FLAG_F15 | DMA MUX Source #25 |

**Table 16-4. DMACHMUX request assignments (continued)**

| DMA requesting module | DMACHMUX source number (ipd_ref_peripher[$N_{periphs}$:1]) |
|---|---|
| eMIOS1_FLAG_F17 | DMA MUX Source #26 |
| eMIOS1_FLAG_F19 | DMA MUX Source #27 |
| eMIOS1_FLAG_F21 | DMA MUX Source #28 |
| Reserved | DMA MUX Source #29 |
| Reserved | DMA MUX Source #30 |
| LINFlexD_A Rx FIFO Fill | DMA MUX Source #31 |
| LINFlexD_A Tx FIFO Fill | DMA MUX Source #32 |
| LINFlexD_B Rx FIFO Fill | DMA MUX Source #33 |
| LINFlexD_B Tx FIFO Fill | DMA MUX Source #34 |
| LINFlexD_C Rx FIFO Fill | DMA MUX Source #35 |
| LINFlexD_C Tx FIFO Fill | DMA MUX Source #36 |
| LINFlexD_D Rx FIFO Fill | DMA MUX Source #37 |
| LINFlexD_D Tx FIFO Fill | DMA MUX Source #38 |
| DRAMC priority manager DMA req | DMA MUX Source #39 |
| QuadSPI_RX | DMA MUX Source #40 |
| Reserved | DMA MUX Source #41 |
| Reserved | DMA MUX Source #42 |
| SGM-CH0 | DMA MUX Source #43 |
| SGM-CH1 | DMA MUX Source #44 |
| SGM-CH2 | DMA MUX Source #45 |
| SGM-CH3 | DMA MUX Source #46 |
| Reserved | DMA MUX Source #47 |
| Reserved | DMA MUX Source #48 |
| Reserved | DMA MUX Source #49 |
| Reserved | DMA MUX Source #50 |
| ADC | DMA MUX Source #51 |
| Reserved | DMA MUX Source #52 |
| RLE Tx FIFO (data out) | DMA MUX Source #53 |
| RLE Rx FIFO (data in) | DMA MUX Source #54 |
| Reserved | DMA MUX Source #55 |
| ALWAYS requestors | DMA MUX Source #56 |
| ALWAYS requestors | DMA MUX Source #57 |

Table 16-4. DMACHMUX request assignments (continued)

| DMA requesting module | DMACHMUX source number (ipd_ref_peripher[$N_{periphs}$:1]) |
|---|---|
| ALWAYS requestors | DMA MUX Source #58 |
| ALWAYS requestors | DMA MUX Source #59 |
| ALWAYS requestors | DMA MUX Source #60 |
| ALWAYS requestors | DMA MUX Source #61 |
| ALWAYS requestors | DMA MUX Source #62 |
| ALWAYS requestors | DMA MUX Source #63 |

[1] Configuring a DMA channel to select source 0 or any reserved sources will disable that DMA channel.

## 16.4 Functional description

This section provides a functional description of the DMA Channel Mux. The primary purpose of the DMA Channel Mux is to provide flexibility in the system's use of the available DMA channels. As such, configuration of the DMA Mux is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Section 16.5.3, "Enabling and configuring sources" is followed, the configuration of the DMA Channel Mux may be changed during the normal operation of the system.

Functionally, the DMA Channel Mux channels may be divided into two classes: Channels, which implement the normal routing functionality plus periodic triggering capability, and channels, which implement only the normal routing functionality.

### 16.4.1 DMA Channels with periodic triggering capability

Besides the normal routing functionality, the first 8 channels of the DMA Mux provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames or packets at fixed intervals without the need for processor intervention. The trigger is generated by the Periodic Interrupt Timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. Please refer to Chapter 32, "Periodic Interrupt Timer (PIT)," for more information on this topic.

Table 16-5 shows the mapping of PIT channels to DMA channels for triggering.

Table 16-5. PIT-DMA channel mapping

| PIT channel number | DMACHMUX channel number for triggering |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

**NOTE**

Because of the dynamic nature of the system (i.e. DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

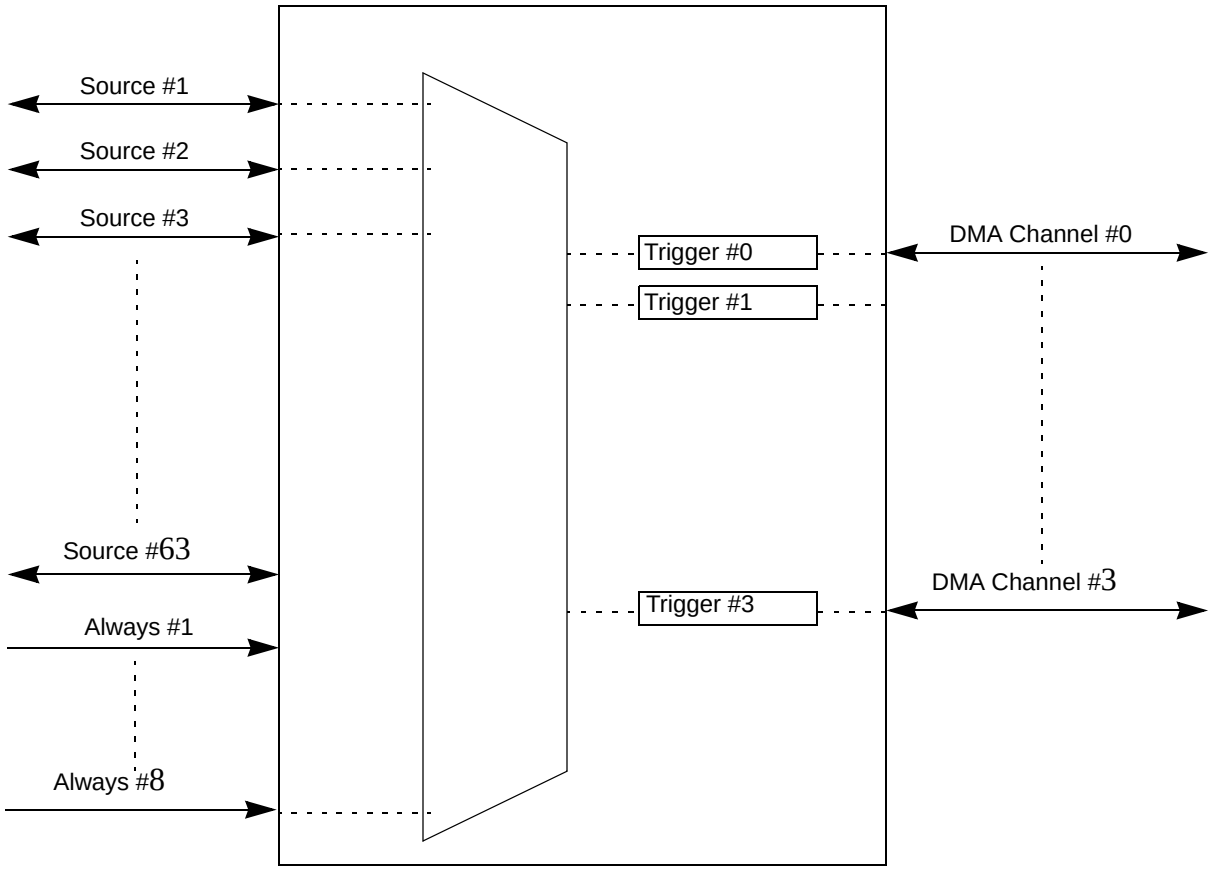



**Figure 16-3. DMA Mux triggered channels**

The DMA channel triggering capability allows the system to “schedule” regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the Peripheral to the DMA until a trigger event has been seen. This is illustrated in Figure 16-4.

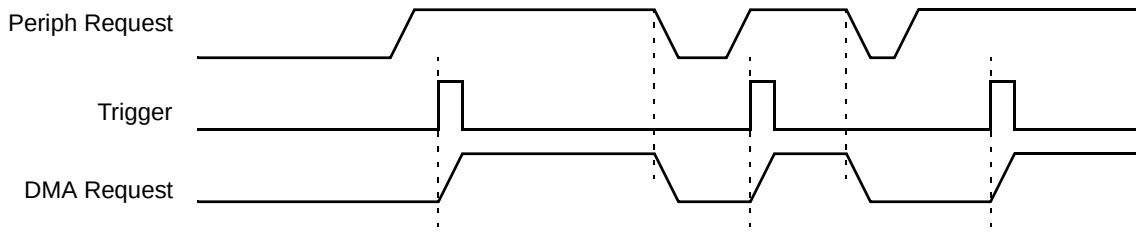



**Figure 16-4. DMA Mux Channel Triggering: Normal Operation**

Once the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral re-asserts its request AND the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, that triggered will be ignored. This situation is illustrated in Figure 16-5.

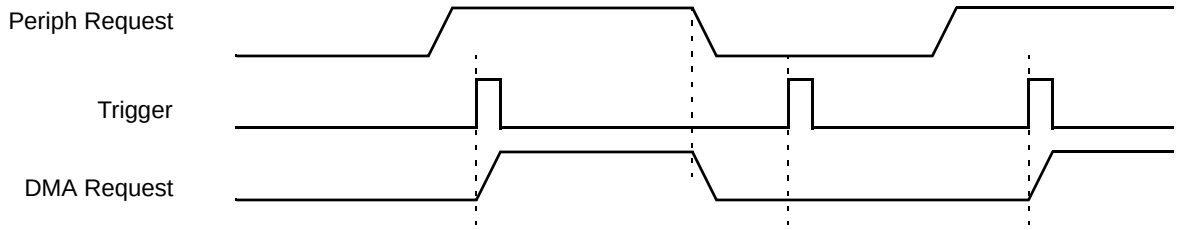



**Figure 16-5. DMA Mux Channel Triggering: Ignored Trigger**

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus. As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. Once setup, the SPI will request DMA transfers (presumably from memory) as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.
- Using the GPIO Ports to drive or sample waveforms. By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger (i.e.-resolution, range of values, etc.) may be found in the Periodic Interrupt Timer (PIT) Block Guide.

### 16.4.2 DMA Channels with no triggering capability

The other channels of the DMA Mux provide the normal routing functionality as described in Section 16.1.3, “Modes of operation.”

### 16.4.3 "Always Enabled" DMA Sources

In addition to the peripherals that can be used as DMA sources, there are 8 additional DMA sources that are "always enabled". Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the "always enabled" sources provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Doing DMA transfers to/from GPIO - Moving data from/to one or more GPIO pins, either un-throttled (i.e.-as fast as possible), or periodically (using the DMA triggering capability).

- Doing DMA transfers from memory to memory - Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Doing DMA transfers from memory to the external bus (or vice-versa) - Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation - Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, a "always enabled" DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require a new "start" event be sent. This can either be a new software activation, or a transfer request from the DMA Channel Mux. The options for doing this are:

- Transfer all data in a single minor loop. By configuring the DMA to transfer all of the data in a single minor loop (i.e.-major loop counter = 1), no re-activation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will incur on the system. For this option, the DMA channel should be disabled in the DMA Channel Mux.
- Use explicit software re-activation. In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to re-activate the channel (by writing to the DMA registers) *after every minor loop*. For this option, the DMA channel should be disabled in the DMA Channel Mux.
- Use a "always enabled" DMA source. In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA Channel Mux does the channel re-activation. For this option, the DMA channel should be enabled and pointing to an "always enabled" source. Note that the re-activation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

## 16.5 Initialization/application information

### 16.5.1 Reset

The reset state of each individual bit is shown within the Register Description section (See Section 16.3.1, “Register descriptions”). In summary, after reset, all channels are disabled and must be explicitly enabled before use.

### 16.5.2 Low Power Considerations

The DMA Channel Mux has no direct involvement with operations to place the MCU into low power modes, however, it is possible for the low power transition to be blocked under certain circumstances.

Specifically, the MCU may not enter a low power mode (HALT/STOP/STANDBY) when all the below conditions are true simultaneously:

- A peripheral with DMA capability is programmed to work on a divided clock.

- This peripheral is programmed to be stopped in the required low power mode and active in RUN Mode.
- AND this peripheral is active with a DMA transfer when software requests a transition to Low Power mode.

To ensure correct operation, ensure that all the DMA enabled peripherals have completed their transfers before requesting transition to low power mode.

### 16.5.3 Enabling and configuring sources

Enabling a source with periodic triggering

1. Determine with which DMA channel the source will be associated. Note that only the first 8 DMA channels have periodic triggering capability.
2. Clear the ENBL and TRIG bits of the DMA channel
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point
4. Configure the corresponding timer
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL and TRIG bits are set

**Example 16-1. Configure source #5 Transmit for use with DMA Channel 2, with periodic triggering capability**

1. Write 0x00 to CHCONFIG2 (Base Address + 0x02)
2. Configure Channel 2 in the DMA, including enabling the channel
3. Configure a timer for the desired trigger interval
4. Write 0xC5 to CHCONFIG2 (Base Address + 0x02)

The following code example illustrates steps #1 and #4 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
```

```
        :
        :
*CHCONFIG2 = 0x00;
*CHCONFIG2 = 0xC5;
```

Enabling a source without periodic triggering

1. Determine with which DMA channel the source will be associated. Note that only the first 8 DMA channels have periodic triggering capability.
2. Clear the ENBL and TRIG bits of the DMA channel
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL is set and the TRIG bit is cleared

**Example 16-2. Configure source #5 Transmit for use with DMA Channel 2, with no periodic triggering capability.**

1. Write 0x00 to CHCONFIG2 (Base Address + 0x02)
2. Configure Channel 2 in the DMA, including enabling the channel
3. Write 0x85 to CHCONFIG2 (Base Address + 0x02)

The following code example illustrates steps #1 and #3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
        :
        :
*CHCONFIG2 = 0x00;
*CHCONFIG2 = 0x85;
```

Disabling a source

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCONFIG registers. Additionally, some module specific configuration may be necessary. Please refer to the appropriate section for more details.

Switching the source of a DMA Channel

1. Disable the DMA channel in the DMA and re-configure the channel for the new source
2. Clear the ENBL and TRIG bits of the DMA channel
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCONFIG register, ensuring that the ENBL and TRIG bits are set

**Example 16-3. Switch DMA Channel 8 from source #5 transmit to source #7 transmit**

1. In the DMA configuration registers, disable DMA channel 8 and re-configure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCONFIG8 (Base Address + 0x08)
3. Write 0x87 to CHCONFIG8 (Base Address + 0x08). (In this example, setting the TRIG bit would have no effect, due to the assumption that channels 8 does not support the periodic triggering functionality).

The following code example illustrates steps #2 and #4 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0xFC084000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCONFIG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCONFIG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCONFIG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCONFIG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCONFIG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCONFIG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCONFIG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCONFIG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCONFIG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCONFIG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCONFIG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCONFIG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCONFIG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCONFIG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCONFIG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCONFIG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
         :
         :
*CHCONFIG8 = 0x00;
*CHCONFIG8 = 0x87;
```

# Chapter 17
# Enhanced Modular IO Subsystem (eMIOS)

## 17.1 Introduction

The configurable enhanced Modular Input/Output Subsystem (eMIOS200 or just eMIOS) provides functionality to generate or measure time events. The eMIOS200 builds on this concept by using a Unified Channel module that provides a superset of the functionality of all the individual MIOS channels, while providing a consistent user interface. Each Unified Channel can be programmed for different functions in different applications of the chip. Besides that, eMIOS200 architecture uses Dedicated Channels which perform specific functions not included in MIOS inheritance.

Figure 17-1 shows the block diagram of the configurable eMIOS200 block.

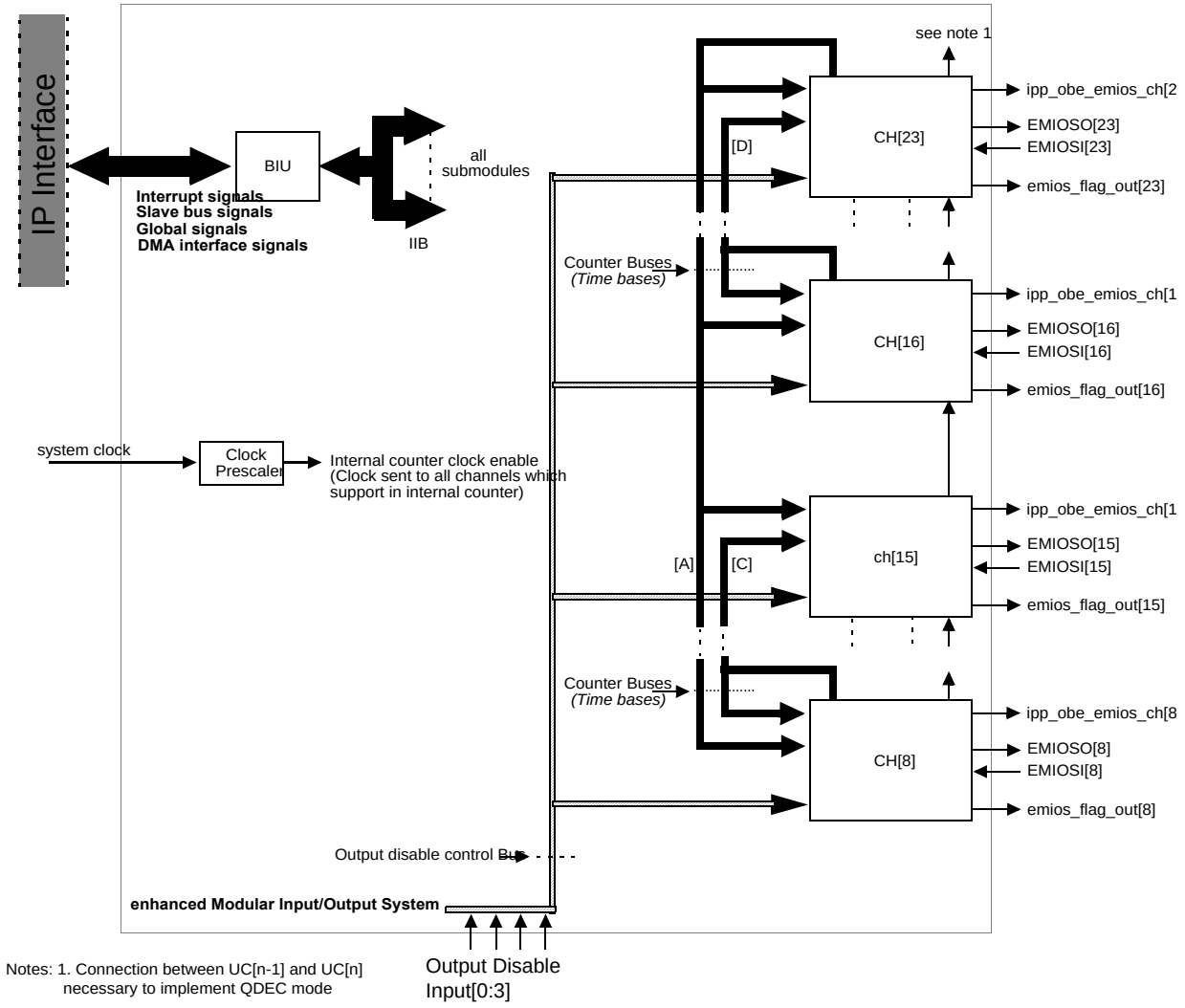



**Figure 17-1. eMIOS200 block diagram**

## 17.2 Features

The basic features of the eMIOS200 are the following:

- 32 channels (16 on eMIOS0 and 16 in eMIOS1) chosen among Unified or Dedicated Channels. Numbered channel 8:23.
- Data registers of 16-bit width
- Counter buses C, and D can be driven by Unified Channel 8, and 16, respectively
- Counter bus A can be driven by the Unified Channel #23
- Each channel has its own time base, alternative to the counter buses
- One Global prescaler
- One Prescaler per channel (CP)
- Shared timebases through the counter buses
- Control and Status bits grouped in a single register
- Synchronization among timebases
- Global flag register
- State of the UC can be frozen for debug purposes
- Motor control capability

## 17.3 Modes of operation

The Unified Channels can be configured to operate in the following modes:

- General purpose input/output
- Single Action Input Capture
- Single Action Output Compare
- Modulus Counter Buffered
- Output Pulse Width and Frequency Modulation Buffered
- Output Pulse Width Modulation Buffered
- Quadrature Decode

These modes are described in Section 17.7.1.1, “UC Modes of Operation.”

Each channel can have a specific set of modes implemented, according to devices requirements.

If an unimplemented mode is selected the results are unpredictable such as writing a reserved value to MODE[0:6] in Section 17.6.2.8, “eMIOS200 UC Control Register (CCR[n]).”

## 17.4 Device-specific information

This device includes two eMIOS modules, eMIOS0 and eMIOS1. Both are 16-channel modules.

The eMIOS provides Timer (IC/OC), PWM functionality, and Quadrature Decode functionality. On this chip, the eMIOS modules have channels clocked by either a modulated or a non-modulated clock.

### 17.4.1 Unsupported features

- Real-Time Signal Bus Client
- Wheel Speed Channels
- eMIOS0 Channels 0–7
  - Channels 9-15 do not support operations on internal counter
- eMIOS1 Channels 0–7

### 17.4.2 Device-specific configuration

- Both eMIOS0 and eMIOS1 can work on any of the four auxiliary clock sources:
  - IRC
  - FXOSC
  - FMPLL0
  - FMPLL1
- For eMIOS0:
  - Counter bus A is driven by Unified Channel #23
  - Counter bus C is driven by Unified Channel #8
  - Counter bus D is driven by Unified Channel #16
  - Unified Channels 9–15 do not have their own time base
- For eMIOS1:
  - Counter bus A is driven by Unified Channel #23
  - Counter bus C is driven by Unified Channel #8
  - Counter bus D is driven by Unified Channel #16

### 17.4.3 eMIOS clocking configuration

The clocking configurations of the eMIOS0 and eMIOS1 modules on this device are shown in Figure 17-2.

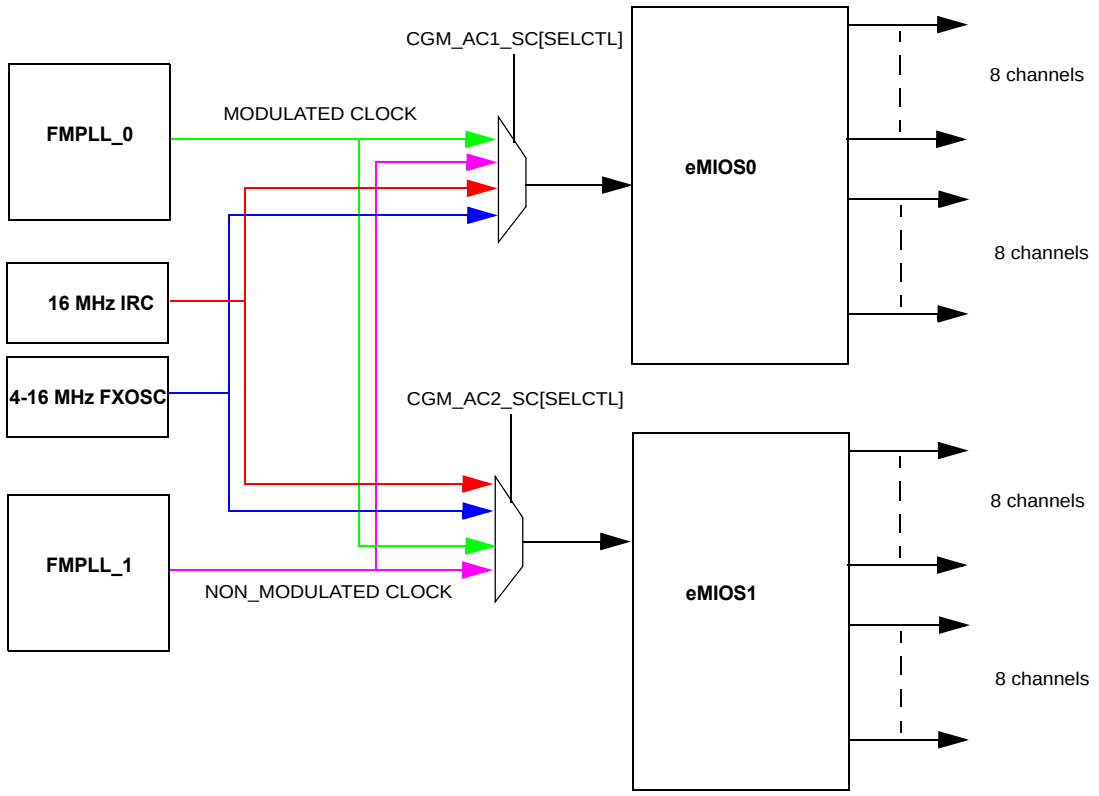



**Figure 17-2. eMIOS clocking configuration**

## 17.4.4 Channel types

The channels of the eMIOS0 and eMIOS1 blocks on this device are implemented using a variety of different channel configurations. The available modes of operation for each channel are shown in Table 17-1 and Table 17-2.

**Table 17-1. eMIOS0 channel configurations**

| Channel number | Channel type | Mode (see Table 17-3) | | | | | |
|---|---|---|---|---|---|---|---|
| | | GPIO | SAIC | SAOC | MCB | OPWFMB | OPWMB |
| 8 | IC/OC counter | x | x | x | x | — | — |
| 9 | IC/OC | x | x | x | — | — | — |
| 10 | | x | x | x | — | — | — |
| 11 | | x | x | x | — | — | — |
| 12 | | x | x | x | — | — | — |
| 13 | | x | x | x | — | — | — |
| 14 | | x | x | x | — | — | — |
| 15 | | x | x | x | — | — | — |

**Table 17-1. eMIOS0 channel configurations (continued)**

| Channel number | Channel type | Mode (see Table 17-3) | | | | | |
|---|---|---|---|---|---|---|---|
| | | GPIO | SAIC | SAOC | MCB | OPWFMB | OPWMB |
| 16 | PWM counter | x | x | x | x | x | x |
| 17 | PWM | x | x | x | — | x | x |
| 18 | | x | x | x | — | x | x |
| 19 | | x | x | x | — | x | x |
| 20 | | x | x | x | — | x | x |
| 21 | | x | x | x | — | x | x |
| 22 | | x | x | x | — | x | x |
| 23 | PWM counter | x | x | x | x | x | x |

**Table 17-2. eMIOS1 channel configurations**

| Channel number | Channel type | Mode (see Table 17-3) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | GPIO | SAIC | SAOC | MCB | OPWFMB | OPWMB | QDEC |
| 8 | IC/OC counter | x | x | x | x | x | x | — |
| 9 | IC/OC | x | x | x | — | x | x | — |
| 10 | | x | x | x | — | x | x | — |
| 11 | | x | x | x | — | x | x | x |
| 12 | | x | x | x | — | x | x | — |
| 13 | | x | x | x | — | x | x | x |
| 14 | | x | x | x | — | x | x | — |
| 15 | | x | x | x | — | x | x | — |
| 16 | PWM counter | x | x | x | x | x | x | — |
| 17 | PWM | x | x | x | — | x | x | — |
| 18 | | x | x | x | — | x | x | — |
| 19 | | x | x | x | — | x | x | x |
| 20 | | x | x | x | — | x | x | — |
| 21 | | x | x | x | — | x | x | x |
| 22 | | x | x | x | — | x | x | — |
| 23 | PWM counter | x | x | x | x | x | x | — |

**Table 17-3. eMIOS channel configuration abbreviations**

| Abbreviation | Meaning |
|---|---|
| GPIO | General-purpose input/output |
| MCB | Modulus counter buffered |
| OPWFMB | Output pulse width and frequency modulation buffered |
| OPWMB | Output pulse width modulation buffered |
| QDEC | Quadrature decode |
| SAIC | Single action input compare |
| SAOC | Single action output compare |

## 17.4.5 Unified Channel Block

Figure 17-3 shows the block diagram of Unified Channel Block as it is implemented in this device.

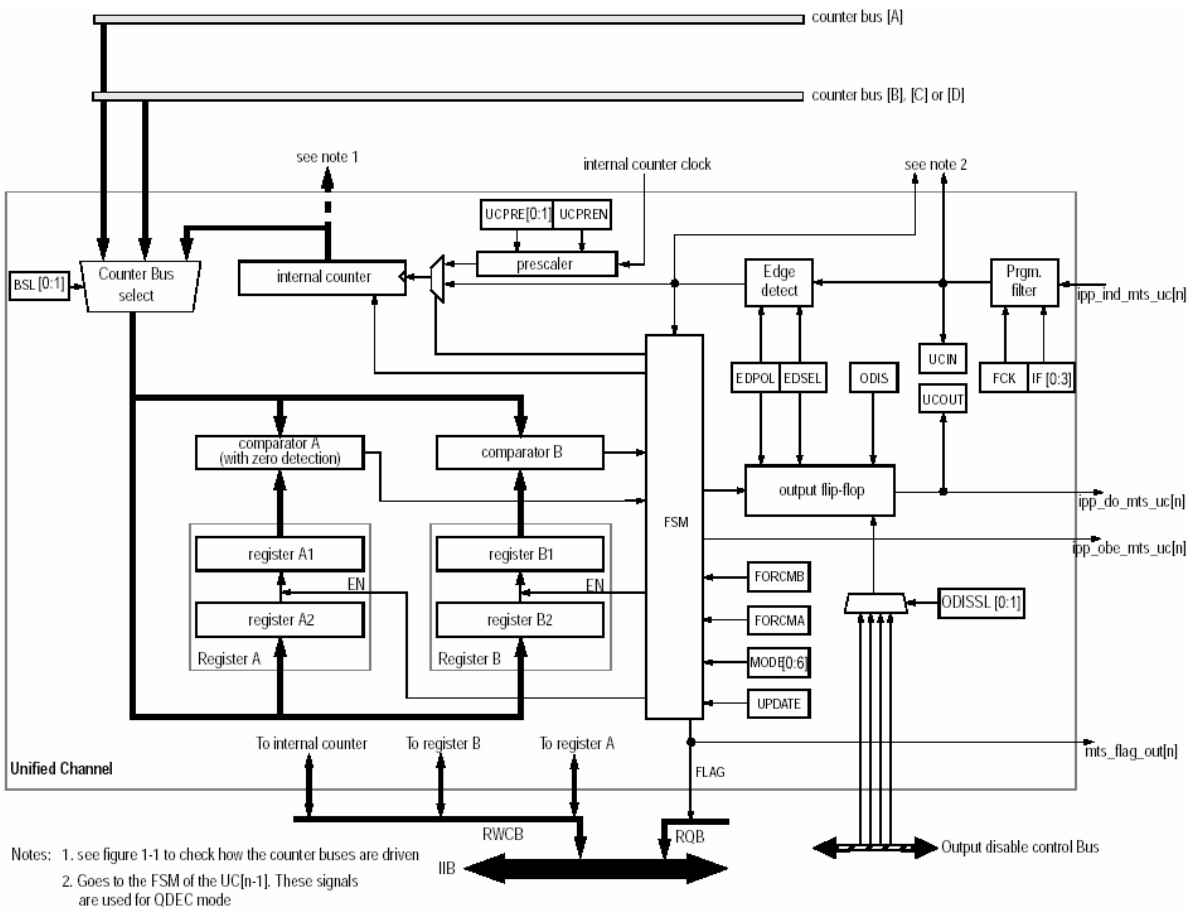



**Figure 17-3. Unified channel block**

## 17.5 External signal description

### 17.5.1 Overview

Each channel has one external input and one external output signal, as described in Table 17-4. The input and output signals are connected to a single bidirectional pin.

### 17.5.2 Detailed signal descriptions

**Table 17-4. External Signals**

| Signal | Direction | Function | Reset State | Pull up |
|---|---|---|---|---|
| emiosi[n] | input | eMIOS200 Channel n input | - | chip dependent |
| emioso[n] | output | eMIOS200 Channel n output | 0/ Hi-Z[1] | chip dependent |
| emios_flag_out[n] | output | eMIOS200 Channel n flag | 0 | chip dependent |

1 Value “0” refers to the reset value of the signal. Hi-Z refers to the state of the external pin if a tristate output buffer is controlled by the corresponding ipp_obe_emios_ch[n] signal.

#### 17.5.2.1 emiosi[n] - eMIOS200 Channel Input Signal

emiosi[n] is synchronized and filtered by the input programmable filter (IPF). The output of the IPF is then used by the channel logic and is available to be read by the MCU through the UCIN bit of the CSR[n] register.

#### 17.5.2.2 emioso[n] - eMIOS200 Channel Output Signal

emioso[n] is a registered output and is available for reading by the MCU through the UCOUT bit of the CSR[n] register. Whilst the channel is operating in input modes the signal state is unknown.

#### 17.5.2.3 emios_flag_out[n] - eMIOS200 Channel Flag Signal

emios_flag_out[n] outputs the state of F[n] bit of GFR register.

## 17.6 Memory map and register description

### 17.6.1 Memory map

The overall address map organization is shown in Table 17-5.

Whenever an access to either an absent register or absent channel is performed the eMIOS200 responds asserting Transfer Error signal from the slave bus interface, as well as for access to reserved address.

**Table 17-5. eMIOS200 memory map**

| eMIOS*n* base address | Description | Location |
|---|---|---|
| **0x000**<br>**0x003** | Module Configuration register (MCR) | on page 661 |
| **0x004**<br>**0x007** | Global FLAG register (GFR) | on page 662 |
| **0x008**<br>**0x00B** | Output Update Disable (OUDR) | on page 663 |
| **0x00C**<br>**0x00F** | Disable Channel (UCDIS) | on page 664 |
| **0x010**<br>**0x11F** | reserved | — |
| **0x120**<br>**0x21F** | Channel [8]<br>to<br>Channel [15] | |
| **0x220**<br>**0x31F** | Channel [16]<br>to<br>Channel [23] | |
| **0x320**<br>**0xFFF** | reserved | — |

### 17.6.1.1 Unified Channel memory map

Addresses of Unified Channel registers are specified as offsets from the channel's base address, otherwise the eMIOS200 base address is used as reference.

Table 17-6 describes the Unified Channel memory map.

**Table 17-6. Unified Channel Memory Map**

| UC[n] base address | Description | Location |
|---|---|---|
| **0x00** | A register (CADR[n]) | on page 666 |
| **0x04** | B register (CBDR[n]) | on page 666 |
| **0x08** | Counter register (CCNTR[n]) | on page 667 |
| **0x0C** | Control register (CCR[n]) | on page 668 |
| **0x10** | Status register (CSR[n]) | on page 672 |
| **0x14** | Alternate A register (ALTCADR[n]) | on page 673 |
| **0x18–0x1F** | reserved | |

## 17.6.2 Register description

All control registers are 32 bits wide. This document illustrates the eMIOS200 with 24 Unified Channels and 16-bit wide data registers.

### 17.6.2.1 eMIOS200 Module Configuration Register (MCR)

The MCR contains Global Control bits for the eMIOS200 block.

address: eMIOS200 base address +0x00

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | MDIS | FRZ | 0 | 0 | GPREN | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GPRE | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-4. eMIOS200 Module Configuration Register (MCR)**

**Table 17-7. MCR field descriptions**

| Field D | escription |
|---|---|
| MDIS | Module Disable<br>Puts the eMIOS200 in low power mode. The MDIS bit is used to stop the clock of the block, except the access to registers MCR, OUDR and UCDIS.<br>1 = Enter low power mode<br>0 = Clock is running |
| FRZ | Freeze<br>Enable the eMIOS200 to freeze the registers of the Unified Channels when Debug Mode is requested at MCU level. Each Unified Channel should have FREN bit set in order to enter freeze state. While in Freeze state, the eMIOS200 continues to operate to allow the MCU access to the Unified Channels registers. The Unified Channel will remain frozen until the FRZ bit is written to zero or the MCU exits Debug mode or the Unified Channel FREN bit is cleared.<br>1 = Stops Unified Channels operation when in Debug mode and the FREN bit is set in the CCR[n] register<br>0 = Exit freeze state |

**Table 17-7. MCR field descriptions (continued)**

| Field D | escription |
|---|---|
| GPREN | Global Prescaler Enable<br>The GPREN bit enables the prescaler counter.<br>1 = Prescaler enabled<br>0 = Prescaler disabled (no clock) and prescaler counter is cleared<br>**Note:** Prescaler must be enabled or no clock (internal or otherwise) will be present on any channel. |
| GPRE[0:7] | Global Prescaler<br>The GPRE[0:7] bits select the clock divider value for the global prescaler, as shown in Table 17-8. |

**Table 17-8. Global Prescaler clock divider**

| GPRE[0:7] | Divide ratio |
|---|---|
| 00000000 | 1 |
| 00000001 | 2 |
| 00000010 | 3 |
| 00000011 | 4 |
| .<br>.<br>.<br>. | .<br>.<br>.<br>. |
| 11111110 | 255 |
| 11111111 | 256 |

### 17.6.2.2 eMIOS200 Global FLAG Register (GFR)

The GFR is a read-only register that groups the FLAG bits from all channels. This organization improves interrupt handling on simpler devices. Each bit relates to one channel.

The two modules on this device, EMIOS0 and EMIOS1, have the same structure for this register as shown in Figure 17-5.

For Unified Channels these bits are mirrors of the FLAG bits in the CSR[n] register.

address: eMIOS0 base address +0x04

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | F23 | F22 | F21 | F20 | F19 | F18 | F17 | F16 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | F15 | F14 | F13 | F12 | F11 | F10 | F9 | F8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-5. eMIOS200 Global FLAG Register (GFR)**

F[n] — Channel [n] Flag bit

Channels that occupy a pair of slots are referred to by their lower slot number (LSB=0 standard), therefore the bits corresponding to their higher slot number always read 0.

### 17.6.2.3 eMIOS200 Output Update Disable (OUDR)

The two modules on this device, EMIOS0 and EMIOS1, have the same structure for this register as shown in Figure 17-6.

address: eMIOS0 base address +0x08

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | OU23 | OU22 | OU21 | OU20 | OU19 | OU18 | OU17 | OU16 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OU15 | OU14 | OU13 | OU12 | OU11 | OU10 | OU9 | OU8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-6. eMIOS200 Output Update Disable Register (OUDR)**

**Table 17-9. OUDR field descriptions**

| Field | Description |
|---|---|
| OU[n] | Channel [n] Output Update Disable bit<br>When running MCB or an output mode, values are written to registers A2 and B2. OU[n] bits are used to disable transfers from registers A2 to A1 and B2 to B1. Each bit controls one channel.<br>1 = Transfers disabled<br>0 = Transfer enabled. Depending on the operation mode, transfer may occur immediately or in the next period. Unless stated otherwise, transfer occurs immediately. |

### 17.6.2.4 eMIOS200 Disable Channel (UCDIS)

The two modules on this device, EMIOS0 and EMIOS1, have the same structure for this register as shown in Figure 17-7.

address: eMIOS0 base address +0x0C

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CHDI S23 | CHDI S22 | CHDI S21 | CHDI S20 | CHDI S19 | CHDI S18 | CHDI S17 | CHDI S16 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CHDI S15 | CHDI S14 | CHDI S13 | CHDI S12 | CHDI S11 | CHDIS 10 | CHDI S9 | CHDI S8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 17-7. eMIOS200 Enable Channel Register (UCDIS)**

**Table 17-10. UCDIS field descriptions**

| Field | Description |
|---|---|
| CHDIS[n] | Enable Channel [n] bit<br>The CHDIS[n] bit is used to disable each of the channels by stopping its respective clock.<br>1 = Channel [n] disabled<br>0 = Channel [n] enabled<br>**Note:** Channels that occupy a pair of slots are referred to as by their lower slot number (LSB=0 standard), therefore the bits corresponding to their higher slot number are reserved and read 0. |

### 17.6.2.5 eMIOS200 UC A Register (CADR[n])

address: UC[n] base address + 0x00

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CADR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-8. eMIOS200 UC A Register (CADR[n])**

Depending on the mode of operation, internal registers A1 or A2, used for matches and captures, can be assigned to address CADR[n]. Both A1 and A2 are cleared by reset. Figure 17-11 summarizes the CADR[n] writing and reading accesses for all operation modes. For more information see section Section 17.7.1.1, “UC Modes of Operation.”

### 17.6.2.6 eMIOS200 UC B Register (CBDR[n])

address: UC[n] base address + 0x04

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CBDR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-9. eMIOS200 UC B register (CBDR[n])**

Depending on the mode of operation, internal registers B1 or B2 can be assigned to address CBDR[n]. Both B1 and B2 are cleared by reset. Table 17-11 summarizes the CBDR[n] writing and reading accesses for all operation modes. For more information see section Section 17.7.1.1, “UC Modes of Operation.”

**Table 17-11. CADR[n], CBDR[n] and ALTCADR[n] value assignments**

| Operation Mode | Register access | | | | | |
|---|---|---|---|---|---|---|
| | write r | ead w | rite r | ead | alt write | alt read |
| GPIO | A1, A2 | A1 | B1, B2 | B1 | A2 | A2 |
| SAIC[1] | — | A2 | B2 | B2 | — | — |
| SAOC[1] | A2 | A1 | B2 | B2 | — | — |
| MCB[1] | A2 | A1 | B2 | B2 | — | — |
| OPWFMB | A2 | A1 | B2 | B1 | — | — |
| OPWMB | A2 | A1 | B2 | B1 | — | — |
| QDEC[1] | A1 | A1 | B2 | B2 | — | — |

1 In these modes, the register CBDR[n] is not used, but B2 can be accessed.

## 17.6.2.7 eMIOS200 UC Counter Register (CCNTR[n])

address: UC[n] base address + 0x08

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W[1] | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CCNTR | | | | | | | | | | | | | | | |
| W[1] | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or reserved

**Figure 17-10. eMIOS200 UC Counter Register (CCNTR[n])**

1 In GPIO mode or Freeze action, this register is writable.

The CCNTR[n] contains the value of the internal counter. When GPIO mode is selected or the channel is frozen, the CCNTR[n] is read/write. For all others modes, the CCNTR[n] is a read-only register. When entering some operation modes, this register is automatically cleared (refer to Section 17.7.1.1, “UC Modes of Operation,” for details).

### 17.6.2.8 eMIOS200 UC Control Register (CCR[n])

The Control register gathers bits reflecting the status of the UC input/output signals and the overflow condition of the internal counter, as well as several read/write control bits.

address: UC[n] base address + 0x0C

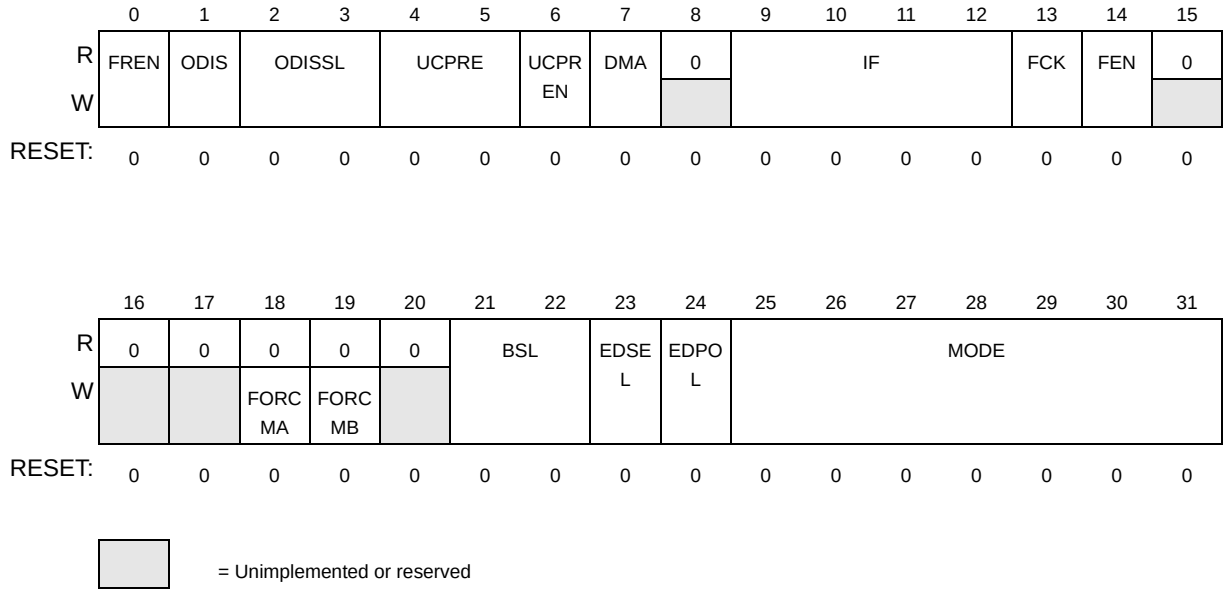



**Figure 17-11. eMIOS200 UC Control Register (CCR[n])**

**Table 17-12. CCR[n] field descriptions**

| Field D | escription |
|---|---|
| FREN | Freeze Enable bit<br>The FREN bit, if set and validated by MCR[FRZ], allows the channel to enter freeze state, freezing all registers values when in debug mode and allowing the MCU to perform debug functions.<br>1 = Freeze UC registers values<br>0 = Normal operation |
| ODIS | Output Disable bit<br>The ODIS bit allows disabling the output pin when running any of the output modes with the exception of GPIO mode.<br>1 = If the flag is set in the channel selected by the ODISSL bits, the output pin goes to EDPOL for OPWFMB and OPWMB modes and to the complement of EDPOL for other output modes, but the Unified Channel continues to operate normally, i.e., it continues to produce FLAG and matches. When the selected Output Disable Input signal is negated, the output pin operates normally<br>0 = The output pin operates normally |
| ODISSL | Output Disable select bits<br>The ODISSL[0:1] bits select one of the four channels which can be used as the output disable signal, as shown in Table 17-13. |
| UCPRE | Prescaler bits<br>The UCPRE[0:1] bits select the clock divider value for the internal prescaler of Unified Channel, as shown in Table 17-14. |

**Table 17-12. CCR[n] field descriptions (continued)**

| Field D | escription |
|---|---|
| UCPREN | Prescaler Enable bit<br>The UCPREN bit enables the prescaler counter.<br>1 = Prescaler enabled<br>0 = Prescaler disabled (no clock)0:1<br>**Note:** Prescaler must be enabled or no clock (internal or otherwise) will be present on the channel. |
| DMA | Direct Memory Access bit<br>The DMA bit selects if the FLAG generation will be used as an interrupt or as a DMA request.<br>1 = Flag/overrun assigned to DMA request<br>0 = Flag/overrun assigned to Interrupt request |
| IF | Input Filter bits<br>The IF[0:3] bits control the programmable input filter, selecting the minimum input pulse width that can pass through the filter, as shown in Table 17-15. For output modes, these bits have no meaning. |
| FCK | Filter Clock select bit<br>The FCK bit selects the clock source for the programmable input filter.<br>1 = main clock<br>0 = prescaled clock |
| FEN | FLAG Enable bit<br>The FEN bit allows the Unified Channel FLAG bit to generate an interrupt signal or a DMA request signal (The type of signal to be generated is defined by the DMA bit).<br>1 = Enable (FlAG will generate an interrupt or DMA request)<br>0 = Disable (FLAG does not generate an interrupt or DMA request) |
| FORCMA | Force Match A bit<br>For output modes, the FORCMA bit is equivalent to a successful comparison on comparator A (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator A, otherwise it has no effect.<br>1 = Force a match at comparator A<br>0 = Has no effect<br>**Note:** For input modes, the FORCMA bit is not used and writing to it has no effect. |
| FORCMB | Force Match B bit<br>For output modes, the FORCMB bit is equivalent to a successful comparison on comparator B (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator B, otherwise it has no effect.<br>1 = Force a match at comparator B<br>0 = Has not effect<br>**Note:** For input modes, the FORCMB bit is not used and writing to it has no effect. |
| BSL | Bus Select bits<br>The BSL[0:1] bits are used to select either one of the counter buses or the internal counter to be used by the Unified Channel. Refer to Table 17-16 for details. |

**Table 17-12. CCR[n] field descriptions (continued)**

| Field D | escription |
|---|---|
| EDSEL | Edge Selection bit<br>For input modes, the EDSEL bit selects whether the internal counter is triggered by both edges of a pulse or just by a single edge as defined by the EDPOL bit. When not shown in the mode of operation description, this bit has no effect.<br>1 = Both edges triggering<br>0 = Single edge triggering defined by the EDPOL bit<br>For GPIO in mode, the EDSEL bit selects if a FLAG can be generated.<br>1 = No FLAG is generated<br>0 = A FLAG is generated as defined by the EDPOL bit<br>For SAOC mode, the EDSEL bit selects the behavior of the output flip-flop at each match.<br>1 = The output flip-flop is toggled<br>0 = The EDPOL value is transferred to the output flip-flop |
| EDPOL | Edge Polarity bit<br>For input modes (except QDEC mode), the EDPOL bit asserts which edge triggers either the internal counter or an input capture or a FLAG. When not shown in the mode of operation description, this bit has no effect.<br>1 = Trigger on a rising edge<br>0 = Trigger on a falling edge<br>For QDEC (MODE[6] cleared), the EDPOL bit selects the count direction according to *direction* signal (UC[n] input).<br>1 = counts up when UC[n] is asserted<br>0 = counts down when UC[n] is asserted<br>**Note:** UC[n-1] EDPOL bit selects which edge clocks the internal counter of UC[n]<br>1 = Trigger on a rising edge<br>0 = Trigger on a falling edge<br>For QDEC (MODE[6] set), the EDPOL bit selects the count direction according to the phase difference.<br>1 = internal counter increments if phase_A is ahead phase_B signal<br>0 = internal counter decrements if phase_A is ahead phase_B signal<br>**Note:** In order to operate properly, EDPOL bit must contain the same value in UC[n] and UC[n-1]<br>For output modes, the EDPOL bit is used to select the logic level on the output pin.<br>1 = A match on comparator A sets the output flip-flop, while a match on comparator B clears it<br>0 = A match on comparator A clears the output flip-flop, while a match on comparator B sets it |
| MODE | Mode selection bits<br>The MODE[0:6] bits select the mode of operation of the Unified Channel, as shown in Table 17-17.<br>**Note:** If a reserved value is written to mode the results are unpredictable. |

**Table 17-13. UC ODISSL selection**

| ODISSL[0:1] | eMIOS0 channel | eMIOS1 channel | input signal |
|---|---|---|---|
| 00 | emios_flag_out[8] | emios_flag_out[8] | Output Disable Input 0 |
| 01 | emios_flag_out[9] | emios_flag_out[9] | Output Disable Input 1 |
| 10 | emios_flag_out[10] | emios_flag_out[10] | Output Disable Input 2 |
| 11 | emios_flag_out[11] | emios_flag_out[11] | Output Disable Input 3 |

**Table 17-14. UC Internal Prescaler Clock Divider**

| UCPRE[0:1] | Divide ratio |
|---|---|
| 00 | 1 |
| 01 | 2 |
| 10 | 3 |
| 11 | 4 |

**Table 17-15. UC Input Filter Bits**

| IF[0:3][1] | Minimum input Pulse width [FLT_CLK periods] |
|---|---|
| 0000 | bypassed[2] |
| 0001 | 02 |
| 0010 | 04 |
| 0100 | 08 |
| 1000 | 16 |
| all others | reserved |

1 Filter latency is 3 clock edges.

2 The input signal is synchronized before arriving to the digital filter.

**Table 17-16. UC BSL bits**

| BSL[0:1] se | lected bus |
|---|---|
| 00 | All channels: counter bus[A] |
| 01 | Channels 8 to 15: counter bus[C]<br>Channels 16 to 23: counter bus[D] |
| 10 | reserved |
| 11 | All channels: internal counter |

**Table 17-17. UC MODE bits**

| MODE[1] | mode of operation |
|---|---|
| 0000000 | General purpose Input/Output mode (input) |
| 0000001 | General purpose Input/Output mode (output) |
| 0000010 | Single Action Input Capture |
| 0000011 | Single Action Output Compare |
| 0000100<br>to<br>0001011 | Reserved |
| 000110b | Quadrature Decode |
| 0001110<br>to<br>1001111 | Reserved |
| 101000b | Modulus Counter Buffered (Up counter) |

**Table 17-17. UC MODE bits (continued)**

| MODE[1] | mode of operation |
|---|---|
| 1010010<br>1010011 | Reserved |
| 10101bb | Modulus Counter Buffered (Up/down counter) |
| 10110b0 | Output Pulse Width and Frequency Modulation Buffered |
| 10110b1<br>to<br>10111b1 | Reserved |
| 11000b0 | Output Pulse Width Modulation Buffered |
| 1100001<br>to<br>1111111 | Reserved |

1 b = adjust parameters for the mode of operation. Refer to Section 17.7.1.1, “UC Modes of Operation,” for details.

### 17.6.2.9 eMIOS200 UC Status Register (CSR[n])

CSR[n] address: UC[n] base address + 0x10

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OVR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | w1c | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OVFL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | UCIN | UCOUT | FLAG |
| W | w1c | | | | | | | | | | | | | | | w1c |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or reserved

**Figure 17-12. eMIOS200 UC Status Register (CSR[n])**

**Table 17-18. CSR[n] field descriptions**

| Field D | escription |
|---|---|
| OVR | Overrun bit. The OVR bit indicates that FLAG generation occurred when the FLAG bit was already set.<br>1 = Overrun has occurred<br>0 = Overrun has not occurred |
| OVFL | Overflow bit. The OVFL bit indicates that an overflow has occurred in the internal counter. OVFL must be cleared by software writing a 1 to the OVFLC bit.<br>1 = An overflow has occurred<br>0 = No overflow |
| UCIN | Unified Channel Input pin bit. The UCIN bit reflects the input pin state after being filtered and synchronized. |
| UCOUT | Unified Channel Output pin bit. The UCOUT bit reflects the output pin state. |
| FLAG | Flag bit. The FLAG bit is set when an input capture or a match event in the comparators occurred.<br>1 = FLAG set event has occurred<br>0 = FLAG cleared<br>**Note:** emios_flag_out reflects the FLAG bit value. When DMA bit is set, the FLAG bit can be cleared by the DMA controller. |

### 17.6.2.10 eMIOS200 UC Alternate A Register (ALTCADR[n])

ALTCADR[n] address: UC[n] base address + 0x14

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ALTCADR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 17-13. eMIOS200 UC Alternate A Register (ALTCADR[n])**

The ALTCADR[n] provides an alternate address to access A2 channel registers in restricted modes (GPIO) only. If CADR[n] is used along with ALTCADR[n], both A1 and A2 registers can be accessed in these modes. Table 17-11 summarizes the ALTCADR[n] writing and reading accesses for all operation modes.

## 17.7 Functional description

The eMIOS200 provides independent channels (UC) that can be configured and accessed by a host MCU. Up to three time bases can be shared by the channels through three counter buses and each channel can generate its own time base.

The eMIOS200 module is based on a multi-bus timer architecture in which several timer channels are used to drive counter buses that are shared among the channels. There are 3 counter buses in the module:

- One global counter bus, shared by all channels
- Two 2 local counter buses, each one dedicated to a slice of 8 channels

Counter bus A is referred to as the global counter bus. Counter buses C and D are the local counter buses.

The eMIOS200 counter buses are driven by channels in specific locations. The global counter bus is driven by the channel in channel slot [23]. Counter buses C and D are driven by channels in slots [8] and [16], respectively. Counter bus A drives all channels. Counter bus C drives channels in slots from [8] through [15]. Counter bus D drives channels in slots from [16] through [23]. Note that the first channel in an 8-channel slice drives the local counter bus for that slice, therefore this channel should not be assigned to be driven by the same counter bus, otherwise a loop occurs. The eMIOS200 Interrupt request signal, DMA transfer request signal among others, are wired to a specific channel, thus the chip integrator should connect those signals having the eMIOS200 channel configuration in mind.

The eMIOS200 block is reset asynchronously. All registers are cleared on reset.

Figure 17-15 describes an eMIOS200 block configured with 32 Unified Channels. Note that the RedLine is also present. Note also that independent of the configuration the channels are fixed in their slots, thus for example if channel [2] is not required this location will be empty, meaning that the other channels locations are not affected. In this case the application software should not access any register located in the channel[2] memory. Any attempt to access those registers will return no meaningful data and a Transfer Error will be generated.

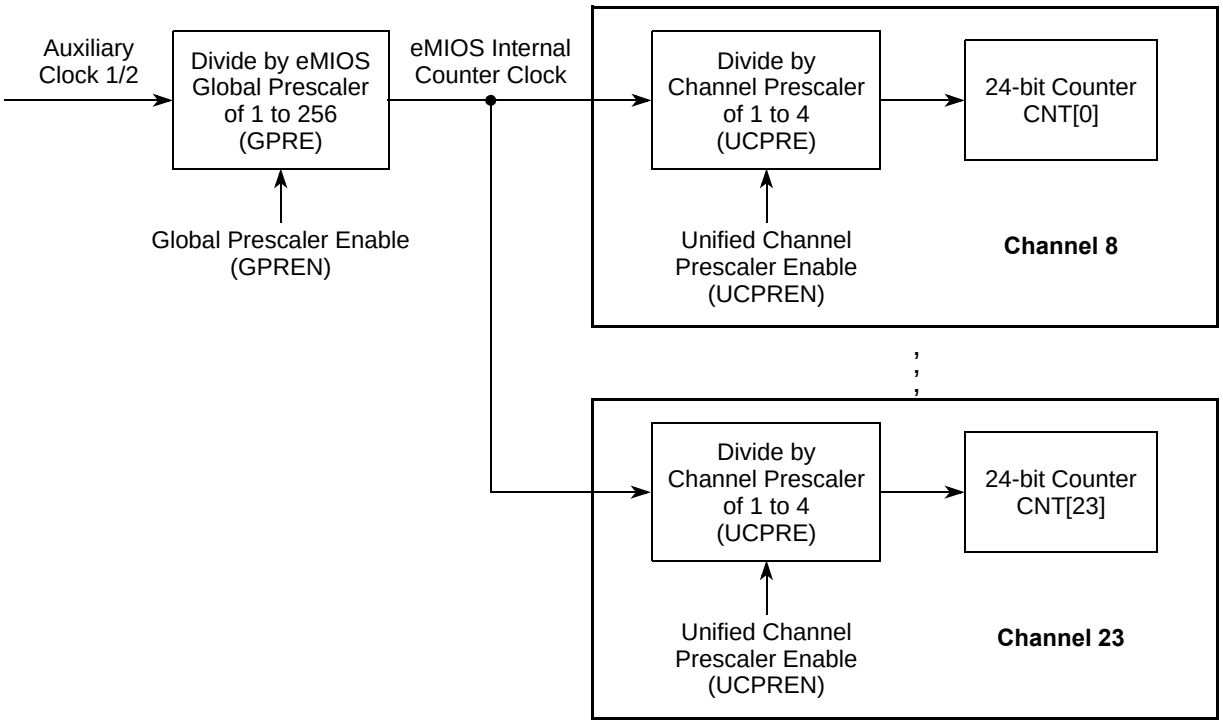



**Figure 17-14. eMIOS prescalers**

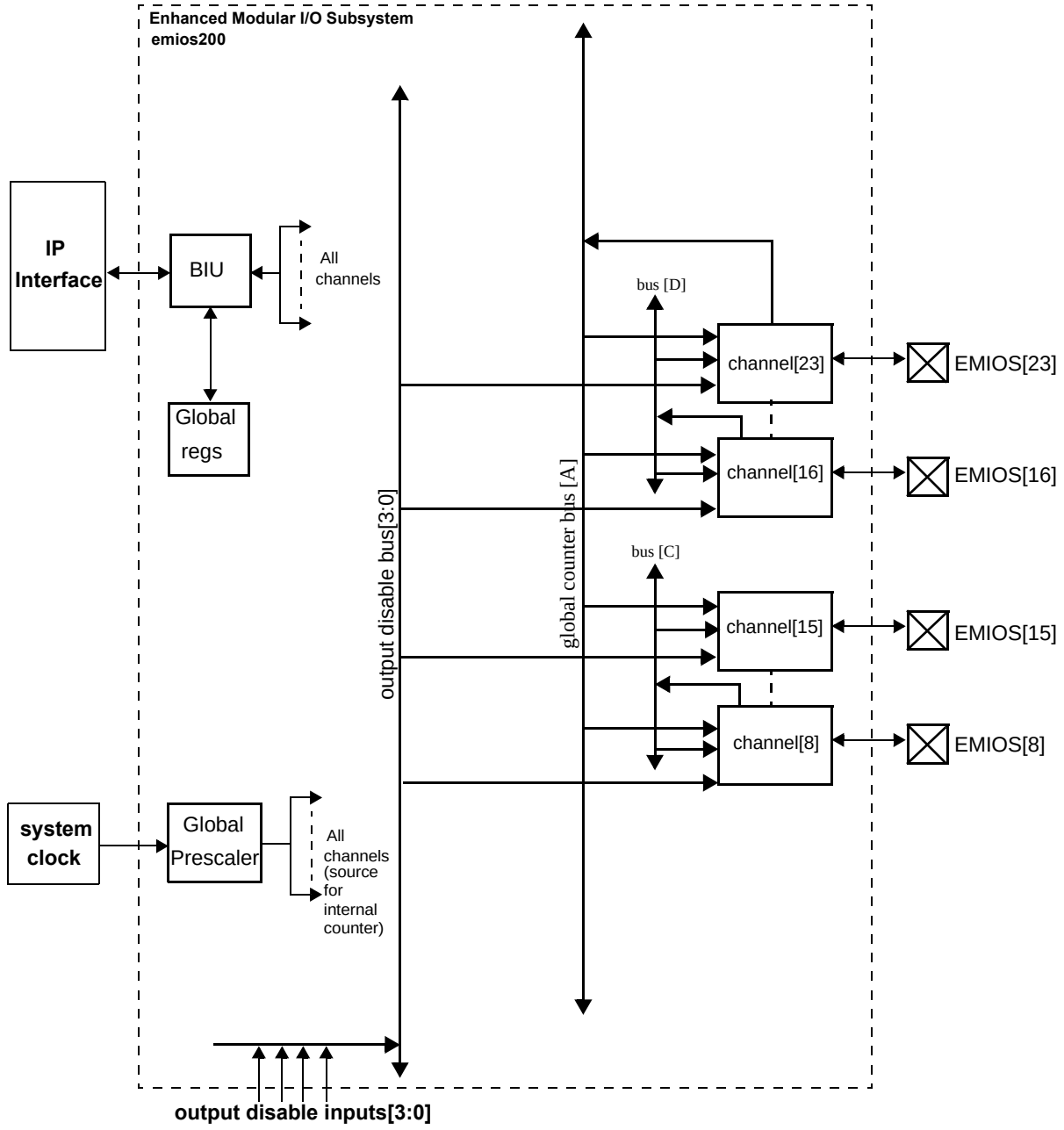



**Figure 17-15. eMIOS200 Full Channel Configuration using Unified Channels only**

## 17.7.1 Unified Channel (UC)

Each Unified Channel consists of:

- Counter bus selector, which selects the time base to be used by the channel for all timing functions
- A programmable clock prescaler

- Two double buffered data registers A and B that allow up to two input capture and/or output compare events to occur before software intervention is needed.
- Two comparators (equal only) A and B, which compares the selected counter bus with the value in the data registers
- Internal counter, which can be used as a local time base or to count input events
- Programmable input filter, which ensures that only valid pin transitions are received by channel
- Programmable input edge detector, which detects the rising, falling or either edges
- An output flip-flop, which holds the logic level to be applied to the output pin
- eMIOS200 Status and Control register
- An Output Disable Input selector that selects which channels flag signal will be used as output disable for the current channel

Figure 17-16 shows both the Unified Channel Control and Datapath block diagram. The Control block is responsible for the generation of signals to control the multiplexes in the Datapath sub-block. Each mode is implemented by a dedicated logic independent from others modes, thus allowing to optimize the logic by disabling the mode and therefore its associated logic. The unused gates are removed during the synthesis phase. Targeting the logic optimization a set of registers is shared by the modes thus providing sequential events to be stored.

The Datapath block provides the channel A and B registers, the internal time base and comparators. Multiplexors select the input of comparators and data for the registers inputs, thus configuring the datapath in order to implement the channel modes. The outputs of A and B comparators are connected to the uc_ctrl control block.

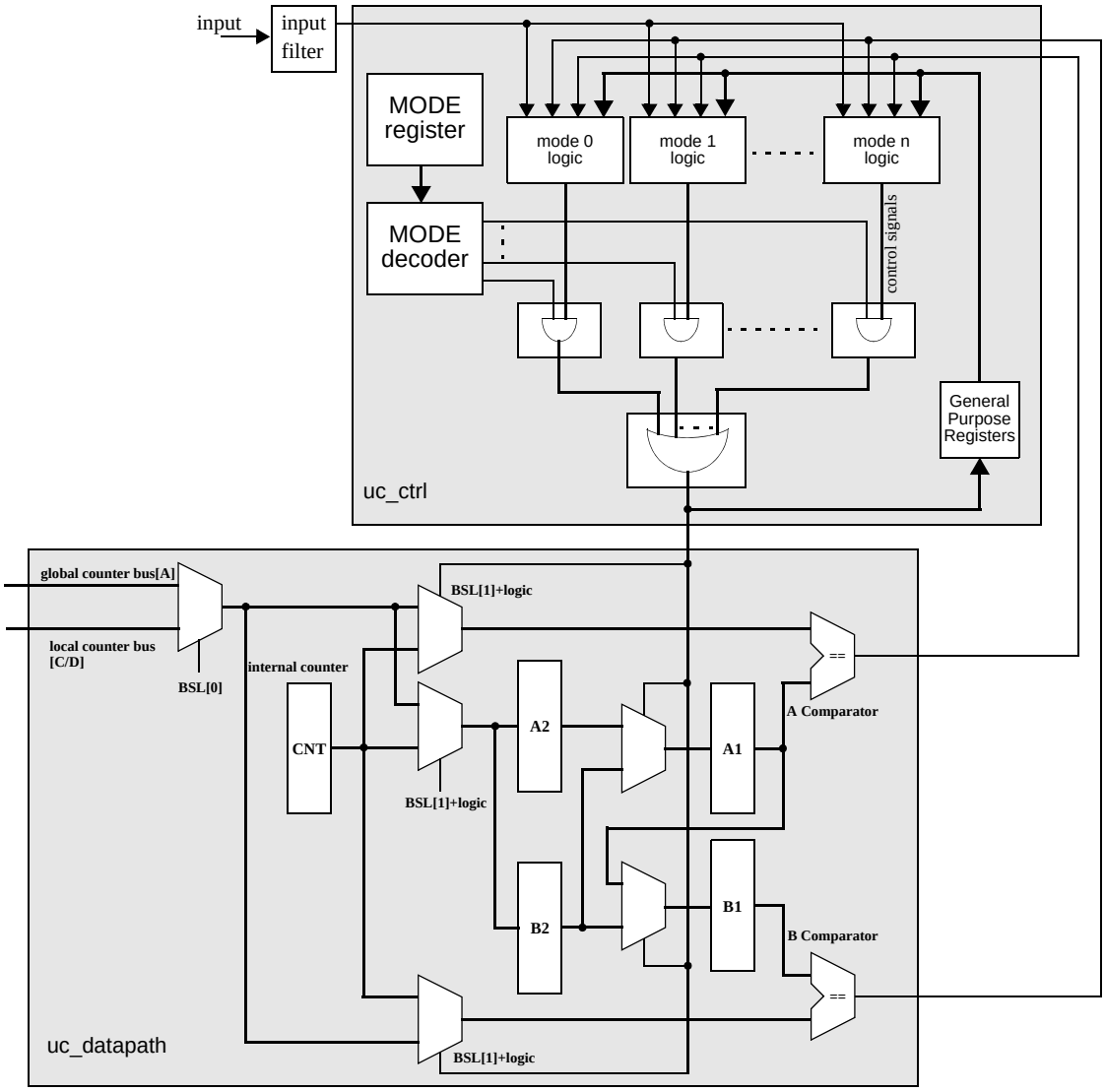



**Figure 17-16. Unified Channel Control and Datapath Block Diagrams**

### 17.7.1.1 UC Modes of Operation

The mode of operation of the Unified Channel is determined by the mode select bits MODE[0:6] in the CCR[n] (see Figure 17-17 for details).

When entering an output mode (except for GPIO mode), the output flip-flop is set to disabled state according to ODIS bit in the CCR[n].

As the internal counter CCNTR[n] continues to run in all modes (except for GPIO mode), it is possible to use this as a time base if the resource is not used in the current mode.

In order to provide smooth waveform generation even if A and B registers are changed on the fly, in the MCB, OPWFMB, and OPWMB the A and B registers are double buffered.

#### 17.7.1.1.1 General purpose Input/Output mode (GPIO) Mode

In GPIO mode, all input capture and output compare functions of the UC are disabled, the internal counter (CCNTR[n]) is cleared and disabled. All control bits remain accessible. In order to prepare the UC for a new operation mode, writing to CADR[n] or CBDR[n] stores the same value in registers A1/A2 or B1/B2, respectively. Writing to ALTCADR[n] stores a value only in register A2.

MODE[6] bit selects between input (MODE[6] = 0) and output (MODE[6] = 1) modes.

**CAUTION**

When changing MODE[0:6], the application software must go to GPIO mode first in order to reset the UC's internal functions properly. Failure to do this could lead to invalid and unexpected output compare or input capture results or the FLAGs being set incorrectly.

In GPIO input mode (MODE[0:6]=0000000), the FLAG generation is determined according to EDPOL and EDSEL bits and the input pin status can be determined by reading the UCIN bit.

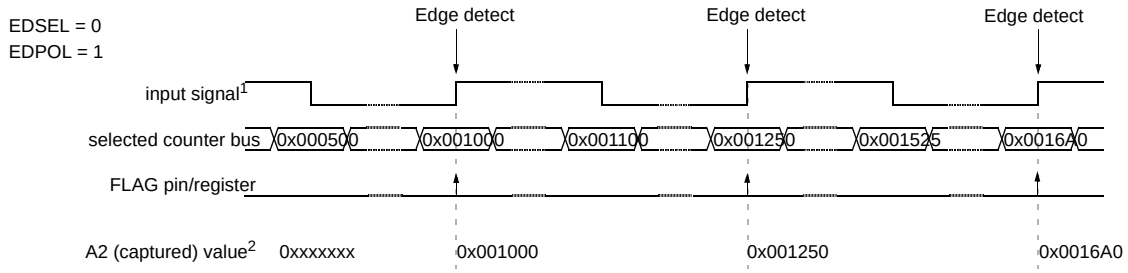
In GPIO output mode (MODE[0:6]=0000001), the Unified Channel is used as a single output port pin and the value of the EDPOL bit is permanently transferred to the output flip-flop.

#### 17.7.1.1.2 Single Action Input Capture (SAIC) Mode

In SAIC mode (MODE[0:6]=0000010), when a triggering event occurs on the input pin, the value on the selected time base is captured into register A2. The FLAG bit is set along with the capture event to indicate that an input capture has occurred. CADR[n] returns the value of register A2. As soon as the SAIC mode is entered coming out from GPIO mode the channel is ready to capture events. The events are captured as soon as they occur thus reading register A always returns the value of the latest captured event. Subsequent captures are enabled with no need of further reads from CADR[n]. The FLAG is set at any time a new event is captured.

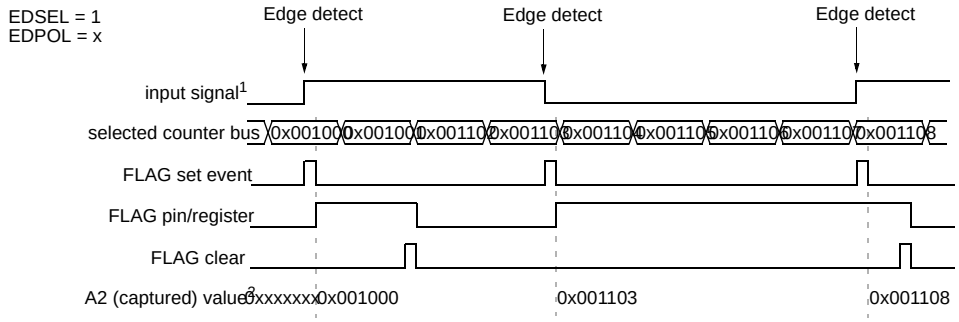
The input capture is triggered by a rising, falling or either edges in the input pin, as configured by the EDPOL and EDSEL bits in CCR[n].

Figure 17-17 and Figure 17-18 shows how the Unified Channel can be used for input capture.




Notes: 1. After input filter
2. CADR[n] <= A2

**Figure 17-17. Single Action Input Capture with rising edge triggering example**




Notes: 1. After input filter
2. CADR[n] <= A2

**Figure 17-18. Single Action Input Capture with both edges triggering example**

### 17.7.1.1.3 Single Action Output Compare (SAOC) Mode

In SAOC mode (MODE[0:6]=0000011) a match value is loaded in register A2 and then immediately transferred to register A1 to be compared with the selected time base. When a match occurs, the EDSEL bit selects whether the output flip-flop is toggled or the value in EDPOL is transferred to it. Along with the match the FLAG bit is set to indicate that the output compare match has occurred. Writing to CADR[n] stores the value in register A2 and reading to CADR[n] returns the value of register A1.

An output compare match can be simulated in software by setting the FORCMA bit in CCR[n]. In this case, the FLAG bit is not set.

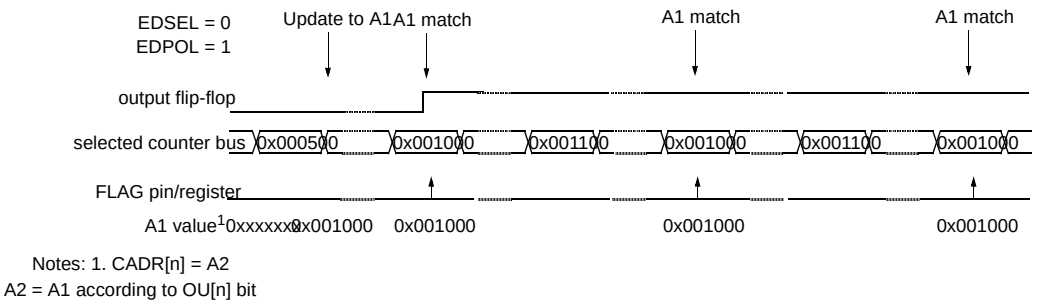
When SAOC mode is entered coming out from GPIO mode the output flip-flop is set to the complement of the EDPOL bit in CCR[n].

Counter bus can be either internal or external and is selected through BSL[0:1] bits.

Figure 17-19 and Figure 17-20 show how the Unified Channel can be used to perform a single output compare with EDPOL value being transferred to the output flip-flop and toggling the output flip-flop at each match, respectively. Note that once in SAOC mode the matches are enabled thus the desired match value on register A1 must be written before the mode is entered. A1 register can be updated at any time thus modifying the match value which will reflect in the output signal generated by the channel. Subsequent matches are enabled with no need of further writes to CADR[n]. The FLAG is set at the same time a match occurs.

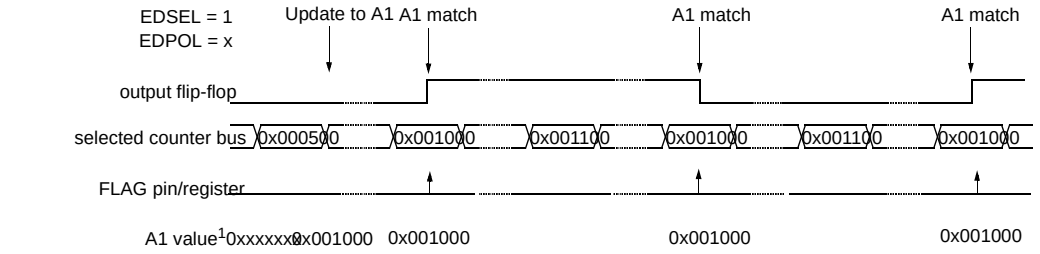
**NOTE**

The channel internal counter in SAOC mode is free-running. It starts counting as soon as the SAOC mode is entered.



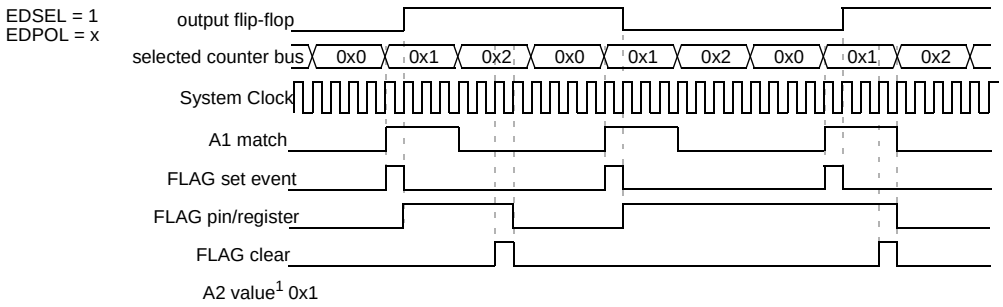

Notes: 1. CADR[n] = A2
A2 = A1 according to OU[n] bit

**Figure 17-19. SAOC example with EDPOL value being transferred to the output flip-flop**




Notes: 1. CADR[n] = A2
A2 = A1 according to OU[n] bit

**Figure 17-20. SAOC example toggling the output flip-flop**




Note: 1. CADR[n] <= A2

### 17.7.1.1.4 Modulus Counter Buffered (MCB) Mode

The MCB mode provides a time base which can be shared with other channels through the internal counter buses. Register A1 is double buffered thus allowing smooth transitions between cycles when changing A2 register value on the fly. A1 register is updated at the cycle boundary, which is defined as when the internal counter transitions to 0x1.

The internal counter values operates within a range from 0x1 up to register A1 value. If when entering MCB mode coming out from GPIO mode the internal counter value is not within that range then the A match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xff_ffff for a 24-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal MCB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to A1 register value range when the MCB mode is entered.

MODE[6] bit selects internal clock source if cleared or external if set. When external clock is selected the input channel pin is used as the channel clock source. The active edge of this clock is defined by EDPOL and EDSEL bits in CCR[n].

When entering in MCB mode, if up counter is selected by MODE[4]=0 (MODE[0:6]=101000b), the internal counter starts counting from its current value to up direction until A1 match occurs. The internal counter is set to 0x1 when its value matches A1 value and a clock tick occurs (either prescaled clock or input pin event).

If up/down counter is selected by setting MODE[4]=1, the counter changes direction at A1 match and counts down until it reaches the value 0x1. After it has reached 0x1 it is set to count in up direction again. B1 register is used to generate a match in order to set the internal counter in up-count direction if up/down mode is selected. Register B1 cannot be changed while this mode is selected.

Note that the MCB mode counts between 0x1 and A1 register value. Only values greater than 0x1 must be written at A1 register. Loading values other than those leads to unpredictable results. The counter cycle period is equal to A1 value in up counter mode. If in up/down counter mode the period is defined by the expression: (2*A1)-2.

Figure 17-21 describes the counter cycle for several A1 values. Register A1 is loaded with A2 register value at the cycle boundary. Thus any value written to A2 register within cycle **n** will be updated to A1 at the next cycle boundary and therefore will be used on cycle **n+1**. The cycle boundary between cycle **n** and cycle **n+1** is defined as when the internal counter transitions from A1 value in cycle **n** to 0x1 in cycle **n+1**. Note that the FLAG is generated at the cycle boundary and has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

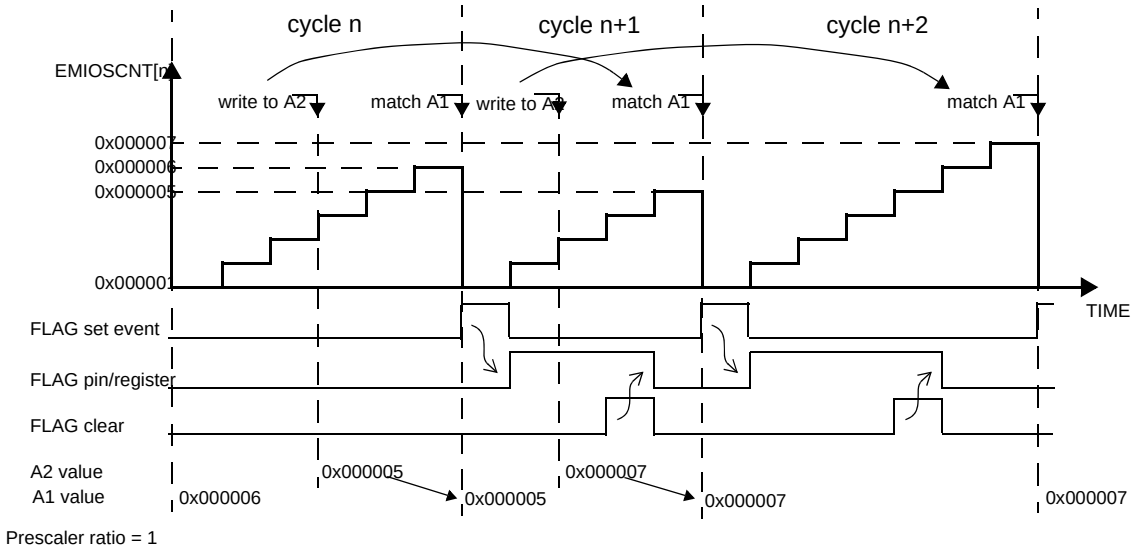



**Figure 17-21. Modulus Counter Buffered (MCB) Up Count mode**

Figure 17-22 describes the MCB in up/down counter mode (MODE[0:6]=10101bb). A1 register is updated at the cycle boundary. If A2 is written in cycle **n**, this new value will be used in cycle **n+1** for A1 match. Flags are generated only at A1 match start if MODE[5] is 0. If MODE[5] is set to 1 flags are also generated at the cycle boundary.

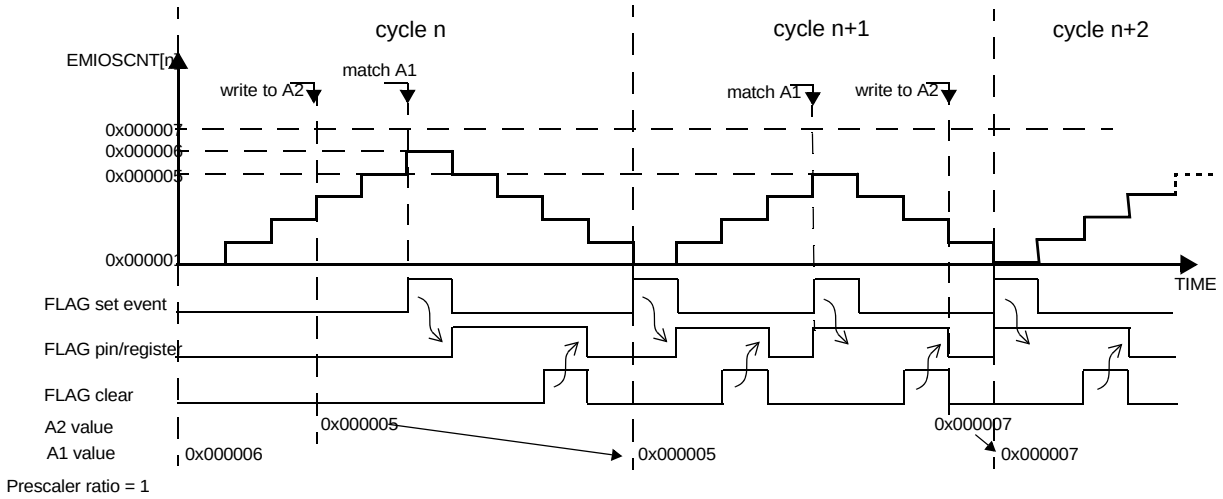



**Figure 17-22. Modulus Counter Buffered (MCB) Up/Down Mode**

Figure 17-23 describes in more detail the A1 register update process in up counter mode. The A1 load signal is generated at the last system clock period of a counter cycle. Thus, A1 is updated with A2 value at the same time that the counter (CCNTR[n]) is loaded with 0x1. The load signal pulse has the duration of one system clock period. If A2 is written within cycle **n** its value is available at A1 at the first clock of cycle **n+1** and the new value is used for match at cycle **n+1**. The update disable bits OU[n] of OUDR can be used to control the update of this register, thus allowing to delay the A1 register update for synchronization purposes.

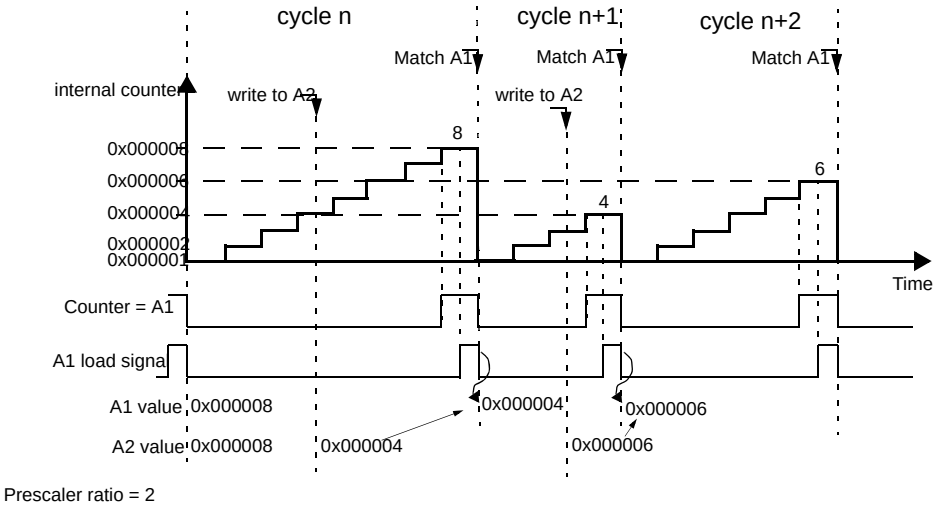



**Figure 17-23. MCB Mode A1 Register Update in Up Counter Mode**

Figure 17-24 describes the A1 register update in up/down counter mode. Note that A2 can be written at any time within cycle **n** in order to be used in cycle **n+1**. Thus A1 receives this new value at the next cycle boundary. Note that the update disable bits OU[n] of OUDR can be used to disable the update of A1 register.

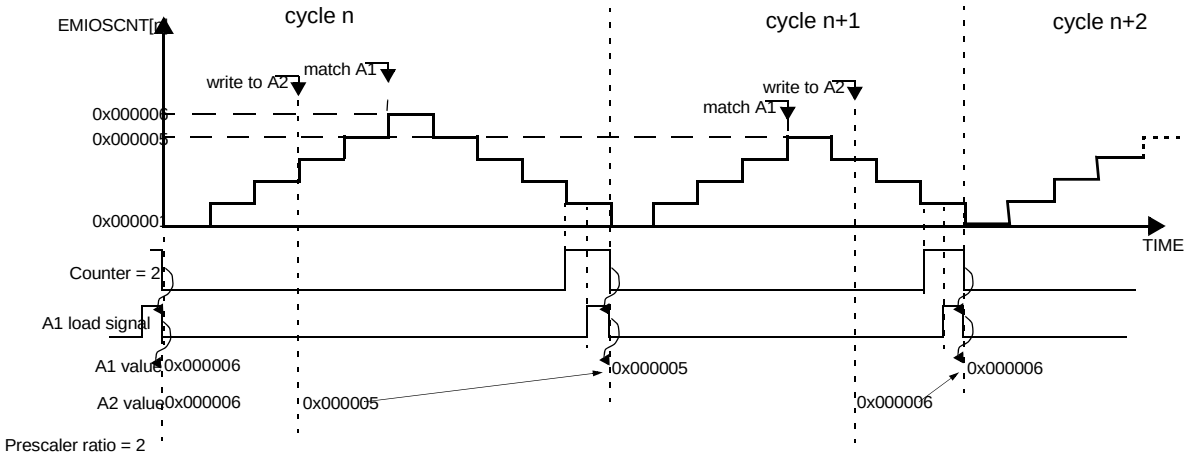



**Figure 17-24. MCB Mode A1 Register Update in Up/Down Counter Mode**

### 17.7.1.1.5 Output Pulse Width and Frequency Modulation Buffered (OPWFMB) Mode

This mode (MODE[0:6]=10110b0) provides waveforms with variable duty cycle and frequency. The internal channel counter is automatically selected as the time base when this mode is selected. A1 register indicates the duty cycle and B1 register the frequency. Both A1 and B1 registers are double buffered to allow smooth signal generation when changing the registers values on the fly. 0% and 100% duty cycles are supported.

At OPWFMB mode entry the output flip-flop is set to the value of the EDPOL bit in the CCR[n].

If when entering OPWFMB mode coming out from GPIO mode the internal counter value is not within that range then the B match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xff_ffff for a 24-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal OPWFMB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to B1 register value range when the OPWFMB mode is entered.

When a match on comparator A occurs the output register is set to the value of EDPOL. When a match on comparator B occurs the output register is set to the complement of EDPOL. B1 match also causes the internal counter to transition to 0x1, thus restarting the counter cycle.

Only values greater than 0x1 are allowed to be written to B1 register. Loading values other than those leads to unpredictable results.

Figure 17-25 describes the operation of the OPWFMB mode regarding output pin transitions and A1/B1 registers match events. Note that the output pin transition occurs when the A1 or B1 match signal is deasserted which is indicated by the A1 match negedge detection signal. If register A1 is set to 0x4 the output pin transitions 4 counter periods after the cycle had started, plus one system clock cycle. Note that in the example shown in Figure 17-25 the internal counter prescaler has a ratio of two.

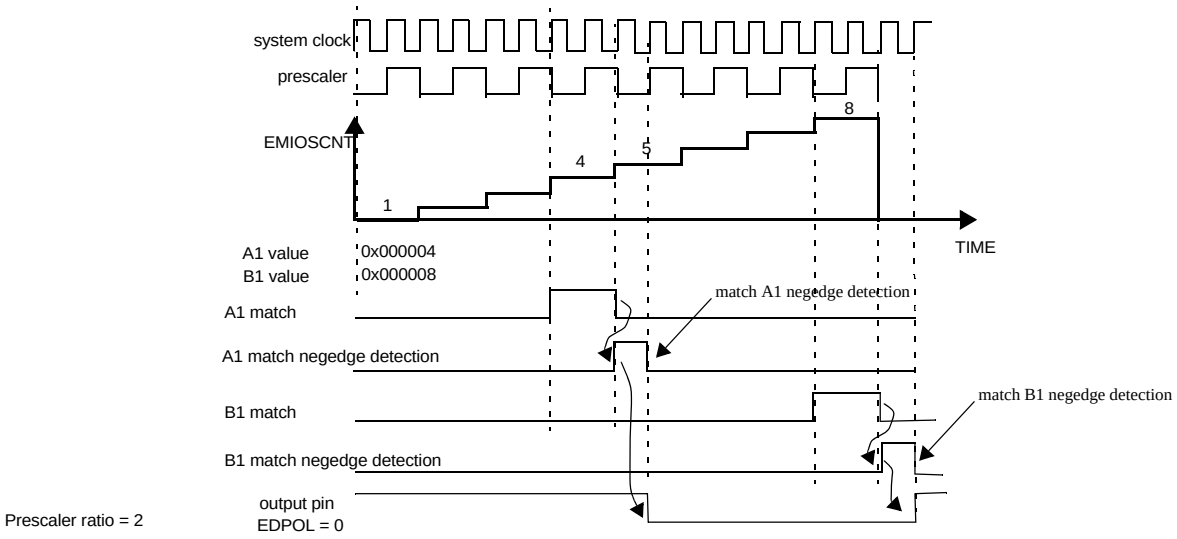



**Figure 17-25. OPWFMB A1 and B1 match to Output Register Delay**

Figure 17-26 describes the generated output signal if A1 is set to 0x0. Since the counter does not reach zero in this mode, the channel internal logic infers a match as if A1=0x1 with the difference that in this case, the posedge of the match signal is used to trigger the output pin transition instead of the negedge used when A1=0x1. Note that A1 posedge match signal from cycle **n+1** occurs at the same time as B1 negedge match signal from cycle **n**. This allows to use the A1 posedge match to mask the B1 negedge match when they occur at the same time. The result is that no transition occurs on the output flip-flop and a 0% duty cycle is generated.

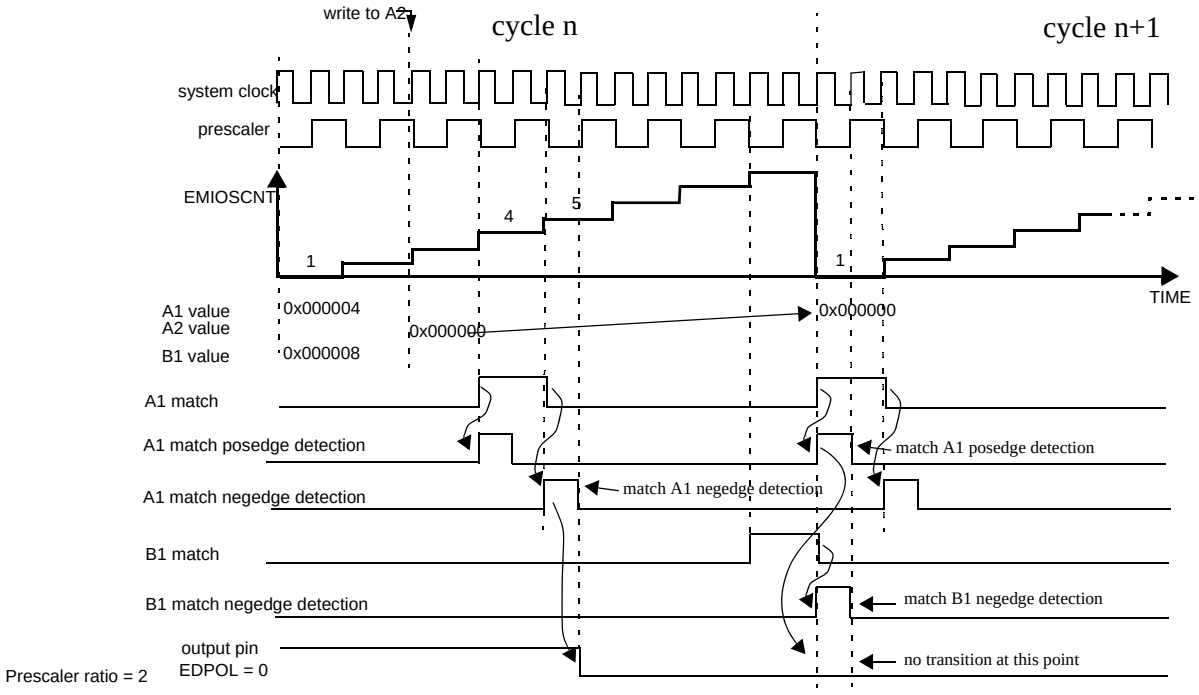



**Figure 17-26. OPWFMB Mode with A1 = 0 (0% duty cycle)**

Figure 17-27 describes the timing for the A1 and B1 registers load. The A1 and B1 load use the same signal which is generated at the last system clock period of a counter cycle. Thus, A1 and B1 are updated respectively with A2 and B2 values at the same time that the counter (CCNTR[n]) is loaded with 0x1. This event is defined as the cycle boundary. The load signal pulse has the duration of one system clock period. If A2 and B2 are written within cycle **n** their values are available at A1 and B1, respectively, at the first clock of cycle **n+1** and the new values are used for matches at cycle **n+1**. The update disable bits OU[n] of OUDR can be used to control the update of these registers, thus allowing to delay the A1 and B1 registers update for synchronization purposes.

In Figure 17-27 it is assumed that both the channel and global prescalers are set to 0x1 (each divide ratio is two), meaning that the channel internal counter transitions at every four system clock cycles. FLAGs can be generated only on B1 matches when MODE[5] is cleared, or on both A1 and B1 matches when MODE[5] is set. Since B1 flag occurs at the cycle boundary, this flag can be used to indicate that A2 or B2 data written on cycle **n** were loaded to A1 or B1, respectively, thus generating matches in cycle **n+1**. Note that the FLAG has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

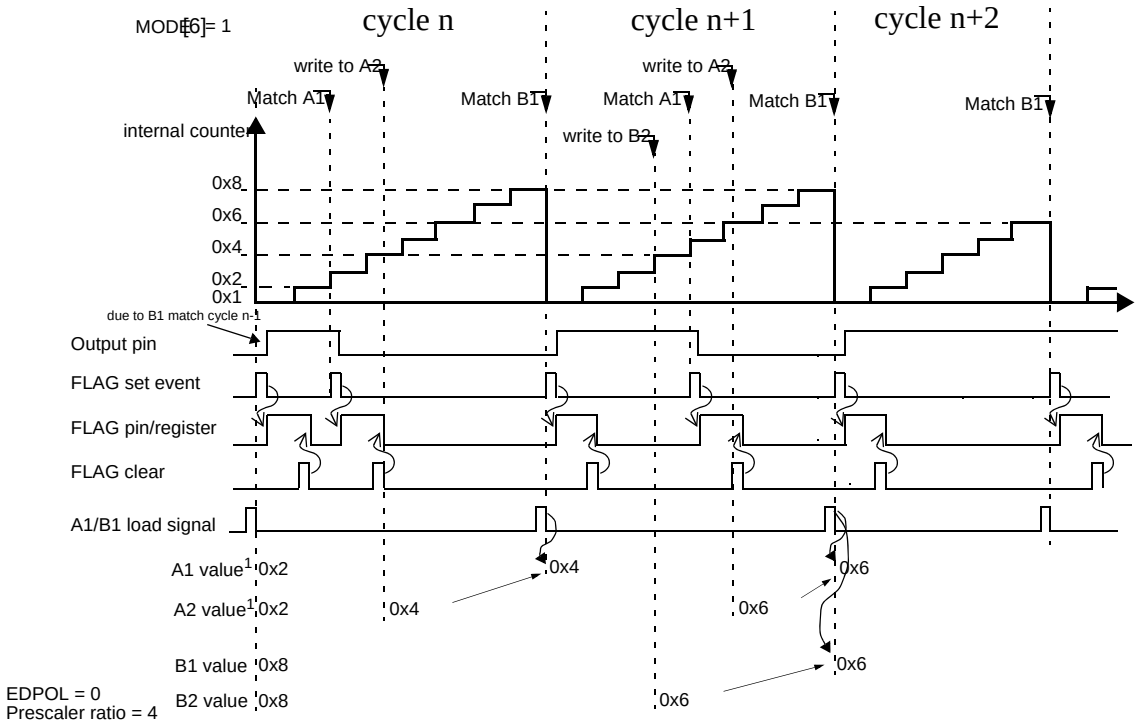



**Figure 17-27. OPWFMB A1 and B1 Registers Update and Flags**

Figure 17-28 describes the operation of the Output Disable feature in OPWFMB mode. The output disable forces the channel output flip-flop to EDPOL bit value. This functionality targets applications that use active high signals and a high to low transition at A1 match. In this case EDPOL should be set to 0. Note that both the channel and global prescalers are set to 0x0 (each divide ratio is one), meaning that the channel internal counter transitions at every system clock cycle.

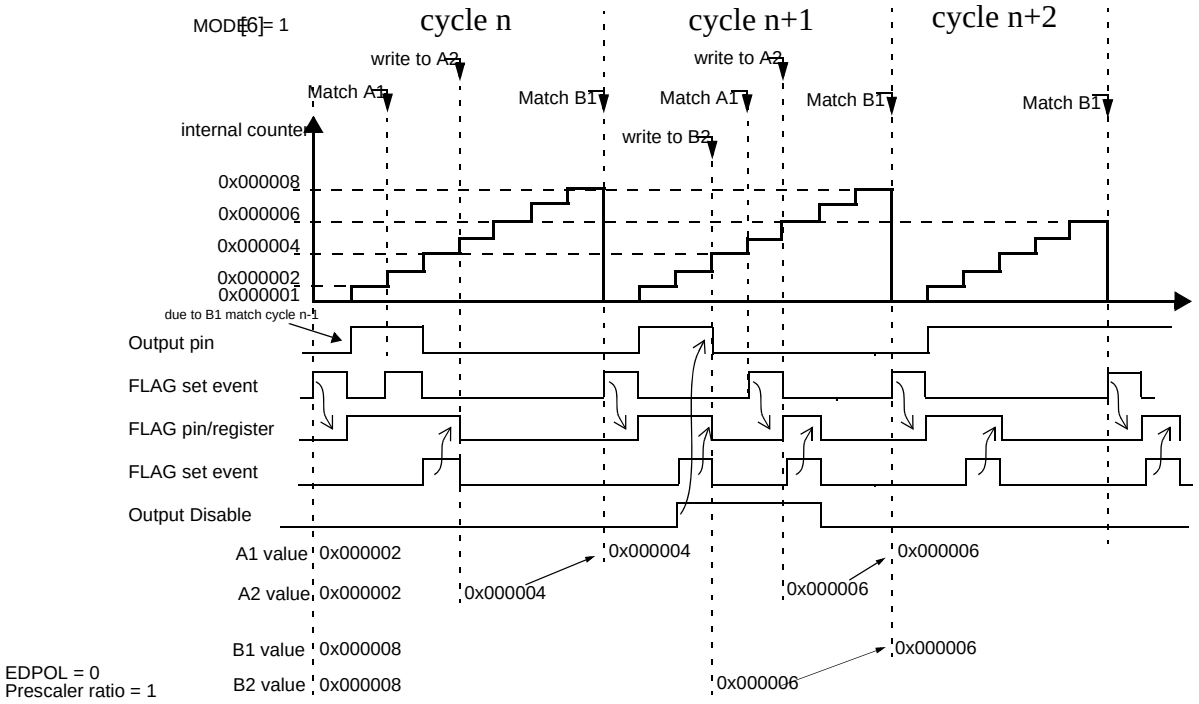



**Figure 17-28. OPWFMB Mode with Active Output Disable**

Note that the output disable has a synchronous operation, meaning that the assertion of the Output Disable input pin causes the channel output flip-flop to transition to EDPOL at the next system clock cycle. If the Output Disable input is deasserted the output pin transition at the following A1 or B1 match.

In Figure 17-28 it is assumed that the Output Disable input is enabled and selected for the Channel. See Section 17.6.2.8, “eMIOS200 UC Control Register (CCR[n]),” for a detailed description about the ODIS and ODISSL bits, respectively enable and selection of the Output Disable inputs.

The FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on comparators A or B respectively. Similarly to a B1 match FORCMB sets the internal counter to 0x1. The FLAG bit is not set by the FORCMA or FORCMB bits being asserted.

Figure 17-29 describes the generation of 100% and 0% duty cycle signals. It is assumed EDPOL =0 and the resultant prescaler value is 1. Initially A1=0x8 and B1=0x8. In this case, B1 match has precedence over A1 match, thus the output flip-flop is set to the complement of EDPOL bit. This cycle corresponds to a 100% duty cycle signal. The same output signal can be generated for any A1 value greater or equal to B1.

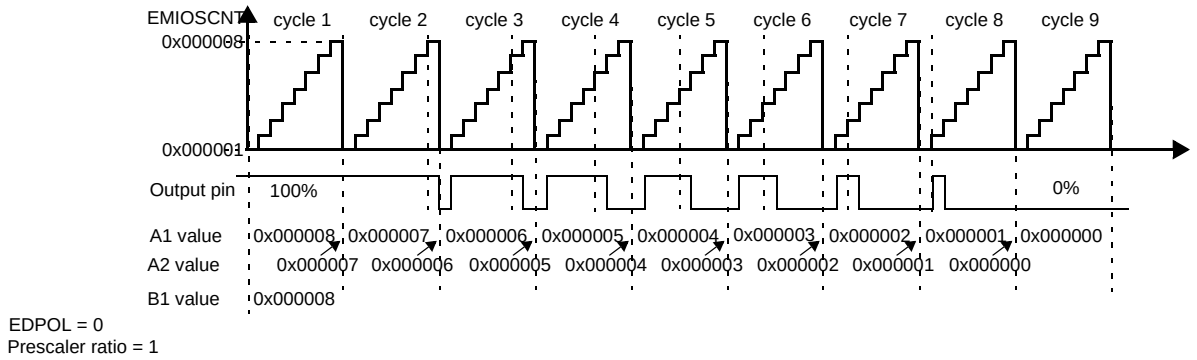



**Figure 17-29. OPWFMB Mode from 100% to 0% Duty Cycle**

A 0% duty cycle signal is generated if A1=0x0 as shown in Figure 17-29 cycle 9. In this case B1=0x8 match from cycle 8 occurs at the same time as the A1=0x0 match from cycle 9. See Figure 17-26 for a description of the A1 and B1 match generation. In this case A1 match has precedence over B1 match and the output signal transitions to EDPOL.

#### 17.7.1.1.6 Output Pulse Width Modulation Buffered (OPWMB) Mode

OPWMB mode (MODE[0:6]=11000b0) is used to generate pulses with programmable leading and trailing edge placement. An external counter driven in MCB Up mode or OPWFMB mode must be selected from one of the counter buses. A1 register value defines the first edge and B1 the second edge. The output signal polarity is defined by the EDPOL bit. If EDPOL is zero, a negative edge occurs when A1 matches the selected counter bus; and a positive edge occurs when B1 matches the selected counter bus.

The A1 and B1 registers are double buffered and updated from A2 and B2, respectively, at the cycle boundary. The load operation is similar to the OPWFMB mode. See Figure 17-27 for more information about A1 and B1 registers update.

FLAG can be generated at B1 matches, when MODE[5] is cleared, or in both A1 and B1 matches, when MODE[5] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A1 or B1 respectively. FLAG bit is not set by the FORCMA and FORCMB operations.

At OPWMB mode entry the output flip-flop is set to the value of the EDPOL bit in the CCR[n].

Following are described some rules applicable to the OPWMB mode:

- B1 matches have precedence over A1 matches if they occur at the same time within the same counter cycle
- A1=0 match from cycle **n** has precedence over B1 match from cycle **n-1**
- A1 matches are masked out if they occur after B1 match within the same cycle
- Any value written to A2 or B2 on cycle **n** is loaded to A1 and B1 registers at the following cycle boundary (assuming OU[n] bit of OUDR is not asserted). Thus the new values will be used for A1 and B1 matches in cycle **n+1**

Figure 17-30 describes the operation of the OPWMB mode regarding A1 and B1 matches and the transition of the channel output pin. In this example EDPOL is set to zero.

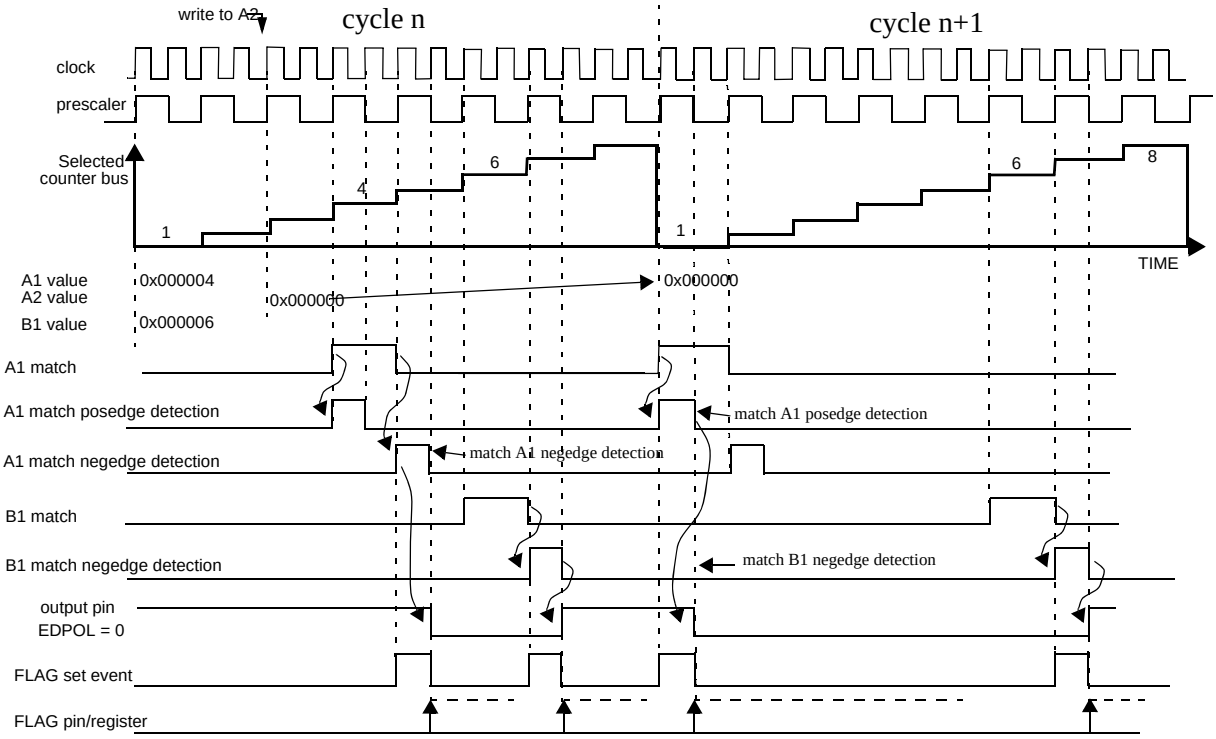



**Figure 17-30. OPWMB Mode Matches and Flags**

Note that the output pin transitions are based on the negedges of the A1 and B1 match signals. Figure 17-30 shows in cycle **n+1** the value of A1 register being set to zero. In this case the match posedge is used instead of the negedge to transition the output flip-flop.

Figure 17-31 describes the channel operation for 0% duty cycle. Note that the A1 match posedge signal occurs at the same time as the B1=0x8 negedge signal. In this case A1 match has precedence over B1 match, causing the output pin to remain at EDPOL bit value, thus generating a 0% duty cycle signal.

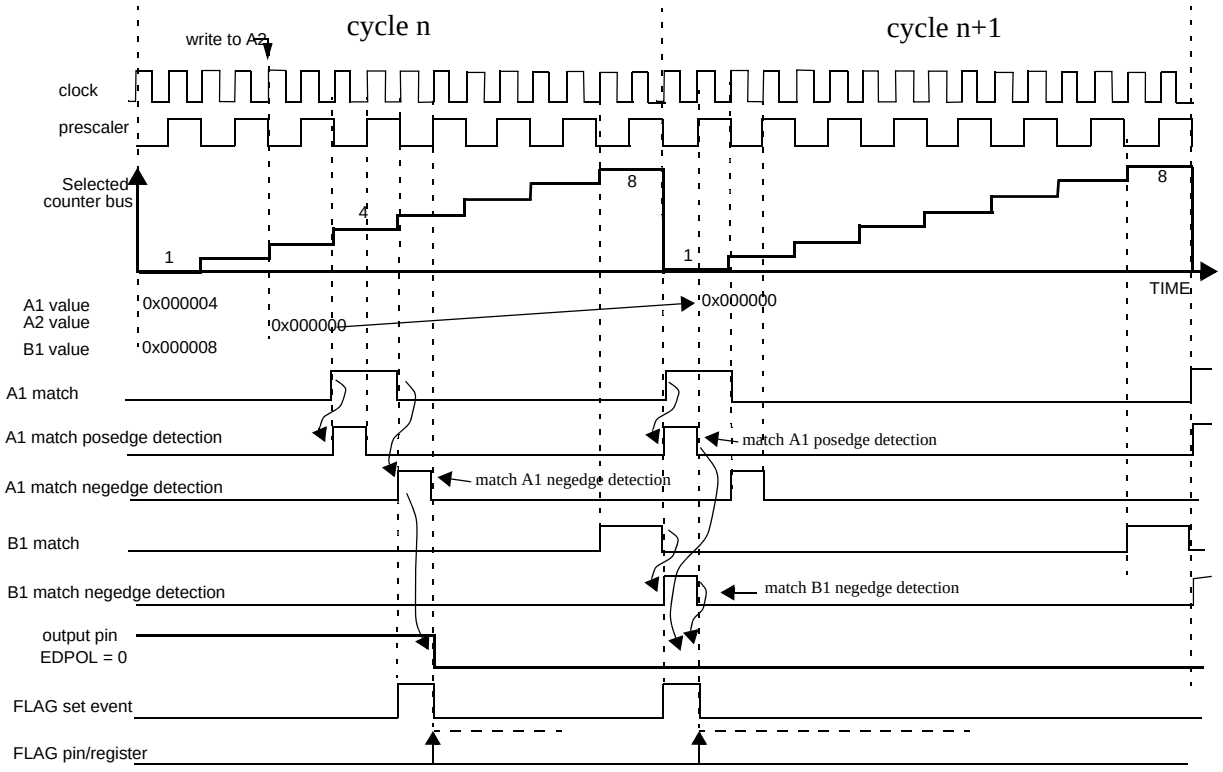



**Figure 17-31. OPWMB Mode with 0% Duty Cycle**

Figure 17-32 describes the operation of the OPWMB mode with the Output Disable signal being asserted. The Output Disable forces a transition in the output pin to the EDPOL bit value. After deasserted, the output disable allows the output pin to transition at the following A1 or B1 match. Note that the Output Disable does not modify the Flag bit behavior. Note that there is one system clock delay between the assertion of the output disable signal and the transition of the output pin to EDPOL.

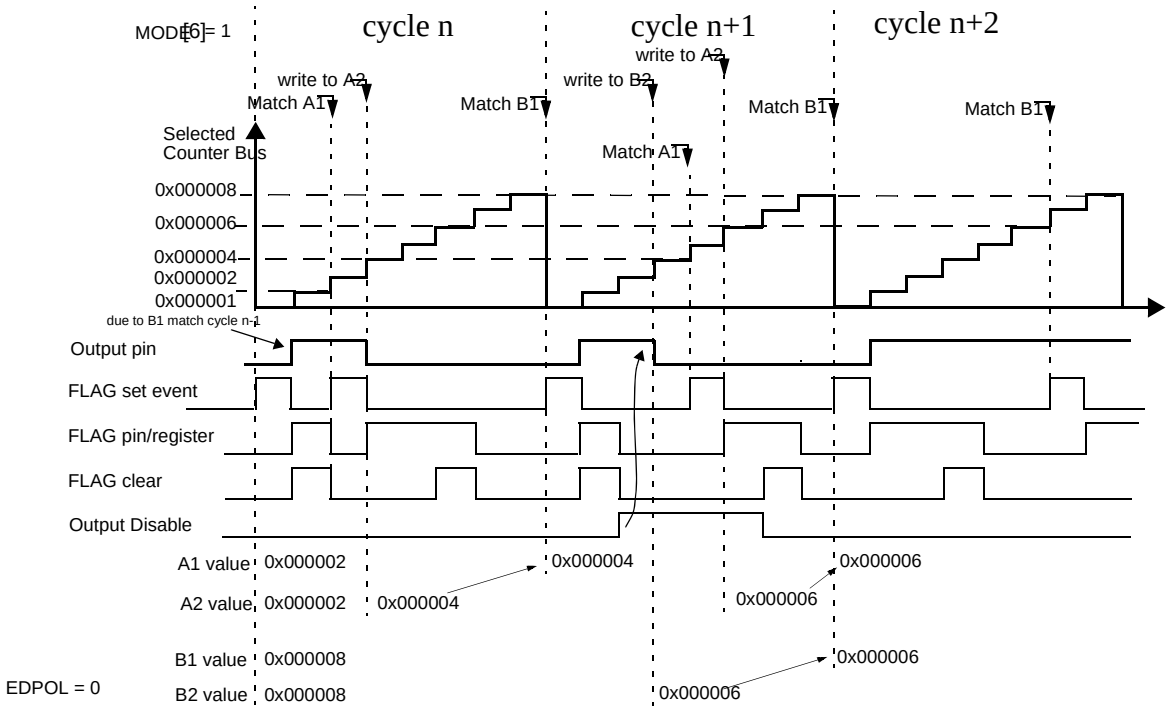



**Figure 17-32. OPWMB Mode with Active Output Disable**

Figure 17-33 shows a waveform changing from 100% to 0% duty cycle. EDPOL in this case is zero. In this example B1 is programmed to the same value as the period of the external selected time base.

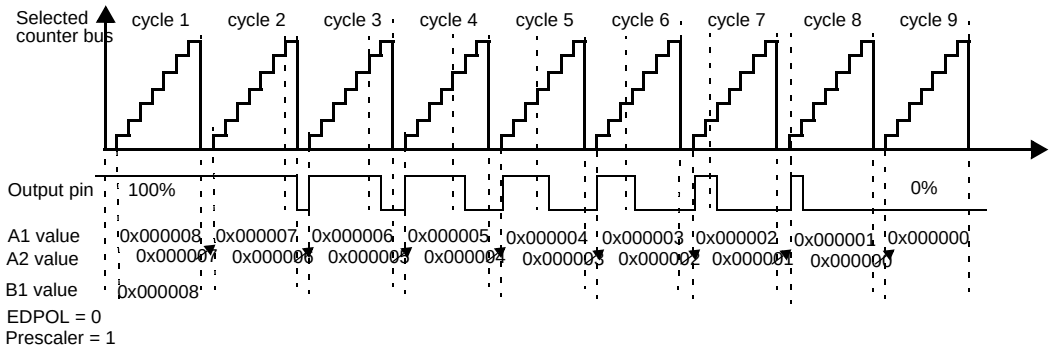



**Figure 17-33. OPWMB Mode from 100% to 0% Duty Cycle**

In Figure 17-33 if B1 is set to a value lower than 0x8 it is not possible to achieve 0% duty cycle by only changing A1 register value. Since B1 matches have precedence over A1 matches the output pin transitions to the opposite of EDPOL bit at B1 match. Note also that if B1 is set to 0x9, for instance, B1 match does not occur, thus a 0% duty cycle signal is generated.

### 17.7.1.1.7 Quadrature Decode (QDEC) Mode

Quadrature decode mode uses UC[n] operating in QDEC mode and the input programmable filter (IPF) from UC[n-1]. Note that UC[n-1] can be configured, at the same time, to an operation mode that does not

use I/O pins, such as MC mode (modulus counter). The connection among the UCs is circular, i.e., when UC[0] is running in QDEC mode, the input programmable filter from UC[23] is being used.

This mode generates a FLAG every time the internal counter matches A1 register. The internal counter is automatically selected and is not cleared when entering this mode.

MODE[] bit selects which type of encoder will be used: *count & direction* encoder or *phase_A & phase_B* encoders.

When operating with *count & direction* encoder (MODE[] cleared), UC[n] input pin must be connected to the *direction* signal and UC[n-1] input pin must be connected to the *count* signal of the quadrature encoder. UC[n] EDPOL bit selects count direction according to *direction* signal and UC[n-1] EDPOL bit selects if the internal counter is clocked by the rising or falling edge of the *count* signal.

When operating with *phase_A & phase_B* encoder (MODE[] set), UC[n] input pin must be connected to the *phase_A* signal and UC[n-1] input pin must be connected to the *phase_B* signal of the quadrature encoder. EDPOL bit selects the count direction according to the phase difference between *phase_A & phase_B* signals.

Figure 17-34 and Figure 17-35 show two Unified Channels configured to quadrature decode mode for *count & direction* encoder and *phase_A & phase_B* encoders, respectively.

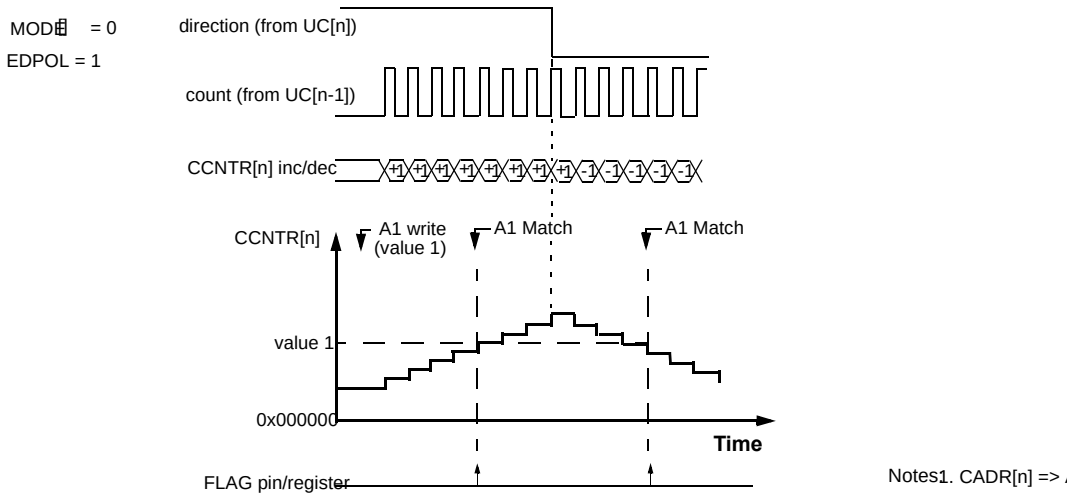



**Figure 17-34. Quadrature Decode mode example with *count & direction* encoder**

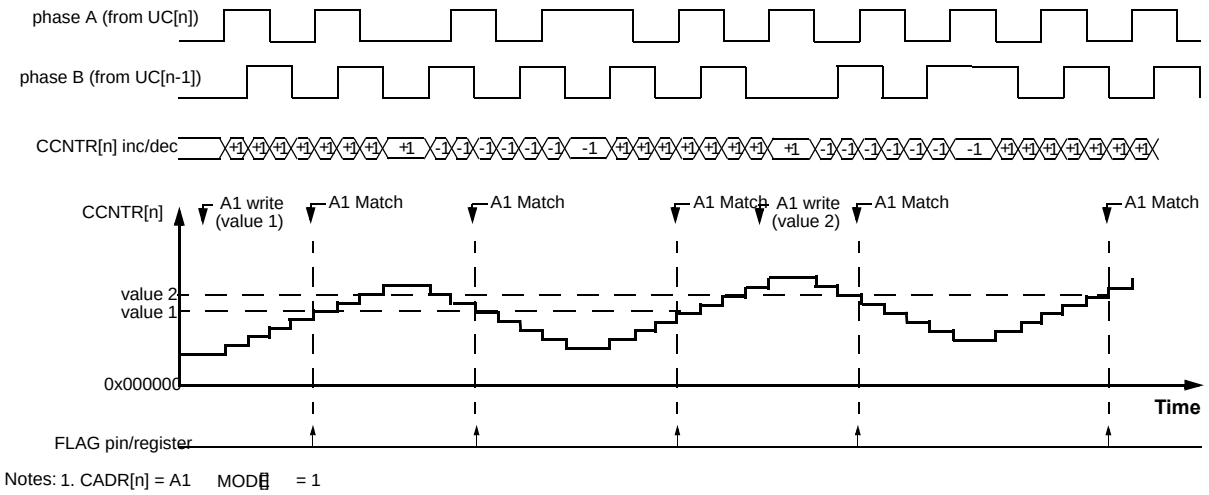



**Figure 17-35. Quadrature Decode mode example with *phase_a & phase_B* encoder**

### 17.7.1.2 Input Programmable Filter (IPF)

The IPF ensures that only valid input pin transitions are received by the Unified Channel edge detector. A block diagram of the IPF is shown in Figure 17-36.

The IPF is a 5-bit programmable up counter that is incremented by the selected clock source, according to bits IF[0:3] in CCR[n].

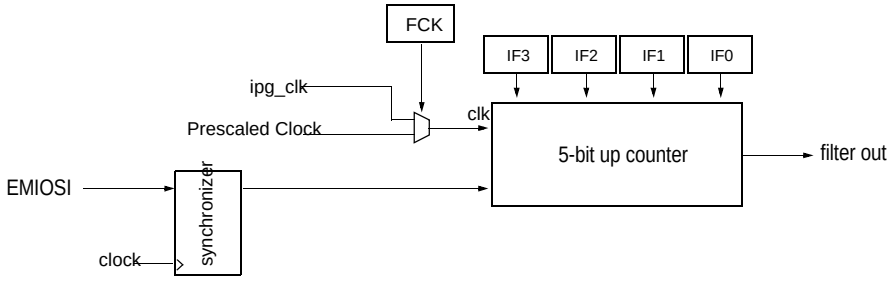



**Figure 17-36. Input Programmable Filter Submodule Diagram**

The input signal is synchronized by system clock. When a state change occurs in this signal, the 5-bit counter starts counting up. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflows occurs, the new pin value is validated. In this case, it is transmitted as a pulse edge to the edge detector. If the opposite edge appears on the pin before validation (overflow), the counter is reset. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the edge detector. A timing diagram of the input filter is shown in Figure 17-37.

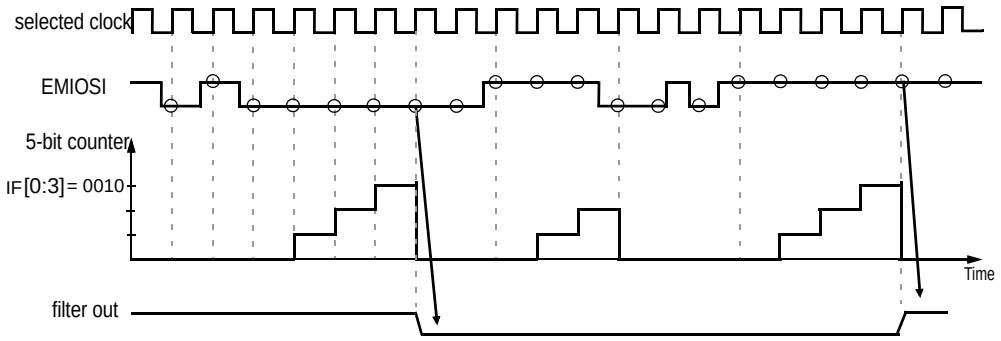



**Figure 17-37. Input Programmable Filter Example**

The filter is not disabled during freeze state.

### 17.7.1.3 Clock Prescaler (CP)

The CP divides the GCP output signal to generate a clock enable for the internal counter of the Unified Channels. The GCP output signal is prescaled by the value defined in Figure 17-14 according to the UCPRE[0:1] bits in CCR[n]. The prescaler is enabled by setting the UCPREN bit in the CCR[n] and can be stopped at any time by clearing this bit, thereby stopping the internal counter in the Unified Channel.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 at both MCR[GPREN] and UCPREN bit in CCR[n], thus disabling prescalers;
2. Write the desired value for prescaling rate at UCPRE[0:1] bits in CCR[n];
3. Enable channel prescaler by writing 1 at UCPREN bit in CCR[n];
4. Enable global prescaler by setting MCR[GPREN].

The prescaler is not disabled during freeze state.

### 17.7.1.4 Effect of Freeze on the Unified Channel

When in debug mode, MCR[FRZ] and the FREN bit in the CCR[n] are both set, the internal counter and Unified Channel capture and compare functions are halted. The UC is frozen in its current state.

During freeze, all registers are accessible. When the Unified Channel is operating in an output mode, the force match functions remain available, allowing the software to force the output to the desired level.

Note that for input modes, any input events that may occur while the channel is frozen are ignored.

When exiting debug mode or freeze enable bit is cleared (MCR[FRZ] or FREN in the CCR[n]) the channel actions resume, but may be inconsistent until channel enters GPIO mode again.

## 17.7.2 Global Clock Prescaler Submodule (GCP)

The GCP divides the system clock to generate a clock for the CPs of the channels. The main clock signal is prescaled by the value defined in Figure 17-8 according to the GPRE[0:7] bits in MCR. The global

prescaler is enabled by setting the MCR[GPREN] bit and can be stopped at any time by clearing this bit, thereby stopping the internal counters in all the channels.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any updtate in the prescaling rate is desired:

1. Write MCR[GPREN] = 0, thus disabling global prescaler;
2. Write the desired value for prescaling rate at GPRE[0:7] bits in MCR;
3. Enable global prescaler by setting MCR[GPREN].

The prescaler is not disabled during freeze state.

#### 17.7.2.1 Effect of Freeze on the GCP

When the MCR[FRZ] bit is set and the module is in debug mode, the operation of GCP submodule is not affected, i.e., there is no freeze function in this submodule.

## 17.8 Initialization/application information

On resetting the eMIOS200 the Unified Channels enter GPIO input mode.

### 17.8.1 Considerations

Before changing an operating mode, the UC must be programmed to GPIO mode and CADR[n] and CBDR[n] must be updated with the correct values for the next operating mode. Then the CCR[n] can be written with the new operating mode. If a UC is changed from one mode to another without performing this procedure, the first operation cycle of the selected time base can be random, i.e., matches can occur in random time if the contents of CADR[n] or CBDR[n] were not updated with the correct value before the time base matches the previous contents of CADR[n] or CBDR[n].

When interrupts are enabled, the software must clear the FLAG bits before exiting the interrupt service routine.

### 17.8.2 Application Information

Correlated output signals can be generated by all output operation modes. Bits OU[n] of OUDR can be used to control the update of these output signals.

In order to guarantee that the internal counters of correlated channels are incremented in the same clock cycle, the internal prescalers must be set up before enabling the global prescaler. If the internal prescalers are set after enabling the global prescaler, the internal counters may increment in the same ratio, but at a different clock cycle.

It is recommended to drive Output Disable Input signals with the emios_flag_out signals of some UCs running in SAIC mode. When an output disable condition happens, the software interrupt routine must service the output channels before servicing the channels running SAIC. This procedure avoid glitches in the output pins.

### 17.8.2.1 Time Base Generation

For the OPWFM with internal clock source operation mode, the internal counter rate can be modified by configuring the clock prescaler ratio. Figure 17-38 shows an example of a time base with prescaler ratio equal to one.

**NOTE**

MCB and OPWFMB modes have a different behavior.

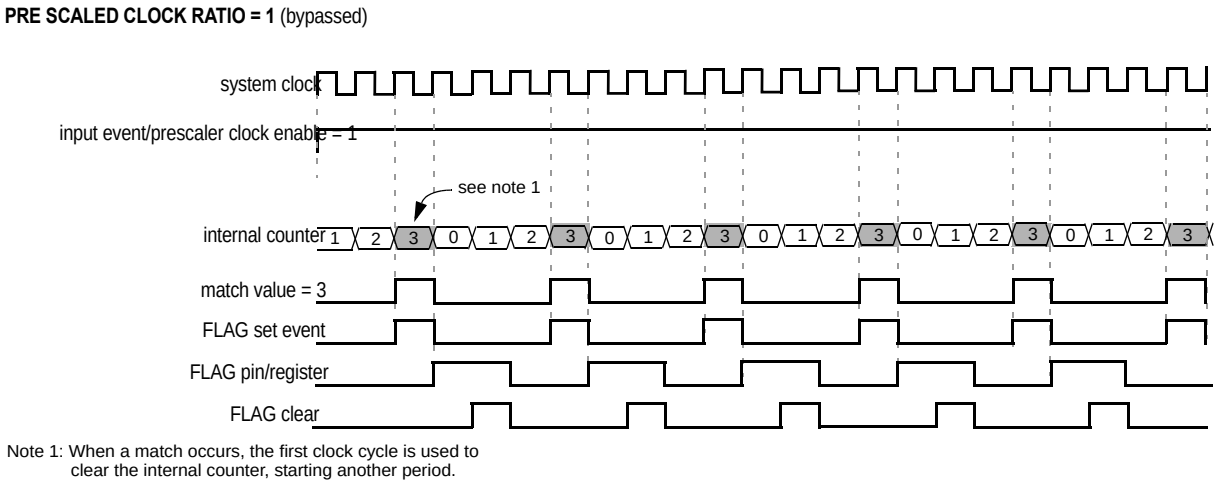


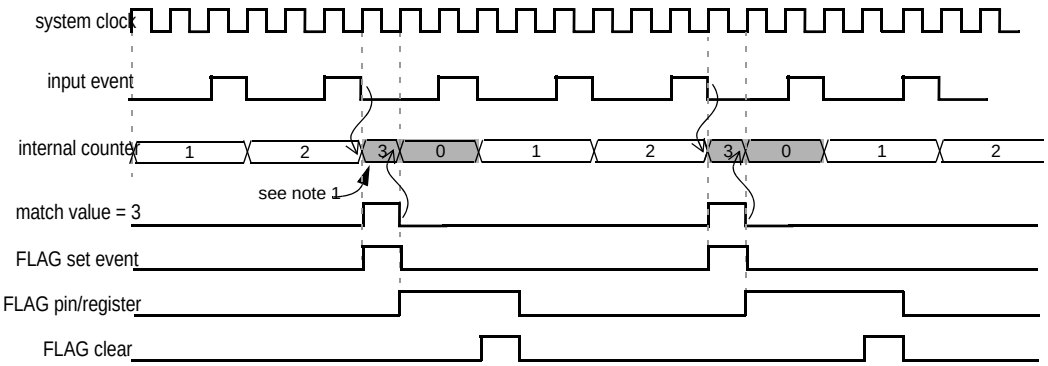

**Figure 17-38. Time base period when running in the fastest prescaler ratio**

If the prescaler ratio is greater than one or external clock is selected, the counter may behave in three different ways depending on the channel mode:

- If MC mode and Clear on Match Start and External Clock source are selected the internal counter behaves as described in Figure 17-39.
- If MC mode and Clear on Match Start and Internal Clock source are selected the internal counter behaves as described in Figure 17-40.
- If MC mode and Clear on Match End are selected the internal counter behaves as described in Figure 17-41.
- If OPWFM mode is selected the internal counter behaves as described in Figure 17-40. The internal counter clears at the start of the match signal, skips the next prescaled clock edge and then increments in the subsequent prescaled clock edge.

**NOTE**

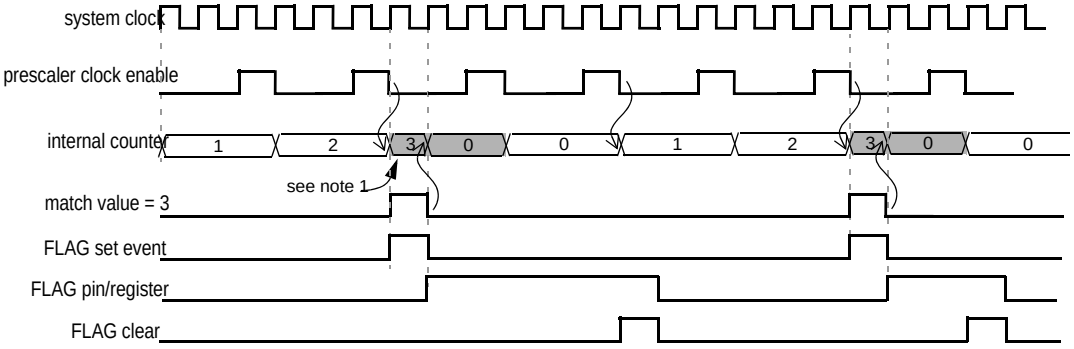
MCB and OPWFMB modes have a different behavior.




Note 1: When a match occurs, the first system clock cycle is used to clear the internal counter, and at the next edge of prescaler clock enable the counter will start counting.

**Figure 17-39. Time base generation with external clock and clear on match start**

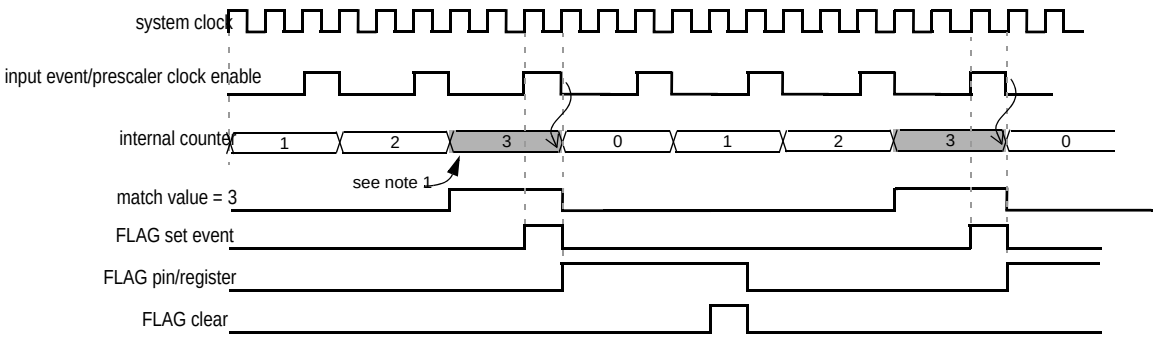
**PRESCALED CLOCK RATIO = 3**




Note 1: When a match occurs, the first clock cycle is used to clear the internal counter, and only after a second edge of pre scaled clock the counter will start counting.

**Figure 17-40. Time base generation with internal clock and clear on match start**

**PRESCALED CLOCK RATIO = 3**




Note 1: The match occurs only when the input event/prescaler clock enable is active. Then, the internal counter is immediately cleared.

**Figure 17-41. Time base generation with clear on match end**

### 17.8.2.2 Coherent Accesses

The FLAG set event can be detected by polling the FLAG bit or by enabling the interrupt or DMA request generation.

Reading the CADR[n] again in the same period of the last read of CBDR[n] may lead to incoherent results. This will occur if the last read of CBDR[n] occurred after a disabled B2 to B1 transfer.

### 17.8.2.3 Channel/Modes Initialization

The following basic steps summarize basic output mode startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler;
2. *[timebase channel]* Disable Channel Prescaler;
3. *[timebase channel]* Write initial value at internal counter;
4. *[timebase channel]* Set A/B register;
5. *[timebase channel]* Set channel to MC(B) Up mode;
6. *[timebase channel]* Set prescaler ratio;
7. *[timebase channel]* Enable Channel Prescaler;
8. *[output channel]* Disable Channel Prescaler;
9. *[output channel]* Set A/B register;
10. *[output channel]* Select timebase input through BSL[1:0] bits;
11. *[output channel]* Enter output mode;
12. *[output channel]* Set prescaler ratio (same ratio as timebase channel);
13. *[output channel]* Enable Channel Prescaler;
14. *[global]* Enable Global Prescaler.

The timebase channel and the output channel may be the same for some applications such as in OPWFM(B) mode or whenever the output channel is intended to run the timebase itself.

At any time the flags can be configured.

# Chapter 18
# Enhanced Direct Memory Access (eDMA)

## 18.1 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes a DMA engine that performs source- and destination-address calculations, and the actual data-movement operations, along with local memory containing transfer control descriptors (TCD) for each channel.

Figure 18-1 is a block diagram of the DMA module.

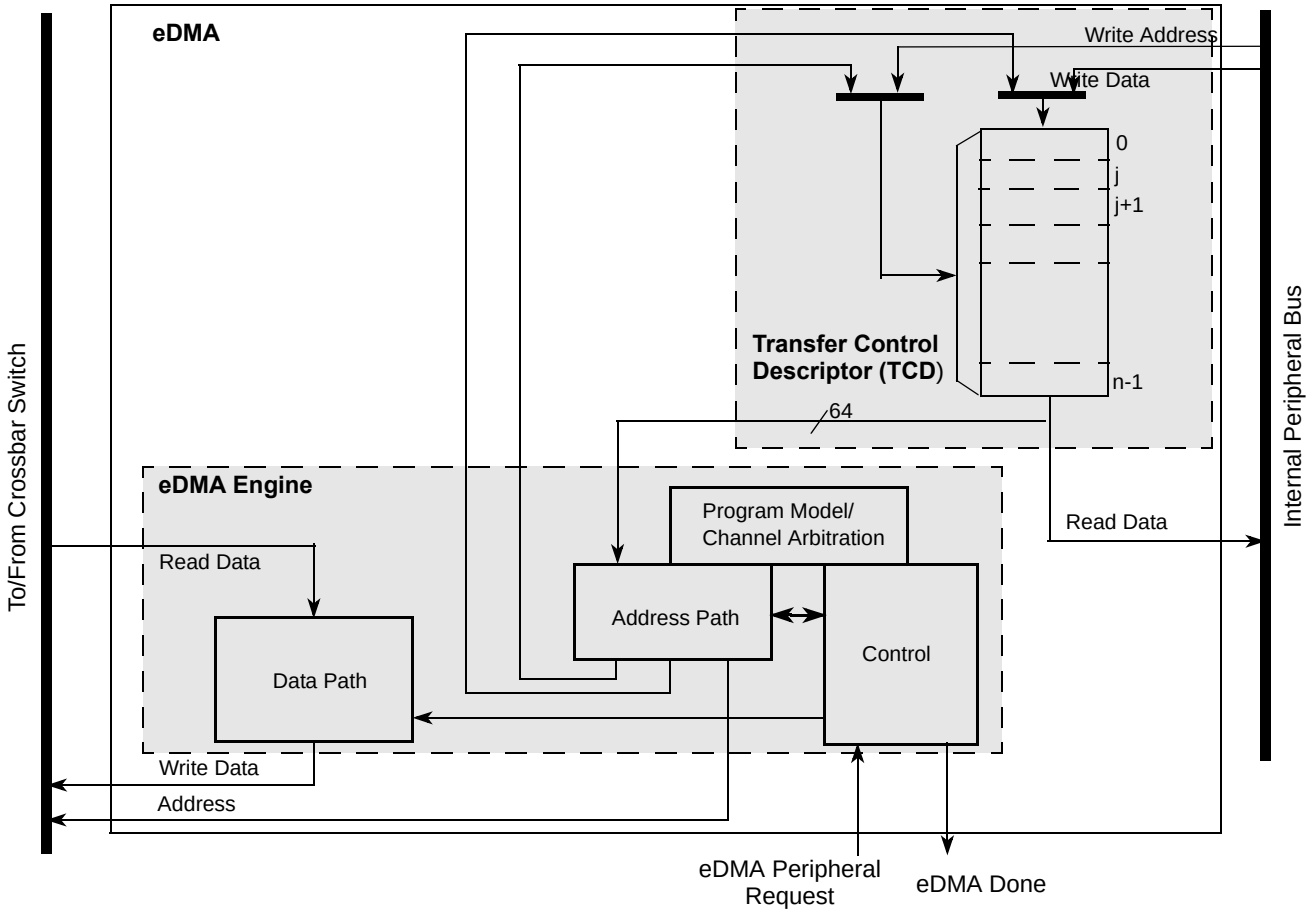



**Figure 18-1. DMA block diagram**

### 18.1.1 Overview

The DMA is a highly-programmable data transfer engine, which has been optimized to minimize the required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known, and is *not* defined within the data packet itself.

Throughout this document, *n* is used to reference the channel number. Additionally, data sizes are defined as byte (8-bit), halfword (16-bit), word (32-bit) and doubleword (64-bit).

### 18.1.2 Features

The DMA module supports the following features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source, destination addresses, transfer size, plus support for enhanced addressing modes
- -channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage to support 16-byte burst transfers
  - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor organized to support two-deep, nested transfer operations
  - An *inner* data transfer loop defined by a “minor” byte transfer count
  - An *outer* data transfer loop defined by a “major” iteration count
- Channel service request via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
    – Independent channel linking at end of minor loop and/or major loop
  - Peripheral-paced hardware requests (one per channel)
  - For all three methods, *one service request per execution of the minor loop is required*
- Support for fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests
  - One interrupt per channel, optionally asserted at completion of major iteration count
  - Error terminations are optionally enabled per channel, and logically summed together to form a small number of error interrupt outputs
- Support for scatter/gather DMA processing
- Support for complex data structures
- Support to cancel transfers via software or hardware

The structure of the transfer control descriptor is fundamental to the operation of the DMA module. It is defined below in a ‘C’ pseudo-code specification, where `int` refers to a 32-bit variable (unless noted otherwise) and `short` is a 16-bit variable.

**NOTE**

To compile these structures, change any periods '.' in the variable name to underscores '_'.

```
typedef union {
    struct {                                /* citer.e_link = 1 */
        unsigned short citer.linkch:6;      /* link channel number, */
        unsigned short citer:9;             /* current (“major”) iteration count */
    } minor_link_enabled;                   /* channel link at end of the minor loop */
    struct {                                /* citer.e_link = 0 */
        unsigned short citer:15;            /* current (“major”) iteration count */
    } minor_link_disabled;                  /* no linking at end of the minor loop */
} t_minor_link_citer;

typedef union {
    struct {                                /* biter.e_link = 1 */
        unsigned short biter.linkch:6;      /* link channel number, */
        unsigned short biter:9;             /* beginning (“major”) iteration count */
    } init_minor_link_enabled;              /* channel link at end of the minor loop */
    struct {                                /* biter.e_link = 0 */
        unsigned short biter:15;            /* beginning (“major”) iteration count */
    } init_minor_link_disabled;             /* no linking at end of the minor loop */
} t_minor_link_biter;

typedef struct {
         unsigned intsaddr;                 /* source address */
         unsigned intsmod:5;                /* source address modulo */
         unsigned intssize:3;               /* source transfer size */
         unsigned intdmod:5;                /* destination address modulo */
         unsigned intdsize:3;               /* destination transfer size */
         short    soff;                     /* signed source address offset */
         unsigned intnbytes;                /* inner (“minor”) byte count */
         int      slast;                    /* last source address adjustment */
         unsigned intdaddr;                 /* destination address */
         unsigned shortciter.e_link:1;      /* enable channel linking on minor loop */
         t_minor_link_citerminor_link_citer;/* conditional current iteration count */
         short    doff;                     /* signed destination address offset */
         int      dlast_sga;                /* last destination address adjustment, or
                                               scatter/gather address (if e_sg = 1) */
         unsigned shortbiter.e_link:1;      /* beginning channel link enable */
         t_minor_link_biterminor_link_biter;/* beginning (“major”) iteration count */
         unsigned intbwc:2;                 /* bandwidth control */
         unsigned intmajor.linkch:6;        /* link channel number */
         unsigned intdone:1;                /* channel done */
         unsigned intactive:1;              /* channel executing */
         unsigned intmajor.e_link:1;        /* enable channel linking on major loop*/
         unsigned inte_sg:1;                /* enable scatter/gather descriptor */
         unsigned intd_req:1;               /* disable ipd_req when done */
         unsigned intint_half:1;            /* interrupt on citer = (biter >> 1) */
         unsigned intint_maj:1;             /* interrupt on major loop completion */
         unsigned intstart:1;               /* explicit channel start */
} tcd                                       /* transfer_control_descriptor */
```

The basic operation of a channel is defined as:

1. The channel is initialized by software loading the transfer control descriptor into the eDMA’s programming model, memory-mapped through the slave bus space, and implemented as local memory.

2. The channel requests service; either explicitly by software, a peripheral request or a linkage from another channel.

**NOTE**

The major loop executes one iteration per service request.

3. The contents of the transfer control descriptor for the activated channel is read from the local memory and loaded into the eDMA engine's internal register file.

4. The eDMA engine executes the data transfer defined by the transfer control descriptor, reading from the source and writing to the destination. The number of iterations in the minor loop is automatically calculated by the eDMA engine. The number of iterations within the minor loop is a function of the number of bytes to transfer (nbytes), the source size (ssize) and the destination size (dsize). The completion of the minor loop is equal to one iteration of the major loop.

5. At the conclusion of the minor loop's execution, certain fields of the transfer control descriptor are written back to the local TCD memory.

The process (steps 2-5) is repeated until the outer major loop's iteration count is exhausted. At that time, additional processing steps are completed, e.g., the optional assertion of an interrupt request signaling the transfer's completion, final adjustments to the source and destination addresses, etc. A more detailed description of the channel processing is listed in the pseudo-code below. This simplified example is intended to represent basic data transfers. Detailed processing associated with the error handling is omitted.

```
/* the given eDMA channel is requesting service by the software assertion of the
   tcd[channel].start bit, the assertion of an enabled ipd_req from a device, or
   the implicit assertion of a channel-to-channel link */

/* begin by reading the transfer control descriptor from the local RAM
   into the local dma_engine registers */
dma_engine        = read_from_local_memory [channel];
dma_engine.active = 1;                 /* set active flag */
dma_engine.done   = 0;                 /* clear done flag */

/* check the transfer control descriptor for consistency */
if (dma_engine.config_error == 0) {

    / * begin execution of the inner "minor" loop transfers */
    {

        /* convert the source transfer size into a byte count */
        switch (dma_engine.ssize) {
        case 0:                         /* 8-bit transfer */
            src_xfr_size = 1;
            break;
        case 1:                         /* 16-bit transfer */
            src_xfr_size = 2;
            break;
        case 2:                         /* 32-bit transfer */
            src_xfr_size = 4;
            break;
        case 3:                         /* 64-bit transfer */
            src_xfr_size = 8;
```

```
    break;
case 4:                         /* 16-byte burst transfer */
    src_xfr_size = 16;
    break;
case 5:                         /* 32-byte burst transfer */
    src_xfr_size = 32;
    break;
}

/* convert the destination transfer size into a byte count */
switch (dma_engine.dsize) {
case 0:                         /* 8-bit transfer */
    dest_xfr_size = 1;
    break;
case 1:                         /* 16-bit transfer */
    dest_xfr_size = 2;
    break;
case 2:                         /* 32-bit transfer */
    dest_xfr_size = 4;
    break;
case 3:                         /* 64-bit transfer */
    dest_xfr_size = 8;
    break;
case 4:                         /* 16-byte burst transfer */
    dest_xfr_size = 16;
    break;
case 5:                         /* 32-byte burst transfer */
    dest_xfr_size = 32;
    break;
}

/* determine the larger of the two transfer sizes, this value reflects */
/* the number of bytes transferred per read->write sequence. */
/* number of iterations of the minor loop = nbytes / xfer_size */
if (dma_engine.ssize < dma_engine.dsize)
    xfr_size = dest_xfer_size;
 else
    xfr_size = src_xfer_size;

/* process the source address, READ data into the buffer*/

/* read "xfr_size" bytes from the source */
/* if the ssize < dsize, do multiple reads to equal the dsize */
/* if the ssize => dsize, do a single read of source data */
number_of_source_reads = xfer_size / src_xfer_size;

for (number_of_source_reads) {
    dma_engine.data = read_from_amba-ahb (dma_engine.saddr, src_xfr_size);

    /* generate the next-state source address */
    /* sum the current saddr with the signed source offset */
    ns_addr = dma_engine.saddr + (int) dma_engine.soff; }

    /* if enabled, apply the power-of-2 modulo to the next-state addr */
    if (dma_engine.smod != 0)
        address_select = (1 << dma_engine.smod) - 1; }
    else
```

```
                address_select = 0xffff_ffff;

            dma_engine.saddr = ns_addr           &  address_select
                        | dma_engine.saddr & ~address_select; }
        }


        /* process the destination address, WRITE data from buffer */

        /* write "xfr_size" bytes to the destination */
        /* if the dsize < ssize, do multiple writes to equal the ssize */
        /* if the dsize => ssize, do a single write of dest data */
        number_of_dest_writes = xfer_size / dest_xfer_size;

        for (number_of_dest_writes) {
            write_to_amba-ahb (dma_engine.daddr, dest_xfr_size) = dma_engine.data;

            /* generate the next-state destination address */
            /* sum the current daddr with the signed destination offset */
            ns_addr = dma_engine.daddr + (int) dma_engine.doff;

            /* if enabled, apply the power-of-2 modulo to the next-state dest addr */
            if (dma_engine.dmod != 0)
                address_select = (1 << dma_engine.dmod) - 1;
            else
                address_select = 0xffff_ffff;

            dma_engine.daddr = ns_addr           &  address_select
                        | dma_engine.daddr & ~address_select;
        }
if (cancel_transfer)
            break;

        /* check for a higher priority channel to service if: */
        /* 1) preemption is enabled */
        /* 2) in fixed arbitration mode */
        /* 3) a higher priority channel is requesting service */
        /* 4) not already servicing a preempting channel */
        if ((DCHPRIn.ecp = 1)  &  fixed_arbitration_mode
            higher_pri_request & ~current_channel_is_preempt)
            service_preempt_channel;

        /* the bandwidth control field determines when the next read/write occurs */
        if (dma_engine.bwc > 1)
            stall_dma_engine (1 << dma_engine.bwc);

        /* decrement the minor loop byte count */
        dma_engine.nbytes = dma_engine.nbytes - xfr_size;

    }while (dma_engine.nbytes > 0) /* end of minor inner loop */

    dma_engine.citer--;            /* decrement major loop iteration count */

    /* if the major loop is not yet exhausted, update certain TCD values in the RAM */
    if (dma_engine.citer != 0) {
        write_to_local_memory [channel].saddr = dma_engine.saddr;
        write_to_local_memory [channel].daddr = dma_engine.daddr;
```

```
        write_to_local_memory [channel].citer = dma_engine.citer;

                          /* if minor loop linking is enabled, make the channel link */
        if (dma_engine.citer.e_link)
            TCD[citer.linkch].start = 1;     /* specified channel service req */

        /* check for interrupt assertion if half of the major iterations are done */
        if (dma_engine.int_half && (dma_engine.citer == (dma_engine.biter >> 1)))
            generate_interrupt (channel);

        dma_engine.active = 0;         /* clear the channel busy flag */

    }
    else { /* major loop is complete, dma_engine.citer == 0 */
      /* since the major loop is complete, perform the final address adjustments */

      /* sum the current {src,dst} addresses with "last" adjustment */
      write_to_local_memory [channel].saddr = dma_engine.saddr + dma_engine.slast;
      write_to_local_memory [channel].daddr = dma_engine.daddr + dma_engine.dlast;
      /* restore the major iteration count to the beginning value */
      write_to_local_memory [channel].citer = dma_engine.biter;

      /* check for interrupt assertion at completion of the major iteration */
      if (dma_engine.int_maj)
        generate_interrupt (channel);

       /* check if the ipd_req is to be disabled at completion of the major iteration */
      if (dma_engine.d_req)
          DMAERQ [channel] = 0;

      /* check for a scatter/gather transfer control descriptor */
      if (dma_engine.e_sg) {
        /* load new transfer control descriptor from the address defined by dlast_sga */
          write_to_local_memory [channel] =
                                    read_from_amba-ahb(dma_engine.dlast_sga,32);
      }
      if (dma_engine.major.e_link)
          TCD[major.linkch].start = 1;       /* specified channel service req */

      dma_engine.active = 0;                /* clear the channel busy flag */
      dma_engine.done = 1;                  /* set the channel done flag */

}
else { /* configuration error detected, abort the channel */
       dma_engine.error_status = error_type; /* record the error */
       dma_engine.active = 0;                /* clear the channel busy flag */
       /* check for interrupt assertion on error */
       if (dma_engine.int_err)
           generate_interrupt (channel);
}
```

For more details, consult Section 18.2.1, Register descriptions and Section 18.3, Functional description.

## 18.2 Memory map/register definition

The eDMA's programming model is partitioned into two sections, both mapped into the slave bus space: the first region defines a number of registers providing control functions, while the second region corresponds to the local transfer control descriptor memory.

Reading an *unimplemented* register bit or memory location will return the value of zero. Write the value of zero to *unimplemented* register bits. Any access to a *reserved* memory location will result in a bus error. *Reserved* memory locations are indicated in the memory map. For 16-channel implementation, reserved memory also includes the high order "H" registers containing channels 63-32 data (i.e., DMAERQH, DMAEEIH, DMAINTH, DMAERRH).

Many of the control registers have a bit width that matches the number of channels implemented in the module, i.e., 16-, 32- or 64-bits in size. Registers associated with a 64-channel design are implemented as two 32-bit registers, and include an "H" and "L" suffixes, signaling the "high" and "low" portions of the control function. The descriptions in this section define the 64-channel implementation. For 16-channel designs, the unused bits are not implemented: reads return zeroes, and writes are ignored.

The eDMA module does not include any logic which provides access control. Rather, this function is supported using the standard access control logic provided by the PBRIDGE controller.

Table 18-1 is a 32-bit view of the DMA's memory map.

**Table 18-1. eDMA 32-bit memory map**

| eDMA Offset | Register | | | |
|---|---|---|---|---|
| 0x0000 | eDMA Control Register (DMACR) | | | |
| 0x0004 | eDMA Error Status (DMAES) | | | |
| 0x0008 | eDMA Enable Request High (DMAERQH, Channels 63-32) | | | |
| 0x000c | eDMA Enable Request Low (DMAERQL, Channels 31-00) | | | |
| 0x0010 | eDMA Enable Error Interrupt High (DMAEEIH, Channels 63-32) | | | |
| 0x0014 | eDMA Enable Error Interrupt Low (DMAEEIL, Channels 31-00) | | | |
| 0x0018 | eDMA Set Enable Request (DMASERQ) | eDMA Clear Enable Request (DMACERQ) | eDMA Set Enable Error Interrupt (DMASEEI) | eDMA Clear Enable Error Interrupt (DMACEEI) |
| 0x001c | eDMA Clear Interrupt Request (DMACINT) | eDMA Clear Error (DMACERR) | eDMA Set Start Bit (DMASSRT) | eDMA Clear Done Status Bit (DMACDNE) |
| 0x0020 | eDMA Interrupt Request High (DMAINTH, Channels 63-32) | | | |
| 0x0024 | eDMA Interrupt Request Low (DMAINTL, Channels 31-00) | | | |
| 0x0028 | eDMA Error High (DMAERRH, Channels 63-32) | | | |
| 0x002c | eDMA Error Low (DMAERRL, Channels 31-00) | | | |
| 0x0030 | eDMA Hardware Request Status High (DMAHRSH, Channels 63-32) | | | |
| 0x0034 | eDMA Hardware Request Status Low (DMAHRSL, Channels 31-00) | | | |
| 0x0100 | eDMA Channel 0 Priority (DCHPRI0) | eDMA Channel 1 Priority (DCHPRI1) | eDMA Channel 2 Priority (DCHPRI2) | eDMA Channel 3 Priority (DCHPRI3) |
| 0x0104 | eDMA Channel 4 Priority (DCHPRI4) | eDMA Channel 5 Priority (DCHPRI5) | eDMA Channel 6 Priority (DCHPRI6) | eDMA Channel 7 Priority (DCHPRI7) |

Table 18-1. eDMA 32-bit memory map (continued)

| eDMA Offset | Register | | | |
|---|---|---|---|---|
| 0x0108 | eDMA Channel 8 Priority (DCHPRI8) | eDMA Channel 9 Priority (DCHPRI9) | eDMA Channel 10 Priority (DCHPRI10) | eDMA Channel 11 Priority (DCHPRI11) |
| 0x010c | eDMA Channel 12 Priority (DCHPRI12) | eDMA Channel 13 Priority (DCHPRI13) | eDMA Channel 14 Priority (DCHPRI14) | eDMA Channel 15 Priority (DCHPRI15) |
| 0x0110 | eDMA Channel 16 Priority (DCHPRI16) | eDMA Channel 17 Priority (DCHPRI17) | eDMA Channel 18 Priority (DCHPRI18) | eDMA Channel 19 Priority (DCHPRI19) |
| 0x0114 | eDMA Channel 20 Priority (DCHPRI20) | eDMA Channel 21 Priority (DCHPRI21) | eDMA Channel 22 Priority (DCHPRI22) | eDMA Channel 23 Priority (DCHPRI23) |
| 0x0118 | eDMA Channel 24 Priority (DCHPRI24) | eDMA Channel 25 Priority (DCHPRI25) | eDMA Channel 26 Priority (DCHPRI26) | eDMA Channel 27 Priority (DCHPRI27) |
| 0x011c | eDMA Channel 28 Priority (DCHPRI28) | eDMA Channel 29 Priority (DCHPRI29) | eDMA Channel 30 Priority (DCHPRI30) | eDMA Channel 31 Priority (DCHPRI31) |
| 0x0120 | eDMA Channel 32 Priority (DCHPRI32) | eDMA Channel 33 Priority (DCHPRI33) | eDMA Channel 34 Priority (DCHPRI34) | eDMA Channel 35 Priority (DCHPRI35) |
| 0x0124 | eDMA Channel 36 Priority (DCHPRI36) | eDMA Channel 37 Priority (DCHPRI37) | eDMA Channel 38 Priority (DCHPRI38) | eDMA Channel 39 Priority (DCHPRI39) |
| 0x0128 | eDMA Channel 40 Priority (DCHPRI40) | eDMA Channel 41 Priority (DCHPRI41) | eDMA Channel 42 Priority (DCHPRI42) | eDMA Channel 43 Priority (DCHPRI43) |
| 0x012c | eDMA Channel 44 Priority (DCHPRI44) | eDMA Channel 45 Priority (DCHPRI45) | eDMA Channel 46 Priority (DCHPRI46) | eDMA Channel 47 Priority (DCHPRI47) |
| 0x0130 | eDMA Channel 48 Priority (DCHPRI48) | eDMA Channel 49 Priority (DCHPRI49) | eDMA Channel 50 Priority (DCHPRI50) | eDMA Channel 51 Priority (DCHPRI51) |
| 0x0134 | eDMA Channel 52 Priority (DCHPRI52) | eDMA Channel 53 Priority (DCHPRI53) | eDMA Channel 54 Priority (DCHPRI54) | eDMA Channel 55 Priority (DCHPRI55) |
| 0x0138 | eDMA Channel 56 Priority (DCHPRI56) | eDMA Channel 57 Priority (DCHPRI57) | eDMA Channel 58 Priority (DCHPRI58) | eDMA Channel 59 Priority (DCHPRI59) |
| 0x013c | eDMA Channel 60 Priority (DCHPRI60) | eDMA Channel 61 Priority (DCHPRI61) | eDMA Channel 62 Priority (DCHPRI62) | eDMA Channel 63 Priority (DCHPRI63) |
| 0x0140-0x0ffc | Reserved | | | |
| 0x1000-0x11fc | TCD00-TCD15 | | | |
| 0x1200-0x13fc | TCD16-TCD31 | | | |
| 0x1400-0x15fc | TCD32-TCD47 | | | |
| 0x1600-0x17fc | TCD48-TCD63 | | | |

## 18.2.1 Register descriptions

### 18.2.1.1 eDMA Control Register (DMACR)

The 32-bit DMACR defines the basic operating configuration of the DMA.

The eDMA arbitrates channel service requests in groups of 16 channels. The 64- and 32-channel configurations have four groups (3,2,1,0) and two groups (1,0), respectively; the 16 channel configuration has only one group (0). Group 3 contains channels 63-48, group 2 contains channels 47-32, group 1 contains channels 31-16, and group 0 contains channels 15-0

Arbitration within a group can be configured to use either a fixed-priority or a round-robin selection. In fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The priorities are assigned by the channel priority registers (see Section 18.2.1.16, “eDMA Channel n Priority (DCHPRIn), n = 0,..., {15,31,63}). In round-robin arbitration mode, the channel priorities are ignored and the channels within each group are cycled through without regard to priority.

The group priorities operate in a similar fashion. In group fixed-priority arbitration mode, channel service requests in the highest priority group are executed first where priority level 3 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPri registers. All group priorities must have unique values prior to any channel service requests occur, otherwise a configuration error will be reported. Unused group priority registers, per configuration, are unimplemented in the DMACR. In group round-robin mode, the group priorities are ignored and the groups are cycled through without regard to priority.

Minor loop offsets are address offset values added to the final source address (saddr) or destination address (daddr) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (mloff) is added to the final source address (saddr), or the final destination address (daddr), or both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (slast and dlast_sga) are used to compute the next saddr and daddr values.

When minor loop mapping is enabled (DMACR[EMLM] = 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: an source enable bit (smloe) to specify the minor loop offset should be applied to the source address (saddr) upon minor loop completion, an destination enable bit (dmloe) to specify the minor loop offset should be applied to the destination address (daddr) upon minor loop completion, and the sign extended minor loop offset value (mloff). The same offset value (mloff) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (smloe set or dmloe set), the nbytes field is reduced to 10 bits. When both minor loop offsets are disabled (smloe cleared and dmloe cleared), the nbytes field is a 30-bit vector.

When minor loop mapping is disabled (DMACR[EMLM] = 0), all 32 bits of TCDn word2 are assigned to the nbytes field. See Section 18.2.1.18, “Transfer Control Descriptor (TCD), for more details.

See Figure 18-2 and Table 18-2 for the DMACR definition.

**Register address: DMA_Offset + 0x0000**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CX | ECX |
| W | | | | | | | | | | | | | | | | |
| RESET: | | | | | | | | | | | | | | | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GRP3PRI | | GRP2PRI | | GRP1PRI | | GRP0PRI | | EML M | CLM | HALT | HOE | ERG A | ERC A | EDB G | EBW |
| W | | | | | | | | | | | | | | | | |
| RESET: | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 18-2. DMA Control Register (DMACR)**

**Table 18-2. DMA Control Register (DMACR) field descriptions**

| Name | Description | Value |
|---|---|---|
| CX | Cancel Transfer | 0 Normal operation.<br>1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to be finished. The cancel takes effect after the last write of the current read/write sequence. The CXFR bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed. |
| ECX | Error Cancel Transfer | 0 Normal operation.<br>1 Cancel the remaining data transfer in the same fashion as the CX cancel transfer. Stop the executing channel and force the minor loop to be finished. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel has been honored. In addition to cancelling the transfer, the ECX treats the cancel as an error condition; thus updating the DMAES register and generating an optional error interrupt (see Section 18.2.1.2, “eDMA Error Status (DMAES)). |
| GRP3PRI | Channel Group 3 Priority | Group 3 priority level when fixed priority group arbitration is enabled. |
| GRP2PRI | Channel Group 2 Priority | Group 2 priority level when fixed priority group arbitration is enabled. |
| GRP1PRI | Channel Group 1 Priority | Group 1 priority level when fixed priority group arbitration is enabled. |
| GRP0PRI | Channel Group 0 Priority | Group 0 priority level when fixed priority group arbitration is enabled. |
| EMLM | Enable Minor Loop Mapping | 0 Minor loop mapping disabled. TCDn.word2 is defined as a 32-bit nbytes field.<br>1 Minor loop mapping enabled. When set, TCDn.word2 is redefined to include individual enable fields, an offset field and the nbytes field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The nbytes field is reduced when either offset is enabled. |
| CLM | Continuous Link Mode | 0 A minor loop channel link made to itself will go through channel arbitration before being activated again.<br>1 A minor loop channel link made to itself will not go through channel arbitration before being activated again. Upon minor loop completion the channel will active again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop. |
| HALT | Halt DMA Operations | 0 Normal operation.<br>1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution will resume when the HALT bit is cleared. |

Table 18-2. DMA Control Register (DMACR) field descriptions

| Name | Description | Value |
|---|---|---|
| HOE | Halt On Error | 0 Normal operation.<br>1 Any error will cause the HALT bit to be set. Subsequently, all service requests will be ignored until the HALT bit is cleared. |
| ERGA | Enable Round Robin Group Arbitration | 0 Fixed priority arbitration is used for selection among the groups.<br>1 Round robin arbitration is used for selection among the groups. |
| ERCA | Enable Round Robin Channel Arbitration | 0 Fixed priority arbitration is used for channel selection within each group.<br>1 Round robin arbitration is used for channel selection within each group. |
| EDBG | Enable Debug | 0 The assertion of the ipg_debug input is ignored.<br>1 The assertion of the ipg_debug input causes the DMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution will resume when either the ipg_debug input is negated or the EDBG bit is cleared. |
| EBW | Enable Buffered Writes | 0 The bufferable write signal (hprot[2]) is not asserted during AMBA AHB writes.<br>1 The bufferable write signal (hprot[2]) is asserted on all AMBA AHB writes except for the last write sequence. |

### 18.2.1.2 eDMA Error Status (DMAES)

The DMAES register provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer control descriptor or an illegal priority register setting in fixed arbitration mode) or an error termination to a bus master read or write cycle.

A configuration error is caused when the starting source or destination address, source or destination offsets, minor loop byte count and the transfer size represent an inconsistent state. The addresses and offsets must be aligned on 0-modulo-transfer_size boundaries, and the minor loop byte count must be a multiple of the source and destination transfer sizes. All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively. In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups. All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when fixed arbitration mode is enabled. If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (dlast_sga) is not aligned on a 32-byte boundary. If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCD.citer.e_link bit does not equal the TCD.biter.e_link bit. All configuration error conditions except scatter/gather and minor loop link error are reported as the channel is activated and assert an error interrupt request, if enabled. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address and current iteration count at the point of the fault. When a system bus error occurs, the channel is terminated after the read or write transaction which is already pipelined after errant access, has completed. If a bus error occurs on the last read prior to beginning the write sequence, the write will execute using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence will execute before the channel is terminated due to the destination bus error.

A transfer may be cancelled by software via the DMACR[CX] bit or hardware via the **dma_cancel_xfer** input signal. When a cancel transfer request is recognized, the eDMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the DMAES register is updated with the cancelled channel number and error cancel bit is set. The TCD of a cancelled channel has the source address and destination address of the last transfer saved in the TCD. It is the responsibility of the user to initialize the TCD again should the channel need to be restarted because the aforementioned fields have been modified by the eDMA engine and no longer represent the original parameters. When a transfer is cancelled via the error cancel transfer mechanism (setting the DMACR[ECX] or asserting the **dma_err_cancel_xfer** input), the channel number is loaded into the ERRCHN field and the ECX and VLD bits are set are set in the DMAES register. In addition, an error interrupt may be generated if enabled. See Section 18.2.1.14, eDMA Error (DMAERRH, DMAERRL) for error interrupt details.The occurrence of any type of error causes the eDMA engine to immediately stop, and the appropriate channel bit in the eDMA Error register to be asserted. At the same time, the details of the error condition are loaded into the DMAES register. The major loop complete indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are *not* affected when an error is detected. See Figure 1 and Table 3 for the DMAES definition.

**Figure 1. eDMA Error Status (DMAES) Register**

**Table 3. eDMA Error Status (DMAES) field descriptions**

| Name | Description | Value |
|---|---|---|
| VLD | Logical OR of all DMAERRH and DMAERRL status bits. | 0 No DMAERR bits are set.<br>1 At least one DMAERR bit is set indicating a valid error exists that has not been cleared. |
| ECX | Transfer cancelled | 0 No cancelled transfers.<br>1 The last recorded entry was a cancelled transfer via the error cancel transfer input. |
| GPE | Group Priority Error | 0 No group priority error.<br>1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique. |
| CPE | Channel Priority Error | 0 No channel priority error.<br>1 The last recorded error was a configuration error in the channel priorities within a group. All channel priorities within a group are not unique. |

Table 3. eDMA Error Status (DMAES) field descriptions (continued)

| Name | Description | Value |
|---|---|---|
| ERRCHN[] | Error/Cancelled Channel Number | The channel number of the last recorded error (excluding GPE and CPE errors)or last recorded transfer that was cancelled with error exit. |
| SAE | Source Address Error | 0 No source address configuration error.<br>1 The last recorded error was a configuration error detected in the TCD.saddr field. TCD.saddr is inconsistent with TCD.ssize. |
| SOE | Source Offset Error | 0 No source offset configuration error.<br>1 The last recorded error was a configuration error detected in the TCD.soff field. TCD.soff is inconsistent with TCD.ssize. |
| DAE | Destination Address Error | 0 No destination address configuration error.<br>1 The last recorded error was a configuration error detected in the TCD.daddr field. TCD.daddr is inconsistent with TCD.dsize. |
| DOE | Destination Offset Error | 0 No destination offset configuration error.<br>1 The last recorded error was a configuration error detected in the TCD.doff field. TCD.doff is inconsistent with TCD.dsize. |
| NCE | Nbytes/Citer Configuration Error | 0 No nbytes/citer configuration error.<br>1 The last recorded error was a configuration error detected in the TCD.nbytes or TCD.citer fields. TCD.nbytes is not a multiple of TCD.ssize and TCD.dsize, or TCD.citer is equal to zero, or TCD.citer.e_link is not equal to TCD.biter.e_link. |
| SGE | Scatter/Gather Configuration Error | 0 No scatter/gather configuration error.<br>1 The last recorded error was a configuration error detected in the TCD.dlast_sga field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCD.e_sg is enabled. TCD.dlast_sga is not on a 32 byte boundary. |
| SBE | Source Bus Error | 0 No source bus error.<br>1 The last recorded error was a bus error on a source read. |
| DBE | Destination Bus Error | 0 No destination bus error.<br>1 The last recorded error was a bus error on a destination write. |

### 18.2.1.3 eDMA Enable Request (DMAERQH, DMAERQL)

The DMAERQ{H,L} registers provide a bit map for the implemented channels {16,32,64} to enable the request signal for each channel. DMAERQH supports channels 63-32, while DMAEQRL covers channels 31-00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the DMASERQ and DMACERQ registers. The eDMA{S,C}ERQ registers are provided so that the request enable for a *single* channel can easily be modified without the need to perform a read-modify-write sequence to the DMAERQ{H,L} registers.

Both the eDMA request input signal and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request flag does *not* affect a channel

service request made explicitly through software or a linked channel request. See Figure 2 and Table 4 for the DMAERQ definition.

**Register address: DMA_Offset + 0x0008 (DMAERQH), +0x000c (DMAERQL)**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ERQ 63 | ERQ 62 | ERQ 61 | ERQ 60 | ERQ 59 | ERQ 58 | ERQ 57 | ERQ 56 | ERQ 55 | ERQ 54 | ERQ 53 | ERQ 52 | ERQ 51 | ERQ 50 | ERQ 49 | ERQ 48 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ERQ 47 | ERQ 46 | ERQ 45 | ERQ 44 | ERQ 43 | ERQ 42 | ERQ 41 | ERQ 40 | ERQ 39 | ERQ 38 | ERQ 37 | ERQ 36 | ERQ 35 | ERQ 34 | ERQ 33 | ERQ 32 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ERQ 31 | ERQ 30 | ERQ 29 | ERQ 28 | ERQ 27 | ERQ 26 | ERQ 25 | ERQ 24 | ERQ 23 | ERQ 22 | ERQ 21 | ERQ 20 | ERQ 19 | ERQ 18 | ERQ 17 | ERQ 16 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ERQ 15 | ERQ 14 | ERQ 13 | ERQ 12 | ERQ 11 | ERQ 10 | ERQ 09 | ERQ 08 | ERQ 07 | ERQ 06 | ERQ 05 | ERQ 04 | ERQ 03 | ERQ 02 | ERQ 01 | ERQ 00 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 2. eDMA Enable Request (DMAERQH, DMAERQL) Registers**

**Table 4. eDMA Enable Request (DMAERQH, DMAERQL) field descriptions**

| Name | Description | Value |
|---|---|---|
| ERQn,<br>n = 0,... 15<br>n = 0,... 31<br>n = 0,... 63 | Enable eDMA Request n | 0 The eDMA request signal for channel n is disabled.<br>1 The eDMA request signal for channel n is enabled. |

As a given channel completes the processing of its major iteration count, there is a flag in the transfer control descriptor that may affect the ending state of the DMAERQ bit for that channel. If the TCD.d_req bit is set, then the corresponding DMAERQ bit is cleared, disabling the eDMA request; else if the d_req bit is cleared, the state of the DMAERQ bit is unaffected.

### 18.2.1.4 eDMA Enable Error Interrupt (DMAEEIH, DMAEEIL)

The DMAEEI{H,L} registers provide a bit map for the implemented channels {16,32,64} to enable the error interrupt signal for each channel. DMAEEIH supports channels 63-32, while DMAEEIL covers channels 31-00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the DMASEEI and DMACEEI registers. The eDMA{S,C}EEI registers are provided so that the error interrupt enable for a *single* channel can easily be modified without the need to perform a read-modify-write sequence to the DMAEEI{H,L} registers.

Both the eDMA error indicator and this error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted. See Figure 3 and Table 5 for the DMAEEI definition.

**Register address: DMA_Offset + 0x0010 (DMAEEIH), +0x0014 (DMAEEIL)**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EEI63 | EEI62 | EEI61 | EEI60 | EEI59 | EEI58 | EEI57 | EEI56 | EEI55 | EEI54 | EEI53 | EEI52 | EEI51 | EEI50 | EEI49 | EEI48 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EEI47 | EEI46 | EEI45 | EEI44 | EEI43 | EEI42 | EEI41 | EEI40 | EEI39 | EEI38 | EEI37 | EEI36 | EEI35 | EEI34 | EEI33 | EEI32 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EEI31 | EEI30 | EEI29 | EEI28 | EEI27 | EEI26 | EEI25 | EEI24 | EEI23 | EEI22 | EEI21 | EEI20 | EEI19 | EEI18 | EEI17 | EEI16 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EEI15 | EEI14 | EEI13 | EEI12 | EEI11 | EEI10 | EEI09 | EEI08 | EEI07 | EEI06 | EEI05 | EEI04 | EEI03 | EEI02 | EEI01 | EEI00 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 3. eDMA Enable Error Interrupt (DMAEEIH, DMAEEIL) Registers**

**Table 5. eDMA Enable Error Interrupt (DMAEEIH, DMAEEIL) field descriptions**

| Name | Description | Value |
|---|---|---|
| EEIn,<br>n = 0,... 15<br>n = 0,... 31<br>n = 0,... 63 | Enable Error Interrupt n | 0 The error signal for channel n does not generate an error interrupt.<br>1 The assertion of the error signal for channel n generate an error interrupt request. |

### 18.2.1.5 eDMA Set Enable Request (DMASERQ)

The DMASERQ register provides a simple memory-mapped mechanism to set a given bit in the DMAERQ{H,L} registers to enable the eDMA request for a given channel. The data value on a register write causes the corresponding bit in the DMAERQ{H,L} register to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing the entire contents of DMAERQ{H,L} to be asserted. If bit 7 is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Figure 4 and Table 6 for the DMASERQ definition.

**Register address: DMA_Offset + 0x0018**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | NOP | SERQ[] | | | | | | |
| RESET: | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 4. eDMA Set Enable Request (DMASERQ) Register**

Table 6. eDMA Set Enable Request (DMASERQ) field descriptions

| Name | Description | Value |
|---|---|---|
| NOP | No Operation | 0 Normal operation.<br>1 No operation, ignore bits 6-0 |
| SERQ[] | Set Enable Request | 0-63 Set the corresponding bit in DMAERQ{H,L}<br>64-127 Set all bits in DMAERQ{H,L} |

### 18.2.1.6 eDMA Clear Enable Request (DMACERQ)

The DMACERQ register provides a simple memory-mapped mechanism to clear a given bit in the DMAERQ{H,L} registers to disable the eDMA request for a given channel. The data value on a register write causes the corresponding bit in the DMAERQ{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAERQ{H,L} to be zeroed, disabling all eDMA request inputs. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word.Reads of this register return all zeroes. See Figure 5 and Table 7 for the DMACERQ definition.

**Register address: DMA_Offset + 0x0019**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | NOP | CERQ[] | | | | | | |
| RESET: | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

Figure 5. eDMA Clear Enable Request (DMACERQ) Register

Table 7. eDMA Clear Enable Request (DMACERQ) field descriptions

| Name | Description | Value |
|---|---|---|
| NOP | No Operation | 0 Normal operation.<br>1 No operation, ignore bits 6-0 |
| CERQ[] | Clear Enable Request | 0-63 Clear corresponding bit in DMAERQ{H,L}<br>64-127 Clear all bits in DMAERQ{H,L} |

### 18.2.1.7 eDMA Set Enable Error Interrupt (DMASEEI)

The DMASEEI register provides a simple memory-mapped mechanism to set a given bit in the DMAEEI{H,L} registers to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the DMAEEI{H,L} register to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing the entire contents of DMAEEI{H,L} to be asserted. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit wordReads of this register return all zeroes. See Figure 6 and Table 8 for the DMASEEI definition.

**Register address: DMA_Offset + 0x001a**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | NOP | SEEI[] | | | | | | |
| RESET: | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 6. eDMA Set Enable Error Interrupt (DMASEEI) Register**

**Table 8. eDMA Set Enable Error Interrupt (DMASEEI) field descriptions**

| Name | Description | Value |
|---|---|---|
| NOP | No Operation | 0 Normal operation.<br>1 No operation, ignore bits 6-0 |
| SEEI[] | Set Enable Error Interrupt | 0-63 Set the corresponding bit in DMAEEI{H,L}<br>64-127 Set all bits in DMAEEI{H,L} |

### 18.2.1.8 eDMA Clear Enable Error Interrupt (DMACEEI)

The DMACEEI register provides a simple memory-mapped mechanism to clear a given bit in the DMAEEI{H,L} registers to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the DMAEEI{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAEEI{H,L} to be zeroed, disabling all eDMA request inputs. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Figure 7 and Table 9 for the DMACEEI definition.

**Register address: DMA_Offset + 0x001b**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | NOP | CEEI[] | | | | | | |
| RESET: | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 7. eDMA Clear Enable Error Interrupt (DMACEEI) Register**

**Table 9. eDMA Clear Enable Error Interrupt (DMACEEI) field descriptions**

| Name | Description | Value |
|---|---|---|
| NOP | No Operation | 0 Normal operation.<br>1 No operation, ignore bits 6-0 |
| CEEI[] | Clear Enable Error Interrupt | 0-63 Clear corresponding bit in DMAEEI{H,L}<br>64-127 Clear all bits in DMAEEI{H,L} |

### 18.2.1.9 eDMA Clear Interrupt Request (DMACINT)

The DMACINT register provides a simple memory-mapped mechanism to clear a given bit in the DMAINT{H,L} registers to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the DMAINT{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAINT{H,L} to be zeroed, disabling all eDMA interrupt requests. If the NOP bit is set,

the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Figure 8 and Table 10 for the DMACINT definition.

**Figure 8. eDMA Clear Interrupt Request (DMACINT) Fields**

**Table 10. eDMA Clear Interrupt Request (DMACINT) field descriptions**

| Name | Description | Value |
|---|---|---|
| NOP | No Operation | 0 Normal operation.<br>1 No operation, ignore bits 6-0 |
| CINT[] | Clear Interrupt Request | 0-63 Clear the corresponding bit in DMAINT{H,L}<br>64-127 Clear all bits in DMAINT{H,L} |

### 18.2.1.10 eDMA Clear Error (DMACERR)

The DMACEER register provides a simple memory-mapped mechanism to clear a given bit in the DMAERR{H,L} registers to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the DMAERR{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the DMAERR{H,L} to be zeroed, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Figure 9 and Table 11 for the DMACERR definition.

**Register address: DMA_Offset + 0x001d**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | NOP | CERR[] | | | | | | |
| RESET: | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 9. eDMA Clear Error (DMACERR) Register**

**Table 11. eDMA Clear Error (DMACERR) field descriptions**

| Name | Description | Value |
|---|---|---|
| NOP | No Operation | 0 Normal operation.<br>1 No operation, ignore bits 6-0 |
| CERR[] | Clear Error Indicator | 0-63 Clear corresponding bit in DMAERR{H,L}<br>64-127 Clear all bits in DMAERR{H,L} |

### 18.2.1.11 eDMA Set START Bit (DMASSRT)

The DMASSRT register provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding Transfer Control Descriptor to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Table 27 for the TCD START bit definition.

**Register address: DMA_Offset + 0x001e**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | NOP | SSRT[] | | | | | | |
| RESET: | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 10. eDMA Set START Bit (DMASSRT) Register**

**Table 12. eDMA Set START Bit (DMASSRT) field descriptions**

| Name | Description | Value |
|---|---|---|
| NOP | No Operation | 0 Normal operation.<br>1 No operation, ignore bits 6-0 |
| SSRT[] | Set START Bit<br>(Channel Service Request) | 0-63 Set the corresponding channel's TCD.start<br>64-127 Set all TCD.start bits |

### 18.2.1.12 eDMA Clear DONE Status (DMACDNE)

The DMACDNE register provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding Transfer Control Descriptor to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing all DONE bits to be cleared. If bit 7 is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes. See Table 27 for the TCD DONE bit definition.

**Register address: DMA_Offset + 0x001f**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | NOP | CDNE[] | | | | | | |
| RESET: | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11. eDMA Clear DONE Status (DMACDNE) Register**

**Table 13. eDMA Clear DONE Status (DMACDNE) field descriptions**

| Name | Description | Value |
|---|---|---|
| NOP | No Operation | 0 Normal operation.<br>1 No operation, ignore bits 6-0 |
| CDNE[] | Clear DONE Status Bit | 0-63 Clear the corresponding channel's DONE bit<br>64-127 Clear all TCD DONE bits |

### 18.2.1.13 eDMA Interrupt Request (DMAINTH, DMAINTL)

The DMAINT{H,L} registers provide a bit map for the implemented channels {16,32,64} signaling the presence of an interrupt request for each channel. DMAINTH supports channels 63-32, while DMAINTL covers channels 31-00. The eDMA engine signals the occurrence of a programmed interrupt upon the completion of a data transfer as defined in the transfer control descriptor by setting the appropriate bit in this register. The outputs of this register are directly routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any given channel, it is software's

responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the DMACINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the DMACINT register. On writes to the DMAINT, a one in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The DMACINT register is provided so the interrupt request for a *single* channel can easily be cleared without the need to perform a read-modify-write sequence to the DMAINT{H,L} registers. See Figure 12 and Table 14 for the DMAINT definition.

**Register address: DMA_Offset + 0x0020 (DMAINTH), +0x0024 (DMAINTL)**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INT63 | INT62 | INT61 | INT60 | INT59 | INT58 | INT57 | INT56 | INT55 | INT54 | INT53 | INT52 | INT51 | INT50 | INT49 | INT48 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INT47 | INT46 | INT45 | INT44 | INT43 | INT42 | INT41 | INT40 | INT39 | INT38 | INT37 | INT36 | INT35 | INT34 | INT33 | INT32 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INT31 | INT30 | INT29 | INT28 | INT27 | INT26 | INT25 | INT24 | INT23 | INT22 | INT21 | INT20 | INT19 | INT18 | INT17 | INT16 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INT15 | INT14 | INT13 | INT12 | INT11 | INT10 | INT09 | INT08 | INT07 | INT06 | INT05 | INT04 | INT03 | INT02 | INT01 | INT00 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 12. eDMA Interrupt Request (DMAINTH, DMAINTL) Registers**

**Table 14. eDMA Interrupt Request (DMAINTH, DMAINTL) field descriptions**

| Name | Description | Value |
|---|---|---|
| INTn,<br>n = 0,... 15<br>n = 0,... 31<br>n = 0,... 63 | DMA Interrupt Request n | 0 The interrupt request for channel n is cleared.<br>1 The interrupt request for channel n is active. |

### 18.2.1.14 eDMA Error (DMAERRH, DMAERRL)

The DMAERR{H,L} registers provide a bit map for the implemented channels {16,32,64} signaling the presence of an error for each channel. DMAERRH supports channels 63-32, while DMAERRL covers channels 31-00. The eDMA engine signals the occurrence of a error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the DMAEEI register, then logically summed across groups of 16, 32 and 64 channels to form several group error interrupt requests which is then routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any eDMA errors, it is software's responsibility to clear the appropriate bit, negating the error interrupt request. Typically, a write to the DMACERR register in the interrupt service

routine is used for this purpose. Recall the normal eDMA channel completion indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are *not* affected when an error is detected.

The contents of this register can also be polled and a non-zero value indicates the presence of a channel error, regardless of the state of the DMAEEI register. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the DMACERR register. On writes to the DMAERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The DMACERR register is provided so the error indicator for a *single* channel can easily be cleared. See Figure 13 and Table 15 for the DMAERR definition.

**Register address: DMA_Offset + 0x0028 (DMAERRH), +0x002c (DMAERRL)**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ERR 63 | ERR 62 | ERR 61 | ERR 60 | ERR 59 | ERR 58 | ERR 57 | ERR 56 | ERR 55 | ERR 54 | ERR 53 | ERR 52 | ERR 51 | ERR 50 | ERR 49 | ERR 48 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ERR 47 | ERR 46 | ERR 45 | ERR 44 | ERR 43 | ERR 42 | ERR 41 | ERR 40 | ERR 39 | ERR 38 | ERR 37 | ERR 36 | ERR 35 | ERR 34 | ERR 33 | ERR 32 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ERR 31 | ERR 30 | ERR 29 | ERR 28 | ERR 27 | ERR 26 | ERR 25 | ERR 24 | ERR 23 | ERR 22 | ERR 21 | ERR 20 | ERR 19 | ERR 18 | ERR 17 | ERR 16 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ERR 15 | ERR 14 | ERR 13 | ERR 12 | ERR 11 | ERR 10 | ERR 09 | ERR 08 | ERR 07 | ERR 06 | ERR 05 | ERR 04 | ERR 03 | ERR 02 | ERR 01 | ERR 00 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 13. eDMA Error (DMAERRH, DMAERRL) Registers**

**Table 15. eDMA Error (DMAERRH, DMAERRL) field descriptions**

| Name | Description | Value |
|---|---|---|
| ERRn,<br>n = 0,... 15<br>n = 0,... 31<br>n = 0,... 63 | DMA Error n | 0 An error in channel n has not occurred.<br>1 An error in channel n has occurred. |

## 18.2.1.15 eDMA Hardware Request Status (DMAHRSH, DMAHRSL)

The DMAHRS{H,L} registers provide a bit map for the implemented channels {16,32,64} to show the current hardware request status for each channel. DMAHRSH supports channels 63-32, while DMAHRSL covers channels 31-00. Hardware request status reflects the current state of the registered and qualified (via the DMAERQ field) ipd_req lines as seen by the DMA2's arbitration logic. This view into the hardware request signals may be used for debug purposes.

See Figure 14 and Figure 16 for the DMAHRS definition.

**Register address: DMA_Offset + 0x0030 (DMAHRSH), +0x0034 (DMAHRSL)**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HRS 63 | HRS 62 | HRS 61 | HRS 60 | HRS 59 | HRS 58 | HRS 57 | HRS 56 | HRS 55 | HRS 54 | HRS 53 | HRS 52 | HRS 51 | HRS 50 | HRS 49 | HRS 48 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HRS 47 | HRS 46 | HRS 45 | HRS 44 | HRS 43 | HRS 42 | HRS 41 | HRS 40 | HRS 39 | HRS 38 | HRS 37 | HRS 36 | HRS 35 | HRS 34 | HRS 33 | HRS 32 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HRS 31 | HRS 30 | HRS 29 | HRS 28 | HRS 27 | HRS 26 | HRS 25 | HRS 24 | HRS 23 | HRS 22 | HRS 21 | HRS 20 | HRS 19 | HRS 18 | HRS 17 | HRS 16 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HRS 15 | HRS 14 | HRS 13 | HRS 12 | HRS 11 | HRS 10 | HRS 09 | HRS 08 | HRS 07 | HRS 06 | HRS 05 | HRS 04 | HRS 03 | HRS 02 | HRS 01 | HRS 00 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 14. eDMA Hardware Request Status (DMAHRSH, DMAHRSL) Registers**

**Table 16. eDMA Hardware Request Status (DMAHRSH, DMAHRSL) field descriptions**

| Name | Description | Value |
|---|---|---|
| HRSn,<br>n = 0,... 15<br>n = 0,... 31<br>n = 0,... 63 | DMA Hardware Request Status n | 0 A hardware service request for channel n is not present.<br>1 A hardware service request for channel n is present.<br><br>**Note:** The hardware request status reflects the state of the request as seen by the arbitration logic. Therefore, this status is affected by the DMAERQn bit. |

### 18.2.1.16 eDMA Channel n Priority (DCHPRIn), n = 0,..., {15,31,63}

When the fixed-priority channel arbitration mode is enabled (DMACR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value, i.e., 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values, otherwise a configuration error will be reported. The range of the priority value is limited to the values of 0–15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the DMACR. See Figure 18-2 and Table 18-2 for the DMACR definition.

Channel preemption is enabled on a per channel basis by setting the ECP bit in the DCHPRIn register. Channel preemption allows the executing channel's data transfers to be temporarily suspended in favor of starting a higher priority channel. After the preempting channel has completed all of its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel will be suspended and the higher priority channel will be serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is only available when fixed arbitration is selected for both group and channel arbitration modes. A channel's ability to preempt another channel can be disabled by setting the DPA bit in the DCHPRIn register. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer; regardless of the lower priority channel's ECP setting. This allows for a pool of low priority, large data moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available a true, high priority channel. See Figure 15 and Table 17 for the DCHPRIn definition.

**Register address: DMA_Offset + 0x100 + n**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | ECP | DPA | GRPPRI[] | | CHPRI[] | | | |
| W | | | | | | | | |
| RESET: | 0 | 0 | * | * | * | * | * | * |

= Unimplemented,
* = defaults to channel number (n) after reset

**Figure 15. eDMA Channel n Priority (DCHPRIn) Register**

**Table 17. eDMA Channel n Priority (DCHPRIn) field descriptions**

| Name | Description | Value |
|---|---|---|
| ECP | Enable Channel Preemption | 0 Channel *n* cannot be suspended by a higher priority channel's service request.<br>1 Channel *n* can be temporarily suspended by the service request of a higher priority channel. |
| DPA | Disable Preempt Ability | 0 Channel *n* can suspend a lower priority channel.<br>1 Channel *n* cannot suspend any channel, regardless of channel priority. |
| GRPPRI[] | Channel n Current Group Priority | Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored. |
| CHPRI[] | Channel n Arbitration Priority | Channel priority when fixed-priority arbitration is enabled. |

### 18.2.1.17 eDMA Channel n Master ID (DCHMIDn), n = 0,..., {15,31,63}

The eDMA master ID replication registers allow the eDMA to use the same protection level and AHB system bus ID of the master programming the eDMA's TCD. When enabled, the eDMA will use the master ID and protection level stored in the DCHMID register instead of the eDMA's default values. When a master (a core for example) programs a TCD, its master ID and protection level are captured when TCD Word 7 control attributes are written. Although the scatter/gather operation can change the contents of

TCD Word 7, that operation does not affect the DCHMIDn registers. See Figure 16 and Table 18 for the DCHMIDn definition.

**Register address: DMA_Offset + 0x140 + n**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| R | EMI | PAL | 0 | 0 | MID[] | | | |
| W | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented,
* = defaults to channel number (n) after reset

**Figure 16. eDMA Channel n Master ID (DCHMIDn) Register**

**Table 18. eDMA Channel n Master ID (DCHMIDn) field descriptions**

| Name | Description | Value |
|---|---|---|
| EMI | Enable Master ID replication | 0 Master ID replication is disabled.<br>1 Master ID replication is enabled. |
| PAL | Privileged Access Level | 0. User protection level for DMA transfers<br>1 Privileged protection level for DMA transfers<br><br>**Note:** The protection level captured in this register reflects the level used when writing the channel's control attributes; TCD Word 7, bits[]. |
| MID[] | Master ID | DMA's master ID when channel *n* is active and master ID replication is enabled.<br><br>**Note:** The master ID captured in this register reflects the ID used when writing the channel's control attributes; TCD Word 7, bits[]. |

### 18.2.1.18 Transfer Control Descriptor (TCD)

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The TCD structure was previously discussed in detail in Section 18.1.2, Features." The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel [n-1]. The definitions of the TCD are presented as eight 32-bit values. Table 19 is a 32-bit view of the basic TCD structure.

**Table 19. TCDn 32-bit memory structure**

| eDMA Offset | TCDn Field | |
|---|---|---|
| 0x1000 + (32 x n) + 0x00 | Source Address (saddr) | |
| 0x1000 + (32 x n) + 0x04 | Transfer Attributes (smod, ssize, dmod, dsize) | Signed Source Address Offset (soff) |
| 0x1000 + (32 x n) + 0x08 | Signed Minor Loop Offset (smloe, dmloe, mloff) | Inner "Minor" Byte Count (nbytes) |
| 0x1000 + (32 x n) + 0x0c | Last Source Address Adjustment (slast) | |
| 0x1000 + (32 x n) + 0x10 | Destination Address (daddr) | |
| 0x1000 + (32 x n) + 0x14 | Current "Major" Iteration Count (citer) | Signed Destination Address Offset (doff) |
| 0x1000 + (32 x n) + 0x18 | Last Destination Address Adjustment/Scatter Gather Address (dlast_sga) | |

**Table 19. TCDn 32-bit memory structure (continued)**

| | | |
|---|---|---|
| 0x1000 + (32 x n) + 0x1c | Beginning “Major” Iteration Count (biter) | Channel Control/Status (bwc, major.linkch, done, active, major.e_link, e_sg, d_req, int_half, int_maj, start) |

Figure 17 and Table 20 define word 0 of the TCDn structure, the saddr field.

**Register address: DMA_Offset + 0x1000 + (32 x n) + 0x00**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | saddr[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R<br>W | saddr[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

= Unimplemented

**Figure 17. TCDn Word 0 (TCDn.saddr) field**

**Table 20. TCDn Word 0 (TCDn.saddr) field description**

| Name | Description | Value |
|---|---|---|
| saddr[] | Source address | Memory address pointing to the source data. |

Figure 18 and Table 21 define word 1 of the TCDn structure, the soff and transfer attribute fields.

**Register address: DMA_Offset + 0x1000 + (32 x n) + 0x04**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | smod[] | | | | | ssize[] | | | dmod[] | | | | | dsize[] | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R<br>W | soff[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

= Unimplemented

**Figure 18. TCDn Word 1 (TCDn.{soff,smod,ssize,dmod,dsize}) fields**

**Table 21. TCDn Word 1 (TCDn.{smod,ssize,dmod,dsize,soff}) field descriptions**

| Name | Description | Value |
|---|---|---|
| smod[] | Source address modulo | 0 Source address modulo feature is disabled.<br><br>non-0<br>The value defines a specific address bit which is selected to be either the value after saddr + soff calculation is performed or the original register value. This feature provides the ability to easily implement a circular data queue. For data queues requiring power-of-2 “size” bytes, the queue should be based at a 0-modulo-size address and the smod field set to the appropriate value to freeze the upper address bits. The bit select is defined as ((1 << smod[4:0]) - 1) where a resulting 1 in a bit location selects the next state address for the corresponding address bit location and a 0 selects the original register value for the corresponding address bit location. For this application, the soff is typically set to the transfer size to implement post-increment addressing with the smod function constraining the addresses to a 0-modulo-size range. |
| ssize[] | Source data transfer size | 000 8-bit<br>001 16-bit<br>010 32-bit<br>011 64-bit<br>100 16-byte<br>101 32-byte<br>110 Reserved<br>111 Reserved |
| dmod[] | Destination address modulo | See the smod[5:0] definition. |
| dsize[] | Destination data transfer size | See the ssize[2:0] definition. |
| soff[] | Source address signed offset | Sign-extended offset applied to the current source address to form the next-state value as each source read is completed. |

Figure 19 and Table 22 define word 2 of the TCDn structure, the nbytes field.

**Register address: DMA_Offset + 0x1000 + (32 x n) + 0x08**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | nbytes[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | nbytes[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

= Unimplemented

**Figure 19. TCDn Word 2 (TCDn.nbytes) field**

**Table 22. TCDn Word 2 (TCDn.nbytes) field description**

| Name | Description | Value |
|---|---|---|
| nbytes[] | Inner “minor” byte transfer count | Number of bytes to be transferred in each service request of the channel.<br><br>As a channel is activated, the contents of the appropriate TCD is loaded into the eDMA engine, and the appropriate reads and writes performed until the complete byte transfer count has been transferred. This is an indivisible operation and cannot be stalled or halted. After the minor count is exhausted, the current values of the saddr and daddr are written back into the local memory, the major iteration count is decremented and restored to the local memory. If the major iteration count is completed, additional processing is performed.<br><br>The nbytes value 0x0000_0000 is interpreted as 0x1_0000_0000, thus specifying a 4 GB transfer. |

When minor loop mapping (DMACR[EMLM] = 1) is enabled, TCD word2 is redefined as four fields: a source minor loop offset enable, a destination minor loop offset enable, a minor loop offset field and a nbytes field.

**Figure 20. TCDn Word 2 (TCDn.nbytes) field (DMACR[EMLM] = 1)**

Figure 21 and Table 23 define word 3 of the TCDn structure, the slast field.

**Register address: DMA_Offset + 0x1000 + (32 x n) + 0x0c**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | slast[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| R<br>W | slast[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

= Unimplemented

**Figure 21. TCDn Word 3 (TCDn.slast) field**

**Table 23. TCDn Word 3 (TCDn.slast) field descriptions**

| Name | Description | Value |
|---|---|---|
| slast[] | Last source address adjustment | Adjustment value added to the source address at the completion of the outer major iteration count.<br><br>This value can be applied to “restore” the source address to the initial value, or adjust the address to reference the next data structure. |

Figure 22 and Table 24 define word 4 of the TCDn structure, the daddr field.

**Register address: DMA_Offset + 0x1000 + (32 x n) + 0x10**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | daddr[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | daddr[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

= Unimplemented

**Figure 22. TCDn Word 4 (TCDn.daddr) field**

**Table 24. TCDn Word 4 (TCDn.daddr) field description**

| Name | Description | Value |
|---|---|---|
| daddr[] | Destination address | Memory address pointing to the destination data. |

Figure 23 and Table 25 define word 5 of the TCDn structure, the citer and doff fields.

**Register address: DMA_Offset + 0x1000 + (32 x n) + 0x14**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | citer.<br>e_link | citer[] or<br>citer.linkch[] | | | | | | citer[] | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | doff[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

= Unimplemented

**Figure 23. TCDn Word 5 (TCDn.{citer,doff}) fields**

**Table 25. TCDn Word 5 (TCDn.{doff,citer}) field descriptions**

| Name | Description | Value |
|---|---|---|
| citer.e_link | Enable channel-to-channel linking on minor loop complete | As the channel completes the inner minor loop, this flag enables the linking to another channel, defined by citer.linkch[5:0]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.start bit of the specified channel. If channel linking is disabled, the citer value is extended to 15 bits in place of a link channel number. If the "major" loop is exhausted, this link mechanism is suppressed in favor of the major.e_link channel linking. *This bit must be equal to the biter.e_link bit otherwise a configuration error will be reported.*<br><br>0 The channel-to-channel linking is disabled.<br>1 The channel-to-channel linking is enabled. |

**Table 25. TCDn Word 5 (TCDn.{doff,citer}) field descriptions (continued)**

| Name | Description | Value |
|---|---|---|
| citer[]<br>or<br>citer.linkch[] | Current “major” iteration count<br>or<br>Link channel number | if (TCD.citer.e_link = 0) then<br>No channel-to-channel linking (or chaining) is performed after the inner "minor" loop is exhausted. TCD word 5, bits [30:25] are used to form a 15 bit citer field.<br><br>else<br>After the "minor" loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by citer.linkch[5:0] by setting that channel’s TCD.start bit.<br><br>The value contained in citer.linkch[5:0] must not exceed the number of implemented channels. |
| citer[] | Current “major” iteration count | This 9 or 15-bit count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. Once the major iteration count is exhausted, the channel performs a number of operations (e.g., final source and destination address calculations), optionally generating an interrupt to signal channel completion before reloading the citer field from the beginning iteration count (biter) field.<br><br>When the citer field is initially loaded by software, it must be set to the same value as that contained in the biter field.<br><br>If the channel is configured to execute a single service request, the initial values of biter and citer should be 0x0001. |
| doff[] | Destination address signed offset | Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed. |

Figure 24 and Table 26 define word 6 of the TCDn structure, the dlast_sga field.

**Register address: DMA_Offset + 0x1000 + (32 x n) + 0x18**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | dlast_sga[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | dlast_sga[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

= Unimplemented

**Figure 24. TCDn Word 6 (TCDn.dlast_sga) field**

**Table 26. TCDn Word 6 (TCDn.dlast_sga) field description**

| Name | Description | Value |
|---|---|---|
| dlast_sga[] | Last destination address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather) | if (TCD.e_sg = 0) then<br>Adjustment value added to the destination address at the completion of the outer major iteration count.<br><br>This value can be applied to “restore” the destination address to the initial value, or adjust the address to reference the next data structure.<br><br>else<br>This address points to the beginning of a 0-modulo-32 region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32, else a configuration error is reported. |

Figure 25 and Table 27 define word 7 of the TCDn structure, the biter and control/status fields.

**Register address: DMA_Offset + 0x1000 + (32 x n) + 0x1c**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | biter[] | | | | | | | | | | | | | | | |
| RESET: | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | bwc | | major.linkch[] | | | | | | done | active | major.e_link | e_sg | d_req | int_half | int_maj | start |
| RESET: | - | - | - | - | - | - | - | - | 0 | 0 | - | - | - | - | - | 0 |

= Unimplemented

**Figure 25. TCDn Word 7 (TCDn.{biter,control/status}) fields**

**Table 27. TCDn Word 7 (TCDn.{biter, control/status}) field descriptions**

| Name | Description | Value |
|---|---|---|
| biter.e_link | Enable channel-to-channel linking on minor loop complete | This is the initial value copied into the citer.e_link field when the major loop is completed. The citer.e_link field controls channel linking during channel execution. *This bit must be equal to the citer.e_link bit otherwise a configuration error will be reported.*<br><br>0 The channel-to-channel linking is disabled.<br>1 The channel-to-channel linking is enabled. |

**Table 27. TCDn Word 7 (TCDn.{biter, control/status}) field descriptions (continued)**

| Name | Description | Value |
|---|---|---|
| biter[]<br>or<br>biter.linkch[] | Beginning “major” iteration count<br>or<br>Beginning Link channel number | This is the initial value copied into the citer field or citer.linkch field when the major loop is completed. The citer fields controls the iteration count and linking during channel execution.<br><br>if (TCD.biter.e_link = 0) then<br>No channel-to-channel linking (or chaining) is performed after the inner "minor" loop is exhausted. TCD word 5, bits [30:25] are used to form a 15 bit biter field.<br><br>else<br>After the "minor" loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by biter.linkch[5:0] by setting that channel’s TCD.start bit.<br><br>The value contained in biter.linkch[5:0] must not exceed the number of implemented channels. |
| biter[] | Beginning “major” iteration count | This is the initial value copied into the citer field or citer.linkch field when the major loop is completed. The citer fields controls the iteration count and linking during channel execution.<br><br>This 9- or 15-bit count represents the beginning major loop count for the channel. As the major iteration count is exhausted, the contents of the entire 16-bit biter entry is reloaded into the 16-bit citer entry.<br><br>When the biter field is initially loaded by software, it must be set to the same value as that contained in the citer field.<br><br>If the channel is configured to execute a single service request, the initial values of biter and citer should be 0x0001. |

**Table 27. TCDn Word 7 (TCDn.{biter, control/status}) field descriptions (continued)**

| Name | Description | Value |
|---|---|---|
| bwc[] | Bandwidth control | This two-bit field provides a mechanism to effectively throttle the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the inner minor loop, it continuously generates read/write, read/write, ... sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the platform's cross-bar arbitration switch. To minimize start-up latency, bandwidth control stalls are suppressed for the first two AHB bus cycles and after the last write of each minor loop.<br><br>The dynamic priority elevation setting elevates the priority of the eDMA as seen by the cross-bar arbitration switch for the executing channel. Dynamic priority elevation is suppressed during the first two AHB bus cycles.<br><br>00 No eDMA engine stalls<br>01 Dynamic priority elevation<br>10 eDMA engine stalls for 4 cycles after each r/w<br>11 eDMA engine stalls for 8 cycles after each r/w |
| major.linkch[] | Link channel number | if (TCD.major.e_link = 0) then<br>No channel-to-channel linking (or chaining) is performed after the outer "major" loop counter is exhausted.<br><br>else<br>After the "major" loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by major.linkch[5:0] by setting that channel's TCD.start bit.<br><br>The value contained in major.linkch[5:0] must not exceed the number of implemented channels. |
| done | Channel done | This flag indicates the eDMA has completed the outer major loop. It is set by the eDMA engine as the citer count reaches zero; it is cleared by software, or the hardware when the channel is activated.<br><br>This bit must be cleared in order to write the major.e_link or e_sg bits. |
| active | Channel active | This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA engine as the inner minor loop completes or if any error condition is detected. |

**Table 27. TCDn Word 7 (TCDn.{biter, control/status}) field descriptions (continued)**

| Name | Description | Value |
|---|---|---|
| major.e_link | Enable channel-to-channel linking on major loop complete | As the channel completes the outer major loop, this flag enables the linking to another channel, defined by major.linkch[5:0]. The link target channel initiates a channel service request via an internal mechanism that sets the TCD.start bit of the specified channel. *To support the dynamic linking coherency model, this field is forced to zero when written to while the TCD.done bit is set.*<br><br>0 The channel-to-channel linking is disabled.<br>1 The channel-to-channel linking is enabled. |
| e_sg | Enable scatter/gather processing | As the channel completes the outer major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses dlast_sga as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure which is loaded as the transfer control descriptor into the local memory. *To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCD.done bit is set.*<br><br>0 The current channel's TCD is "normal" format.<br>1 The current channel's TCD specifies a scatter gather format. The dlast_sga field provides a memory pointer to the next TCD to be loaded into this channel after the outer major loop completes its execution. |
| d_req | Disable request | If this flag is set, the eDMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.<br><br>0 The channel's DMAERQ bit is not affected.<br>1 The channel's DMAERQ bit is cleared when the outer major loop is complete. |
| int_half | Enable an interrupt when major counter is half complete | If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the DMAINT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (citer == (biter >> 1)). This halfway point interrupt request is provided to support double-buffered schemes or other types of data movement where the processor needs an early indication of the transfer's progress.<br><br>0 The half-point interrupt is disabled.<br>1 The half-point interrupt is enabled.<br>**Note:** If biter = 1, do not use int_half; use int_major instead. |

Table 27. TCDn Word 7 (TCDn.{biter, control/status}) field descriptions (continued)

| Name | Description | Value |
|---|---|---|
| int_maj | Enable an interrupt when major iteration count completes | If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the DMAINT register when the current major iteration count reaches zero.<br><br>0 The end-of-major loop interrupt is disabled.<br>1 cThe end-of-major loop interrupt is enabled. |
| start | Channel start | If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution.<br><br>0 The channel is not explicitly started.<br>1 The channel is explicitly started via a software initiated service request. |

## 18.3 Functional description

This section provides an overview of the microarchitecture and functional operation of the eDMA module.

### 18.3.1 eDMA microarchitecture

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer control descriptor local memory. Additionally, the eDMA engine is further partitioned into four submodules, which are detailed below.

- eDMA engine
  - *addr_path*: This module implements registered versions of two channel transfer control descriptors: channel “x” and channel “y”, and is responsible for all the master bus address calculations. All the implemented channels provide the exact same functionality. This hardware structure allows the data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel service request is asserted while the first channel is active. Once a channel is activated, it runs until the minor loop is completed unless preempted by a higher priority channel. This capability provides a mechanism (optionally enabled by DCHPRIn[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution.

    When any other channel is activated, the contents of its transfer control descriptor is read from the local memory and loaded into the registers of the other addr_path.channel_{x,y}. Once the inner minor loop completes execution, the addr_path hardware writes the new values for the TCDn.{saddr, daddr, citer} back into the local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCDn.citer field, and a possible fetch of the next TCDn from memory as part of a scatter/gather operation.
  - *data_path*: This module implements the actual bus master read/write datapath. It includes 32 bytes of register storage (matching the maximum transfer size) and the necessary mux logic to support any required data alignment. The AMBA-AHB read data bus is the primary input, and

the AHB write data bus is the primary output.

The addr_ and data_path modules directly support the 2-stage pipelined AMBA-AHB bus. The addr_path module represents the 1st stage of the bus pipeline (the address phase), while the data_path module implements the 2nd stage of the pipeline (the data phase).

  - *pmodel_charb*: This module implements the first section of eDMA's programming model as well as the channel arbitration logic. The programming model registers are connected to the IPS bus (not shown). The ipd_req[n] inputs and dma_ipi_int[n] outputs are also connected to this module (via the control logic).
  - *control*: This module provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read, destination write operations until the number of bytes specified in the inner 'minor loop' byte count has been moved. For descriptors where the sizes are not equal, multiple access of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

- transfer_control_descriptor local memory
  - *memory controller*: This logic implements the required dual-ported controller, handling accesses from both the eDMA engine as well as references from the IPS bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the IPS transaction is stalled. The hooks to a BIST controller for the local TCD memory are included in this module.
  - *memory array*: The TCD is implemented using a single-ported, synchronous compiled RAM memory array

### 18.3.2 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments. As shown in Figure 18-28, the first segment involves the channel service request. In the diagram, this example uses the assertion of the ipd_req[n] signal to request service for channel n. Channel service request via software and the TCDn.start bit follows the same basic flow as an ipd_req. The ipd_req[n] input signal is registered internally and then routed to through the DMA engine, first through the control module, then into the programming model/channel arbitration (pmodel_charb) module. In the next cycle, the channel arbitration is performed, either using the fixed-priority or round-robin algorithm. After the arbitration is complete, the activated channel number is sent through the address path (addr_path) and converted into the required address to access the TCD local memory. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the dma_engine.addr_path.channel_{x,y} registers. The TCD memory is organized 64-bits in width to minimize the time needed to fetch the activated channel's descriptor and load it into the dma_engine.addr_path.channel_{x,y} registers.

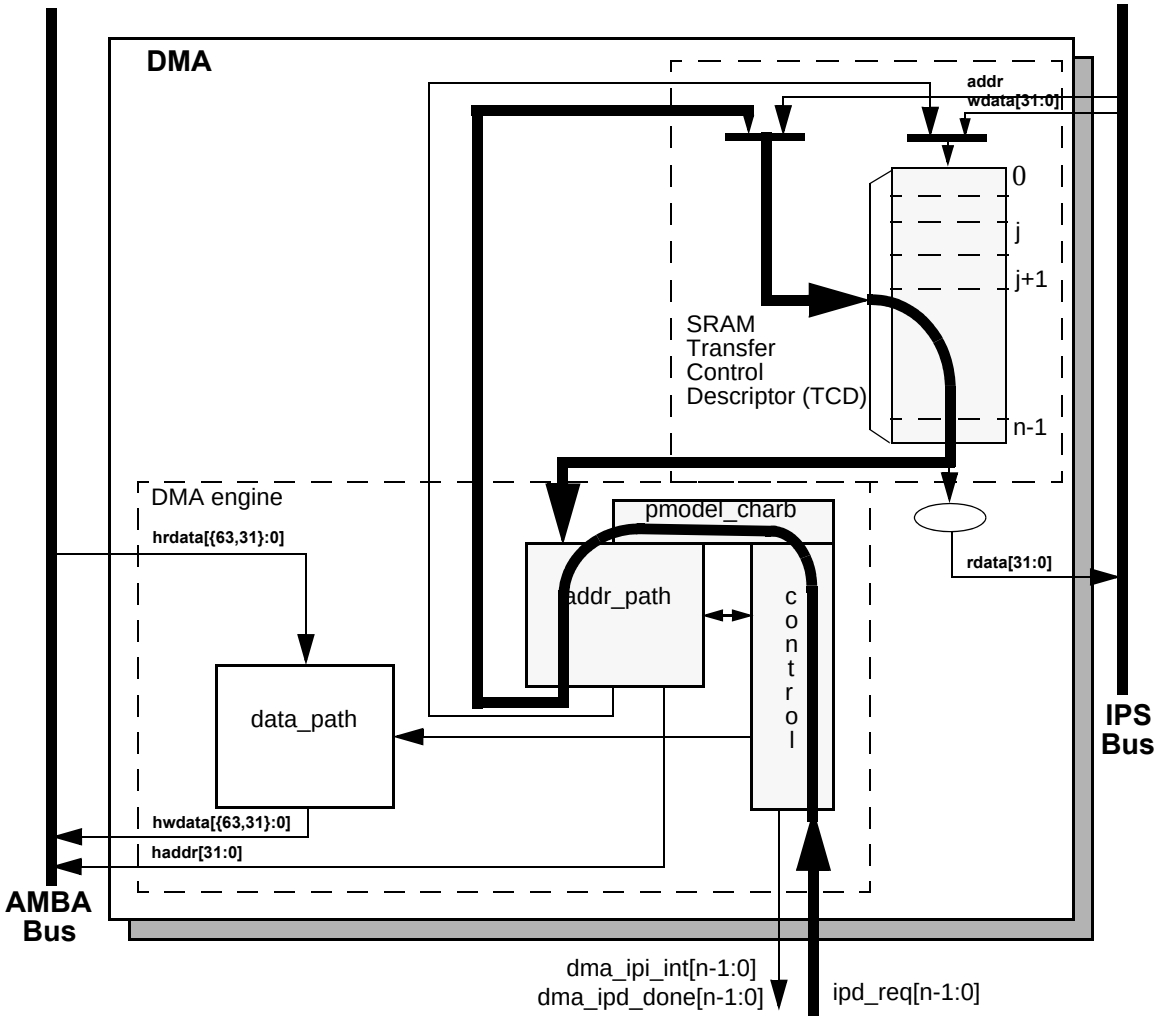



**Figure 18-28. eDMA operation, part 1**

In the second part of the basic data flow as shown in Figure 18-29, the modules associated with the data transfer (addr_path, data_path and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data_path module until it is gated onto the AMBA-AHB bus during the destination write. This source read/destination write processing continues until the inner minor byte count has been transferred. The dma_ipd_done[n] signal is asserted at the end of the minor byte count transfer.

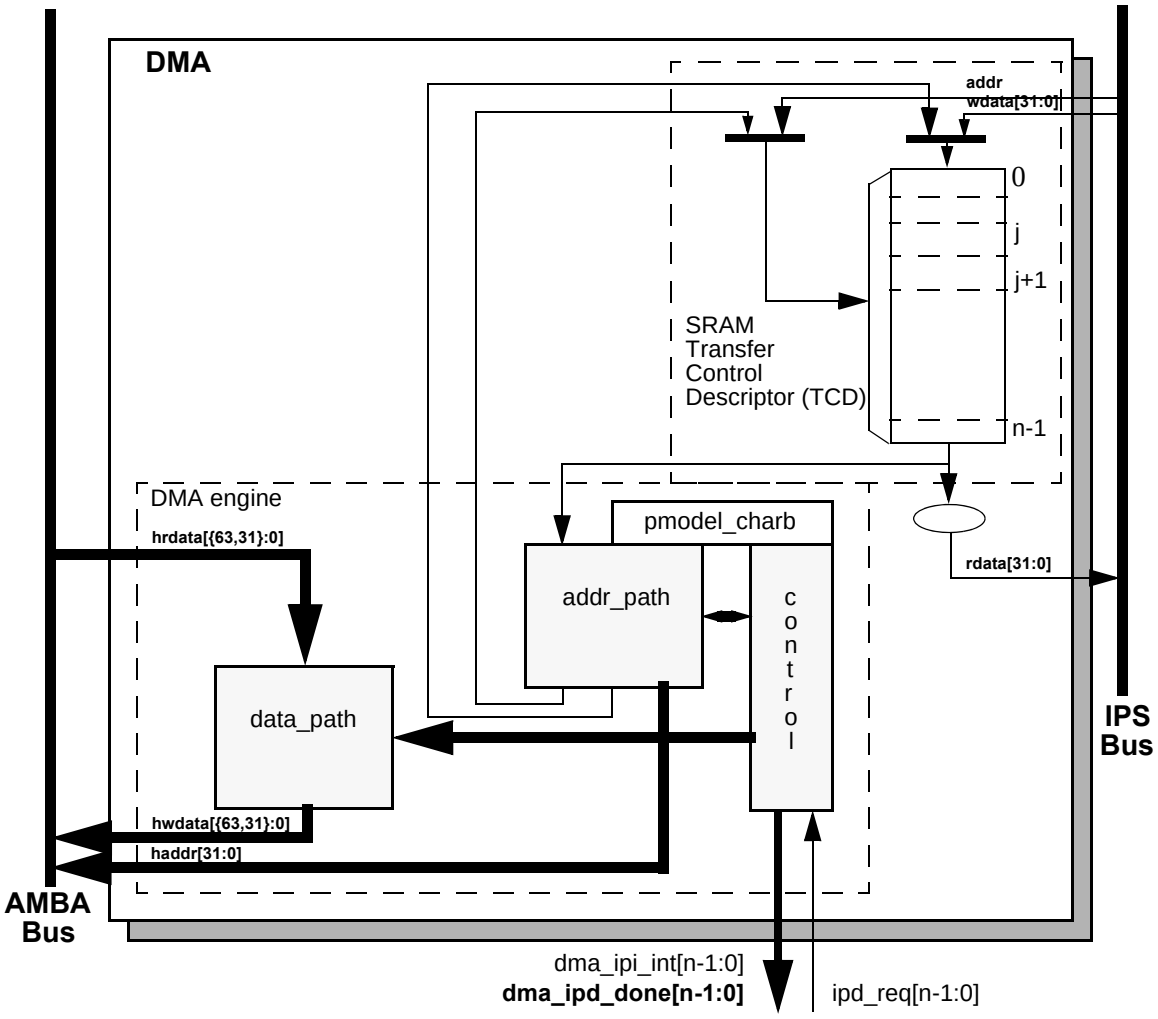



**Figure 18-29. eDMA operation, part 2**

Once the inner minor byte count has been moved, the final phase of the basic data flow is performed. In this segment, the addr_path logic performs the required updates to certain fields in the channel's TCD, e.g., saddr, daddr, citer. If the outer major iteration count is exhausted, then there are additional operations which are performed. These include the final address adjustments and reloading of the biter field into the citer. Additionally, assertion of an optional interrupt request occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor. The updates to the TCD memory and the assertion of an interrupt request are shown in Figure 18-30.

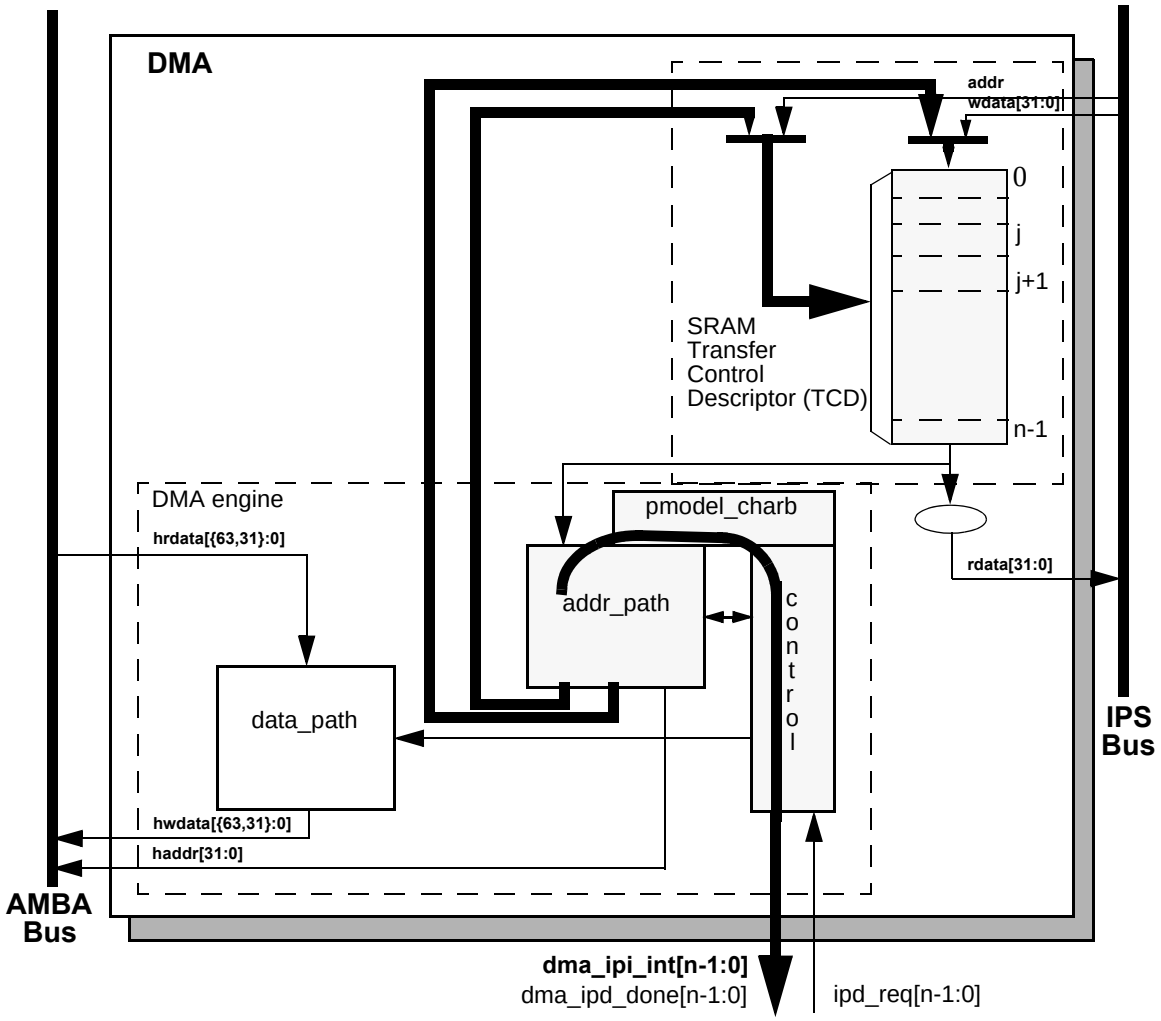



**Figure 18-30. eDMA operation, part 3**

## 18.3.3 eDMA performance

This section addresses the performance of the eDMA module, focusing on two separate metrics. In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces. In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests which can be serviced in a fixed time is a more interesting metric. In this environment, the speed of the source and destination address spaces remains important, but the microarchitecture of the eDMA also factors significantly into the resulting metric.

The peak transfer rates for several different source and destination transfers are shown in Table 18-1. The following assumptions apply to Table 18-1 and Table 18-2:

- Platform SRAM can be accessed with zero wait-states when viewed from the AMBA-AHB data phase

- All IPS reads require two wait-states, and IPS writes three wait-states, again viewed from the system bus data phase
- All IPS accesses are 32 bits in size

Table 18-1 presents a peak transfer rate comparison, measured in megabytes per second. In this table, the Platform_SRAM-to-Platform_SRAM transfers occur at the native platform datapath width, i.e., either 32- or 64-bits per access. For all transfers involving the IPS bus, 32-bit transfer sizes are used. In all cases, the transfer rate includes the time to read the source plus the time to write the destination.

**Table 18-1. eDMA peak transfer rates [MB/s]**

| Platform Speed, Width | Platform SRAM-to-Platform SRAM | 32-bit IPS-to-Platform SRAM | Platform SRAM-to-32-bit IPS |
|---|---|---|---|
| 66.7 MHz, 32-bit | 133.3 | 66.7 | 53.3 |
| 66.7 MHz, 64-bit | 266.7 | 66.6 | 53.3 |
| 83.3 MHz, 32-bit | 166.7 | 83.3 | 66.7 |
| 83.3 MHz, 64-bit | 333.3 | 83.3 | 66.7 |
| 100.0 MHz, 32-bit | 200.0 | 100.0 | 80.0 |
| 100.0 MHz, 64-bit | 400.0 | 100.0 | 80.0 |
| 133.3 MHz, 32-bit | 266.7 | 133.3 | 106.7 |
| 133.3 MHz, 64-bit | 533.3 | 133.3 | 106.7 |
| 150.0 MHz, 32-bit | 300.0 | 150.0 | 120.0 |
| 150.0 MHz, 64-bit | 600.0 | 150.0 | 120.0 |

The second performance metric is a measure of the number of DMA requests which can be serviced in a given amount of time. For this metric, it is assumed the peripheral request causes the channel to move a single IPS-mapped operand to/from the platform SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel. The DMA design supports the following hardware service request sequence:

- Cycle 1: ipd_req[n] is asserted
- Cycle 2: The ipd_req[n] is registered locally in the DMA module and qualified (TCD.start bit initiated requests start at this point with the registering of the IPS write to TCD word7)
- Cycle 3: Channel arbitration begins
- Cycle 4: Channel arbitration completes. The transfer control descriptor local memory read is initiated.
- Cycle 5 - 6: The first two parts of the activated channel’s TCD is read from the local memory. The memory width to the DMA engine is 64 bits, so the entire descriptor can be accessed in four cycles.
- Cycle 7: The first AMBA-AHB read cycle is initiated, as the third part of the channel’s TCD is read from the local memory. Depending on the state of the platform’s crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
- Cycle 8 - ?: The last part of the TCD is read in. This cycle represents the 1st data phase for the read, and the address phase for the destination write.
  The exact timing from this point is a function of the response times for the channel’s read and write accesses. In this case of an IPS read and a platform SRAM write, the combined data phase time is 4 cycles. For an SRAM read and IPS write, it is 5 cycles.

- Cycle ?+1: This cycle represents the data phase of the last destination write
- Cycle ?+2: The DMA engine completes the execution of the inner minor loop and prepares to write back the required TCDn fields into the local memory. TCD word7 is read and checked for channel linking or scatter/gather requests.
- Cycle ?+3: The appropriate fields in the first part of the TCDn are written back into the local memory
- Cycle ?+4: The fields in the second part of the TCDn are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
- Cycle ?+5: The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the AHB system bus, DMA requests can be processed every 9 cycles. Assuming an average of the access times associated with IPS-to-SRAM (4 cycles) and SRAM-to-IPS (5 cycles), eDMA requests can be processed every 11.5 cycles (4 + (4+5)÷2 + 3). This is the time from Cycle 4 to Cycle "?+5". The resulting peak request rate, as a function of the platform frequency, is shown in Table 18-2. This metric represents millions of requests per second.

**Table 18-2. eDMA peak request rate [MReq/sec]**

| Platform Speed | Request Rate (zero wait state) | Request Rate (with wait states) |
|---|---|---|
| 66.6 MHz | 7.4 | 5.8 |
| 83.3 MHz | 9.2 | 7.2 |
| 100.0 MHz | 11.1 | 8.7 |
| 133.3 MHz | 14.8 | 11.6 |
| 150.0 MHz | 16.6 | 13.0 |

A general formula to compute the peak request rate (with overlapping requests) is:

PEAKreq = freq ÷ [ entry + (1 + read_ws) + (1 + write_ws) + exit ]

where:

PEAKreq - peak request rate

freq - platform frequency

entry - channel startup (4 cycles)

read_ws - wait states seen during the system bus read data phase

write_ws - wait states seen during the system bus write data phase

exit - channel shutdown (3 cycles)

For example: consider a platform with the following characteristics:

- Platform SRAM can be accessed with one wait-state when viewed from the AMBA-AHB data phase
- All IPS reads require two wait-states, and IPS writes three wait-states, again viewed from the system bus data phase
- Platform operates at 150 MHz

For an SRAM to IPS transfer:

PEAKreq = 150 MHz ÷ [ 4 + (1 + 1) + (1 + 3) + 3 ] cycles = 11.5 Mreq/sec

For an IPS to SRAM transfer:

PEAKreq = 150 MHz ÷ [ 4 + (1 + 2) + (1 + 1) + 3 ] cycles = 12.5 Mreq/sec

Assuming an even distribution of the two transfer types, the average Peak Request Rate would be:

PEAKreq = (11.5 Mreq/sec + 12.5 Mreq/sec) ÷ 2 = 12.0 Mreq/sec

The minimum number of cycles to perform a single read/write, zero wait states on the system bus, from a cold start (where no channel is executing, DMA is idle) are:

- 11 cycles for a software (TCD.start bit) request
- 12 cycles for a hardware (ipd_req signal) request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the ipd_req signals. For the peak request rate calculations above, the arbitration and request registering is absorbed in or overlap the previous executing channel.

**NOTE**

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

## 18.4 Initialization/application information

### 18.4.1 DMA initialization

A typical initialization of the DMA is:

1. Write the DMACR register if a configuration other than the default is desired.
2. Write the channel priority levels into the DCHPRIn registers if a configuration other than the default is desired.
3. Enable error interrupts in the DMAEEI registers if so desired.
4. Write the 32 byte TCD for each channel that may request service.
5. Enable any hardware service requests via the DMAERQ register.

6. Request channel service by either software (setting the TCD.start bit) or by hardware (slave device asserting its ipd_req signal).

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The DMA engine will read the entire TCD for the selected channel into its internal address path module. As the TCD is being read, the first transfer is initiated on the AHB bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCD.saddr) to the destination (as defined by the destination address, TCD.daddr) continue until the specified number of bytes (TCD.nbytes) have been transferred. When the transfer is complete, the DMA engine's local TCD.saddr, TCD.daddr, and TCD.citer are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing is executed, i.e. interrupts, major loop channel linking, and scatter/gather operations, if enabled.

### 18.4.2 DMA programming errors

The DMA performs various tests on the Transfer Control Descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of two errors; Group Priority Error and Channel Priority Error, GPE and CPE in the DMAES register respectively.

For all error types other than Group or Channel Priority Errors, the channel number causing the error is recorded in the DMAES register. If the error source is not removed before the next activation of the problem channel, the error will be detected and recorded again.

The sequence listed below is correct. For item 2, the dma_ipd_ack{done} lines will assert only if the selected channel is requesting service via the ipd_req signal. I think the typical application will enable error interrupts for all channels. So the user will get an error interrupt, but the channel number for the DMAERR register and the error interrupt request line may be wrong because they reflect the selected channel.

Channel priority errors are identified within a group after that group has been selected as the active group. For example:

1. The DMA is configured for fixed group and fixed channel arbitration modes.
2. Group3 is the highest priority and all channels are unique in that group.
3. Group2 is the next highest priority and has two channels with the same priority level.
4. If Group3 has any service requests, those requests will be executed.
5. Once all of Group3 requests have completed, Group2 will be the next active group.
6. If Group2 has a service request, then an undefined channel in Group2 will be selected and a channel priority error will occur.
7. This will repeat until the all of Group2 requests have been removed or a higher priority Group3 request comes in.

A group priority error is global and any request in any group will cause a group priority error.

In general, if priority levels are not unique, the highest (channel/group) priority that has an active request will be selected, but the lowest numbered (channel/group) with that priority will be selected by arbitration

and executed by the DMA engine. The hardware service request handshake signals, error interrupts and error reporting will be associated with the selected channel.

## 18.4.3 DMA arbitration mode considerations

### 18.4.3.1 Fixed group arbitration, fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group will be selected to execute. If the DMA is programmed so the channels within one group use “fixed” priorities, and that group is assigned the highest “fixed” priority of all groups, it is possible for that group to take all the bandwidth of the DMA controller — i.e. no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request.

The advantage of this scenario is that latency can be small for channels that need to be serviced quickly.

Preemption is available in this scenario only.

### 18.4.3.2 Round-robin group arbitration, fixed channel arbitration

The occurrence of one or more DMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with an service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round robin algorithm will select the highest pending request from the next group in the round robin sequence. Servicing continues round robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced.

The advantage of this scenario is that no one group uses all the DMA bandwidth.

The highest priority channel selection latency is potentially greater than fixed/fixed arbitration.

Excessive request rates on high priority channels can prevent the servicing of lower priority channels in the same group.

### 18.4.3.3 Round-robin group arbitration, round-robin channel arbitration

Groups will be serviced as described in Section 18.4.3.2, “Round-robin group arbitration, fixed channel arbitration, but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round robin manner, any channel that generates DMA requests faster than a combination of the group round robin service rate and the channel service rate for its group will not prevent the servicing of other channels in its group. Any DMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high.

All channels are treated equally. Priority levels are not used in round robin/round robin mode.

#### 18.4.3.4 Fixed group arbitration, round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Section 18.4.3.1, “Fixed group arbitration, fixed channel arbitration, but all the channels in the highest priority group will be serviced.

Service latency will be short on the highest priority group, but can become much longer as the group priority decreases.

### 18.4.4 DMA transfer

#### 18.4.4.1 Single request

To perform a simply transfer of ‘n’ bytes of data with one activation, set the major loop to one (TCD.citer = TCD.biter = 1). The data transfer will begin after the channel service request is acknowledged and the channel is selected to execute. Once the transfer is complete, the TCD.done bit will be set and an interrupt will be generated if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The DMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a word wide port located at 0x2000. The address offsets are programmed in increments to match the size of the transfer; one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

TCD.citer = TCD.biter = 1

TCD.nbytes = 16

TCD.saddr = 0x1000

TCD.soff = 1

TCD.ssize = 0

TCD.slast = -16

TCD.daddr = 0x2000

TCD.doff = 4

TCD.dsize = 2

TCD.dlast_sga= -16

TCD.int_maj = 1

TCD.start = 1 (TCD.word7 should be written last after all other fields have been initialized)

All other TCD fields = 0

This generates the following sequence of events:

1. IPS write to the TCD.start bit requests channel service
2. The channel is selected by arbitration for servicing
3. DMA engine writes: TCD.done = 0, TCD.start = 0, TCD.active = 1
4. DMA engine reads: channel TCD data from local memory to internal register file
5. The source to destination transfers are executed as follows:
   a. read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003)
   b. write_word(0x2000) -> *first iteration of the minor loop*
   c. read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007)
   d. write_word(0x2004) -> *second iteration of the minor loop*
   e. read_byte(0x1008), read_byte(0x1009), read_byte(0x100a), read_byte(0x100b)
   f. write_word(0x2008) -> *third iteration of the minor loop*
   g. read_byte(0x100c), read_byte(0x100d), read_byte(0x100e), read_byte(0x100f)
   h. write_word(0x200c) -> *last iteration of the minor loop -> major loop complete*
6. DMA engine writes: TCD.saddr = 0x1000, TCD.daddr = 0x2000, TCD.citer = 1 (TCD.biter)
7. DMA engine writes: TCD.active = 0, TCD.done = 1, DMAINT[n] = 1
8. The channel retires

The DMA goes idle or services next channel.

#### 18.4.4.2 Multiple requests

The next example is the same as the previous example, with the exception of transferring 32 bytes via two hardware requests. The only fields that change are the major loop iteration count and the final address offsets. The DMAis programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests is enabled in the DMAERQ register, channel service requests are initiated by the slave device.

TCD.citer = TCD.biter = 2

TCD.slast = -32

TCD.dlast_sga = -32

This would generate the following sequence of events:

1. First hardware (ipd_req) request for channel service
2. The channel is selected by arbitration for servicing
3. DMA engine writes: TCD.done = 0, TCD.start = 0, TCD.active = 1
4. DMA engine reads: channel TCD data from local memory to internal register file
5. The source to destination transfers are executed as follows:
   a. read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003)
   b. write_word(0x2000) -> *first iteration of the minor loop*
   c. read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007)
   d. write_word(0x2004) -> *second iteration of the minor loop*
   e. read_byte(0x1008), read_byte(0x1009), read_byte(0x100a), read_byte(0x100b)
   f. write_word(0x2008) -> *third iteration of the minor loop*
   g. read_byte(0x100c), read_byte(0x100d), read_byte(0x100e), read_byte(0x100f)
   h. write_word(0x200c) -> *last iteration of the minor loop*
6. DMA engine writes: TCD.saddr = 0x1010, TCD.daddr = 0x2010, TCD.citer = 1
7. DMA engine writes: TCD.active = 0
8. The channel retires -> *one iteration of the major loop*

The DMA goes idle or services next channel.

9. Second hardware (ipd_req) requests channel service
10. The channel is selected by arbitration for servicing
11. DMA engine writes: TCD.done = 0, TCD.start = 0, TCD.active = 1
12. DMA engine reads: channel TCD data from local memory to internal register file
13. The source to destination transfers are executed as follows:
    a. read_byte(0x1010), read_byte(0x1011), read_byte(0x1012), read_byte(0x1013)
    b. write_word(0x2010) -> *first iteration of the minor loop*
    c. read_byte(0x1014), read_byte(0x1015), read_byte(0x1016), read_byte(0x1017)
    d. write_word(0x2014) -> *second iteration of the minor loop*
    e. read_byte(0x1018), read_byte(0x1019), read_byte(0x101a), read_byte(0x101b)
    f. write_word(0x2018) -> *third iteration of the minor loop*
    g. read_byte(0x101c), read_byte(0x101d), read_byte(0x101e), read_byte(0x101f)

h. write_word(0x201c) -> *last iteration of the minor loop -> major loop complete*

14. DMA engine writes: TCD.saddr = 0x1000, TCD.daddr = 0x2000, TCD.citer = 2 (TCD.biter)
15. DMA engine writes: TCD.active = 0, TCD.done = 1, DMAINT[n] = 1
16. The channel retires -> *major loop complete*

The DMA goes idle or services the next channel.

## 18.4.5 TCD status

### 18.4.5.1 Minor loop complete

There are two methods to test for minor loop completion when using software initiated service requests. The first method is to read the TCD.citer field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD.start bit AND the TCD.active bit. The minor loop complete condition is indicated by both bits reading zero after the TCD.start was written to a one. Polling the TCD.active bit may be inconclusive because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

1. TCD.start = 1, TCD.active = 0, TCD.done = 0 (channel service request via software)
2. TCD.start = 0, TCD.active = 1, TCD.done = 0 (channel is executing)
3. TCD.start = 0, TCD.active = 0, TCD.done = 0 (channel has completed the minor loop and is idle)
   or
   TCD.start = 0, TCD.active = 0, TCD.done = 1 (channel has completed the major loop and is idle)

The best method to test for minor loop completion when using hardware initiated service requests is to read the TCD.citer field and test for a change. The hardware request and acknowledge handshakes signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware activated channel:

1. ipd_req asserts (channel service request via hardware)
2. TCD.start = 0, TCD.active = 1, TCD.done = 0 (channel is executing)
3. TCD.start = 0, TCD.active = 0, TCD.done = 0 (channel has completed the minor loop and is idle)
   or
   TCD.start = 0, TCD.active = 0, TCD.done = 1 (channel has completed the major loop and is idle)

For both activation types, the major loop complete status is explicitly indicated via the TCD.done bit.

The TCD.start bit is cleared automatically when the channel begins execution regardless of how the channel was activated.

#### 18.4.5.2 Active channel TCD reads

The DMA will read back the ‘true’ TCD.saddr, TCD.daddr, and TCD.nbytes values if read while a channel is executing. The ‘true’ values of the saddr, daddr, and nbytes are the values the DMA engine is currently using in its internal register file and not the values in the TCD local memory for that channel. The addresses (saddr and daddr) and nbytes (decrements to zero as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

#### 18.4.5.3 Preemption status

Preemption is only available when *fixed* arbitration is selected for *both* group and channel arbitration modes. A preempt-able situation is one in which a preempt-enabled channel is running and a higher priority request becomes active. When the DMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the relative priority of the actively running and the outstanding requests become undefined. Channel and/or group priorities are treated as equal (constantly rotating) when round-robin arbitration mode is selected.

The TCD.active bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one iteration of the major loop. Two TCD.active bits set at the same time in the overall TCD map indicates a higher priority channel is actively preempting a lower priority channel.

The worst case latency when switching to a preempt channel is the summation of:

- arbitration latency (2 cycles)
- bandwidth control stalls (if enabled)
- the time to execute two read/write sequences (including AHB bus holds; a system dependency driven by the slave devices or the crossbar)

### 18.4.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD.start bit of another channel (or itself) thus initiating a service request for that channel. This operation is automatically performed by the DMA engine at the conclusion of the major or minor loop when properly enabled.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD.citer.e_link field are used to determine whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, with the initial fields of:

TCD.citer.e_link = 1

TCD.citer.linkch = 0xC

TCD.citer value = 0x4

TCD.major.e_link = 1

TCD.major.linkch = 0x7

will execute as:

1. minor loop done -> set channel 12 TCD.start bit
2. minor loop done -> set channel 12 TCD.start bit
3. minor loop done -> set channel 12 TCD.start bit
4. minor loop done, major loop done -> set channel 7 TCD.start bit

When minor loop linking is enabled (TCD.citer.e_link = 1), the TCD.citer field uses a nine bit vector to form the current iteration count.

When minor loop linking is disabled (TCD.citer.e_link = 0), the TCD.citer field uses a 15 bit vector to form the current iteration count. The bits associated with the TCD.citer.linkch field are concatenated onto the citer value to increase the range of the citer.

**NOTE**

The TCD.citer.e_link bit and the TCD.biter.e_link bit must equal or a configuration error will be reported. The citer and biter vector widths must be equal to calculate the major loop, half-way done interrupt point.

## 18.4.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

### 18.4.7.1 Dynamic priority changing

The following two options are recommended for dynamically changing ***channel*** priority levels:

1. Switch to round-robin channel arbitration mode, change the channel priorities, then switch back to fixed arbitration mode.
2. Disable all the channels within a group, then change the channel priorities within that group only, then enable the appropriate channels.

The following two options are available for dynamically changing ***group*** priority levels:

1. Switch to round-robin group arbitration mode, change the group priorities, then switch back to fixed arbitration mode.
2. Disable ALL channels, change the group priorities, then enable the appropriate channels.

### 18.4.7.2 Dynamic channel linking and dynamic scatter/gather

Dynamic channel linking and dynamic scatter/gather is the process of changing the TCD.major.e_link or TCD.e_sg bits during channel execution. These bits are read from the TCD local memory at the ***end*** of channel execution thus allowing the user to enable either feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the DMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer’s model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link or dynamic scatter/gather request:

1. Set the TCD.major.e_link bit.
2. Read back the TCD.major.e_link bit.
3. Test the TCD.major.e_link request status:
   a. If the bit is set, the dynamic link attempt was successful.
   b. If the bit is cleared, the attempted dynamic link did not succeed, the channel was already retiring.

This same coherency model is true for dynamic scatter/gather operations. For both dynamic requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set indicating the major loop is complete.

**NOTE**

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the DMA engine after a channel begins execution.

## 18.4.8 Hardware request release timing

This section provides a timing diagram for deasserting the ipd_req hardware request signal. Figure 18-31 shows two read write sequences with grey indicating the release of the ipd_req hardware request signal.

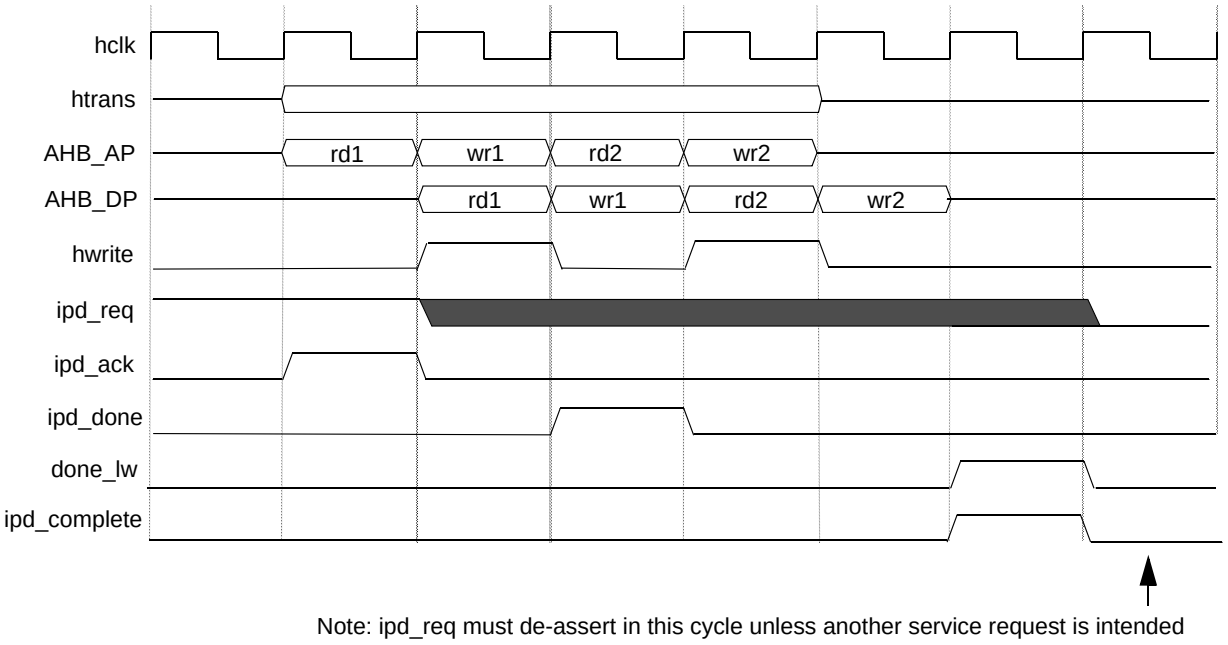



**Figure 18-31. ipd_req hardware handshake**

This page is intentionally left blank.

# Chapter 19
# Error Correction Status Module (ECSM)

## 19.1 Introduction

The Error Correction Status Module (ECSM) provides a myriad of miscellaneous control functions for the device including program-visible information about configuration, a reset status register, and optional features such as information on memory errors reported by error-correcting codes. Its Supervisor-mode access protection feature provides access protection for slave modules INTC, ECSM, MPU, PRAM2P, STM, and SWT.

## 19.2 Overview

The Error Correction Status Module is mapped into the IPS space and supports a number of miscellaneous control functions for the device.

## 19.3 Features

The ECSM includes these features:

- Program-visible information on the device configuration and revision
- Optional registers for capturing information on memory errors if error-correcting codes (ECC) are implemented
- Optional registers to specify the generation of single- and double-bit memory data inversions for test purposes if error-correcting codes are implemented
- Spp_ips_reg_protection provides privileged-only access to selected on-platform slave devices: INTC, ECSM, MPU, STM, and SWT.

## 19.4 Memory map and register description

### 19.4.1 Memory map

Table 19-1 provides the register map of the ECSM.

**Table 19-1. ECSM memory map**

| Address offset | Register | Location |
|---|---|---|
| 0x00–0x20 | Reserved | |
| 0x24 | Miscellaneous User-Defined Control Register (MUDCR) | on page 752 |
| 0x28–0x42 | Reserved | |
| 0x43 | ECC Configuration Register (ECR) | on page 754 |
| 0x44–0x46 | Reserved | |
| 0x47 | ECC Status Register (ESR) | on page 755 |

Table 19-1. ECSM memory map (continued)

| Address offset | Register | Location |
|---|---|---|
| 0x48–0x49 | Reserved | |
| 0x4A | ECC Error Generation Register (EEGR) | on page 757 |
| 0x4C–0x4F | Reserved | |
| 0x50 | Flash ECC Address Register (FEAR) | on page 760 |
| 0x54–0x55 | Reserved | |
| 0x56 | Flash ECC Master Number Register (FEMR) | on page 761 |
| 0x57 | Flash ECC Attributes (FEAT) | on page 762 |
| 0x58–5B | Reserved | |
| 0x5C | Flash ECC Data Register (FEDR) | on page 762 |
| 0x60 | RAM ECC Address Register (REAR) | on page 763 |
| 0x64 | Reserved | |
| 0x65 | RAM ECC Syndrome Register (RESR) | on page 764 |
| 0x66 | RAM ECC Master Register (REMR) | on page 766 |
| 0x67 | RAM ECC Attributes (REAT) | on page 767 |
| 0x68–0x6B | Reserved | |
| 0x6C | RAM ECC Data Register (REDR) | on page 767 |

## 19.4.2 Register description

Attempted accesses to reserved addresses result in an error termination, while attempted writes to read-only registers are ignored and do not terminate with an error. Unless noted otherwise, writes to the programming model must match the size of the register, e.g., an n-bit register only supports n-bit writes, etc. Attempted writes of a different size than the register width produce an error termination of the bus cycle and no change to the targeted register.

### 19.4.2.1 Miscellaneous User-Defined Control Register (MUDCR)

The MUDCR provides a program-visible register for user-defined control functions. It typically is used as configuration control for miscellaneous device-level modules. The contents of this register is simply output from ECSM to other modules where the user-defined control functions are implemented. See Figure 19-1 and Table 19-2 for the Miscellaneous User-Defined Control Register definition.

Register address: ECSM Base + 0x24

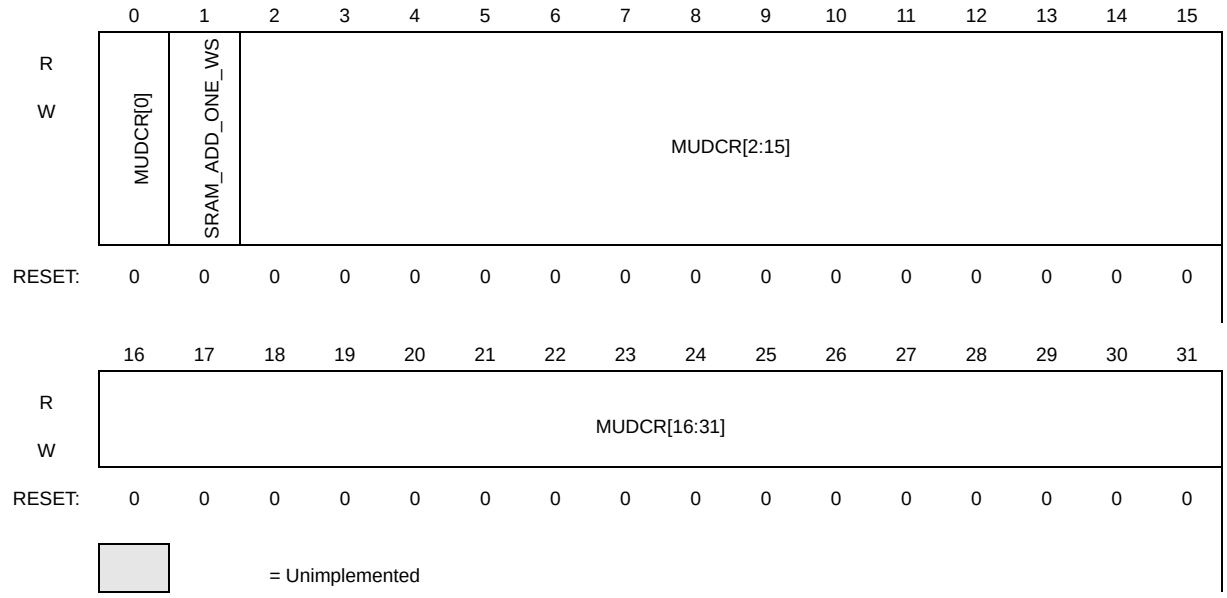



**Figure 19-1. Miscellaneous User-Defined Control (MUDCR) Register**

**Table 19-2. Miscellaneous User-Defined Control Register (MUDCR) field descriptions**

| Field De | scription |
|---|---|
| MUDCR[0], MUDCR[2:31] | User-Defined control Register<br>0 = The control associated with this MUDCR bit is disabled.<br>1 = The control associated with this MUDCR bit is enabled. |
| SRAM_ADD_ ONE_WS | Platform RAM wait-state control.<br>This bit is used to select whether the platform RAM controller will insert 1-wait state into every read access made to the platform RAM arrays. One wait state is required when the System Frequency is >64MHz.<br>0 = The platform RAM controller operates as a 0-wait state controller<br>1 = The platform RAM controller operates as a 1-wait state controller<br>**Note:** This bit must not be changed while there are ongoing transfer on the SRAM. |

### 19.4.2.2 ECC registers

For designs including error-correcting code (ECC) implementations to improve the quality and reliability of memories, there are a number of program-visible registers for the sole purpose of reporting and logging of memory failures. These optional registers include:

- ECC Configuration Register (ECR)
- ECC Status Register (ESR)
- ECC Error Generation Register (EEGR)
- Flash ECC Address Register (FEAR)
- Flash ECC Master Number Register (FEMR)
- Flash ECC Attributes Register (FEAT)
- Flash ECC Data Register (FEDR)

- RAM ECC Address Register (REAR)
- RAM ECC Syndrome Register (RESR)
- RAM ECC Master Number Register (REMR)
- RAM ECC Attributes Register (REAT)
- RAM ECC Data Register (REDR)

The details on the ECC registers are provided in the subsequent sections. If the design does not include ECC on the memories, these addresses are reserved locations within the ECSM's programming model.

### 19.4.2.3 ECC Configuration Register (ECR)

The ECC Configuration Register is an 8-bit control register for specifying which types of memory errors are reported. In all systems with ECC, the occurrence of a non-correctable error causes the current access to be terminated with an error condition. In many cases, this error termination is reported directly by the initiating bus master. However, there are certain situations where the occurrence of this type of non-correctable error is not reported by the master. Examples include speculative instruction fetches which are discarded due to a change-of-flow operation, and buffered operand writes. The ECC reporting logic in the ECSM provides an optional error interrupt mechanism to signal all non-correctable memory errors. In addition to the interrupt generation, the ECSM captures specific information (memory address, attributes and data, bus master number, etc.) which may be useful for subsequent failure analysis.

The reporting of single-bit memory corrections can only be enabled via a an SoC-configurable module input signal. While not directly accessible to a user, this capability is viewed as important for error logging and failure analysis.

See Figure 19-2 and Table 19-3 for the ECC Configuration Register definition.

Register address: ECSM Base + 0x43

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | ER1BR | EF1BR | 0 | 0 | ERNCR | EFNCR |
| W | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 19-2. ECC Configuration (ECR) Register**

**Table 19-3. ECC Configuration (ECR) field descriptions**

| Field De | scription |
|---|---|
| ER1BR | Enable RAM 1-bit Reporting<br>0 = Reporting of single-bit RAM corrections is disabled.<br>1 = Reporting of single-bit RAM corrections is enabled.<br><br>This bit can only be set if the SoC-configurable input enable signal is asserted. The occurrence of a single-bit RAM correction generates a ECSM ECC interrupt request as signalled by the assertion of ESR[R1BC]. The address, attributes and data are also captured in the REAR, RESR, REMR, REAT and REDR registers. |

Table 19-3. ECC Configuration (ECR) field descriptions (continued)

| Field De | scription |
|---|---|
| EF1BR | Enable Flash 1-bit Reporting<br>0 = Reporting of single-bit flash corrections is disabled.<br>1 = Reporting of single-bit flash corrections is enabled.<br><br>This bit can only be set if the SoC-configurable input enable signal is asserted. The occurrence of a single-bit flash correction generates a ECSM ECC interrupt request as signalled by the assertion of ESR[F1BC]. The address, attributes and data are also captured in the FEAR, FEMR, FEAT and FEDR registers. |
| ERNCR | Enable RAM Non-Correctable Reporting<br>0 = Reporting of non-correctable RAM errors is disabled.<br>1 = Reporting of non-correctable RAM errors is enabled.<br><br>The occurrence of a non-correctable multi-bit RAM error generates a ECSM ECC interrupt request as signalled by the assertion of ESR[RNCE]. The faulting address, attributes and data are also captured in the REAR, RESR, REMR, REAT and REDR registers. |
| EFNCR | Enable Flash Non-Correctable Reporting<br>0 = Reporting of non-correctable flash errors is disabled.<br>1 = Reporting of non-correctable flash errors is enabled.<br><br>The occurrence of a non-correctable multi-bit flash error generates a ECSM ECC interrupt request as signalled by the assertion of ESR[FNCE]. The faulting address, attributes and data are also captured in the FEAR, FEMR, FEAT and FEDR registers. |

### 19.4.2.4 ECC Status Register (ESR)

The ECC Status Register is an 8-bit control register for signaling which types of properly-enabled ECC events have been detected. The ESR signals the last, properly-enabled memory event to be detected. ECC interrupt generation is separated into single-bit error detection/correction, uncorrectable error detection and the combination of the two as defined by the following boolean equations. In these equations, “&” refers to a bitwise AND operator and “|” refers to a bitwise OR operator; bitwise AND has precedence of bitwise OR.

Bitwise AND has precedence of bitwise OR.

ECSM_ECC1BIT_IRQ

= ECR[ER1BR] & ESR[R1BC]// ram, 1-bit correction

| ECR[EF1BR] & ESR[F1BC]// flash, 1-bit correction

ECSM_ECCRNCR_IRQ

= ECR[ERNCR] & ESR[RNCE]// ram, noncorrectable error

ECSM_ECCFNCR_IRQ

= ECR[EFNCR] & ESR[FNCE]// flash, noncorrectable error

ECSM_ECC2BIT_IRQ

= ECSM_ECCRNCR_IRQ// ram, noncorrectable error

| ECSM_ECCFNCR_IRQ// flash, noncorrectable error

ECSM_ECC_IRQ

= ECSM_ECC1BIT_IRQ // 1-bit correction

| ECSM_ECC2BIT_IRQ// noncorrectable error

where the combination of a properly-enabled category in the ECR and the detection of the corresponding condition in the ESR produces the interrupt request.

The ECSM allows a maximum of one bit of the ESR to be asserted at any given time. This preserves the association between the ESR and the corresponding address and attribute registers, which are loaded on each occurrence of an properly-enabled ECC event. If there is a pending ECC interrupt and another properly-enabled ECC event occurs, the ECSM hardware automatically handles the ESR reporting, clearing the previous data and loading the new state and thus guaranteeing that only a single flag is asserted.

To maintain the coherent software view of the reported event, the following sequence in the ECSM error interrupt service routine is suggested:

1. Read the ESR and save it.
2. Read and save all the address and attribute reporting registers.
3. Re-read the ESR and verify the current contents matches the original contents. If the two values are different, go back to step 1 and repeat.
4. When the values are identical, write a 1 to the asserted ESR flag to negate the interrupt request.

See Figure 19-3 and Table 19-4 for the ECC Status Register definition.

Register address: ECSM Base + 0x47

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | R1BC | F1BC | 0 | 0 | RNCE | FNCE |
| W | | | w1c | w1c | | | w1c | w1c |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 19-3. ECC Status (ESR) Register**

**Table 19-4. ECC Status (ESR) Field Descriptions**

| Field Description | |
|---|---|
| R1BC | RAM 1-bit Correction<br>0 = No reportable single-bit RAM correction has been detected.<br>1 = A reportable single-bit RAM correction has been detected.<br><br>This bit can only be set if ECR[ER1BR] is asserted. The occurrence of a properly-enabled single-bit RAM correction generates a ECSM ECC interrupt request. The address, attributes and data are also captured in the REAR, RESR, REMR, REAT and REDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect. |
| F1BC | Flash 1-bit Correction<br>0 = No reportable single-bit flash correction has been detected.<br>1 = A reportable single-bit flash correction has been detected.<br><br>This bit can only be set if ECR[EF1BR] is asserted. The occurrence of a properly-enabled single-bit flash correction generates a ECSM ECC interrupt request. The address, attributes and data are also captured in the FEAR, FEMR, FEAT and FEDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect. |
| RNCE | RAM Non-Correctable Error<br>0 = No reportable non-correctable RAM error has been detected.<br>1 = A reportable non-correctable RAM error has been detected.<br><br>The occurrence of a properly-enabled non-correctable RAM error generates a ECSM ECC interrupt request. The faulting address, attributes and data are also captured in the REAR, RESR, REMR, REAT and REDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect. |
| FNCE | Flash Non-Correctable Error<br>0 = No reportable non-correctable flash error has been detected.<br>1 = A reportable non-correctable flash error has been detected.<br><br>The occurrence of a properly-enabled non-correctable flash error generates a ECSM ECC interrupt request. The faulting address, attributes and data are also captured in the FEAR, FEMR, FEAT and FEDR registers. To clear this interrupt flag, write a 1 to this bit. Writing a 0 has no effect. |

In the event that multiple status flags are signaled simultaneously, ECSM records the event with the R1BC as highest priority, then F1BC, then RNCE, and finally FNCE.

### 19.4.2.5 ECC Error Generation Register (EEGR)

The ECC Error Generation Register is a 16-bit control register used to force the generation of single- and double-bit data inversions in the memories with ECC, most notably the RAM. This capability is provided for two purposes:

- It provides a software-controlled mechanism for “injecting” errors into the memories during data writes to verify the integrity of the ECC logic.
- It provides a mechanism to allow testing of the software service routines associated with memory error logging.

It should be noted that while the EEGR is associated with the RAM, similar capabilities exist for the flash, i.e., the ability to program the non-volatile memory with single- or double-bit errors is supported for the same two reasons previously identified.

For both types of memories (RAM and flash), the intent is to generate errors during data write cycles, such that subsequent reads of the corrupted address locations generate ECC events, either single-bit corrections or double-bit noncorrectable errors that are terminated with an error response.

The enabling of these error generation modes requires the same SoC-configurable input enable signal (as that used to enable single-bit correction reporting) be asserted.

See Figure 19-4 and Table 19-5 for the ECC Configuration Register definition.

Register address: ECSM Base + 0x4a

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RWBIT | 0 | FRC1BI | FR11BI | 0 | 0 | FRCNCI | FR1NCI | 0 | ERRBIT | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 19-4. ECC Error Generation (EEGR) Register**

**Table 19-5. EEGR field descriptions**

| Field Description | |
|---|---|
| RWBIT | Redundant writable bit<br>This bit has no function, but may be read and written. |
| FRC1BI | Force RAM Continuous 1-Bit Data Inversions<br>0 = No RAM continuous 1-bit data inversions are generated.<br>1 = 1-bit data inversions in the RAM are continuously generated.<br><br>The assertion of this bit forces the RAM controller to create 1-bit data inversions, as defined by the bit position specified in ERRBIT[0:6], continuously on every write operation.<br><br>The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT is inverted to introduce a 1-bit ECC event in the RAM.<br><br>After this bit has been enabled to generate another continuous 1-bit data inversion, it must be cleared before being set again to properly re-enable the error generation logic.<br><br>This bit can only be set if the same SoC configurable input enable signal (as that used to enable single-bit correction reporting) is asserted.<br><br>**Note:** The only allowable values for the 4 control bit enables {FR11BI, FRC1BI, FRCNCI, FR1NCI} are {0,0,0,0}, {1,0,0,0}, {0,1,0,0}, {0,0,1,0} and {0,0,0,1}. All other values result in undefined behavior. |

**Table 19-5. EEGR field descriptions (continued)**

| Field De | scription |
|---|---|
| FR11BI | Force RAM One 1-bit Data Inversion<br>0 = No RAM single 1-bit data inversion is generated.<br>1 = One 1-bit data inversion in the RAM is generated.<br><br>The assertion of this bit forces the RAM controller to create one 1-bit data inversion, as defined by the bit position specified in ERRBIT[0:6], on the first write operation after this bit is set.<br><br>The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT is inverted to introduce a 1-bit ECC event in the RAM.<br><br>After this bit has been enabled to generate a single 1-bit data inversion, it must be cleared before being set again to properly re-enable the error generation logic.<br><br>This bit can only be set if the same SoC configurable input enable signal (as that used to enable single-bit correction reporting) is asserted.<br><br>**Note:** The only allowable values for the 4 control bit enables {FR11BI, FRC1BI, FRCNCI, FR1NCI} are {0,0,0,0}, {1,0,0,0}, {0,1,0,0}, {0,0,1,0} and {0,0,0,1}. All other values result in undefined behavior. |
| FRCNCI | Force RAM Continuous Noncorrectable Data Inversions<br>0 = No RAM continuous 2-bit data inversions are generated.<br>1 = 2-bit data inversions in the RAM are continuously generated.<br><br>The assertion of this bit forces the RAM controller to create 2-bit data inversions, as defined by the bit position specified in ERRBIT[0:6] and the overall odd parity bit, continuously on every write operation.<br><br>After this bit has been enabled to generate another continuous noncorrectable data inversion, it must be cleared before being set again to properly re-enable the error generation logic.<br><br>The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT and the overall odd parity bit are inverted to introduce a 2-bit ECC error in the RAM.<br><br>**Note:** The only allowable values for the 4 control bit enables {FR11BI, FRC1BI, FRCNCI, FR1NCI} are {0,0,0,0}, {1,0,0,0}, {0,1,0,0}, {0,0,1,0} and {0,0,0,1}. All other values result in undefined behavior. |
| FR1NCI | Force RAM One Noncorrectable Data Inversions<br>0 = No RAM single 2-bit data inversions are generated.<br>1 = One 2-bit data inversion in the RAM is generated.<br><br>The assertion of this bit forces the RAM controller to create one 2-bit data inversion, as defined by the bit position specified in ERRBIT[0:6] and the overall odd parity bit, on the first write operation after this bit is set.<br><br>The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT and the overall odd parity bit are inverted to introduce a 2-bit ECC error in the RAM.<br><br>After this bit has been enabled to generate a single 2-bit error, it must be cleared before being set again to properly re-enable the error generation logic.<br><br>**Note:** The only allowable values for the 4 control bit enables {FR11BI, FRC1BI, FRCNCI, FR1NCI} are {0,0,0,0}, {1,0,0,0}, {0,1,0,0}, {0,0,1,0} and {0,0,0,1}. All other values result in undefined behavior. |

**Table 19-5. EEGR field descriptions (continued)**

| Field De | scription |
|---|---|
| ERRBIT | Error Bit Position<br>The vector defines the bit position which is complemented to create the data inversion on the write operation. For the creation of 2-bit data inversions, the bit specified by this field plus the odd parity bit of the ECC code are inverted.<br><br>The RAM controller follows a vector bit ordering scheme where LSB=0. Errors in the ECC syndrome bits can be generated by setting this field to a value greater than the RAM width. For example, consider a 32-bit RAM implementation.<br>The 32-bit ECC approach requires 7 code bits for a 32-bit word. For PRAM data width of 32 bits, the actual SRAM (32b data + 7b for ECC) = 39 bits. The following association between the ERRBIT field and the corrupted memory bit is defined:<br><br>if ERRBIT = 0, then RAM[0] of the odd bank is inverted<br>if ERRBIT = 1, then RAM[1] of the odd bank is inverted<br>...<br>if ERRBIT = 31, then RAM[31] of the odd bank is inverted<br>if ERRBIT = 64, then ECC Parity[0] of the odd bank is inverted<br>if ERRBIT = 65, then ECC Parity[1] of the odd bank is inverted<br>...<br>if ERRBIT = 70, then ECC Parity[6] of the odd bank is inverted<br><br>For ERRBIT values of 32 to 63 and greater than 70, no bit position is inverted. |

If an attempt to force a non-correctable inversion (by asserting EEGR[FRCNCI] or EEGR[FRC1NCI]) and EEGR[ERRBIT] equals 64, then no data inversion will be generated.

### 19.4.2.6 Flash ECC Address Register (FEAR)

The FEAR is a 32-bit register for capturing the address of the last, properly-enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash causes the address, attributes and data associated with the access to be loaded into the FEAR, FEMR, FEAT and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 19-5 and Table 19-6 for the Flash ECC Address Register definition.

Register address: ECSM Base + 0x50

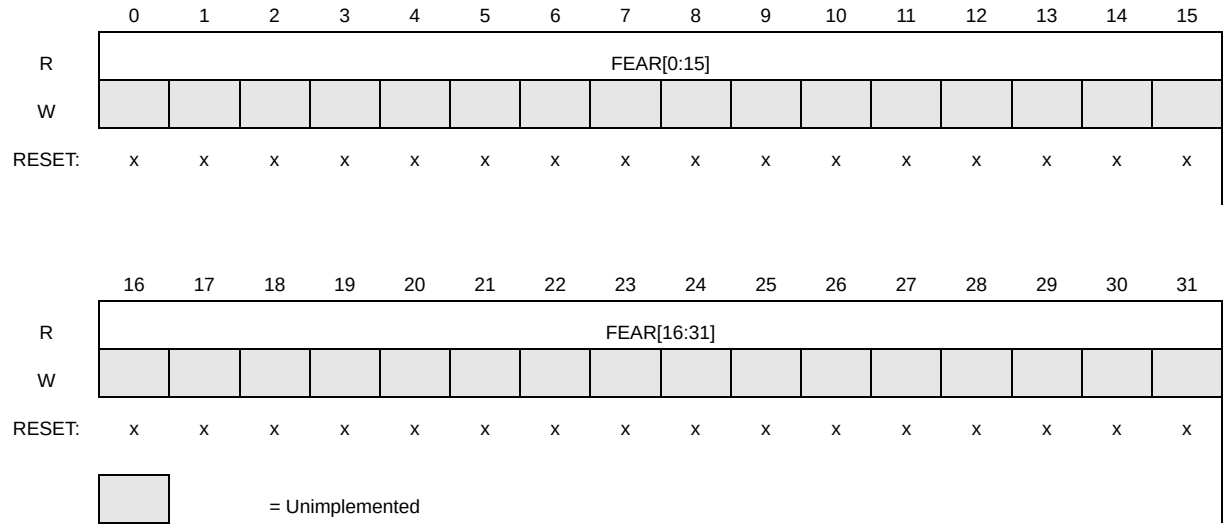



**Figure 19-5. Flash ECC Address (FEAR) Register**

**Table 19-6. Flash ECC Address (FEAR) Field Descriptions**

| Name De | scription |
|---|---|
| 0-31<br>FEAR[0:31] | Flash ECC Address Register<br>This 32-bit register contains the faulting access address of the last, properly-enabled flash ECC event. |

### 19.4.2.7 Flash ECC Master Number Register (FEMR)

The FEMR is a 4-bit register for capturing the AXBS bus master number of the last, properly-enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash causes the address, attributes and data associated with the access to be loaded into the FEAR, FEMR, FEAT and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 19-6 and Table 19-7 for the Flash ECC Master Number Register definition.

Register address: ECSM Base + 0x56

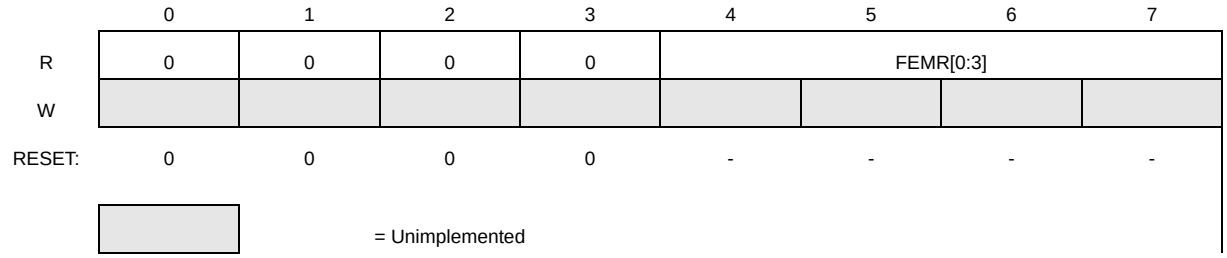



**Figure 19-6. Flash ECC Master Number (FEMR) Register**

**Table 19-7. Flash ECC Master Number (FEMR) Field Descriptions**

| Name D | escription |
|---|---|
| 4-7<br>FEMR[0:3] | Flash ECC Master Number Register<br>This 4-bit register contains the AXBS bus master number of the faulting access of the last, properly-enabled flash ECC event. |

### 19.4.2.8 Flash ECC Attributes (FEAT) register

The FEAT is an 8-bit register for capturing the AXBS bus master attributes of the last, properly-enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash causes the address, attributes and data associated with the access to be loaded into the FEAR, FEMR, FEAT and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 19-7 and Table 19-8 for the Flash ECC Attributes Register definition.

Register address: ECSM Base + 0x57

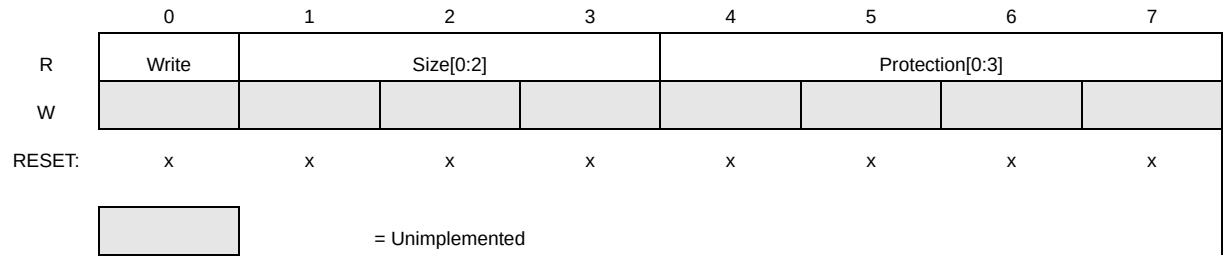



**Figure 19-7. Flash ECC Attributes (FEAT) Register**

**Table 19-8. Flash ECC Attributes (FEAT) Field Descriptions**

| Name De | scription |
|---|---|
| 0<br>Write | AMBA-AHB HWRITE<br>0 = AMBA-AHB read access<br>1 = AMBA-AHB write access |
| 1-3<br>Size[0:2] | AMBA-AHB HSIZE[0:2]<br>0b000 = 8-bit AMBA-AHB access<br>0b001 = 16-bit AMBA-AHB access<br>0b010 = 32-bit AMBA-AHB access<br>0b1xx = Reserved |
| 4-7<br>Protection[0:3] | AMBA-AHB HPROT[0:3]<br>Protection[3]: Cacheable 0 = Non-cacheable,1 = Cacheable<br>Protection[2]: Bufferable0 = Non-bufferable,1 = Bufferable<br>Protection[1]: Mode 0 = User mode, 1 = Supervisor mode<br>Protection[0]: Type 0 = I-Fetch, 1 = Data |

### 19.4.2.9 Flash ECC Data Register (FEDR)

The FEDR is a 32-bit register for capturing the data associated with the last, properly-enabled ECC event in the flash memory. Depending on the state of the ECC Configuration Register, an ECC event in the flash causes the address, attributes and data associated with the access to be loaded into the FEAR, FEMR,

FEAT and FEDR registers, and the appropriate flag (F1BC or FNCE) in the ECC Status Register to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 19-8 and Table 19-9 for the Flash ECC Data Register definition.

Register address: ECSM Base +0x5C

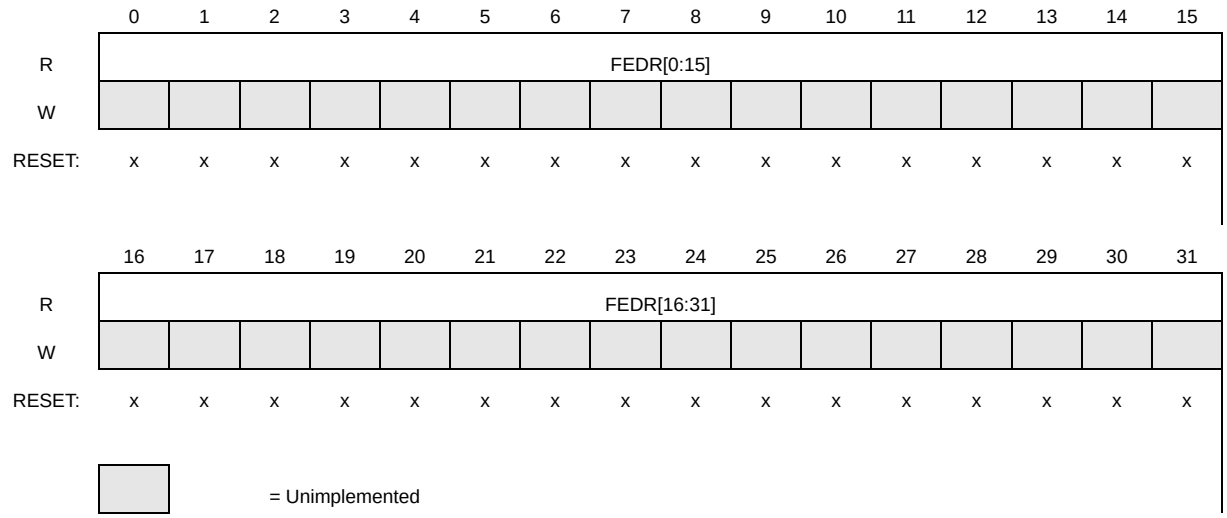



**Figure 19-8. Flash ECC Data (FEDR) Register**

**Table 19-9. Flash ECC Data (FEDR) Field Descriptions**

| Name Des | cription |
|---|---|
| 0-31<br>FEDR[0:31] | Flash ECC Data Register<br>This 32-bit register contains the data associated with the faulting access of the last, properly-enabled flash ECC event. The register contains the data value taken directly from the data bus. |

## 19.4.2.10 RAM ECC Address Register (REAR)

The REAR is a 32-bit register for capturing the address of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the REAR, RESR, REMR, REAT and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 19-9 and Table 19-10 for the RAM ECC Address Register definition.

Register address: ECSM Base + 0x60

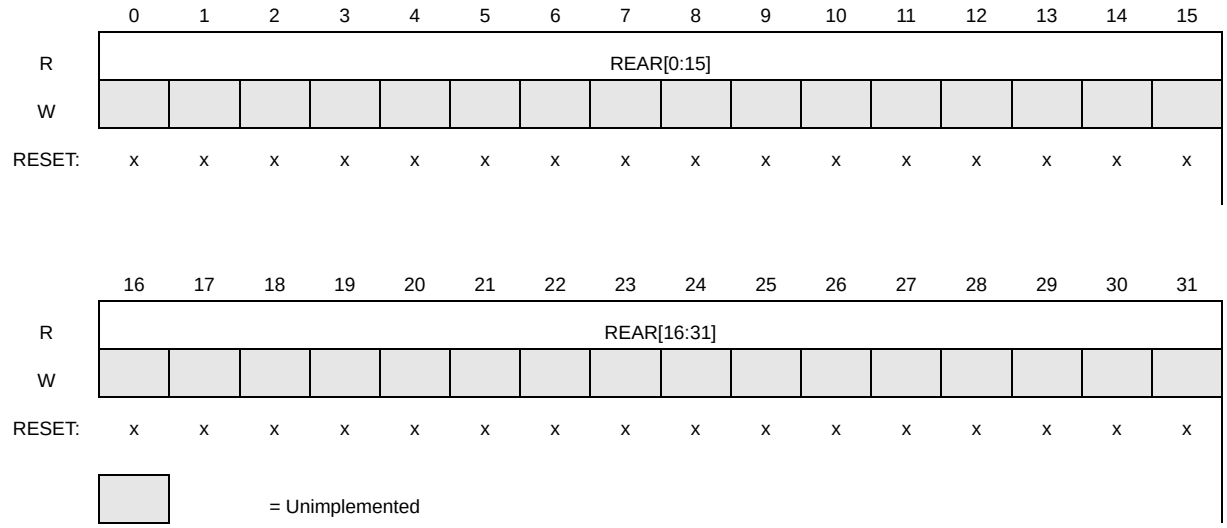



**Figure 19-9. RAM ECC Address (REAR) Register**

**Table 19-10. RAM ECC Address (REAR) Field Descriptions**

| Name Desc | ription |
|---|---|
| 0-31<br>REAR[0:31] | RAM ECC Address Register<br>This 32-bit register contains the faulting access address of the last, properly-enabled RAM ECC event. |

## 19.4.2.11 RAM ECC Syndrome Register (RESR)

The RESR is an 8-bit register for capturing the error syndrome of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the REAR, RESR, REMR, REAT and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 19-10 and Table 19-11 for the RAM ECC Syndrome Register definition.

Register address: ECSM Base + 0x65

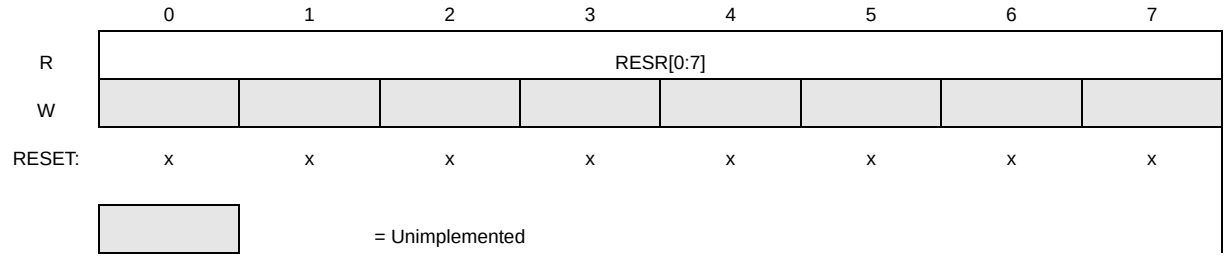



**Figure 19-10. RAM ECC Syndrome (RESR) Register**

**Table 19-11. RAM ECC Syndrome (RESR) Field Descriptions**

| Name Desc | ription |
|---|---|
| 0-7<br>RESR[0:7] | RAM ECC Syndrome Register<br>This 8-bit syndrome field includes 6 bits of Hamming decoded parity plus an odd-parity bit for the entire 39-bit (32-bit data + 7 ECC) code word. The upper 7 bits of the syndrome specify the exact bit position in error for single-bit correctable codewords, and the combination of a non-zero 7-bit syndrome plus overall incorrect parity bit signal a multi-bit, non-correctable error.<br><br>For correctable single-bit errors, the mapping shown in Table 19-11 associates the upper 7 bits of the syndrome with the data bit in error. |

**Note:** Table 19-11 associates the upper 7 bits of the ECC syndrome with the exact data bit in error for single-bit correctable codewords. This table follows the bit vectoring notation where the LSB=0. Note that the syndrome value of 0x01 implies no error condition but this value is not readable when the PRESR is read for the no error case.

**Table 19-12. RAM Syndrome Mapping for Single-Bit Correctable Errors**

| RESR[0:7] | Data Bit in Error |
|---|---|
| 0x00 | ECC ODD[0] |
| 0x01 | No Error |
| 0x02 | ECC ODD[1] |
| 0x04 | ECC ODD[2] |
| 0x06 | DATA ODD BANK[31] |
| 0x08 | ECC ODD[3] |
| 0x0a | DATA ODD BANK[30] |
| 0x0c | DATA ODD BANK[29] |
| 0x0e | DATA ODD BANK[28] |
| 0x10 | ECC ODD[4] |
| 0x12 | DATA ODD BANK[27] |
| 0x14 | DATA ODD BANK[26] |
| 0x16 | DATA ODD BANK[25] |
| 0x18 | DATA ODD BANK[24] |
| 0x1a | DATA ODD BANK[23] |
| 0x1c | DATA ODD BANK[22] |
| 0x50 | DATA ODD BANK[21] |
| 0x20 | ECC ODD[5] |
| 0x22 | DATA ODD BANK[20] |
| 0x24 | DATA ODD BANK[19] |
| 0x26 | DATA ODD BANK[18] |
| 0x28 | DATA ODD BANK[17] |
| 0x2a | DATA ODD BANK[16 |
| 0x2c | DATA ODD BANK[15] |
| 0x58 | DATA ODD BANK[14] |
| 0x30 | DATA ODD BANK[13] |
| 0x32 | DATA ODD BANK[12] |
| 0x34 | DATA ODD BANK[11] |
| 0x64 | DATA ODD BANK[10] |

**Table 19-12. RAM Syndrome Mapping for Single-Bit Correctable Errors (continued)**

| RESR[0:7] | Data Bit in Error |
|---|---|
| 0x38 | DATA ODD BANK[9] |
| 0x62 | DATA ODD BANK[8] |
| 0x70 | DATA ODD BANK[7] |
| 0x60 | DATA ODD BANK[6] |
| 0x40 | ECC ODD[6] |
| 0x42 | DATA ODD BANK[5] |
| 0x44 | DATA ODD BANK[4] |
| 0x46 | DATA ODD BANK[3] |
| 0x48 | DATA ODD BANK[2] |
| 0x4a | DATA ODD BANK[1] |
| 0x4c | DATA ODD BANK[0] |
| 0x03,0x05.....0x4d | Multiple bit error |
| > 0x4d | Multiple bit error |

### 19.4.2.12 RAM ECC Master Number Register (REMR)

The REMR is a 4-bit register for capturing the AXBS bus master number of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the REAR, RESR, REMR, REAT and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 19-11 and Table 19-13 for the RAM ECC Master Number Register definition.

Register address: ECSM Base + 0x66

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | REMR[0:3] | | | |
| W | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | x | x | x | x |

= Unimplemented

**Figure 19-11. RAM ECC Master Number (REMR) Register**

**Table 19-13. RAM ECC Master Number (REMR) Field Descriptions**

| Name D | escription |
|---|---|
| 4-7<br>REMR[0:3] | RAM ECC Master Number Register<br>This 4-bit register contains the AXBS bus master number of the faulting access of the last, properly-enabled RAM ECC event. |

### 19.4.2.13 RAM ECC Attributes (REAT) register

The REAT is an 8-bit register for capturing the AXBS bus master attributes of the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the REAR, RESR, REMR, REAT and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 19-12 and Table 19-14 for the RAM ECC Attributes Register definition.

Register address: ECSM Base + 0x67

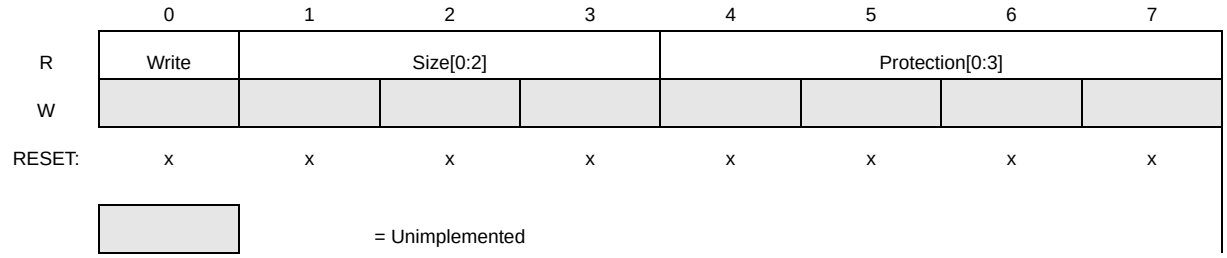



**Figure 19-12. RAM ECC Attributes (REAT) Register**

**Table 19-14. RAM ECC Attributes (REAT) Field Descriptions**

| Name Desc | ription |
|---|---|
| 0<br>Write | AMBA-AHB HWRITE<br>0 = AMBA-AHB read access<br>1 = AMBA-AHB write access |
| 1-3<br>Size[0:2] | AMBA-AHB HSIZE[0:2]<br>0b000 = 8-bit AMBA-AHB access<br>0b001 = 16-bit AMBA-AHB access<br>0b010 = 32-bit AMBA-AHB access<br>0b1xx = Reserved |
| 4-7<br>Protection[0:3] | AMBA-AHB HPROT[0:3]<br>Protection[3]: Cacheable 0 = Non-cacheable, 1 = Cacheable<br>Protection[2]: Bufferable 0 = Non-bufferable,1 = Bufferable<br>Protection[1]: Mode 0 = User mode, 1 = Supervisor mode<br>Protection[0]: Type 0 = I-Fetch, 1 = Data |

### 19.4.2.14 RAM ECC Data Register (REDR)

The REDR is a 32-bit register for capturing the data associated with the last, properly-enabled ECC event in the RAM memory. Depending on the state of the ECC Configuration Register, an ECC event in the RAM causes the address, attributes and data associated with the access to be loaded into the REAR, RESR, REMR, REAT and REDR registers, and the appropriate flag (R1BC or RNCE) in the ECC Status Register to be asserted.

The data captured on a multi-bit non-correctable ECC error is undefined.

This register can only be read from the IPS programming model; any attempted write is ignored. See Figure 19-13 and Table 19-15 for the RAM ECC Data Register definition.

Register address: ECSM Base +0x6c

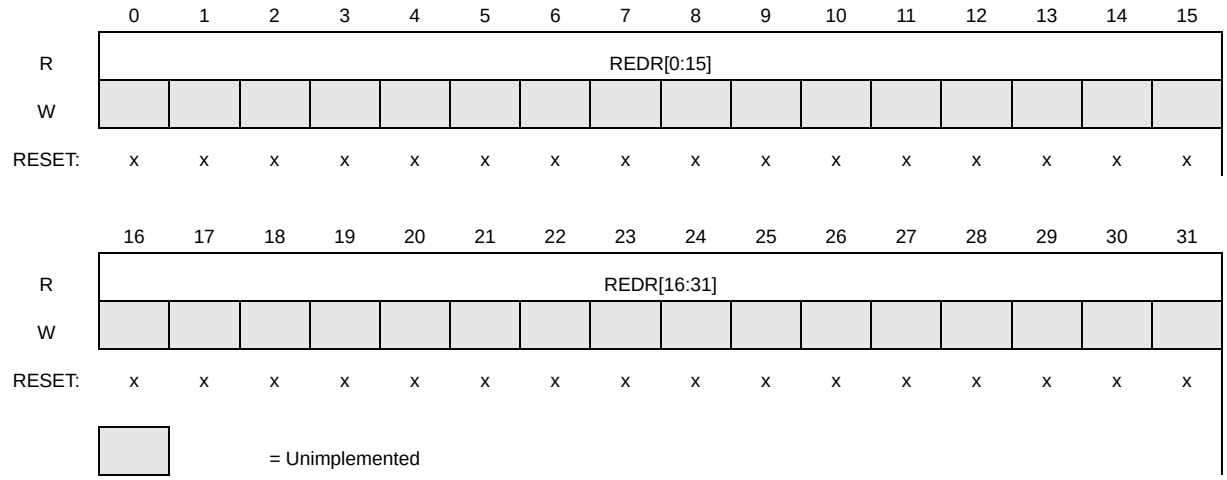



**Figure 19-13. RAM ECC Data (REDR) Register**

**Table 19-15. RAM ECC Data (REDR) Field Descriptions**

| Name Des | cription |
|---|---|
| 0-31<br>REDR[0:31] | RAM ECC Data Register<br>This 32-bit register contains the data associated with the faulting access of the last, properly-enabled RAM ECC event. The register contains the data value taken directly from the data bus. |

## 19.4.3 High Priority Enables

e200 processors can be configured to support critical and/or external interrupts. Furthermore, each processor can be configured to employ priority elevation on critical and/or external interrupt events. Critical interrupts come from outside the platform, and are routed directly to the processor's critical interrupt input. External interrupts are routed through the interrupt controller. In addition to the interrupt notification signals, various processor-specific configuration flags from the processor's Machine Check Register (MCR[ee,ce]) and the Hardware Implementation Register (HID1) are sent to the ECSM to determine when interrupt servicing is enabled and when high-priority elevation should be enabled. If the corresponding processor is configured to allow high-priority elevation on critical interrupt events, the ECSM generates the high-priority signal upon critical interrupt detection and holds it active throughout the duration of interrupt servicing. If the corresponding processor is configured to allow high-priority elevation on external interrupt events, the ECSM generates the high-priority signal upon external interrupt detection and holds it active throughout the duration of interrupt servicing. During interrupt servicing the processor status output, p_stat, is monitored for indication of a return from interrupt (rfi).

Great care needs to be taken when using the priority elevation as it can enable a master to starve the rest of the masters in the system. Please see Chapter 9, "Crossbar Switch (XBAR)," for information on priority elevation.

### 19.4.4 Supervisor mode access protection

The supervisor mode access protection logic provides hardware enforcement of supervisor mode access protection for five on-platform IPS modules: INTC, ECSM, MPU, STM, and SWT. This logic resides between the on-platform bus sourced by the AIPS bus controller and the individual slave modules. It monitors the bus access type (supervisor or user) and if a user access is attempted, the transfer is terminated with an error and inhibited from reaching the slave module. Identical logic is replicated for each of the five, targeted slave modules.

This page is intentionally left blank.

# Chapter 20
# FlexCAN

## 20.1 Introduction

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification. A general block diagram is shown in Figure 20-1, which describes the main sub-blocks implemented in the FlexCAN module, including two embedded memories, one for storing Message Buffers (MB) and another one for storing Rx Individual Mask Registers. Support for up to 64 Message Buffers is provided (see the Device User Guide for the actual size implemented on the MCU). The functions of the sub-modules are described in subsequent sections.

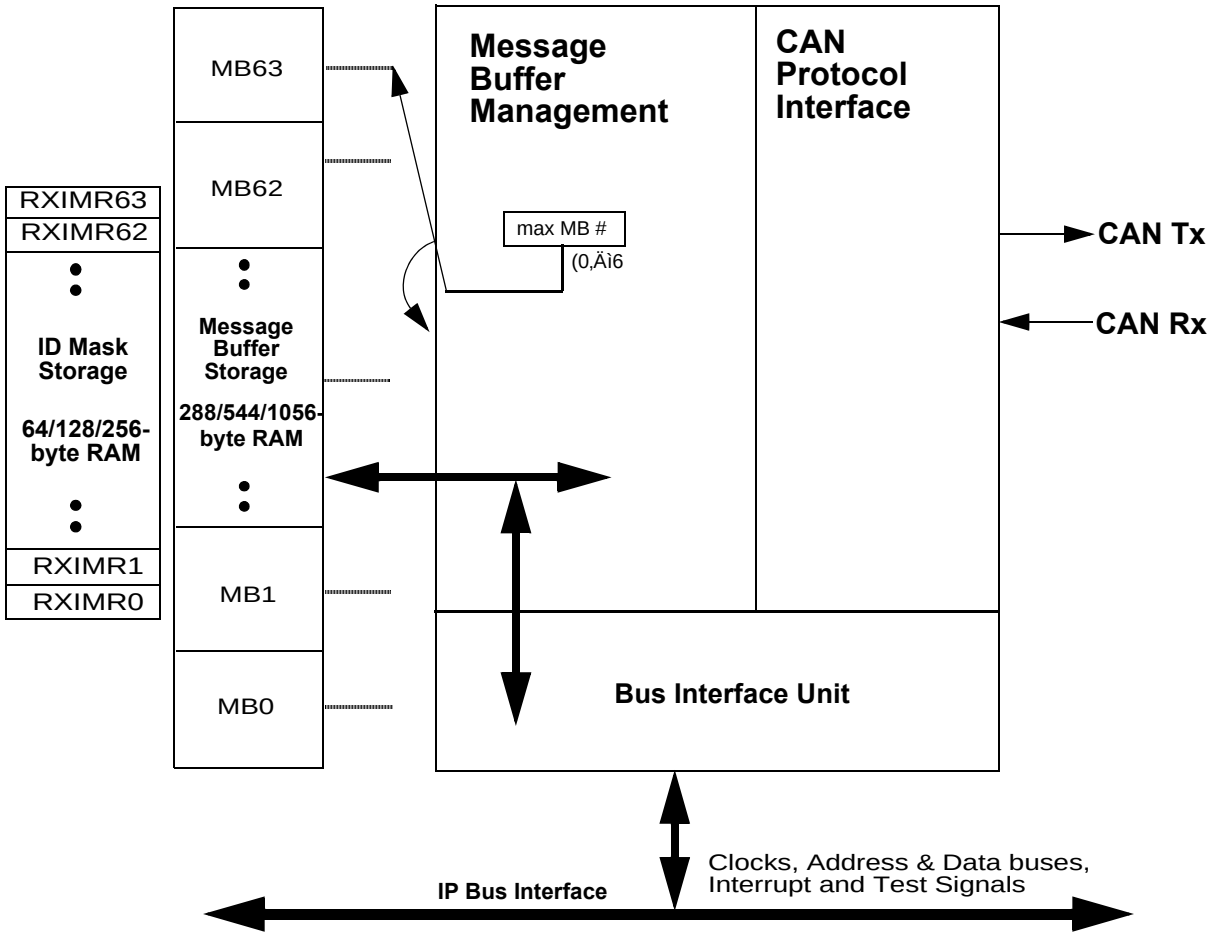



**Figure 20-1. FlexCAN block diagram**

### 20.1.1 Overview

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. The FlexCAN module is a full implementation of the

CAN protocol specification, Version 2.0 B [Ref. 1], which supports both standard and extended message frames. A flexible number of Message Buffers (16, 32 or 64) is also supported. The Message Buffers are stored in an embedded RAM dedicated to the FlexCAN module. Please refer to the device-specific information section for the actual number of Message Buffers configured in the MCU.

The CAN Protocol Interface (CPI) sub-module manages the serial communication on the CAN bus, requesting RAM access for receiving and transmitting message frames, validating received messages and performing error handling. The Message Buffer Management (MBM) sub-module handles Message Buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms. The Bus Interface Unit (BIU) sub-module controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs and test signals are accessed through the Bus Interface Unit.

### 20.1.2 FlexCAN module features

The FlexCAN module includes these distinctive features:

- Full Implementation of the CAN protocol specification, Version 2.0B
  - Standard data and remote frames
  - Extended data and remote frames
  - Zero to eight bytes data length
  - Programmable bit rate up to 1 Mb/sec
  - Content-related addressing
- Flexible Message Buffers (up to 64) of zero to eight bytes data length
- Each MB configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask Registers per Message Buffer
- Includes up to 1024 bytes (64 MBs) of RAM used for MB storage
- Includes up to 256 bytes (64 MBs) of RAM used for individual Rx Mask Registers
- Full featured Rx FIFO with storage capacity for 6 frames and internal pointer handling
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 8 extended, 16 standard or 32 partial (8 bits) IDs, with individual masking capability
- Selectable backwards compatibility with previous FlexCAN version
- Programmable clock source to the CAN Protocol Interface, either bus clock or crystal oscillator
- Unused MB and Rx Mask Register space can be used as general purpose RAM space
- Listen only mode capability
- Programmable loop-back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number or highest priority
- Time Stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)

- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes

### 20.1.3 Modes of operation

The FlexCAN module has four functional modes: Normal Mode (User and Supervisor), Freeze Mode, Listen-Only Mode and Loop-Back Mode. There is also a low-power mode, Disable Mode.

- Normal Mode (User or Supervisor):

  In Normal Mode, the module operates receiving and/or transmitting message frames, errors are handled normally and all the CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.
- Freeze Mode:

  It is enabled when the FRZ bit in the MCR Register is asserted. If enabled, Freeze Mode is entered when the HALT bit in MCR is set or when Debug Mode is requested at MCU level. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Section 20.5.9.1, “Freeze mode“ for more information.
- Listen-Only Mode:

  The module enters this mode when the LOM bit in the Control Register is asserted. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode [Ref. 1]. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.
- Loop-Back Mode:

  The module enters this mode when the LPB bit in the Control Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic ‘1’). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.
- Module Disable Mode:

  This low power mode is entered when the MDIS bit in the MCR Register is asserted. On setting the MDIS bit, the module does not shut down the clocks, it only requests the shut down to the integration logic (CCB) that can or cannot disable the clocks. The clocks to the message buffer modules can be kept running also with the module disabled to keep the the memory available for general use. . Exit from this mode is done by negating the CAN_MCR[MDIS] bit. See Section 20.5.9.2, “Module Disable mode“ for more information.

## 20.2 Device-specific information

This device implements FlexCAN 0, FlexCAN 1, and FlexCAN 2. The "x" appended to signal names signifies the FlexCAN module to which the signal applies. Thus CANTX_0 is the transmit signal that

applies to FlexCAN 0, CANTX_1 is the transmit signal that applies to FlexCAN 1, and so on. All FlexCAN modules feature 64 Message Buffers.

## 20.3 External signal description

### 20.3.1 Overview

The FlexCAN module has two I/O signals connected to the external MCU pins. These signals are summarized in Table 20-1 and described in more detail in the next sub-sections.

**Table 20-1. FlexCAN Signals**

| Signal Name[1] | Direction | Description |
|---|---|---|
| CAN Rx | Input | CAN Receive Pin |
| CAN Tx | Output | CAN Transmit Pin |

1 The actual MCU pins may have different names. Please consult the Device User Guide for the actual signal names.

### 20.3.2 Signal descriptions

#### 20.3.2.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level ‘0’. Recessive state is represented by logic level ‘1’.

#### 20.3.2.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level ‘0’. Recessive state is represented by logic level ‘1’.

## 20.4 Memory map and register description

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the MCU. The addresses presented here are relative to the base address.

The address space occupied by FlexCAN has 96 bytes for registers starting at the module base address, followed by MB storage space in embedded RAM starting at address 0x0060, and an extra ID Mask storage space in a separate embedded RAM starting at address 0x0880.

### 20.4.1 FlexCAN memory mapping

The complete memory map for a FlexCAN module with 64 MBs capability is shown in Table 20-2. Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor

or Unrestricted access by programming the MCR[SUPV] bit. These registers are identified as S/U in the Access column of Table 20-2.

The IFRH and IMRH registers are considered reserved space when FlexCAN is configured with 16 or 32 MBs. The Rx Global Mask (RXGMASK), Rx Buffer 14 Mask (RX14MASK) and the Rx Buffer 15 Mask (RX15MASK) registers are provided for backwards compatibility, and are not used when the BCC bit in MCR is asserted.

The address ranges 0x0060–0x047F and 0x0880–0x097F are occupied by two separate embedded memories. These two ranges are completely occupied by RAM (1056 and 256 bytes, respectively) only when FlexCAN is configured with 64 MBs. When it is configured with 16 MBs, the memory sizes are 288 and 64 bytes, so the address ranges 0x0180–0x047F and 0x08C0–0x097F are considered reserved space. When it is configured with 32 MBs, the memory sizes are 544 and 128 bytes, so the address ranges 0x0280–0x047F and 0x0900–0x097F are considered reserved space. Furthermore, if the BCC bit in MCR is negated, then the whole Rx Individual Mask Registers address range (0x0880–0x097F) is considered reserved space.

**Table 20-2. Module memory map**

| Address | Use | Access Type | Affected by hard reset | Affected by soft reset | Location |
|---|---|---|---|---|---|
| Base + 0x0000 | Module Configuration (MCR) | S | Yes | Yes | on page 781 |
| Base + 0x0004 | Control Register (CTRL) | S/U | Yes | No | on page 785 |
| Base + 0x0008 | Free Running Timer (TIMER) | S/U | Yes | Yes | on page 788 |
| Base + 0x000C | Reserved | | | | |
| Base + 0x0010 | Rx Global Mask (RXGMASK) | S/U | Yes | No | on page 789 |
| Base + 0x0014 | Rx Buffer 14 Mask (RX14MASK) | S/U | Yes | No | on page 791 |
| Base + 0x0018 | Rx Buffer 15 Mask (RX15MASK) | S/U | Yes | No | on page 791 |
| Base + 0x001C | Error Counter Register (ECR) | S/U | Yes | Yes | on page 791 |
| Base + 0x0020 | Error and Status Register | S/U | Yes | Yes | on page 793 |
| Base + 0x0024 | Interrupt Mask Register High (IMRH) | S/U | Yes | Yes | on page 795 |
| Base + 0x0028 | Interrupt Mask Register Low (IMRL) | S/U | Yes | Yes | on page 796 |
| Base + 0x002C | Interrupt Flag Register High (IFRH) | S/U | Yes | Yes | on page 797 |
| Base + 0x0030 | Interrupt Flag Register High (IFRL) | S/U | Yes | Yes | on page 798 |
| Base + 0x0034–0x007F | Reserved | | | | |
| Base + 0x0080–0x017F | Message Buffers MB0–MB15 | S/U | No | No | — |
| Base + 0x0180–0x027F | Message Buffers MB16–MB31 | S/U | No | No | — |
| Base + 0x0280–0x047F | Message Buffers MB32–MB63 | S/U | No | No | — |

**Table 20-2. Module memory map (continued)**

| Address | Use | Access Type | Affected by hard reset | Affected by soft reset | Location |
|---|---|---|---|---|---|
| Base + 0x0480-087F | Reserved | | | | |
| Base + 0x0880-0x08BF | Rx Individual Mask Registers RXIMR0-RXIMR15 | S/U | No | No | on page 799 |
| Base + 0x08C0-0x08FF | Rx Individual Mask Registers RXIMR16-RXIMR31 | S/U | No | No | on page 799 |
| Base + 0x0900-0x097F | Rx Individual Mask Registers RXIMR32-RXIMR63 | S/U | No | No | on page 799 |

The FlexCAN module stores CAN messages for transmission and reception using a Message Buffer structure. Each individual MB is formed by 16 bytes mapped on memory as described in Table 20-3. Table 20-3 shows a Standard/Extended Message Buffer (MB0) memory map, using 16 bytes total (0x80–0x8F space).

**Table 20-3. Message Buffer MB0 Memory Mapping**

| Address Offset | MB Field |
|---|---|
| 0x80 | Control and Status (C/S) |
| 0x84 | Identifier Field |
| 0x88–0x8F | Data Field 0 – Data Field 7 (1 byte each) |

## 20.4.2 Message Buffer Structure

The Message Buffer structure used by the FlexCAN module is represented in Figure 20-2. Both Extended and Standard Frames (29-bit Identifier and 11-bit Identifier, respectively) used in the CAN specification (Version 2.0 Part B) are represented.

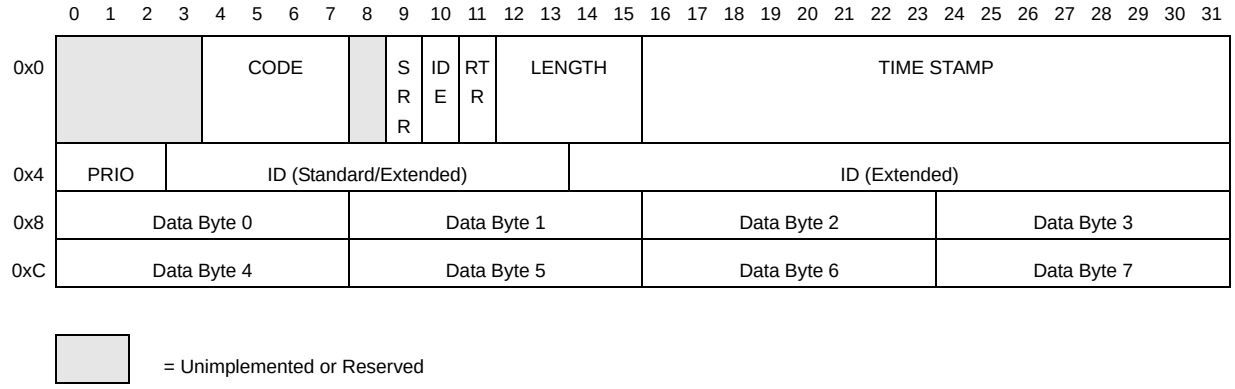



**Figure 20-2. Message Buffer structure**

**Table 20-4. Message Buffer structure field descriptions**

| Field | Description |
|---|---|
| CODE | Message Buffer Code<br>This 4-bit field can be accessed (read or write) by the CPU and by the Flexcan module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 20-5 and Table 20-6. See Section 20.5, “Functional description“ for additional information. |
| SRR | Substitute Remote Request<br>Fixed recessive bit, used only in extended format. It must be set to ‘1’ by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as arbitration loss.<br>1 = Recessive value is compulsory for transmission in Extended Format frames<br>0 = Dominant is not a valid value for transmission in Extended Format frames |
| IDE | ID Extended Bit<br>This bit identifies whether the frame format is standard or extended.<br>1 = Frame format is extended<br>0 = Frame format is standard |
| RTR | Remote Transmission Request<br>This bit is used for requesting transmissions of a data frame. If FlexCAN transmits this bit as ‘1’ (recessive) and receives it as ‘0’ (dominant), it is interpreted as arbitration loss. If this bit is transmitted as ‘0.’ (dominant), then if it is received as ‘1’ (recessive), the FlexCAN module treats it as bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission. Do not configure last Message Buffer to be RTR frame.<br>1 = Indicates the current MB has a Remote Frame to be transmitted<br>0 = Indicates the current MB has a Data Frame to be transmitted |
| LENGTH | Length of Data in Bytes<br>This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see Table 20-2). In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR=1, the Frame to be transmitted is a Remote Frame and does not include the data field, regardless of the Length field. |
| TIME STAMP | Free-Running Counter Time Stamp<br>This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus. |
| PRIO | Local priority<br>This 3-bit field is only used when LPRIO_EN bit is set in MCR and it only makes sense for Tx buffers. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Section 20.5.3, “Arbitration process“. |
| ID | Frame Identifier<br>In Standard Frame format, only the 11 most significant bits (3 to 13) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In Extended Frame format, all bits are used for frame identification in both receive and transmit cases. |
| DATA | Data Field<br>Up to eight bytes can be used for a data frame. For Rx frames, the data is stored as it is received from the CAN bus. For Tx frames, the CPU prepares the data field to be transmitted within the frame. |

**Table 20-5. Message Buffer Code for Rx Buffers**

| Rx Code BEFORE Rx New Frame | Description | Rx Code AFTER Rx New Frame | Comment |
|---|---|---|---|
| 0000 | INACTIVE: MB is not active. | – | MB does not participate in the matching process. |
| 0100 | EMPTY: MB is active and empty. | 0010 | MB participates in the matching process. When a frame is received successfully, the code is automatically updated to FULL. |
| 0010 | FULL: MB is full. | 0010 | The act of reading the C/S word followed by unlocking the MB does not make the code return to EMPTY. It remains FULL. If a new frame is written to the MB after the C/S word was read and the MB was unlocked, the code still remains FULL. |
| | | 0110 | If the MB is FULL and a new frame is overwritten to this MB before the CPU had time to read it, the code is automatically updated to OVERRUN. Refer to Section 20.5.5, "Matching Process" for details about overrun behavior. |
| 0110 | OVERRUN: a frame was overwritten into a full buffer. | 0010 | If the code indicates OVERRUN but the CPU reads the C/S word and then unlocks the MB, when a new frame is written to the MB the code returns to FULL. |
| | | 0110 | If the code already indicates OVERRUN, and yet another new frame must be written, the MB will be overwritten again, and the code will remain OVERRUN. Refer to Section 20.5.5, "Matching Process" for details about overrun behavior. |
| 0XY1[1] | BUSY: Flexcan is updating the contents of the MB. The CPU must not access the MB. | 0010 | An EMPTY buffer was written with a new frame (XY was 01). |
| | | 0110 | A FULL/OVERRUN buffer was overwritten (XY was 11). |

[1] Note that for Tx MBs (see Table 20-6), the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register.

**Table 20-6. Message Buffer Code for Tx Buffers**

| RTR | Initial Tx code | Code after successful transmission | Description |
|---|---|---|---|
| X | 1000 | – | INACTIVE: MB does not participate in the arbitration process. |
| 0 | 1100 | 1000 | Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state. |

Table 20-6. Message Buffer Code for Tx Buffers (continued)

| RTR | Initial Tx code | Code after successful transmission | Description |
|---|---|---|---|
| 1 | 1100 | 0100 | Transmit remote frame unconditionally once. After transmission, the MB automatically becomes an Rx MB with the same ID. |
| 0 | 1010 | 1010 | Transmit a data frame whenever a remote request frame with the same ID is received. This MB participates simultaneously in both the matching and arbitration processes. The matching process compares the ID of the incoming remote request frame with the ID of the MB. If a match occurs this MB is allowed to participate in the current arbitration process and the Code field is automatically updated to '1110' to allow the MB to participate in future arbitration runs. When the frame is eventually transmitted successfully, the Code automatically returns to '1010' to restart the process again. |
| 0 | 1110 | 1010 | This is an intermediate code that is automatically written to the MB by the MBM as a result of match to a remote request frame. The data frame will be transmitted unconditionally once and then the code will automatically return to '1010'. The CPU can also write this code with the same effect. |

### 20.4.3 Rx FIFO structure

When the FEN bit is set in the MCR, the memory area from 0x80 to 0xFF (which is normally occupied by MBs 0 to 7) is used by the reception FIFO engine. Figure 20-3 shows the Rx FIFO data structure. The region 0x0-0xC contains an MB structure which is the port through which the CPU reads data from the FIFO (the oldest frame received and not read yet). The region 0x10-0xDF is reserved for internal use of the FIFO engine. The region 0xE0-0xFF contains an 8-entry ID table that specifies filtering criteria for accepting frames into the FIFO. Figure 20-4 shows the three different formats that the elements of the ID table can assume, depending on the IDAM field of the MCR. Note that all elements of the table must have the same format. See Section 20.5.7, "Rx FIFO" for more information.

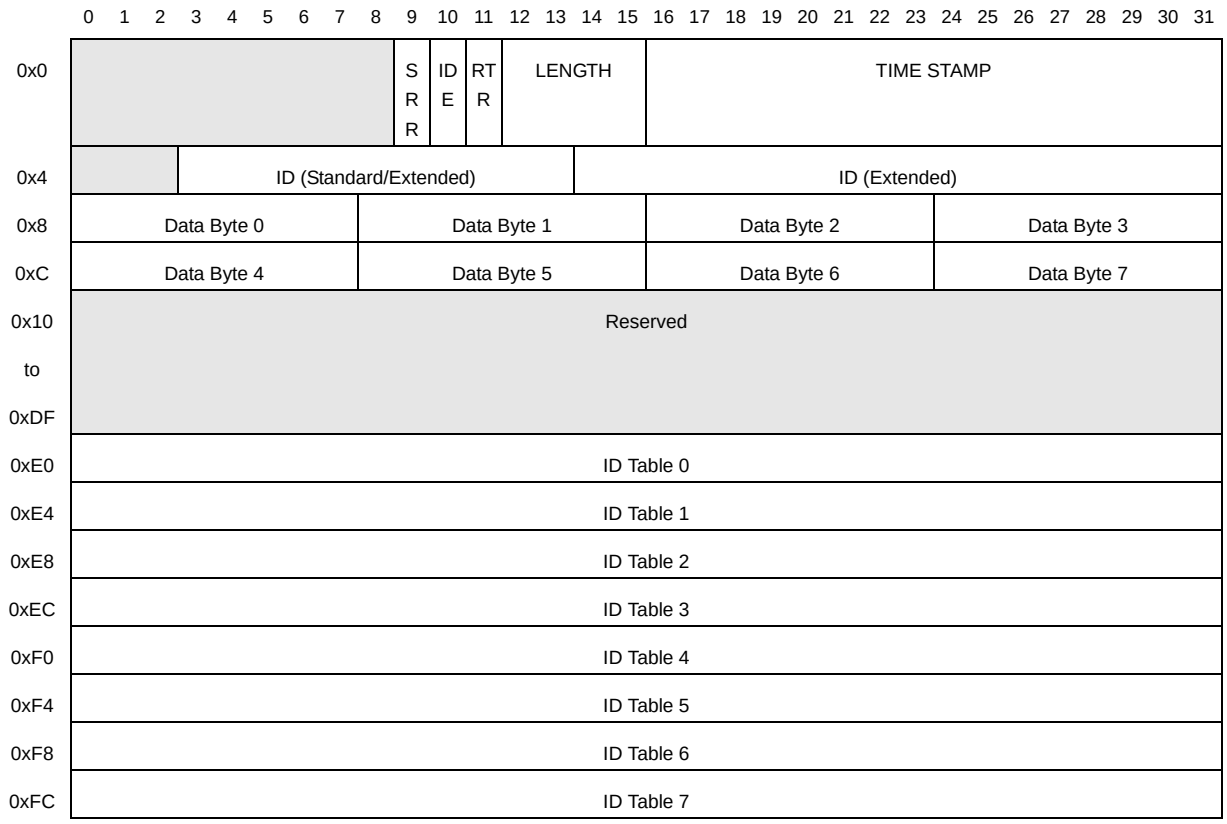



**Figure 20-3. Rx FIFO Structure**

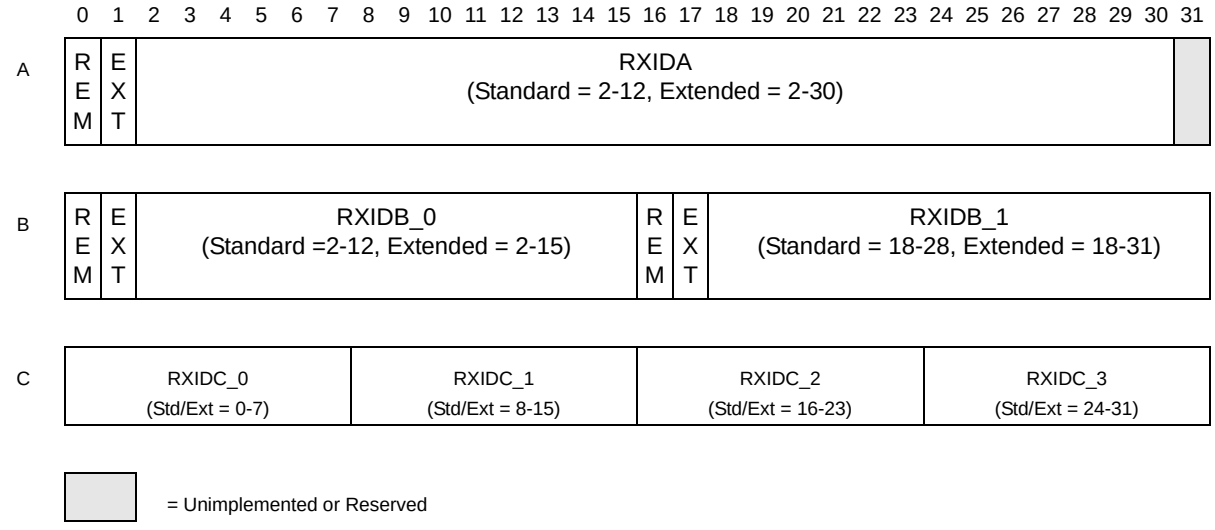



**Figure 20-4. ID Table 0 - 7**

Table 20-7. Rx FIFO structure field descriptions

| Field | Description |
|---|---|
| REM | Remote Frame<br>This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID.<br>1 = Remote Frames can be accepted and data frames are rejected<br>0 = Remote Frames are rejected and data frames can be accepted |
| EXT | Extended Frame<br>Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.<br>1 = Extended frames can be accepted and standard frames are rejected<br>0 = Extended frames are rejected and standard frames can be accepted |
| RXIDA | Rx Frame Identifier (Format A)<br>Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (2 to 12)are used for frame identification. In the extended frame format, all bits are used. |
| RXIDB_0,<br>RXIDB_1 | Rx Frame Identifier (Format B)<br>Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (2 to 12)are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID. |
| RXIDC_0,<br>RXIDC_1,<br>RXIDC_2,<br>RXIDC_3 | Rx Frame Identifier (Format C)<br>Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID. |

## 20.4.4 Register descriptions

The FlexCAN registers are described in this section in ascending address order.

### 20.4.4.1 Module Configuration Register (MCR)

This register defines global system configurations, such as the module operation mode (e.g., low power) and maximum message buffer configuration. Most of the fields in this register can be accessed at any time, except the MAXMB field, which should only be changed while the module is in Freeze Mode.

Base + 0x0000

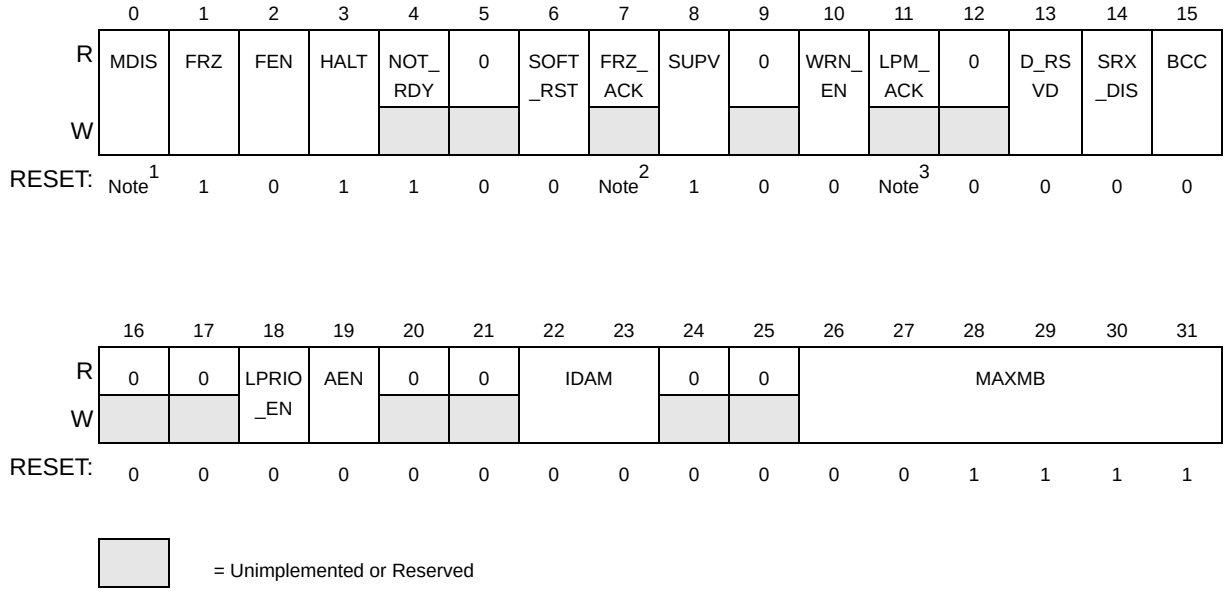



**Figure 20-5. Module Configuration Register (MCR)**

1 Reset value of this bit is different on various platforms. Consult the specific MCU documentation to determine its value.

2 Different on various platforms, but it is always the opposite of the MDIS reset value.

3 Different on various platforms, but it is always the same as the MDIS reset value.

**Table 20-8. MCR field descriptions**

| Field | Description |
|---|---|
| MDIS | Module Disable<br>On setting the MDIS bit, the module does not shut down the clocks, it only requests the shut down to the integration logic (CCB) that can or cannot disable the clocks. The clocks to the message buffer modules can be kept running also with the module disabled, to keep the the memory available for general use. See Section 20.5.9.2, “Module Disable mode“ for more information.<br>1 = Disable the FlexCAN module<br>0 = Enable the FlexCAN module |
| FRZ | Freeze Enable<br>The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR Register is set or when Debug Mode is requested at MCU level. When FRZ is asserted, FlexCAN is enabled to enter Freeze Mode. Negation of this bit field causes FlexCAN to exit from Freeze Mode.<br>1 = Enabled to enter Freeze Mode<br>0 = Not enabled to enter Freeze Mode |
| FEN | FIFO Enable<br>This bit controls whether the FIFO feature is enabled or not. When FEN is set, MBs 0 to 7 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xFF) is used by the FIFO engine. See Section 20.4.3, “Rx FIFO structure“ and Section 20.5.7, “Rx FIFO“ for more information.<br>1 = FIFO enabled<br>0 = FIFO not enabled |

**Table 20-8. MCR field descriptions (continued)**

| Field | Description |
|---|---|
| HALT | Halt FlexCAN<br>Assertion of this bit puts the FlexCAN module into Freeze Mode. The CPU should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FlexCAN before this bit is cleared. While in Freeze Mode, the CPU has write access to the Error Counter Register, that is otherwise read-only. Freeze Mode can not be entered while FlexCAN is in any of the low power modes. See Section 20.5.9.1, “Freeze mode“ for more information.<br>1 = Enters Freeze Mode if the FRZ bit is asserted.<br>0 = No Freeze Mode request. |
| NOT_RDY | FlexCAN Not Ready<br>This read-only bit indicates that FlexCAN is either in Disable Mode or Freeze Mode. It is negated once FlexCAN has exited these modes.<br>1 = FlexCAN module is either in Disable Mode Freeze Mode<br>0 = FlexCAN module is either in Normal Mode, Listen-Only Mode or Loop-Back Mode |
| SOFT_RST | Soft Reset<br>When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR, IMRL, IMRH, IFRL, and IFRH. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected:<br>• CTRL<br>• RXIMR0–RXIMR63<br>• RXGMASK, RX14MASK, RX15MASK<br>• all Message Buffers<br>The SOFT_RST bit can be asserted directly by the CPU when it writes to the MCR Register, but it is also asserted when global soft reset is requested at MCU level. Since soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFT_RST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed.<br>Soft reset cannot be applied while clocks are shut down in any of the low power modes. The module should be first removed from low power mode, and then soft reset can be applied.<br>1 = Resets the registers marked as “affected by soft reset” in Table 20-2<br>0 = No reset request |
| FRZ_ACK | Freeze Mode Acknowledge<br>This read-only bit indicates that FlexCAN is in Freeze Mode and its prescaler is stopped. The Freeze Mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZ_ACK bit to know when FlexCAN has actually entered Freeze Mode. If Freeze Mode request is negated, then this bit is negated once the FlexCAN prescaler is running again. If Freeze Mode is requested while FlexCAN is in any of the low power modes, then the FRZ_ACK bit will only be set when the low power mode is exited. See Section 20.5.9.1, “Freeze mode“ for more information.<br>1 = FlexCAN in Freeze Mode, prescaler stopped<br>0 = FlexCAN not in Freeze Mode, prescaler running |
| SUPV | Supervisor Mode<br>This bit configures some of the FlexCAN registers to be either in Supervisor or Unrestricted memory space. The registers affected by this bit are marked as S/U in the Access Type column of Table 20-2. Reset value of this bit is ‘1’, so the affected registers start with Supervisor access restrictions.<br>1 = Affected registers are in Supervisor memory space. Any access without supervisor permission behaves as though the access was done to an unimplemented register location<br>0 = Affected registers are in Unrestricted memory space |

**Table 20-8. MCR field descriptions (continued)**

| Field | Description |
|---|---|
| WRN_EN | Warning Interrupt Enable<br>When asserted, this bit enables the generation of the TWRN_INT and RWRN_INT flags in the Error and Status Register. If WRN_EN is negated, the TWRN_INT and RWRN_INT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated.<br>1 = TWRN_INT and RWRN_INT bits are set when the respective error counter transition from <96 to ≥ 96.<br>0 = TWRN_INT and RWRN_INT bits are zero, independent of the values in the error counters. |
| LPM_ACK | Low Power Mode Acknowledge<br>This read-only bit indicates that FlexCAN is in Disable Mode. This mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPM_ACK bit to know when FlexCAN has actually entered low power mode. See Section 20.5.9.2, “Module Disable mode“” for more information.<br>1 = FlexCAN is either in Disable Mode<br>0 = FlexCAN not in any of the low power modes |
| D_RSVD | Reserved bit.<br>This bit is writable but should be kept as value 0. |
| SRX_DIS | Self Reception Disable<br>This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception.<br>1 = Self reception disabled<br>0 = Self reception enabled |
| BCC | Backwards Compatibility Configuration<br>This bit is provided to support Backwards Compatibility with previous FlexCAN versions. When this bit is negated, the following configuration is applied:<br>• For MCUs supporting individual Rx ID masking, this feature is disabled. Instead of individual ID masking per MB, FlexCAN uses its previous masking scheme with RXGMASK, RX14MASK and RX15MASK.<br>• The reception queue feature is disabled. Upon receiving a message, if the first MB with a matching ID that is found is still occupied by a previous unread message, FlexCAN will not look for another matching MB. It will override this MB with the new message and set the CODE field to ‘0110’ (overrun).<br>Upon reset this bit is negated, allowing legacy software to work without modification.<br>1 = Individual Rx masking and queue feature are enabled.<br>0 = Individual Rx masking and queue feature are disabled. |
| LPRIO_EN | Local Priority Enable<br>This bit is provided for backwards compatibility reasons. It controls whether the local priority feature is enabled or not. It is used to extend the ID used during the arbitration process. With this extended ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames.<br>1 = Local Priority enabled<br>0 = Local Priority disabled |

**Table 20-8. MCR field descriptions (continued)**

| Field | Description |
|---|---|
| IDAM | ID Acceptance Mode<br>This 2-bit field identifies the format of the elements of the Rx FIFO filter table, as shown in Table 20-9. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Section 20.4.3, “Rx FIFO structure“. |
| MAXMB | Maximum Number of Message Buffers<br>This 6-bit field defines the maximum number of message buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to 16 MB configuration. This field should be changed only while the module is in Freeze Mode.<br>**Maximum MBs in use = MAXMB + 1**<br><br>**Note:** MAXMB has to be programmed with a value smaller or equal to the number of available Message Buffers, otherwise FlexCAN will not transmit or receive frames. |

**Table 20-9. IDAM coding**

| IDAM | Format | Explanation |
|---|---|---|
| 00 | A | One full ID (standard or extended) per filter element. |
| 01 | B | Two full standard IDs or two partial 14-bit extended IDs per filter element. |
| 10 | C | Four partial 8-bit IDs (standard or extended) per filter element. |
| 11 | D | All frames rejected. |

### 20.4.4.2 Control Register (CTRL)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back Mode, Listen Only Mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler. Most of the fields in this register should only be changed while the module is in Disable Mode or in Freeze Mode. Exceptions are the BOFF_MSK, ERR_MSK, TWRN_MSK, RWRN_MSK and BOFF_REC bits, that can be accessed at any time.

Base + 0x0004

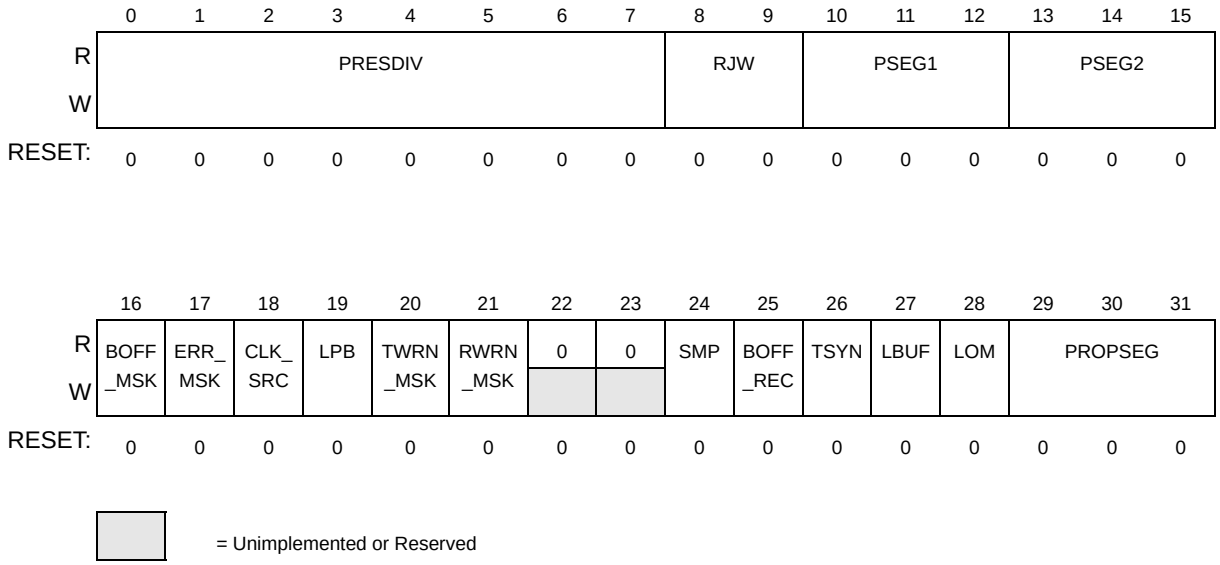



**Figure 20-6. Control Register (CTRL)**

**Table 20-10. Control Register (CTRL) Field Descriptions**

| Field | Description |
|---|---|
| 0-7<br>PRESDIV | Prescaler Division Factor<br>This 8-bit field defines the ratio between the CPI clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the CPI clock frequency. The Maximum value of this register is 0xFF, that gives a minimum Sclock frequency equal to the CPI clock frequency divided by 256. For more information refer to Section 20.5.8.4, “Protocol timing“.<br>**Sclock frequency = CPI clock frequency / (PRESDIV + 1)** |
| 8-9<br>RJW | Resync Jump Width<br>This 2-bit field defines the maximum number of time quanta[1] that a bit time can be changed by one resynchronization. The valid programmable values are 0–3.<br>**Resync Jump Width = RJW + 1.** |
| 10-12<br>PSEG1 | PSEG1 — Phase Segment 1<br>This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time. The valid programmable values are 0–7.<br>**Phase Buffer Segment 1 = (PSEG1 + 1) x Time-Quanta.** |
| 13-15<br>PSEG2 | PSEG2 — Phase Segment 2<br>This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time. The valid programmable values are 1–7.<br>**Phase Buffer Segment 2 = (PSEG2 + 1) x Time-Quanta.** |
| 16<br>BOFF_MSK | Bus Off Mask<br>This bit provides a mask for the Bus Off Interrupt.<br>1= Bus Off interrupt enabled<br>0 = Bus Off interrupt disabled |

**Table 20-10. Control Register (CTRL) Field Descriptions (continued)**

| Field | Description |
|---|---|
| 17<br>ERR_MSK | Error Mask<br>This bit provides a mask for the Error Interrupt.<br>1 = Error interrupt enabled<br>0 = Error interrupt disabled |
| 18<br>CLK_SRC | CAN Engine Clock Source<br>This bit selects the clock source to the CAN Protocol Interface (CPI) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (Sclock). In order to guarantee reliable operation, this bit should only be changed while the module is in Disable Mode. See Section 20.5.8.4, “Protocol timing“ for more information.<br>1 = The CAN engine clock source is the bus clock<br>0 = The CAN engine clock source is the oscillator clock<br><br>**Note:** This clock selection feature may not be available in all MCUs. A particular MCU may not have a PLL, in which case it would have only the oscillator clock, or it may use only the PLL clock feeding the FlexCAN module. In these cases, this bit has no effect on the module operation. |
| 19<br>TWRN_MSK | Tx Warning Interrupt Mask<br>This bit provides a mask for the Tx Warning Interrupt associated with the TWRN_INT flag in the Error and Status Register. This bit has no effect if the WRN_EN bit in MCR is negated and it is read as zero when WRN_EN is negated.<br>1 = Tx Warning Interrupt enabled<br>0 = Tx Warning Interrupt disabled |
| 20<br>RWRN_MSK | Rx Warning Interrupt Mask<br>This bit provides a mask for the Rx Warning Interrupt associated with the RWRN_INT flag in the Error and Status Register. This bit has no effect if the WRN_EN bit in MCR is negated and it is read as zero when WRN_EN is negated.<br>1 = Rx Warning Interrupt enabled<br>0 = Rx Warning Interrupt disabled |
| 21<br>LPB | Loop Back<br>This bit configures FlexCAN to operate in Loop-Back Mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic ‘1’). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated.<br>1 = Loop Back enabled<br>0 = Loop Back disabled |
| 24<br>SMP | Sampling Mode<br>This bit defines the sampling mode of CAN bits at the Rx input.<br>1 = Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples, a majority rule is used<br>0 = Just one sample is used to determine the bit value |

**Table 20-10. Control Register (CTRL) Field Descriptions (continued)**

| Field | Description |
|---|---|
| 25<br>BOFF_REC | Bus Off Recovery Mode<br>This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFF_REC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will resynchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFF_REC bit can be re-asserted again during Bus Off, but it will only be effective the next time the module enters Bus Off. If BOFF_REC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery.<br>1 = Automatic recovering from Bus Off state disabled<br>0 = Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B |
| 26<br>TSYN | Timer Sync Mode<br>This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message (i.e., global network time). If the FEN bit in MCR is set (FIFO enabled), MB8 is used for timer synchronization instead of MB0.<br>1 = Timer Sync feature enabled<br>0 = Timer Sync feature disabled |
| 27<br>LBUF | Lowest Buffer Transmitted First<br>This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIO_EN bit does not affect the priority arbitration.<br>1 = Lowest number buffer is transmitted first<br>0 = Buffer with highest priority is transmitted first |
| 28<br>LOM | Listen-Only Mode<br>This bit configures FlexCAN to operate in Listen Only Mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode [Ref. 1]. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.<br>1 = FlexCAN module operates in Listen Only Mode<br>0 = Listen Only Mode is deactivated |
| 29-31<br>PROPSEG | Propagation Segment<br>This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7.<br>Propagation Segment Time = (PROPSEG + 1) * Time-Quanta.<br>Time-Quantum = one Sclock period. |

1 One time quantum is equal to the Sclock period.

### 20.4.4.3 Free Running Timer (TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0000 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is clocked by the FlexCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When

there is no message on the bus, it counts using the previously programmed baud rate. During Freeze Mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

Writing to the timer is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

Base + 0x0008

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TIMER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-7. Free Running Timer (TIMER)**

### 20.4.4.4 Rx Global Mask (RXGMASK)

This register is provided for legacy support and for low cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in MCR causes the RXGMASK Register to have no effect on the module operation. For MCUs not supporting individual masks per MB, this register is always effective.

RXGMASK is used as acceptance mask for all Rx MBs, excluding MBs 14–15, which have individual mask registers. When the FEN bit in MCR is set (FIFO enabled), the RXGMASK also applies to all elements of the ID filter table, except elements 6-7, which have individual masks.

The contents of this register must be programmed while the module is in Freeze Mode, and must not be modified when the module is transmitting or receiving frames.

To avoid mask misalignment for Rx FIFO, it is recommended to take one of the following actions for avoiding problems:

- Do not enable the RxFIFO. If MCR[FEN]=0 then the Rx FIFO is disabled and thus the masksRXGMASK, RX14MASK and RX15MASK do not affect it.
- Enable Rx Individual Mask Registers. If MCR[BCC] = 1, then the RXIMRs are enabled and thus the masks RXGMASK, RX14MASK and RX15MASK are not used.
- Do not use masks RXGMASK, RX14MASK and RX15MASK (that is, let them in reset value) when MCR[FEN] = 1 and MCR[BCC] = 0. In this case, filtering processes for both Rx MBs and Rx FIFO are not affected by those masks.
- Do not configure any MB as Rx (that is, let all MBs as either Tx or inactive) when MCR[FEN] = 1 and MCR[BCC] = 0. In this case, the masks RXGMASK, RX14MASK and RX15MASK can be used to affect ID Tables without affecting filtering process for Rx MBs.

Base + 0x0010

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MI31 | MI30 | MI29 | MI28 | MI27 | MI26 | MI25 | MI24 | MI23 | MI22 | MI21 | MI20 | MI19 | MI18 | MI17 | MI16 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MI15 | MI14 | MI13 | MI12 | MI11 | MI10 | MI9 | MI8 | MI7 | MI6 | MI5 | MI4 | MI3 | MI2 | MI1 | MI0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

= Unimplemented or Reserved

**Figure 20-8. Rx Global Mask Register (RXGMASK)**

**Table 20-11. Rx Global Mask Register (RXGMASK) Field Descriptions**

| Field | Description |
|---|---|
| 0-31<br>MI31 - MI0 | Mask Bits<br>For normal Rx MBs, the mask bits affect the ID filter programmed on the MB. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR).<br>1 = The corresponding bit in the filter is checked against the one received<br>0 = the corresponding bit in the filter is “don’t care” |

### 20.4.4.5 Rx 14 Mask (RX14MASK)

This register is provided for legacy support and for low cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in MCR causes the RX14MASK Register to have no effect on the module operation.

RX14MASK is used as acceptance mask for the Identifier in Message Buffer 14. When the FEN bit in MCR is set (FIFO enabled), the RXG14MASK also applies to element 6 of the ID filter table. This register has the same structure as the Rx Global Mask Register. It must be programmed while the module is in Freeze Mode, and must not be modified when the module is transmitting or receiving frames.

- Address Offset: 0x14
- Reset Value: 0xFFFF_FFFF

### 20.4.4.6 Rx 15 Mask (RX15MASK)

This register is provided for legacy support and for low cost MCUs that do not have the individual masking per Message Buffer feature. For MCUs supporting individual masks per MB, setting the BCC bit in MCR causes the RX15MASK Register to have no effect on the module operation.

When the BCC bit is negated, RX15MASK is used as acceptance mask for the Identifier in Message Buffer 15. When the FEN bit in MCR is set (FIFO enabled), the RXG14MASK also applies to element 7 of the ID filter table. This register has the same structure as the Rx Global Mask Register. It must be programmed while the module is in Freeze Mode, and must not be modified when the module is transmitting or receiving frames.

- Address Offset: 0x18
- Reset Value: 0xFFFF_FFFF

### 20.4.4.7 Error Counter Register (ECR)

This register has 2 8-bit fields reflecting the value of two FlexCAN error counters: Transmit Error Counter (Tx_Err_Counter field) and Receive Error Counter (Rx_Err_Counter field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read only except in Freeze Mode, where they can be written by the CPU.

Writing to the Error Counter Register while in Freeze Mode is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data will take some time to be actually written to the register. If desired, software can poll the register to discover when the data was actually written.

FlexCAN responds to any bus state as described in the protocol, e.g. transmit ‘Error Active’ or ‘Error Passive’ flag, delay its transmission start time (‘Error Passive’) and avoid any influence on the bus when in ‘Bus Off’ state. The following are the basic rules for FlexCAN bus state transitions.

- If the value of Tx_Err_Counter or Rx_Err_Counter increases to be greater than or equal to 128, the FLT_CONF field in the Error and Status Register is updated to reflect ‘Error Passive’ state.

- If the FlexCAN state is ‘Error Passive’, and either Tx_Err_Counter or Rx_Err_Counter decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLT_CONF field in the Error and Status Register is updated to reflect ‘Error Active’ state.
- If the value of Tx_Err_Counter increases to be greater than 255, the FLT_CONF field in the Error and Status Register is updated to reflect ‘Bus Off’ state, and an interrupt may be issued. The value of Tx_Err_Counter is then reset to zero.
- If FlexCAN is in ‘Bus Off’ state, then Tx_Err_Counter is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, Tx_Err_Counter is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the Tx_Err_Counter. When Tx_Err_Counter reaches the value of 128, the FLT_CONF field in the Error and Status Register is updated to be ‘Error Active’ and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the Tx_Err_Counter value.
- If during system start-up, only one node is operating, then its Tx_Err_Counter increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACK_ERR bit in the Error and Status Register). After the transition to ‘Error Passive’ state, the Tx_Err_Counter does not increment anymore by acknowledge errors. Therefore the device never goes to the ‘Bus Off’ state.
- If the Rx_Err_Counter increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to ‘Error Active’ state.

Base + 0x001C

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Rx_Err_Counter | | | | | | | | Tx_Err_Counter | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-9. Error Counter Register (ECR)**

### 20.4.4.8 Error and Status Register (ESR)

This register reflects various error conditions, some general status of the device and it is the source of four interrupts to the CPU. The reported error conditions (bits 16-21) are those that occurred since the last time the CPU read this register. The CPU read action clears bits 16-21. Bits 22-27 are status bits.

Most bits in this register are read only, except TWRN_INT, RWRN_INT, BOFF_INT and ERR_INT, that are interrupt flags that can be cleared by writing ‘1’ to them (writing ‘0’ has no effect). See Section 20.5.10, “Interrupts“ for more details.

Base + 0x0020

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | TWRN_INT | RWRN_INT |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BIT1_ERR | BIT0_ERR | ACK_ERR | CRC_ERR | FRM_ERR | STF_ERR | TX_WRN | RX_WRN | IDLE | TXRX | FLT_CONF | | 0 | BOFF_INT | ERR_INT | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-10. Error and Status Register (ESR)**

**Table 20-12. Error and Status Register (ESR) Field Descriptions**

| Field | Description |
|---|---|
| 14<br>TWRN_INT | Tx Warning Interrupt Flag<br>If the WRN_EN bit in MCR is asserted, the TWRN_INT bit is set when the TX_WRN flag transition from ‘0’ to ‘1’, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control Register (TWRN_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’. Writing ‘0’ has no effect.<br>1 = The Tx error counter transition from < 96 to ≥ 96<br>0 = No such occurrence |
| 15<br>RWRN_INT | Rx Warning Interrupt Flag<br>If the WRN_EN bit in MCR is asserted, the RWRN_INT bit is set when the RX_WRN flag transition from ‘0’ to ‘1’, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRN_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’. Writing ‘0’ has no effect.<br>1 = The Rx error counter transition from < 96 to ≥ 96<br>0 = No such occurrence |

**Table 20-12. Error and Status Register (ESR) Field Descriptions (continued)**

| Field | Description |
|---|---|
| 16<br>BIT1_ERR | Bit1 Error<br>This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message.<br>1 = At least one bit sent as recessive is received as dominant<br>0 = No such occurrence<br><br>**Note:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. |
| 17<br>BIT0_ERR | Bit0 Error<br>This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message.<br>1 = At least one bit sent as dominant is received as recessive<br>0 = No such occurrence |
| 18<br>ACK_ERR | Acknowledge Error<br>This bit indicates that an Acknowledge Error has been detected by the transmitter node, i.e., a dominant bit has not been detected during the ACK SLOT.<br>1 = An ACK error occurred since last read of this register<br>0 = No such occurrence |
| 19<br>CRC_ERR | Cyclic Redundancy Check Error<br>This bit indicates that a CRC Error has been detected by the receiver node, i.e., the calculated CRC is different from the received.<br>1 = A CRC error occurred since last read of this register.<br>0 = No such occurrence |
| 20<br>FRM_ERR | Form Error<br>This bit indicates that a Form Error has been detected by the receiver node, i.e., a fixed-form bit field contains at least one illegal bit.<br>1 = A Form Error occurred since last read of this register<br>0 = No such occurrence |
| 21<br>STF_ERR | Stuffing Error<br>This bit indicates that a Stuffing Error has been detected.<br>1 = A Stuffing Error occurred since last read of this register.<br>0 = No such occurrence. |
| 22<br>TX_WRN | TX Error Counter<br>This bit indicates when repetitive errors are occurring during message transmission.<br>1 = TX_Err_Counter ≥ 96<br>0 = No such occurrence |
| 23<br>RX_WRN | Rx Error Counter<br>This bit indicates when repetitive errors are occurring during message reception.<br>1 = Rx_Err_Counter ≥ 96<br>0 = No such occurrence |
| 24<br>IDLE | CAN bus IDLE state<br>This bit indicates when CAN bus is in IDLE state.<br>1 = CAN bus is now IDLE<br>0 = No such occurrence |

**Table 20-12. Error and Status Register (ESR) Field Descriptions (continued)**

| Field | Description |
|---|---|
| 25<br>TXRX | Current FlexCAN status (transmitting/receiving)<br>This bit indicates if FlexCAN is transmitting or receiving a message when the CAN bus is not in IDLE state. This bit has no meaning when IDLE is asserted.<br>1 = FlexCAN is transmitting a message (IDLE=0)<br>0 = FlexCAN is receiving a message (IDLE=0) |
| 26-27<br>FLT_CONF | Fault Confinement State<br>This 2-bit field indicates the Confinement State of the FlexCAN module, as shown in Table 20-13. If the LOM bit in the Control Register is asserted, the FLT_CONF field will indicate “Error Passive”. Since the Control Register is not affected by soft reset, the FLT_CONF field will not be affected by soft reset if the LOM bit is asserted. |
| 29<br>BOFF_INT | Bus Off’ Interrupt<br>This bit is set when FlexCAN enters ‘Bus Off’ state. If the corresponding mask bit in the Control Register (BOFF_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’. Writing ‘0’ has no effect.<br>1 = FlexCAN module entered ‘Bus Off’ state<br>0 = No such occurrence |
| 30<br>ERR_INT | Error Interrupt<br>This bit indicates that at least one of the Error Bits (bits 16-21) is set. If the corresponding mask bit in the Control Register (ERR_MSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to ‘1’.Writing ‘0’ has no effect.<br>1 = Indicates setting of any Error Bit in the Error and Status Register<br>0 = No such occurrence |
| 31 | Reserved |

**Table 20-13. Fault Confinement State**

| Value | Meaning |
|---|---|
| 00 | Error Active |
| 01 | Error Passive |
| 1X | Bus Off |

### 20.4.4.9 Interrupt Mask Register High (IMRH)

This register allows any number of a range of 32 Message Buffer Interrupts to be enabled or disabled. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (i.e. when the corresponding IFRH bit is set).

Base + 0x0024

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF 63M | BUF 62M | BUF 61M | BUF 60M | BUF 59M | BUF 58M | BUF 57M | BUF 56M | BUF 55M | BUF 54M | BUF 53M | BUF 52M | BUF 51M | BUF 50M | BUF 49M | BUF 48M |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF 47M | BUF 46M | BUF 45M | BUF 44M | BUF 43M | BUF 42M | BUF 41M | BUF 40M | BUF 39M | BUF 38M | BUF 37M | BUF 36M | BUF 35M | BUF 34M | BUF 33M | BUF 32M |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-11. Interrupt Mask Register High (IMRH)**

**Table 20-14. IMRH field descriptions**

| Field | Description |
|---|---|
| BUF63M – BUF32M | Buffer $MB_i$ Mask<br>Each bit enables or disables the respective FlexCAN Message Buffer (MB32 to MB63) Interrupt.<br>1 = The corresponding buffer Interrupt is enabled<br>0 = The corresponding buffer Interrupt is disabled<br><br>**Note:** Setting or clearing a bit in the IMRH register can assert or negate an interrupt request, if the corresponding IFRH bit is set. |

### 20.4.4.10 Interrupt Mask Register Low (IMRL)

This register allows to enable or disable any number of a range of 32 Message Buffer Interrupts. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception (i.e., when the corresponding IFRL bit is set).

Base + 0x0028

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

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

**Figure 20-12. Interrupt Mask Register Low (IMRL)**

**Table 20-15. IMRL field descriptions**

| Field | Description |
|---|---|
| BUF31M – BUF0M | BUF31M–BUF0M — Buffer $MB_i$ Mask<br>Each bit enables or disables the respective FlexCAN Message Buffer (MB0 to MB31) Interrupt.<br>1 = The corresponding buffer Interrupt is enabled<br>0 = The corresponding buffer Interrupt is disabled<br><br>**Note:** Setting or clearing a bit in the IMRL register can assert or negate an interrupt request, if the corresponding IFRL bit is set. |

### 20.4.4.11 Interrupt Flag Register High (IFRH)

This register defines the flags for 32 Message Buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFRH bit. If the corresponding IMRH bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing it to ‘1’. Writing ‘0’ has no effect.

Base + 0x002C

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF 63I | BUF 62I | BUF 61I | BUF 60I | BUF 59I | BUF 58I | BUF 57I | BUF 56I | BUF 55I | BUF 54I | BUF 53I | BUF 52I | BUF 51I | BUF 50I | BUF 49I | BUF 48I |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF 47I | BUF 46I | BUF 45I | BUF 44I | BUF 43I | BUF 42I | BUF 41I | BUF 40I | BUF 39I | BUF 38I | BUF 37I | BUF 36I | BUF 35I | BUF 34I | BUF 33I | BUF 32I |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-13. Interrupt Flag Register High (IFRH)**

**Table 20-16. IFRH field descriptions**

| Field | Description |
|---|---|
| BUF32I – BUF63I | Buffer $MB_i$ Interrupt<br>Each bit flags the respective FlexCAN Message Buffer (MB32 to MB63) interrupt.<br>1 = The corresponding buffer has successfully completed transmission or reception<br>0 = No such occurrence |

### 20.4.4.12 Interrupt Flag Register Low (IFRL)

This register defines the flags for 32 Message Buffer interrupts and FIFO interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFRL bit. If the corresponding IMRL bit is set, an interrupt will be generated. The Interrupt flag must be cleared by writing it to ‘1’. Writing ‘0’ has no effect.

When the MCR[FEN] is set (FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I - BUF0I) is changed to support the FIFO operation. BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, while BUF4I to BUF0I are not used.

Base + 0x0030

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF 31I | BUF 30I | BUF 29I | BUF 28I | BUF 27I | BUF 26I | BUF 25I | BUF 24I | BUF 23I | BUF 22I | BUF 21I | BUF 20I | BUF 19I | BUF 18I | BUF 17I | BUF 16I |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF 15I | BUF 14I | BUF 13I | BUF 12I | BUF 11I | BUF 10I | BUF 9I | BUF 8I | BUF 7I | BUF 6I | BUF 5I | BUF 4I | BUF 3I | BUF 2I | BUF 1I | BUF 0I |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-14. Interrupt Flag Register Low (IFRL)**

**Table 20-17. IFRL field descriptions**

| Field | Description |
|---|---|
| BUF31I – BUF8I | Buffer $MB_i$ Interrupt<br>Each bit flags the respective FlexCAN Message Buffer (MB8 to MB31) interrupt.<br>1 = The corresponding MB has successfully completed transmission or reception<br>0 = No such occurrence |
| BUF7I | Buffer MB7 Interrupt or “FIFO Overflow”<br>If the FIFO is not enabled, this bit flags the interrupt for MB7. If the FIFO is enabled, this flag indicates an overflow condition in the FIFO (frame lost because FIFO is full).<br>1 = MB7 completed transmission/reception or FIFO overflow<br>0 = No such occurrence |
| BUF6I | Buffer MB6 Interrupt or “FIFO Warning”<br>If the FIFO is not enabled, this bit flags the interrupt for MB6. If the FIFO is enabled, this flag indicates that 4 out of 6 buffers of the FIFO are already occupied (FIFO almost full).<br>1 = MB6 completed transmission/reception or FIFO almost full<br>0 = No such occurrence |
| BUF5I | Buffer MB5 Interrupt or “Frames available in FIFO”<br>If the FIFO is not enabled, this bit flags the interrupt for MB5. If the FIFO is enabled, this flag indicates that at least one frame is available to be read from the FIFO.<br>1 = MB5 completed transmission/reception or frames available in the FIFO<br>0 = No such occurrence |
| BUF4I – BUF0I | Buffer $MB_i$ Interrupt or “reserved”<br>If the FIFO is not enabled, these bits flag the interrupts for MB0 to MB4. If the FIFO is enabled, these flags are not used and must be considered as reserved locations.<br>1 = Corresponding MB completed transmission/reception<br>0 = No such occurrence |

### 20.4.4.13 Rx Individual Mask Registers (RXIMR0–RXIMR63)

These registers are used as acceptance masks for ID filtering in Rx MBs and the FIFO. If the FIFO is not enabled, one mask register is provided for each available Message Buffer, providing ID masking capability

on a per Message Buffer basis. When the FIFO is enabled (FEN bit in MCR is set), the first 8 Mask Registers apply to the 8 elements of the FIFO filter table (on a one-to-one correspondence), while the rest of the registers apply to the regular MBs, starting from MB8.

The Individual Rx Mask Registers are implemented in RAM, so they are not affected by reset and must be explicitly initialized prior to any reception. Furthermore, they can only be accessed by the CPU while the module is in Freeze Mode. Out of Freeze Mode, write accesses are blocked and read accesses will return “all zeros”. Furthermore, if the BCC bit in the MCR Register is negated, any read or write operation to these registers results in access error.

Base + 0x0004

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MI31 | MI30 | MI29 | MI28 | MI27 | MI26 | MI25 | MI24 | MI23 | MI22 | MI21 | MI20 | MI19 | MI18 | MI17 | MI16 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MI15 | MI14 | MI13 | MI12 | MI11 | MI10 | MI9 | MI8 | MI7 | MI6 | MI5 | MI4 | MI3 | MI2 | MI1 | MI0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-15. Rx Individual Mask Registers (RXIMR0 - RXIMR63)**

**Table 20-18. Rx Individual Mask Registers (RXIMR0 - RXIMR63) Field Descriptions**

| Field | Description |
|---|---|
| 0-31<br>MI31–MI0 | Mask Bits<br>For normal Rx MBs, the mask bits affect the ID filter programmed on the MB. For the Rx FIFO, the mask bits affect all bits programmed in the filter table (ID, IDE, RTR).<br>1 = The corresponding bit in the filter is checked against the one received<br>0 = the corresponding bit in the filter is “don’t care” |

## 20.5 Functional description

### 20.5.1 Overview

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of up to 64 Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Section 20.4.2, “Message Buffer Structure“). The memory corresponding to the first 8 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 8 extended IDs or 16 standard IDs or 32 8-bit ID slices), each one with its own individual mask register. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the

same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be “active” at a given time if it can participate in the matching and arbitration algorithms that are happening at that time. An Rx MB with a ‘0000’ code is inactive (refer to Table 20-5). Similarly, a Tx MB with a ‘1000’ or ‘1001’ code is also inactive (refer to Table 20-6). An MB not programmed with ‘0000’, ‘1000’ or ‘1001’ will be temporarily deactivated (will not participate in the current arbitration or matching run) when the CPU writes to the C/S field of that MB (see Section 20.5.6.1, “Message Buffer deactivation“).

## 20.5.2 Transmit process

In order to transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

- Write the ID word.
- Write the data bytes.
- Write the Length, Control and Code fields of the Control and Status word to activate the MB.

Once the MB is activated in the fourth step, it will participate into the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the Code field in the Control and Status word is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new Code field after transmission depends on the code that was used to activate the MB in step four (see Table 20-5 and Table 20-6 in Section 20.4.2, “Message Buffer Structure“).

## 20.5.3 Arbitration process

The arbitration process is an algorithm executed by the MBM that scans the whole MB memory looking for the highest priority message to be transmitted. All MBs programmed as transmit buffers will be scanned to find the lowest ID[1] or the lowest MB number or the highest priority, depending on the LBUF and LPRIO_EN bits on the Control Register. The arbitration process is triggered in the following events:

- During the CRC field of the CAN frame
- During the error delimiter field of the CAN frame
- During Intermission, if the winner MB defined in a previous arbitration was deactivated, or if there was no MB to transmit, but the CPU wrote to the C/S word of any MB after the previous arbitration finished
- When MBM is in Idle or Bus Off state and the CPU writes to the C/S word of any MB
- Upon leaving Freeze Mode

1. Actually, if LBUF is negated, the arbitration considers not only the ID, but also the RTR and IDE bits placed inside the ID at the same positions they are transmitted in the CAN frame.

When LBUF is asserted, the LPRIO_EN bit has no effect and the lowest number buffer is transmitted first. When LBUF and LPRIO_EN are both negated, the MB with the lowest ID is transmitted first but. If LBUF is negated and LPRIO_EN is asserted, the PRIO bits augment the ID used during the arbitration process. With this extended ID concept, arbitration is done based on the full 32-bit ID and the PRIO bits define which MB should be transmitted first, therefore MBs with PRIO = 000 have higher priority. If two or more MBs have the same priority, the regular ID will determine the priority of transmission. If two or more MBs have the same priority (3 extra bits) and the same regular ID, the lowest MB will be transmitted first.

Once the highest priority MB is selected, it is transferred to a temporary storage space called Serial Message Buffer (SMB), which has the same structure as a normal MB but is not user accessible. This operation is called “move-out” and after it is done, write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in HALT or BUS OFF
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the DLC (Data Length Code) value is bigger.

### 20.5.4 Receive process

To be able to receive CAN frames into the mailbox MBs, the CPU must prepare one or more Message Buffers for reception by executing the following steps:

- Write the ID word
- Write ‘0100’ to the Code field of the Control and Status word to activate the MB

Once the MB is activated in the third step, it will be able to receive frames that match the programmed ID. At the end of a successful reception, the MB is updated by the MBM as follows:

- The value of the Free Running Timer is written into the Time Stamp field
- The received ID, Data (8 bytes at most) and Length fields are stored
- The Code field in the Control and Status word is updated (see Table 20-5 and Table 20-6 in Section 20.4.2, “Message Buffer Structure“)
- A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit

Upon receiving the MB interrupt, the CPU should service the received frame using the following procedure:

- Read the Control and Status word (mandatory – activates an internal lock for this buffer)
- Read the ID field (optional – needed only if a mask was used)
- Read the Data field
- Read the Free Running Timer (optional – releases the internal lock)

Upon reading the Control and Status word, if the BUSY bit is set in the Code field, then the CPU should defer the access to the MB until this bit is negated. Reading the Free Running Timer is not mandatory. If

not executed the MB remains locked, unless the CPU reads the C/S word of another MB. Note that only a single MB is locked at a time. The only mandatory CPU read operation is the one on the Control and Status word to assure data coherency (see Section 20.5.6, “Data Coherence“).

The CPU should synchronize to frame reception by the status flag bit for the specific MB in one of the interrupt flag (IFRL, IFRH) registers and not by the Code field of that MB. Polling the Code field does not work because once a frame was received and the CPU services the MB (by reading the C/S word followed by unlocking the MB), the Code field will not return to EMPTY. It will remain FULL, as explained in Table 20-5. If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the MB, the MB is actually deactivated from any currently ongoing matching process. As a result, a newly received frame matching the ID of that MB may be lost. In summary: *never do polling by reading directly the C/S word of the MBs. Instead, read the interrupt flag (IFRL, IFRH) registers.*

Note that the received ID field is always stored in the matching MB, thus the contents of the ID field in an MB may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists an Rx matching MB, provided the SRX_DIS bit in the MCR is not asserted. If SRX_DIS is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN frames through the FIFO, the CPU must enable and configure the FIFO during Freeze Mode (see Section 20.5.7, “Rx FIFO“). Upon receiving the frames available interrupt from FIFO, the CPU should service the received frame using the following procedure:

- Read the Control and Status word (optional – needed only if a mask was used for IDE and RTR bits)
- Read the ID field (optional – needed only if a mask was used)
- Read the Data field
- Clear the frames available interrupt (mandatory – release the buffer and allow the CPU to read the next FIFO entry)

### 20.5.5 Matching Process

The matching process is an algorithm executed by the MBM that scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the 8-entry ID table from FIFO is scanned first and then, if a match is not found within the FIFO table, the other MBs are scanned. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

When the frame is received, it is temporarily stored in a hidden auxiliary MB called Serial Message Buffer (SMB). The matching process takes place during the CRC field of the received frame. If a matching ID is found in the FIFO table or in one of the regular MBs, the contents of the SMB will be transferred to the FIFO or to the matched MB during the 6th bit of the End-Of-Frame field of the CAN protocol. This operation is called “move-in”. If any protocol error (CRC, ACK, etc.) is detected, than the move-in operation does not happen.

For the regular mailbox MBs, an MB is said to be “free to receive” a new frame if the following conditions are satisfied:

- The MB is not locked (see Section 20.5.6.2, “Message Buffer lock mechanism“)
- The Code field is either EMPTY or else it is FULL or OVERRUN but the CPU has already serviced the MB (read the C/S word and then unlocked the MB)

If the first MB with a matching ID is not “free to receive” the new frame, then the matching algorithm keeps looking for another free MB until it finds one. If it can not find one that is free, then it will overwrite the last matching MB (unless it is locked) and set the Code field to OVERRUN (refer to Table 20-5 and Table 20-6). If the last matching MB is locked, then the new message remains in the SMB, waiting for the MB to be unlocked (see Section 20.5.6.2, “Message Buffer lock mechanism“).

Suppose, for example, that the FIFO is disabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not “free to receive”, so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are “free to receive”, so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the Code field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

The matching algorithm described above can be changed to be the same one used in previous versions of the FlexCAN module. When the BCC bit in MCR is negated, the matching algorithm stops at the first MB with a matching ID that it founds, whether this MB is free or not. As a result, the message queueing feature does not work if the BCC bit is negated.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. Please refer to Section 20.4.4.13, “Rx Individual Mask Registers (RXIMR0–RXIMR63)“. During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is “don’t care”. Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed if the BCC bit is asserted and while the module is in Freeze Mode.

FlexCAN also supports an alternate masking scheme with only three mask registers (RGXMASK, RX14MASK and RX15MASK) for backwards compatibility. This alternate masking scheme is enabled when the BCC bit in the MCR Register is negated.

## 20.5.6 Data Coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in Transmit process and Section 20.5.4, “Receive process“. Any form of CPU accessing an MB structure within FlexCAN other than those specified may cause FlexCAN to behave in an unpredictable way.

### 20.5.6.1 Message Buffer deactivation

Deactivation is mechanism provided to maintain data coherence when the CPU writes to the Control and Status word of active MBs out of Freeze Mode. Any CPU write access to the Control and Status word of an MB causes that MB to be excluded from the transmit or receive processes during the current matching or arbitration round. The deactivation is temporary, affecting only for the current match/arbitration round.

The purpose of deactivation is data coherency. The match/arbitration process scans the MBs to decide which MB to transmit or receive. If the CPU updates the MB in the middle of a match or arbitration process, the data of that MB may no longer be coherent, therefore deactivation of that MB is done.

Even with the coherence mechanism described above, writing to the Control and Status word of active MBs when not in Freeze Mode may produce undesirable results. Examples are:

- Matching and arbitration are one-pass processes. If MBs are deactivated after they are scanned, no re-evaluation is done to determine a new match/winner. If an Rx MB with a matching ID is deactivated during the matching process after it was scanned, then this MB is marked as invalid to receive the frame, and FlexCAN will keep looking for another matching MB within the ones it has not scanned yet. If it can not find one, then the message will be lost. Suppose, for example, that two MBs have a matching ID to a received frame, and the user deactivated the first matching MB after FlexCAN has scanned the second. The received frame will be lost even if the second matching MB was “free to receive”.
- If a Tx MB containing the lowest ID is deactivated after FlexCAN has scanned it, then FlexCAN will look for another winner within the MBs that it has not scanned yet. Therefore, it may transmit an MB with ID that may not be the lowest at the time because a lower ID might be present in one of the MBs that it had already scanned before the deactivation.
- There is a point in time until which the deactivation of a Tx MB causes it not to be transmitted (end of move-out). After this point, it is transmitted but no interrupt is issued and the Code field is not updated.

### 20.5.6.2 Message Buffer lock mechanism

Besides MB deactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an “active not empty” Rx MB, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and thus it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB. The MB locking is done to prevent a new frame to be written into the MB while the CPU is reading it.

**NOTE**

The locking mechanism only applies to Rx MBs which have a code different than INACTIVE (‘0000’) or EMPTY[1] (‘0100’). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two

1. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior will be honoured when the BCC bit is negated.

MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no “free to receive” MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB. If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the Code field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the Code field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

**NOTE**

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Deactivation takes precedence over locking. If the CPU deactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the MB.

### 20.5.7 Rx FIFO

The receive-only FIFO is enabled by asserting the FEN bit in the MCR. The reset value of this bit is zero to maintain software backwards compatibility with previous versions of the module that did not have the FIFO feature. When the FIFO is enabled, the memory region normally occupied by the first 8 MBs (0x80-0xFF) is now reserved for use of the FIFO engine (see Section 20.4.3, “Rx FIFO structure“). Management of read and write pointers is done internally by the FIFO engine. The CPU can read the received frames sequentially, in the order they were received, by repeatedly accessing a Message Buffer structure at the beginning of the memory.

The FIFO can store up to six frames pending service by the CPU. An interrupt is sent to the CPU when new frames are available in the FIFO. Upon receiving the interrupt, the CPU must read the frame (accessing an MB in the 0x80 address) and then clear the interrupt. The act of clearing the interrupt triggers the FIFO engine to replace the MB in 0x80 with the next frame in the queue, and then issue another interrupt to the CPU. If the FIFO is full and more frames continue to be received, an OVERFLOW interrupt is issued to the CPU and subsequent frames are not accepted until the CPU creates space in the FIFO by reading one or more frames. A warning interrupt is also generated when four frames are accumulated in the FIFO.

A powerful filtering scheme is provided to accept only frames intended for the target application, thus reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of 8 32-bit registers that can be configured to one of the following formats (see also Section 20.4.3, “Rx FIFO structure“):

- Format A: 8 extended or standard IDs (including IDE and RTR)
- Format B: 16 standard IDs or 16 extended 14-bit ID slices (including IDE and RTR)
- Format C: 32 standard or extended 8-bit ID slices

**NOTE**

A chosen format is applied to all eight registers of the filter table. It is not possible to mix formats within the table.

The eight elements of the filter table are individually affected by the first eight Individual Mask Registers (RXIMR0 - RXIMR7), allowing very powerful filtering criteria to be defined. The rest of the RXIMR, starting from RXIM8, continue to affect the regular MBs, starting from MB8. If the BCC bit is negated (or if the RXIMR are not available for the particular MCU), then the FIFO filter table is affected by the legacy mask registers as follows: element 6 is affected by RX14MASK, element 7 is affected by RX15MASK and the other elements (0 to 5) are affected by RXGMASK.

## 20.5.8 CAN protocol related features

### 20.5.8.1 Remote frames

Remote frame is a special kind of frame. The user can program a MB to be a Request Remote Frame by writing the MB as Transmit with the RTR bit set to ‘1’. After the Remote Request frame is transmitted successfully, the MB becomes a Receive Message Buffer, with the same ID as before.

When a Remote Request frame is received by FlexCAN, its ID is compared to the IDs of the transmit message buffers with the Code field ‘1010’. If there is a matching ID, then this MB frame will be transmitted. Note that if the matching MB has the RTR bit set, then FlexCAN will transmit a Remote Frame as a response.

A received Remote Request Frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match.

In the case that a Remote Request Frame was received and matched an MB, this message buffer immediately enters the internal arbitration process, but is considered as normal Tx MB, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

If the Rx FIFO is enabled (bit FEN set in MCR), FlexCAN will not generate an automatic response for Remote Request Frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID).

### 20.5.8.2 Overload frames

FlexCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

### 20.5.8.3 Time stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of “move-in” in the TIME STAMP field, providing network behavior with respect to time.

Note that the Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization. Refer to TSYN description in Section 20.4.4.2, “Control Register (CTRL)“.

### 20.5.8.4 Protocol timing

Figure 20-16 shows the structure of the clock generation circuitry that feeds the CAN Protocol Interface (CPI) sub-module. The clock source bit (CLK_SRC) in the CTRL Register defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock (generally from a PLL). In order to guarantee reliable operation, the clock source should be selected while the module is in Disable Mode (bit MDIS set in the Module Configuration Register).

**NOTE**

On setting the MDIS bit , the module does not shut down the clocks, it only requests the shut down to the integration logic (CCB) that can or cannot disable the clocks. The clocks to the message buffer modules can be kept running also with the module disabled to keep the the memory available for general use

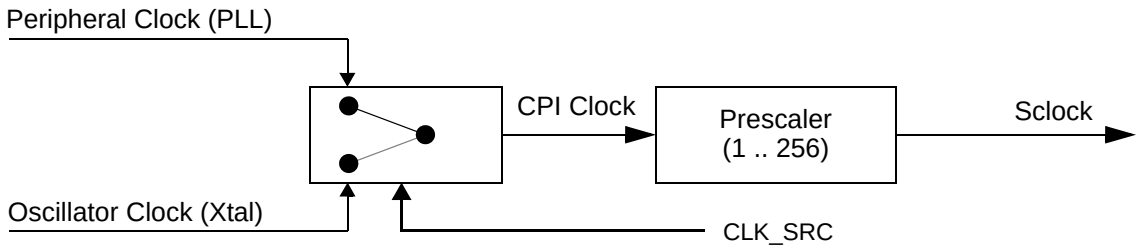



**Figure 20-16. CAN engine clocking scheme**

The crystal oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than PLL generated clocks.

**NOTE**

This clock selection feature may not be available in all MCUs. A particular MCU may not have a PLL, in which case it would have only the oscillator clock, or it may use only the PLL clock feeding the FlexCAN module. In these cases, the CLK_SRC bit in the CTRL Register has no effect on the module operation.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See Section 20.4.4.2, “Control Register (CTRL)“.

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the ‘time quantum’ used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(Prescaler\ value)}$$

A bit time is subdivided into three segments[1] (reference Figure 20-17 and Table 20-19):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL Register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL Register (plus 1) to be 2 to 8 time quanta long

$$Bit\ Rate = \frac{f_{Tq}}{(number\ of\ Time\ Quanta)}$$

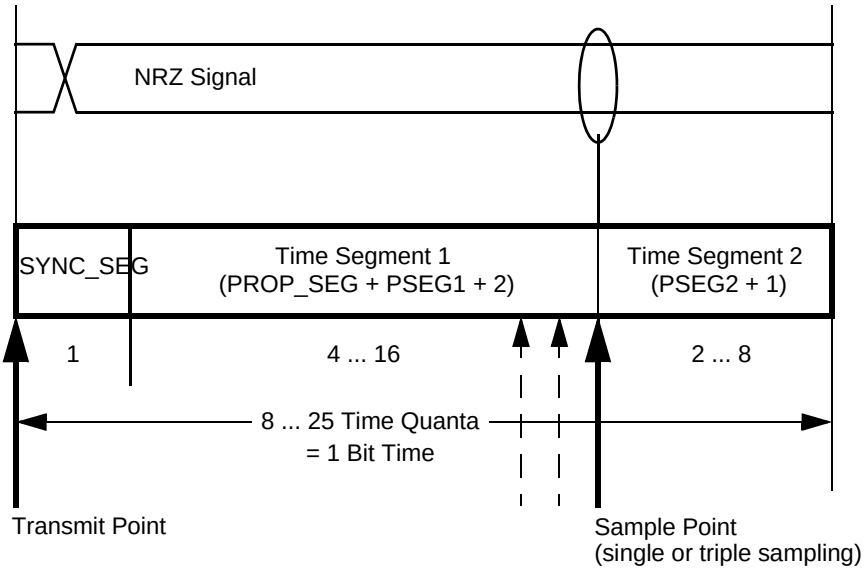



**Figure 20-17. Segments within the Bit Time**

1. For further explanation of the underlying concepts please refer to ISO/DIS 11519–1, Section 10.3. Reference also the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

**Table 20-19. Time Segment Syntax**

| Syntax | Description |
|---|---|
| SYNC_SEG | System expects transitions to occur on the bus during this period. |
| Transmit Point | A node in transmit mode transfers a new value to the CAN bus at this point. |
| Sample Point | A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample. |

Table 20-20 gives an overview of the CAN compliant segment settings and the related parameter values.

**Table 20-20. CAN Standard Compliant Bit Time Segment Settings**

| Time Segment 1 | Time Segment 2 | Resynchronization Jump Width |
|---|---|---|
| 5 .. 10 | 2 | 1 .. 2 |
| 4 .. 11 | 3 | 1 .. 3 |
| 5 .. 12 | 4 | 1 .. 4 |
| 6 .. 13 | 5 | 1 .. 4 |
| 7 .. 14 | 6 | 1 .. 4 |
| 8 .. 15 | 7 | 1 .. 4 |
| 9 .. 16 | 8 | 1 .. 4 |

**NOTE**

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard. For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module.

### 20.5.8.5 Arbitration and Matching Timing

During normal transmission or reception of frames, the arbitration, matching, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in Figure 20-18.

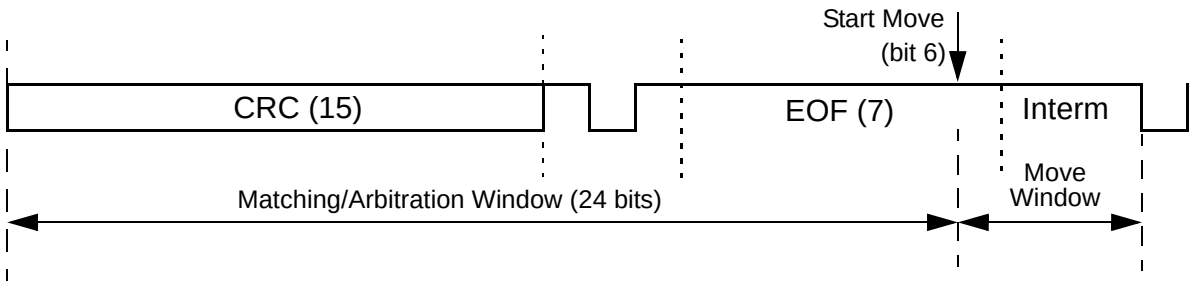



**Figure 20-18. Arbitration, Match and Move Time Windows**

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 20-20
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, i.e. the PLL can not be programmed to divide down the oscillator clock
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in Table 20-21

**Table 20-21. Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate**

| Number of Message Buffers | Minimum Ratio |
|---|---|
| 16 | 8 |
| 32 | 8 |
| 64 | 16 |

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, so the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in Table 20-21 can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2). As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least 2. For prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

## 20.5.9 Modes of operation: Details

### 20.5.9.1 Freeze mode

This mode is entered by asserting the HALT bit in the MCR Register or when the MCU is put into Debug Mode. In both cases it is also necessary that the FRZ bit is asserted in the MCR Register and the module is not in any of the low power modes (Disable, Stop). When Freeze Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze Mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible.

Exiting Freeze Mode is done in one of the following ways:

- CPU negates the FRZ bit in the MCR Register
- The MCU is removed from Debug Mode and/or the HALT bit is negated

Once out of Freeze Mode, FlexCAN tries to resynchronize to the CAN bus by waiting for 11 consecutive recessive bits.

#### 20.5.9.2 Module Disable mode

This low power mode is entered when the MDIS bit in the MCR Register is asserted. On setting the MDIS bit, the module does not shut down the clocks, it only requests the shut down to the integration logic (CCB) that can or cannot disable the clocks. The clocks to the message buffer modules can be kept running also with the module disabled to keep the the memory available for general use.

If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the CPI and MBM sub-modules
- Sets the NOT_RDY and LPM_ACK bits in MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Free Running Timer, the Error Counter Register and the Message Buffers, which cannot be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit, which will resume the clocks and negate the LPM_ACK bit.

### 20.5.10 Interrupts

The module can generate up to 8 interrupt sources (6 interrupts due to message buffers and 2 interrupts due to ORed interrupts from MBs, Bus Off, Error, Tx Warning, and Rx Warning.

The individual MB interrupts are grouped as follows:

- Groups of four interrupts (up to MB 16)
- MB16_31
- MB32_63

These are then used as the interrupt source.

Each one of the message buffers can be an interrupt source, if its corresponding mask bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the IFRL or

IFRH registers. The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the CPU writes it to ‘1’ (unless another interrupt is generated at the same time).

**NOTE**

It must be guaranteed that the CPU only clears the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

**NOTE**

Care should be taken when using bit manipulation instructions (BSET) to clear flags that have a write-to-clear mechanism, as this can cause other flags in the same register to become cleared and not get serviced. The recommended way to clear interrupt flags is to use load and store instructions with bit masks

If the Rx FIFO is enabled (MCR[FEN] bit set), the interrupts corresponding to MBs 0 to 7 have a different behavior. Bit 7 of the IFRL becomes the “FIFO Overflow” flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the “Frames Available in FIFO flag” and bits 4-0 are unused. See Section 20.4.4.12, “Interrupt Flag Register Low (IFRL)“ for more information.

A combined interrupt for all MBs is also generated by an Or of all the interrupt sources from MBs. This interrupt gets generated when any of the MBs generates an interrupt. In this case the CPU must read the IFRL and IFRH registers to determine which MB caused the interrupt.

The other 4 interrupt sources (Bus Off, Error, Tx Warning, and Rx Warning) generate interrupts like the MB ones, and can be read from the Error and Status Register. The Bus Off, Error, Tx Warning and Rx Warning interrupt mask bits are located in the Control Register.

### 20.5.11 Bus interface

The CPU access to FlexCAN registers are subject to the following rules:

- Read and write access to supervisor registers in User Mode results in access error.
- Read and write access to unimplemented or reserved address space also results in access error. Any access to unimplemented MB or Rx Individual Mask Register locations results in access error. Any access to the Rx Individual Mask Register space when the BCC bit in MCR is negated results in access error.
- If MAXMB is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that the Rx Individual Mask Registers can only be accessed in Freeze Mode, and this is still true for unused space within this memory. Note also that reserved words within RAM cannot be used. As an example, suppose FlexCAN is configured with 64 MBs and MAXMB is programmed with zero. The maximum number of MBs in this case becomes one. The MB memory starts at 0x0060, but the space from 0x0060 to 0x007F is reserved (for SMB usage), and the space from 0x0080 to 0x008F is used by

the one MB. This leaves us with the available space from 0x0090 to 0x047F. The available memory in the Mask Registers space would be from 0x0884 to 0x097F.

**NOTE**

Unused MB space must not be used as general purpose RAM while FlexCAN is transmitting and receiving CAN frames.

## 20.6 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

### 20.6.1 FlexCAN initialization sequence

The FlexCAN module may be reset in three ways:

- MCU level hard reset, which resets all memory mapped registers asynchronously
- MCU level soft reset, which resets some of the memory mapped registers synchronously (refer to Table 20-2 to see what registers are affected by soft reset)
- SOFT_RST bit in MCR, which has the same effect as the MCU level soft reset

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFT_RST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in any of the low power modes. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source (CLK_SRC bit) should be selected while the module is in Disable Mode. After the clock source is selected and the module is enabled (MDIS bit negated), FlexCAN automatically goes to Freeze Mode. In Freeze Mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled and the FRZ_ACK and NOT_RDY bits in the MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze Mode (see Section 20.5.9.1, “Freeze mode“). The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register
  - — Enable the individual filtering per MB and reception queue features by setting the BCC bit
  - — Enable the warning interrupts by setting the WRN_EN bit
  - — If required, disable frame self reception by setting the SRX_DIS bit
  - — Enable the FIFO by setting the FEN bit
  - — Enable the local priority feature by setting the LPRIO_EN bit
- Initialize the Control Register
  - — Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW

- Determine the bit rate by programming the PRESDIV field
- Determine the internal arbitration mode (LBUF bit)

- Initialize the Message Buffers
  - The Control and Status word of all Message Buffers must be initialized
  - If FIFO was enabled, the 8-entry ID table must be initialized
  - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers
- Set required interrupt mask bits in the mask registers (for all MB interrupts) and in CTRL Register (for Bus Off and Error interrupts)
- Negate the MCR[HALT] bit

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

### 20.6.2 FlexCAN Addressing and RAM size configurations

There are 3 RAM configurations that can be implemented within the FlexCAN module. The possible configurations are:

- For 16 MBs: 288 bytes for MB memory and 64 bytes for Individual Mask Registers
- For 32 MBs: 544 bytes for MB memory and 128 bytes for Individual Mask Registers
- For 64 MBs: 1056 bytes for MB memory and 256 bytes for Individual Mask Registers

In each configuration the user can program the maximum number of MBs that will take part in the matching and arbitration processes using the MAXMB field in the MCR Register. For 16 MB configuration, MAXMB can be any number between 0–15. For 32 MB configuration, MAXMB can be any number between 0–31. For 64 MB configuration, MAXMB can be any number between 0–63.

This page is intentionally left blank.

# Chapter 21
# Flash Memory

## 21.1 Introduction

This chapter presents information about the following components on this device:

- The 2 MB flash memory module
- The 2-port platform flash memory controller (PFLASH2P)

The primary function of the flash memory is to serve as electrically programmable and erasable non-volatile memory. The NVM memory can be used for instruction and data storage. The flash is a non-volatile solid-state silicon memory device consisting of blocks of single-transistor storage elements, an electrical means for selectively adding (programming) and removing (erasing) charge from these elements, and a means of selectively sensing (reading) the charge stored in these elements. The flash is addressable by word (32 bits) and page (128 bits).

The flash memory module is arranged as two main functional units. The first functional unit is the flash core (FC). The FC is composed of arrayed non-volatile storage elements, sense amplifiers, row selects, column selects, charge pumps, and redundancy logic. The arrayed storage elements in the FC are sub-divided into physically separate units referred to as blocks.

The second functional unit of the flash is the memory interface (MI). The MI contains the registers and logic which control the operation of the FC. The MI is also the interface to the platform flash memory controller (PFLASH2P).

The PFLASH2P has two AHB-Lite slave ports enabling efficient used of a single flash memory module by multiple AHB masters. Each AHB port has dedicated line buffers to support single cycle read accesses and to limit accesses to the flash array. Port0 of the PFLASH2P is dedicated for CPU accesses to the flash. Port1 of the PFLASH2P is for all other AHB master accesses to the flash

### 21.1.1 Block diagram

Figure 21-1 shows a block diagram of the flash memory module. The PFLASH2P is addressed through the system bus while the flash control and status registers are addressed through the slave (peripheral) bus.

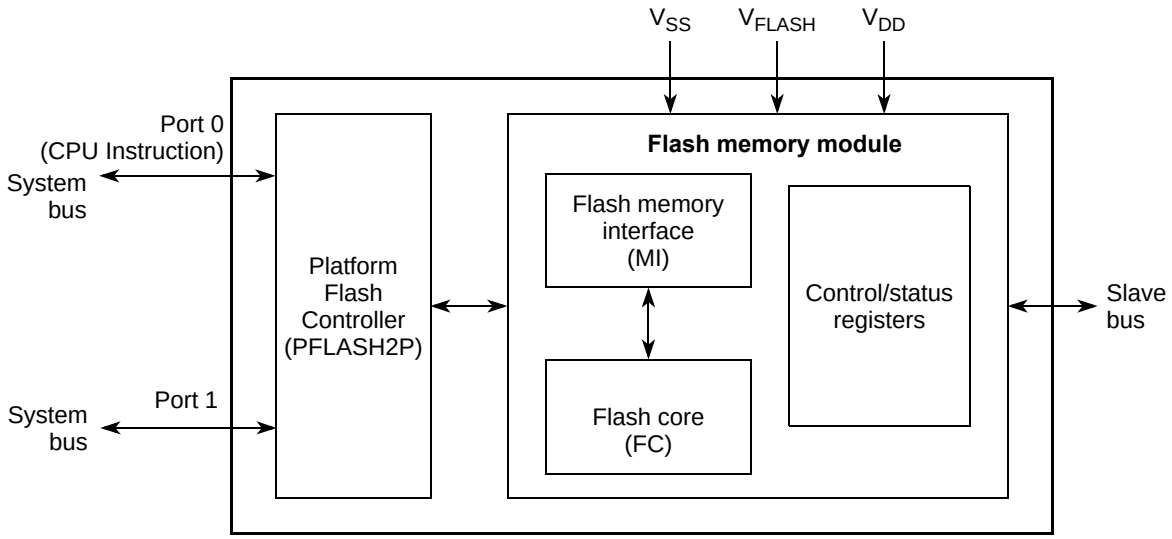



**Figure 21-1. Flash system block diagram**

## 21.1.2 Flash memory block segmentation

The flash memory core has three address spaces. The low-address space is 256 KB. The mid-address space is also 256 KB. The high-address space is 1.5 MB. The 256 KB of low memory is implemented using eight 16 KB blocks and two 64 KB blocks. The mid-address memory is implemented using two 128 KB blocks. The high memory is implemented using three 512 KB blocks.

Figure 21-2 shows the segmentation for the flash memory on MPC5645S.

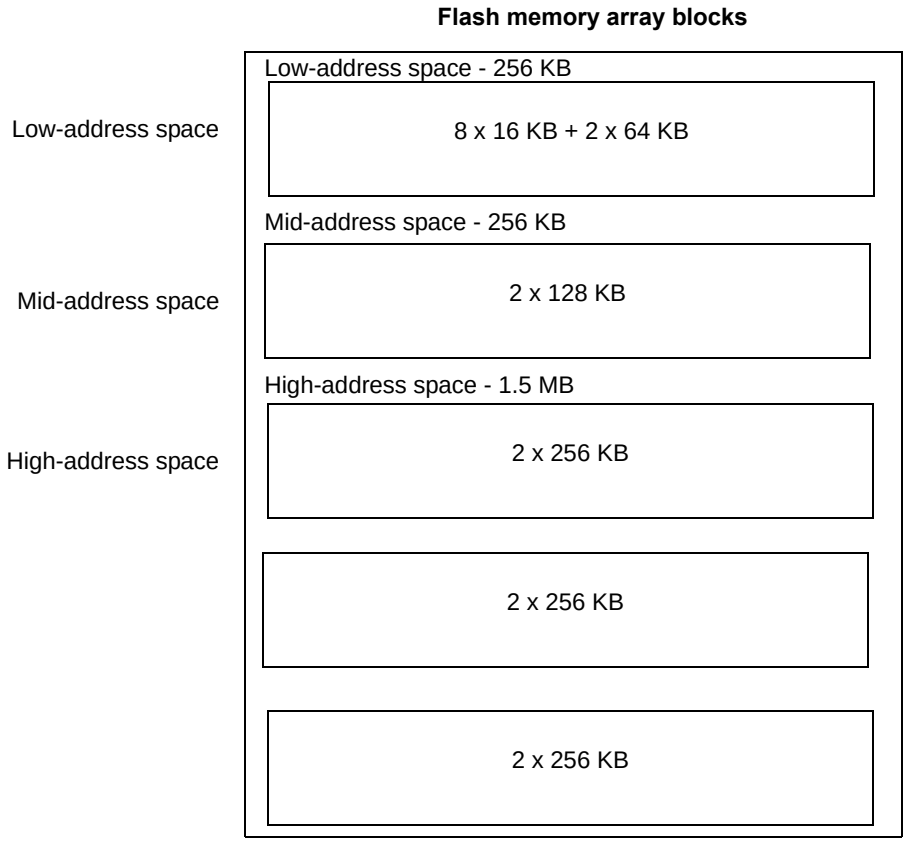



**Figure 21-2. Flash memory segmentation**

### 21.1.3 Features

The flash memory has these major features:

- 2MB of flash memory configured with 8 x 16KB, 2 x 64kB, 2 x 128KB and 6 x 256KB blocks.
- Support for a 64-bit data bus for instruction fetch, CPU data, DMA and Display Controller (DCU accesses).
- Byte, halfword, word and doubleword reads are supported. Only aligned word and doubleword writes are supported.
- Configurable read buffering and line prefetch support. Two sets of four line read buffers (128 bits wide) and a prefetch controller are used to support single-cycle read responses for hits in the buffers.
- Hardware and software configurable read and write access protections on a per-master basis.
- Interface to the flash array controller is pipelined with a depth of 1, allowing overlapped accesses to proceed in parallel for interleaved or pipelined flash array designs.
- Configurable access timing allowing use in a wide range of system frequencies.
- Multiple-mapping support and mapping-based block access timing (0-31 additional cycles) allowing use for emulation of other memory types.

- Software programmable block program/erase restriction control for low, mid and high address spaces.
- Erase of selected block(s).
- Read page size of 128 bits (4 words).
- ECC with single-bit correction, double-bit detection.
- Minimum program size is 2 consecutive 32 bit words, aligned on a 0-modulo-8 byte address, due to ECC.
- Embedded hardware program and erase algorithm.
- Read While Write (RWW) with multiple partitions.
- Sleep mode for low power stand-by.
- Erase suspend, program suspend and erase-suspended program.
- Automotive flash which meets automotive endurance and reliability requirements.
- Shadow information stored in non-volatile shadow block.

### 21.1.4 Modes of operation

The Flash module supports the following modes of operation:

#### 21.1.4.1 Flash User Mode

User mode is the default operating mode of the flash module. In this mode, it is possible to read and write, program and erase the flash module.

#### 21.1.4.2 Low Power Mode

In Low Power mode the flash memory module turns off most current sources, although logic/charge pumps to enable quick recovery to read are enabled for faster wake up time than Power Down mode.

#### 21.1.4.3 Power Down Mode

In Power Down mode, the flash module turns off all DC current sources and no reads from or writes to the module and registers are possible. All power dissipation is due to leakage in this mode.

#### 21.1.4.4 User Test Mode (UTest)

User test mode (UTest) provides a limited set of tests to end users.

## 21.2 External signal description

There are no external signals for the flash module but it should be noted that the VDDE_B I/O supply is shared with the flash module. See Chapter 3, “Signal Description.”

## 21.3 Memory map and registers

This section provides a detailed description of all flash memory and PFLASH2P registers.

### 21.3.1 Module Memory Map

The flash memory map is shown in Table 21-1. The addresses are given as an offset to the flash memory base address.

The flash register memory map is shown in Table 21-2. There are no program-visible registers that physically reside inside the flash. The flash receives control and configuration information from the flash array controller to determine operating configurations. These are part of the flash array controller's configuration registers mapped into the IPS address space but are described herein. These registers should only be referenced with 32-bit accesses. Also included in the flash memory map is a block that contains non-volatile configuration values that are used to initialize certain flash and SoC features. This is known as the Shadow flash and is included in this table for completeness but is not intended for storage of user program or data.

**Table 21-1. Flash Memory Map**

| Offset from FLASH_BASE (0x0000_0000) | Use | Block[1] | Partition |
|---|---|---|---|
| 0x0000_0000 | Low-address space | L0 | 1 |
| 0x0000_4000 | | L1 | |
| 0x0000_8000 | | L2 | |
| 0x0000_C000 | | L3 | |
| 0x0001_0000 | | L4 | 2 |
| 0x0001_4000 | | L5 | |
| 0x0001_8000 | | L6 | |
| 0x0001_C000 | | L7 | |
| 0x0002_0000 | | L8 | 3 |
| 0x0003_0000 | | L9 | |
| 0x0004_0000 | Mid-address space | M0 | 4 |
| 0x0006_0000 | | M1 | |
| 0x0008_0000 | High-address space | H0 | 5 |
| 0x000C_0000 | | H1 | |
| 0x0010_0000 | | H2 | 6 |
| 0x0014_0000 | | H3 | |
| 0x0018_0000 | | H4 | 7 |
| 0x001C_0000 | | H5 | |
| 0x0020_0000–0x00FF_BFFF | Reserved | | |

**Table 21-1. Flash Memory Map (continued)**

| Offset from FLASH_BASE (0x0000_0000) | Use | Block[1] | Partition |
|---|---|---|---|
| 0x00FF_C000–0x00FF_FDD7 | General use | S | All[2] |
| 0x00FF_FDD8 | Serial passcode (0xFEED_FACE_CAFE_BEEF) | | |
| 0x00FF_FDE0 | Censorship control word (0x55AA_55AA) | | |
| 0x00FF_FDE4 | General use | | |
| 0x00FF_FDE8 | LML reset configuration (0x0010_0000) | | |
| 0x00FF_FDEC | General use | | |
| 0x00FF_FDF0 | HBL reset configuration (0x0FFF_FFFF) | | |
| 0x00FF_FDF4 | General use | | |
| 0x00FF_FDF8 | SLL reset configuration (0x000F_FFFF) | | |
| 0x00FF_FDFC | General use | | |
| 0x00FF_FE00 | PFAPR reset configuration | | |
| 0x00FF_FE04 | General use | | |
| 0x00FF_FE08 | PFSACC reset configuration | | |
| 0x00FF_FE0C | General use | | |
| 0x00FF_FE10 | NVUSRO Register | | |
| 0x00FF_FE20 - 0x00FF_FFFF | General use | | |

1 L*n* = Low Address Space, M*n* = Mid Address Space, H*n* = High Address Space, S = Shadow Block.

2 For read while write operations, the shadow row behaves as if it is in all partitions.

**Table 21-2. Flash memory configuration register memory map**

| Offset from FLASH_REGS_BASE (0xC3F8_8000) | Register | Location |
|---|---|---|
| 0x0000 | MCR—Module configuration register | on page 823 |
| 0x0004 | LML—Low-/Mid-address space block locking register | on page 827 |
| 0x0008 | HBL—High-address space block locking register | on page 828 |
| 0x000C | SLL—Secondary low-/mid-address space block locking register | on page 829 |
| 0x0010 | LMS—Low-/mid-address space block select register | on page 830 |
| 0x0014 | HBS—High-address space block select register | on page 831 |
| 0x0018 | ADR—Address register | on page 832 |
| 0x001C | PFCRP0—Platform flash configuration register for port 0 | on page 833 |
| 0x0020 | PFCRP1—Platform flash configuration register for port 1 | on page 833 |
| 0x0024 | PFAPR—Platform flash access protection register | on page 836 |
| 0x0028 | PFSACC—Platform flash supervisor access control register | on page 837 |
| 0x002C | PFDACC—Platform flash data access control register | on page 839 |

Table 21-2. Flash memory configuration register memory map (continued)

| Offset from FLASH_REGS_BASE (0xC3F8_8000) | Register | Location |
|---|---|---|
| 0x0030 – 0x0038 | Reserved | |
| 0x003C | UT0—UTest register 0 | on page 839 |
| 0x0040 | UT0—UTest register 1 | on page 841 |
| 0x0044 | UT0—UTest register 2 | on page 842 |
| 0x0048 | UM0—User multiple input signature register 0 | on page 842 |
| 0x004C | UM1—User multiple input signature register 1 | on page 842 |
| 0x0050 | UM2—User multiple input signature register 2 | on page 842 |
| 0x0054 | UM3—User multiple input signature register 3 | on page 842 |
| 0x0058 | UM4—User multiple input signature register 4 | on page 842 |
| 0x0048 – 0x3FFF | Reserved | |

## 21.3.2 Register descriptions

This section lists the flash memory registers in address order and describes the registers and their bit fields.

### 21.3.2.1 Module Configuration Register (MCR)

The MCR register is shown in Figure 21-3 and Table 21-3.

Offset: FLASH_REGS_BASE + 0x0000 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | SIZE | | | 0 | LAS | | | 0 | 0 | 0 | MAS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EER | RWE | SBC | 0 | PEAS | DONE | PEG | 0 | 0 | 0 | 0 | PGM | PSUS | ERS | ESUS | EHV |
| W | w1c | w1c | w1c | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 21-3. Module Configuration Register (MCR)

**Table 21-3. MCR Field Descriptions**

| Field | Description |
|---|---|
| SIZE | Array Space Size. The value of the SIZE field depends on the size of the flash module. For MPC5645S, this bit field is 0b101, indicating a 2.0 MB array size (with 1.5 MB in high-address space). SIZE is read only. |
| LAS | Low Address Space. The value of the LAS field corresponds to the configuration of the Low Address Space. For MPC5668x, this bit field is 0b100, indicating eight 16 KB blocks and two 64 KB blocks. LAS is read only. |
| MAS | Mid Address Space. The value of the MAS field corresponds to the configuration of the Mid Address Space. The value of the MAS field depends on the size of the flash module. For MPC5645S, this bit field is 0b0, indicating two 128 KB blocks. MAS is read only. |
| EER | ECC Event Error. EER provides information on previous reads. If a double bit detection occurred, the EER bit is set to a 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. This bit may not be set by the user. In the event of a single bit detection and correction, this bit is not be set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0 Reads are occurring normally.<br>1 An ECC Error occurred during a previous read. |
| RWE | Read While Write Event Error. RWE provides information on previous RWW reads. If a Read While Write error occurs, this bit is set to 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. This bit may not be written to a 1 by the user. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last reset, or clearing of RWE) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0 Reads are occurring normally.<br>1 A Read While Write Error occurred during a previous read. |
| SBC | Single Bit Correction. SBC provides information on previous reads provided the UT0[SPCE] is set. If a single bit correction occurred, the SBC bit is set to a 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. If SBC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of SBC) did not require a correction. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0 Reads are occurring without corrections.<br>1 A Single Bit Correction occurred during a previous read. |
| PEAS | Program/Erase Access Space. PEAS is used to indicate which space is valid for program and erase operations, either main array space or shadow space. PEAS = 0 indicates that the main address space is active for all FC program and erase operations. PEAS = 1 indicates the shadow address space is active for program/erase. The value in PEAS is captured and held when the shadow block is enabled with the first interlock write done for program or erase operations. The value of PEAS is retained between sampling events (i.e.,subsequent first interlock writes). The value in PEAS may be changed during erase-suspended program, and reverts back to its original state once the erase-suspended program is completed. PEAS is read only.<br>0 Shadow address space is disabled for program/erase and main address space enabled.<br>1 Shadow address space is enabled for program/erase and main address space disabled. |
| DONE | State Machine Status. Indicates if the flash module is performing a high-voltage operation. DONE is set to a 1 on termination of the flash module reset, at the end of program and erase high-voltage sequences and after a successful abort of a high voltage operation. DONE is cleared upon commencement of a high voltage operation or on the resumption of a suspended operation.<br>0 Flash is executing a high-voltage operation<br>1 Flash is not executing a high-voltage operation |

**Table 21-3. MCR Field Descriptions (continued)**

| Field | Description |
|---|---|
| PEG | Program/Erase Good. The PEG bit indicates the completion status of the last flash program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation causes PEG to be cleared, indicating the sequence failed. PEG is set to a 1 when the module is reset. PEG is read only.<br>The value of PEG is valid only when PGM = 1 and/or ERS = 1 and after DONE transitions from 0 to 1 due to an abort or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by PSUS or ESUS being set to logic 1. If PGM and ERS are both 1 when DONE makes a qualifying 0 to 1 transition the value of PEG indicates the completion status of the PGM sequence. This happens in an erase-suspended program operation.<br>0 Program or erase operation failed.<br>1 Program or erase operation successful.<br><br>**Note:** If program or erases are attempted on blocks that are locked, the response from flash is PEG = 1, indicating that the operation was successful, and the contents of the block are properly protected from the program or erase operation. |
| PGM | Program. PGM is used to set up flash for a program operation. A 0 to 1 transition of PGM initiates a program sequence. A 1 to 0 transition of PGM ends the program sequence. PGM can be set only under one of the following conditions:<br>• User mode read (ERS is low and UTE is low).<br>• Erase suspend (ERS and ESUS are 1) with EHV low.<br>PGM can be cleared by the user only when PSUS and EHV are low and DONE is high. PGM is cleared on reset.<br>0 Flash is not executing a program sequence.<br>1 Flash is executing a program sequence.<br><br>**Note:** In an erase-suspended program, programming Flash locations in blocks which were being operated on in the erase may corrupt FC data. This should be avoided due to reliability implications. |
| PSUS | Program Suspend. PSUS is used to indicate the flash module is in program suspend or in the process of entering a suspend state. The module is in program suspend when PSUS = 1 and DONE = 1. PSUS can be set high only when PGM and EHV are high. A 0 to 1 transition of PSUS starts the sequence which sets DONE and places the flash module in program suspend.<br>PSUS can be cleared only when DONE and EHV are high. A 1 to 0 transition of PSUS with EHV = 1 starts the sequence which clears DONE and returns the flash module to program. The module cannot exit program suspend and clear DONE while EHV is low. PSUS is cleared on reset.<br>0 Program sequence is not suspended.<br>1 Program sequence is suspended. |
| ERS | Erase. ERS is used to set up flash for an erase operation. A 0 to 1 transition of ERS initiates an erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can only be set only in user mode read (PGM is low and UTE is low). ERS can be cleared by the user only when ESUS and EHV are low and DONE is high. ERS is cleared on reset.<br>0 Flash is not executing an erase sequence.<br>1 Flash is executing an erase sequence. |

**Table 21-3. MCR Field Descriptions (continued)**

| Field | Description |
|---|---|
| ESUS | Erase Suspend. ESUS is used to indicate that the flash module is in erase suspend or in the process of entering a suspend state. The module is in erase suspend when ESUS = 1 and DONE = 1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the flash in erase suspend.<br>ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV = 1 starts the sequence which clears DONE and returns the module to erase. The flash module cannot exit erase suspend and clear DONE while EHV is low. ESUS is cleared on reset.<br>0 Erase sequence is not suspended.<br>1 Erase sequence is suspended. |
| EHV | Enable High Voltage. The EHV bit enables the flash module for a high voltage program/erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a program/erase sequence. EHV may be set, initiating a program/erase, after an interlock under one of the following conditions:<br>• Erase (ERS = 1, ESUS = 0).<br>• Program (ERS = 0, ESUS = 0, PGM = 1, PSUS = 0).<br>• Erase-suspended program (ERS = 1, ESUS = 1, PGM = 1, PSUS = 0).<br>If a program operation is to be initiated while an erase is suspended the user must clear EHV while in erase suspend before setting PGM.<br>In normal operation, a 1 to 0 transition of EHV with DONE high, PSUS and ESUS low terminates the current program/erase high voltage operation.<br>When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the suspend bit for the current program/erase sequence low. An abort causes the value of PEG to be cleared, indicating a failed program/erase; address locations being operated on by the aborted operation contain indeterminate data after an abort.<br>A suspended operation cannot be aborted. EHV may be written during suspend. EHV must be high for the flash module to exit suspend. EHV may not be written after a suspend bit is set high and before DONE transitions high. EHV may not be set low after the current suspend bit is set low and before DONE transitions low.<br>0 Flash is not enabled to perform a high voltage operation.<br>1 Flash is enabled to perform a high voltage operation.<br><br>**Note:** Aborting a high voltage operation leaves FC addresses in an indeterminate data state. This may be recovered by executing an erase on the affected blocks. |

#### 21.3.2.1.1 MCR Simultaneous Register Writes

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit by bit basis in the preceding section. The write locks detailed in the previous section do not consider the effects of trying to write two or more bits simultaneously. The effects of writing bits simultaneously which put the module in an illegal state are detailed here.

The flash module does not allow the user to write bits simultaneously which put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in Table 21-4.

**Table 21-4. MCR Bit Set/Clear Priority Levels**

| Priority Level | MCR Bit(s) |
|---|---|
| 1 | ERS |
| 2 | PGM |
| 3 | EHV |
| 4 | ESUS, PSUS |

If the user attempts to write two or more MCR bits simultaneously then only the bit with the lowest priority level is written. Setting two bits with the same priority level is prevented by existing write locks or do not put the flash in an illegal state.

For example, setting ERS and PGM simultaneously results in only ERS being set. Attempting to clear EHV while setting PSUS results in EHV being cleared, while PSUS is unaffected.

### 21.3.2.2 Low/Mid Address Space Block Locking Register (LML)

The Low/Mid Address Block Locking Register (LML) provides a means to protect blocks from being modified. These bits, along with bits in the Secondary LLOCK (SLL), determine if the block is locked from program or erase. An “OR” of LML and SLL determine the final lock status.

**NOTE**

A reset value of 1* in Figure 21-4 indicates that the reset value of these registers is determined by Flash values in the shadow block. An erased shadow block causes the reset value to be 1.

The LML register is shown in Figure 21-4 and Table 21-5.

Offset: FLASH_REGS_BASE + 0x0004 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LME | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SLOCK | 0 | 0 | MLOCK | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1* | 0 | 0 | 1* | 1* |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | LLOCK | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* |

**Figure 21-4. Low/Mid Address Block Locking Register (LML)**

**Table 21-5. LML Field Descriptions**

| Field | Description |
|---|---|
| LME | Low/Mid Address Lock Enable. This bit is used to enable the Lock registers (SLOCK, MLOCK and LLOCK) to be set or cleared by register writes. This bit is a status bit only, and may not be written or cleared, and the reset value is 0. The method to set this bit is to write a password, and if the password matches, the LME bit is set to reflect the status of enabled, and is enabled until a reset operation occurs. For LME, the password 0xA1A1_1111 must be written to the LML register.<br>0 Low/Mid Address Locks are disabled, and cannot be modified.<br>1 Low/Mid Address Locks are enabled to be written. |
| SLOCK | Shadow Lock. This bit is used to lock the shadow block from programs and erases. The SLOCK register is not writable once an interlock write is completed until MCR[DONE] is set at the completion of the requested operation. Likewise, SLOCK register is not writable if a high voltage operation is suspended. SLOCK is also not writeable during UTest operations, when AIE is high.<br>Upon reset, information from the shadow block is loaded into the SLOCK register. The SLOCK bit may be written as a register. Reset causes the bits to go back to their shadow block value. The default value of the SLOCK bits (assuming erased shadow location) is locked. SLOCK is not writable unless LME is high.<br>0 The shadow block can receive program and erase pulses.<br>1 The shadow block is locked for program and erase. |
| MLOCK[1:0] | Mid Address Space Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for Mid Address Space starts with MLOCK[0] and continues until all blocks are accounted.<br><br>The lock register is not writable once an interlock write is completed until MCR[DONE] is set at the completion of the requested operation. Likewise, the lock register is not writable if a high voltage operation is suspended. MLOCK is also not writeable during UTest operations, when AIE is high.<br>Upon reset, information from the shadow block is loaded into the block registers. The LOCK bits may be written as a register. Reset causes the bits to go back to their shadow block value. The default value of the LOCK bits (assuming erased shadow location) is locked.<br>In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1 (independent of the shadow block), and register writes have no effect.<br><br>MLOCK is not writable unless LME is high. |
| LLOCK[9:0] | Low Address Space Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for Low Address Space starts with LLOCK[0] and continues until all blocks are accounted.<br><br>For more details on LLOCK, please see MLOCK bit description.<br><br>LLOCK is not writable unless LME is high. |

### 21.3.2.3 High Address Space Block Locking Register (HBL)

The High Address Space Block Locking Register (HBL) provides a means to protect blocks from being modified.

**NOTE**

A reset value of 1* in Figure 21-5 indicates that the reset value of these registers is determined by Flash values in the shadow block. An erased shadow block causes the reset value to be 1.

The HBL register is shown in Figure 21-5 and Table 21-6.

Offset: FLASH_REGS_BASE + 0x0008 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | HLOCK | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1* | 1* | 1* | 1* | 1* | 1* |

**Figure 21-5. High Address Space Block Locking Register (HBL)**

**Table 21-6. HBL Field Descriptions**

| Field | Description |
|---|---|
| HBE | High Address Lock Enable This bit is used to enable the Lock registers (HLOCK) to be set or cleared by register writes. This bit is a status bit only, and may not be written or cleared, and the reset value is 0. The method to set this bit is to provide a password, and if the password matches, the HBE bit is set to reflect the status of enabled, and is enabled until a reset operation occurs. For HBE, the password 0xB2B2_2222 must be written to the HBL register.<br>0 High Address Locks are disabled, and cannot be modified.<br>1 High Address Locks are enabled to be written. |
| HLOCK[5:0] | High Address Space Block Lock. HLOCK has the same characteristics as LLOCK. Please see this description for more information. The block numbering for High Address Space starts with HLOCK[0] and continues until all blocks are accounted.<br>HLOCK is not writable unless HBE is high. |

### 21.3.2.4 Secondary Low/Mid Address Space Block Locking Register (SLL)

The Secondary Low/Mid Address Block Locking Register (SLL) provides an alternative means to protect blocks from being modified. This has the effect of creating a “tiered” locking scheme to enable different flash users to provide different default locking on blocks. These bits, along with bits in the LLOCK (LML), determine if the block is locked from program or erase. An “OR” of LML and SLL determine the final lock status.

**NOTE**

A reset value of 1* in Figure 21-6 indicates that the reset value of these registers is determined by Flash values in the shadow block. An erased shadow block causes the reset value to be 1.

The SLL register is shown in Figure 21-6 and Table 21-7.

Offset: FLASH_REGS_BASE + 0x000C — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SLE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SS LOCK | 0 | 0 | SM LOCK | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1* | 0 | 0 | 1* | 1* |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | SLLOCK | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* | 1* |

**Figure 21-6. Secondary Low/Mid Address Block Locking Register (SLL)**

**Table 21-7. SLL Field Descriptions**

| Field | Description |
|---|---|
| SLE | Secondary Low/Mid Address Lock Enable. This bit is used to enable the Lock registers (SSLOCK, SMLOCK, and SLLOCK) to be set or cleared by register writes. This bit is a status bit only, and may not be written or cleared, and the reset value is 0. The method to set this bit is to provide a password, and if the password matches, the SLE bit is set to reflect the status of enabled, and is enabled until a reset operation occurs. For SLE, the password 0xC3C3_3333 must be written to the SLL register<br>0 Secondary Low/Mid Address Locks are disabled, and cannot be modified.<br>1 Secondary Low/Mid Address Locks are enabled to be written. |
| SSLOCK | Secondary Shadow Lock. This bit is an alternative method that may be used to lock the shadow block from programs and erases. SSLOCK has the same description as SLOCK. SSLOCK is not writable unless SLE is high. |
| SMLOCK[1:0] | Secondary Mid Address Block Lock. This bit is an alternative method that may be used to lock the Mid Address Space blocks from programs and erases. SMLOCK has the same description as MLOCK. SMLOCK is not writable unless SLE is high. |
| SLLOCK[9:0] | Secondary Low Address Block Lock. This bit is an alternative method that may be used to lock the Low Address Space blocks from programs and erases. SLLOCK has the same description as LLOCK. SLLOCK is not writable unless SLE is high. |

### 21.3.2.5 Low/Mid Address Space Block Select Register (LMS)

The Low/Mid Address Space Block Select Register (LMS) provides a means to select blocks to be operated on during erase.

The LMS register is shown in Figure 21-7 and Table 21-8.

Offset: FLASH_REGS_BASE + 0x0010 Access: User read/write

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

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

**Figure 21-7. Low/Mid Address Space Block Select Register (LMS)**

**Table 21-8. LMS Field Descriptions**

| Field | Description |
|---|---|
| MSEL[1:0] | Mid Address Space Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or un-selected.<br>The blocks must be selected (or un-selected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until MCR[DONE] is set at the completion of the requested operation, or if a high voltage operation is suspended. MSEL is also not writeable during UTest operations, when AIE is high.<br>In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to un-selected, and are not writable. The reset value is always 0, and register writes have no effect. |
| LSEL[9:0] | Low Address Space Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or un-selected.<br>The blocks must be selected (or un-selected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until MCR[DONE] is set at the completion of the requested operation, or if a high voltage operation is suspended. LSEL is also not writeable during UTest operations, when AIE is high.<br>In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to un-selected, and are not writable. The reset value is always 0, and register writes have no effect. |

### 21.3.2.6 High Address Space Block Select Register (HBS)

The High Address Space Block Select Register (HBS) provides a means to select blocks to be operated on during erase.

The HBS register is shown in Figure 21-8 and Table 21-9.

Offset: FLASH_REGS_BASE + 0x0014 Access: User read/write

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

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

**Figure 21-8. High Address Space Block Select Register (HBS)**

**Table 21-9. HBS Field Descriptions**

| Field | Description |
|---|---|
| HSEL[5:0] | High Address Space Block Select. High Address Block Select has the same characteristics as LSEL. |

### 21.3.2.7 Address Register (ADR)

The Address register (ADR) provides the first failing address in the event module failures (ECC or PGM/Erase state machine)

The ADR register is shown in Figure 21-9 and Table 21-10.

Offset: FLASH_REGS_BASE + 0x0018 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SAD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ADDR | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 21-9. Address Register (ADR)**

**Table 21-10. ADR Field Descriptions**

| Field | Description |
|---|---|
| SAD | Shadow Address. The SAD bit qualifies the address captured during an ECC Event Error, Single Bit Correction, or State Machine operation.<br>The SAD register is not writable.<br>0 Address Captured is from Main Array Space.<br>1 Address Captured is from Shadow Array Space. |
| ADDR[20:3] | Address. The ADR register provides the first failing address in the event of ECC event error (MCR[EER] set), single bit correction (MCR[SBC] set), as well as providing the address of a failure that may have occurred in a state machine operation (MCR[PEG] cleared). ECC event errors take priority over single bit corrections, which take priority over state machine errors. This is especially valuable in the event of a RWW operation, where the read senses an ECC error or single bit correction, and the state machine fails simultaneously. This address is always a Double Word address that selects 64 bits.<br>The ADR register is writable, and can be used in the UTest ECC Logic Check. If the ECC logic check is enabled (UT0[EIE] = 1) then the ADR register will not update for ECC event error, single bit correction, or state machine errors.<br>If MCR[EER] or MCR[SBC] are set, the ADR register is locked from writing. MCR[PEG] does not affect the writability of the ADR register. |

### 21.3.2.8 Platform Flash Configuration Registers (PFCRP0 and PFCRP1)

The PFLASH configuration register for port 0 (PFCRP0) is used to specify operation of port p0 of the flash memory module. This register also has two bits (ARB and PRI) to control arbitration between the p0/p1 ports.

The PFLASH configuration register for port 1 (PFCRP1) is used to specify operation of port p1 of the flash memory module.

The PFCRP*n* registers are shown in Figure 21-10, Figure 21-11, and Table 21-11.

Master ID mapping can be found in Table 9-1.

Offset: FLASH_REGS_BASE + 0x001C — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LBCFG | | | | ARB | PRI | 0 | 0 | M7PFE | M6PFE | M5PFE | M4PFE | M3PFE | M2PFE | M1PFE | M0PFE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | APC | | | WWSC | | RWSC | | | 0 | DPFEN | 0 | IPFEN | 0 | PFLIM | | BFEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |

**Figure 21-10. Platform Flash Configuration Register for Port 0 (PFCRP0)**

Offset: FLASH_REGS_BASE + 0x0020 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LBCFG | | | | 0 | 0 | 0 | 0 | M7PFE | M6PFE | M5PFE | M4PFE | M3PFE | M2PFE | M1PFE | M0PFE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | APC | | | WWSC | | RWSC | | | 0 | DPFEN | 0 | IPFEN | 0 | PFLIM | | BFEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |

**Figure 21-11. Platform Flash Configuration Register for Port 1 (PFCRP1)**

**Table 21-11. PFCRP0 and PFCRP1 field descriptions**

| Field | Description |
|---|---|
| LBCFG | Line Buffer Configuration. Controls the configuration of the four line buffers in the PFLASH controller. The buffers can be organized as a pool of available resources or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, it is allocated to the least recently used buffer within the group and the just-fetched entry then marked as most recently used. If the flash access is for the next sequential line, the buffer is not marked as most recently used until the given address produces a buffer hit.<br>For PFCRP0, this field is set to 0b0000 by hardware reset. For PFCRP1, this field is set to 0b0011 by hardware reset.<br>xx00 All four buffers are available for any flash access, i.e., there is no partitioning of the buffers based on the access type.<br>xx01 Reserved.<br>xx10 The buffers are partitioned into two groups: buffers 0 and 1 allocated for instruction fetches and buffers 2 and 3 for data accesses.<br>xx11 The buffers are partitioned into two groups: buffers 0,1, 2 allocated for instruction fetches and buffer 3 for data accesses. |
| ARB | Arbitration Mode. This field controls which arbitration mode is used. In both fixed priority or round-robin modes, write requests are prioritized higher than read requests, and read requests are prioritized higher than speculative prefetch requests whenever both ports issue concurrent requests. This bit is set to 1 by hardware reset.<br>0 Fixed-priority arbitration is used; the port specified in PRI has highest fixed priority.<br>1 Round-robin arbitration is used.<br>**Note:** This bit is only available in PFCRP0. For PFCRP1, treat this bit as reserved with a reset value of 0. |
| PRI | Fixed Priority. Controls which port has highest fixed priority when fixed priority arbitration is selected. This field has no effect when operating in round-robin mode. This bit is cleared by hardware reset.<br>0 Port p0 is given highest fixed priority.<br>1 Port p1 is given highest fixed priority.<br>**Note:** This bit is only available in PFCRP0. For PFCRP1, treat this bit as reserved with a reset value of 0. |
| M*n*PFE<br>*n* = 0:2, 4:6 | Master *n* Prefetch Enable. Used to control whether prefetching may be triggered based on the AHB hmaster attribute. For example, M0PFE enables prefetching for accesses where hmaster[3:0] = 0b0000. Likewise, M4PFE enables prefetching only when hmaster[3:0] == 0b0100. Note that hmaster[3] is ignored when determining which M*n*PFE to use for a given access. These bits are cleared by hardware reset.<br>0 No prefetching may be triggered by this master.<br>1 Prefetching may be triggered by this master.<br>**Note:** These bits refer to the master ID, not the AMBA port number. Please refer to Table 9-1 for details. |

**Table 21-11. PFCRP0 and PFCRP1 field descriptions (continued)**

| Field | Description |
|---|---|
| APC[2:0] | Address Pipelining Control. Used to control the number of cycles between pipelined access requests.<br>This field must be set to a value corresponding to the operating frequency of the PFLASH. The settings are documented in the *MPC5645S Microcontroller Data Sheet*. Higher operating frequencies require non-zero settings for this field for proper flash operation.<br>000 Accesses may be pipelined back-to-back.<br>001 Access requests require one additional hold cycle.<br>010 Access requests require two additional hold cycles.<br>...<br>110 Access requests require six additional hold cycles.<br>111 No address pipelining.<br><br>**Note:** The settings for APC and RWSC should be the same. |
| WWSC[1:0] | Write Wait State Control. Used to control the number of wait states to be added to the best case flash array access time for writes. This field must be set to a value corresponding to the operating frequency of the PFLASH. Higher operating frequencies require non-zero settings for this field for proper flash operation. This field is set to 0b11 by hardware reset.<br>00 No additional wait-states are added.<br>01 One additional wait-state is added.<br>10 Two additional wait-states are added.<br>11 Three additional wait-states are added. |
| RWSC[2:0] | Read Wait State Control. Used to control the number of wait states to be added to the best case flash array access time for reads. This field must be set to a value corresponding to the operating frequency of the PFLASH and the actual read access time of the PFLASH. This field is set to 0b111 by hardware reset.<br>000 No additional wait states are added.<br>001 One additional wait state is added.<br>...<br>111 Seven additional wait states are added.<br><br>**Note:** The settings for APC and RWSC should be the same. |
| DPFEN | Data Prefetch Enable. Enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset.<br>0 No prefetching is triggered by a data read access.<br>1 Prefetching may be triggered by any data read access. |
| IPFEN | Instruction Prefetch Enable. Enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset.<br>0 No prefetching is triggered by an instruction read access.<br>1 Prefetching may be triggered by any instruction read access. |

**Table 21-11. PFCRP0 and PFCRP1 field descriptions (continued)**

| Field | Description |
|---|---|
| PFLIM[1:0] | PFLASH Prefetch Limit. Controls the prefetch algorithm used by the PFLASH prefetch controller. This field defines a limit on the maximum number of sequential prefetches that are attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset.<br>00 No prefetching or buffering is performed.<br>01 The referenced line is prefetched on a buffer miss, i.e., prefetch on miss.<br>1*x* the referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), i.e., prefetch on miss or hit. |
| BFEN | PFLASH Line Read Buffers Enable. Enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit is cleared by hardware reset.<br>0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared.<br>1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled. |

## 21.3.2.9 Platform Flash Access Protection Register (PFAPR)

Offset: FLASH_REGS_BASE + 0x0024 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | M7AP | | M6AP | | M5AP | | M4AP | | M3AP | | M2AP | | M1AP | | M0AP | |
| Reset | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SHSACC | | | | 0 | 0 | 0 | 0 | SHDACC | | | | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | * | * | * | * | 0 | 0 | 0 | 0 | * | * | * | * | 0 | 0 | 0 | 0 |

* = Initialized by hardware reset

**Figure 21-12. PFlash Access Protection Register (PFAPR)**

**Table 21-12. PFAPR field descriptions**

| Field | Description |
|---|---|
| M*n*AP | Master *n* Access Protection. These fields are used to control whether read and write accesses to the flash memory are allowed based on the master ID of a requesting master.<br>00 No accesses may be performed by this master.<br>01 Only read accesses may be performed by this master.<br>10 Only write accesses may be performed by this master.<br>11 Both read and write accesses may be performed by this master.<br>**Note:** For a list of master IDs, see the M*n*PFE field description in Table 21-11. |

**Table 21-12. PFAPR field descriptions (continued)**

| Field | Description |
|---|---|
| SHSACC[7:4] | Shadow Block Supervisor Access Control. This bit field defines supervisor/user mode access control for each 4 KB sector within the shadow block region of the flash array.<br>0 Shadow block sector *n* can be accessed in both user and supervisor mode.<br>1 Shadow block sector *n* can be accessed only in supervisor mode. An attempted user mode access is terminated with an AHB error response. If the requesting bus master is the processor core, the ERROR response typically generates an instruction abort or data abort exception.<br><br>This field is mapped into the shadow block (shadow_block = 0x00FF_C000) with sector base addresses of:<br>SHSACC[4] = shadow_block + 0x0000<br>SHSACC[5] = shadow_block + 0x1000<br>SHSACC[6] = shadow_block + 0x2000<br>SHSACC[7] = shadow_block + 0x3000<br><br>This field is initialized by hardware reset to the value contained in address 0x3E00 of the shadow block of the flash array. An erased or unprogrammed flash sets this field to 0xFF. The contents of the PFAPR are combined with the SHSACC field to determine the final flash attributes. |
| SHDACC[7:4] | Shadow Block Data Access Control. This bit field defines code/data access control for each 4 KByte sector within the shadow block region of the flash array.<br>0 Shadow block sector *n* can only be accessed as data. An attempted instruction fetch access is terminated with an AHB error response. If the requesting bus master is the processor core, the ERROR response typically generates an instruction abort or data abort exception.<br>1 Shadow block sector *n* can be accessed as either code or data.<br><br>This field is mapped into the shadow block using the same definition as the SHSACC field above.<br><br>This field is initialized by hardware reset to the value contained in address 0x3E00 of the shadow block of the flash array. An erased or unprogrammed flash sets this field to 0xFF.<br><br>The contents of the PFAPR are combined with the SHDACC field to determine the final flash attributes. |

## 21.3.2.10 PFlash Supervisor Access Control Register (PFSACC)

Offset: FLASH_REGS_BASE + 0x0028 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | SACC[30:16] | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SACC[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |

* = Initialized by hardware reset

**Figure 21-13. PFlash Supervisor Access Control Register (PFSACC)**

**Table 21-13. PFSACC field descriptions**

| Field | Description |
|---|---|
| SACC | Supervisor Access Control. This bit field defines supervisor/user mode access control for each sector within the main flash array.<br>0 Flash array sector *n* can be accessed in both user and supervisor mode.<br>1 Flash array sector *n* can be accessed only in supervisor mode. An attempted user mode access is terminated with an AHB error response. If the requesting bus master is the processor core, the ERROR response typically generates an instruction abort or data abort exception.<br>The mapping of this bit field to the main flash array is defined in **Table 21-14**<br>This field is initialized by hardware reset to the value contained in address 0x3E08 of the shadow block of the flash array. An erased or unprogrammed flash sets this field to 0xFFFF_FFFF. |

**Table 21-14. {S,D}ACC Register to Flash Array Mapping**

| Register Bit | Starting Flash Array Address | Sector Size |
|---|---|---|
| xACC[0] | 0x00_0000 | 16 KB |
| xACC[1] | 0x00_4000 | 16 KB |
| xACC[2] | 0x00_8000 | 16 KB |
| xACC[3] | 0x00_C000 | 16 KB |
| xACC[4] | 0x01_0000 | 16 KB |
| xACC[5] | 0x01_4000 | 16 KB |
| xACC[6] | 0x01_8000 | 16 KB |
| xACC[7] | 0x01_C000 | 16 KB |
| xACC[8] | 0x02_0000 | 16 KB |
| xACC[9] | 0x02_4000 | 16 KB |
| xACC[10] | 0x02_8000 | 16 KB |
| xACC[11] | 0x02_C000 | 16 KB |
| xACC[12] | 0x03_0000 | 16 KB |
| xACC[13] | 0x03_4000 | 16 KB |
| xACC[14] | 0x03_8000 | 16 KB |
| xACC[15] | 0x03_C000 | 16 KB |
| xACC[16] | 0x04_0000 | 256 KB |
| xACC[17] | 0x08_0000 | 256 KB |
| xACC[18] | 0x0C_0000 | 256 KB |
| xACC[19] | 0x10_0000 | 256 KB |
| xACC[20] | 0x14_0000 | 256 KB |
| xACC[21] | 0x18_0000 | 256 KB |
| xACC[22] | 0x1C_0000 | 256 KB |
| xACC[23] | 0x20_0000 | 256 KB |
| xACC[24] | 0x24_0000 | 256 KB |
| xACC[25] | 0x28_0000 | 256 KB |
| xACC[26] | 0x2C_0000 | 256 KB |
| xACC[27] | 0x30_0000 | 256 KB |

**Table 21-14. {S,D}ACC Register to Flash Array Mapping (continued)**

| Register Bit | Starting Flash Array Address | Sector Size |
|---|---|---|
| xACC[28] | 0x34_0000 | 256 KB |
| xACC[29] | 0x38_0000 | 256 KB |
| xACC[30] | 0x3C_0000 | 256 KB |
| xACC[31] | Reserved | |

### 21.3.2.11 PFlash Data Access Control Register (PFDACC)

Offset: FLASH_REGS_BASE + 0x002C Access: User read/write

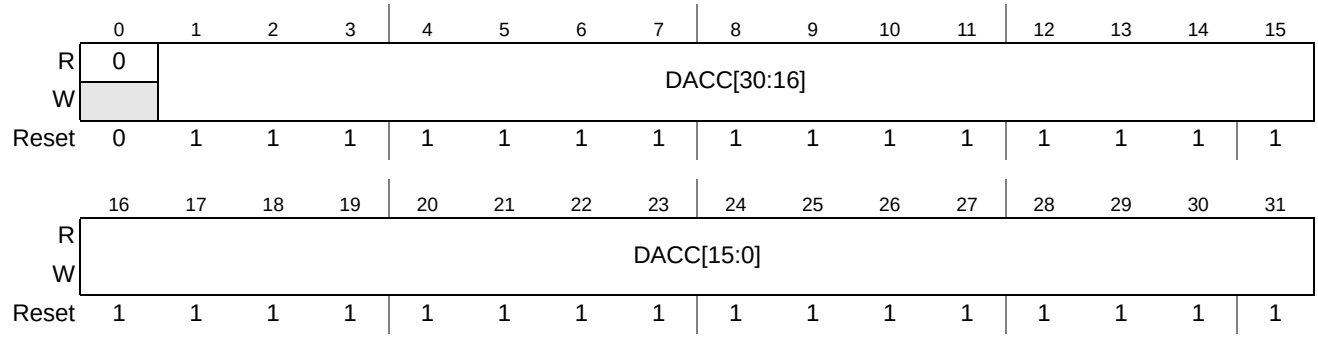



**Figure 21-14. PFlash Data Access Control Register (PFSACC)**

**Table 21-15. PFDACC field descriptions**

| Field | Description |
|---|---|
| DACC | Data Access Control. This bit field defines code/data access control for each sector within the main flash array.<br>0 Flash array sector *n* can be accessed only by a data reference. An attempted instruction fetch is terminated with an AHB error response. If the requesting bus master is the processor core, the ERROR response typically generates an instruction abort or data abort exception.<br>1 Flash array sector *n* can be accessed either as an instruction or data reference.<br>The mapping of this bit field to the main flash array is defined in Table 21-14.<br>This field is initialized by hardware reset to the value contained in address 0x3E10 of the shadow block of the flash array. |

### 21.3.2.12 User Test Register 0 (UT0)

The User Test Register 0 (UT0) provides a means to control UTest. The UTest mode gives the users of the flash module the ability to perform test features on the flash. This register is only writable when the flash is put into UTest mode by writing a passcode.

Offset: FLASH_REGS_BASE + 0x003C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | UTE | SCBE | 0 | 0 | 0 | 0 | 0 | 0 | DSI | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | MRE | MRV | EIE | AIS | AIE | AID |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 21-15. User Test Register 0 (UT0)**

**Table 21-16. UT0 Field Descriptions**

| Field | Description |
|---|---|
| UTE | UTest Enable. This status bit gives indication when UTest is enabled. All bits in UT0, UT1, UT2, UM0, UM1, UM2, UM3, and UM4 are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write. The UTE password will only be accepted if MCR[PGM] = 0 and MCR [ERS] = 0 (program and erase are not being requested). UTE can only be cleared if UT0[AID] = 1, UT0[AIE] and UT0[EIE] = 0. While clearing UTE, writes to set AIE or set EIE will be ignored. For UTE, the password 0xF9F9_9999 must be written to the UT0 register. |
| SCBE | Single Bit Correction Enable. SBC enables Single Bit Correction results to be observed in MCR[SBC]. Also is used as an enable for interrupt signals created by the flash module. ECC corrections that occur when SBCE is cleared will not be logged.<br>0 Single Bit Corrections observation is disabled.<br>1 Single Bit Correction observation is enabled. |
| DSI | Data Syndrome Input. These bits enable checks of ECC logic by allowing check bits to be input into the ECC logic and then read out by doing array reads or array integrity checks. The DSI[7:0] correspond to the 8 ECC check bits on a double word. |
| MRE | Margin Read Enable. MRE combined with MRV enables Factory Margin Reads to be done. Margin reads are only active during Array Integrity Checks. Normal user reads are not affected by MRE. MRE is not writable if AID is low.<br>0 Margin reads are not enabled.<br>1 Margin reads are enabled during Array Integrity Checks. |
| MRV | Margin Read Value. MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV = 1) or to a programmed level (MRV = 0). In order for this value to be valid, MRE must also be set. MRV is not writable if AID is low.<br>0 Zero's margin reads are requested.<br>1 One's margin reads are requested. |
| EIE | ECC Data Input Enable. EIE enables the input registers (DSI and DAI) to be the source of data for the array. This is useful in the ECC logic check. If this bit is set, data read through a PFLASH2P read request will be from the DSI and DAI registers when an address match is achieved to the ADR register. EIE is not simultaneously writable to a 1 as UTI is being cleared to a 0.<br>0 Data read is from the flash array.<br>1 Data read is from the DSI and DAI registers. |

**Table 21-16. UT0 Field Descriptions (continued)**

| Field | Description |
|---|---|
| AIS | Array Integrity Sequence. AIS determines the address sequence to be used during array integrity checks. The default sequence (AIS = 0) is meant to replicate sequences normal “user” code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS = 1) is just logically sequential.<br>It should be noted that the time to run a sequential sequence is significantly shorter than the time to run the proprietary sequence. If MRE is set, AIS has no effect.<br>0 Array integrity sequence is proprietary sequence.<br>1 Array integrity sequence is sequential. |
| AIE | Array Integrity Enable. AIE set to one starts the array integrity check done on all selected and unlocked blocks. The address sequence selected is determined by AIS, and the MISR (UM0 through UM4) can be checked after the operation is complete, to determine if a correct signature is obtained. Once an Array Integrity operation is requested (AIE = 1), it may be terminated by clearing AIE if the operation has finished (AID = 1) or aborted by clearing AIE if the operation is ongoing (AID = 0). AIE is not simultaneously writable to a 1 as UTI is being cleared to a 0.<br>0 Array integrity checks are not enabled.<br>1 Array integrity checks are enabled. |
| AID | Array Integrity Done. AID is cleared upon an Array integrity check being enabled (to signify the operation is ongoing). Once completed, AID is set to indicate that the array integrity check is complete. At this time the MISR (UMR registers) can be checked. AID can not be written, and is status only.<br>0 Array integrity check is ongoing.<br>1 Array integrity check is done. |

## 21.3.2.13 User Test Register 1 (UT1)

The User Test Register 1 (UT1) provides added controllability to UTest.

Offset: FLASH_REGS_BASE + 0x0040 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DAI | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DAI | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-16. User Test Register 1 (UT1)**

**Table 21-17. UT1 Field Descriptions**

| Field | Description |
|---|---|
| DAI<br>[31:0] | Data Array Input. These bits enable checks of ECC logic by allowing data bits to be input into the ECC logic and then read out by doing array reads or array integrity checks. The DAI[31:0] correspond to the 32 Array bits representing Word 0 of the double word selected in the ADR register. |

### 21.3.2.14 User Test Register 2 (UT2)

Offset: FLASH_REGS_BASE + 0x0044 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DAI | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DAI | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-17. User Test Register 2 (UT2)**

**Table 21-18. UT2 Field Descriptions**

| Field | Description |
|---|---|
| DAI [63:32] | Data Array Input. These bits enable checks of ECC logic by allowing data bits to be input into the ECC logic and then read out by doing array reads or array integrity checks. The DAI[63:32] correspond to the 32 Array bits representing Word 1of the double word selected in the ADR register. |

### 21.3.2.15 User Multiple Input Signature Register [0:4] (UM*n*)

The User Multiple Input Signature Registers (UM*n*) provide a means to evaluate array integrity.

Offset FLASH_REGS_BASE + 0x0048 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MISR[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-18. User Multiple Input Signature Register 0 (UM0)**

Offset FLASH_REGS_BASE + 0x004C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MISR[63:32] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-19. User Multiple Input Signature Register 1 (UM1)**

Offset FLASH_REGS_BASE + 0x0050 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MISR[95:64] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-20. User Multiple Input Signature Register 2 (UM2)**

Offset FLASH_REGS_BASE + 0x0054 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MISR[127:96] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-21. User Multiple Input Signature Register 3 (UM3)**

Offset FLASH_REGS_BASE + 0x0058 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | MISR[143:128] | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 21-22. User Multiple Input Signature Register 4 (UM4)**

**Table 21-19. UM*n* Field Descriptions**

| Field | Description |
|---|---|
| MISR | Multiple Input Signature Register bits. The MISR bitfields accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the read transfer error signal. The MISR can be seeded to any value by writing the MISR registers.<br><br>The MISR register provides a means to calculate a MISR during Array Integrity operations.<br><br>The MISR can be represented by the following polynomial:<br>$x^{145} + x^6 + x^5 + x^1 + 1$<br>The MISR is calculated by taking the previous MISR value and then “exclusive ORing” the new data. In addition the most significant bit (in this case it is MISR[144]), is then “exclusive ORed” into input of MISR[6], MISR[5], MISR[1], and MISR[0]. The result of the “exclusive OR” is shifted left on each read.<br><br>The MISR register is used in Array Integrity operations.<br>If during address sequencing, reads extend into an invalid address location (i.e. greater than the maximum address for a given array size) or locked/un-selected blocks. Reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not re-calculated, and the previous value is retained. |

### 21.3.2.16 Nonvolatile private censorship PassWord 0 register (NVPWD0)

Offset: 0x03DD8[1] Access: Read / Write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PWD31 | PWD30 | PWD29 | PWD28 | PWD27 | PWD26 | PWD25 | PWD24 | PWD23 | PWD22 | PWD21 | PWD20 | PWD19 | PWD18 | PWD17 | PWD16 |
| Reset[2] | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PWD15 | PWD14 | PWD13 | PWD12 | PWD11 | PWD10 | PWD09 | PWD08 | PWD07 | PWD06 | PWD05 | PWD04 | PWD03 | PWD02 | PWD01 | PWD00 |
| Reset | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**Figure 1. Nonvolatile private censorship PassWord 0 register (NVPWD0)**

1 See device memory map table for base address information of shadow flash.

2 Reset values labeled “X” are loaded from the NVLML value which is preprogrammed at the factory. The default value from the factory for this register is 0xFEED_FACE. The default value can be reprogrammed by the user.

The nonvolatile private censorship password 0 register contains the 32 LSB of the Password used to validate the Censorship information contained in NVSCC0–1 registers.

**Table 23. NVPWD0 field descriptions**

| Field | Description |
|---|---|
| PWD[31:00] | *PassWorD 31-00* (Read/Write)<br>The PWD31-00 registers represent the 32 LSB of the Private Censorship Password. |

### 21.3.2.17 Nonvolatile private censorship PassWord 1 register (NVPWD1)

The nonvolatile private censorship password 1 register contains the 32 MSB of the Password used to validate the Censorship information contained in NVSCC0–1 registers.

**NOTE**

In a secured device, starting with a serial boot, it is possible to read the content of the four flash memory locations where the RCHW can be stored. For example if the RCHW is stored at address 0x00000000, the reads at address 0x00000000, 0x00000004, 0x00000008 and 0x0000000C will return a correct value. Any other flash memory address cannot be accessed.

Offset: 0x03DDC[1] Access: Read / Write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PWD63 | PWD62 | PWD61 | PWD60 | PWD59 | PWD58 | PWD57 | PWD56 | PWD55 | PWD54 | PWD53 | PWD52 | PWD51 | PWD50 | PWD49 | PWD48 |
| Reset[2] | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PWD47 | PWD46 | PWd45 | PWD44 | PWD43 | PWD42 | PWD41 | PWD40 | PWD39 | PWD38 | PWD37 | PWD36 | PWD35 | PWD34 | PWD33 | PWD32 |
| Reset | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**Figure 2. Nonvolatile private censorship PassWord 1 register (NVPWD1)**

1 See device memory map table for base address information of shadow flash.

2 Reset values labeled “X” are loaded from the NVLML value which is preprogrammed at the factory. The default value from the factory for this register is 0xCAFE_BEEF. The default value can be reprogrammed by the user.

**Table 24. NVPWD1 field descriptions**

| Field | Description |
|---|---|
| **PWD[63:32]** | *PassWorD 63-32* (Read/Write)<br>The PWD63-32 registers represent the 32 MSB of the Private Censorship Password. |

## 21.3.2.18 Nonvolatile System Censoring Information 0 register (NVSCC0)

Offset: 0x03DE0[1] Delivery value: 0x55AA_55AA

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SC15 | SC14 | SC13 | SC12 | SC11 | SC10 | SC9 | SC8 | SC7 | SC6 | SC5 | SC4 | SC3 | SC2 | SC1 | SC0 |
| Reset[2] | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CW15 | CW14 | CW13 | CW12 | CW11 | CW10 | CW9 | CW8 | CW7 | CW6 | CW5 | CW4 | CW3 | CW2 | CW1 | CW0 |
| Reset | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**Figure 3. Nonvolatile System Censoring Information 0 register (NVSCC0)**

1 See device memory map table for base address information of shadow flash.

2 Reset values labeled “X” are loaded from the NVLML value which is preprogrammed at the factory. The default value from the factory for this register is 0x55AA_55AA. The default value can be reprogrammed by the user.

The nonvolatile system censoring information 0 register stores the 32 LSB of the Censorship Control Word of the device.

The NVSCC0 is a nonvolatile register located in the Shadow sector: it is read during the reset phase of the flash memory module and the protection mechanisms are activated consequently.

The parts are delivered uncensored to the user.

**Table 25. NVSCC0 field descriptions**

| Field | Description |
|---|---|
| SC[15:0] | *Serial Censorship control word 15-0* (Read/Write)<br>These bits represent the 16 LSB of the Serial Censorship Control Word (SCCW).<br>If SC15-0 = 0x55AA, the Public Access is disabled.<br>If SC15-0 ≠ 0x55AA, the Public Access is enabled. |
| CW[15:0] | *Censorship control Word 15-0* (Read/Write)<br>These bits represent the 16 LSB of the Censorship Control Word (CCW).<br>If CW15-0 = 0x55AA, the Censored Mode is disabled.<br>If CW15-0 ≠ 0x55AA, the Censored Mode is enabled. |

## 21.3.2.19 Nonvolatile User Options register (NVUSRO)

Offset: 0x03E18[1] Access: Read / Write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WATCHDOG_EN | UO30 | UO29 | UO29 | UO27 | UO26 | SMD_PAD3V5V | UO24 | UO23 | UO22 | UO21 | UO20 | UO19 | UO18 | UO17 | UO16 |
| Reset[2] | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | UO15 | UO14 | UO13 | UO12 | UO11 | UO10 | UO09 | UO08 | UO07 | UO06 | UO05 | UO04 | UO03 | UO02 | UO01 | UO01 |
| Reset | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |

**Figure 4. Nonvolatile User Options register (NVUSRO)**

1 See device memory map table for base address information of shadow flash.

2 Reset values labeled “X” are loaded from the NVLML value which is preprogrammed at the factory. The default value from the factory for this register is 0xXXXX_XXXX. The default value can be reprogrammed by the user.

The Nonvolatile User Options Register contains configuration information for the user application.

The NVUSRO register is a 64-bit register, of which the 32 most significant bits 63:32 are ‘don’t care’ and eventually used to manage ECC codes.

**Table 26. NVUSRO field descriptions**

| Field | Description |
|---|---|
| **UO** | *User Options* (Read/Write)<br>The UO generic field is reset based on the information stored in NVUSRO. |
| **STCU_EN** | Self MBIST Enable (Read/Write)<br>1: Disable after reset<br>0: Enable after reset<br>Default manufacturing value before flash memory initialization is ‘1’. |
| **UO[20:04]** | *User Options 20:04* (Read/Write)<br>The UO20-4 generic registers are reset based on the information stored in NVUSRO. |
| **PAD3V5V[0]** | High voltage supply for $V_{DD_HV_B}$ domain<br>0: High voltage supply is 5.0 V<br>1: High voltage supply is 3.3 V<br>Default manufacturing value before flash memory initialization is ‘1’ (3.3 V) which should ensure correct minimum slope for boundary scan. |
| **PAD3V5V[1]** | High voltage supply for $V_{DD_HV_A}$ domain<br>0: High voltage supply is 5.0 V<br>1: High voltage supply is 3.3 V<br>Default manufacturing value before flash memory initialization is ‘1’ (3.3 V) which should ensure correct minimum slope for boundary scan. |
| **UO[01]** | *User Options 01* (Read/Write)<br>The UO01 generic register is reset based on the information stored in NVUSRO. |
| **WATCHDOG_EN** | 0: Disable after reset<br>1: Enable after reset<br>Default manufacturing value before flash memory initialization is ‘1’ |

## 21.4 Functional description

The Flash module may operate in various modes. The modes that are available include:

- User mode
- Low Power mode
- Power Down mode
- UTest mode

Each of these modes are discussed in the following chapters, and more details provided about each mode.

### 21.4.1 User mode

In user mode the flash module may be read and written (register writes and interlock writes), programmed or erased. The following sub-sections define all actions that may be performed in user mode.

#### 21.4.1.1 Read and write

The default state of the flash module is read. The main and shadow address space can be read only in the read state. The MCR is always available for read, except when the module is in Low Power mode or Power Down mode. The flash module enters the read state on reset. The module is in the read state under four sets of conditions:

- The read state is active when the module is enabled (User Mode Read or UTEST).
- The read state is active when MCR[PGM] and/or MCR[ERS] are high and High Voltage operation is ongoing (Read While Write).

**NOTE**

Reads done to the partition(s) being operated on (either erased or programmed) result in an error and the MCR[RWE] bit is set.

- The read state is active when MCR[PGM] and MCR[PSUS] are high (Program suspend).
- The read state is active when MCR[ERS] and MCR[ESUS] are high and MCR[PGM] is low (Erase suspend).

In the flash module, FC reads return 128 bits (1 page). MCR reads return 32 bits of data.

**WARNING**

FC reads are done through the PFLASH2P. In many cases the PFLASH2P does “read page buffering” to allow sequential reads to be done with higher performance. This could provide a Data Coherency issue that must be handled with software. Data Coherency may be an issue after a program or erase operation, as well as shadow block operations.

In user mode, registers may be written. Array may be written to do interlock writes.

Register reads to unmapped register address space return all 0’s.

Array reads attempted to invalid locations result in indeterminate data and indeterminate error flags. Invalid locations occur when addressing is done to blocks that do not exist in non $2^n$ array sizes.

Register writes to unmapped register address space have no effect.

Interlock writes attempted to invalid locations (due to blocks that do not exist in non $2^n$ array sizes), result in an interlock occurring, but attempts to program these blocks does not occur since they are forced to be locked. Erase occurs to selected and unlocked blocks even if the interlock write is to an invalid location.

#### 21.4.1.2 Flash programming

A flash program sequence operates on any page within the FC. Up to 4 words within the page may be altered in a single program operation. Whenever the array is program, the ECC bits also get programmed. ECC is handled on a 64 bit boundary. Thus, if only 1 word in any given 64 bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word results in an operation failure (most likely). It is recommended that all programming operations be from 64 bits to 128 bits, and

be 64 bit aligned. The programming operation should completely fill selected ECC segments within the page. Only one program is allowed per 64 bit ECC segment between erases.

**WARNING**

In rare cases “over programming” of a 64 bit ECC segment may be done (EEPROM emulation).

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

**NOTE**

If a logic 0 is attempted to be “over programmed” by a logic 1, the resulting operation will fail (MCR[PEG] = 0), and the 0’s that are interlocked will be merged (ORed) with 0’s that are already present in the 64 bit ECC segment.

Addresses in locked/disabled blocks cannot be programmed. The user may program the values in any or all of 4 words, within a page, with a single program sequence. Page-bound words have addresses which differ only in address bits [3:2]. The program operation consists of the following sequence of events:

1. Change the value in the MCR[PGM] bit from a 0 to a 1.

**NOTE**

Ensure the block that contains the address to be programmed is unlocked.

2. Write the first address to be programmed with the program data. The flash module latches address bits [20:4] and SOC specific shadow enable at this time. The flash module latches data written as well. This write is referred to as a program data interlock write. An interlock write may be as large as 64 bits, and as small as 32 bits.
3. If more than 1 word or double word is to be programmed, write each additional address in the page with data to be programmed. This is referred to as a program data write. The flash module ignores address bits [20:4] and SOC specific shadow enable for program data writes. All unwritten data words default to 0xffff ffff.
4. Write a logic 1 to the MCR[EHV] bit to start the internal program sequence or skip to step  to terminate.
5. Wait until the MCR[DONE] bit goes high.
6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic 0 to the MCR[PGM] bit to terminate the program sequence.

The program sequence is presented graphically in Figure 21-27. The program suspend operation detailed in Figure 21-27 is discussed in Section 21.4.1.2.2, “Program suspend/resume.”

The first write after a program is initiated determines the page address to be programmed. Program may be initiated with the 0 to 1 transition of the MCR[PGM] bit or by clearing the MCR[EHV] bit at the end of a previous program. This first write is referred to as an interlock write. The interlock write determines if the shadow or normal array space is to be programmed by sampling SOC specific shadow enable and causing MCR[PEAS] to be set/cleared.

In the case of an erase-suspended program, the values in MCR[PEAS] may be modified via the program interlock write, enabling erase-suspended programs to and from shadow space.

An interlock write must be performed before setting EHV. The user may terminate a program sequence by clearing MCR[PGM] prior to setting MCR[EHV].

After the interlock write, additional writes only affect the data to be programmed at the word location determined by address bits [3:2]. Unwritten locations default to a data value of 0xffff_ffff. If multiple writes are done to the same location the data for the last write is used in programming.

While DONE is low, EHV is high and PSUS is low the user may clear EHV, resulting in a program abort. A program abort forces the module to step 8 of the program sequence. An aborted program results in PEG being set low, indicating a failed operation. The data space being operated on before the abort contains indeterminate data. The user may not abort a program sequence while in program suspend.

**WARNING**

Aborting a program operation leaves the FC addresses being programmed in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

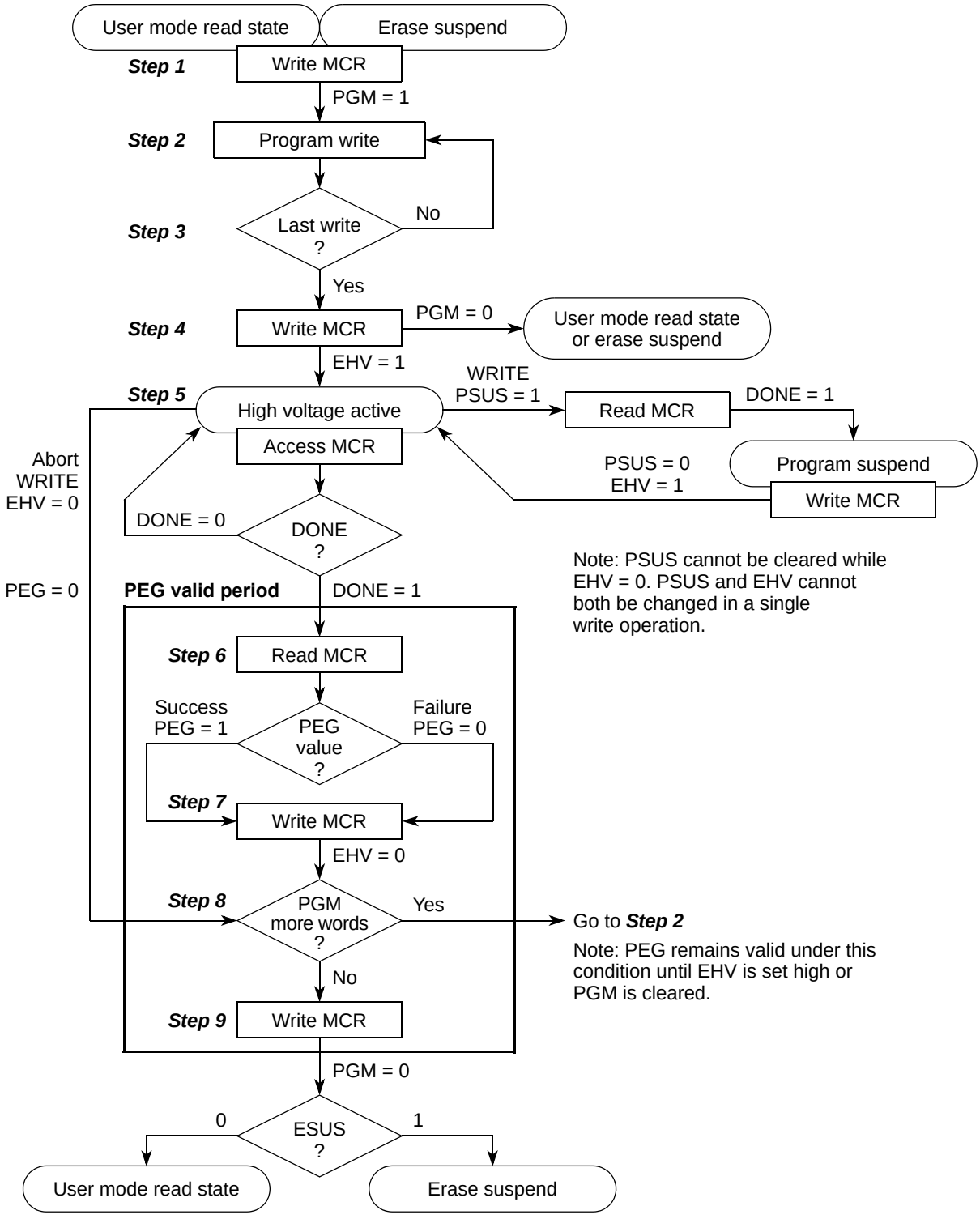



**Figure 21-27. Program Sequence**

#### 21.4.1.2.1 Program software locking

A software mechanism is provided to independently lock/unlock each High, Mid and Low Address Space block against program and erase.

Software Locking is done through the LML (Low/Mid Address Space Block Lock) or HBL (High Address Space Block Lock) registers. These may be written through register writes, and may be read through register reads.

#### 21.4.1.2.2 Program suspend/resume

The program sequence may be suspended to allow read access to the FC. It is not possible to erase during a program suspend, or program during a program suspend. Read While Write may also be used to read the array during a program sequence providing the read is to a different partition.

A program suspend can be initiated by changing the value of the MCR[PSUS] bit from a 0 to a 1. MCR[PSUS] can be set high at any time when MCR[PGM] and MCR[EHV] are high. A 0 to 1 transition of MCR[PSUS] causes the flash module to start the sequence to enter program suspend, which is a read state. The user must wait until MCR[DONE] = 1 before the module is suspended. At this time FC reads may be attempted. MCR[DONE] goes high no more than Tpsus (Appendix A) after MCR[PSUS] is set to a 1. Once suspended, the FC may only be read. Reads to the block(s) being programmed/erased return indeterminate data.

The program sequence is resumed by writing a logic 0 to MCR[PSUS]. MCR[EHV] must be set to a 1 before clearing MCR[PSUS] to resume operation. When the operation resumes, the flash module continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

**WARNING**

Repeated suspends at a high frequency may result in the operation timing out, and the flash module will respond by completing the operation with a fail code (MCR[PEG] = 0). The minimum time between suspends to ensure this does not occur is 100uS.

### 21.4.1.3 Flash Erase

Erase changes the value stored in all bits of the selected block(s) to logic 1. An erase sequence operates on any combination of blocks in the Low, Mid or High Address Space, or the shadow block. The erase sequence is fully automated within the flash. The user only needs to select the blocks to be erased and initiate the erase sequence. Locked/disabled blocks cannot be erased. If multiple blocks are selected for erase during an erase sequence, the blocks are erased sequentially starting with the lowest numbered block and terminating with the highest. The erase sequence consists of the following sequence of events:

1. Change the value in the MCR[ERS] bit from 0 to a 1.
2. Select the block, or blocks to be erased by writing ones to the appropriate registers in LMS or HBS registers. If the shadow block is to be erased, this step may be skipped, and LMS and HBS are ignored.

**NOTE**

Lock and Select are independent. If a block is selected and locked, no erase occurs.

3. Write to any address in flash. This is referred to as an erase interlock write. The interlock write causes the values of SOC specific shadow enable to be captured and causing MCR[PEAS] to be set/cleared.
4. Write a logic 1 to the MCR[EHV] bit to start an internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.
6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR[ERS] bit to terminate the erase.

The erase sequence is presented graphically in Figure 21-28. The erase suspend operation detailed in Figure 21-28 is discussed in section Section 21.4.1.3.2, “Erase Suspend/Resume.”

After setting ERS, one write, referred to as an interlock write, must be performed before EHV can be set to a 1. This interlock causes the values of SOC specific shadow enable to be captured. Data words written during erase sequence interlock writes are ignored. The user may terminate the erase sequence by clearing ERS before setting EHV.

An erase operation may be aborted by clearing EHV assuming DONE is low, EHV is high and ESUS is low. An erase abort forces the module to step 8 of the erase sequence. An aborted erase results in PEG being set low, indicating a failed operation. The block(s) being operated on before the abort contain indeterminate data. The user may not abort an erase sequence while in erase suspend.

**WARNING**

Aborting an erase operation leaves the FC blocks being erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

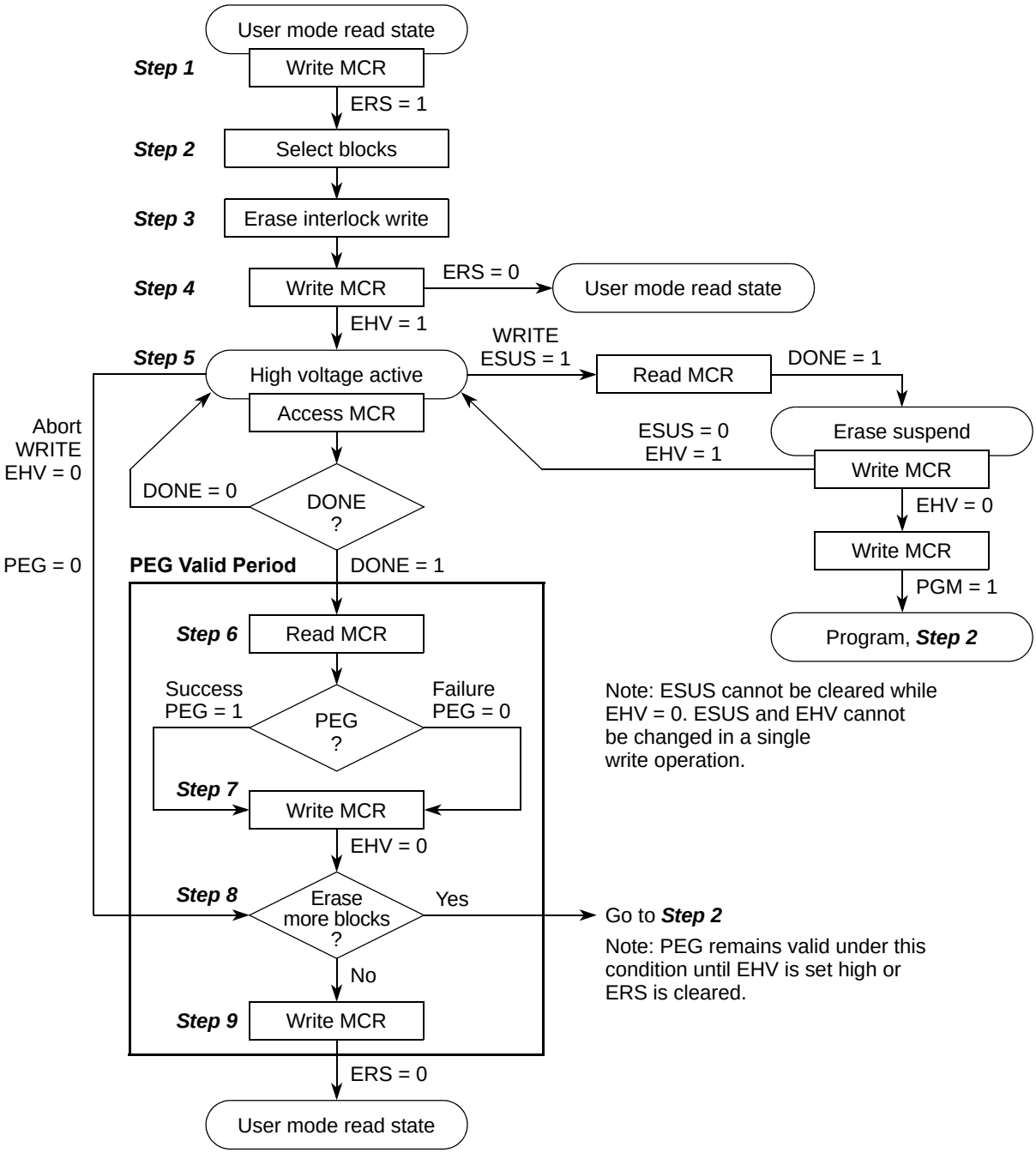



**Figure 21-28. Erase Sequence**

### 21.4.1.3.1 Erase Software Locking

Software Locking affect erase operation. For details on this see Section 21.4.1.2.1, “Program software locking.

#### 21.4.1.3.2 Erase Suspend/Resume

The erase sequence may be suspended to allow read access to the FC. The erase sequence may also be suspended to program (Erase-Suspended Program) the FC. A program started during erase suspend can in turn be suspended. Only one erase suspend and one program suspend are allowed at a time during an operation. It is not possible to erase during an erase suspend, or program during a program suspend. During suspend, all reads to FC locations targeted for program and blocks targeted for erase return indeterminate data. Programming locations in blocks targeted for erase during erase-suspended program may result in corrupted data. Read While Write may also be used to read the array during an erase sequence providing the read is to a partition not selected for erase.

An erase suspend can be initiated by changing the value of the MCR[ESUS] bit from a 0 to a 1. MCR[ESUS] can be set to a 1 at any time when MCR[ERS] and MCR[EHV] are high and MCR[PGM] is low. A 0 to 1 transition of MCR[ESUS] causes the module to start the sequence which places it in erase suspend. The user must wait until MCR[DONE] = 1 before the module is suspended and further actions are attempted. MCR[DONE] goes high no more than Tesus (Appendix A) after MCR[ESUS] is set to a 1. Once suspended, the array may be read or a program sequence may be initiated (erase-suspended program). Before initiating a program sequence the user must first clear MCR[EHV]. If a program sequence is initiated the values of SOC specific shadow enable is recaptured. Once the erase-suspended program is completed, the value of PEAS is returned to its' "erase" value. FC reads while MCR[ESUS] = 1 from the block(s) being erased return indeterminate data.

The erase sequence is resumed by writing a logic 0 to MCR[ESUS]. MCR[EHV] must be set to a 1 and MCR[PGM] must be cleared (in the event of an erase suspended program) before MCR[ESUS] can be cleared to resume the operation. The module continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

**WARNING**

Repeated suspends at a high frequency may result in the operation timing out, and the flash module will respond by completing the operation with a fail code (MCR[PEG] = 0). The minimum time between erase suspends to ensure this does not occur is 200uS.

**WARNING**

In an erase-suspended program, programming FC locations in blocks which were being operated on in the erase may corrupt FC data.

### 21.4.2 Low Power mode

After Low Power mode is requested, the flash memory module turns off most current sources, although logic/charge pumps to enable quick recovery to read are enabled for faster wake up time than Power Down mode.

When in Low Power mode, register access is prevented. FC accesses are also prevented until Power Down mode is exited. FC reads and writes may occur as soon as sleep mode is exited.

The flash module returns to its pre-Low Power state when enabled in all cases unless in the process of executing a program or erase high voltage operation at the time of sleep. If the flash memory module is

put into Low Power mode during a program or erase high voltage operation, the appropriate suspend bit is set to a 1. The user may resume the program or erase operation at the time the module is enabled by clearing the appropriate suspend bit. EHV must be high for the module to resume operation. If both the ESUS and PSUS bits are set to a 1 the user must clear PSUS to resume the program. The erase may be resumed after the program ends.

### 21.4.3 Power Down mode

After entering Power Down mode, the flash module turns off all DC current sources and no reads from or writes to the module are possible. All power dissipation is due to leakage in this mode.

When in Power Down mode, register access is prevented. FC accesses are also prevented until Power Down mode is exited.

The flash memory module returns to its pre-Power Down state when enabled in all cases unless in the process of executing a program or erase high voltage operation at the time of entering Power Down mode. If the flash memory module is configured to enter Power Down mode during a program or erase high voltage operation, the appropriate suspend bit is set to a 1. The user may resume the program or erase operation at the time the module is enabled by clearing the appropriate suspend bit. EHV must be high for the module to resume operation. If both the ESUS and PSUS bits are set to a 1 the user must clear PSUS to resume the program. The erase may be resumed after the program ends.

### 21.4.4 UTest Mode

UTest mode is a mode that customers can put the flash module in to do specific tests to check the integrity of the Flash module.

#### 21.4.4.1 Array Integrity Self Check

Array Integrity is checked using a pre-defined address sequence (based on UT0[AIS]), and this operation is executed on selected and unlocked blocks. The data to be read is customer specific, thus a customer can provide user code into the flash and the correct MISR value is calculated. The customer is free to provide any random or non-random code, and a valid MISR signature is calculated. Once the operations is completed, the results of the reads can be checking by reading the MISR value, to determine if an incorrect read, or ECC detection was noted. Array integrity is controlled by the system clock (IPG), and it is required that the Read Wait States and Address Pipelined control registers in the PFLASH2P be set to match the user defined frequency being used. The Array Integrity Check consists of the following sequence of events:

1. Enable UTest mode.
2. Select the block, or blocks to be receive array integrity check by writing ones to the appropriate registers in LMS or HBS registers.

**NOTE**

Locked Blocks can be tested with Array Integrity if selected in LMS and HBS.

**NOTE**

It is not possible to do UTest operations on the shadow block.

3. If desired, Set the UT0[AIS] bit to 1 for sequential addressing only.

**NOTE**

For normal integrity checks of the flash memory, sequential addressing is recommended.

**NOTE**

If it is required to more fully check the read path (in a diagnostic mode), it is recommend that AIS be left at 0, to use the address sequence that checks the read path more fully, and examine read transitions. This sequence takes more time.

4. Seed the MISR UM0 thru UM4 with desired values.
5. Set the UT0[AIE] bit.
   c) If desired, the Array Integrity operation may be aborted prior to UT0[AID] going high. This may be done by clearing the UT0[AIE] bit and then continuing to the next step. It should be noted that in the event of an aborted array integrity check the MISR registers will contain a signature for the portion of the operation that was completed prior to the abort, and will not be deterministic. Prior to doing another array integrity operation, the UM0, UM1, UM2 and UM3 registers may need to be initialized to the desired seed value by doing register writes.
6. Wait until the UT0[AID] bit goes high.
7. Read values in the MISR registers (UM0 through UM4) to ensure correct signature.
8. Write a logic 0 to the UT0[AIE] bit.

### 21.4.4.2 Factory Margin Read

Factory Margin Read must be done following “Initial Factory Conditions”. One Factory Margin Read is allowed per erase.

Factory Margin Read may be done to selected and unlocked blocks by combining UT0[MRE] and UT0[MRV] with the Array Integrity check. If UT0[MRE] is set, UT0[AIS] has no affect, and the reads will be done sequentially.

The data to be read is customer specific, thus a customer can provide user code into the flash and the correct MISR value is calculated. The customer is free to provide any random or non-random code, and a valid MISR signature is calculated. Once the operations is completed, the results of the reads can be checking by reading the MISR value. Factory Margin Read is a self timed event, and is independent of system clocks, or wait states selected. Margin ECC corrections or detections are not done during the Factory Margin Read test:

1. Enable UTest mode.
2. Select the block, or blocks to be receive margin read check by writing ones to the appropriate registers in LMS or HBS/EHS registers. Make sure that selected blocks are also unlocked.

**NOTE**

It is not possible to do UTest operations on the shadow block.

**NOTE**

It is possible to do User Mode array reads during the Factory Margin Read test, if desired, but the partition rules for Read While Write used during program and erase are in effect during Factory Margin Reads.

**NOTE**

If it is desired to do 2 or more margin reads, and it is desired to re-seed the MISR, then a reset is required between operations. If the subsequent margin reads can be performed with the previously calculated MISR value, then a reset is not required.

3. Set the UT0[MRE] bit.
4. Set the UT0[MRV] bit to desired value depending on it is desired to do One's Margin or Zero's Margin.
5. Seed the MISR UM0 thru UM4 with desired values.
6. Set the UT0[AIE] bit.
   a) If desired, the Margin Read operation may be aborted prior to UT0[AID] going high. This may be done by clearing the UT0[AIE] bit and then continuing to the next step. It should be noted that in the event of an aborted Margin Read check the MISR registers will contain a signature for the portion of the operation that was completed prior to the abort, and will not be deterministic.
7. Wait until the UT0[AID] bit goes high.
8. Read values in the MISR registers (UM0 through UM4) to ensure correct signature.
9. Write a logic 0 to the UT0[AIE] bit.

#### 21.4.4.3 ECC Logic Check

ECC logic can be checked by providing data to be read in the UT0[DSI], UT1[DAI] and/or UT2[DAI] registers. Then array reads can be done, ensuring expected results. The ECC Logic Check consists of the following sequence of events:

1. Enable UTest mode.
2. Write UT0[EIE] to 1.
3. Write UT0[DSI], UT1[DAI] and/or UT2[DAI] bits to provide data and check bit values to be read. Single or Double bit detections/corrections can be simulated by properly choosing Data and Check Bit combinations.
4. Write double word address to receive the data inputted in step 3 into the ADR register.
5. Reads can now be done through the PFLASH2P in a Read Request type fashion. In the event of a PFLASH2P read requested from an address that matches the address in the ADR register, expected data, and corrections or detections should be observed based on data written into the UT0[DSI], UT1[DAI] and/or UT2[DAI] registers. MCR[EER] and MCR[SBC] can be checked to evaluate the status of reads done.

**NOTE**

In the event of an ECC error or Single Bit Correction, during the ECC Logic Check (UTO[EIE] high), the ADR register will not be loaded, and the address tagged to receive the UT0[DSI], UT1[DAI] and/or UT2[DAI] values will be persevered.

6. Once completed, clear the UT0[EIE] bit to 0.

## 21.4.5 PFLASH2P

The PFLASH2P has two AHB-Lite slave ports and a single flash array interface. The dual ported design of the PFLASH2P enables efficient use of a single flash memory array the CPU and other AHB masters. Each AHB port has dedicated line buffers to support single-cycle read accesses and to limit accesses to the flash array.

The PFLASH2P generates read and write enables, the flash array address, write size, and write data as inputs to the flash array controller. The PFLASH2P captures read data from the flash array interface and drives it onto the appropriate AHB port.

If line buffering is enabled, when data is read from the array it is stored in a line buffer. Up to four lines of data (128 bits) are buffered by the PFLASH2P for each AHB port.

If pre-fetching is enabled, data is read in advance and stored in the line buffers allowing single-cycle (zero AHB wait-states) read data responses on buffer hits. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Arbitration between the two AHB ports for access to the flash interface is primarily based on the type of access; writes have priority over reads which have priority over prefetches. If both ports are doing the same type of access, priority is based on the settings of the arbitration and priority bits in the PFCRP0 register.

### 21.4.5.1 Line Read Buffers and Prefetch Operation

The PFLASH2P_H7Fb contains four read buffers per AHB port which are used to hold line and ECC data read from the flash array. Each buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis. Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access to the PFLASH2P_H7Fb may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented to the next-higher 16 byte boundary, and a flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

Buffers may be in one of six states, listed here in prioritized order:

- Invalid - the buffer contains no valid data
- Used - the buffer contains valid data which has been provided to satisfy an AHB burst type read
- Valid - the buffer contains valid data which has been provided to satisfy an AHB single type read

- Prefetched - the buffer contains valid data which has been prefetched to satisfy a potential future AHB access
- Busy AHB - the buffer is currently being used to satisfy an AHB burst read
- Busy Fill - the buffer has been allocated to receive data from the flash array, and the array access is still in progress

Selection of a buffer to be loaded on a miss is based on the following replacement algorithm:

1. First, the buffers are examined to determine if there are any invalid buffers. If there are multiple invalid buffers, the one to be used is selected using a reverse numeric priority, where buffer 0 is selected first, then buffer 1, etc.
2. If there are no invalid buffers, the least-recently-used buffer is selected for replacement.

Once the candidate line buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the just-loaded buffer is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Several algorithms are available for prefetch control which trade off performance for power. More aggressive prefetching increases power due to the number of wasted (discarded) prefetches, but may increase performance by lowering average read latency.

In order for prefetching to occur, PFCRPx[BFEN] must be set to ‘1’; PFCRPx[PFLIM] must be non-zero; either PFCRPx[IPFEN] or PFCRPx[DPFEN] must be ‘1’ and PFCRPx[MxPFE] must be ‘1’.

#### 21.4.5.2 Instruction / Data Prefetch Triggering

Prefetch triggering may be enabled for instruction reads via the PFCRPx[IPFEN] control bit, while prefetching for data reads is enabled via the PFCRPx[DPFEN] control bit. Additionally, the PFCRPx[PFLIM] must also be set to enable prefetching. Prefetches are never triggered by write cycles.

#### 21.4.5.3 Per-Master Prefetch Triggering

Prefetch triggering may be controlled for individual bus masters via the PFCRPx[MxPFE] control field.

#### 21.4.5.4 Buffer Allocation

Allocation of the line read buffers is controlled via the PFCRPx control register for each AHB port. The LBCFG field of this register defines the operating organization of the four line buffers. The buffers can be organized as a “pool” of available resources with all four buffers available for either instruction or data. They can also be configured with a fixed partition between buffers allocated to instruction or data accesses. For the fixed partition, two configurations are supported. In one configuration, buffers 0 and 1 are allocated for instruction fetches and buffers 2 and 3 for data accesses. In the second configuration, buffers 0, 1 and 2 are allocated for instruction fetches and buffer 3 reserved for data accesses. In this con-

figuration data prefetching is disabled.

#### 21.4.5.5 Buffer Invalidation

The line read buffers may be invalidated by clearing the PFCRPx[BFEN] bit, which also disables the buffers. Software may then restore the PFCRPx[BFEN] bit to its previous state, and the buffers will have been invalidated.

## 21.5 Initialization Information

A reset is the highest priority operation for the flash module and terminates all other operations.

The flash module uses reset to initialize register and status bits to their default reset values. If the flash module is executing a program or erase operation (PGM and/or ERS = 1) and a reset is issued, the operation is aborted and the module disables the high voltage logic without damage to the high voltage circuits. Reset aborts all operations and forces the flash module into user mode ready to receive accesses.

After reset is requested, MCR[DONE] goes low, and remains low during reset and reset recovery. At the end of reset recovery, MCR[DONE] transitions from a 0 to a 1.

After reset is negated, register reads may be done, although it should be noted that registers that require updating from shadow information, or other inputs, may not read updated values until MCR[DONE] transitions high.

During reset recovery, register writes are not allowed until the MCR[DONE] bit transitions high to indicate reset recovery is completed.

## 21.6 Application information

### 21.6.1 Background

Flash array access is relatively slow compared to a full speed system clock based on the PLL. To prevent wait states on every flash access, line buffers are implemented. While wait states are required between the flash array and line buffer, no wait states are required between a line buffer and the system bus. For example, if the CPU is accessing sequential instructions starting at location 0, the first 32 bits (one line) fetched will require wait states. The number of wait states is based on system clock frequency. However, subsequent instructions contained in that 128 bit line buffer can be accessed without wait states.

Furthermore, with prefetching configured, the next sequential instructions outside the current line buffer can be prefetched to different line buffer. After fetching all the instructions in current line buffer, the next instruction is fetched for the next line buffer without delay.

Prefetching only helps performance when sequential accesses typically occur, such as for instructions. Since data typically is not arranged sequentially (except for perhaps graphic data) prefetching for data generally is not recommended.

The flash module on this device has two ports. Port 0 is always connected to the CPU. Port 1 is connected to the other non-CPU masters.

Configuring the flash bus interface parameters is done by writing to the Platform Flash Configuration Registers PFCRP0:1 and Platform Flash Access Protection Register PFAPR.

### 21.6.2 Flash memory setting recommendations

Table 21-1 provides an example of recommended settings for a common scenario with this device. This example assumes Port 0 (core) instruction accesses are typically sequential, but not data. Port 1 (other masters) will not have any instruction accesses. For illustration, this example assumes port 1 accesses have a significant amount of sequential data (such as for graphics) which are larger than a line buffer, so prefetching data would make sense. If graphic data were not in the internal flash, then prefetching data on port 1 would not be expected to be a benefit.

**Table 21-1. General flash memory setting recommendations for 125 MHz system clock**

| Parameter | General recommendations | | | |
|---|---|---|---|---|
| | Port 0 (CPU instruction only) | | Port 1 (CPU data, other masters) | |
| | Parameter symbol in register PFCR0 | Comments | Parameter symbol in register PFCR1 | Comments |
| Line Buffer Configuration | BFEN = 1 | Enable port's buffers | BFEN = 1 | Enable port's buffers |
| Instruction Prefetch Enable | IPFEN = 1 | Instructions are mostly sequential so prefeching can improve performance. | IPFEN = 0 | No instruction access on port 1 |
| Data Prefetch Enable | DPFEN = 0 | Data accesses are expected to generally be random not sequential | DPFEN = 1 | Enable prefetching assuming there is significant sequential data |
| Prefetch Limit | PFLIM = 3 | Prefetch on hit or miss | PFLIM = 1 | Prefetch on miss only (allows more bandwidth for core) |
| Line Buffer Configuration | LBCFG = 3 | Allocate 3 line buffers for instructions, 1 for data | LBCFG = 0 | All 4 line buffers available for any access |
| Read Wait States | RWSC = 3 | Values are system clock frequency dependent | RWSC = 3 | Values are system clock frequency dependent |
| Write Wait States | WWSC = 3 | | WWSC = 3 | |
| Adv. Pipeline Ctl. | APC = 3 | | APC = 3 | |
| Result value for recommendations in PFCR0 = 0x3001_7B17, PFCR1 = 0x007C_7B43 | | | | |

Table 21-2 illustrates flash access and protection by master. Note that PFAPR's initial value is loaded from shadow flash location 0x3E00 after reset. The "Master" numbers correspond to the crossbar masters, which for this device can be found in Table 9-1.

**Table 21-2. Access and protection setting recommendations**

| Parameter | Parameter symbol in PFAPR | Comments |
|---|---|---|
| Arbitration Mode | ARBM = 3 | Start with round-bin (2 or 3).Change to fixed priority if application analysis indicates improved performance. |
| Master n Prefetch Disable | MnPFD = 0 for core instructions, eDMA and DCU3; 1 for core data | Start with allowing prefetching (0) for CPU instructions since it is expected the core will have mostly sequential instruction accesses. Also, allow prefetching for eDMA and DCU3, assuming there are large blocks of graphic data accessed. |
| Master n Access Protection | MnAP = 3 for core data, 1 for core instructions, eDMA & DCU3 | Assuming only the CPU will program flash, allow read and write access (3) for the CPU data bus, but read access only (1) for CPU instructions, eDMA and DCU3. |

This page is intentionally left blank.

# Chapter 22
# Graphics Accelerator Gasket (GXG)

## 22.1 Introduction

The Graphics Accelerator Gasket (GXG) provides the OpenVG Graphics Accelerator (GFX2D) with a 32-bit IPS to AHB bridge to the slave port and a 64-bit AXI to AHB bridge to the master port. It also provides a direct AXI connection to the DRAM Controller.

The GXG has an address filter consisting of four programmable windows. Upon address detection, the GXG has several byte swapping options per window for the read or write data bus. To save on-chip graphics RAM space during write transactions, the GXG has the capability to suppress transactions upon address detection of alpha buffers, and to convert color depth from 32 to 24 bits per pixel upon address detection of frame buffers. During read transactions of detected addresses, an 8-bit constant is returned for each alpha byte component.

**Table 22-1. Acronyms and Abbreviated Terms**

| Term | Meaning |
|---|---|
| AIPS | AHB 2.v6 to IPS Interface Unit |
| AHB | Advanced High Performance Bus |
| AHB 2.v6 | AMBA AHB-Lite version 2.0 with v6 extensions |
| AMBA | Advanced Microcontroller Bus Architecture |
| ARGB | Alpha, Red, Green, Blue components of 32-bit color format |
| AXI | AMBA Advanced Extensible Interface |
| AXBS | AMBA Crossbar Switch |
| DRAM | Dynamic Random Access Memory |
| GFX2D | OpenVG Graphics Accelerator |
| GRAM | Graphics Random Access Memory |
| GXG | Graphics Accelerator Gasket |
| IPS | Skyblue line IP Interface |
| QuadSPI | Quad Serial Peripheral Interface |

### 22.1.1 Block Diagram

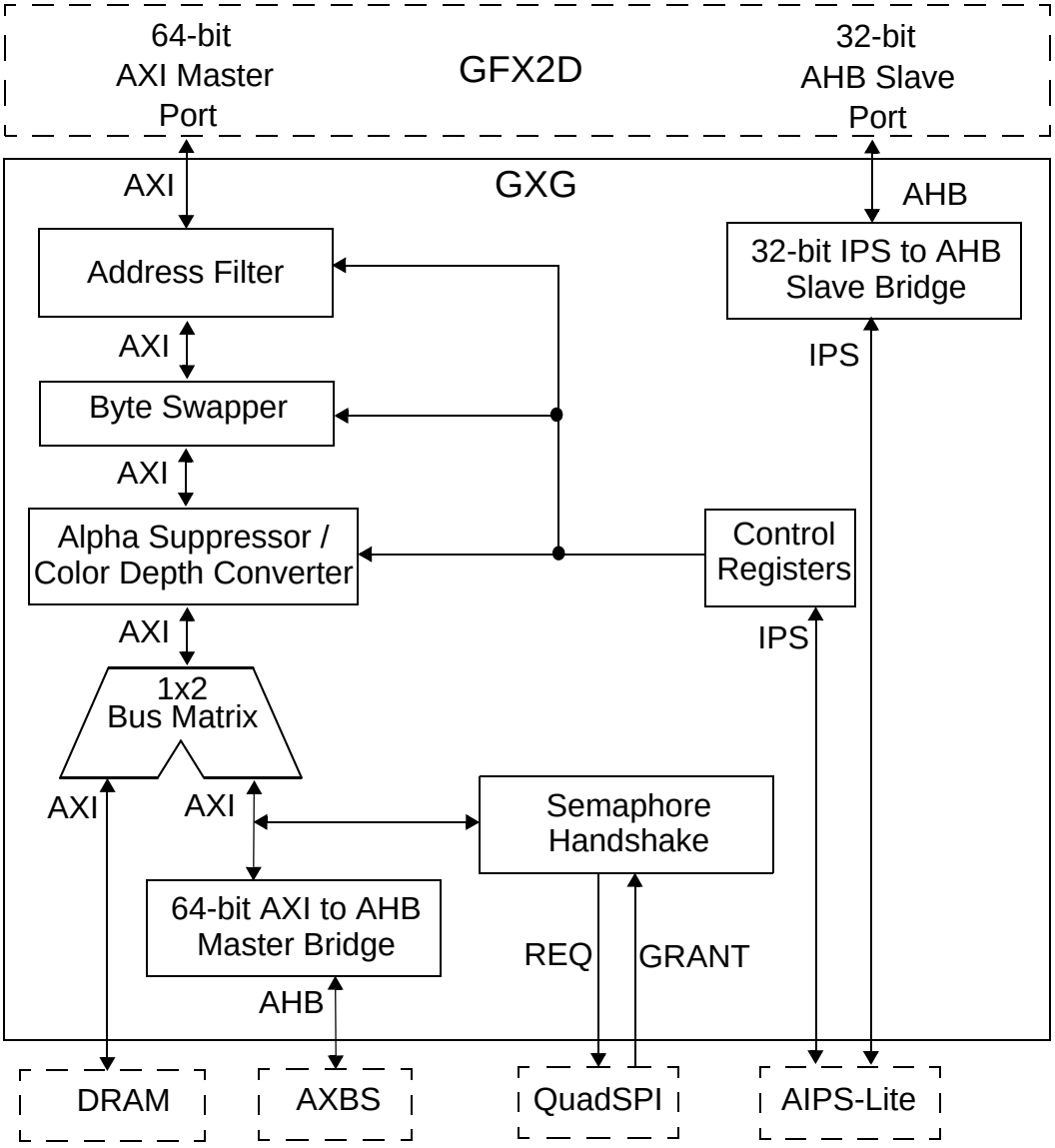



**Figure 22-1. Graphics Accelerator Gasket Block Diagram**

### 22.1.2 Features

- 32-bit slave port IPS to AHB
- 64-bit master port AXI to AHB
- AXI bus matrix
  - Active transactions to only a single slave at a time
  - Transactions forwarded in the same order that they were initiated.
- Address Filter with up to four programmable windows
- Byte Swapper for read and write data
  - Supports only 64-bit access
- Frame buffer Color Depth Conversion (32bpp to 24bpp)
  - Programmable destination base address
  - Byte indicator for alpha component
  - Supports only 32-byte incrementing bursts
- Alpha buffer Write Suppression and Read Constant Return
- Semaphore handshake for exclusive access to external serial flash

## 22.2 External Signal Description

The GXG does not include any external signals.

## 22.3 Memory Map and Register Definition

This section provides a detailed description of all GXG registers.

### 22.3.1 Memory Map

The GXG 16K space memory map is shown in Table 22-2. The address of each register is given as an offset to the GXG base address.

**Table 22-2. GXG Memory Map**

| Offset | Register | Access | Reset Value | Section/Page |
|---|---|---|---|---|
| 0x00 | GXGCNFG0—Window 0 Configuration | R/W | 0x0000_0000 | 22.3.2.1/22-870 |
| 0x04 | GXGBASE0—Window 0 Destination Base Address | R/W | 0x0000_0000 | 22.3.2.2/22-871 |
| 0x08 | GXGFRST0—Window 0 First Address | R/W | 0x0000_0000 | 22.3.2.3/22-872 |
| 0x0C | GXGLAST0—Window 0 Last Address | R/W | 0x0000_0000 | 22.3.2.4/22-872 |
| 0x10 | GXGCNFG1—Window 1 Configuration | R/W | 0x0000_0000 | 22.3.2.1/22-870 |
| 0x14 | GXGBASE1—Window 1 Destination Base Address | R/W | 0x0000_0000 | 22.3.2.2/22-871 |
| 0x18 | GXGFRST1—Window 1 First Address | R/W | 0x0000_0000 | 22.3.2.3/22-872 |
| 0x1C | GXGLAST1—Window 1 Last Address | R/W | 0x0000_0000 | 22.3.2.4/22-872 |
| 0x20 | GXGCNFG2—Window 2 Configuration | R/W | 0x0000_0000 | 22.3.2.1/22-870 |
| 0x24 | GXGBASE2—Window 2 Destination Base Address | R/W | 0x0000_0000 | 22.3.2.2/22-871 |
| 0x28 | GXGFRST2—Window 2 First Address | R/W | 0x0000_0000 | 22.3.2.3/22-872 |
| 0x2C | GXGLAST2—Window 2 Last Address | R/W | 0x0000_0000 | 22.3.2.4/22-872 |
| 0x30 | GXGCNFG3—Window 3 Configuration | R/W | 0x0000_0000 | 22.3.2.1/22-870 |
| 0x34 | GXGBASE3—Window 3 Destination Base Address | R/W | 0x0000_0000 | 22.3.2.2/22-871 |
| 0x38 | GXGFRST3—Window 3 First Address | R/W | 0x0000_0000 | 22.3.2.3/22-872 |
| 0x3C | GXGLAST3—Window 3 Last Address | R/W | 0x0000_0000 | 22.3.2.4/22-872 |

**Table 22-2. GXG Memory Map (continued)**

| Offset | Register | Access | Reset Value | Section/Page |
|---|---|---|---|---|
| 0x40 | GXGSTRIDE - GFX2D Stride Setting | R/W | 0x0000_0001 | 22.3.2.5/22-873 |
| 0x3FFF | Reserved | | | |

## 22.3.2 Registers Description

This section describes the GXG registers and their bit fields.

### 22.3.2.1 Window Configuration (GXGCNFG0-3)

Offset 0x00 (GXGCNFG0)
0x10 (GXGCNFG1)
0x20 (GXGCNFG2)
0x30 (GXGCNFG3)

Access: User read/write

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | RWD | RHW | RBY | 0 | WWD | WHW | WBY | ALPHA | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MODE | | 0 | 0 | STRIDE | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-2. Window Configuration (GXGCNFG0-3)**

**Table 22-3. Window Configuration (GXGCNFG0-3) Field Description**

| Field | Description |
|---|---|
| 31 | Reserved. |
| 30<br>RWD | Read word-wide swap. See Section 22.4.5 for details.<br>0 Word swapper disabled.<br>1 Word swapper enabled. |
| 29<br>RHW | Read half word-wide swap. See Section 22.4.5 for details.<br>0 Halfword swapper disabled.<br>1 Halfword swapper enabled. |
| 28<br>RBY | Read byte-wide swap. See Section 22.4.5 for details.<br>0 Byte swapper disabled.<br>1 Byte swapper enabled. |
| 27 | Reserved. |
| 26<br>WWD | Write word-wide swap. See Section 22.4.5 for details.<br>0 Word swapper disabled.<br>1 Word swapper enabled. |
| 25<br>WHW | Write halfword-wide swap. See Section 22.4.5 for details.<br>0 Halfword swapper disabled.<br>1 Halfword swapper enabled. |
| 24<br>WBY | Write byte-wide swap. See Section 22.4.5 for details.<br>0 Byte swapper disabled.<br>1 Byte swapper enabled. |
| 23–16<br>ALPHA | Alpha. 8-bit constant used on read access to return the frame buffer alpha component or the 8-bit alpha buffer. |

Table 22-3. Window Configuration (GXGCNFG0-3) Field Description (continued)

| Field | Description |
|---|---|
| 15–14<br>MODE | Buffer Conversion mode.<br>00 No conversions (transparent).<br>01 Alpha buffer write suppression and constant read return.<br>10 Frame buffer color depth conversion for pixels with alpha component on last byte ([31:24]) of pixel format.<br>11 Frame buffer color depth conversion for pixels with alpha component on first byte ([7:0]) of pixel format. |
| 13–12 | Reserved. |
| 11–0<br>STRIDE | Stride. Width of the frame buffer (number of pixels) in physical RAM. |

### 22.3.2.2 Window Destination Base Address (GXGBASE0-3)

Offset 0x04 (GXGBASE0)
0x14 (GXGBASE1)
0x24 (GXGBASE2)
0x34 (GXGBASE3)

Access: User read/write

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BASE[31:16] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BASE[15:5] | | | | | | | | | | | BASE[4:0] | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 22-3. Window Destination Base Address (GXGBASE0-3)

Table 22-4. Window Destination Base Address (GXGBASE0-3) Field Descriptions

| Field | Description |
|---|---|
| 31–0<br>BASE | Destination Base Address. Translated addresses during color depth conversion are remapped to this base address. |

### 22.3.2.3 Window First Address (GXGFRST0-3)

Offset 0x08 (GXGFRST0)
0x18 (GXGFRST1)
0x28 (GXGFRST2)
0x38 (GXGFRST3)

Access: User read/write

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FIRST[31:16] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FIRST[15:5] | | | | | | | | | | | FIRST[4:0] | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-4. Window First Address (GXGFRST0-3)**

**Table 22-5. Window First Address (GXGFRST0-3) Field Descriptions**

| Field | Description |
|---|---|
| 31–0<br>FIRST | First Address. Start address boundary for window. Addresses greater than or equal (>=) to FIRST and less than (<) LAST are detected by window. |

### 22.3.2.4 Window Last Address (GXGLAST0-3)

Offset 0x0C (GXGLAST0)
0x1C (GXGLAST1)
0x2C (GXGLAST2)
0x3C (GXGLAST3)

Access: User read/write

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LAST[31:16] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LAST[15:5 | | | | | | | | | | | LAST[4:0] | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-5. Window Last Address (GXGLAST0-3)**

**Table 22-6. Window Last Address (GXGLAST0-3) Field Descriptions**

| Field | Description |
|---|---|
| 31–0<br>LAST | Last Address. End address boundary for window. Addresses greater than or equal (>=) to FIRST and less than (<) LAST are detected by window. |

### 22.3.2.5 GFX2D Stride Setting (GXGSTRIDE)

Offset 0x40 Access: User read/write

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

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

**Figure 22-6. GFX2D Stride Setting (GXGSTRIDE)**

**Table 22-7. GFX2D Stride Setting (GXGSTRIDE) Field Descriptions**

| Field | Description |
|---|---|
| 2-0<br>STRIDE | Configuration of stride used by the GXG to match the setting applied to the GFX2D in an OpenVG driver.<br>3'b000 - 4096<br>3'b001 – 2048<br>3'b010 – 1024<br>3'b011 – 512<br>3'b100 – 256<br>3'b101 - 128<br>3'b110 - 64 |

## 22.4 Functional Description

### 22.4.1 IPS to AHB bridge

This bridge acts as an interface between the IP Skyblue specification and the AHB bus. The IPS signals are connected to the AIPS block and the AHB signals are connected to the GFX2D 32-bit slave port.

A transfer error is asserted with an access to an IPS address offset greater than or equal to 0x800.

### 22.4.2 AXI to AHB bridge

This unidirectional bridge converts 64-bit AXI transactions into appropriate 64-bit AHB transactions and handles the multiplexing of the data channels. It has an AXI slave port as an input port and an AHB-Lite master port as an output port.

### 22.4.3 1x2 AXI bus matrix

The AXI bus matrix enables the GFX2D 64-bit master port to communicate with two AXI slave ports. It has active transactions to only a single slave at a time and is responsible for forwarding all AXI transactions in the same order that the transactions were initiated.

Table 22-8 shows the default parameter settings for the AXI bus matrix.

**Table 22-8. Bus matrix default parameters**

| Slave Ports | Regions | Start Address | End Address | Mapped Regions |
|---|---|---|---|---|
| Slave 0 (DRAM Controller) | 00 | 0x0000_0000 | 0x1FFF_FFFF | No |
| | 01 | 0x2000_0000 | 0x3FFF_FFFF | Yes |
| | 02 | 0xA000_0000 | 0xBFFF_FFFF | Yes |
| | 03 | 0xC000_0000 | 0xFFFF_FFFF | No |
| Slave 1 (Cross Bar Switch) | 10 | 0x4000_0000 | 0x5FFF_FFFF | Yes |
| | 11 | 0x6000_0000 | 0x7FFF_FFFF | Yes |
| | 12 | 0x8000_0000 | 0x8FFF_FFFF | Yes |
| | 13 | 0x9000_0000 | 0x9FFF_FFFF | No |

### 22.4.4 Address Filter

Since the type of access (which internal client) is not tagged by the GFX2D master port, an address filter is used to screen for addresses that are within four programmable address windows. Transactions with detected addresses are modified according to the configuration parameters of the respective window. Transactions with addresses that are outside all of the windows proceed without modifications.

If address regions overlap, window 0 has priority over window 1, which has priority over window 2, which has priority over window 3. A window is disabled if LAST address is programmed to be less than or equal to FIRST address.

### 22.4.5 Byte Swapper

The address filter is used to screen for transactions that need byte swapping. The purpose of the byte swapper is to provide endianness compatibility across platforms. Transactions with detected addresses are modified with the byte swapping parameters of the respective window. Transactions with addresses that are outside all of the windows proceed without modifications.

The byte swapper is onlyThe byte swapper contains three cascaded multiplexers for the read data bus and three cascaded multiplexers for the write data bus and byte strobes. The three type of multiplexers are:

1. Byte-wide swapper. See Figure 22-7.
2. Halfword-wide swapper. See Figure 22-8.
3. Word-wide swapper. See Figure 22-9.




**Figure 22-7. Byte-wide Swap**

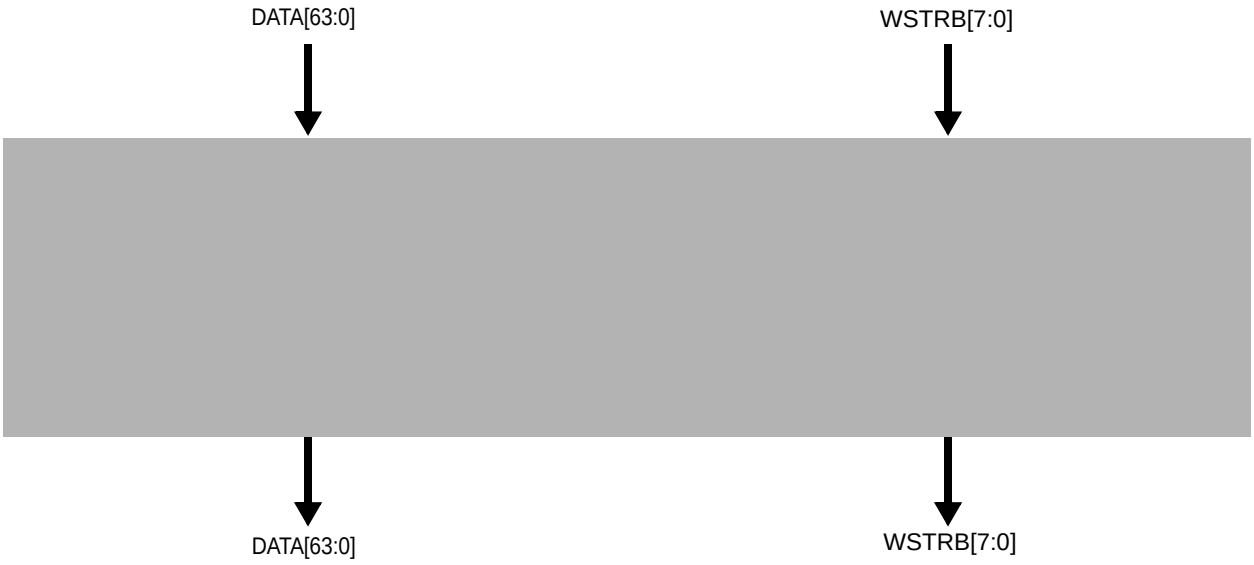



**Figure 22-8. Halfword-wide Swap**

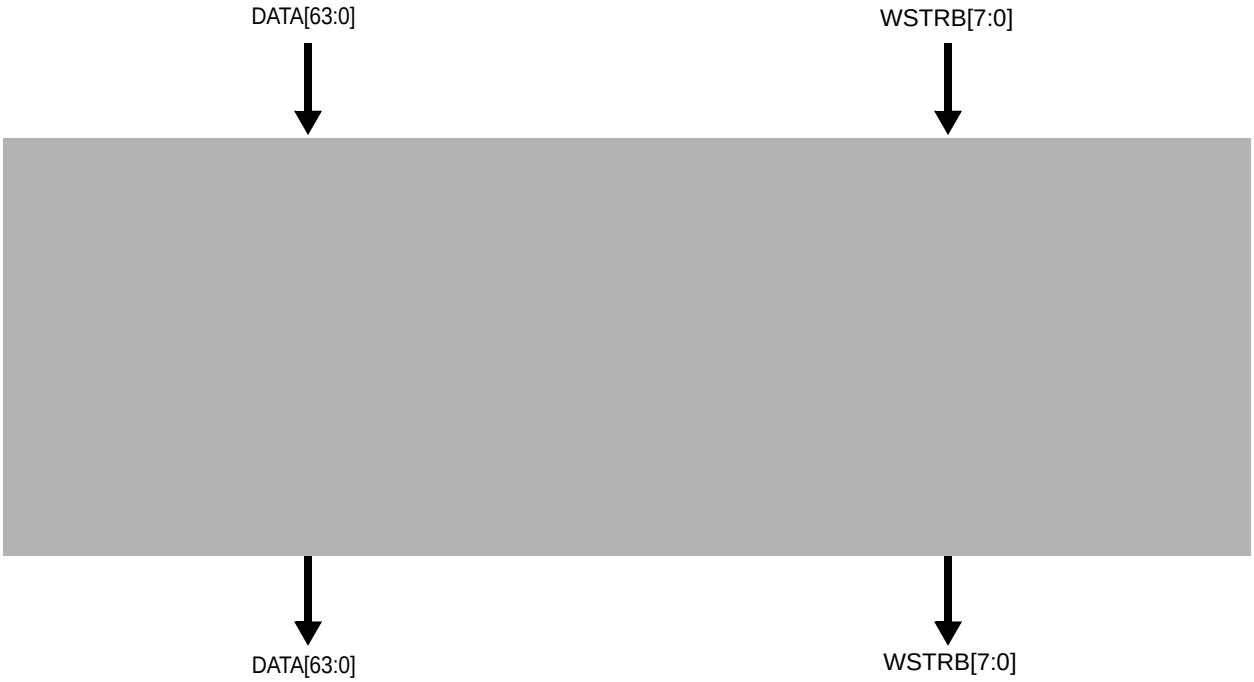



**Figure 22-9. Word-wide Swap**

Each swapper can be enabled separately. See Table 22-3 for the description of the byte swapper enable bits. Table 22-9 shows the mapping of the read or write data bus. Table 22-10 shows the mapping of the write byte strobes.

**Table 22-9. 64-bit data bus mapping**

| Input data | | | [63:56] | [55:48] | [47:40] | [39:32] | [31:24] | [23:16] | [15:8] | [7:0] |
|---|---|---|---|---|---|---|---|---|---|---|
| **WD** | **HW** | **BY** | **Output data** | | | | | | | |
| 0 | 0 | 0 | [63:56] | [55:48] | [47:40] | [39:32] | [31:24] | [23:16] | [15:8] | [7:0] |
| 0 | 0 | 1 | [55:48] | [63:56] | [39:32] | [47:40] | [23:16] | [31:24] | [7:0] | [15:8] |
| 0 | 1 | 0 | [47:40] | [39:32] | [63:56] | [55:48] | [15:8] | [7:0] | [31:24] | [23:16] |
| 0 | 1 | 1 | [39:32] | [47:40] | [55:48] | [63:56] | [7:0] | [15:8] | [23:16] | [31:24] |
| 1 | 0 | 0 | [31:24] | [23:16] | [15:8] | [7:0] | [63:56] | [55:48] | [47:40] | [39:32] |
| 1 | 0 | 1 | [23:16] | [31:24] | [7:0] | [15:8] | [55:48] | [63:56] | [39:32] | [47:40] |
| 1 | 1 | 0 | [15:8] | [7:0] | [31:24] | [23:16] | [47:40] | [39:32] | [63:56] | [55:48] |
| 1 | 1 | 1 | [7:0] | [15:8] | [23:16] | [31:24] | [39:32] | [47:40] | [55:48] | [63:56] |

**Table 22-10. Write byte strobes mapping**

| Input Write Byte Strobes | | | WSTRB[7] | WSTRB[6] | WSTRB[5] | WSTRB[4] | WSTRB[3] | WSTRB[2] | WSTRB[1] | WSTRB[0] |
|---|---|---|---|---|---|---|---|---|---|---|
| **WD** | **HW** | **BY** | **Output Write Byte Strobes** | | | | | | | |
| 0 | 0 | 0 | WSTRB[7] | WSTRB[6] | WSTRB[5] | WSTRB[4] | WSTRB[3] | WSTRB[2] | WSTRB[1] | WSTRB[0] |
| 0 | 0 | 1 | WSTRB[6] | WSTRB[7] | WSTRB[4] | WSTRB[5] | WSTRB[2] | WSTRB[3] | WSTRB[0] | WSTRB[1] |
| 0 | 1 | 0 | WSTRB[5] | WSTRB[4] | WSTRB[7] | WSTRB[6] | WSTRB[1] | WSTRB[0] | WSTRB[3] | WSTRB[2] |
| 0 | 1 | 1 | WSTRB[4] | WSTRB[5] | WSTRB[6] | WSTRB[7] | WSTRB[0] | WSTRB[1] | WSTRB[2] | WSTRB[3] |
| 1 | 0 | 0 | WSTRB[3] | WSTRB[2] | WSTRB[1] | WSTRB[0] | WSTRB[7] | WSTRB[6] | WSTRB[5] | WSTRB[4] |
| 1 | 0 | 1 | WSTRB[2] | WSTRB[3] | WSTRB[0] | WSTRB[1] | WSTRB[6] | WSTRB[7] | WSTRB[4] | WSTRB[5] |
| 1 | 1 | 0 | WSTRB[1] | WSTRB[0] | WSTRB[3] | WSTRB[2] | WSTRB[5] | WSTRB[4] | WSTRB[7] | WSTRB[6] |
| 1 | 1 | 1 | WSTRB[0] | WSTRB[1] | WSTRB[2] | WSTRB[3] | WSTRB[4] | WSTRB[5] | WSTRB[6] | WSTRB[7] |

### 22.4.6 Frame buffer color depth converter

To save space in on-chip GRAM, the GXG implements a color depth converter to reduce GFX2D 32-bpp frame buffers to 24-bpp before storing into RAM. Since conversion is only done for frame buffers, the GXG uses the address filter to detect frame buffer access. Each window of the address filter can be enabled for color depth conversion by programming the respective MODE bits. The GXG supports conversions only for 32-byte transactions with 16-byte aligned input addresses. The GFX2D stride must match the GXG stride which is adjusted using the GXGSTRIDE register.

During the address phase of the transaction, the input address from the GFX2D is translated to an output address in physical RAM according to Equation 22-1.

***Eqn. 22-1***

$$OFS[23:0] = INPADR[31:2] - FIRSTn[31:2]$$

$$OUTADR = BASEn + 4 \times (OFS[23:(12 + Z)] \times STRIDEn) + 3 \times OFS[11:0]$$

| | | |
|---|---|---|
| INPADR | – | Input address from GFX2D to be translated. |
| FIRSTn | – | Window first address. See Section 22.3.2.3. |
| OFS | – | Intermediate offset calculation |
| BASEn | – | Destination base address. See Section 22.3.2.2. |
| STRIDEn | – | Width of the frame buffer in physical RAM. See Section 22.3.2.1. |
| OUTADR | – | Translated output address to physical RAM. |
| Z | – | Value of GXGSTRIDE register (0 to 6). See Section 22.3.2.5. |

During the data phase of write transactions, the 8-bit alpha component of each 32-bit pixel is suppressed, and the byte strobes of partially written double-words (first and last double-words depending on alignment) are masked.

Table 22-11 and Table 22-12 show the color depth conversion mapping. The GFX2D address shown is relative to the FIRST address and the physical address shown is relative to the BASE address.

During the data phase of read transactions, the programmable 8-bit ALPHA constant is returned in the alpha component byte lane. If MODE[1] is set to enable color depth conversion, MODE[0] indicates which byte is the alpha component (first or last byte) in the 32-bit pixel value.

**Table 22-11. Color Depth Conversion mapping (MODE[0] = 0)**

| Offset Address | 64-bit GFX2D Data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | [7:0] | [15:8] | [23:16] | [31:24] | [39:32] | [47:40] | [55:48] | [63:56] |
| **0x00** | $R_0$ | $G_0$ | $B_0$ | $A_0$ | $R_1$ | $G_1$ | $B_1$ | $A_1$ |
| **0x08** | $R_2$ | $G_2$ | $B_2$ | $A_2$ | $R_3$ | $G_3$ | $B_3$ | $A_3$ |
| **0x10** | $R_4$ | $G_4$ | $B_4$ | $A_4$ | $R_5$ | $G_5$ | $B_5$ | $A_5$ |
| **0x18** | $R_6$ | $G_6$ | $B_6$ | $A_6$ | $R_7$ | $G_7$ | $B_7$ | $A_7$ |
| **–** | – | | | | | | | |
| **0x50** | $R_{20}$ | $G_{20}$ | $B_{20}$ | $A_{20}$ | $R_{21}$ | $G_{21}$ | $B_{21}$ | $A_{21}$ |
| **0x58** | $R_{22}$ | $G_{22}$ | $B_{22}$ | $A_{22}$ | $R_{23}$ | $G_{23}$ | $B_{23}$ | $A_{23}$ |
| **0x60** | $R_{24}$ | $G_{24}$ | $B_{24}$ | $A_{24}$ | $R_{25}$ | $G_{25}$ | $B_{25}$ | $A_{25}$ |
| **0x68** | $R_{26}$ | $G_{26}$ | $B_{26}$ | $A_{26}$ | $R_{27}$ | $G_{27}$ | $B_{27}$ | $A_{27}$ |
| **–** | – | | | | | | | |
| **0xE0** | $R_{56}$ | $G_{56}$ | $B_{56}$ | $A_{56}$ | $R_{57}$ | $G_{57}$ | $B_{57}$ | $A_{57}$ |
| **0xE8** | $R_{58}$ | $G_{58}$ | $B_{58}$ | $A_{58}$ | $R_{59}$ | $G_{59}$ | $B_{59}$ | $A_{59}$ |
| **0xF0** | $R_{60}$ | $G_{60}$ | $B_{60}$ | $A_{60}$ | $R_{61}$ | $G_{61}$ | $B_{61}$ | $A_{61}$ |
| **0xF8** | $R_{62}$ | $G_{62}$ | $B_{62}$ | $A_{62}$ | $R_{63}$ | $G_{63}$ | $B_{63}$ | $A_{63}$ |

| Offset Address | 64-bit Physical RAM Data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | [7:0] | [15:8] | [23:16] | [31:24] | [39:32] | [47:40] | [55:48] | [63:56] |
| **0x00** | $R_0$ | $G_0$ | $B_0$ | $R_1$ | $G_1$ | $B_1$ | $R_2$ | $G_2$ |
| **0x08** | $B_2$ | $R_3$ | $G_3$ | $B_3$ | $R_4$ | $G_4$ | $B_4$ | $R_5$ |
| **0x10** | $G_5$ | $B_5$ | $R_6$ | $G_6$ | $B_6$ | $R_7$ | $G_7$ | $B_7$ |
| **0x18** | $R_8$ | $G_8$ | $B_8$ | $R_9$ | $G_9$ | $B_9$ | $R_{10}$ | $G_{10}$ |
| **–** | – | | | | | | | |
| **0x38** | $B_{18}$ | $R_{19}$ | $G_{19}$ | $B_{19}$ | $R_{20}$ | $G_{20}$ | $B_{20}$ | $R_{21}$ |
| **0x40** | $G_{21}$ | $B_{21}$ | $R_{22}$ | $G_{22}$ | $B_{22}$ | $R_{23}$ | $G_{23}$ | $B_{23}$ |
| **0x48** | $R_{24}$ | $G_{24}$ | $B_{24}$ | $R_{25}$ | $G_{25}$ | $B_{25}$ | $R_{26}$ | $G_{26}$ |
| **0x50** | $B_{26}$ | $R_{27}$ | $G_{27}$ | $B_{27}$ | $R_{28}$ | $G_{28}$ | $B_{28}$ | $R_{29}$ |
| **–** | – | | | | | | | |
| **0xA8** | $R_{56}$ | $G_{56}$ | $B_{56}$ | $R_{57}$ | $G_{57}$ | $B_{57}$ | $R_{58}$ | $G_{58}$ |
| **0xB0** | $B_{58}$ | $R_{59}$ | $G_{59}$ | $B_{59}$ | $R_{60}$ | $G_{60}$ | $B_{60}$ | $R_{61}$ |
| **0xB8** | $G_{61}$ | $B_{61}$ | $R_{62}$ | $G_{62}$ | $B_{62}$ | $R_{63}$ | $G_{63}$ | $B_{63}$ |
| **0xC0** | $R_{64}$ | $G_{64}$ | $B_{64}$ | $R_{65}$ | $G_{65}$ | $B_{65}$ | $R_{66}$ | $G_{66}$ |

**Table 22-12. Color Depth Conversion mapping (MODE[0] = 1)**

| Offset Address | 64-bit GFX2D Data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | [7:0] | [15:8] | [23:16] | [31:24] | [39:32] | [47:40] | [55:48] | [63:56] |
| – | – | | | | | | | |
| 0x30 | $A_{12}$ | $R_{12}$ | $G_{12}$ | $B_{12}$ | $A_{13}$ | $R_{13}$ | $G_{13}$ | $B_{13}$ |
| 0x38 | $A_{14}$ | $R_{14}$ | $G_{14}$ | $B_{14}$ | $A_{15}$ | $R_{15}$ | $G_{15}$ | $B_{15}$ |
| 0x40 | $A_{16}$ | $R_{16}$ | $G_{16}$ | $B_{16}$ | $A_{17}$ | $R_{17}$ | $G_{17}$ | $B_{17}$ |
| 0x48 | $A_{18}$ | $R_{18}$ | $G_{18}$ | $B_{18}$ | $A_{19}$ | $R_{19}$ | $G_{19}$ | $B_{19}$ |
| – | – | | | | | | | |
| 0x80 | $A_{32}$ | $R_{32}$ | $G_{32}$ | $B_{32}$ | $A_{33}$ | $R_{33}$ | $G_{33}$ | $B_{33}$ |
| 0x88 | $A_{34}$ | $R_{34}$ | $G_{34}$ | $B_{34}$ | $A_{35}$ | $R_{35}$ | $G_{35}$ | $B_{35}$ |
| 0x90 | $A_{36}$ | $R_{36}$ | $G_{36}$ | $B_{36}$ | $A_{37}$ | $R_{37}$ | $G_{37}$ | $B_{37}$ |
| 0x98 | $A_{38}$ | $R_{38}$ | $G_{38}$ | $B_{38}$ | $A_{39}$ | $R_{39}$ | $G_{39}$ | $B_{39}$ |
| – | – | | | | | | | |
| 0xB0 | $A_{44}$ | $R_{44}$ | $G_{44}$ | $B_{44}$ | $A_{45}$ | $R_{45}$ | $G_{45}$ | $B_{45}$ |
| 0xB8 | $A_{46}$ | $R_{46}$ | $G_{46}$ | $B_{46}$ | $A_{47}$ | $R_{47}$ | $G_{47}$ | $B_{47}$ |
| 0xC0 | $A_{48}$ | $R_{48}$ | $G_{48}$ | $B_{48}$ | $A_{49}$ | $R_{49}$ | $G_{49}$ | $B_{49}$ |
| 0xC8 | $A_{50}$ | $R_{50}$ | $G_{50}$ | $B_{50}$ | $A_{51}$ | $R_{51}$ | $G_{51}$ | $B_{51}$ |

| Offset Address | 64-bit Physical RAM Data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | [7:0] | [15:8] | [23:16] | [31:24] | [39:32] | [47:40] | [55:48] | [63:56] |
| – | – | | | | | | | |
| 0x20 | $B_{10}$ | $R_{11}$ | $G_{11}$ | $B_{11}$ | $R_{12}$ | $G_{12}$ | $B_{12}$ | $R_{13}$ |
| 0x28 | $G_{13}$ | $B_{13}$ | $R_{14}$ | $G_{14}$ | $B_{14}$ | $R_{15}$ | $G_{15}$ | $B_{15}$ |
| 0x30 | $R_{16}$ | $G_{16}$ | $B_{16}$ | $R_{17}$ | $G_{17}$ | $B_{17}$ | $R_{18}$ | $G_{18}$ |
| 0x38 | $B_{18}$ | $R_{19}$ | $G_{19}$ | $B_{19}$ | $R_{20}$ | $G_{20}$ | $B_{20}$ | $R_{21}$ |
| – | – | | | | | | | |
| 0x60 | $R_{32}$ | $G_{32}$ | $B_{32}$ | $R_{33}$ | $G_{33}$ | $B_{33}$ | $R_{34}$ | $G_{34}$ |
| 0x68 | $B_{34}$ | $R_{35}$ | $G_{35}$ | $B_{35}$ | $R_{36}$ | $G_{36}$ | $B_{36}$ | $R_{37}$ |
| 0x70 | $G_{37}$ | $B_{37}$ | $R_{38}$ | $G_{38}$ | $B_{38}$ | $R_{39}$ | $G_{39}$ | $B_{39}$ |
| 0x78 | $R_{40}$ | $G_{40}$ | $B_{40}$ | $R_{41}$ | $G_{41}$ | $B_{41}$ | $R_{42}$ | $G_{42}$ |
| – | – | | | | | | | |
| 0x80 | $B_{42}$ | $R_{43}$ | $G_{43}$ | $B_{43}$ | $R_{44}$ | $G_{44}$ | $B_{44}$ | $R_{45}$ |
| 0x88 | $G_{45}$ | $B_{45}$ | $R_{46}$ | $G_{46}$ | $B_{46}$ | $R_{47}$ | $G_{47}$ | $B_{47}$ |
| 0x90 | $R_{48}$ | $G_{48}$ | $B_{48}$ | $R_{49}$ | $G_{49}$ | $B_{49}$ | $R_{50}$ | $G_{50}$ |
| 0x98 | $B_{50}$ | $R_{51}$ | $G_{51}$ | $B_{51}$ | $R_{52}$ | $G_{52}$ | $B_{52}$ | $R_{53}$ |

### 22.4.7 Alpha buffer write suppressor

Certain graphic accelerator operations may perform redundant memory writes. This particularly applies to alpha buffer operations. To save space in RAM, each GXG window can suppress writes into memory at addresses specified in the window. In addition reads from this window will be replaced with a fixed "alpha" value.

Program this mode by selecting the value 1 for the value of MODE in the window configuration register. The fixed "alpha" value is provided by the ALPHA bit-field in the same register.

### 22.4.8 Serial flash exclusive access

The GXG is configured to use the Exclusive Access capability of the QuadSPI module. This is provided to avoid the situation where a graphic operation is underway and another master requests a different QuadSPI operation thus disrupting the flow of data to the GXG. If a graphic operation requires access to the QuadSPI then the GXG will suspend operations until the QuadSPI modules grants exclusive access. Once access is granted the GXG will commence the graphic operation and on conclusion will release the exclusive access.

See the QuadSPI chapter for more details on the operation of the Exclusive Access function.

Table 22-13 shows the default parameter settings for the exclusive access.

**Table 22-13. Exclusive access default parameters**

| Start Address | End Address |
|---|---|
| 0x8000_0000 | 0x8FFF_FDFF |

This page is intentionally left blank.

# Chapter 23
# Graphics Static RAM (GSRAM)

## 23.1 Introduction

This document provides a description of the graphics static RAM (GSRAM). Contained in this document is a user's level view of performance and overall block functionality descriptions.

### 23.1.1 Overview

The GSRAM on this device consists of a block of RAM and an asynchronous reset, RAM array controller (PRAM2P) with dual AHB input ports. The module acts as a dual ported memory controller interfacing two AHB input ports and a RAM array. This controller has specific enhancements from previous controllers including the two ports, connection to a half speed memory array, port specific configurability, and fill function to put the RAM into a known state without using any system bus bandwidth.

Figure 23-1 shows the GSRAM integrated into a basic platform.

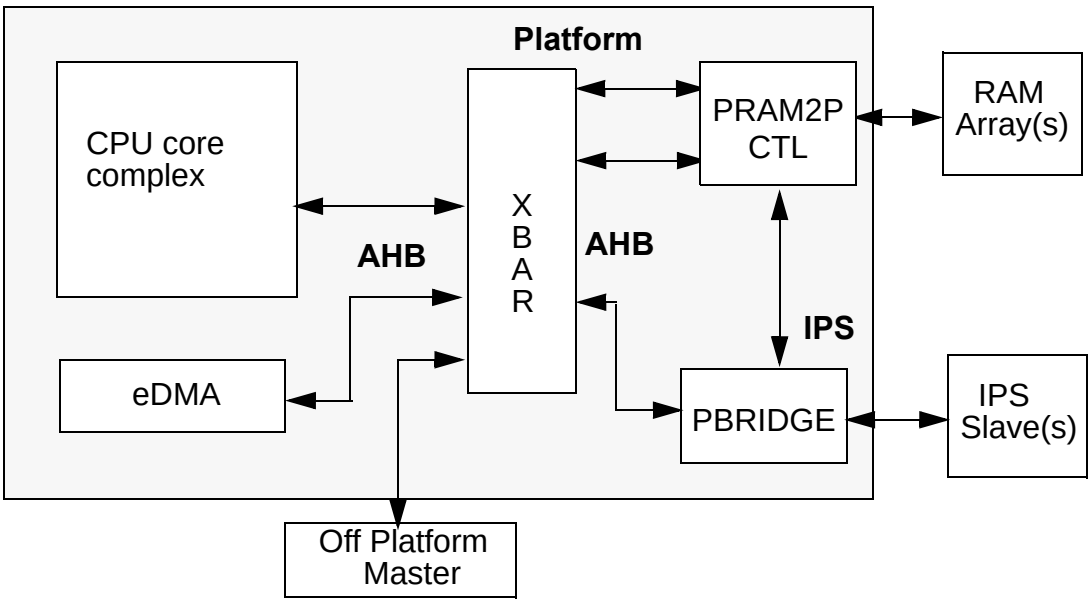



**Figure 23-1. Simplified system block diagram**

In Figure 23-1 the platform bus is the AMBA AHB, the XBAR is the AHB crossbar switch (arbiter), and the PBRIDGE is the AHB to IPS protocol interface.

### 23.1.2 Features

- Supports up to 8 MB of RAM
- Dual 64-bit AHB input ports
- 128-bit slower RAM array interface

- Support for byte (8-bit), half word (16-bit), word (32-bit) and double word (64-bit) access sizes
- Full support for AMBA v6 extensions related to unaligned accesses
  - Includes support for non-contiguous byte strobes on writes
- Independent data buffers (one per AHB port) for maximum system performance
  - Optimized for burst transfers (read + write)
  - Programmable read prefetch capabilities
- 32-bit IPS interface for access to program model
  - Arbitration control giving priority to port 0 or port 1 or round robin
- Port 0 and 1 prefetch control
- Fill function for memory initialization
  - Region of initialization is configurable with start and end addresses on modulo 32 byte boundaries
  - Initialization can be a load of all-zeroes or all-ones
  - Status register available for the system to monitor fill operation

### 23.1.3 Modes of operation

The GSRAM supports all modes of operation. If a standby or low power mode is desired, it should be taken care of off platform.

## 23.2 External signal description

This module has no external signals.

### 23.2.1 Memory map

The GSRAM programming model consists of six 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid and will return an error. Other types of invalid accesses include: writes to read only registers, and accesses to reserved addresses.

**Table 23-1. GSRAM memory map**

| Address offset | Register | Access | Reset value | Location |
|---|---|---|---|---|
| **General Registers** | | | | |
| 0x0 | PRAM2P_CR - Control Register | R/W | 0x0000_0000 | on page 885 |
| 0x4 | PRAM2P_SR—Status register | R | 0x0000_0000 | on page 886 |
| 0x8 | PRAM2P_BEG—Beginning address for Fill register | R/W | 0x0000_0000 | on page 886 |
| 0xC | PRAM2P_END—End Address for Fill register | R/W | 0x0000_0000 | on page 887 |
| 0x10 | PRAM2P_FIL—Fill register | R/W | 0x0000_0000 | on page 887 |

## 23.2.2 Register descriptions

The following sections detail the individual registers within the GSRAM programming model.

### 23.2.2.1 PRAM2P Control Register (PRAM2P_CR)

The PRAM2P_CR register contains fields for controlling and configuring the 2 slave ports on the GSRAM.

Offset 0x0 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PFE1 | PFE0 | 0 | 0 | PRI | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 23-2. PRAM2P Control Register(PRAM2P_CR)**

**Table 23-2. PRAM2P_CR Field Descriptions**

| Field | Description |
|---|---|
| PFE*n* | Port *n* Prefetch Enable.<br>0 Array prefetching for port n disabled.<br>1 Array prefetching for port n enabled. |
| PRI | Prioritization of slave ports' access to the RAM array.<br>00 Round Robin.<br>01 Port 0 has priority over port 1.<br>10 Port 1 has priority over port 0.<br>11 Reserved. |

### 23.2.2.2 PRAM2P Status Register (PRAM2P_SR)

The PRAM2P_SR is s read only register containing the RAM array busy flag. All writes to this register will return an error.

Offset 0x4 Access: User read

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

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

**Figure 23-3. PRAM2P Control Register(PRAM2P_CR)**

**Table 23-3. PRAM2P_CR Field Descriptions**

| Field | Description |
|---|---|
| BUSY | When a Fill operation is in progress, the portion of the array that is “to be filled” is unavailable and accesses to those area will be returned with an error. Areas not targeted as well as areas that have already been filled are accessible and can interrupt the Fill operation's access to the array. The BUSY bit indicates that a Fill operation is in progress.<br>0 Array is available.<br>1 A Fill operation is in progress. |

### 23.2.2.3 PRAM2P Fill Region Begin Address Register (PRAM2P_BEG)

Offset 0x8 Access: User read/write

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

**Figure 23-4. PRAM2P Fill Region Begin Address Register (PRAM2P_BEG)**

**Table 23-4. PRAM2P_BEG Field Descriptions**

| Field | Description |
|---|---|
| BEG_ ADDR [23:4] | Fill operation will begin at GSRAM base address + {BEG_ADDR[23:4],4’b0}. While the GSRAM is busy filling indicated by the BUSY bit in the PRAM2P_SR, this register will not be writable. Any write access to this register will return an error. |

### 23.2.2.4 PRAM2P Fill Region End Address Register (PRAM2P_END)

Offset 0xC

Access: User read/write

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

**Figure 23-5. PRAM2P Fill Region End Address Register (PRAM2P_END)**

**Table 23-5. PRAM2P_END Field Descriptions**

| Field | Description |
|---|---|
| END_ADDR [23:4] | Fill operation will end at GSRAM base address + {END_ADDR[23:4],4'b0}. While the GSRAM is buy filling indicated by the BUSY bit in the PRAM2P_SR, this register will not be writable. Any write access to this register will return an error. |

### 23.2.2.5 PRAM2P Fill Register (PRAM2P_FIL)

The PRAM2P Fill Register is used to begin the initialization of the RAM array. The Array can be filled with either ones or zeroes depending on the FILL bit.

Offset 0x10

Access: User read/write

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

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

**Figure 23-6. PRAM2P Fill Register(PRAM2P_FIL)**

**Table 23-6. PRAM2P_FIL Field Descriptions**

| Field | Description |
|---|---|
| FILL | FILL is a writable bit which determines whether the memory is filled with ones or zeroes. |
| START | When START is written 0x1, the fill operation will be begin writing 0x0s or 0x1s to the RAM array from the address in PRAM2P_BEG,23.2.2.3, to PRAM2P_END, 23.2.2.4. While filling, all accesses to the parts of the array that are "to be filled" will return an error. Areas not targeted as well as areas that have already been filled are accessible and can interrupt the Fill operation's access to the array This register will also be read only until the fill operation is complete returning an error on all writes. |

## 23.3 Functional description

The GSRAM has three main ports: two AMBA-AHB slave ports and one RAM array master port. The slave ports are running at the same frequency as the rest of the system. To help system speed with larger RAM arrays, the RAM array is operating at a slower speed. With two active ports and a slower array clock, the access times can vary. Assuming that the other port is idle and the array is running at half speed, each port should see the performance described in Table 23-7.

**Table 23-7. Ideal Access performance for**

| Access Type | Size | Cycles[1] |
|---|---|---|
| Read | Single 8,16,32, and 64-bit | 2 or 3 (1 if read hits buffer) |
| | 4-beat bursts | 2-1-1-1 or 3-1-1-1 |
| | sequential 4-beat bursts with prefetching | 2-1-1-1-1-1-1..... or 3-1-1-1-1-1-1-1... |
| Write | Single 8, 16, 32, and 64-bit | 1 |
| | 4-beat bursts | 1-1-1-1 or 2-1-1-1 |

1 Because of the slower array clock, the AMBA-AHB access may come in out of phase with the array side causing an extra cycle in the initial access.

The PRAM2P_CR includes configurability bits to help customize the GSRAM performance for a particular application. This includes giving one port priority over the other and enabling prefetching on reads.

Table 23-8 shows which masters are assigned to which ports for the MPC5645S. If a particular port is given priority over the other, it will have priority access to the array. If a burst on the non prioritized port has begun however, it will not be interrupted by a read from the prioritized port and it will be held until the burst is complete.

**Table 23-8. Master assignments**

| Port 0 | Port 1 |
|---|---|
| GFX2D | e200z4d instruction port |
| | e200z4d data port |
| | eDMA |
| | VIU2 |
| | DCULite |
| | DCU3 |

Each port has a read buffer which will be filled with 128 bits on each read. If prefetching is enabled in the PRAM2P_CR, the port will access the next location in the array and load it into a prefetch buffer. If that location is accessed, it will move into the read buffer, and the prefetch buffer will go to the array for the next location and so on. If there are a lot of different masters accessing the same port, the prefetch feature

may actually cause a performance degradation. For this reason the PFEn bits in the PRAM2P_CR allow enabling and disabled of the prefetch feature.

When bursting a total size larger than the read buffer, the consequent accesses needed from the array will be prefetched whether prefetching is enabled or not. In the case of a wrap burst, the correct prefetch is accessed whether the next access is above or below the current read location.

The GSRAM provides a memory initialization function with the register PRAM2P_FIL. The value in the FILL bit will be loaded into every bit in the RAM array from PRAM2P_BEG to PRAM2P_END. If PRAM2P_END is less than or equal to PRAM2P_BEG, the command will be ignored. If PRAM2P_END is greater than the memory size, the command will also be ignored. During a fill operation, all accesses to locations in the array that will be filled will be returned with an error. As the fill progresses, the areas which have been filled will be accessible again. All writes to PRAM2P_BEG, PRAM2P_END, or PRAM2P_FIL will also return an error during a fill. If a port is currently busy with an access when the fill request is started, the GSRAM will wait until both ports are idle before beginning the fill. Software should read the PRAM2P_SR register to confirm availability before accessing the RAM.

## 23.4 Initialization information

Out of reset, all of the RAM array entries are unknown. The best way to initialize the RAM array to a known state is to use the fill function provided in the GSRAM to initialize it to ones or zeroes.

## 23.5 Application information

One should take care in configuring the GSRAM using the PRAM2P_CR. If priority is given to one port, all of the masters on the adjacent port could be starved out of the RAM and left waiting. Along the same lines, if prefetch is enabled on a port which has several masters accessing several areas, there may be a lot of excessive accesses to the RAM array slowing down the other port’s access to the RAM array.

This page is intentionally left blank.

# Chapter 24
# IEEE 1149.1 Test Access Port Controller (JTAGC)

## 24.1 Introduction

The JTAG port of the device consists of three inputs and one output. These pins include test data input (TDI), test data output (TDO), test mode select (TMS), and test clock input (TCK). TDI, TDO, TMS, and TCK are compliant with the IEEE 1149.1-2001 standard and are shared with the NDI through the test access port (TAP) interface.

IEEE 1149.7 (cJTAG) is not supported on this device.

## 24.2 Block diagram

Figure 24-1 is a block diagram of the JTAG Controller (JTAGC).

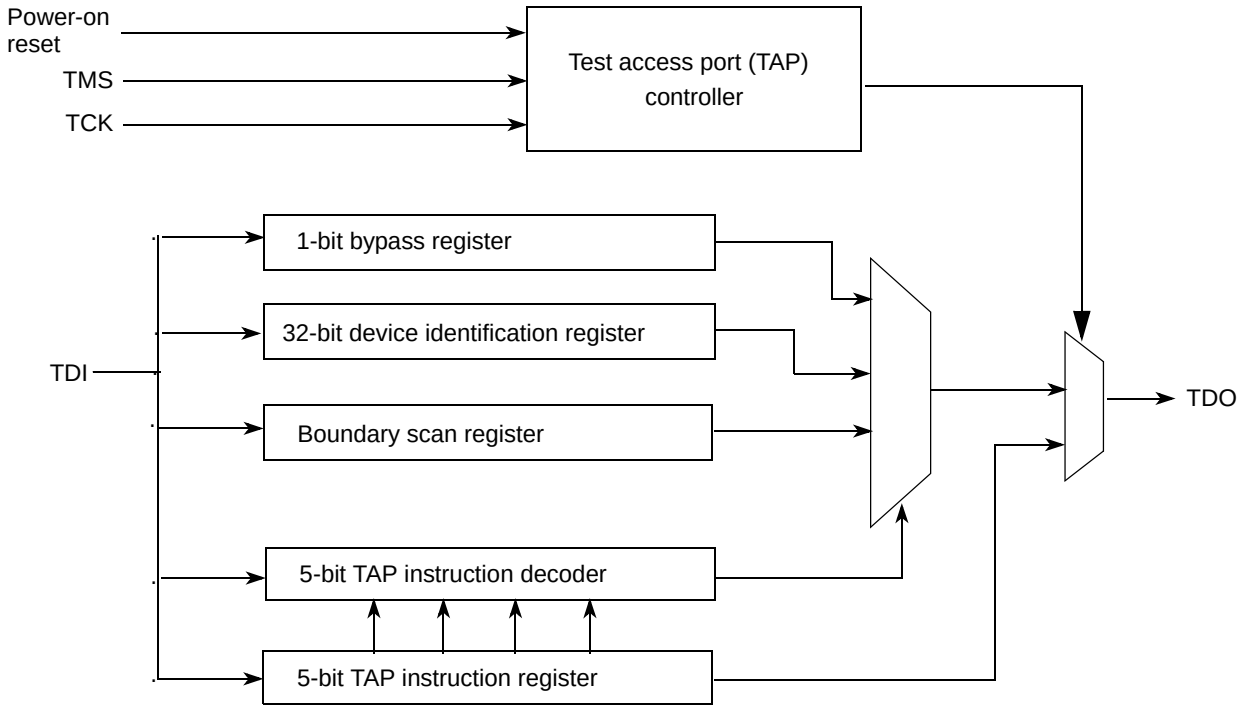



**Figure 24-1. JTAG Controller Block Diagram**

## 24.3 Overview

The JTAGC provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. In addition, instructions can be executed that allow the Test Access Port (TAP) to be shared with other modules on the MCU. All data input to and output from the JTAGC is communicated in serial format.

## 24.4 Features

The JTAGC is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
- 4 pins (see Section 24.6, “External signal description”)
  - TDI
  - TMS
  - TCK
  - TDO
- A 5-bit instruction register that supports several IEEE 1149.1-2001 defined instructions, as well as several public and private MCU specific instructions
- Three test data registers, a bypass register, and a device identification register
- A TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry

## 24.5 Modes of operation

The JTAGC uses a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

### 24.5.1 Reset

The JTAGC is placed in reset when the TAP controller state machine is in the TEST-LOGIC-RESET state. The TEST-LOGIC-RESET state is entered upon the assertion of the power-on reset signal, or through TAP controller state machine transitions controlled by TMS. Asserting power-on reset results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the test-logic-reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered.
- The instruction register is loaded with the IDCODE instruction.

In addition, execution of certain instructions can result in assertion of the internal system reset. These instructions include EXTEST.

### 24.5.2 IEEE 1149.1-2001 defined test modes

The JTAGC supports several IEEE 1149.1-2001 defined test modes. The test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data registers that can operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is external to JTAGC but can be accessed by JTAGC TAP through EXTEST, SAMPLE, and SAMPLE/PRELOAD instructions. The functionality of each test mode is explained in more detail in Section 24.8.4, “JTAGC Instructions.”

#### 24.5.2.1 Bypass Mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

#### 24.5.2.2 TAP Sharing Mode

There are three selectable auxiliary TAP controllers that share the TAP with the JTAGC. Selectable TAP controllers include the Nexus port controller (NPC) and PLATFROM. The instructions required to grant ownership of the TAP to the auxiliary TAP controllers are ACCESS_AUX_TAP_NPC, ACCESS_AUX_TAP_ONCE, ACCESS_AUX_TAP_TCU. Instruction opcodes for each instruction are shown in Table 24-3.

When the access instruction for an auxiliary TAP is loaded, control of the JTAG pins is transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive.

For more information on the TAP controllers refer to Chapter 30, “Nexus Development Interface (NDI).”

## 24.6 External signal description

The JTAGC consists of four signals that connect to off-chip development tools and allow access to test support functions. The JTAGC signals are outlined in Table 24-1.

**Table 24-1. JTAG signal properties**

| Name | I/O | Function | Reset State |
|---|---|---|---|
| TCK[1] | I | Test clock | Pull Up |
| TDI | I | Test data in | Pull Up |
| TDO | O | Test data out | High Z |
| TMS | I | Test mode select | Pull Up |

[1] In low power mode, TCk frequency should not be greater than 16 MHz or IRC.

All 4 JTAG pins (TCK/TMS/TDI/TDO) are shared with GPIO pins, so that the software may configure these pins as input/output by programming the appropriate registers.

To ensure the proper working of JTAG, these registers have a reset value such that these pins behave as JTAG pins when the POR is lifted:

- TDI : input/pull-up
- TCK : Input/pull-up
- TMS : input/pull-up
- TDO : high-Z/pull-disabled

## 24.7 Memory map and register description

This section provides a detailed description of the JTAGC registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

### 24.7.1 Instruction Register

The JTAGC uses a 5-bit instruction register as shown in Figure 24-2. The instruction register allows instructions to be loaded into the module to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the update-IR and test-logic-reset TAP controller states. Synchronous entry into the test-logic-reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the test-logic-reset state results in asynchronous loading of the IDCODE instruction. During the capture-IR TAP controller state, the instruction shift register is loaded with the value 0b10101, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

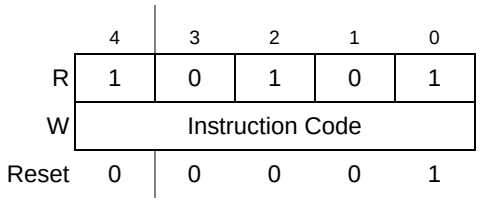



**Figure 24-2. 5-Bit Instruction Register**

### 24.7.2 Bypass Register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, or reserve instructions are active. After entry into the capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

### 24.7.3 Device Identification Register

The device identification register, shown in Figure 24-3, allows the part revision number, design center, part identification number, and manufacturer identity code to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the update-DR state.

IR[4:0]: 0_0001 (IDCODE) Access: R/O

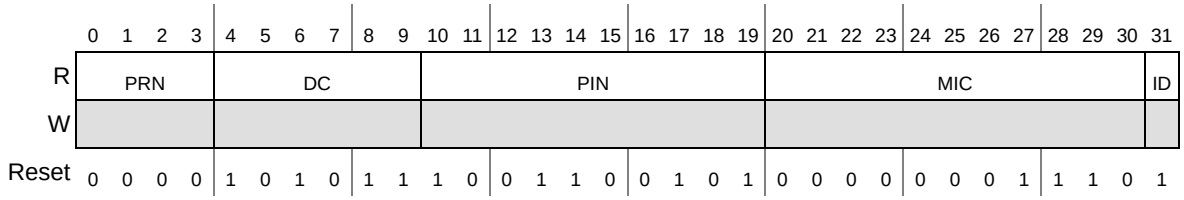



**Figure 24-3. Device Identification Register**

**Table 24-2. Device Identification Register Field Descriptions**

| Field | Description |
|---|---|
| 0–3<br>PRN | Part revision number. Contains the revision number of the device. This field changes with each revision of the device or module. |
| 4–9<br>DC | Design center. |
| 10–19<br>PIN | Part identification number. Contains the part number of the device. |
| 20–30<br>MIC | Manufacturer identity code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID for Freescale, 0xE |
| 31<br>ID | IDCODE register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1. |

### 24.7.4 Boundary Scan Register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Section 24.8.5, “Boundary Scan.” The size of the boundary scan register is 464 bits.

## 24.8 Functional description

### 24.8.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the test-logic-reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

### 24.8.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. For more detail on TAP sharing via JTAGC instructions refer to Section 24.8.4.2, “ACCESS_AUX_TAP_x Instructions.”

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in Figure 24-4. This applies for the instruction register, test data registers, and the bypass register.

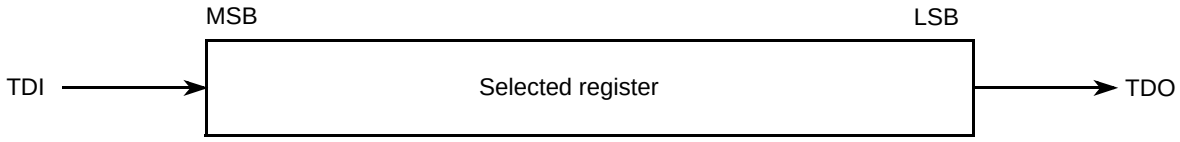
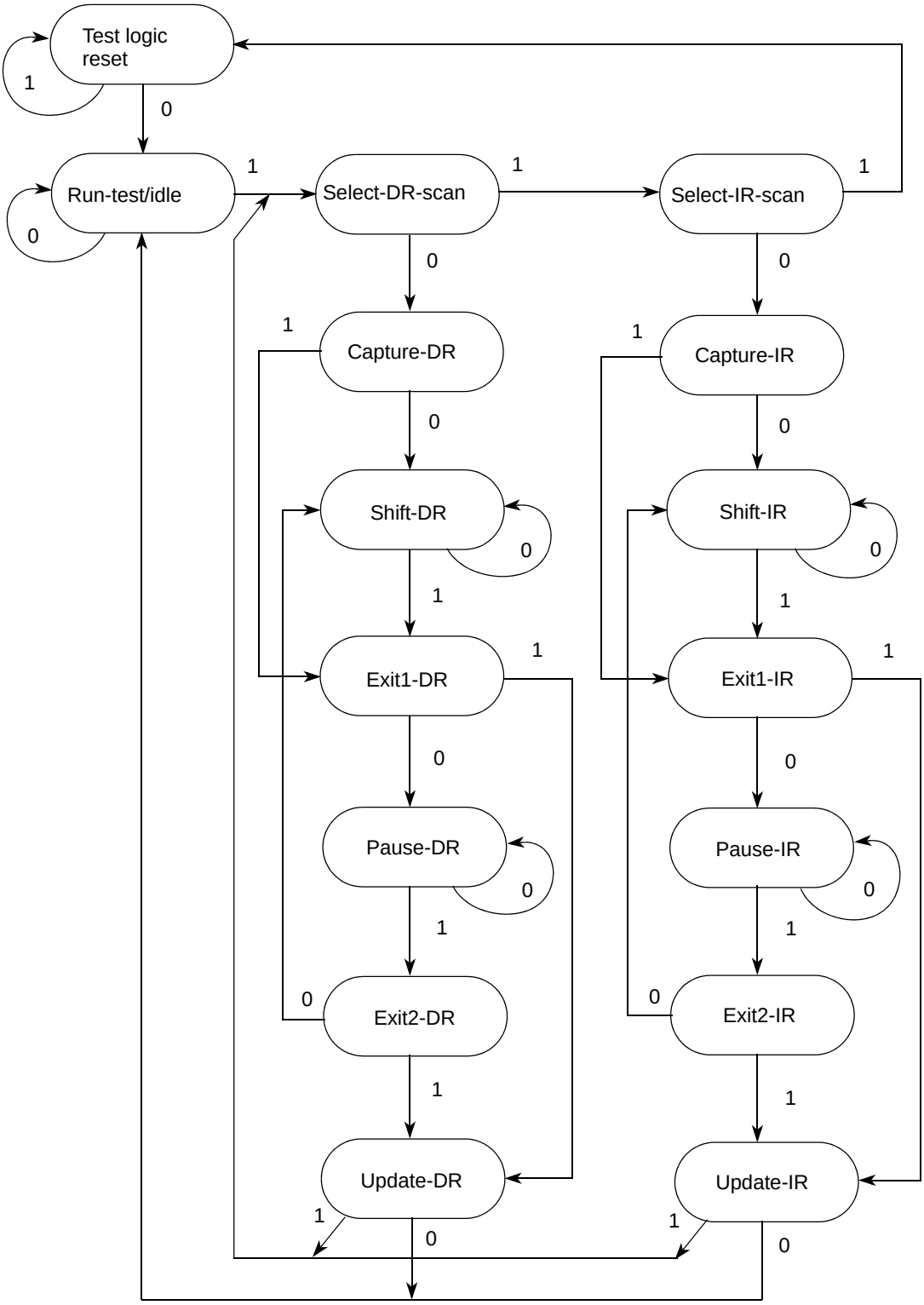



**Figure 24-4. Shifting Data Through a Register**

### 24.8.3 TAP Controller State Machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. Figure 24-5 shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal.

As Figure 24-5 shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the test-logic-reset state.




NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

**Figure 24-5. IEEE 1149.1-2001 TAP Controller Finite State Machine**

#### 24.8.3.1 Selecting an IEEE 1149.1-2001 Register

Access to the JTAGC data registers is done by loading the instruction register with any of the JTAGC instructions while the JTAGC is enabled. Instructions are shifted in via the select-IR-scan path and loaded in the update-IR state. At this point, all data register access is performed via the select-DR-scan path.

The select-DR-scan path is used to read or write the register data by shifting in the data (LSB first) during the shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting can be terminated after fetching the required number of bits.

### 24.8.4 JTAGC Instructions

This section gives an overview of each instruction, refer to the IEEE 1149.1-2001 standard for more details.

The JTAGC implements the IEEE 1149.1-2001 defined instructions listed in Table 24-3.

**Table 24-3. JTAG Instructions**

| Instruction | Code[4:0] | Instruction Summary |
|---|---|---|
| IDCODE | 00001 | Selects device identification register for shift |
| SAMPLE/PRELOAD | 00010 | Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation |
| SAMPLE | 00011 | Selects boundary scan register for shifting and sampling without disturbing functional operation |
| EXTEST | 00100 | Selects boundary scan register while applying preloaded values to output pins and asserting functional reset |
| ACCESS_AUX_TAP_TCU | 11011 | Grants the TCU ownership of the TAP |
| ACCESS_AUX_TAP_ONCE | 10001 | Grants the PLATFROM ownership of the TAP |
| ACCESS_AUX_TAP_NPC | 10000 | Grants the Nexus port controller (NPC) ownership of the TAP |
| BYPASS | 11111 | Selects bypass register for data operations |
| Factory Debug Reserved[1] | 00101<br>00110<br>01010 | Intended for factory debug only |
| Reserved[2] | All Other Codes | Decoded to select bypass register |

1 Intended for factory debug, and not customer use

2 Freescale reserves the right to change the decoding of reserved instruction codes

Table 24-4 shows the implementation for silicon cut1. By mistake, the Access to Nexus Port Controller is not using the standard PowerPC instruction.

For silicon cut2, the instruction coding will be changed to be 100% compatible with existing PowerPC.

Table 24-4. JTAG Instructions for silicon cut1

| Instruction | Code[4:0] | Instruction Summary |
|---|---|---|
| IDCODE | 00001 | Selects device identification register for shift |
| SAMPLE/PRELOAD | 00010 | Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation |
| SAMPLE | 00011 | Selects boundary scan register for shifting and sampling without disturbing functional operation |
| EXTEST | 00100 | Selects boundary scan register while applying preloaded values to output pins and asserting functional reset |
| ACCESS_AUX_TAP_TCU | 10000 | Grants the TCU ownership of the TAP |
| ACCESS_AUX_TAP_ONCE | 10001 | Grants the PLATFORM ownership of the TAP |
| ACCESS_AUX_TAP_NPC | 10010 | Grants the Nexus port controller (NPC) ownership of the TAP |
| BYPASS | 11111 | Selects bypass register for data operations |
| Factory Debug Reserved[1] | 00101<br>00110<br>01010 | Intended for factory debug only |
| Reserved[2] | All Other Codes | Decoded to select bypass register |

1 Intended for factory debug, and not customer use

2 Freescale reserves the right to change the decoding of reserved instruction codes

### 24.8.4.1 BYPASS Instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

### 24.8.4.2 ACCESS_AUX_TAP_*x* Instructions

The ACCESS_AUX_TAP_*x* instructions allow the Nexus modules on the MCU to take control of the TAP. When this instruction is loaded, control of the TAP pins is transferred to the selected auxiliary TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive.

### 24.8.4.3 EXTEST — External Test Instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the

internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

#### 24.8.4.4 IDCODE Instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC is reset.

#### 24.8.4.5 SAMPLE Instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the update-DR state. Both the data capture and the shift operation are transparent to system operation.

#### 24.8.4.6 SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE part of the instruction samples the system data and control signals on the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising-edge of TCK in the capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD part of the instruction initializes the boundary scan register cells before selecting the EXTEST instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the update-DR state. The data is applied to the external output pins by the EXTEST instruction. System operation is not affected.

### 24.8.5 Boundary Scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

## 24.9 e200z0 OnCE Controller

The e200z0 core OnCE controller supports a complete set of Nexus 1 debug features, as well as providing access to the Nexus2+ configuration registers. A complete discussion of the e200z0 OnCE debug features is available in the *e200z0 Reference Manual*.

### 24.9.1 e200z0 OnCE Controller Block Diagram

Figure 24-6 is a block diagram of the e200z0 OnCE block.

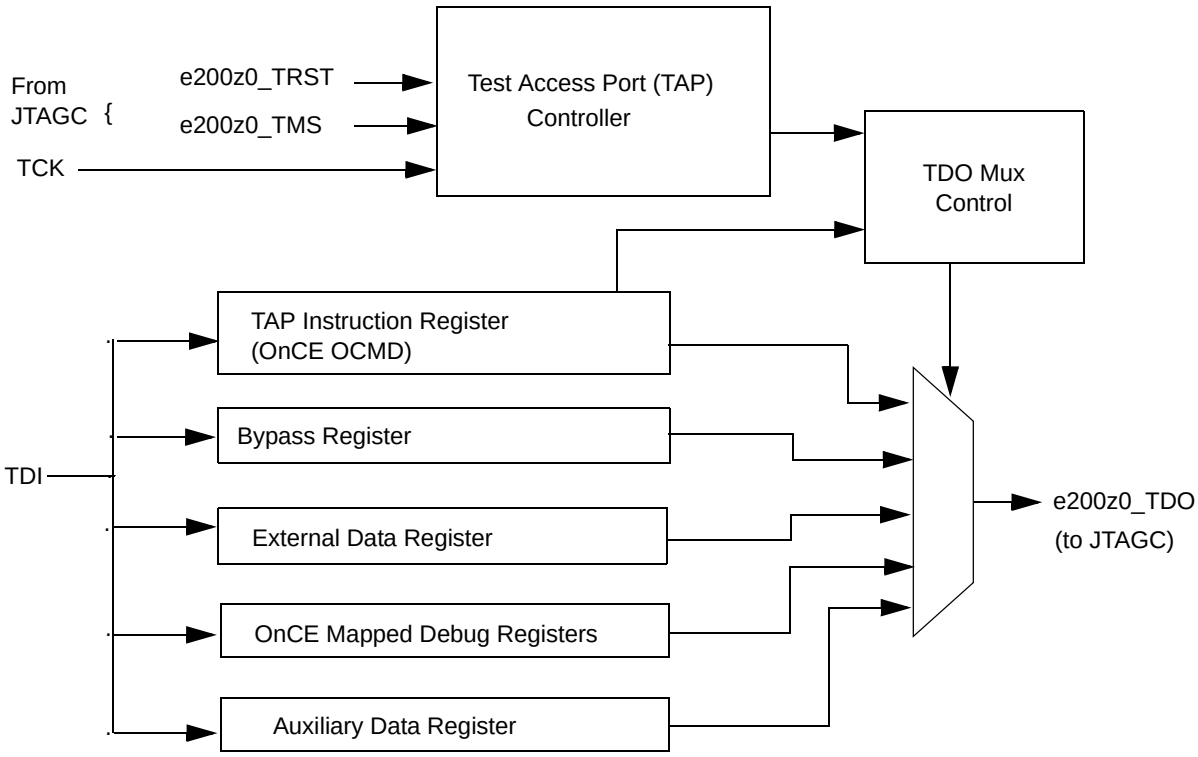



**Figure 24-6. e200z0 OnCE Block Diagram**

### 24.9.2 e200z0 OnCE Controller Functional Description

The functional description for the e200z0 OnCE controller is the same as for the JTAGC, with the differences described below.

#### 24.9.2.1 Enabling the TAP Controller

To access the e200z0 OnCE controller, the proper JTAGC instruction needs to be loaded in the JTAGC instruction register, as discussed in Section 24.5.2.2, “TAP Sharing Mode.

## 24.9.3 e200z0 OnCE Controller Register Description

Most e200z0 OnCE debug registers are fully documented in the *e200z0 Reference Manual*.

### 24.9.3.1 OnCE Command Register (OCMD)

The OnCE command register (OCMD) is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the e200z0 OnCE Decoder. The OCMD is shown in Table 24-5. The OCMD is updated when the TAP controller enters the update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the update-DR state must be transitioned through in order for an access to occur. In addition, the update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | R/W | GO | EX | RS[0:6] | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

**Table 24-5. OnCE Command Register (OCMD)**

**Table 24-6. e200z0 OnCE Register Addressing**

| RS[0:6] | Register Selected |
|---|---|
| 000 0000 – 000 0001 | Reserved |
| 000 0010 | JTAG ID (read-only) |
| 000 0011 – 000 1111 | Reserved |
| 001 0000 | CPU Scan Register (CPUSCR) |
| 001 0001 | No Register Selected (Bypass) |
| 001 0010 | OnCE Control Register (OCR) |
| 001 0011 – 001 1111 | Reserved |
| 010 0000 | Instruction Address Compare 1 (IAC1) |
| 010 0001 | Instruction Address Compare 2 (IAC2) |
| 010 0010 | Instruction Address Compare 3 (IAC3) |
| 010 0011 | Instruction Address Compare 4 (IAC4) |
| 010 0100 | Data Address Compare 1 (DAC1) |
| 010 0101 | Data Address Compare 2 (DAC2) |
| 010 0110 | Data Value Compare 1 (DVC1) |

**Table 24-6. e200z0 OnCE Register Addressing (continued)**

| RS[0:6] | Register Selected |
|---|---|
| 010 0111 | Data Value Compare 2 (DVC2) |
| 010 1000 – 010 1111 | Reserved |
| 011 0000 | Debug Status Register (DBSR) |
| 011 0001 | Debug Control Register 0 (DBCR0) |
| 011 0010 | Debug Control Register 1 (DBCR1) |
| 011 0011 | Debug Control Register 2 (DBCR2) |
| 011 0100 – 101 1111 | Reserved (do not access) |
| 110 1111 | Reserved (do not access) |
| 111 0000 – 111 1001 | General Purpose Register Selects [0:9] |
| 111 1010 – 111 1011 | Reserved |
| 111 1100 | Nexus2+ Access |
| 111 1101 | LSRL Select<br>(factory test use only) |
| 111 1110 | Enable_OnCE |
| 111 1111 | Bypass |

## 24.10 Initialization/Application Information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC module and enable access to registers, the following sequence is required:

1. Place the JTAGC in reset through TAP controller state machine transitions controlled by TMS
2. Load the appropriate instruction for the test or action to be performed.

This page is intentionally left blank.

# Chapter 25
# Inter-Integrated Circuit Bus Controller Module ($I^2C$)

## 25.1 Introduction

### 25.1.1 Overview

The Inter-Integrated Circuit ($I^2C$™ or IIC) bus is a two wire bidirectional serial bus that provides a simple and efficient method of data exchange between devices. It minimizes the number of external connections to devices and does not require an external address decoder.

This bus is suitable for applications requiring occasional communications over a short distance between a number of devices. It also provides flexibility, allowing additional devices to be connected to the bus for further expansion and system development.

The interface is designed to operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of module clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

### 25.1.2 Features

The $I^2C$ module has the following key features:

- Compatible with $I^2C$ Bus standard
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection

Features not supported:

- No support for general call address
- Not compliant to ten-bit addressing

### 25.1.3 Block diagram

The block diagram of the $I^2C$ module is shown in Figure 25-1.

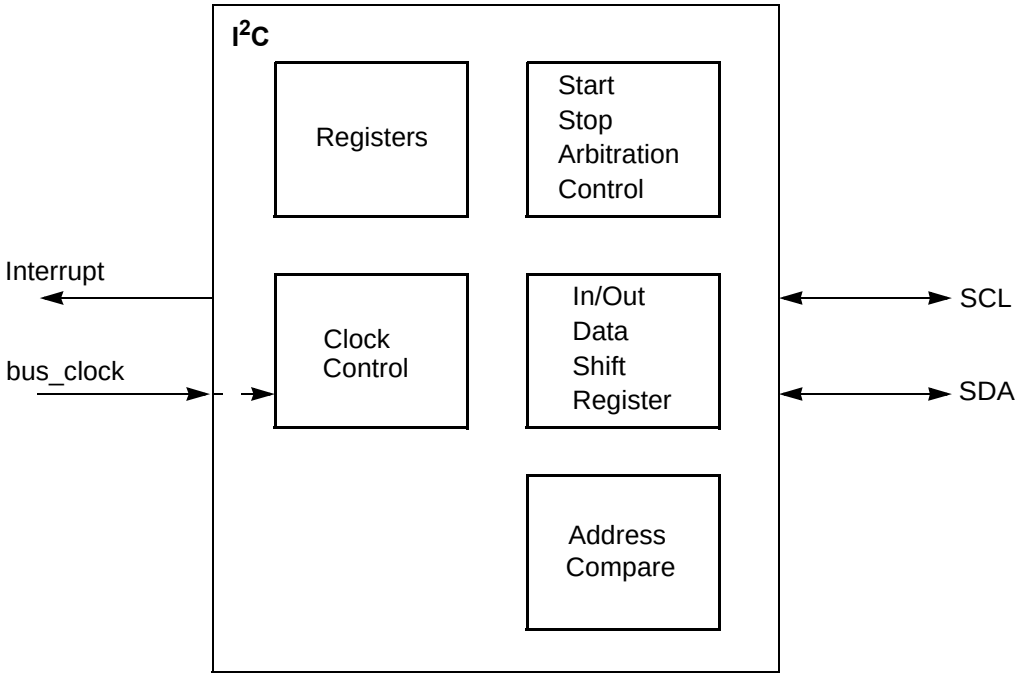



**Figure 25-1. I$^2$C block diagram**

## 25.2 Modes of operation

The I$^2$C module has the following modes of operation:

- Run mode: This is the basic mode of operation.
- Stop mode: This is the lowest power saving mode and allows the system to turn of all the clocks to the I$^2$C module. This state can only be entered when there are no active transfers on the bus.

## 25.3 External signal description

### 25.3.1 Overview

The Inter-Integrated Circuit (I$^2$C) module has 2 external pins.

### 25.3.2 Detailed signal descriptions

#### 25.3.2.1 SCL

This is the bidirectional Serial Clock Line (SCL) of the module, compatible with the I$^2$C-Bus specification.

#### 25.3.2.2 SDA

This is the bidirectional Serial Data line (SDA) of the module, compatible with the I$^2$C-Bus specification.

## 25.4 Memory map and register description

### 25.4.1 Overview

This section provides a detailed description of all memory-mapped registers in the $I^2C$ module.

### 25.4.2 Module memory map

The memory map for the $I^2C$ module is given below in Table 25-1. The total address for each register is the sum of the base address for the $I^2C$ module and the address offset for each register.

**Table 25-1. Module memory map**

| Address Reg | ister | Location |
|---|---|---|
| Base + 0x00 | $I^2C$ Bus Address Register (IBAD) | on page 907 |
| Base + 0x01 | $I^2C$ Bus Frequency Divider Register (IBFD) | on page 908 |
| Base + 0x02 | $I^2C$ Bus Control Register (IBCR) | on page 917 |
| Base + 0x03 | $I^2C$ Bus Status Register (IBSR) | on page 918 |
| Base + 0x04 | $I^2C$ Bus Data I/O Register (IBDR) | on page 920 |
| Base + 0x05 | $I^2C$ Bus Interrupt Configuration Register (IBIC) | on page 920 |
| Base + 0x06 | Unused | — |
| Base + 0x07 | Unused | — |
| Base + 0x08 – Base + 0x3FFF | Reserved | — |

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the IBDF register for the frequency divider is accessible by a 16-bit READ/WRITE to address Base + 0x000, but performing a 16-bit access to Base + 0x001 is illegal.

### 25.4.3 Register description

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

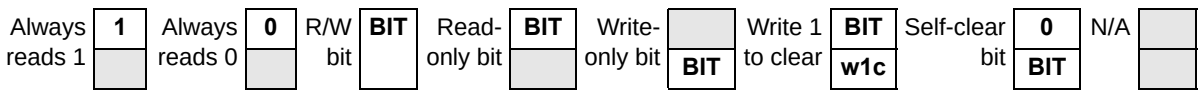



**Figure 25-2. Key to Register Fields**

#### 25.4.3.1 $I^2C$ Bus Address Register

This register contains the address the $I^2C$ Bus will respond to, when addressed as a slave; note that it is not the address sent on the bus during the address transfer.

Offset 0x00000 Access: Supervisor/Test

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | ADR | | | | | | | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 25-3. I$^2$C Bus Address Register (IBAD)**

**Table 25-2. IBAD Field Descriptions**

| Field Description | |
|---|---|
| ADR | Slave Address. Specific slave address to be used by the I$^2$C Bus module.<br>**Note:** The default mode of I$^2$C Bus is slave mode for an address match on the bus. |

### 25.4.3.2 I$^2$C Bus Frequency Divider Register

Offset 0x0001 Access: Supervisor/Test

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | IBC | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 25-4. I$^2$C Bus Frequency Divider Register (IBFD)**

**Table 25-3. IBFD Field Descriptions**

| Field Description | |
|---|---|
| IBC | I-Bus Clock Rate. This field is used to prescale the clock for bit rate selection. The bit clock generator is implemented as a prescale divider. The IBC bits are decoded to give the Tap and Prescale values as follows:<br>0–1 select the prescaled shift register (see Table 25-4)<br>2–4 select the prescaler divider (see Table 25-5)<br>5–7 select the shift register tap point (see Table 25-6) |

**Table 25-4. I-Bus Multiplier Factor**

| IBC[0:1] | MUL |
|---|---|
| 00 | 01 |
| 01 | 02 |
| 10 | 04 |
| 11 | RESERVED |

**Table 25-5. I-Bus Prescaler Divider Values**

| IBC[2:4] | scl2start (clocks) | scl2stop (clocks) | scl2tap (clocks) | tap2tap (clocks) |
|---|---|---|---|---|
| 000 | 2 | 7 | 4 | 1 |
| 001 | 2 | 7 | 4 | 2 |
| 010 | 2 | 9 | 6 | 4 |
| 011 | 6 | 9 | 6 | 8 |
| 100 | 14 | 17 | 14 | 16 |
| 101 | 30 | 33 | 30 | 32 |
| 110 | 62 | 65 | 62 | 64 |
| 111 | 126 | 129 | 126 | 128 |

**Table 25-6. I-Bus Tap and Prescale Values**

| IBC[5:7] | SCL Tap (clocks) | SDA Tap (clocks) |
|---|---|---|
| 000 | 5 | 1 |
| 001 | 6 | 1 |
| 010 | 7 | 2 |
| 011 | 8 | 2 |
| 100 | 9 | 3 |
| 101 | 10 | 3 |
| 110 | 12 | 4 |
| 111 | 15 | 4 |

The number of clocks from the falling edge of SCL to the first tap (Tap[1]) is defined by the values shown in the scl2tap column of Table 25-5. All subsequent tap points are separated by $2^{IBC[2:4]}$ as shown in the tap2tap column in Table 25-5. The SCL Tap is used to generate the SCL period and the SDA Tap is used to determine the delay from the falling edge of SCL to the change of state of SDA i.e. the SDA hold time.

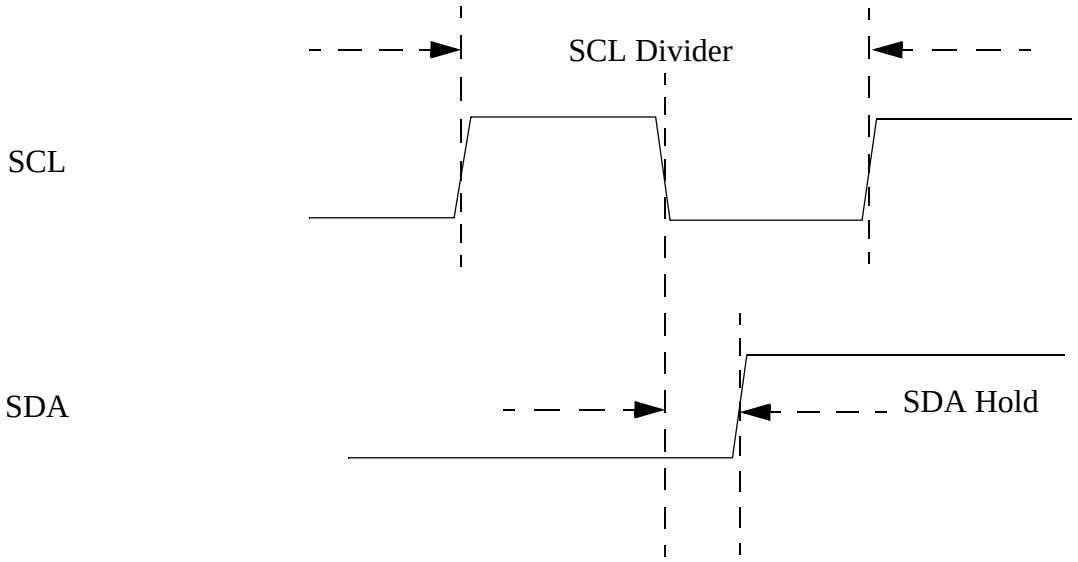



**Figure 25-5. SDA Hold Time**

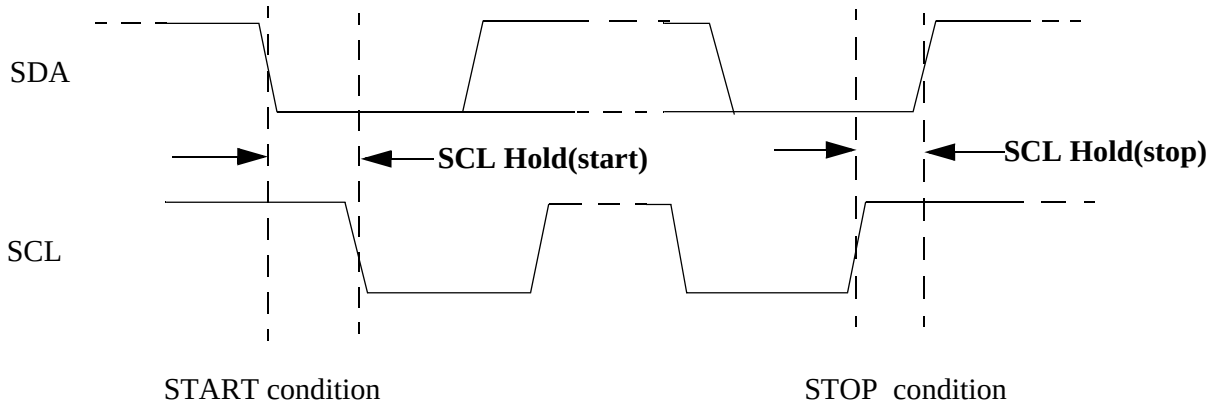



**Figure 25-6. SCL Divider and SDA Hold**

The equation used to generate the divider values from the IBFD bits is:

$$\text{SCL Divider} = \text{MUL} \times \{2 \times (\text{scl2tap} + [(\text{SCL_Tap} - 1) \times \text{tap2tap}] + 2)\} \quad \textit{Eqn. 25-1}$$

The SDA hold delay is equal to the CPU clock period multiplied by the SDA Hold value shown in Table 25-7. The equation used to generate the SDA Hold value from the IBFD bits is:

$$\text{SDA Hold} = \text{MUL} \times \{\text{scl2tap} + [(\text{SDA_Tap} - 1) \times \text{tap2tap}] + 3\} \quad \textit{Eqn. 25-2}$$

The equation for SCL Hold values to generate the start and stop conditions from the IBFD bits is:

$$\text{SCL Hold(start)} = \text{MUL} \times [\text{scl2start} + (\text{SCL_Tap} - 1) \times \text{tap2tap}] \quad \textit{Eqn. 25-3}$$

$$\text{SCL Hold(stop)} = \text{MUL} \times [\text{scl2stop} + (\text{SCL_Tap} - 1) \times \text{tap2tap}] \quad \textit{Eqn. 25-4}$$

## Table 25-7. I²C Divider and Hold Values

| | IBC (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|---|
| MUL = 1 | 00 | 20 | 7 | 6 | 11 |
| | 01 | 22 | 7 | 7 | 12 |
| | 02 | 24 | 8 | 8 | 13 |
| | 03 | 26 | 8 | 9 | 14 |
| | 04 | 28 | 9 | 10 | 15 |
| | 05 | 30 | 9 | 11 | 16 |
| | 06 | 34 | 10 | 13 | 18 |
| | 07 | 40 | 10 | 16 | 21 |
| | 08 | 28 | 7 | 10 | 15 |
| | 09 | 32 | 7 | 12 | 17 |
| | 0A | 36 | 9 | 14 | 19 |
| | 0B | 40 | 9 | 16 | 21 |
| | 0C | 44 | 11 | 18 | 23 |
| | 0D | 48 | 11 | 20 | 25 |
| | 0E | 56 | 13 | 24 | 29 |
| | 0F | 68 | 13 | 30 | 35 |
| | 10 | 48 | 9 | 18 | 25 |
| | 11 | 56 | 9 | 22 | 29 |
| | 12 | 64 | 13 | 26 | 33 |
| | 13 | 72 | 13 | 30 | 37 |
| | 14 | 80 | 17 | 34 | 41 |
| | 15 | 88 | 17 | 38 | 45 |
| | 16 | 104 | 21 | 46 | 53 |
| | 17 | 128 | 21 | 58 | 65 |
| | 18 | 80 | 9 | 38 | 41 |
| | 19 | 96 | 9 | 46 | 49 |
| | 1A | 112 | 17 | 54 | 57 |

## Table 25-7. I²C Divider and Hold Values (continued)

| | IBC (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|---|
| MUL = 1 | 1B | 128 | 17 | 62 | 65 |
| | 1C | 144 | 25 | 70 | 73 |
| | 1D | 160 | 25 | 78 | 81 |
| | 1E | 192 | 33 | 94 | 97 |
| | 1F | 240 | 33 | 118 | 121 |
| | 20 | 160 | 17 | 78 | 81 |
| | 21 | 192 | 17 | 94 | 97 |
| | 22 | 224 | 33 | 110 | 113 |
| | 23 | 256 | 33 | 126 | 129 |
| | 24 | 288 | 49 | 142 | 145 |
| | 25 | 320 | 49 | 158 | 161 |
| | 26 | 384 | 65 | 190 | 193 |
| | 27 | 480 | 65 | 238 | 241 |
| | 28 | 320 | 33 | 158 | 161 |
| | 29 | 384 | 33 | 190 | 193 |
| | 2A | 448 | 65 | 222 | 225 |
| MUL = 1 | 2B | 512 | 65 | 254 | 257 |
| | 2C | 576 | 97 | 286 | 289 |
| | 2D | 640 | 97 | 318 | 321 |
| | 2E | 768 | 129 | 382 | 385 |
| | 2F | 960 | 129 | 478 | 481 |
| | 30 | 640 | 65 | 318 | 321 |
| | 31 | 768 | 65 | 382 | 385 |
| | 32 | 896 | 129 | 446 | 449 |
| | 33 | 1024 | 129 | 510 | 513 |
| | 34 | 1152 | 193 | 574 | 577 |
| | 35 | 1280 | 193 | 638 | 641 |
| | 36 | 1536 | 257 | 766 | 769 |
| | 37 | 1920 | 257 | 958 | 961 |
| | 38 | 1280 | 129 | 638 | 641 |
| | 39 | 1536 | 129 | 766 | 769 |
| | 3A | 1792 | 257 | 894 | 897 |

## Table 25-7. I²C Divider and Hold Values (continued)

| | IBC (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|---|
| MUL = 1 | 3B | 2048 | 257 | 1022 | 1025 |
| | 3C | 2304 | 385 | 1150 | 1153 |
| | 3D | 2560 | 385 | 1278 | 1281 |
| | 3E | 3072 | 513 | 1534 | 1537 |
| | 3F | 3840 | 513 | 1918 | 1921 |
| MUL = 2 | 40 | 40 | 14 | 12 | 22 |
| | 41 | 44 | 14 | 14 | 24 |
| | 42 | 48 | 16 | 16 | 26 |
| | 43 | 52 | 16 | 18 | 28 |
| | 44 | 56 | 18 | 20 | 30 |
| | 45 | 60 | 18 | 22 | 32 |
| | 46 | 68 | 20 | 26 | 36 |
| | 47 | 80 | 20 | 32 | 42 |
| | 48 | 56 | 14 | 20 | 30 |
| | 49 | 64 | 14 | 24 | 34 |
| | 4A | 72 | 18 | 28 | 38 |
| MUL = 2 | 4B | 80 | 18 | 32 | 42 |
| | 4C | 88 | 22 | 36 | 46 |
| | 4D | 96 | 22 | 40 | 50 |
| | 4E | 112 | 26 | 48 | 58 |
| | 4F | 136 | 26 | 60 | 70 |
| | 50 | 96 | 18 | 36 | 50 |
| | 51 | 112 | 18 | 44 | 58 |
| | 52 | 128 | 26 | 52 | 66 |
| | 53 | 144 | 26 | 60 | 74 |
| | 54 | 160 | 34 | 68 | 82 |
| | 55 | 176 | 34 | 76 | 90 |
| | 56 | 208 | 42 | 92 | 106 |
| | 57 | 256 | 42 | 116 | 130 |
| | 58 | 160 | 18 | 76 | 82 |
| | 59 | 192 | 18 | 92 | 98 |
| | 5A | 224 | 34 | 108 | 114 |

Table 25-7. I²C Divider and Hold Values (continued)

| | IBC (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|---|
| MUL = 2 | 5B | 256 | 34 | 124 | 130 |
| | 5C | 288 | 50 | 140 | 146 |
| | 5D | 320 | 50 | 156 | 162 |
| | 5E | 384 | 66 | 188 | 194 |
| | 5F | 480 | 66 | 236 | 242 |
| | 60 | 320 | 28 | 156 | 162 |
| | 61 | 384 | 28 | 188 | 194 |
| | 62 | 448 | 32 | 220 | 226 |
| | 63 | 512 | 32 | 252 | 258 |
| | 64 | 576 | 36 | 284 | 290 |
| | 65 | 640 | 36 | 316 | 322 |
| | 66 | 768 | 40 | 380 | 386 |
| | 67 | 960 | 40 | 476 | 482 |
| | 68 | 640 | 28 | 316 | 322 |
| | 69 | 768 | 28 | 380 | 386 |
| | 6A | 896 | 36 | 444 | 450 |
| MUL = 2 | 6B | 1024 | 36 | 508 | 514 |
| | 6C | 1152 | 44 | 572 | 578 |
| | 6D | 1280 | 44 | 636 | 642 |
| | 6E | 1536 | 52 | 764 | 770 |
| | 6F | 1920 | 52 | 956 | 962 |
| | 70 | 1280 | 36 | 636 | 642 |
| | 71 | 1536 | 36 | 764 | 770 |
| | 72 | 1792 | 52 | 892 | 898 |
| | 73 | 2048 | 52 | 1020 | 1026 |
| | 74 | 2304 | 68 | 1148 | 1154 |
| | 75 | 2560 | 68 | 1276 | 1282 |
| | 76 | 3072 | 84 | 1532 | 1538 |
| | 77 | 3840 | 84 | 1916 | 1922 |
| | 78 | 2560 | 36 | 1276 | 1282 |
| | 79 | 3072 | 36 | 1532 | 1538 |
| | 7A | 3584 | 68 | 1788 | 1794 |

## Table 25-7. I²C Divider and Hold Values (continued)

| | IBC (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|---|
| MUL = 2 | 7B | 4096 | 68 | 2044 | 2050 |
| | 7C | 4608 | 100 | 2300 | 2306 |
| | 7D | 5120 | 100 | 2556 | 2562 |
| | 7E | 6144 | 132 | 3068 | 3074 |
| | 7F | 7680 | 132 | 3836 | 3842 |
| MUL = 4 | 80 | 80 | 28 | 24 | 44 |
| | 81 | 88 | 28 | 28 | 48 |
| | 82 | 96 | 32 | 32 | 52 |
| | 83 | 104 | 32 | 36 | 56 |
| | 84 | 112 | 36 | 40 | 60 |
| | 85 | 120 | 36 | 44 | 64 |
| | 86 | 136 | 40 | 52 | 72 |
| | 87 | 160 | 40 | 64 | 84 |
| | 88 | 112 | 28 | 40 | 60 |
| | 89 | 128 | 28 | 48 | 68 |
| | 8A | 144 | 36 | 56 | 76 |
| MUL = 4 | 8B | 160 | 36 | 64 | 84 |
| | 8C | 176 | 44 | 72 | 92 |
| | 8D | 192 | 44 | 80 | 100 |
| | 8E | 224 | 52 | 96 | 116 |
| | 8F | 272 | 52 | 120 | 140 |
| | 90 | 192 | 36 | 72 | 100 |
| | 91 | 224 | 36 | 88 | 116 |
| | 92 | 256 | 52 | 104 | 132 |
| | 93 | 288 | 52 | 120 | 148 |
| | 94 | 320 | 68 | 136 | 164 |
| | 95 | 352 | 68 | 152 | 180 |
| | 96 | 416 | 84 | 184 | 212 |
| | 97 | 512 | 84 | 232 | 260 |
| | 98 | 320 | 36 | 152 | 164 |
| | 99 | 384 | 36 | 184 | 196 |
| | 9A | 448 | 68 | 216 | 228 |

Table 25-7. I²C Divider and Hold Values (continued)

| | IBC (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|---|
| MUL = 4 | 9B | 512 | 68 | 248 | 260 |
| | 9C | 576 | 100 | 280 | 292 |
| | 9D | 640 | 100 | 312 | 324 |
| | 9E | 768 | 132 | 376 | 388 |
| | 9F | 960 | 132 | 472 | 484 |
| | A0 | 640 | 68 | 312 | 324 |
| | A1 | 768 | 68 | 376 | 388 |
| | A2 | 896 | 132 | 440 | 452 |
| | A3 | 1024 | 132 | 504 | 516 |
| | A4 | 1152 | 196 | 568 | 580 |
| | A5 | 1280 | 196 | 632 | 644 |
| | A6 | 1536 | 260 | 760 | 772 |
| | A7 | 1920 | 260 | 952 | 964 |
| | A8 | 1280 | 132 | 632 | 644 |
| | A9 | 1536 | 132 | 760 | 772 |
| | AA | 1792 | 260 | 888 | 900 |
| MUL = 4 | AB | 2048 | 260 | 1016 | 1028 |
| | AC | 2304 | 388 | 1144 | 1156 |
| | AD | 2560 | 388 | 1272 | 1284 |
| | AE | 3072 | 516 | 1528 | 1540 |
| | AF | 3840 | 516 | 1912 | 1924 |
| | 30 | 2560 | 260 | 1272 | 1284 |
| | B1 | 3072 | 260 | 1528 | 1540 |
| | B2 | 3584 | 516 | 1784 | 1796 |
| | B3 | 4096 | 516 | 2040 | 2052 |
| | B4 | 4608 | 772 | 2296 | 2308 |
| | B5 | 5120 | 772 | 2552 | 2564 |
| | B6 | 6144 | 1028 | 3064 | 3076 |
| | B7 | 7680 | 1028 | 3832 | 3844 |
| | B8 | 5120 | 516 | 2552 | 2564 |
| | B9 | 6144 | 516 | 3064 | 3076 |
| | BA | 7168 | 1028 | 3576 | 3588 |

**Table 25-7. I²C Divider and Hold Values (continued)**

| | IBC (hex) | SCL Divider (clocks) | SDA Hold (clocks) | SCL Hold (start) | SCL Hold (stop) |
|---|---|---|---|---|---|
| MUL = 4 | BB | 8192 | 1028 | 4088 | 4100 |
| | BC | 9216 | 1540 | 4600 | 4612 |
| | BD | 10240 | 1540 | 5112 | 5124 |
| | BE | 12288 | 2052 | 6136 | 6148 |
| | BF | 15360 | 2052 | 7672 | 7684 |

### 25.4.3.3 I²C Bus Control Register

Offset 0x0002 Access: Supervisor/Test

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | MDIS | IBIE | MS/$\overline{SL}$ | Tx/$\overline{Rx}$ | NOACK | 0 | DMAEN | D_RSVD |
| W | | | | | | RSTA | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 25-7. I²C Bus Control Register (IBCR)**

**Table 25-8. IBCR Field Descriptions**

| Field | Description |
|---|---|
| MDIS | Module disable. This bit controls the software reset of the entire I²C Bus module.<br>1 The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed<br>0 The I²C Bus module is enabled. This bit must be cleared before any other IBCR bits have any effect<br>**Note:** If the I²C Bus module is enabled in the middle of a byte transfer, the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the I²C Bus module losing arbitration, after which, bus operation would return to normal. |
| IBIE | I-Bus Interrupt Enable.<br>1 Interrupts from the I²C Bus module are enabled. An I²C Bus interrupt occurs provided the IBIF bit in the status register is also set.<br>0 Interrupts from the I²C Bus module are disabled. Note that this does not clear any currently pending interrupt condition |
| MS/$\overline{SL}$ | Master/Slave mode select. Upon reset, this bit is cleared. When this bit is changed from 0 to 1, a START signal is generated on the bus and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave. A STOP signal should be generated only if the IBIF flag is set. MS/$\overline{SL}$ is cleared without generating a STOP signal when the master loses arbitration.<br>1 Master Mode<br>0 Slave Mode |

**Table 25-8. IBCR Field Descriptions (continued)**

| Field De | scription |
|---|---|
| Tx/$\overline{Rx}$ | Transmit/Receive mode select. This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high.<br>1 Transmit<br>0 Receive |
| NOACK | Data Acknowledge disable. This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The $I^2C$ module will always acknowledge address matches, provided it is enabled, regardless of the value of NOACK. Note that values written to this bit are only used when the $I^2C$ Bus is a receiver, not a transmitter.<br>1 No acknowledge signal response is sent (i.e., acknowledge bit = 1)<br>0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data |
| RSTA | Repeat Start. Writing a 1 to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration.<br>1 Generate repeat start cycle<br>0 No effect |
| DMAEN | DMA Enable. When this bit is set, the DMA TX and RX lines will be asserted when the $I^2C$ module requires data to be read or written to the data register. No Transfer Done interrupts will be generated when this bit is set, however an interrupt will be generated if the loss of arbitration or addressed as slave conditions occur. The DMA mode is only valid when the $I^2C$ module is configured as a Master and the DMA transfer still requires CPU intervention at the start and the end of each frame of data. See the DMA Application Information section for more details.<br>1 Enable the DMA TX/RX request signals<br>0 Disable the DMA TX/RX request signals |
| D_RSVD | Reserved bit.<br>This bit is writable but should be kept as value 0. |

## 25.4.3.4 $I^2C$ Bus Status Register

Offset 0x0003 Access: Supervisor/Test[1]

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | TCF | IAAS | IBB | IBAL | 0 | SRW | IBIF | RXAK |
| W | | | | w1c | | | w1c | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 25-8. $I^2C$ Bus Status Register (IBSR)**

[1] With the exception of IBIF and IBAL, which are software clearable.

**Table 25-9. IBSR Field Descriptions**

| Field Des | cription |
|---|---|
| TCF | Transfer complete. While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. Note that this bit is only valid during or immediately following a transfer to the $I^2C$ module or from the $I^2C$ module.<br>1 Transfer complete<br>0 Transfer in progress |
| IAAS | Addressed as a slave. When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set its Tx/Rx mode accordingly. Writing to the I-Bus Control Register clears this bit.<br>1 Addressed as a slave<br>0 Not addressed |
| IBB | Bus busy. This bit indicates the status of the bus. When a START signal is detected, the IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state.<br>1 Bus is busy<br>0 Bus is Idle |
| IBAL | Arbitration Lost. The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances:<br>• SDA is sampled low when the master drives a high during an address or data transmit cycle.<br>• SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle.<br>• A start cycle is attempted when the bus is busy.<br>• A repeated start cycle is requested in slave mode.<br>• A stop condition is detected when the master did not request it.<br>This bit must be cleared by software, by writing a one to it. A write of zero has no effect. |
| SRW | Slave Read/Write. When IAAS is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By programming this bit, the CPU can select slave transmit/receive mode according to the command of the master.<br>1 Slave transmit, master reading from slave<br>0 Slave receive, master writing to slave |
| IBIF | I-Bus Interrupt Flag. The IBIF bit is set when one of the following conditions occurs:<br>• Arbitration lost (IBAL bit set)<br>• Byte transfer complete (TCF bit set and DMAEN bit not set)<br>• Addressed as slave (IAAS bit set)<br>• NoAck from Slave (MS & Tx bits set)<br>• $I^2C$ Bus going idle (IBB high-low transition and enabled by BIIE)<br>A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit. In DMA mode (DMAEN set), a byte transfer complete condition will not trigger the setting of IBIF. All other conditions still apply. |
| RXAK | Received Acknowledge. This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. This bit is valid only after transfer is complete.<br>1 No acknowledge received<br>0 Acknowledge received |

### 25.4.3.5 I²C Bus Data I/O Register

Offset 0x0004 Access: Supervisor/Test

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | DATA | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 25-9. I²C Bus Data I/O Register (IBDR)**

In master transmit mode, when data is written to IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred. Note that the Tx/$\overline{\text{Rx}}$ bit in the IBCR must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the I²C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the last byte received while the I²C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I²C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to the IBDR following assertion of MS/$\overline{\text{SL}}$ is used for the address transfer and should comprise the calling address (in position D7–D1) concatenated with the required R/$\overline{\text{W}}$ bit (in position D0).

### 25.4.3.6 I²C Bus Interrupt Configuration Register

Offset 0x0005 Access: Supervisor/Test

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | BIIE | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 25-10. I²C Bus Interrupt Configuration Register (IBIC)**

**Table 25-10. IBIC Field Descriptions**

| Field | Description |
|---|---|
| BIIE | Bus Idle Interrupt Enable bit. This config bit can be used to enable the generation of an interrupt once the I²C bus becomes idle. Once this bit is set, an IBB high-low transition will set the IBIF bit. This feature can be used to signal to the CPU the completion of a STOP on the I²C bus.<br>1 Bus Idle Interrupts enabled<br>0 Bus Idle Interrupts disabled |

## 25.5 Functional description

### 25.5.1 General

This section provides a complete functional description of the Inter-Integrated Circuit ($I^2C$).

### 25.5.2 I-Bus Protocol

The $I^2C$ Bus system uses a Serial Data line (SDA) and a Serial Clock Line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. A logical AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer and STOP signal. They are described briefly in the following sections and illustrated in Figure 25-11.

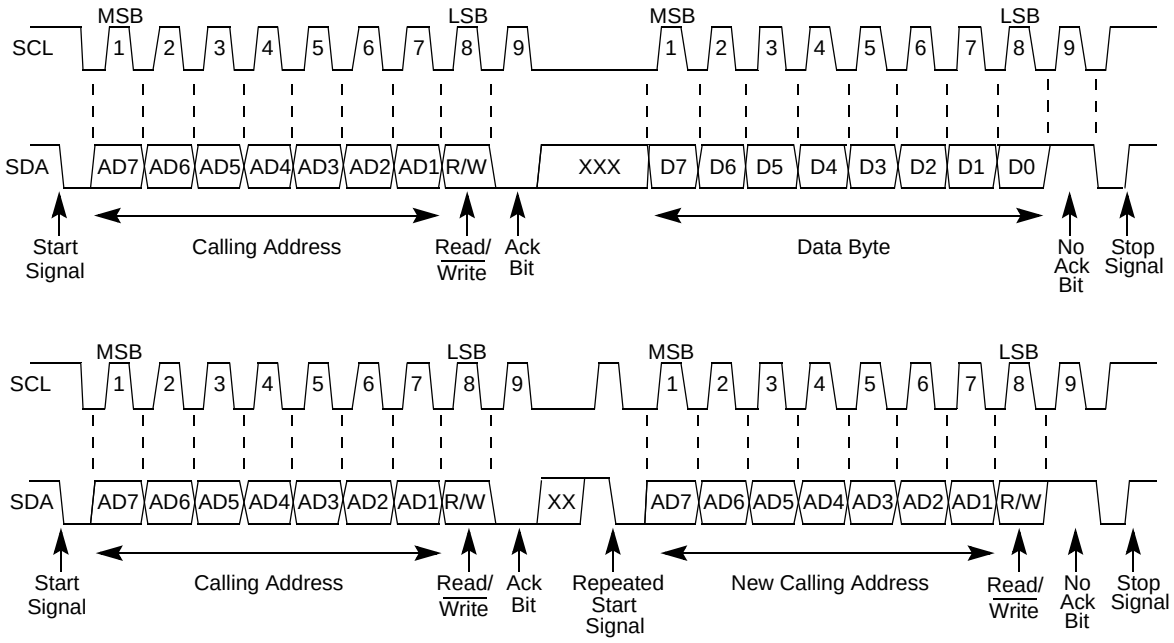



**Figure 25-11. $I^2C$ Bus Transmission Signals**

#### 25.5.2.1 START Signal

When the bus is free, i.e. no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in Figure 25-11, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

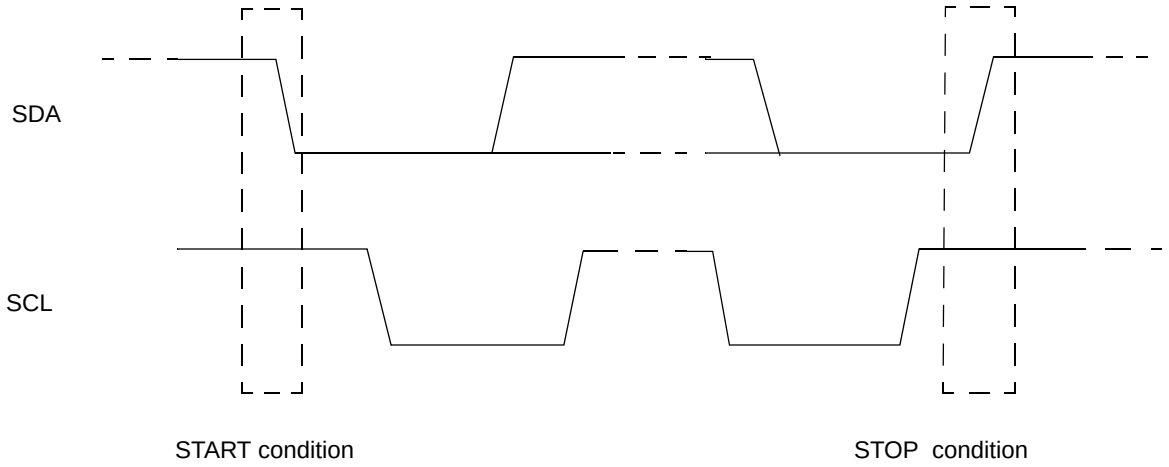



**Figure 25-12. Start and Stop conditions**

### 25.5.2.2 Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer - the slave transmits data to the master

0 = Write transfer - the master transmits data to the slave

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 25-11).

No two slaves in the system may have the same address. If the $I^2C$ Bus is master, it must not transmit an address that is equal to its own slave address. The $I^2C$ Bus cannot be master and slave at the same time. However, if arbitration is lost during an address cycle the $I^2C$ Bus will revert to slave mode and operate correctly, even if it is being addressed by another master.

### 25.5.2.3 Data Transfer

Once successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master.

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device.

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 25-11. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte must be followed by an acknowledge bit, which is signalled from the receiving device by pulling the SDA low at the ninth clock. Therefore, one complete data byte transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop signal to abort the data transfer or a start signal (repeated start) to commence a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte transmission, it means 'end of data' to the slave, so the slave releases the SDA line for the master to generate a STOP or START signal.

#### 25.5.2.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical “1” (see Figure 25-11).

The master can generate a STOP even if the slave has generated an acknowledge, at which point the slave must release the bus.

#### 25.5.2.5 Repeated START Signal

As shown in Figure 25-11, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

#### 25.5.2.6 Arbitration Procedure

The Inter-IC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic “1” while another master transmits logic “0”. The losing masters immediately switch over to slave receive mode and stop driving the SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

#### 25.5.2.7 Clock Synchronization

Since wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and once a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 25-13). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.

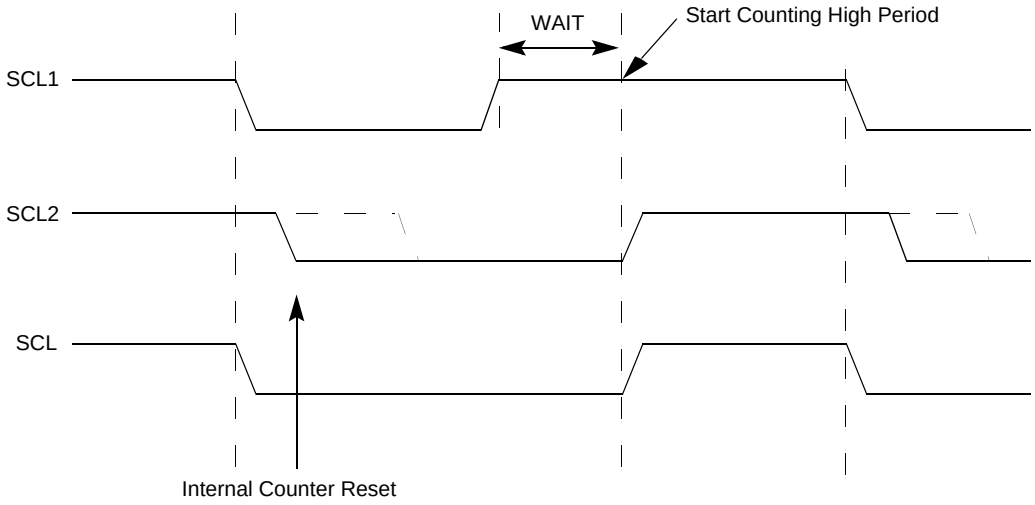



**Figure 25-13. I$^2$C Bus Clock Synchronization**

### 25.5.2.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

### 25.5.2.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

## 25.5.3 Interrupts

### 25.5.3.1 General

The I$^2$C module uses only one interrupt vector.

**Table 25-11. Interrupt Summary**

| Interrupt | Offset | Vector | Priority | Source | Description |
|---|---|---|---|---|---|
| I$^2$C Interrupt | — | — | — | IBAL, TCF, IAAS, IBB bits in IBSR register | When any of IBAL, TCF or IAAS bits is set an interrupt may be caused based on Arbitration lost, Transfer Complete or Address Detect conditions. If enabled by BIIE, the deassertion of IBB can also cause an interrupt, indicating that the bus is idle. |

#### 25.5.3.2 Interrupt Description

There are five types of internal interrupts in the $I^2C$. The interrupt service routine can determine the interrupt type by reading the Status Register.

$I^2C$ Interrupt can be generated on

- Arbitration Lost condition (IBAL bit set)
- Byte Transfer condition (TCF bit set)
- Address Detect condition (IAAS bit set)
- No Acknowledge from slave received when expected
- Bus Going Idle (IBB bit not set)

The $I^2C$ interrupt is enabled by the IBIE bit in the $I^2C$ Control Register. It must be cleared by writing '1' to the IBIF bit in the interrupt service routine. The Bus Going Idle interrupt needs to be additionally enabled by the BIIE bit in the IBIC register.

## 25.6 Initialization/Application Information

### 25.6.1 $I^2C$ Programming Examples

#### 25.6.1.1 Initialization Sequence

Reset will put the $I^2C$ Bus Control Register to its default state. Before the interface can be used to transfer serial data, an initialization procedure must be carried out, as follows:

1. Update the Frequency Divider Register (IBFD) and select the required division ratio to obtain SCL frequency from system clock.
2. Update the $I^2C$ Bus Address Register (IBAD) to define its slave address.
3. Clear the IBDIS bit of the $I^2C$ Bus Control Register (IBCR) to enable the $I^2C$ interface system.
4. Modify the bits of the $I^2C$ Bus Control Register (IBCR) to select Master/Slave mode, Transmit/Receive mode and interrupt enable or not. Optionally also modify the bits of the $I^2C$ Bus Interrupt Config Register (IBIC) to further refine the interrupt behavior.

#### 25.6.1.2 Generation of START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the $I^2C$ Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system

clock and the SCL period, it may be necessary to wait until the I$^2$C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (bit 5, IBSR ==1)// wait in loop for IBB flag to clear
bit4 and bit 5, IBCR = 1// set transmit and master mode, i.e. generate start condition
IBDR = calling_address// send the calling address to the data register
while (bit 5, IBSR ==0)// wait in loop for IBB flag to be set
```

### 25.6.1.3 Post-Transfer Software Response

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I$^2$C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing 1 (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress by reading the IBDR data register in receive mode or writing the IBDR in transmit mode. The TCF bit should not be used as a data transfer complete flag as the flag timing is dependent on a number of factors including the I$^2$C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag.

Software may service the I$^2$C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Note that polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

Note that when a “Transfer Complete” interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit in IBDR, then the Tx/Rx bit should be toggled at this stage. If Master does not receive an ACK from Slave, then transmission must be re-initiated or terminated.

During slave mode address cycles (IAAS=1) the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IAAS=0, the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

The following is an example software sequence for 'master transmitter' in the interrupt routine.

```
clear bit 1, IBSR// Clear the IBIF flag
if (bit 5, IBCR ==0)
slave_mode()// run slave mode routine
if (bit 4, IBCR ==0))
receive_mode()// run receive_mode routine
if (bit 0, IBSR == 1)// if NO ACK
     end();// end transmission
else
IBDR = data_to_transmit// transmit next byte of data
```

### 25.6.1.4 Generation of STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a stop condition is generated by a master transmitter.

```
if (tx_count == 0) or// check to see if all data bytes have been transmitted
    (bit 0, IBSR == 1) {// or if no ACK generated
         clear bit 5, IBCR// generate stop condition
        }
else {
IBDR = data_to_transmit// write byte of data to DATA register
         tx_count --// decrement counter
      }  // return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the transmit acknowledge bit (TXAK) before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following is an example showing how a STOP signal is generated by a master receiver.

```
rx_count --// decrease the rx counter
if (rx_count ==1)// 2nd last byte to be read ?
   bit 3, IBCR = 1// disable ACK
if (rx_count == 0)// last byte to be read ?
  bit 1, IBCR = 0// generate stop signal
else
data_received = IBDR// read RX data and store
```

### 25.6.1.5 Generation of Repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
bit 2, IBCR = 1// generate another start ( restart)
IBDR == calling_address// transmit the calling address
```

### 25.6.1.6 Slave Mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an 'end of data' signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

### 25.6.1.7 Arbitration Lost

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave receive mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware will inhibit the transmission, switch the MS/SL bit from 1 to 0 without generating a STOP condition, generate an interrupt to CPU and set the IBAL to indicate that the attempt to engage the bus is failed. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

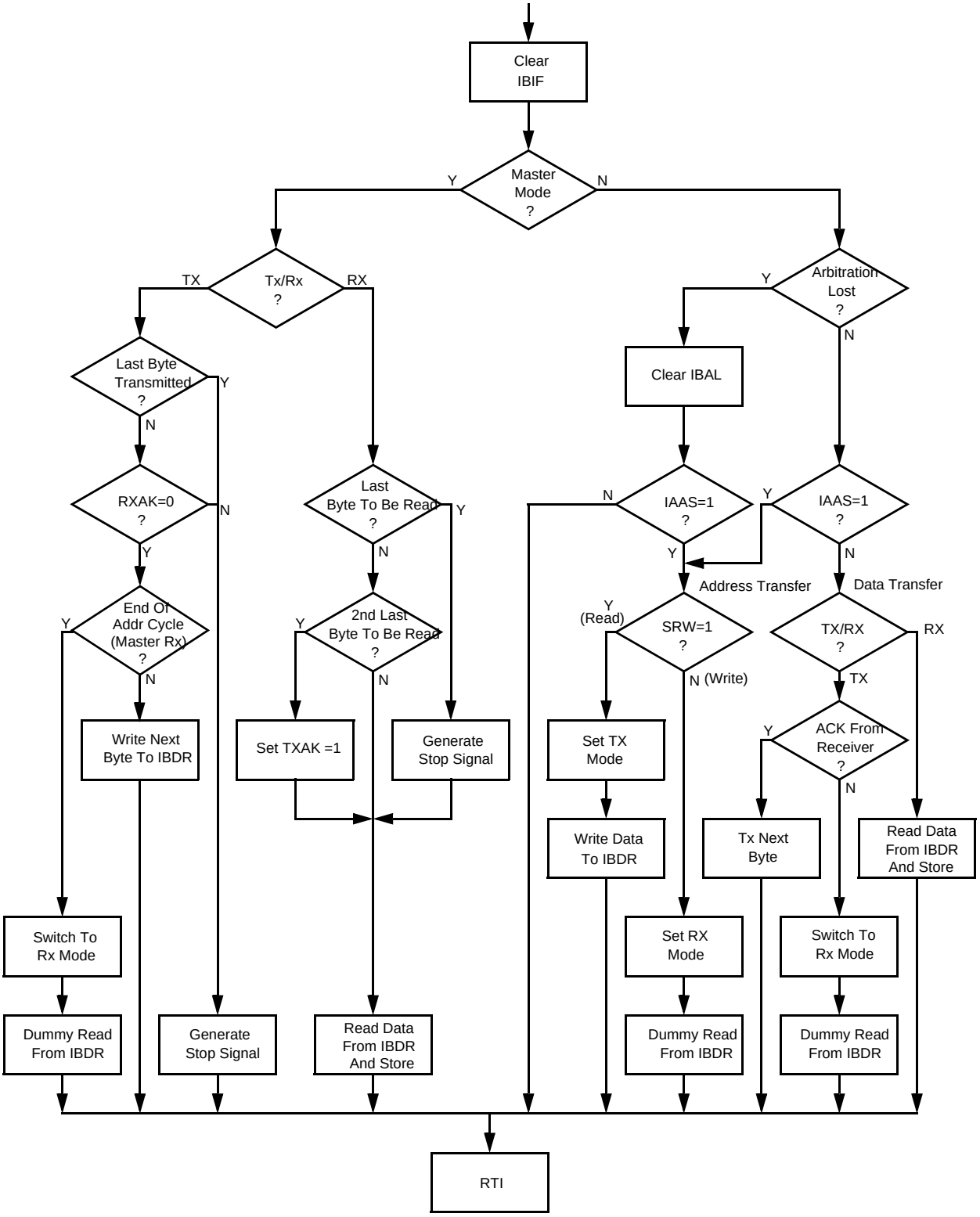



**Figure 25-14. Flow-Chart of Typical I²C Interrupt Routine**

### 25.6.2 DMA application information

The DMA interface on the I2C is not completely autonomous and requires intervention from the CPU to start and to terminate the frame transfer. DMA mode is only valid for Master transmit and Master receive modes. Software must ensure that the IBCR[DMAEN] bit is not set when the $I^2C$ module is configured in master mode.

The DMA controller must only transfer one byte of data per Tx/Rx request. This is because there is no FIFO on the $I^2C$ block.

The CPU should also keep the $I^2C$ interrupt enabled during a DMA transfer to detect the arbitration lost condition and take action to recover from this situation. The IBCR[DMAEN] bit works as a disable for the transfer complete interrupt. This means that during normal transfers (no errors) there will always be either an interrupt or a request to the DMA controller, depending on the setting of the DMAEN bit. All error conditions will trigger an interrupt and require CPU intervention. The address match condition will not occur in DMA mode as the $I^2C$ should never be configured for slave operation.

# Chapter 26
# Interrupt Controller (INTC)

## 26.1 Introduction

The INTC provides priority-based preemptive scheduling of interrupt service requests (ISRs). This scheduling scheme is suitable for statically scheduled hard real-time systems. The INTC supports 171 interrupt requests. It is targeted to work with Power Architecture technology and automotive applications where the ISRs nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, the time from the assertion of the peripheral's interrupt request to when the processor is performing useful work to service the interrupt request needs to be minimized. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 16 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Because each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the priority ceiling protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource can not preempt each other.

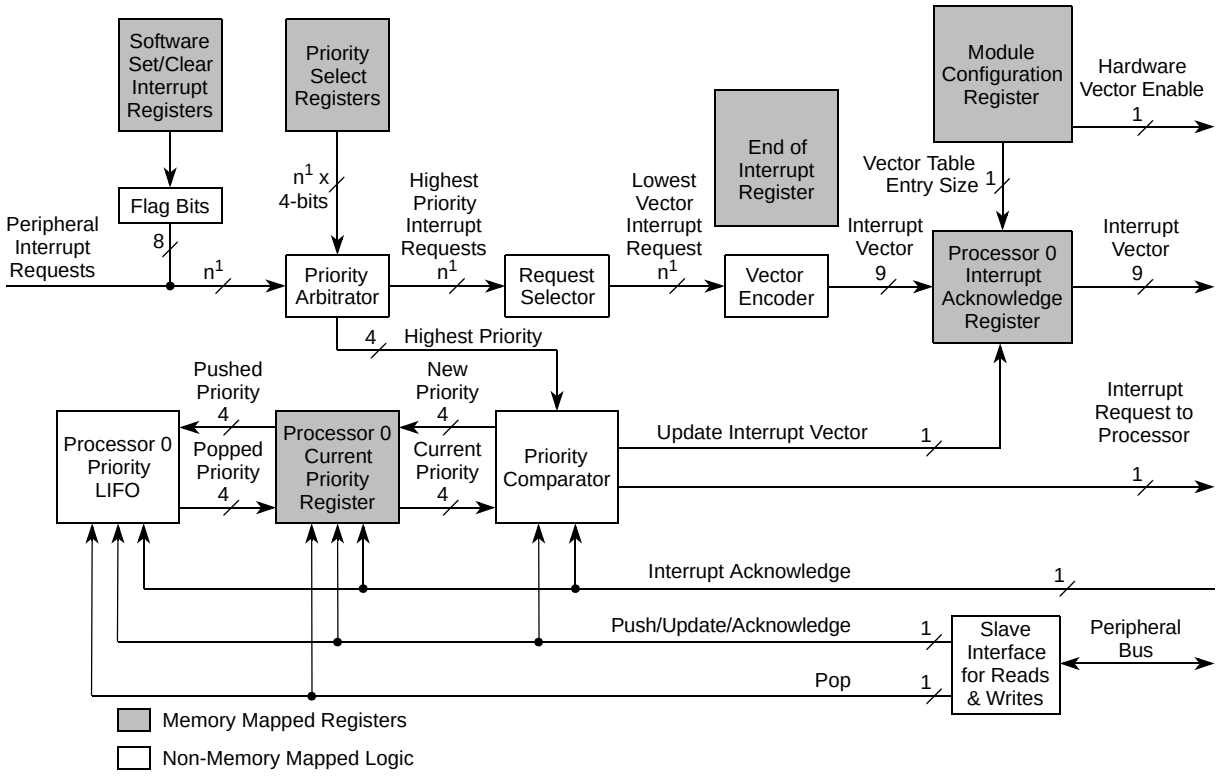
Multiple processors can assert interrupt requests to each other through software configurable interrupt requests. These software configurable interrupt requests can also be used to separate the work involved in servicing an interrupt request into a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software configurable interrupt request to finish the servicing in a lower priority ISR. Therefore these software configurable interrupt requests can be used instead of the peripheral ISR scheduling a task through the RTOS.

## 26.2 Features

- Supports 163 peripheral and 8 software-configurable interrupt request sources
- Unique 9-bit vector per interrupt source
- Each interrupt source can be programmed to one of 16 priorities
- Preemption
  - Preemptive prioritized interrupt requests to processor
  - ISR at a higher priority preempts ISRs or tasks at lower priorities
  - Automatic pushing or popping of preempted priority to or from a LIFO
  - Ability to modify the ISR or task priority; modifying the priority can be used to implement the priority ceiling protocol for accessing shared resources.
- Low latency - three clocks from receipt of interrupt request from peripheral to interrupt request to processor

## 26.3 Block diagram

Figure 26-1 is a block diagram of the INTC.




1 The total number of interrupt sources is 171, which includes 8 software sources.

**Figure 26-1. INTC block diagram**

## 26.4 Modes of operation

### 26.4.1 Normal mode

In normal mode, the INTC has two handshaking modes with the processor: software vector mode and hardware vector mode.

#### 26.4.1.1 Software vector mode

In software vector mode, the interrupt exception handler software must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for a given processor when its associated HVEN bit in INTC_MCR is negated. The hardware vector enable signal to processor 0 or processor 1 is driven as negated when its associated HVEN bit is negated. The vector is read from INTC_IACKR. Reading the INTC_IACKR negates the interrupt request to the associated processor. Even if a higher priority interrupt request arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in INTC_CPR onto the associated LIFO and updates PRI in the associated INTC_CPR with the new priority.

Furthermore, the interrupt vector to the processor is driven as all 0s. The interrupt acknowledge signal from the associated processor is ignored.

#### 26.4.1.2 Hardware vector mode

In hardware vector mode, the hardware signals the interrupt vector from the INTC in conjunction with a processor that can use that vector. This hardware causes the first instruction to be executed in handling the interrupt request to the processor to be specific to that vector. Therefore, the interrupt exception handler is specific to a peripheral or software configurable interrupt request rather than being common to all of them. The INTC uses hardware vector mode for a given processor when the associated HVEN bit in the INTC_MCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted. When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software configurable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKR field in the INTC_IACKR, depending on which processor was assigned to handle a given interrupt source.

The processor negates the interrupt request to the processor driven by the INTC by asserting the interrupt acknowledge signal for one clock. Even if a higher priority interrupt request arrived while waiting for the interrupt acknowledge, the interrupt request to the processor will negate for at least one clock.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the associated INTC_CPR does not occur when the associated interrupt acknowledge signal asserts and INTC_SSCIR0_3–INTC_SSCIR4_7 is written at a time such that the PRI value in the associated INTC_CPR register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR is updated with the new priority, and the associated LIFO is neither pushed or popped.

#### 26.4.1.3 Debug mode

The INTC operation in debug mode is identical to its operation in normal mode.

#### 26.4.1.4 Stop Mode

The INTC supports stop mode. The INTC can have its clock input disabled at any time by the clock driver on the device. While its clocks are disabled, the INTC registers are not accessible.

The INTC requires clocking in order for a peripheral interrupt request to generate an interrupt request to the processor. Since the INTC is not clocked in stop mode, peripheral interrupt requests can not be used as a wakeup source, unless the clock, reset, and power module (CRP) supports that interrupt request as a wakeup source.

## 26.5 Memory map and register description

### 26.5.1 Module memory map

Table 26-1 shows the INTC memory map.

**Table 26-1. INTC memory map**

| Offset from INTC_BASE_ADDR[1] | Register A | ccess | Reset Value | Location |
|---|---|---|---|---|
| 0x0000 | INTC_MCR—INTC module configuration register | R/W | 0x0000_0000 | on page 935 |
| 0x0004 | Reserved | — | — | — |
| 0x0008 | INTC_CPR—INTC current priority register | R/W | 0x0000_000F | on page 935 |
| 0x00C | Reserved | | | |
| 0x0010 | INTC_IACKR—INTC interrupt acknowledge register | R[2]/W | 0x0000_0000 | on page 937 |
| 0x0014 | Reserved | | | |
| 0x0018 | INTC_EOIR—INTC end of interrupt register | W | 0x0000_0000 | on page 937 |
| 0x001C | Reserved | | | |
| 0x0020–0x0027 | INTC_SSCIR[0:7]—INTC software set/clear interrupt register [0:7] | R/W | 0x0000_0000 | on page 938 |
| 0x0028–0x003C | Reserved | — | — | — |
| 0x0040–0x012C | INTC_PSRn -INTC priority select register [0:238][3] | R/W | 0x0000_0000 | on page 939 |

1 INTC_BASE_ADDR = 0xFFF4_8000

2 When the HVEN bit in the INTC module configuration register (INTC_MCR) is asserted, a read of the INTC_IACKR has no side effects.

3 The PRI fields are “reserved” for peripheral interrupt requests whose vectors are labeled as Reserved in Figure 26-3

### 26.5.2 Register description

With exception of the INTC_SSCI*n* and INTC_PSR*n*, all registers are 32 bits in width. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of eight bits, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSCI*n* and INTC_PSR*n* are 8 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

In software vector mode, the side effects of a read of INTC_IACKR arethe same regardless of the size of the read. In either software or hardware vector mode, the size of a write to either INTC_SSCIR0_3–INTC_SSCIR4_7 or INTC_EOIR does not affect the operation of the write.

### 26.5.2.1 INTC Module Configuration Register (INTC_MCR)

The module configuration register is used to configure options of the INTC.

Offset: 0x0000 Access: User read/write

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

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

**Figure 26-2. INTC Module Configuration Register (INTC_MCR)**

**Table 26-2. INTC_MCR Field Descriptions**

| Field De | scription |
|---|---|
| VTES | Vector table entry size. Controls the number of ‘0’s to the right of INTVEC in Section 26.5.2.3, “INTC Interrupt Acknowledge Register (INTC_IACKR)”). If the contents of INTC_IACKR are used as an address of an entry in a vectortable as in software vector mode, then the number of rightmost ‘0’s will determine the size of each vector table entry. VTES impacts software vector mode operation but also affects INTC_IACKR[INTVEC] position in both hardware vector mode and software vector mode.<br>0 4 bytes.<br>1 8 bytes. |
| HVEN | Hardware vector enable. Controls whether the INTC is in hardware vector mode or software vector mode. Refer to Section 26.4, “Modes of operation,” for the details of the handshaking with the processor in each mode.<br>0 Software vector mode.<br>1 Hardware vector mode. |

### 26.5.2.2 INTC Current Priority Register for Processor (INTC_CPR)

Offset: 0x0008 Access: User read/write

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

**Figure 26-3. INTC Current Priority Register (INTC_CPR)**

**Table 26-3. INTC_CPR Field Descriptions**

| Field De | scription |
|---|---|
| PRI | Priority. PRI is the priority of the currently executing ISR according to the field values defined in Table 26-4. |

The INTC_CPR masks any peripheral or software settable interrupt request set at the same or lower priority as the current value of the INTC_CPR[PRI] field from generating an interrupt request to the processor. When the INTC interrupt acknowledge register (INTC_IACKR) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR) is written, the LIFO is popped into the INTC_CPR's PRI field.

The masking priority can be raised or lowered by writing to the PRI field, supporting the PCP. Refer to Section 26.7.5, "Priority Ceiling Protocol."

**NOTE**

A store to modify the PRI field that closely precedes or follows an access to a shared resource can result in a non-coherent access to the resource. Refer to Section 26.7.5.2, "Ensuring coherency" for example code to ensure coherency.

**Table 26-4. PRI values**

| PRI Mean | ing |
|---|---|
| 1111 | Priority 15—highest priority |
| 1110 | Priority 14 |
| 1101 | Priority 13 |
| 1100 | Priority 12 |
| 1011 | Priority 11 |
| 1010 | Priority 10 |
| 1001 | Priority 9 |
| 1000 | Priority 8 |
| 0111 | Priority 7 |
| 0110 | Priority 6 |
| 0101 | Priority 5 |
| 0100 | Priority 4 |
| 0011 | Priority 3 |
| 0010 | Priority 2 |
| 0001 | Priority 1 |
| 0000 | Priority 0—lowest priority |

### 26.5.2.3 INTC Interrupt Acknowledge Register (INTC_IACKR)

Offset: 0x0010 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VTBA (most significant 16 bits) | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VTBA (least significant five bits) | | | | INTVEC[1] | | | | | | | | | | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1 When the VTES bit in INTC_MCR is asserted, INTVEC is shifted to the left one bit. Bit 29 is read as a 0. VTBA is narrowed to 20 bits in width.

**Figure 26-4. INTC Interrupt Acknowledge Register (INTC_IACKR)**

**Table 26-5. INTC_IACKR Field Descriptions**

| Field De | scription |
|---|---|
| VTBA | Vector Table Base Address. Can be the base address of a vector table of addresses of ISRs. The VTBA only uses the leftmost 20 bits when the VTES bit in INTC_MCR is asserted. |
| INTVEC | Interrupt Vector.It is the vector of the peripheral or software configurable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.<br>**Note:** If INTC_MCR[VTES] = 1, then the INTVEC field is shifted left one position to bits 20–28. VTBA is then shortened by one bit to bits 0–19. |

The interrupt acknowledge register provides a value that can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

In software vector mode, the INTC_IACKR has side effects from reads. Therefore, it must not be speculatively read while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKR does not have side effects in hardware vector mode.

### 26.5.2.4 INTC End-of-Interrupt Register (INTC_EOIR)

Offset 0x0018 Access: write only

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

**Figure 26-5. INTC End-of-Interrupt Register (INTC_EOIR)**

Writing to the end-of-interrupt register signals the end of the servicing of the interrupt request. When the INTC_EOIR is written, the priority last pushed on the LIFO is popped into INTC_CPR. An exception to this behavior is described in Section 26.4.1.2, “Hardware vector mode.” The values and size of data written to the INTC_EOIR are ignored. The values and sizes written to this register neither update the INTC_EOIR contents or affect whether the LIFO pops. For possible future compatibility, write four bytes of all 0s to the INTC_EOIR.

Reading the INTC_EOIR has no effect on the LIFO.

### 26.5.2.5 INTC Software Set/Clear Interrupt Registers (INTC_SSCIR0_3–INTC_SSCIR4_7)

Offset: 0x0020 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLR0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLR1 |
| W | | | | | | | SET0 | | | | | | | | SET1 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLR2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLR3 |
| W | | | | | | | SET2 | | | | | | | | SET3 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 26-6. INTC Software Set/Clear Interrupt Register 0–3 (INTC_SSCIR[0:3])**

Offset: 0x0024 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLR4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLR5 |
| W | | | | | | | SET4 | | | | | | | | SET5 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLR6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLR7 |
| W | | | | | | | SET6 | | | | | | | | SET7 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 26-7. INTC Software Set/Clear Interrupt Register 4–7 (INTC_SSCIR[4:7])**

**Table 26-6. INTC_SSCIR[0:7] Field Descriptions**

| Field De | scription |
|---|---|
| SET | Set Flag Bits. Writing a 1 sets the corresponding CLR*x* bit. Writing a 0 has no effect. Each SET*x* always will be read as a 0. |
| CLR | Clear Flag Bits. CLR*x* is the flag bit. Writing a 1 to CLR*x* clears it provided that a 1 is not written simultaneously to its corresponding SET*x* bit. Writing a 0 to CLR*x* has no effect.<br>0 Interrupt request not pending within INTC.<br>1 Interrupt request pending within INTC. |

The software set/clear interrupt registers support the setting or clearing of software configurable interrupt request. These registers contain eight independent sets of bits to set and clear a corresponding flag bit by software. Excepting being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC like a peripheral interrupt request. Writing a 1 to SET*x* will leave SET*x* unchanged at 0 but sets CLR*x*. Writing a 0 to SET*x* has no effect. CLR*x* is the flag bit. Writing a 1 to CLR*x* clears it. Writing a 0 to CLR*x* has no effect. If a 1 is written simultaneously to a pair of SET*x* and CLR*x* bits, CLR*x* will be asserted, regardless of whether CLR*x* was asserted before the write.

### 26.5.2.6 INTC Priority Select Registers (INTC_PSR0_3–INTC_PSR206_238)

Offset: 0x0040 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PRI0 | | | | 0 | 0 | 0 | 0 | PRI1 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PRI2 | | | | 0 | 0 | 0 | 0 | PRI3 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 26-8. INTC Priority Select Register 0–3 (INTC_PSR[0:3])**

Offset: 0x00D0 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PRI236 | | | | 0 | 0 | 0 | 0 | PRI237 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PRI238 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 26-9. INTC Priority Select Register 236-238 (INTC_PSR[236:238])**

**Table 26-7. INTC_PSR0_3–INTC_PSR236_238 field descriptions**

| Field De | scription |
|---|---|
| PRI*n* | Priority Select. PRI*x* selects the priority for interrupt requests. Refer to Section 26.6, “Functional description.” |

**Table 26-8. INTC Priority Select Register Address Offsets**

| INTC_PSR*x*_*x* Offs | et Address |
|---|---|
| INTC_PSR0_3 | 0x0040 |
| INTC_PSR4_7 | 0x0044 |
| INTC_PSR8_11 | 0x0048 |
| INTC_PSR12_15 | 0x004C |
| INTC_PSR16_19 | 0x0050 |
| INTC_PSR20_23 | 0x0054 |
| INTC_PSR24_27 | 0x0058 |
| INTC_PSR28_31 | 0x005C |
| INTC_PSR32_35 | 0x0060 |
| INTC_PSR36_39 | 0x0064 |
| INTC_PSR40_43 | 0x0068 |
| INTC_PSR44_47 | 0x006C |
| INTC_PSR48_51 | 0x0070 |
| INTC_PSR52_55 | 0x0074 |
| INTC_PSR56_59 | 0x0078 |
| INTC_PSR60_63 | 0x007C |
| INTC_PSR64_67 | 0x0080 |

| INTC_PSR*x*_*x* Offset | Address |
|---|---|
| INTC_PSR120_123 | 0x00B8 |
| INTC_PSR124_127 | 0x00BC |
| INTC_PSR128_131 | 0x00C0 |
| INTC_PSR132_135 | 0x00C4 |
| INTC_PSR136_139 | 0x00C8 |
| INTC_PSR140_143 | 0x00CC |
| INTC_PSR144_147 | 0x00D0 |
| INTC_PSR148_151 | 0x00D4 |
| INTC_PSR152_155 | 0x00D8 |
| INTC_PSR156_159 | 0x00DC |
| INTC_PSR160_163 | 0x00E0 |
| INTC_PSR164_167 | 0x00E4 |
| INTC_PSR168_171 | 0x00E8 |
| INTC_PSR172_175 | 0x00EC |
| INTC_PSR176_179 | 0x00F0 |
| INTC_PSR180_183 | 0x00F4 |
| INTC_PSR184_187 | 0x00F8 |

**Table 26-8. INTC Priority Select Register Address Offsets (continued)**

| INTC_PSR*x*_*x* Offs | et Address |
|---|---|
| INTC_PSR68_71 | 0x0084 |
| INTC_PSR72_75 | 0x0088 |
| INTC_PSR76_79 | 0x008C |
| INTC_PSR80_83 | 0x0090 |
| INTC_PSR84_87 | 0x0094 |
| INTC_PSR88_91 | 0x0098 |
| INTC_PSR92_95 | 0x009C |
| INTC_PSR96_99 | 0x00A0 |
| INTC_PSR100_103 | 0x00A4 |
| INTC_PSR104_107 | 0x00A8 |
| INTC_PSR108_111 | 0x00AC |
| INTC_PSR112_115 | 0x00B0 |
| INTC_PSR116_119 | 0x00B4 |

| INTC_PSR*x*_*x* Offset | Address |
|---|---|
| INTC_PSR188_191 | 0x00FC |
| INTC_PSR192_195 | 0x0100 |
| INTC_PSR196_199 | 0x0104 |
| INTC_PSR200_203 | 0x0108 |
| INTC_PSR204_207 | 0x010C |
| INTC_PSR208_211 | 0x0110 |
| INTC_PSR212_215 | 0x0114 |
| INTC_PSR216_219 | 0x0118 |
| INTC_PSR220_223 | 0x011C |
| INTC_PSR224_227 | 0x0120 |
| INTC_PSR228_231 | 0x0124 |
| INTC_PSR232_235 | 0x0128 |
| INTC_PSR236_238 | 0x012C |

## 26.6 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor.

**NOTE**

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software settable interrupt request, whose PRI*n* value in INTC_PSR0_3–INTC_PSR236_238 is higher than the PRI value in INTC_CPR, negates before the interrupt request to the processor for that peripheral or software settable interrupt request is acknowledged, the interrupt request to the processor still can assert or will remain asserted for that peripheral or software settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software settable interrupt request. Also, the PRI value in the INTC_CPR will be updated with the corresponding PRI*n* value in INTC_PSR*n*. Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral or, alternatively, setting its mask bit has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

**Table 26-9. Interrupt vectors**

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| Section A (CPU section) | | | | |
| — | IVOR0 | 16 | Critical Input<br>(INTC software vector mode) | CPU |
| — | IVOR1 | 16 | Machine check / NMI | CPU |
| — | IVOR2 | 16 | Data Storage | CPU |
| — | IVOR3 | 16 | Instruction Storage | CPU |
| — | IVOR4/<br>IVPR | 16 | External Input<br>(INTC software vector mode) | CPU |
| — | IVOR5 | 16 | Alignment | CPU |
| — | IVOR6 | 16 | Program | CPU |
| — | IVOR7 | 16 | Floating-point unavailable | CPU |
| — | IVOR8 | 16 | System call | CPU |
| — | IVOR9 | 96 | AP unavailable | CPU |
| — | IVOR10 | 16 | Debug | CPU |
| — | IVOR11 | | Fixed interval timer | CPU |
| — | IVOR12 | | Watchdog timer | CPU |
| — | IVOR13 | | Data TLB error | CPU |
| — | IVOR14 | | Instruction TLB error | CPU |
| — | IVOR15 | | Debug | CPU |
| — | IVOR16–<br>IVOR31 | | Reserved | |
| — | IVOR32 | | SPE unavailable exception | Core |
| — | IVOR33 | | EFP data exception | Core |
| — | IVOR34 | | EFP round exception | Core |
| Section B (On-Platform Peripherals) | | | | |
| 0 | 0x0000 | 16 | Software setable flag 0 | Software |
| 1 | 0x0010 | 16 | Software setable flag 1 | Software |
| 2 | 0x0020 | 16 | Software setable flag 2 | Software |
| 3 | 0x0030 | 16 | Software setable flag 3 | Software |

**Table 26-9. Interrupt vectors (continued)**

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| 4 | 0x0040 | 16 | Software setable flag 4 | Software |
| 5 | 0x0050 | 16 | Software setable flag 5 | Software |
| 6 | 0x0060 | 16 | Software setable flag 6 | Software |
| 7 | 0x0070 | 16 | Software setable flag 7 | Software |
| 8–9 | 0x0080–0x0090 | 32 | Reserved | |
| 10 | 0x00A0 | 16 | Combined Error | eDMA |
| 11 | 0x00B0 | 16 | Channel 0 | eDMA |
| 12 | 0x00C0 | 16 | Channel 1 | eDMA |
| 13 | 0x00D0 | 16 | Channel 2 | eDMA |
| 14 | 0x00E0 | 16 | Channel 3 | eDMA |
| 15 | 0x00F0 | 16 | Channel 4 | eDMA |
| 16 | 0x0100 | 16 | Channel 5 | eDMA |
| 17 | 0x0110 | 16 | Channel 6 | eDMA |
| 18 | 0x0120 | 16 | Channel 7 | eDMA |
| 19 | 0x0130 | 16 | Channel 8 | eDMA |
| 20 | 0x0140 | 16 | Channel 9 | eDMA |
| 21 | 0x0150 | 16 | Channel 10 | eDMA |
| 22 | 0x0160 | 16 | Channel 11 | eDMA |
| 23 | 0x0170 | 16 | Channel 12 | eDMA |
| 24 | 0x0180 | 16 | Channel 13 | eDMA |
| 25 | 0x0190 | 16 | Channel 14 | eDMA |
| 26 | 0x01A0 | 16 | Channel 15 | eDMA |
| 27 | 0x01B0 | 16 | | Reserved |
| 28 | 0x01C0 | 16 | Timeout | Software Watchdog (SWT0) |
| 29 | 0x01D0 | 16 | | Reserved |
| 30 | 0x01E0 | 16 | Match on channel 0 | STM |
| 31 | 0x01F0 | 16 | Match on channel 1 | STM |

**Table 26-9. Interrupt vectors (continued)**

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| 32 | 0x0200 | 16 | Match on channel 2 | STM |
| 33 | 0x0210 | 16 | Match on channel 3 | STM |
| 34 | 0x0220 | 16 | | Reserved |
| 35 | 0x0230 | 16 | ECC_DBD_PlatformFlash \| ECC_DBD_PlatformRAM | ECSM |
| 36 | 0x0240 | 16 | ECC_SBC_PlatformFlash \| ECC_SBC_PlatformRAM | ECSM |
| 37 | 0x0250 | 16 | | Reserved |
| Section C | | | | |
| 38 | 0x0260 | 16 | RTC | Real Time Clock (RTC) |
| 39 | 0x0270 | 16 | API | Autonomous Periodic Interrupt (API) |
| 40 | 0x0280 | 16 | | Reserved |
| 41 | 0x0290 | 16 | SIU External IRQ_0 | System Integration Unit Lite (SIUL) |
| 42 | 0x02A0 | 16 | SIU External IRQ_1 | System Integration Unit Lite (SIUL) |
| 43 | 0x02B0 | 16 | SIU External IRQ_2 | System Integration Unit Lite (SIUL) |
| 44 | 0x02C0 | 16 | | Reserved |
| 45 | 0x02D0 | 16 | | Reserved |
| 46 | 0x02E0 | 16 | WakeUp_IRQ_0 | WKPU |
| 47 | 0x02F0 | 16 | WakeUp_IRQ_1 | WKPU |
| 48 | 0x0300 | 16 | WakeUp_IRQ_2 | WKPU |
| 49 | 0x0310 | 16 | WakeUp_IRQ_3 | WKPU |
| 50 | 0x0320 | 16 | | Reserved |
| 51 | 0x0330 | 16 | Safe Mode Interrupt | MC_ME |
| 52 | 0x0340 | 16 | Mode Transition Interrupt | MC_ME |
| 53 | 0x0350 | 16 | Invalid Mode Interrupt | MC_ME |
| 54 | 0x0360 | 16 | Invalid Mode Config | MC_ME |
| 55 | 0x0370 | 16 | | Reserved |
| 56 | 0x0380 | 16 | Functional and destructive reset alternate event interrupt (ipi_int) | MC_RGM |

**Table 26-9. Interrupt vectors (continued)**

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| 57 | 0x0390 | 16 | XOSC counter expired (ipi_int_osc) | XOSC |
| 58 | 0x03A0 | 16 | | Reserved |
| 59 | 0x03B0 | 16 | PITimer Channel 0 | Periodic Interrupt Timer (PIT) |
| 60 | 0x03C0 | 16 | PITimer Channel 1 | Periodic Interrupt Timer (PIT) |
| 61 | 0x03D0 | 16 | PITimer Channel 2 | Periodic Interrupt Timer (PIT) |
| 62 | 0x03E0 | 16 | ADC_EOC | Analog to Digital Converter 0 (ADC0) |
| 63 | 0x03F0 | 16 | ADC_ER | Analog to Digital Converter 0 (ADC0) |
| 64 | 0x0400 | 16 | ADC_WD | Analog to Digital Converter 0 (ADC0) |
| 65 | 0x0410 | 16 | FLEXCAN_ESR[ERR_INT] | FlexCAN 0 (CAN0) |
| 66 | 0x0420 | 16 | FLEXCAN_ESR_BOFF \| FLEXCAN_Transmit_Warning \| FLEXCAN_Receive_Warning | FlexCAN 0 (CAN0) |
| 67 | 0x0430 | 16 | | Reserved |
| 68 | 0x0440 | 16 | FLEXCAN_BUF_00_03 | FlexCAN 0 (CAN0) |
| 69 | 0x0450 | 16 | FLEXCAN_BUF_04_07 | FlexCAN 0 (CAN0) |
| 70 | 0x0460 | 16 | FLEXCAN_BUF_08_11 | FlexCAN 0 (CAN0) |
| 71 | 0x0470 | 16 | FLEXCAN_BUF_12_15 | FlexCAN 0 (CAN0) |
| 72 | 0x0480 | 16 | FLEXCAN_BUF_16_31 | FlexCAN 0 (CAN0) |
| 73 | 0x0490 | 16 | FLEXCAN_BUF_32_63 | FlexCAN 0 (CAN0) |
| 74 | 0x04A0 | 16 | DSPI_SR[TFUF]<br>DSPI_SR[RFOF] | DSPI 0 |
| 75 | 0x04B0 | 16 | DSPI_SR[EOQF] | DSPI 0 |
| 76 | 0x04C0 | 16 | DSPI_SR[TFFF] | DSPI 0 |
| 77 | 0x04D0 | 16 | DSPI_SR[TCF] | DSPI 0 |
| 78 | 0x04E0 | 16 | DSPI_SR[RFDF] | DSPI 0 |
| 79 | 0x04F0 | 16 | LINFlex_RXI | LINFlex 0 |
| 80 | 0x0500 | 16 | LINFlex_TXI | LINFlex 0 |
| 81 | 0x0510 | 16 | LINFlex_ERR | LINFlex 0 |
| 82 | 0x0520 | 16 | | Reserved |

**Table 26-9. Interrupt vectors (continued)**

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| 83 | 0x0530 | 16 | | Reserved |
| 84 | 0x0540 | 16 | | Reserved |
| 85 | 0x0550 | 16 | FLEXCAN_ESR[ERR_INT] | FlexCAN 1 (CAN1) |
| 86 | 0x0560 | 16 | FLEXCAN_ESR_BOFF \| FLEXCAN_Transmit_Warning \| FLEXCAN_Receive_Warning | FlexCAN 1 (CAN1) |
| 87 | 0x0570 | 16 | | Reserved |
| 88 | 0x0580 | 16 | FLEXCAN_BUF_00_03 | FlexCAN 1 (CAN1) |
| 89 | 0x0590 | 16 | FLEXCAN_BUF_04_07 | FlexCAN 1 (CAN1) |
| 90 | 0x05A0 | 16 | FLEXCAN_BUF_08_11 | FlexCAN 1 (CAN1) |
| 91 | 0x05B0 | 16 | FLEXCAN_BUF_12_15 | FlexCAN 1 (CAN1) |
| 92 | 0x05C0 | 16 | FLEXCAN_BUF_16_31 | FlexCAN 1 (CAN1) |
| 93 | 0x05D0 | 16 | FLEXCAN_BUF_32_63 | FlexCAN 1 (CAN1) |
| 94 | 0x05E0 | 16 | DSPI_SR[TFUF]<br>DSPI_SR[RFOF] | DSPI 1 |
| 95 | 0x05F0 | 16 | DSPI_SR[EOQF] | DSPI 1 |
| 96 | 0x0600 | 16 | DSPI_SR[TFFF] | DSPI 1 |
| 97 | 0x0610 | 16 | DSPI_SR[TCF] | DSPI 1 |
| 98 | 0x0620 | 16 | DSPI_SR[RFDF] | DSPI 1 |
| 99 | 0x0630 | 16 | LINFlex_RXI | LINFlex 1 |
| 100 | 0x0640 | 16 | LINFlex_TXI | LINFlex 1 |
| 101 | 0x0650 | 16 | LINFlex_ERR | LINFlex 1 |
| 102 | 0x0660 | 16 | | Reserved |
| 103 | 0x0670 | 16 | | Reserved |
| 104 | 0x0680 | 16 | | Reserved |
| 105 | 0x0690 | 16 | FLEXCAN_ESR[ERR_INT] | FlexCan 2 (CAN2) |
| 106 | 0x06A0 | 16 | FLEXCAN_ESR_BOFF \| FLEXCAN_Transmit_Warning \| FLEXCAN_Receive_Warning | FlexCan 2 (CAN2) |
| 107 | 0x06B0 | 16 | | Reserved |

**Table 26-9. Interrupt vectors (continued)**

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| 108 | 0x06C0 | 16 | FLEXCAN_BUF_00_03 | FlexCan 2 (CAN2) |
| 109 | 0x06D0 | 16 | FLEXCAN_BUF_04_07 | FlexCan 2 (CAN2) |
| 110 | 0x06E0 | 16 | FLEXCAN_BUF_08_11 | FlexCan 2 (CAN2) |
| 111 | 0x06F0 | 16 | FLEXCAN_BUF_12_15 | FlexCan 2 (CAN2) |
| 112 | 0x0700 | 16 | FLEXCAN_BUF_16_31 | FlexCan 2 (CAN2) |
| 113 | 0x0710 | 16 | FLEXCAN_BUF_32_63 | FlexCan 2 (CAN2) |
| 114 | 0x0720 | 16 | DSPI_SR[TFUF]<br>DSPI_SR[RFOF] | DSPI 2 |
| 115 | 0x0730 | 16 | DSPI_SR[EOQF] | DSPI 2 |
| 116 | 0x0740 | 16 | DSPI_SR[TFFF] | DSPI 2 |
| 117 | 0x0750 | 16 | DSPI_SR[TCF] | DSPI 2 |
| 118 | 0x0760 | 16 | DSPI_SR[RFDF] | DSPI 2 |
| 119 | 0x0770 | 16 | LINFlex_RXI | LINFlex 2 |
| 120 | 0x0780 | 16 | LINFlex_TXI | LINFlex 2 |
| 121 | 0x0790 | 16 | LINFlex_ERR | LINFlex 2 |
| 122 | 0x07A0 | 16 | LINFlex_RXI | LINFlex 3 |
| 123 | 0x07B0 | 16 | LINFlex_TXI | LINFlex 3 |
| 124 | 0x07C0 | 16 | LINFlex_ERR | LINFlex 3 |
| 125 | 0x07D0 | 16 | IBIF | Inter-IC Bus Interface Controller 0 (I2C0) |
| 126 | 0x07E0 | 16 | IBIF | Inter-IC bus interface Controller 1 (I2C1) |
| 127 | 0x07F0 | 16 | PITimer Channel 3 | Periodic Interrupt Timer (PIT) |
| 128 | 0x0800 | 16 | PITimer Channel 4 | Periodic Interrupt Timer (PIT) |
| 129 | 0x0810 | 16 | PITimer Channel 5 | Periodic Interrupt Timer (PIT) |
| 130 | 0x0820 | 16 | PITimer Channel 6 | Periodic Interrupt Timer (PIT) |
| 131 | 0x0830 | 16 | PITimer Channel 7 | Periodic Interrupt Timer (PIT) |
| 132 | 0x0840 | 16 | | Reserved |
| 133 | 0x0850 | 16 | | Reserved |
| 134 | 0x0860 | 16 | | Reserved |

**Table 26-9. Interrupt vectors (continued)**

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| 135 | 0x0870 | 16 | | Reserved |
| 136 | 0x0880 | 16 | | Reserved |
| 137 | 0x0890 | 16 | | Reserved |
| 138 | 0x08A0 | 16 | | Reserved |
| 139 | 0x08B0 | 16 | | Reserved |
| 140 | 0x08C0 | 16 | | Reserved |
| 141 | 0x08D0 | 16 | EMIOS0_GFR[F8] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 142 | 0x08E0 | 16 | EMIOS0_GFR[F9] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 143 | 0x08F0 | 16 | EMIOS0_GFR[F10] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 144 | 0x0900 | 16 | EMIOS0_GFR[F11] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 145 | 0x0910 | 16 | EMIOS0_GFR[F12] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 146 | 0x0920 | 16 | EMIOS0_GFR[F13] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 147 | 0x0930 | 16 | EMIOS0_GFR[F14] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 148 | 0x0940 | 16 | EMIOS0_GFR[F15] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 149 | 0x0950 | 16 | EMIOS0_GFR[F16] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 150 | 0x0960 | 16 | EMIOS0_GFR[F17] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 151 | 0x0970 | 16 | EMIOS0_GFR[F18] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 152 | 0x0980 | 16 | EMIOS0_GFR[F19] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 153 | 0x0990 | 16 | EMIOS0_GFR[F20] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 154 | 0x09A0 | 16 | EMIOS0_GFR[F21] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |

**Table 26-9. Interrupt vectors (continued)**

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| 155 | 0x09B0 | 16 | EMIOS0_GFR[F22] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| 156 | 0x09C0 | 16 | EMIOS0_GFR[F23] | Enhanced Modular I/O Subsystem 0 (eMIOS0) |
| Section D | | | | |
| 157 | 0x09D0 | 16 | EMIOS1_GFR[F8] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 158 | 0x09E0 | 16 | EMIOS1_GFR[F9] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 159 | 0x09F0 | 16 | EMIOS1_GFR[F10] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 160 | 0x0A00 | 16 | EMIOS1_GFR[F11] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 161 | 0x0A10 | 16 | EMIOS1_GFR[F12] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 162 | 0x0A20 | 16 | EMIOS1_GFR[F13] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 163 | 0x0A30 | 16 | EMIOS1_GFR[F14] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 164 | 0x0A40 | 16 | EMIOS1_GFR[F15] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 165 | 0x0A50 | 16 | EMIOS1_GFR[F16] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 166 | 0x0A60 | 16 | EMIOS1_GFR[F17] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 167 | 0x0A70 | 16 | EMIOS1_GFR[F18] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 168 | 0x0A80 | 16 | EMIOS1_GFR[F19] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 169 | 0x0A90 | 16 | EMIOS1_GFR[F20] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 170 | 0x0AA0 | 16 | EMIOS1_GFR[F21] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 171 | 0x0AB0 | 16 | EMIOS1_GFR[F22] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |

**Table 26-9. Interrupt vectors (continued)**

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| 172 | 0x0AC0 | 16 | EMIOS1_GFR[F23] | Enhanced Modular I/O Subsystem 1 (eMIOS1) |
| 173 | 0x0AD0 | 16 | IBIF | Inter-IC bus interface Controller 2 (I2C2) |
| 174 | 0x0AE0 | 16 | IBIF | Inter-IC Bus Interface Controller 3 (I2C3) |
| 175 | 0x0AF0 | 16 | VIU_IRQ | Video Input Unit (VIU2) |
| 176 | 0x0B00 | 16 | | Reserved |
| 177 | 0x0B10 | 16 | | Reserved |
| 178 | 0x0B20 | 16 | | Reserved |
| 179 | 0x0B30 | 16 | | Reserved |
| 180 | 0x0B40 | 16 | FIFO OV/UV | DRAM Controller |
| 181 | 0x0B50 | 16 | INT | DRAM Priority Manager |
| 182 | 0x0B60 | 16 | | Reserved |
| 183 | 0x0B70 | 16 | SGM_IRG | Sound Generator Module (SGM) |
| 184 | 0x0B80 | 16 | VS_BLANK, LS_BF_VS, VSYNC | Display Control Unit (DCU3) |
| 185 | 0x0B90 | 16 | UNDRUN | Display Control Unit (DCU3) |
| 186 | 0x0BA0 | 16 | PARERR | Display Control Unit (DCU3) |
| 187 | 0x0BB0 | 16 | PDI | Display Control Unit (DCU3) |
| 188 | 0x0BC0 | 16 | VS_BLANK, LS_BF_VS, VSYNC | Display Control Unit Lite (DCULite) |
| 189 | 0x0BD0 | 16 | UNDRUN | Display Control Unit Lite (DCULite) |
| 190 | 0x0BE0 | 16 | PARERR | Display Control Unit Lite (DCULite) |
| 191 | 0x0BF0 | 16 | PDI | Display Control Unit Lite (DCULite) |
| 192 | 0x0C00 | 16 | MCTOI, SCDetect[0:23] | Stepper Motor Controller (SMC0) |
| 193 | 0x0C10 | 16 | MCZI, AOVI | Stepper Stall Detect 0 (SSD0) |
| 194 | 0x0C20 | 16 | MCZI, AOVI | Stepper Stall Detect 1 (SSD1) |
| 195 | 0x0C30 | 16 | MCZI, AOVI | Stepper Stall Detect 2 (SSD2) |
| 196 | 0x0C40 | 16 | MCZI, AOVI | Stepper Stall Detect 3 (SSD3) |
| 197 | 0x0C50 | 16 | MCZI, AOVI | Stepper Stall Detect 4 (SSD4) |
| 198 | 0x0C60 | 16 | MCZI, AOVI | Stepper Stall Detect 5 (SSD5) |

Table 26-9. Interrupt vectors (continued)

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| 199 | 0x0C70 | 16 | | Reserved |
| 200 | 0x0C80 | 16 | | Reserved |
| 201 | 0x0C90 | 16 | | Reserved |
| 202 | 0x0CA0 | 16 | | Reserved |
| 203 | 0x0CB0 | 16 | | Reserved |
| 204 | 0x0CC0 | 16 | | Reserved |
| 205 | 0x0CD0 | 16 | | Reserved |
| 206 | 0x0CE0 | 16 | | Reserved |
| 207 | 0x0CF0 | 16 | RLE_INT | RLE Decoder |
| 208 | 0x0D00 | 16 | | Reserved |
| 209 | 0x0D10 | 16 | | Reserved |
| 210 | 0x0D20 | 16 | | Reserved |
| 211 | 0x0D30 | 16 | | Reserved |
| 212 | 0x0D40 | 16 | | Reserved |
| 213 | 0x0D50 | 16 | | Reserved |
| 214 | 0x0D60 | 16 | | Reserved |
| 215 | 0x0D70 | 16 | | Reserved |
| 216 | 0x0D80 | 16 | | Reserved |
| 217 | 0x0D90 | 16 | | Reserved |
| 218 | 0x0DA0 | 16 | | Reserved |
| 219 | 0x0DB0 | 16 | | Reserved |
| 220 | 0x0DC0 | 16 | | Reserved |
| 221 | 0x0DD0 | 16 | | Reserved |
| 222 | 0x0DE0 | 16 | | Reserved |
| 223 | 0x0DF0 | 16 | | Reserved |
| 224 | 0x0E00 | 16 | | Reserved |
| 225 | 0x0E10 | 16 | | Reserved |

Table 26-9. Interrupt vectors (continued)

| IRQ # | Offset | Size (bytes) | Resource | Module |
|---|---|---|---|---|
| 226 | 0x0E20 | 16 | | Reserved |
| 227 | 0x0E30 | 16 | | Reserved |
| 228 | 0x0E40 | 16 | Overrun | QuadSPI |
| 229 | 0x0E50 | 16 | | Reserved |
| 230 | 0x0E60 | 16 | TFFF | QuadSPI |
| 231 | 0x0E70 | 16 | TCF | QuadSPI |
| 232 | 0x0E80 | 16 | RFDF | QuadSPI |
| 233 | 0x0E90 | 16 | CERR | QuadSPI |
| 234 | 0x0EA0 | 16 | | Reserved |
| 235 | 0x0EB0 | 16 | | Reserved |
| 236 | 0x0EC0 | 16 | | Reserved |
| 237 | 0x0ED0 | 16 | | Reserved |
| 238 | 0x0EE0 | 16 | IRQ_0 | GFX2D |

## 26.6.1 Interrupt Request Sources

The INTC has two types of interrupt requests, peripheral and software configurable. These interrupt requests can assert on any clock cycle.

### 26.6.1.1 Peripheral Interrupt Requests

An interrupt event in a peripheral's hardware sets a flag bit that resides in the peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC to the time that the INTC starts to drive the interrupt request to the processor is three clocks.

External interrupts are handled by the SIU (see Section 43.6.4, "External interrupts").

### 26.6.1.2 Software configurable Interrupt Requests

An interrupt request is triggered by software by writing a 1 to a SET*x* bit in INTC_SSCIR0_3–INTC_SSCIR4_7. This write sets the corresponding flag bit, CLR*x*, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR*x* bit.

The time from the write to the SET*x* bit to the time that the INTC starts to drive the interrupt request to the processor is four clocks.

#### 26.6.1.3 Unique Vector for Each Interrupt Request Source

Each peripheral and software configurable interrupt request is assigned a hardwired unique 9-bit vector. Software configurable interrupts 0–7 are assigned vectors 0–7 respectively. The peripheral interrupt requests are assigned vectors 8 to as high as needed to include all the peripheral interrupt requests. The peripheral interrupt request input ports at the boundary of the INTC block are assigned specific hardwired vectors within the INTC.

### 26.6.2 Priority Management

The asserted interrupt requests are compared to each other based on their PRI*x* values set in INTC_PSR0_3–INTC_PSR236_238. The result is compared to PRI in the associated INTC_CPR. The results of those comparisons manage the priority of the ISR executed by the associated processor. The associated LIFO also assists in managing that priority.

#### 26.6.2.1 Current Priority and Preemption

The priority arbitrator, selector, encoder, and comparator subblocks shown in Figure 26-1 compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software configurable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software settable interrupt request is generated for INTC interrupt acknowledge register (INTC_IACKR), and if in hardware vector mode, for the interrupt vector provided to the processor.

##### 26.6.2.1.1 Priority Arbitrator Subblock

The priority arbitrator subblock for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software configurable. The output of the priority arbitrator subblock is the highest of those priorities assigned to a given processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the associated request selector subblock.

##### 26.6.2.1.2 Request Selector Subblock

If only one interrupt request from the associated priority arbitrator subblock is asserted, then it is passed as asserted to the associated vector encoder subblock. If multiple interrupt requests from the associated priority arbitrator subblock are asserted, only the one with the lowest vector passes as asserted to the associated vector encoder subblock. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software configurable interrupt requests.

##### 26.6.2.1.3 Vector Encoder Subblock

The vector encoder subblock generates the unique 9-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

#### 26.6.2.1.4 Priority Comparator Subblock

The priority comparator submodule compares the highest priority output from the priority arbitrator submodule with PRI in INTC_CPR. If the priority comparator submodule detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in INTC_CPR or the PRI value in INTC_CPR is lowered below this highest priority. This highest priority then becomes the new priority which will be written to PRI in INTC_CPR when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI*n* in INTC_PSR*n* are zero will not cause a preemption because their PRI*n* will not be higher than PRI in INTC_CPR.

### 26.6.2.2 Last-In First-Out (LIFO)

The LIFO stores the preempted PRI values from the INTC_CPR. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the INTC_CPR does not need to be loaded from the INTC_CPR and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the INTC_CPR.

The PRI value in the INTC_CPR is pushed onto the LIFO when the INTC_IACKR is read in softwarevector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode. The priority is popped into PRI in the INTC_CPR whenever the INTC_EOIR is written.

Although the INTC supports 16 priorities, an ISR executing with PRI in the INTC_CPR equal to 15 will not be preempted. Therefore, the LIFO supports the stacking of 15 priorities. However, the LIFO is only 14 entries deep. An entry for a priority of 0 is not needed because of how pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 15 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop ‘0’s if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

The LIFO is not memory mapped.

## 26.6.3 Handshaking with Processor

### 26.6.3.1 Software Vector Mode Handshaking

This section describes handshaking in software vector mode.

#### 26.6.3.1.1 Acknowledging Interrupt Request to Processor

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode and the handshake near the end of the interrupt exception handler, is shown in Figure 26-10. The INTC examines the peripheral and software configurable interrupt requests. When it finds an asserted peripheral or software configurable interrupt request with a higher priority than PRI in the associated INTC_CPR, it asserts the interrupt request to the processor. The INTVEC field in the associated INTC_IACKR is updated with the preempting interrupt request’s vector when the interrupt request to the processor is asserted. The

INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of handshaking process is described in Section 26.4.1.1, “Software vector mode”.

#### 26.6.3.1.2 End of Interrupt Exception Handler

Before the interrupt exception handling completes, INTC end-of-interrupt register (INTC_EOIR) must be written.When written, the associated LIFO is popped so the preempted priority is restored into PRI of the INTC_CPR. Before it is written, the peripheral or software configurable flag bit must be cleared so that the peripheral or software configurable interrupt request is negated.

**NOTE**

To ensure proper operation across all eSys MCUs, execute an MBAR or MSYNC instruction between the access to clear the flag bit and the write to the INTC_EOIR.

When returning from the preemption, the INTC does not search for the peripheral or software settable interrupt request whose ISR was preempted. Depending on how much the ISR progressed, that interrupt request may no longer even be asserted. When PRI in INTC_CPR is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it was to next execute before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler’s prolog or epilog.

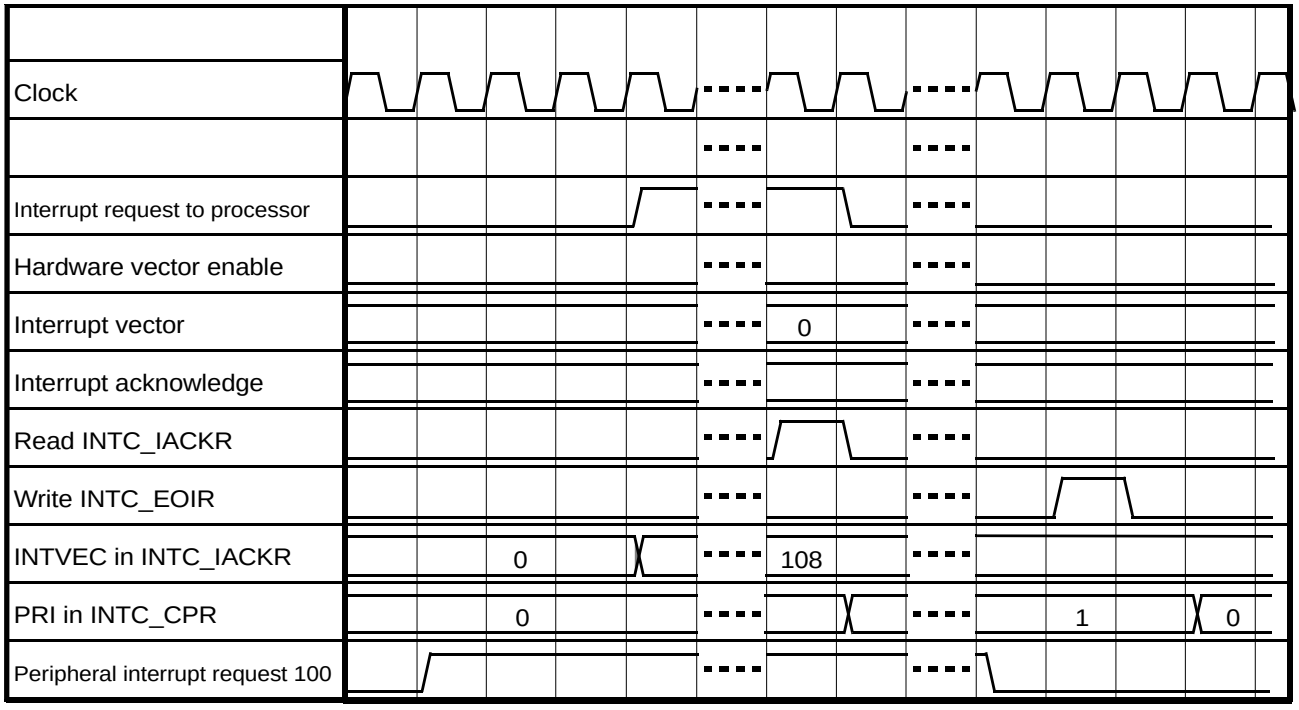



**Figure 26-10. Software Vector Mode Handshaking Timing Diagram**

### 26.6.3.2 Hardware Vector Mode Handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 26-11. As in software vector mode, the INTC examines the peripheral and software settable interrupt requests, and when it finds an asserted one with a higher priority than PRI in INTC_CPR, it asserts the interrupt request to the processor. The INTVEC field in the INTC_IACKR is updated with the preempting peripheral or software settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. In addition, the value of the interrupt vector to the processor matches the value of the INTVEC field in the INTC_IACKR. The rest of the handshaking is described in Section 26.4.1.2, "Hardware vector mode."

The handshaking near the end of the interrupt exception handler, that is the writing to the INTC_EOIR, is the same as in software vector mode. Refer to Section 26.6.3.1.2, "End of Interrupt Exception Handler."

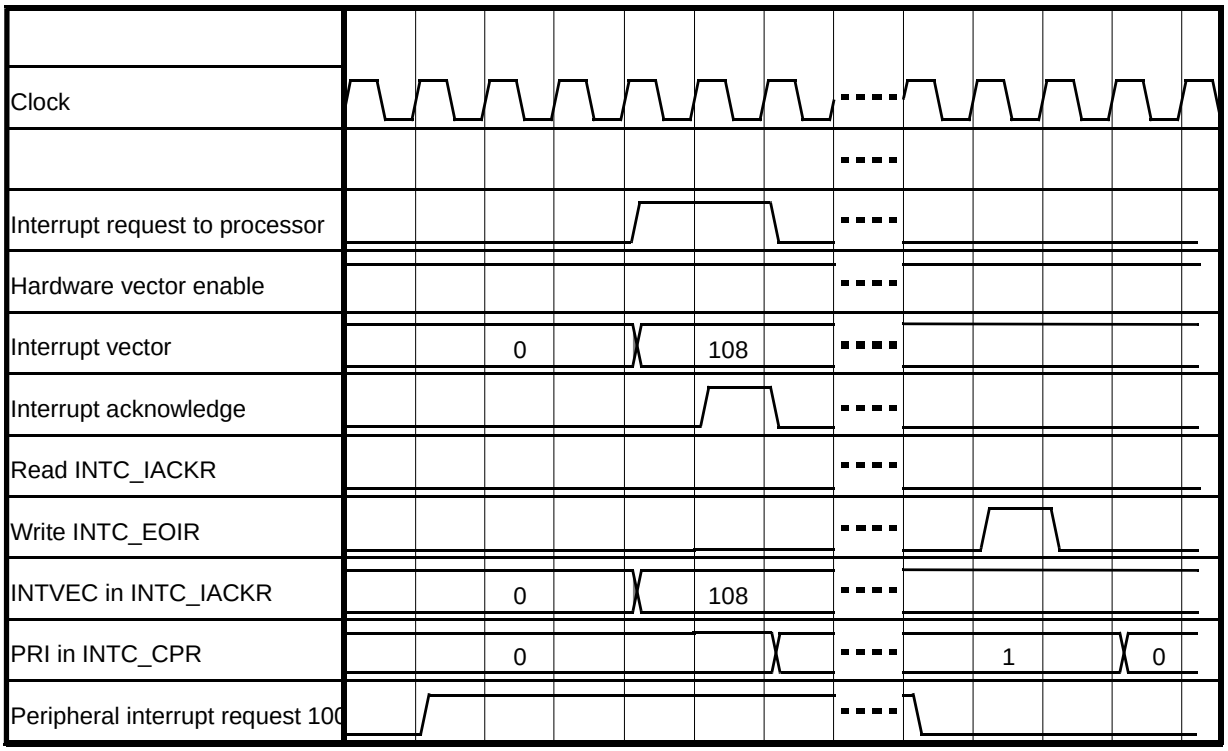



**Figure 26-11. Hardware Vector Mode Handshaking Timing Diagram**

## 26.7 Initialization/Application Information

### 26.7.1 Initialization Flow

After exiting reset, all of the PRI*n* fields in INTC priority select registers (INTC_PSR0_3–INTC_PSR236_238) will be zero, and PRI in INTC current priority register (INTC_CPR) will be 15. These reset values will prevent the INTC from asserting the interrupt request to

the processor. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software settable interrupt requests to cause an interrupt request to the processor is:interrupt_request_initialization:

```
interrupt_request_initialization:
configure VTES and HVEN in INTC_MCR
configure VTBA in INTC_IACKR
raise the PRIn fields in INTC_PSRn
set the enable bits or clear the mask bits for the peripheral interrupt requests
lower PRI in INTC_CPR to zero
enable processor recognition of interrupts
```

## 26.7.2 Interrupt Exception Handler

These example interrupt exception handlers use Power Architecture assembly code.

### 26.7.2.1 Software Vector Mode

```
interrupt_exception_handler:
code to create stack frame, save working register, and save SRR0 and SRR1
lis      r3,INTC_IACKR@ha           # form adjusted upper half of INTC_IACKR address
lwz      r3,INTC_IACKR@l(r3)        # load INTC_IACKR, which clears request to processor
lwz      r3,0x0(r3)                 # load address of ISR from vector table
wrteei 1                            # enable processor recognition of interrupts

code to save rest of context required by e500 EABI

mtlr     r3                         # move INTC_IACKR contents into link register
blrl                                # branch to ISR; link register updated with epilog
                                    # address

epilog:
code to restore most of context required by e500 EABI

# Popping the LIFO after the restoration of most of the context and the disabling of processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                                # ensure store to clear flag bit has completed
lis      r3,INTC_EOIR@ha            # form adjusted upper half of INTC_EOIR address
li       r4,0x0                     # form 0 to write to INTC_EOIR
wrteei   0                          # disable processor recognition of interrupts
stw      r4,INTC_EOIR@l(r3)         # store to INTC_EOIR, informing INTC to lower priority

code to restore SRR0 and SRR1, restore working registers, and delete stack frame

rfi


vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
.
.
.
address of ISR for interrupt with vector 510
address of ISR for interrupt with vector 511
```

```
ISRx:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr # return to epilog
```

### 26.7.2.2 Hardware Vector Mode

This interrupt exception handler is useful with processor and system bus implementations which support a hardware vector. This example assumes that each interrupt_exception_handler*x* only has space for four instructions, and therefore a branch to interrupt_exception_handler_continued*x* is needed.

```
interrupt_exception_handlerx:
b interrupt_exception_handler_continuedx# 4 instructions available, branch to continue
interrupt_exception_handler_continuedx:
code to create stack frame, save working register, and save SRR0 and SRR1

wrteei   1                # enable processor recognition of interrupts

code to save rest of context required by e500 EABI

bl       ISRx             # branch to ISR for interrupt with vector x


epilog:
code to restore most of context required by e500 EABI

# Popping the LIFO after the restoration of most of the context and the disabling of processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                      # ensure store to clear flag bit has completed
lis      r3,INTC_EOIR@ha  # form adjusted upper half of INTC_EOIR address
li       r4,0x0           # form 0 to write to INTC_EOIR
wrteei   0                # disable processor recognition of interrupts
stw      r4,INTC_EOIR@l(r3) # store to INTC_EOIR, informing INTC to lower priority

code to restore SRR0 and SRR1, restore working registers, and delete stack frame

rfi

ISRx:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC
blr                       # branch to epilog
```

## 26.7.3 ISR, RTOS, and Task Hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC current priority register (INTC_CPR) having a value of 0. The RTOS will execute the tasks according to whatever priority scheme that it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR priority 0 and outside

the control of the RTOS, the RTOS executes at INTC_CPR priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPR priority 0.

If a task shares a resource with an ISR and the PCP is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR while the shared resource is being accessed.

An ISR whose PRI*n* in INTC priority select registers (INTC_PSR0_3–INTC_PSR236_238) has a value of 0 will not cause an interrupt request to the processor, even if its peripheral or software settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

### 26.7.4 Order of Execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software configurable interrupt requests. However, if multiple peripheral or software configurable interrupt requests are asserted, more than one has the highest priority, and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software configurable interrupt requests asserted.

The example in Table 26-10 shows the order of execution of both ISRs with different priorities and the same priority

**Table 26-10. Order of ISR Execution Example**

| Step | Step Description | Code Executing at End of Step | | | | | | PRI in INTC_CPR at End of Step |
|---|---|---|---|---|---|---|---|---|
| | | RTOS | SR108[1] | ISR208 | ISR308 | ISR408 | Interrupt Exception Handler | |
| 1 | RTOS at priority 0 is executing. | X | | | | | | 0 |
| 2 | Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken. | | X | | | | | 1 |
| 3 | Peripheral interrupt request 400 at priority 4 is asserts. Interrupt taken. | | | | | X | | 4 |
| 4 | Peripheral interrupt request 300 at priority 3 is asserts. | | | | | X | | 4 |
| 5 | Peripheral interrupt request 200 at priority 3 is asserts. | | | | | X | | 4 |
| 6 | ISR408 completes. Interrupt exception handler writes to INTC_EOIR. | | | | | | X | 1 |
| 7 | Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first. | | | X | | | | 3 |

**Table 26-10. Order of ISR Execution Example (continued)**

| Step | Step Description | Code Executing at End of Step | | | | | | PRI in INTC_CPR at End of Step |
|---|---|---|---|---|---|---|---|---|
| | | RTOS | ISR108[1] | ISR208 | ISR308 | ISR408 | Interrupt Exception Handler | |
| 8 | ISR208 completes. Interrupt exception handler writes to INTC_EOIR. | | | | | | X | 1 |
| 9 | Interrupt taken. ISR308 starts to execute. | | | | X | | | 3 |
| 10 | ISR308 completes. Interrupt exception handler writes to INTC_EOIR. | | | | | | X | 1 |
| 11 | ISR108 completes. Interrupt exception handler writes to INTC_EOIR. | | | | | | X | 0 |
| 12 | RTOS continues execution. | X | | | | | | 0 |

1 ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software configurable interrupt requests.

## 26.7.5 Priority Ceiling Protocol

### 26.7.5.1 Elevating Priority

The PRI field in INTC_CPR is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR to 3, the ceiling of all of the ISR priorities. After they release the resource, the PRI value in INTC_CPR can be lowered. If they do not raise their priority, ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

### 26.7.5.2 Ensuring coherency

A scenario can cause non-coherent accesses to the shared resource. For example, ISR1 and ISR2 are both running on the same CPU and both share a resource. ISR1 has a lower priority than ISR2. ISR1 is

executing and writes to the INTC_CPR. The instruction following this store is a store to a value in a shared coherent data block. Either immediately before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible that both stores will be executed. ISR2 thereby thinks that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent corruption of a coherent data block, modifications to PRI in INTC_CPR can be made by those system services with the code sequence:

```
disable processor recognition of interrupts
PRI modification
enable processor recognition of interrupts
```

## 26.7.6 Selecting Priorities According to Request Rates and Deadlines

The selection of the priorities for the ISRs can be made using rate monotonic scheduling (RMS) or a superset of it, deadline monotonic scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3; however, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC has 16 priorities, which may be less than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, etc. With this approach, a range of ISR request rates of $2^{16}$ could be included, regardless of the number of ISRs.

Reducing the number of priorities reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They do not need to use the PCP to access the shared resource.

## 26.7.7 Software configurable Interrupt Requests

The software configurable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and they may also be used by processors to interrupt other processors in a multiple processor system.

### 26.7.7.1 Scheduling a Lower Priority Portion of an ISR

A portion of an ISR needs to be executed at the PRI*x* value in INTC_PSR0_3–INTC_PSR236_238, which becomes the PRI value in INTC_CPR with the interrupt acknowledge. The ISR, however, can have a

portion that does not need to be executed at this higher priority. Therefore, executing the later portion that does not need to be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than what the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET*x* bit in INTC_SSCIR0_3–INTC_SSCIR4_7. Writing a 1 to SET*x* causes a software configurable interrupt request. This software configurable interrupt request will usually have a lower PRI*x* value in the INTC_PSR*x*_*x* and will not cause preemptive scheduling inefficiencies. After generating a software settable interrupt request, the higher priority ISR completes. The lower priority ISR is scheduled according to its priority. Execution of the higher priority ISR is not resumed after the completion of the lower priority ISR.

### 26.7.7.2 Scheduling an ISR on Another Processor

Because the SET*x* bits in the INTC_SSCIR*x*_*x* are memory mapped, processors in multiple-processor systems can schedule ISRs on the other processors. One application is that one processor wants to command another processor to perform a piece of work and the initiating processor does not need to use the results of that work. If the initiating processor is concerned that the processor executing the software configurable ISR has not completed the work before asking it to again execute the ISR, it can check if the corresponding CLR*x* bit in INTC_SSCIR*x*_*x* is asserted before again writing a 1 to the SET*x* bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. To do this, the first processor writes a 1 to a SET*x* bit on the second processor. After accessing the block of data, the second processor clears the corresponding CLR*x* bit and then writes 1 to a SET*x* bit on the first processor, informing it that it can now access the block of data.

## 26.7.8 Lowering Priority Within an ISR

A common method for avoiding preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (see Section 26.7.7.1, "Scheduling a Lower Priority Portion of an ISR") is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

**NOTE**

Lowering the PRI value in INTC_CPR within an ISR to below the ISR's corresponding PRI value in INTC_PSR0_3–INTC_PSR236_238 allows more preemptions than the LIFO depth can support.

Therefore, the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

## 26.7.9 Negating an Interrupt Request Outside of its ISR

### 26.7.9.1 Negating an Interrupt Request as a Side Effect of an ISR

Some peripherals have flag bits that can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits and their corresponding interrupt requests. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

### 26.7.9.2 Negating Multiple Interrupt Requests in One ISR

An ISR can clear other flag bits besides its own. One reason that an ISR clears multiple flag bits is because it serviced those flag bits, and therefore the ISRs for these flag bits do not need to be executed.

### 26.7.9.3 Proper Setting of Interrupt Request Priority

Whether an interrupt request negates outside its own ISR due to the side effect of an ISR execution or the intentional clearing a flag bit, the priorities of the peripheral or software configurable interrupt requests for these other flag bits must be selected properly. Their PRI*x* values in INTC_PSR0_3–INTC_PSR236_238 must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC_SSCIR0_3–INTC_SSCIR4_7 as the clearing of the flag bit that caused the present ISR to be executed (see Section 26.6.3.1.2, “End of Interrupt Exception Handler”).

A flag bit whose enable bit or mask bit negates its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request’s PRI*x* value in INTC_PSR*x*_*x*.

## 26.7.10 Examining LIFO contents

In normal mode, the user does not need to know the contents of the LIFO. He may not even know how deeply the LIFO is nested. However, if he wants to read the contents, such as in debug mode, they are not memory mapped. The contents can be read by popping the LIFO and reading the PRI field in either INTC_CPR. The code sequence is:

```
pop_lifo:
store to INTC_EOIR
load INTC_CPR, examine PRI, and store onto stack
if PRI is not zero or value when interrupts were enabled, branch to pop_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
load stacked PRI value and store to INTC_CPR
load INTC_IACKR
if stacked PRI values are not depleted, branch to push_lifo
```

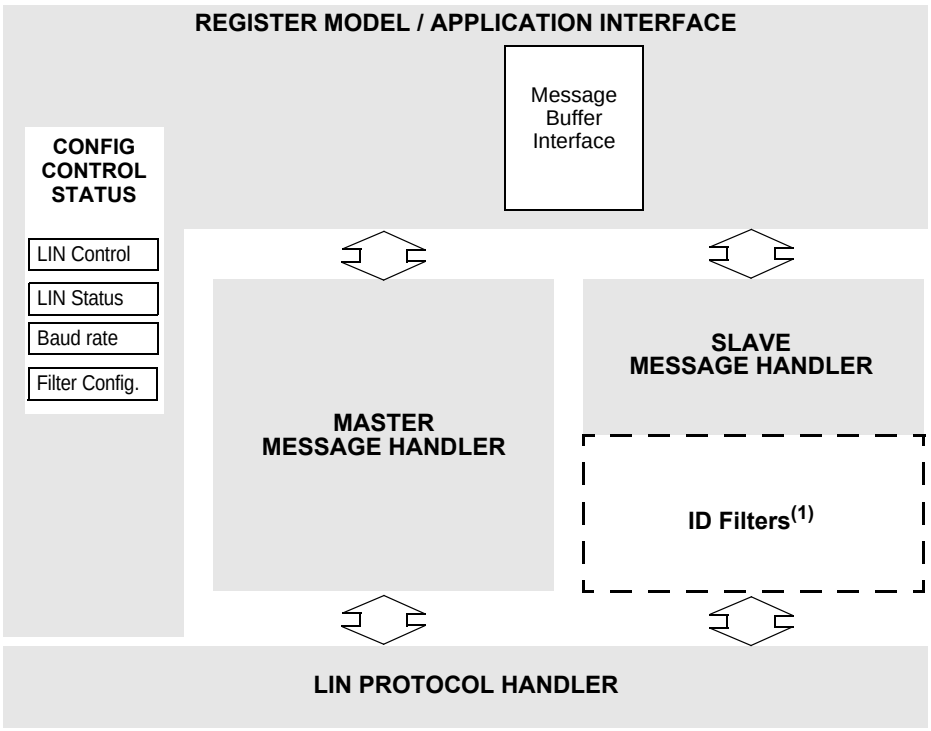
This page is intentionally left blank.

# Chapter 27
# LIN Controller (LINFlexD)

## 27.1 Introduction

The LINFlexD (Local Interconnect Network Flexible with DMA support) controller interfaces the LIN network and supports the LIN protocol versions 1.3, 2.0, 2.1 and J2602 in both Master and Slave modes. LINFlexD includes a LIN mode that provides additional features (compared to standard UART) to ease LIN implementation, improve system robustness, minimize CPU load and allow slave node resynchronization.

Figure 27-1 shows the LINFlexD block diagram.




[1] Filter activation optional

**Figure 27-1. LINFlexD block diagram**

## 27.2 Main features

The LINFlexD controller can operate in several modes, each of which has a distinct set of features. These distinct features are described in the following sections.

In addition, the LINFlexD controller has several features common to all modes:

- Fractional baud rate generator

- 3 operating modes for power saving and configuration registers lock
  - Initialization
  - Normal
  - Sleep
- 2 test modes
  - Loop Back
  - Self Test
- Maskable interrupts

### 27.2.1 LIN mode features

- Supports LIN protocol versions 1.3, 2.0, 2.1 and J2602
- Master mode with autonomous message handling
- Classic and enhanced checksum calculation and check
- Single 8-byte buffer for transmission/reception
- Extended frame mode for In-application Programming purposes
- Wake-up event on dominant bit detection
- True LIN field state machine
- Advanced LIN error detection
- Header, response and frame timeout
- Slave mode
  - Autonomous header handling
  - Autonomous transmit/receive data handling
- LIN automatic resynchronization, allowing operation with FIRC as clock source
- Identifier filters for autonomous message handling in Slave mode

### 27.2.2 UART mode features

- Full-duplex communication
- Selectable frame size:
  - 8-bit frame
  - 9-bit frame
  - 16-bit frame
  - 17-bit frame
- Selectable parity:
  - Even
  - Odd
  - 0
  - 1

- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management

## 27.3 The LIN protocol

The LIN (Local Interconnect Network) is a serial communication protocol. The topology of a LIN network is shown in Figure 27-2. A LIN network consists of:

- One master
- Several slave
- The LIN bus

A master node contains the master task as well as a slave task, all other nodes contain a slave task only. The master node decides when and which frame shall be transferred on the bus. The slave task provides the data to be transported by the frame.

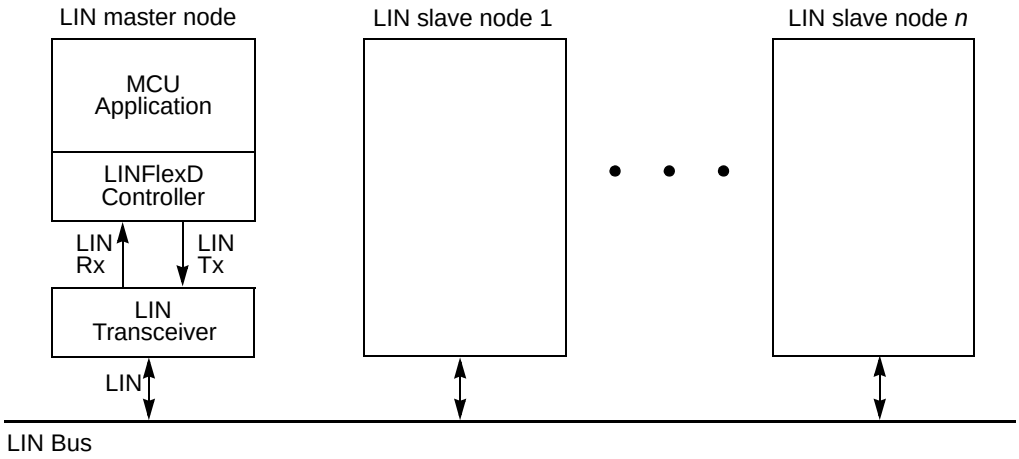



**Figure 27-2. LIN network topology**

### 27.3.1 Dominant and recessive logic levels

The LIN bus defines two logic levels, “dominant” and “recessive”, as follows:

- Dominant: logical low level (0)
- Recessive: logical high level (1)

### 27.3.2 LIN frames

A frame consists of a header provided by the master task and a response provided by the slave task, as shown in Figure 27-3.

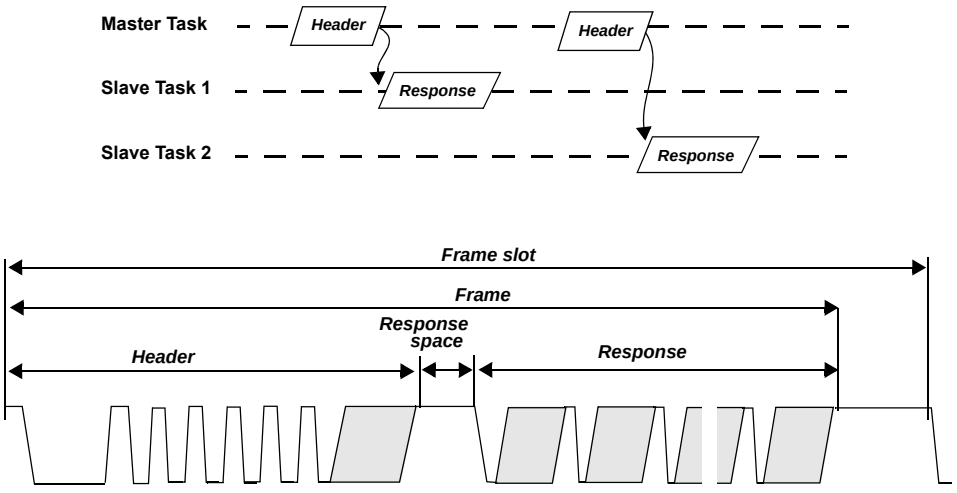



**Figure 27-3. LIN frame structure**

## 27.3.3 LIN header

The header consists of:

- A break field (described in Section 27.3.3.1, Break field)
- A sync  (described in Section 27.3.3.2, Sync)
- An identifier (described in Section 27.3.4.2, Identifier)

The slave task associated with the identifier provides the response.

### 27.3.3.1 Break field

The break field, shown in Figure 27-4, is used to signal the beginning of a new frame. It is always generated by the master and consists of:

- At least 13 dominant bits including the start bit
- At least one recessive bit that functions as break delimiter




**Figure 27-4. Break field**

### 27.3.3.2 Sync

The sync pattern is a byte consisting of alternating dominant and recessive bits as shown in Figure 27-5. It forms a data value of 0x55.




**Figure 27-5. Sync pattern**

## 27.3.4 Response

The response consists of:

- A data field (described in Section 27.3.4.1, Data field)
- A checksum (described in Section 27.3.4.3, Checksum)

The slave task interested in the data associated with the identifier receives the response and verifies the checksum.

### 27.3.4.1 Data field

The structure of the data field transmitted on the LIN bus is shown in Figure 27-6. The LSB of the data is sent first and the MSB last. The start bit is encoded as a dominant bit and the stop bit is encoded as a recessive bit.

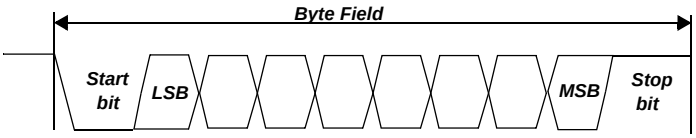



**Figure 27-6. Structure of the data field**

### 27.3.4.2 Identifier

The identifier, shown in Figure 27-7, consists of two sub-fields:

- The identifier value (in bits 0–5)
- The identifier parity (in bits 6–7)

The parity bits P0 and P1 are defined as follows:

- P0 = ID0 xor ID1 xor ID2 xor ID4
- P1 = not(ID1 xor ID3 xor ID4 xor ID5)

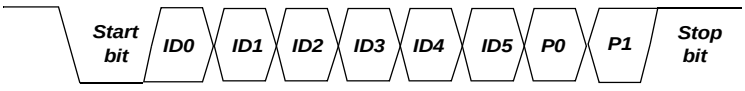



**Figure 27-7. Identifier**

### 27.3.4.3 Checksum

The checksum contains the inverted 8-bit sum (with carry) over one of two possible groups:

- The classic checksum sums all data bytes, and is used for communication with LIN 1.3 slaves.
- The enhanced checksum sums all data bytes and the identifier, and is used for communication with LIN 2.0 (or later) slaves.

## 27.4 LINFlexD and software intervention

The increasing number of communication peripherals embedded on microcontrollers (for example, CAN, LIN, SPI) requires more and more CPU resources for the communication management. Even a 32-bit microcontroller is overloaded if its peripherals do not provide high level features to autonomously handle the communication.

Even though the LIN protocol with a maximum baud rate of 20 kbps is relatively slow, it still generates a non-negligible load on the CPU if the LIN is implemented on a standard UART, as is usually the case.

To minimize the CPU load in Master mode, LINFlexD handles the LIN messages autonomously.

In Master mode, once the software has triggered the header transmission, LINFlexD does not request any software (that is, application) intervention until the next header transmission request in transmission mode or until the checksum reception in reception mode.

To minimize the CPU load in Slave mode, LINFlexD requires software intervention only to:

- Trigger transmission or reception or data discard depending on the identifier
- Write data into the buffer (transmission mode) or read data from the buffer (reception mode) after checksum reception

If filter mode is activated for Slave mode, LINFlexD requires software intervention only to write data into the buffer (transmission mode) or read data from the buffer (reception mode)

The software uses the control, status and configuration registers to:

- Configure LIN parameters (for example, baud rate or mode)
- Request transmissions
- Handle receptions
- Manage interrupts
- Configure LIN error and timeout detection
- Process diagnostic information

The message buffer stores transmitted or received LIN frames.

## 27.5 Summary of operating modes

The LINFlexD controller has three operating modes:

- Normal
- Initialization
- Sleep

After a hardware reset, the LINFlexD controller is in Sleep mode to reduce power consumption.

The transitions between these modes are shown in Figure 27-8. The software instructs LINFlexD to enter Initialization mode or Sleep mode by setting LINCR1[INIT] or LINCR1[SLEEP], respectively.

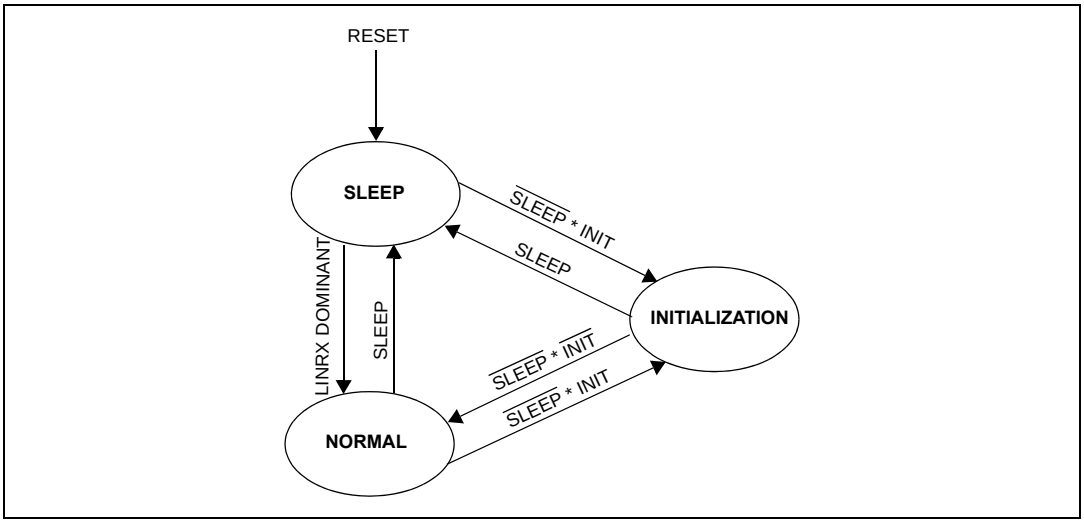



**Figure 27-8. LINFlexD controller operating modes**

In addition to these controller-level operating modes, the LINFlexD controller also supports several protocol-level modes:

- LIN mode:
  - Master mode
  - Slave mode
  - Slave mode with identifier filtering
  - Slave mode with automatic resynchronization
- UART mode
- Test modes:
  - Loop Back mode
  - Self Test mode

These modes are discussed in detail in subsequent sections.

## 27.6 Controller-level operating modes

### 27.6.1 Initialization mode

The software initialization can be done while the hardware is in Initialization mode. To enter or exit this mode, the software sets or clears LINCR1[INIT], respectively.

In Initialization mode, all message transfers to and from the LIN bus are stopped and the LIN bus output (LINTX) is recessive.

Entering Initialization mode does not change any of the configuration registers.

To initialize the LINFlexD controller, the software must:

- Select the desired mode (Master, Slave or UART)
- Set up the baud rate register
- If LIN Slave mode with filter activation is selected, initialize the identifier list

### 27.6.2 Normal mode

After initialization is complete, the software must clear LINCR1[INIT] to put the LINFlexD controller into Normal mode.

### 27.6.3 Sleep (low-power) mode

To reduce power consumption, LINFlexD has a low-power mode called Sleep mode. In this mode, the LINFlexD clock is stopped. Consequently, the LINFlexD will not update the status bits, but software can still access the LINFlexD registers.

To enter this mode, the software must set LINCR1[SLEEP].

LINFlexD can be awakened (exit Sleep mode) in one of two ways:

- The software clears LINCR1[SLEEP]
- Automatic wake-up is enabled (LINCR1[AWUM] is set) and LINFlexD detects LIN bus activity (that is, if a wakeup pulse of 150 µs is detected on the LIN bus)

On LIN bus activity detection, hardware automatically performs the wake-up sequence by clearing LINCR1[SLEEP] if LINCR1[AWUM] is set. To exit from Sleep mode if LINCR1[AWUM] is cleared, the software must clear LINCR1[SLEEP] when a wake-up event occurs.

## 27.7 LIN modes

### 27.7.1 Master mode

In Master mode, the software uses the message buffer to handle the LIN messages.

Master mode is selected when LINCR1[MME] is set.

#### 27.7.1.1 LIN header transmission

According to the LIN protocol, any communication on the LIN bus is triggered by the master sending a header. The header is transmitted by the master task while the data is transmitted by the slave task of a node.

To transmit a header with LINFlexD the application must set up the identifier, the data field length and configure the message (direction and checksum type) in the BIDR register before requesting the header transmission by setting LINCR2[HTRQ].

#### 27.7.1.2 Data transmission (transceiver as publisher)

When the master node is publisher of the data corresponding to the identifier sent in the header, then the slave task of the master has to send the data in the Response part of the LIN frame. Therefore, the software must provide the data to LINFlexD before requesting the header transmission. The software stores the data in the message buffer BDR. According to the data field length LINFlexD transmits the data and the checksum. The software uses the BIDR[CCS] bit to configure the checksum type (classic or enhanced) for each message.

The direction of the message buffer is controlled by the BIDR[DIR] bit. When the software sets this bit the response is sent by LINFlexD (publisher). Clearing this bit configures the message buffer as subscriber.

#### 27.7.1.3 Data reception (transceiver as subscriber)

To receive data from a slave node, the master sends a header with the corresponding identifier. LINFlexD stores the data received from the slave in the message buffer and stores the message status in the LINSR.

#### 27.7.1.4 Error detection and handling

LINFlexD is able to detect and handle LIN communication errors. A code stored in the LIN error status register (LINESR) signals the errors to the software.

Table 27-1 lists the errors detected in Master mode and the LINFlexD controller's response to these errors.

**Table 27-1. Errors in Master mode**

| Error | Description | LINFlexD response to error |
|---|---|---|
| Bit error | During transmission, the value read back from the bus differs from the transmitted value | • Stops the transmission of the frame after the corrupted bit<br>• Generates an interrupt if LINIER[BEIE] is set<br>• Returns to idle state |
| Framing error | A dominant state has been sampled on the stop bit of the currently received character (sync field, identifier, or data field) | If encountered during reception:<br>• Discards the current frame<br>• Generates an interrupt if LINIER[FEIE] is set<br>• Returns immediately to idle state |
| Checksum error | The computed checksum does not match the received checksum | If encountered during reception:<br>• Discards the current frame<br>• Generates an interrupt if LINIER[CEIE] is set<br>• Returns to idle state |
| Response and frame timeout | Refer to Section 27.12.1, 8-bit timeout counter, for more details | |

### 27.7.2 Slave mode

In Slave mode the software uses the message buffer to handle the LIN messages.

Slave mode is selected when the LINCR1[MME] is cleared.

### 27.7.2.1 Data transmission (transceiver as publisher)

When LINFlexD receives the identifier, an RX interrupt is generated. The software must:

- Read the received ID in the BIDR register
- Fill the BDR registers
- Specify the data field length using the BIDR[DFL] field
- Trigger the data transmission by setting LINCR2[DTRQ]

One or several identifier filters can be configured for transmission by setting the DIR bits in the corresponding IFCR registers and activated by setting one or several bits in the IFER register.

When at least one identifier filter is configured in transmission and activated, and if the received ID matches the filter, a specific TX interrupt is generated.

Typically, the software has to copy the data from RAM locations to the BDRL and BDRM registers. To copy the data to the right location, the software has to identify the data by means of the identifier. To avoid this and to ease the access to the RAM locations, the LINFlexD controller provides a Filter Match Index. This index value is the number of the filter which matched the received identifier.

The software can use the index in the IFMI register to directly access the pointer which points to the right data array in the RAM area and copy this data to the BDRL and BDRM registers (see Figure 27-10).

Using a filter avoids the software having to configure the direction, the data field length and the checksum type in the BDIR register. The software fills the BDRL and BDRM registers and triggers the data transmission by setting LINCR2[DTRQ].

If LINFlexD cannot provide enough TX identifier filters to handle all identifiers the software has to transmit data for, then a filter can be configured in mask mode (refer to Section 27.7.3, Slave mode with identifier filtering) in order to manage several identifiers with one filter only.

### 27.7.2.2 Data reception (transceiver as subscriber)

When LINFlexD receives the identifier, an RX interrupt is generated. The software must:

- Read the received ID in the BIDR register
- Specify the data field length using the BIDR[DFL] field before the reception of the stop bit of the first byte of data field

When the checksum reception is completed, an RX interrupt is generated to allow the software to read the received data in the BDR registers.

One or several identifier filters can be configured for reception by clearing the DIR bit in the corresponding IFCR registers and activated by clearing one or several bits in the IFER register.

When at least one identifier filter is configured in reception and activated, and if the received ID matches the filter, an RX interrupt is generated after the checksum reception only.

Typically, the software has to copy the data from the BDRL and BDRM registers to RAM locations. To copy the data to the right location, the software has to identify the data by means of the identifier. To avoid

this and to ease the access to the RAM locations, the LINFlexD controller provides a Filter Match Index. This index value is the number of the filter which matched the received identifier.

The software can use the index in the IFMI register to directly access the pointer which points to the right data array in the RAM area and copy this data from the BDRL and BDRM registers to the RAM (see Figure 27-10).

Using a filter avoids the software reading the ID value in the BIDR register, and configuring the direction, the data field length and the checksum type in the BIDR register.

If LINFlexD cannot provide enough RX identifier filters to handle all identifiers the software has to receive the data for, then a filter can be configured in mask mode (refer to Section 27.7.3, Slave mode with identifier filtering) in order to manage several identifiers with one filter only.

#### 27.7.2.3 Data discard

When LINFlexD receives the identifier, an RX interrupt is generated. If the received identifier does not concern the node, the software must set LINCR2[DDRQ]. LINFlexD returns to idle state.

#### 27.7.2.4 Error detection and handling

Table 27-2 lists the errors detected in Slave mode and the LINFlexD controller's response to these errors.

**Table 27-2. Errors in Slave mode**

| Error | Description | LINFlexD response to error |
|---|---|---|
| Bit error | During transmission, the value read back from the bus differs from the transmitted value | • Stops the transmission of the frame after the corrupted bit<br>• Generates an interrupt if LINIER[BEIE] is set<br>• Returns to idle state |
| Framing error | A dominant state has been sampled on the stop bit of the currently received character (sync field, identifier, or data field) | If encountered during reception:<br>• Discards the current frame<br>• Generates an interrupt if LINIER[FEIE] is set<br>• Returns immediately to idle state |
| Checksum error | The computed checksum does not match the received checksum | If encountered during reception:<br>• Discards the received frame<br>• Generates an interrupt if LINIER[CEIE] is set<br>• Returns to idle state |
| Header error | An error occurred during header reception (break delimiter error, inconsistent sync field, header timeout) | If encountered during header reception, a break field error, an inconsistent sync field, or a timeout:<br>• Discards the header<br>• Generates an interrupt if LINIER[HEIE] is set<br>• Returns to idle state |

#### 27.7.2.5 Valid header

A received header is considered as valid when it has been received correctly according to the LIN protocol.

If a valid break field and break delimiter come before the end of the current header, or at any time during a data field, the current header or data is discarded and the state machine synchronizes on this new break.

#### 27.7.2.6 Valid message

A received or transmitted message is considered as valid when the data has been received or transmitted without error according to the LIN protocol.

#### 27.7.2.7 Overrun

After the message buffer is full, the next valid message reception causes an overrun and a message is lost. The LINFlexD controller sets LINSR[BOF] to signal the overrun condition. Which message is lost depends on the configuration of the RX message buffer:

- If the buffer lock function is disabled (LINCR1[RBLM] cleared), the last message stored in the buffer is overwritten by the new incoming message. In this case, the latest message is always available to the software.
- If the buffer lock function is enabled (LINCR1[RBLM] set), the most recent message is discarded and the previous message is available in the buffer.

### 27.7.3 Slave mode with identifier filtering

In the LIN protocol, the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently a transmitter broadcasts its message to all receivers. When a slave node receives a header, it decides - depending on the identifier value - whether the software needs to receive or send a response. If the message does not target the node, it must be discarded without software intervention.

To fulfill this requirement, the LINFlexD controller provides configurable filters in order to request software intervention only if needed. This hardware filtering saves CPU resources which would otherwise be needed by software for filtering.

The filtering is accomplished through the use of IFCR registers. These registers have the names IFCR0 through IFCR. This section also uses the nomenclature $IFCR_{2n}$ and $IFCR_{2n+1}$; in this nomenclature, n is an integer, and the corresponding IFCR register is calculated using the formula in the subscript.

#### 27.7.3.1 Filter submodes

Usually each of the eight IFCR registers is used to filter one dedicated identifier, but this means that the LINFlexD controller could filter a maximum of eight identifiers. In order to be able to handle more identifiers, the filters can be configured to operate as masks.

Table 27-3 describes the two available filter submodes.

**Table 27-3. Filter submodes**

| Submode | Description |
|---|---|
| Identifier list | Both filter registers are used as identifier registers. All bits of the incoming identifier must match the bits specified in the filter register. This is the default submode for the LINFlexD controller. |
| Mask | The identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”. |

The bit mapping and register organization in these two submodes is shown in Figure 27-9.

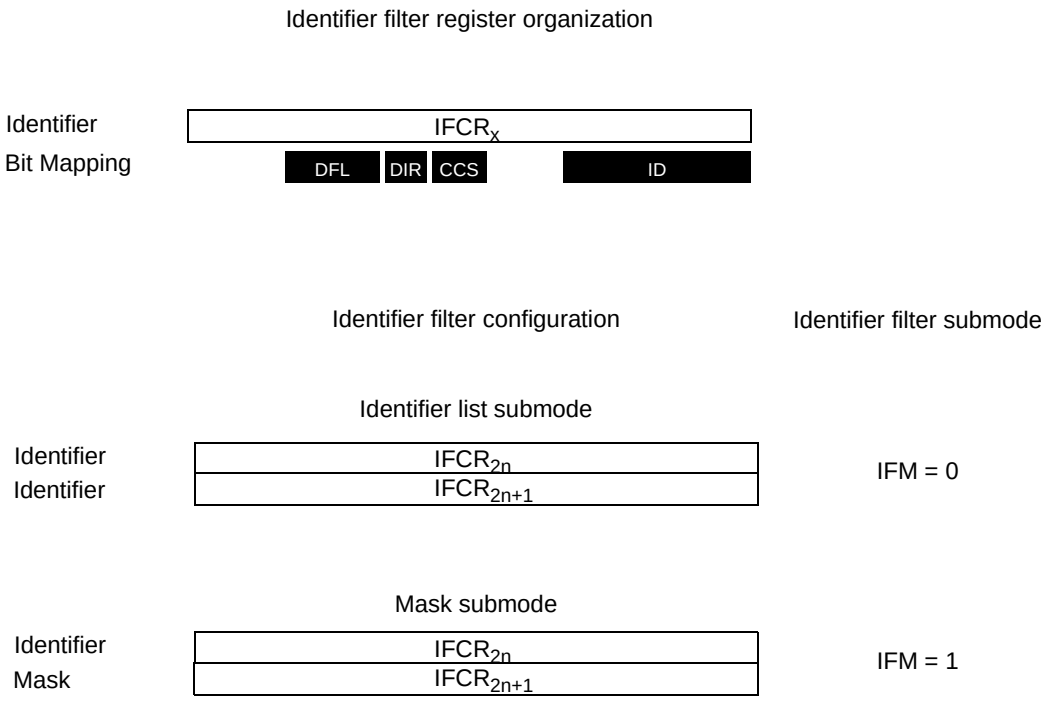



**Figure 27-9. Filter configuration - register organization**

### 27.7.3.2 Identifier filter submode configuration

The identifier filters are configured in the IFCR registers. To configure an identifier filter the filter must first be deactivated by clearing the corresponding bit in the IFER[FACT] field. The submode (identifier list or mask) for the corresponding IFCR register is configured by the IFMR[IFM] field. For each filter, the IFCR register is used to configure:

- The ID or mask
- The direction (TX or RX)
- The data field length
- The checksum type

If no filter is active, an RX interrupt is generated on any received identifier event.

If at least one active filter is configured as TX, all received identifiers matching this filter generate a TX interrupt.

If at least one active filter is configured as RX, all received identifiers matching this filter generate an RX interrupt.

If no active filter is configured as RX, all received identifiers not matching TX filter(s) generate an RX interrupt.

Further details are provided in Table 27-4 and Figure 27-10.

**Table 27-4. Filter to interrupt vector correlation**

| Number of active filters | Number of active filters configured as TX | Number of active filters configured as RX | Interrupt vector |
|---|---|---|---|
| 0 | 0 | 0 | - RX interrupt on all IDs |
| a<br>(a > 0) | a | 0 | - TX interrupt on IDs matching the filters,<br>- RX interrupt on all other IDs if BF bit is set, no RX interrupt if BF bit is reset |
| n<br>(n = a + b) | a<br>(a > 0) | b<br>(b > 0) | - TX interrupt on IDs matching the TX filters,<br>- RX interrupt on IDs matching the RX filters,<br>- all other IDs discarded (no interrupt) |
| b<br>(b > 0) | 0 | b | - RX interrupt on IDs matching the filters,<br>- TX interrupt on all other IDs if BF bit is set, no TX interrupt if BF bit is reset |

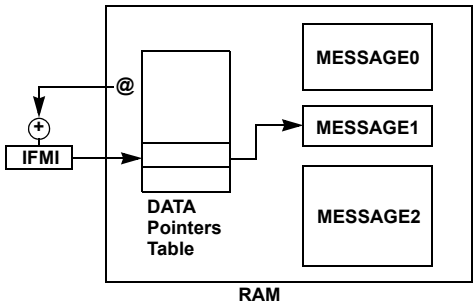



**Figure 27-10. Identifier match index**

## 27.7.4 Slave mode with automatic resynchronization

Automatic resynchronization must be enabled in Slave mode if $f_{ipg_clock_lin}$ tolerance is greater than 1.5%. This feature compensates a  deviation up to 14%, as specified in the LIN standard.

This mode is similar to Slave mode as described in Section 27.7.2, Slave mode, with the addition of automatic resynchronization enabled by the LINCR1[LASE] bit. In this mode LINFlexD adjusts the fractional baud rate generator after each synch field reception.

### 27.7.4.1 Automatic resynchronization method

When automatic resynchronization is enabled, after each LIN break, the time duration between five falling edges on RDI is sampled on  as shown in Figure 27-11. Then the LFDIV value (and its associated LINIBRR and LINFBRR registers) are automatically updated at the end of the fifth falling edge. During LIN sync field measurement, the LINFlexD state machine is stopped and no data is transferred to the data register.

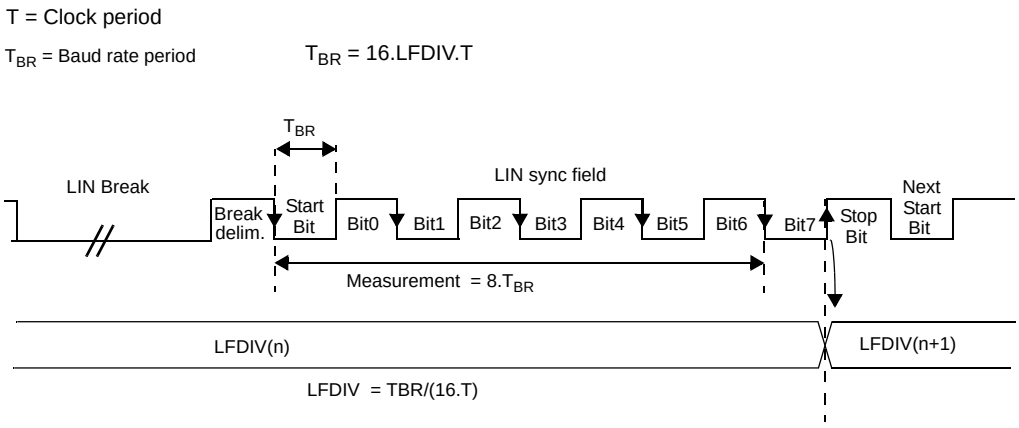



**Figure 27-11. LIN sync field measurement**

LFDIV is an unsigned fixed point number. The mantissa is coded on 20 bits in the LINIBRR register and the fraction is coded on 4 bits in the LINFBRR register.

If LINCR1[LASE] is set, LFDIV is automatically updated at the end of each LIN sync field.

Three registers are used internally to manage the auto-update of the LINFlexD divider (LFDIV):

- LFDIV_NOM (nominal value written by software at LINIBRR and LINFBRR addresses)
- LFDIV_MEAS (results of the Field Synch measurement)
- LFDIV (used to generate the local baud rate)

On transition to idle, break or break delimiter state due to any error or on reception of a complete frame, hardware reloads LFDIV with LFDIV_NOM.

### 27.7.4.2 Deviation error on the sync field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN sync field (relative to the master oscillator). Two checks are performed in parallel.

The first check is based on a measurement between the first falling edge and the last falling edge of the sync field:

- If D1 > 14.84%, LHE is set.
- If D1 < 14.06%, LHE is not set.
- If 14.06% < D1 < 14.84%, LHE can be either set or reset depending on the dephasing between the signal on LINFlexD_RX pin the $f_{ipg_clock_lin}$ clock.

The second check is based on a measurement of time between each falling edge of the sync field:

- If D2 > 18.75%, LHE is set.
- If D2 < 15.62%, LHE is not set.
- If 15.62% < D2 < 18.75%, LHE can be either set or reset depending on the dephasing between the signal on LINFlexD_RX pin the $f_{ipg_clock_lin}$ clock.

Note that the LINFlexD does not need to check if the next edge occurs slower than expected. This is covered by the check for deviation error on the full synch byte.

## 27.8 Test modes

The LINFlexD controller includes two test modes, Loop Back mode and Self Test mode. They can be selected by the LBKM and SFTM bits in the LINCR1 register. These bits must be configured while LINFlexD is in Initialization mode. After one of the two test modes has been selected, LINFlexD must be started in Normal mode.

### 27.8.1 Loop Back mode

LINFlexD can be put in Loop Back mode by setting LINCR1[LBKM]. In Loop Back mode, the LINFlexD treats its own transmitted messages as received messages. This is illustrated in Figure 27-12.

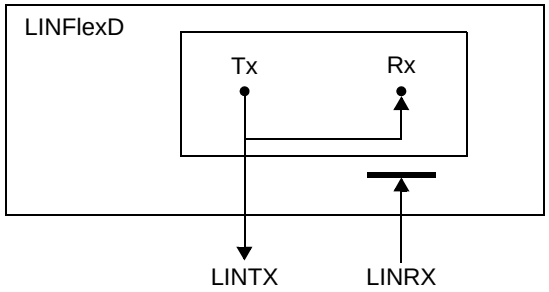



**Figure 27-12. LINFlexD in Loop Back mode**

This mode is provided for self-test functions. To be independent of external events, the LIN core ignores the LINRX signal. In this mode, the LINFlexD performs an internal feedback from its Tx output to its Rx

input. The actual value of the LINRX input pin is disregarded by the LINFlexD. The transmitted messages can be monitored on the LINTX pin.

### 27.8.2 Self Test mode

LINFlexD can be put in Self Test mode by setting LINCR1[LBKM] and LINCR1[SFTM]. This mode can be used for a “Hot Self Test”, meaning the LINFlexD can be tested as in Loop Back mode but without affecting a running LIN system connected to the LINTX and LINRX pins. In this mode, the LINRX pin is disconnected from the LINFlexD and the LINTX pin is held recessive. This is illustrated in Figure 27-13.

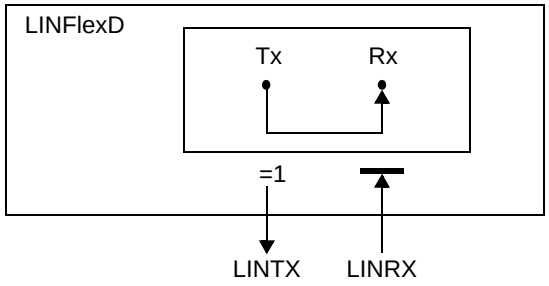



**Figure 27-13. LINFlexD in Self Test mode**

## 27.9 UART mode

The main features of UART mode are presented in Section 27.2.2, UART mode features.

### 27.9.1 Data frame structure

#### 27.9.1.1 8-bit data frame

The 8-bit UART data frame is shown in Figure 27-14. The 8th bit can be a data or a parity bit. Parity (even, odd, 0, or 1) can be selected by the UARTCR[PC] field. An even parity is set if the modulo-2 sum of the 7 data bits is 1. An odd parity is cleared in this case.

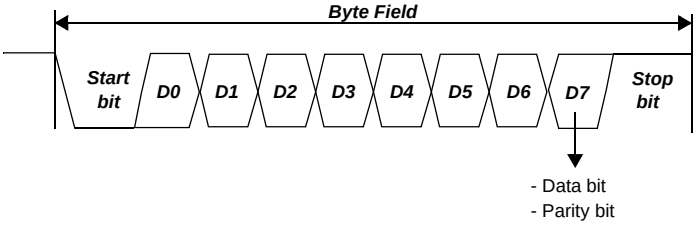



**Figure 27-14. UART mode 8-bit data frame**

#### 27.9.1.2 9-bit data frame

The 8-bit UART data frame is shown in Figure 27-15. The 9th bit is a parity bit. Parity (even, odd, 0, or 1) can be selected by the by the UARTCR[PC] field. An even parity is set if the modulo-2 sum of the 7 data

bits is 1. An odd parity is cleared in this case. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

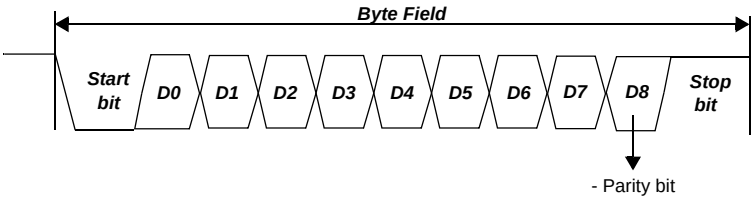



**Figure 27-15. UART mode 9-bit data frame**

### 27.9.1.3 16-bit data frame

The 16-bit UART data frame is shown in Figure 27-16. The 16th bit can be a data or a parity bit. Parity (even, odd, 0, or 1) can be selected by the UARTCR[PC] field. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

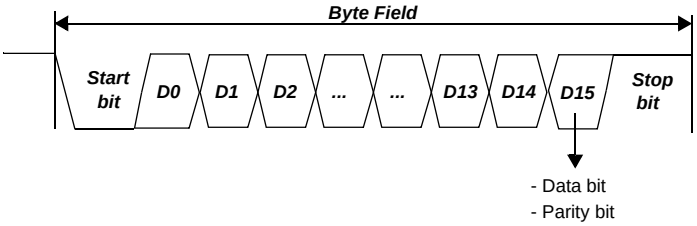



**Figure 27-16. UART mode 16-bit data frame**

### 27.9.1.4 17-bit data frame

The 17-bit UART data frame is shown in Figure 27-17. The 17th bit is the parity bit. Parity (even, odd, 0, or 1) can be selected by the UARTCR[PC] field. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

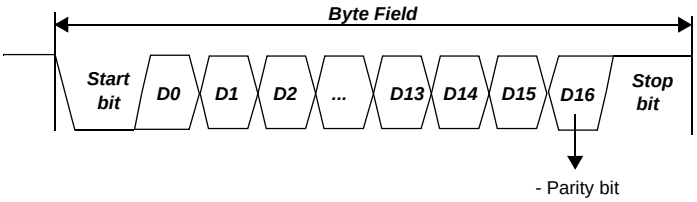



**Figure 27-17. UART mode 17-bit data frame**

## 27.9.2 Buffer

The 8-byte buffer is divided into two parts — one for receiver and one for transmitter — as shown in Table 27-5.

Table 27-5. UART buffer structure

| BDR | UART mode |
|---|---|
| 0 | Tx0 |
| 1 | Tx1 |
| 2 | Tx2 |
| 3 | Tx3 |
| 4 | Rx0 |
| 5 | Rx1 |
| 6 | Rx2 |
| 7 | Rx3 |

For 16-bit frames, the lower 8 bits will be written in BDR0 and the upper 8 bits will be written in BDR1.

### 27.9.3 UART transmitter

In order to start transmission in UART mode, the UARTCR[UART] and UARTCR[TXEN] bits must be set. Transmission starts when BDR0 (least significant data byte) is programmed. The number of bytes transmitted is equal to the value configured by the UARTCR[TDFLTFC] field (see Table 27-18).

The Transmit buffer size is as follows:

- 4 bytes when UARTCR[WL1] = 0
- 2 half-words when UARTCR[WL1] = 1

Therefore, the maximum transmission that can be triggered is 4 bytes (2 half-words). After the programmed number of bytes has been transmitted, the UARTSR[DTFTFF] flag is set. If the UARTCR[TXEN] field is cleared during a transmission, the current transmission is completed, but no further transmission can be invoked. The buffer can be configured in FIFO mode (mandatory when DMA Tx is enabled) by setting UARTCR[TFBM].

The access to the BDRL register is shown in Table 27-6.

Table 27-6. BDRL access in UART mode

| Access | Mode[1] | Word length[2] | IPS operation result |
|---|---|---|---|
| Write Byte0 | FIFO | Byte | OK |
| Write Byte1-2-3 | FIFO | Byte | IPS transfer error |
| Write Half-word0-1 | FIFO | Byte | IPS transfer error |
| Write Word | FIFO | Byte | IPS transfer error |
| Write Byte0-1-2-3 | FIFO | Half-word | IPS transfer error |
| Write Half-word0 | FIFO | Half-word | OK |
| Write Half-word1 | FIFO | Half-word | IPS transfer error |
| Write Word | FIFO | Half-word | IPS transfer error |

Table 27-6. BDRL access in UART mode (continued)

| Access | Mode[1] | Word length[2] | IPS operation result |
|---|---|---|---|
| Read Byte0-1-2-3 | FIFO | Byte/Half-word | IPS transfer error |
| Read Half-word0-1 | FIFO | Byte/Half-word | IPS transfer error |
| Read Word | FIFO | Byte/Half-word | IPS transfer error |
| Write Byte0-1-2-3 | BUFFER | Byte/Half-word | OK |
| Write Half-word0-1 | BUFFER | Byte/Half-word | OK |
| Write Word | BUFFER | Byte/Half-word | OK |
| Read Byte0-1-2-3 | BUFFER | Byte/Half-word | OK |
| Read Half-word0-1 | BUFFER | Byte/Half-word | OK |
| Read Word | BUFFER | Byte/Half-word | OK |

1 As specified by UARTCR[TFBM]

2 As specified by the WL1 and WL0 bits of the UARTCR register. In UART FIFO mode (UARTCR[TFBM] = 1),any read operation causes an IPS transfer error.

## 27.9.4 UART receiver

Reception of a data byte is started as soon as the software completes the following tasks in order:

1. Exits Initialization mode
2. Sets the UARTCR[RXEN] field
3. Detects the start bit

There is a dedicated data buffer for received data bytes. Its size is as follows:

- 4 bytes when UARTCR[WL1] = 0
- 2 half-words when UARTCR[WL1] = 1

After the programmed number (RDFL bits) of bytes has been received, the UARTSR[DRFRFE] field is set. If the UARTCR[RXEN] field is cleared during a reception, the current reception is completed, but no further reception can be invoked until UARTCR[RXEN] is set again.

The buffer can be configured in FIFO mode (required when DMA Rx is enabled) by setting UARTCR[RFBM].

The access to the BDRM register is shown in Table 27-7.

Table 27-7. BDRM access in UART mode

| Access | Mode[1] | Word length[2] | IPS operation result |
|---|---|---|---|
| Read Byte4 | FIFO | Byte | OK |
| Read Byte5-6-7 | FIFO | Byte | IPS transfer error |
| Read Half-word2-3 | FIFO | Byte | IPS transfer error |
| Read Word | FIFO | Byte | IPS transfer error |

**Table 27-7. BDRM access in UART mode (continued)**

| Access | Mode[1] | Word length[2] | IPS operation result |
|---|---|---|---|
| Read Byte4-5-6-7 | FIFO | Half-word | IPS transfer error |
| Read Half-word2 | FIFO | Half-word | OK |
| Read Half-word3 | FIFO | Half-word | IPS transfer error |
| Read Word | FIFO | Half-word | IPS transfer error |
| Write Byte4-5-6-7 | FIFO | Byte/Half-word | IPS transfer error |
| Write Half-word2-3 | FIFO | Byte/Half-word | IPS transfer error |
| Write Word | FIFO | Byte/Half-word | IPS transfer error |
| Read Byte4-5-6-7 | BUFFER | Byte/Half-word | OK |
| Read Half-word2-3 | BUFFER | Byte/Half-word | OK |
| Read Word | BUFFER | Byte/Half-word | OK |
| Write Byte4-5-6-7 | BUFFER | Byte/Half-word | IPS transfer error |
| Write Half-word2-3 | BUFFER | Byte/Half-word | IPS transfer error |
| Write Word | BUFFER | Byte/Half-word | IPS transfer error |

1 As specified by UARTCR[RFBM]

2 As specified by the WL1 and WL0 bits of the UARTCR register

Table 27-8 lists some common scenarios, controller responses, and suggestions when the LINFlexD controller is acting as a UART receiver.

**Table 27-8. UART receiver scenarios**

| Scenario | Responses and suggestions |
|---|---|
| The software does not know (in advance) how many bytes will be received. | Do not program UARTCR[RDFLRFC] in advance. When this field is zero (as it is after reset), reception occurs on a byte-by-byte basis. Therefore, the state machine will move to IDLE state after each byte is received. |
| UARTCR[RDFLRFC] is programmed for a certain number of bytes received, but the actual number of bytes received is smaller. | The reception will hang. In this case, the software must monitor the UARTSR[TO] field, and move to IDLE state by setting LINCR1[SLEEP]. |
| A STOP request arrives before the reception is completed. | The request is acknowledged only after the programmed number of data bytes are received. In other words, the STOP request is not serviced immediately. In this case, the software must monitor the UARTSR[TO] field and move the state machine to IDLE state as appropriate. The stop request will be serviced only after this is complete. |
| A parity error occurs during the reception of a byte. | The corresponding UARTSR[PE*n*] field is set. No interrupt is generated. |

Table 27-8. UART receiver scenarios (continued)

| Scenario | Responses and suggestions |
|---|---|
| A framing error occurs during the reception of a byte. | • UARTSR[FE] is set.<br>• If LINIER[FEIE] = 1, an interrupt is generated. This interrupt is helpful in identifying which byte has the framing error, since there is only one register bit for framing errors. |
| A new byte has been received, but the last received frame has not been read from the buffer (UARTSR[RMB] has not yet been cleared by the software) | • An overrun error will occur (UARTSR[BOF] will be set).<br>• One message will be lost (depending on the setting of LINCR[RBLM]).<br>• An interrupt is generated if LINIER[BOIE] is set. |

## 27.10 Memory map and register description

Table 27-9 shows the LINFlexD memory/register map for the LINFlex_0 module. Table 27-10 shows the LINFlexD memory/register map for LINFlex_1, LINFlex_2 and LINFlex_3 (these modules support master mode only and thus have no slave filter control registers). See the device memory map for the base addresses.

Table 27-9. LINFlexD memory map (LINFlex_0 only)

| Address offset | Register description | Location |
|---|---|---|
| 0x00 | LIN control register 1 (LINCR1) | on page 988 |
| 0x04 | LIN interrupt enable register (LINIER) | on page 991 |
| 0x08 | LIN status register (LINSR) | on page 992 |
| 0x0C | LIN error status register (LINESR) | on page 995 |
| 0x10 | UART mode control register (UARTCR) | on page 996 |
| 0x14 | UART mode status register (UARTSR) | on page 999 |
| 0x18 | LIN timeout control status register (LINTCSR) | on page 1001 |
| 0x1C | LIN output compare register (LINOCR) | on page 1002 |
| 0x20 | LIN timeout control register (LINTOCR) | on page 1003 |
| 0x24 | LIN fractional baud rate register (LINFBRR) | on page 1004 |
| 0x28 | LIN integer baud rate register (LINIBRR) | on page 1004 |
| 0x2C | LIN checksum field register (LINCFR) | on page 1005 |
| 0x30 | LIN control register 2 (LINCR2) | on page 1006 |
| 0x34 | Buffer identifier register (BIDR) | on page 1007 |
| 0x38 | Buffer data register least significant (BDRL) | on page 1008 |
| 0x3C | Buffer data register most significant (BDRM) | on page 1009 |
| 0x40 | Identifier filter enable register (IFER) | on page 1010 |
| 0x44 | Identifier filter match index (IFMI) | on page 1010 |
| 0x48 | Identifier filter mode register (IFMR) | on page 1011 |

**Table 27-9. LINFlexD memory map (LINFlex_0 only) (continued)**

| Address offset | Register description | Location |
|---|---|---|
| 0x4C–0x88 | Identifier filter control registers 0–15 (IFCR0–IFCR15) | on page 1012 |
| 0x8C | Global control register (GCR) | on page 1013 |
| 0x90 | UART preset timeout register (UARTPTO) | on page 1014 |
| 0x94 | UART current timeout register (UARTCTO) | on page 1015 |
| 0x98 | DMA Tx enable register (DMATXE) | on page 1016 |
| 0x9C | DMA Rx enable register (DMARXE) | on page 1016 |

**Table 27-10. LINFlexD memory map (LINFlex_1/LINFlex_2/LINFlex_3 only)**

| Address offset | Register description | Location |
|---|---|---|
| 0x00 | LIN control register 1 (LINCR1) | on page 988 |
| 0x04 | LIN interrupt enable register (LINIER) | on page 991 |
| 0x08 | LIN status register (LINSR) | on page 992 |
| 0x0C | LIN error status register (LINESR) | on page 995 |
| 0x10 | UART mode control register (UARTCR) | on page 996 |
| 0x14 | UART mode status register (UARTSR) | on page 999 |
| 0x18 | LIN timeout control status register (LINTCSR) | on page 1001 |
| 0x1C | LIN output compare register (LINOCR) | on page 1002 |
| 0x20 | LIN timeout control register (LINTOCR) | on page 1003 |
| 0x24 | LIN fractional baud rate register (LINFBRR) | on page 1004 |
| 0x28 | LIN integer baud rate register (LINIBRR) | on page 1004 |
| 0x2C | LIN checksum field register (LINCFR) | on page 1005 |
| 0x30 | LIN control register 2 (LINCR2) | on page 1006 |
| 0x34 | Buffer identifier register (BIDR) | on page 1007 |
| 0x38 | Buffer data register least significant (BDRL) | on page 1008 |
| 0x3C | Buffer data register most significant (BDRM) | on page 1009 |
| 0x40 | Identifier filter enable register (IFER) | on page 1010 |
| 0x44 | Identifier filter match index (IFMI) | on page 1010 |
| 0x48 | Identifier filter mode register (IFMR) | on page 1011 |
| 0x4C | Global control register (GCR) | on page 1013 |
| 0x50 | UART preset timeout register (UARTPTO) | on page 1014 |
| 0x54 | UART current timeout register (UARTCTO) | on page 1015 |
| 0x58 | DMA Tx enable register (DMATXE) | on page 1016 |
| 0x5C | DMA Rx enable register (DMARXE) | on page 1016 |

## 27.10.1 LIN control register 1 (LINCR1)

Offset:0x00 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CCD[1] | CFD[1] | LASE[1] | AWUM[1] | MBL[1] | | | | BF[1] | SFTM[1] | LBKM[1] | MME[1] | SBDT[1] | RBLM[1] | SLEEP | INIT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0/1[2] | 0 | 0 | 1 | 0 |

1 These fields are writable only in Initialization mode (LINCR1[INIT] = 1).

2 Resets to 0 in Slave mode and to 1 in Master mode

**Figure 27-18. LIN control register 1 (LINCR1)**

**Table 27-11. LINCR1 field descriptions**

| Field | Description |
|---|---|
| CCD | Checksum Calculation disable<br>This bit is used to disable the checksum calculation (see Table 27-12).<br>0: Checksum calculation is done by hardware. When this bit is reset the LINCFR register is read-only.<br>1: Checksum calculation is disabled. When this bit is set the LINCFR register is read/write. User can program this register to send a software calculated CRC (provided CFD is reset).<br>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode. |
| CFD | Checksum field disable<br>This bit is used to disable the checksum field transmission (see Table 27-12).<br>0: Checksum field is sent after the required number of data bytes is sent.<br>1: No checksum field is sent.<br>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode. |
| LASE | LIN Slave Automatic Resynchronization Enable<br>0: Automatic resynchronization disable<br>1: Automatic resynchronization enable<br>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode. |
| AWUM | Automatic Wake-Up Mode<br>This bit controls the behavior of the LINFlexD hardware during Sleep mode.<br>0: The Sleep mode is exited on software request by clearing the SLEEP bit of the LINCR register.<br>1: The Sleep mode is exited automatically by hardware on RX dominant state detection. The SLEEP bit of the LINCR register is cleared by hardware whenever WUF bit in LINSR is set.<br>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode. |
| MBL | LIN Master Break Length<br>These bits indicate the Break length in Master mode (see Table 27-13).<br>Note: These bits can be written in Initialization mode only. They are read-only in Normal or Sleep mode. |

**Table 27-11. LINCR1 field descriptions (continued)**

| Field | Description |
|---|---|
| BF | Bypass filter<br>0: No interrupt if ID does not match any filter<br>1: An RX interrupt is generated on ID not matching any filter<br>Notes:<br>• If no filter is activated, this bit is reserved.<br>• This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode. |
| SFTM | Self Test Mode<br>This bit controls the Self Test mode. For more details please refer to Section 27.8.2, Self Test mode.<br>0: Self Test mode disable<br>1: Self Test mode enable<br>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode. |
| LBKM | Loop Back Mode<br>This bit controls the Loop Back mode. For more details please refer to Section 27.8.1, Loop Back mode.<br>0: Loop Back mode disable<br>1: Loop Back mode enable<br>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode |
| MME | Master Mode Enable<br>0: Slave mode enable<br>1: Master mode enable<br>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode. |
| SBDT | Slave Mode Break Detection Threshold<br>0: 11-bit break<br>1: 10-bit break<br>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode. |
| RBLM | Receive Buffer Locked Mode<br>0: Receive Buffer not locked on overrun. Once the Slave Receive Buffer is full the next incoming message overwrites the previous one.<br>1: Receive Buffer locked against overrun. Once the Receive Buffer is full the next incoming message is discarded.<br>Note: This bit can be written in Initialization mode only. It is read-only in Normal or Sleep mode. |
| SLEEP | Sleep Mode Request<br>This bit is set by software to request LINFlexD to enter Sleep mode.<br>This bit is cleared by software to exit Sleep mode or by hardware if the AWUM bit in LINCR1 and the WUF bit in LINSR are set (see Table 27-14). |
| INIT | Initialization Request<br>The software sets this bit to switch hardware into Initialization mode. If the SLEEP bit is reset, LINFlexD enters Normal mode when clearing the INIT bit (see Table 27-14). |

**Table 27-12. Checksum bits configuration**

| CFD | CCD | LINCFR | Checksum sent |
|---|---|---|---|
| 1 | 1 | Read/Write | None |
| 1 | 0 | Read-only | None |
| 0 | 1 | Read/Write | Programmed in LINCFR by bits CF[0:7] |

**Table 27-12. Checksum bits configuration**

| CFD | CCD | LINCFR | Checksum sent |
|---|---|---|---|
| 0 | 0 | Read-only | Hardware calculated |

**Table 27-13. LIN master break length selection**

| MBL | Length |
|---|---|
| 0000 | 10-bit |
| 0001 | 11-bit |
| 0010 | 12-bit |
| 0011 | 13-bit |
| 0100 | 14-bit |
| 0101 | 15-bit |
| 0110 | 16-bit |
| 0111 | 17-bit |
| 1000 | 18-bit |
| 1001 | 19-bit |
| 1010 | 20-bit |
| 1011 | 21-bit |
| 1100 | 22-bit |
| 1101 | 23-bit |
| 1110 | 36-bit |
| 1111 | 50-bit |

**Table 27-14. Operating mode selection**

| SLEEP | INIT | Operating mode |
|---|---|---|
| 1 | 0 | Sleep (reset value) |
| x | 1 | Initialization |
| 0 | 0 | Normal |

## 27.10.2 LIN interrupt enable register (LINIER)

Offset: 0x04 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SZIE | OCIE | BEIE | CEIE | HEIE | 0 | 0 | FEIE | BOIE | LSIE | WUIE | DBFIE | DBEIETOIE | DRIE | DTIE | HRIE |
| W | w1c | w1c | w1c | w1c | w1c | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-19. LIN interrupt enable register (LINIER)**

**Table 27-15. LINIER field descriptions**

| Field | Description |
|---|---|
| SZIE | Stuck at Zero Interrupt Enable<br>0: No interrupt when SZF bit in LINESR or UARTSR is set<br>1: Interrupt generated when SZF bit in LINESR or UARTSR is set |
| OCIE | Output Compare Interrupt Enable<br>0: No interrupt when OCF bit in LINESR or UARTSR is set<br>1: Interrupt generated when OCF bit in LINESR or UARTSR is set |
| BEIE | Bit Error Interrupt Enable<br>0: No interrupt when BEF bit in LINESR is set<br>1: Interrupt generated when BEF bit in LINESR is set |
| CEIE | Checksum Error Interrupt Enable<br>0: No interrupt on Checksum error<br>1: Interrupt generated when checksum error flag (CEF) is set in LINESR |
| HEIE | Header Error Interrupt Enable<br>0: No interrupt on Break Delimiter error, Synch Field error, ID field error<br>1: Interrupt generated on Break Delimiter error, Synch Field error, ID field error |
| FEIE | Framing Error Interrupt Enable<br>0: No interrupt on Framing error<br>1: Interrupt generated on Framing error |
| BOIE | Buffer Overrun Interrupt Enable<br>0: No interrupt on Buffer overrun<br>1: Interrupt generated on Buffer overrun |

**Table 27-15. LINIER field descriptions (continued)**

| Field | Description |
|---|---|
| LSIE | LIN State Interrupt Enable<br>0: No interrupt on LIN state change<br>1: Interrupt generated on LIN state change<br>This interrupt can be used for debugging purposes. It has no status flag but is reset when writing '1111' into the LIN state bits in the LINSR register. |
| WUIE | Wake-up Interrupt Enable<br>0: No interrupt when WUF bit in LINSR or UARTSR is set<br>1: Interrupt generated when WUF bit in LINSR or UARTSR is set |
| DBFIE | Data Buffer Full Interrupt Enable<br>0: No interrupt when buffer data register is full<br>1: Interrupt generated when data buffer register is full |
| DBEIETOIE | Data Buffer Empty Interrupt Enable / Timeout Interrupt Enable<br>0: No interrupt when buffer data register is empty<br>1: Interrupt generated when data buffer register is empty<br>**Note:** An interrupt is generated if this bit is set and one of the following is true:<br>LINFlexD is in LIN mode and LINSR[DBEF] is set<br>LINFlexD is in UART mode and UARTSR[TO] is set |
| DRIE | Data Reception Complete Interrupt Enable<br>0: No interrupt when data reception is completed<br>1: Interrupt generated when data received flag (DRF) in LINSR or UARTSR is set |
| DTIE | Data Transmitted Interrupt Enable<br>0: No interrupt when data transmission is completed<br>1: Interrupt generated when data transmitted flag (DTF) is set in LINSR or UARTSR register |
| HRIE | Header Received Interrupt Enable<br>0: No interrupt when a valid LIN header has been received<br>1: Interrupt generated when a valid LIN header has been received, that is, HRF bit in LINSR register is set |

## 27.10.3 LIN status register (LINSR)

Offset: 0x08 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LINS | | | | 0 | 0 | RMB | 0 | RBSY | RPS | WUF | DBFF | DBEF | DRF | DTF | HRF |
| W | w1c | | | | | | w1c | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-20. LIN status register (LINSR)**

**Table 27-16. LINSR field descriptions**

| Field | Description |
|---|---|
| LINS | LIN state<br>LIN mode states description<br>**0000: Sleep mode**<br>LINFlexD is in Sleep mode to save power consumption.<br>**0001: Initialization mode**<br>LINFlexD is in Initialization mode.<br>**0010: Idle**<br>This state is entered on several events:<br>• SLEEP bit and INIT in LINCR1 register have been cleared by software,<br>• A falling edge has been received on RX pin and AWUM bit is set,<br>• The previous frame reception or transmission has been completed or aborted.<br>**0011: Break**<br>In Slave mode, a falling edge followed by a dominant state has been detected. Receiving Break.<br>Note: In Slave mode, in case of error new LIN state can be either Idle or Break depending on last bit state. If last bit is dominant new LIN state is Break, otherwise Idle.<br>In Master mode, Break transmission ongoing.<br>**0100: Break Delimiter**<br>In Slave mode, a valid Break has been detected. Refer to LINCR1 register for break length configuration (10-bit or 11-bit). Waiting for a rising edge.<br>In Master mode, Break transmission has been completed. Break Delimiter transmission is ongoing.<br>**0101: Synch Field**<br>In Slave mode, a valid Break Delimiter has been detected (recessive state for at least one bit time). Receiving Synch Field.<br>In Master mode, Synch Field transmission is ongoing.<br>**0110: Identifier Field**<br>In Slave mode, a valid Synch Field has been received. Receiving ID Field.<br>In Master mode, identifier transmission is ongoing.<br>**0111: Header reception/transmission completed**<br>In Slave mode, a valid header has been received and identifier field is available in the BIDR register.<br>In Master mode, header transmission is completed.<br>**1000: Data reception/transmission**<br>Response reception/transmission is ongoing.<br>**1001: Checksum**<br>Data reception/transmission completed. Checksum reception/transmission ongoing.<br>In UART mode, only the following states are flagged by the LIN state bits:<br>• Init<br>• Sleep<br>• Idle<br>• Data transmission/reception |
| RMB | Release Message Buffer<br>0: Buffer is free<br>1: Buffer ready to be read by software. This bit must be cleared by software after reading data received in the buffer.<br>This bit is cleared by hardware in Initialization mode. |
| RBSY | Receiver Busy Flag<br>0: Receiver is Idle<br>1: Reception ongoing<br>Note: In Slave mode, after header reception, if DIR bit in BIDR is reset and reception starts then this bit is set. In this case, user cannot set DTRQ bit in LINCR2. |
| RPS | LIN receive pin state<br>This bit reflects the current status of LINRX pin for diagnostic purposes. |

**Table 27-16. LINSR field descriptions (continued)**

| Field | Description |
|---|---|
| WUF | Wake-up Flag<br>This bit is set by hardware and indicates to the software that LINFlexD has detected a falling edge on the LINRX pin when<br>• slave is in Sleep mode,<br>• master is in Sleep mode or idle state.<br>This bit must be cleared by software. It is reset by hardware in Initialization mode. An interrupt is generated if WUIE bit in LINIER is set. |
| DBFF | Data Buffer Full Flag<br>This bit is set by hardware and indicates the buffer is full. It is set only when receiving extended frames (DFL > 7).<br>This bit must be cleared by software.<br>It is reset by hardware in Initialization mode. |
| DBEF | Data Buffer Empty Flag<br>This bit is set by hardware and indicates the buffer is empty. It is set only when transmitting extended frames (DFL > 7).<br>This bit must be cleared by software, once buffer has been filled again, in order to start transmission.<br>This bit is reset by hardware in Initialization mode. |
| DRF | Data Reception Completed Flag<br>This bit is set by hardware and indicates the data reception is completed.<br>This bit must be cleared by software.<br>It is reset by hardware in Initialization mode.<br>Note: This flag is not set in case of bit error or framing error. |
| DTF | Data Transmission Completed Flag<br>This bit is set by hardware and indicates the data transmission is completed.<br>This bit must be cleared by software.<br>It is reset by hardware in Initialization mode.<br>Note: This flag is not set in case of bit error if IOBE bit is reset. |
| HRF | Header Reception Flag<br>This bit is set by hardware and indicates a valid header reception is completed.<br>This bit must be cleared by software.<br>This bit is reset by hardware in Initialization mode and at end of completed or aborted frame.<br>Note: If filters are enabled, this bit is set only when identifier software filtering is required, that is to say:<br>• all filters are inactive and BF bit in LINCR1 is set<br>• no match in any filter and BF bit in LINCR1 is set<br>• TX filter match |

## 27.10.4 LIN error status register (LINESR)

Offset: 0x0C Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SZF | OCF | BEF | CEF | SFEF | BDEF | IDPEF | FEF | BOF | 0 | 0 | 0 | 0 | 0 | 0 | NF |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | | | | | | | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-21. LIN error status register (LINESR)**

**Table 27-17. LINESR field descriptions**

| Field | Description |
|---|---|
| SZF | Stuck at zero Flag<br>This bit is set by hardware when the bus is dominant for more than a 100-bit time. It is cleared by software. |
| OCF | Output Compare Flag<br>0: No output compare event occurred<br>1: The content of the counter has matched the content of OC1[0:7] or OC2[0:7] in LINOCR. If this bit is set and IOT bit in LINTCSR is set, LINFlexD moves to Idle state.<br>If LTOM bit in LINTCSR register is set then OCF is reset by hardware in Initialization mode. If LTOM bit is reset, then OCF maintains its status whatever the mode is. |
| BEF | Bit Error Flag<br>This bit is set by hardware and indicates to the software that LINFlexD has detected a bit error. This error can occur during response field transmission (Slave and Master modes) or during header transmission (in Master mode).<br>This bit is cleared by software. |
| CEF | Checksum error Flag<br>This bit is set by hardware and indicates that the received checksum does not match the hardware calculated checksum.<br>This bit is cleared by software.<br>Note: This bit is never set if CCD or CFD bit in LINCR1 register is set. |
| SFEF | Synch Field Error Flag<br>This bit is set by hardware and indicates that a Synch Field error occurred (inconsistent Synch Field). |
| BDEF | Break Delimiter Error Flag<br>This bit is set by hardware and indicates that the received Break Delimiter is too short (less than one bit time). |

**Table 27-17. LINESR field descriptions (continued)**

| Field | Description |
|---|---|
| IDPEF | Identifier Parity Error Flag<br>This bit is set by hardware and indicates that a Identifier Parity error occurred.<br>Note: Header interrupt is triggered when SFEF or BDEF or IDPEF bit is set and HEIE bit in LINIER is set. |
| FEF | Framing Error Flag<br>This bit is set by hardware and indicates to the software that LINFlexD has detected a framing error (invalid stop bit). This error can occur during reception of any data in the response field (Master or Slave mode) or during reception of Synch Field or Identifier Field in Slave mode. |
| BOF | Buffer Overrun Flag<br>This bit is set by hardware when a new data byte is received and the buffer full flag is not cleared. If RBLM in LINCR1 is set then the new byte received is discarded. If RBLM is reset then the new byte overwrites the buffer. It can be cleared by software. |
| NF | Noise Flag<br>This bit is set by hardware when noise is detected on a received character. This bit is cleared by software. |

## 27.10.5 UART mode control register (UARTCR)

Offset: 0x10 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TDFLTFC[1] | | | | RDFLRFC[1] | | | | | | | | | | | |
| | | | | | | | RFBM | TFBM[2] | WL[1][2] | PC1[2] | RXEN | TXEN | PC0[2] | PCE[2] | WL[0][2] | UART[2] |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1 These fields are read/write in UART buffer mode and read-only in other modes.
2 These fields are writable only in Initialization mode (LINCR1[INIT] = 1).

**Figure 27-22. UART mode control register (UARTCR)**

**Table 27-18. UARTCR field descriptions**

| Field | Description |
|---|---|
| TDFLTFC | Transmitter data field length / Tx FIFO counter<br>This field has one of two functions depending on the mode of operation as follows:<br>• When LINFlexD is in UART buffer mode (TFBM = 0), TDFLTFC defines the number of bytes to be transmitted. The field is read/write in this configuration. The first bit is reserved and not implemented.<br>The permissible values are as follows (with X representing the unimplemented first bit):<br>0bX00: 1 byte<br>0bX01: 2 bytes<br>0bX10: 3 bytes<br>0bX11: 4 bytes<br>When the UART data length is configured as half-word (WL = 0b10 or 0b11), the only valid values for TDFLTFC are 0b001 and 0b011.<br>• When LINFlexD is in UART FIFO mode (TFBM = 1), TDFLTFC contains the number of entries (bytes) of the Tx FIFO. The field is read-only in this configuration.<br>The permissible values are as follows:<br>0b000: Empty<br>0b001: 1 byte<br>0b010: 2 bytes<br>0b011: 3 bytes<br>0b100: 4 bytes<br>All other values are reserved.<br>This field is meaningful and can be programmed only when the UART bit is set. |
| RDFLRFC | Receiver data field length / Rx FIFO counter<br>This field has one of two functions depending on the mode of operation as follows:<br>• When LINFlexD is in UART buffer mode (RFBM = 0), RDFLRFC defines the number of bytes to be received. The field is read/write in this configuration. The first bit is reserved and not implemented.<br>The permissible values are as follows (with X representing the unimplemented first bit):<br>0bX00: 1 byte<br>0bX01: 2 bytes<br>0bX10: 3 bytes<br>0bX11: 4 bytes<br>When the UART data length is configured as half-word (WL = 0b10 or 0b11), the only valid values for RDFLRFC are 0b001 and 0b011.<br>• When LINFlexD is in UART FIFO mode (RFBM = 1), RDFLRFC contains the number of entries (bytes) of the Rx FIFO. The field is read-only in this configuration.<br>The permissible values are as follows:<br>0b000: Empty<br>0b001: 1 byte<br>0b010: 2 bytes<br>0b011: 3 bytes<br>0b100: 4 bytes<br>All other values are reserved.<br>This field is meaningful and can be programmed only when the UART bit is set. |
| RFBM | Rx FIFO/buffer mode<br>0 Rx buffer mode enabled<br>1 Rx FIFO mode enabled (mandatory in DMA Rx mode)<br><br>This field can be programmed in initialization mode only when the UART bit is set. |

**Table 27-18. UARTCR field descriptions (continued)**

| Field | Description |
|---|---|
| TFBM | Tx FIFO/buffer mode<br>0 Tx buffer mode enabled<br>1 Tx FIFO mode enabled (mandatory in DMA Tx mode)<br><br>This field can be programmed in initialization mode only when the UART bit is set. |
| RXEN | Receiver Enable<br>0: Receiver disabled<br>1: Receiver enabled<br><br>This field can be programmed only when the UART bit is set. |
| TXEN | Transmitter Enable<br>0: Transmitter disabled<br>1: Transmitter enabled<br><br>This field can be programmed only when the UART bit is set.<br>Note: Transmission starts when this bit is set and when writing DATA0 in the BDRL register. |
| PC | Parity control<br>00 Parity sent is even<br>01 Parity sent is odd<br>10 A logical 0 is always transmitted/checked as parity bit<br>11 A logical 1 is always transmitted/checked as parity bit<br><br>This field can be programmed in initialization mode only when the UART bit is set. |
| PCE | Parity Control Enable<br>0: Parity transmit/check disabled<br>1: Parity transmit/check enabled<br><br>This field can be programmed in Initialization mode only when the UART bit is set. |
| WL | Word length in UART mode<br>00 7 bits data + parity<br>01 8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1<br>10 15 bits data + parity<br>11 16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1<br><br>This field can be programmed in Initialization mode only when the UART bit is set. |
| UART | UART mode enable<br>0: LIN mode<br>1: UART mode<br><br>This field can be programmed in Initialization mode only. |

## 27.10.6 UART mode status register (UARTSR)

Offset: 0x14 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SZF | OCF | PE3 | PE2 | PE1 | PE0 | RMB | FEF | BOF | RPS | WUF | 0 | TO | DRFRFE | DTFTFF | NF |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-23. UART mode status register (UARTSR)**

**Table 27-19. UARTSR field descriptions**

| Field | Description |
|---|---|
| SZF | Stuck at zero Flag<br>This bit is set by hardware when the bus is dominant for more than a 100-bit time. It is cleared by software. |
| OCF | OCF Output Compare Flag<br>0: No output compare event occurred<br>1: The content of the counter has matched the content of OC1[0:7] or OC2[0:7] in LINOCR.<br>An interrupt is generated if the OCIE bit in LINIER register is set. |
| PE3 | Parity Error Flag Rx3<br>This bit indicates if there is a parity error in the corresponding received byte (Rx3). No interrupt is generated if this error occurs.<br>0: No parity error<br>1: Parity error |
| PE2 | Parity Error Flag Rx2<br>This bit indicates if there is a parity error in the corresponding received byte (Rx2). No interrupt is generated if this error occurs.<br>0: No parity error<br>1: Parity error |
| PE1 | Parity Error Flag Rx1<br>This bit indicates if there is a parity error in the corresponding received byte (Rx1). No interrupt is generated if this error occurs.<br>0: No parity error<br>1: Parity error |
| PE0 | Parity Error Flag Rx0<br>This bit indicates if there is a parity error in the corresponding received byte (Rx0). No interrupt is generated if this error occurs.<br>0: No parity error<br>1: Parity error |

**Table 27-19. UARTSR field descriptions (continued)**

| Field | Description |
|---|---|
| RMB | Release Message Buffer<br>0: Buffer is free<br>1: Buffer ready to be read by software. This bit must be cleared by software after reading data received in the buffer.<br>This bit is cleared by hardware in Initialization mode. |
| FEF | Framing Error Flag<br>This bit is set by hardware and indicates to the software that LINFlexD has detected a framing error (invalid stop bit). |
| BOF | FIFO/buffer overrun flag<br>This bit is set by hardware when a new data byte is received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode, this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message is discarded regardless of the value of LINCR1[RBLM].<br>If LINCR1[RBLM] = 1, the new byte received is discarded.<br>If LINCR1[RBLM] = 0, the new byte overwrites buffer.<br>This field can be cleared by writing a 1 to it. An interrupt is generated if LINIER[BOIE] is set. |
| RPS | LIN Receive Pin State<br>This bit reflects the current status of LINRX pin for diagnostic purposes. |
| WUF | Wake-up Flag<br>This bit is set by hardware and indicates to the software that LINFlexD has detected a falling edge on the LINRX pin in Sleep mode.<br>This bit must be cleared by software. It is reset by hardware in Initialization mode.<br>An interrupt i generated if WUIE bit in LINIER is set. |
| TO | Timeout<br>The LINFlexD controller sets this field when a UART timeout occurs — that is, when the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). This field should be cleared by software. The GCR[SR] field should be used to reset the receiver FSM to idle state in case of UART timeout for UART reception depending on the application both in buffer and FIFO mode.<br>An interrupt is generated when LINIER[DBEIETOIE] is set on the Error interrupt line in UART mode. |
| DRFRFE | Data reception completed flag / Rx FIFO empty flag<br>The LINFlexD controller sets this field as follows:<br>• In UART buffer mode (RFBM = 0), it indicates that the num ber of bytes programmed in RDFL has been received. This field should be cleared by software. An interrupt is generated if LINIER[DRIE] is set. This field is set in case of framing error, parity error, or overrun. This field reflects the same value as in LINESR when in Initialization mode and UART bit is set.<br>• In UART FIFO mode (RFBM = 1), it indicates that the Rx FIFO is empty. This field is a read-only field used internally by the DMA Rx interface. |
| DTFTFF | Data transmission completed flag / Tx FIFO full flag<br>The LINFlexD controller sets this field as follows:<br>• In UART buffer mode (TFBM = 0), it indicates that the data transmission is completed. This field should be cleared by software. An interrupt is generated if LINIER[DTIE] is set. This field reflects the same value as in LINESR when in Initialization mode and UART bit is set.<br>• In UART FIFO mode (TFBM = 1), it indicates that the Tx FIFO is full. This field is a read-only field used internally by the DMA Tx interface. |
| NF | Noise Flag<br>This bit is set by hardware when noise is detected on a received character. This bit is cleared by software. |

## 27.10.7 LIN timeout control status register (LINTCSR)

Offset: 0x18 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | LTOM | IOT | TOCE | CNT[0:7] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

[1] These fields are writable only in Initialization mode (LINCR1[INIT] = 1).

**Figure 27-24. LIN timeout control status register (LINTCSR)**

**Table 27-20. LINTCSR field descriptions**

| Name | Description |
|---|---|
| LTOM | LIN timeout mode<br>0: LIN timeout mode (header, response and frame timeout detection)<br>1: Output compare mode<br>This bit can be set/cleared in Initialization mode only. |
| IOT | Idle on Timeout<br>0: LIN state machine not reset to Idle on timeout event<br>1: LIN state machine reset to Idle on timeout event<br>This bit can be set/cleared in Initialization mode only. |
| TOCE | Timeout counter enable<br>0: Timeout counter disable. OCF bit in LINESR or UARTSR is not set on an output compare event.<br>1: Timeout counter enable. OCF bit is set if an output compare event occurs.<br>TOCE bit is configurable by software in Initialization mode. If LIN state is not Init and if timer is in LIN timeout mode, then hardware takes control of TOCE bit. |
| CNT | Counter Value<br>These bits indicate the LIN Timeout counter value. |

## 27.10.8 LIN output compare register (LINOCR)

Offset: 0x1C Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OC2[1] | | | | | | | | OC1[1] | | | | | | | |
| W | w1c[1] | | | | | | | | w1c[1] | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

1 If LINTCSR[LTOM] = 1, these fields are read-only.

**Figure 27-25. LIN output compare register (LINOCR)**

**Table 27-21. LINOCR field descriptions**

| Field | Description |
|---|---|
| OC2 | Output compare 2 value<br>These bits contain the value to be compared to the value of LINTCSR[CNT]. |
| OC1 | Output compare 1 value<br>These bits contain the value to be compared to the value of LINTCSR[CNT]. |

## 27.10.9 LIN timeout control register (LINTOCR)

Offset: 0x20 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | RTO | | | | 0 | HTO[3] | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0/1[1] | 0/1[2] | 1 | 1 | 0 | 0 |

1 Resets to 1 in Slave mode and to 0 in Master mode
2 Resets to 0 in Slave mode and to 1 in Master mode
3 HTO field can only be written in slave mode, LINCR1[MME] = 0.

**Figure 27-26. LIN timeout control register (LINTOCR)**

**Table 27-22. LINTOCR field descriptions**

| Field | Description |
|---|---|
| RTO | Response timeout value<br>This register contains the response timeout duration (in bit time) for 1 byte.<br>The reset value is 0xE = 14, corresponding to $T_{Response_Maximum} = 1.4 \times T_{Response_Nominal}$ |
| HTO | Header timeout value<br>This register contains the header timeout duration (in bit time). This value does not include the first 11 dominant bits of the Break. The reset value depends on which mode LINFlexD is in.<br>HTO can be written only for Slave mode. |

## 27.10.10 LIN fractional baud rate register (LINFBRR)

Offset: 0x24 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DIV_F[1] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1 This field is writable only in Initialization mode, LINCR1[INIT] = 1.

**Figure 27-27. LIN timeout control register (LINTOCR)**

**Table 27-23. LINFBRR field descriptions**

| Field | Description |
|---|---|
| DIV_F | Fraction bits of LFDIV<br>The 4 fraction bits define the value of the fraction of the LINFlexD divider (LFDIV).<br>Fraction (LFDIV) = Decimal value of DIV_F / 16.<br><br>This register can be written in Initialization mode only, LINCR1[INIT] = 1. |

## 27.10.11 LIN integer baud rate register (LINIBRR)

Offset: 0x28 Access: User read/write

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

1 This field is writable only in Initialization mode (LINCR1[INIT] = 1).

**Figure 27-28. LIN integer baud rate register (LINIBRR)**

**Table 27-24. LINIBRR field descriptions**

| Field | Description |
|---|---|
| DIV_M | LFDIV mantissa<br>These bits define the LINFlexD divider (LFDIV) mantissa value (see Table 27-25).<br>This register can be written in Initialization mode only. |

**Table 27-25. Integer baud rate selection**

| DIV_M | Mantissa |
|---|---|
| 0x0 | LIN clock disabled |
| 0x1 | 1 |
| ... | ... |
| 0xFFFFE | 1048574 |
| 0xFFFFF | 1048575 |

## 27.10.12 LIN checksum field register (LINCFR)

Offset: 0x2C Access: User read/write

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

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

**Figure 27-29. LIN checksum field register (LINCFR)**

**Table 27-26. LINCFR field descriptions**

| Field | Description |
|---|---|
| CF | Checksum bits<br>When LINCR1[CCD] is cleared, these bits are read-only. When LINCR1[CCD] is set, these bits are read/write. See Table 27-12. |

## 27.10.13 LIN control register 2 (LINCR2)

Offset: 0x30 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | IOBE[1] | IOPE[1] | WURQ | DDRQ | DTRQ | ABRQ | HTRQ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | w1c | w1c | w1c | w1c | w1c | | | | | | | | |
| Reset | 0 | 1 | 0/1[2] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

[1] These fields are writable only in Initialization mode (LINCR1[INIT] = 1.

[2] Resets to 1 in Slave mode and to 0 in Master mode

**Figure 27-30. LIN control register 2 (LINCR2)**

**Table 27-27. LINCR2 field descriptions**

| Field | Description |
|---|---|
| IOBE | Idle on Bit Error<br>0: Bit error does not reset LIN state machine<br>1: Bit error reset LIN state machine<br>This bit can be set/cleared in Initialization mode only (LINCR1[INIT]) = 1. |
| IOPE | Idle on Identifier Parity Error<br>0: Identifier Parity error does not reset LIN state machine.<br>1: Identifier Parity error reset LIN state machine.<br>This bit can be set/cleared in Initialization mode only (LINCR1[INIT]) = 1. |
| WURQ | Wake-up Generation Request<br>Setting this bit generates a wake-up pulse. It is reset by hardware when the wake-up character has been transmitted. The character sent is copied from DATA0 in BDRL buffer. Note that this bit cannot be set in Sleep mode. Software has to exit Sleep mode before requesting a wake-up. Bit error is not checked when transmitting the wake-up request. |
| DDRQ | Data Discard Request<br>Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlexD has moved to idle state. In Slave mode, this bit can be set only when HRF bit in LINSR is set and identifier did not match any filter. |
| DTRQ | Data Transmission Request<br>Set by software in Slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when HRF bit in LINSR is set.<br>Cleared by hardware when the request has been completed or aborted or on an error condition.<br>In Master mode, this bit is set by hardware when DIR bit in BIDR is set and header transmission is completed. |

**Table 27-27. LINCR2 field descriptions (continued)**

| Field | Description |
|---|---|
| ABRQ | Abort Request<br>Set by software to abort the current transmission.<br>Cleared by hardware when the transmission has been aborted. LINFlexD aborts the transmission at the end of the current bit.<br>This bit can also abort a wake-up request.<br>It can also be used in UART mode. |
| HTRQ | Header Transmission Request<br>Set by software to request the transmission of the LIN header.<br>Cleared by hardware when the request has been completed or aborted.<br>This bit has no effect in UART mode. |

## 27.10.14 Buffer identifier register (BIDR)

This register contains the fields that identify a transaction and provide other information related to it.

All the fields in this register must be updated when an ID filter (enabled) in slave mode (Tx or Rx) matches the ID received.

Offset: 0x34 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-31. Buffer identifier register (BIDR)**

**Table 27-28. BIDR field descriptions**

| Field | Description |
|---|---|
| DFL | Data Field Length<br>These bits define the number of data bytes in the response part of the frame.<br>DFL = Number of data bytes - 1.<br>Normally, LIN uses only DFL[0:2] to manage frames with a maximum of 8 bytes of data. Identifier filters are compatible with DFL[0:2] and DFL[0:5] . DFL[3:5] are provided to manage extended frames. |
| DIR | Direction<br>This bit controls the direction of the data field.<br>0: LINFlexD receives the data and copy them in the BDR registers.<br>1: LINFlexD transmits the data from the BDR registers. |

**Table 27-28. BIDR field descriptions (continued)**

| Field | Description |
|---|---|
| CCS | Classic Checksum<br>This bit controls the type of checksum applied on the current message.<br>0: Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher.<br>1: Classic Checksum covering Data fields only. This is compatible with LIN specification 1.3 and below. |
| ID | Identifier<br>Identifier part of the identifier field without the identifier parity. |

## 27.10.15 Buffer data register least significant (BDRL)

Offset: 0x38 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA3 | | | | | | | | DATA2 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DATA1 | | | | | | | | DATA0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-32. Buffer data register least significant (BDRL)**

**Table 27-29. BDRL field descriptions**

| Field | Description |
|---|---|
| DATA3 | Data Byte 3<br>Data byte 3 of the data field |
| DATA2 | Data Byte 2<br>Data byte 2 of the data field |
| DATA1 | Data Byte 1<br>Data byte 1 of the data field |
| DATA0 | Data Byte 0<br>Data byte 0 of the data field |

## 27.10.16 Buffer data register most significant (BDRM)

Offset: 0x3C

Access: User read/write

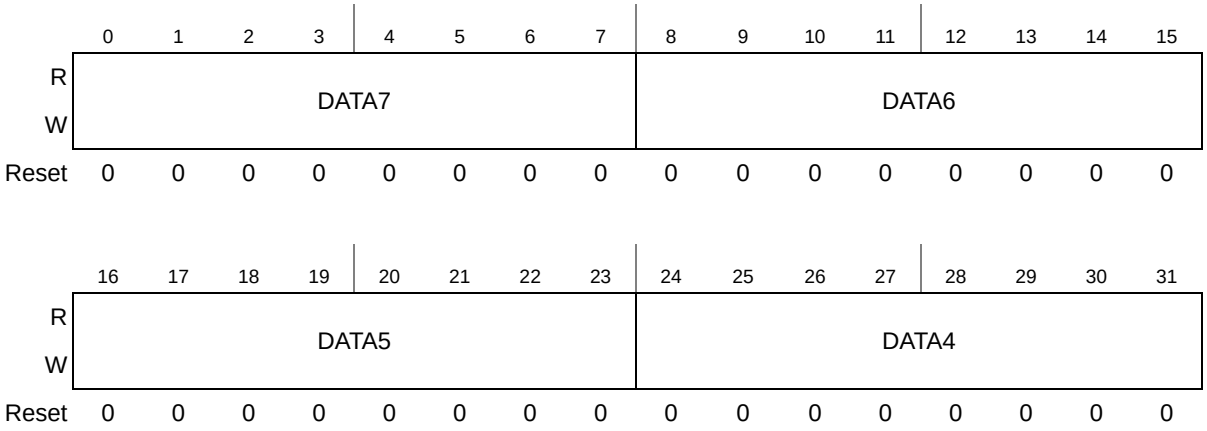



**Figure 27-33. Buffer data register most significant (BDRM)**

**Table 27-30. BDRM field descriptions**

| Field | Description |
|---|---|
| DATA7 | Data Byte 7<br>Data byte 7 of the data field |
| DATA6 | Data Byte 6<br>Data byte 6 of the data field |
| DATA5 | Data Byte 5<br>Data byte 5 of the data field |
| DATA4 | Data Byte 4<br>Data byte 4 of the data field |

## 27.10.17 Identifier filter enable register (IFER)

Offset: 0x40 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FACT[1] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1 This field is writable only in Initialization mode (LINCR1[INIT] = 1).

**Figure 27-34. Identifier filter enable register (IFER)**

**Table 27-31. IFER field descriptions**

| Field | Description |
|---|---|
| FACT | Filter active<br>The software sets the bit FACT[x] to activate the filter x in identifier list mode.<br>In identifier mask mode bits FACT(2n + 1) have no effect on the corresponding filters as they act as masks for the Identifiers 2n.<br>These bits can be set/cleared in Initialization mode only. |

## 27.10.18 Identifier filter match index (IFMI)

Offset: 0x44 Access: User read/write

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

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

**Figure 27-35. Identifier filter match index (IFMI)**

**Table 27-32. IFMI field descriptions**

| Field | Description |
|---|---|
| IFMI | Filter match index<br>This register contains the index corresponding to the received ID. It can be used to directly write or read the data in RAM (refer to Section 27.7.2, Slave mode, for more details).<br>When no filter matches, IFMI = 0. When Filter n is matching, IFMI = n + 1. |

## 27.10.19 Identifier filter mode register (IFMR)

Offset:0x48 Access: User read/write

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

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

**Figure 27-36. Identifier filter mode register (IFMR)**

**Table 27-33. IFMR field descriptions**

| Field | Description |
|---|---|
| IFM | Filter mode<br>0 Filters 2*n* and 2*n* + 1 are in identifier list mode.<br>1 Filters 2*n* and 2*n* + 1 are in mask mode (filter 2*n* + 1 is the mask for the filter 2*n*).<br>(See Table 27-34.) |

**Table 27-34. IFMR[IFM] configuration**

| Bit | Value | Result |
|---|---|---|
| IFM[0] | 0 | Filters 0 and 1 are in identifier list mode. |
| | 1 | Filters 0 and 1 are in mask mode (filter 1 is the mask for the filter 0). |
| IFM[1] | 0 | Filters 2 and 3 are in identifier list mode. |
| | 1 | Filters 2 and 3 are in mask mode (filter 3 is the mask for the filter 2). |
| IFM[2] | 0 | Filters 4 and 5 are in identifier list mode. |
| | 1 | Filters 4 and 5 are in mask mode (filter 5 is the mask for the filter 4). |

**Table 27-34. IFMR[IFM] configuration (continued)**

| Bit | Value | Result |
|---|---|---|
| IFM[3] | 0 | Filters 6 and 7 are in identifier list mode. |
| | 1 | Filters 6 and 7 are in mask mode (filter 7 is the mask for the filter 6). |
| IFM[4] | 0 | Filters 8 and 9 are in identifier list mode. |
| | 1 | Filters 8 and 9 are in mask mode (filter 9 is the mask for the filter 8). |
| IFM[5] | 0 | Filters 10 and 11 are in identifier list mode. |
| | 1 | Filters 10 and 11 are in mask mode (filter 11 is the mask for the filter 10). |
| IFM[6] | 0 | Filters 12 and 13 are in identifier list mode. |
| | 1 | Filters 12 and 13 are in mask mode (filter 13 is the mask for the filter 12). |
| IFM[7] | 0 | Filters 14 and 15 are in identifier list mode. |
| | 1 | Filters 14 and 15 are in mask mode (filter 15 is the mask for the filter 14). |

## 27.10.20 Identifier filter control registers (IFCR0–IFCR15)

The function of these registers is different depending on which mode the LINFlexD controller is in, as described in Table 27-35.

**Table 27-35. IFCR functionality based on mode**

| Mode | IFCR functionality |
|---|---|
| Identifier list | Each IFCR register acts as a filter. |
| Identifier mask | If a = (number of filters) / 2, and n = 0 to (a - 1),<br>then IFCR[2n] acts as a filter and IFCR[2n+1] acts as the mask for IFCR[2n]. |

Offsets: 0x4C–0x88 (16 registers) Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DFL[1] | | | | | | DIR[1] | CCS[1] | 0 | 0 | ID[1] | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1 These fields are writable only in Initialization mode (LINCR1[INIT] = 1).

**Figure 27-37. Identifier filter control registers (IFCR0–IFCR15)**

**Table 27-36. IFCR field descriptions**

| Field | Description |
|---|---|
| DFL | Data Field Length<br>This field defines the number of data bytes in the response part of the frame. |
| DIR | Direction<br>This bit controls the direction of the data field.<br>0: LINFlexD receives the data and copy them in the BDRL and BDRM registers.<br>1: LINFlexD transmits the data from the BDRL and BDRM registers. |
| CCS | Classic Checksum<br>This bit controls the type of checksum applied on the current message.<br>0: Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher.<br>1: Classic Checksum covering Data fields only. This is compatible with LIN specification 1.3 and below. |
| ID | Identifier<br>Identifier part of the identifier field without the identifier parity. |

## 27.10.21 Global control register (GCR)

This register can be programmed only in Initialization mode. The configuration specified in this register applies in both LIN and UART modes.

Offset: 0x8C — Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | TDFBM[1] | RDFBM[1] | TDLIS[1] | RDLIS[1] | STOP[1] | 0 |
| W | | | | | | | | | | | | | | | | SR[1] |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

[1] This field is writable only in Initialization mode (LINCR1[INIT] = 1).

**Figure 27-38. Global control register (GCR)**

**Table 27-37. GCR field descriptions**

| Field | Description |
|---|---|
| TDFBM | Transmit data first bit MSB<br>This field controls the first bit of transmitted data (payload only) as MSB/LSB in both UART and LIN modes.<br>0 The first bit of transmitted data is LSB – that is, the first bit transmitted is mapped on the LSB bit (BDR(0), BDR(8), BDR(16), BDR(24)).<br>1 The first bit of transmitted data is MSB – that is, the first bit transmitted is mapped on the MSB bit (BDR(7), BDR(15), BDR(23), BDR(31)). |
| RDFBM | Received data first bit MSB<br>This field controls the first bit of received data (payload only) as MSB/LSB in both UART and LIN modes.<br>0 The first bit of received data is LSB – that is, the first bit received is mapped on the LSB bit (BDR(0), BDR(8), BDR(16), BDR(24)).<br>1 The first bit of received data is MSB – that is, the first bit received is mapped on the MSB bit (BDR(7), BDR(15), BDR(23), BDR(31)). |
| TDLIS | Transmit data level inversion selection<br>This field controls the data inversion of transmitted data (payload only) in both UART and LIN modes.<br>0 Transmitted data is not inverted.<br>1 Transmitted data is inverted. |
| RDLIS | Received data level inversion selection<br>This field controls the data inversion of received data (payload only) in both UART and LIN modes.<br>0 Received data is not inverted.<br>1 Received data is inverted. |
| STOP | Stop bit configuration<br>This field controls the number of stop bits in transmitted data in both UART and LIN modes. The stop bit is configured for all the fields (delimiter, sync, ID, checksum, and payload).<br>0 One stop bit<br>1 Two stop bits |
| SR | Soft reset<br>If the software writes a “1” to this field, the LINFlexD controller executes a soft reset in which the FSMs, FIFO pointers, counters, timers, status registers, and error registers are reset but the configuration registers are unaffected. The resetting of this bit should also be done by software only (its not cleared by hardware automatically).<br>This field always reads “0”. |

## 27.10.22 UART preset timeout register (UARTPTO)

This register contains the preset timeout value in UART mode, and is used to monitor the IDLE state of the reception line. The timeout detection uses this register and the UARTCTO register described in Section 27.10.23, UART current timeout register (UARTCTO).

Offset: 0x90 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PTO | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 27-39. UART preset timeout register (UARTPTO)**

**Table 27-38. UARTPTO field descriptions**

| Field | Description |
|---|---|
| PTO | Preset value of the timeout counter<br>Do not set PTO = 0 (otherwise, UARTSR[TO] would immediately be set). |

## 27.10.23 UART current timeout register (UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (see Section 27.10.22, UART preset timeout register (UARTPTO)) to monitor the IDLE state of the reception line. UART timeout works in both CPU and DMA modes.

The timeout counter:

- Starts at zero and counts upward
- Is clocked with the baud rate clock prescaled by a hard-wired scaling factor of 16
- Is automatically enabled when UARTCR[RXEN] = 1

Offset: 0x94 Access: User read/write

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

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

**Figure 27-40. UART current timeout register (UARTCTO)**

**Table 27-39. UARTCTO field descriptions**

| Field | Description |
|---|---|
| CTO | Current value of the timeout counter<br>This field is reset whenever one of the following occurs:<br><br>• A new value is written to the UARTPTO register<br>• The value of this field matches the value of UARTPTO[PTO]<br>• A hard or soft reset occurs<br>• New incoming data is received<br><br>When CTO matches the value of UARTPTO[PTO], UARTSR[TO] is set. |

## 27.10.24 DMA Tx enable register (DMATXE)

This register enables the DMA Tx interface.

Offset: 0x98 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DTE 15 | DTE 14 | DTE 13 | DTE 12 | DTE 11 | DTE 10 | DTE 9 | DTE 8 | DTE 7 | DTE6 | DTE 5 | DTE 4 | DTE 3 | DTE 2 | DTE 1 | DTE 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-41. DMA Tx enable register (DMATXE)**

**Table 27-40. DMATXE field descriptions**

| Field | Description |
|---|---|
| DTE*n* | DMA Tx channel *n* enable<br>0 DMA Tx channel *n* disabled<br>1 DMA Tx channel *n* enabled<br><br>**Note:** When DMATXE = 0x0, the DMA Tx interface FSM is forced (soft reset) into the IDLE state. |

## 27.10.25 DMA Rx enable register (DMARXE)

This register enables the DMA Rx interface.

Offset: 0x9C Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DRE15 | DRE14 | DRE13 | DRE12 | DRE11 | DRE10 | DRE9 | DRE8 | DRE7 | DRE6 | DRE5 | DRE4 | DRE3 | DRE2 | DRE1 | DRE0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-42. DMA Rx enable register (DMARXE)**

**Table 27-41. DMARXE field descriptions**

| Field | Description |
|---|---|
| DRE*n* | DMA Rx channel *n* enable<br>0 DMA Rx channel *n* disabled<br>1 DMA Rx channel *n* enabled<br><br>**Note:** When DMARXE = 0x0, the DMA Rx interface FSM is forced (soft reset) into the IDLE state. |

## 27.11 DMA interface

The LINFlexD DMA interface offers a parametric and programmable solution with the following features:

- LIN Master node, TX mode: single DMA channel
- LIN Master node, RX mode: single DMA channel
- LIN Slave node, TX mode: 1 to N DMA channels where N = max number of ID filters
- LIN Slave node, RX mode: 1 to N DMA channels where N = max number of ID filters
- UART node, TX mode: single DMA channel
- UART node, RX mode: single DMA channel + timeout

The LINFlexD controller interacts with an enhanced direct memory access (eDMA) controller; see the description of that controller for details on its operation and the transfer control descriptors (TCDs) referenced in this section.

### 27.11.1 Master node, TX mode

On a master node in TX mode, the DMA interface requires a single TX channel. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is shown in Figure 27-43.

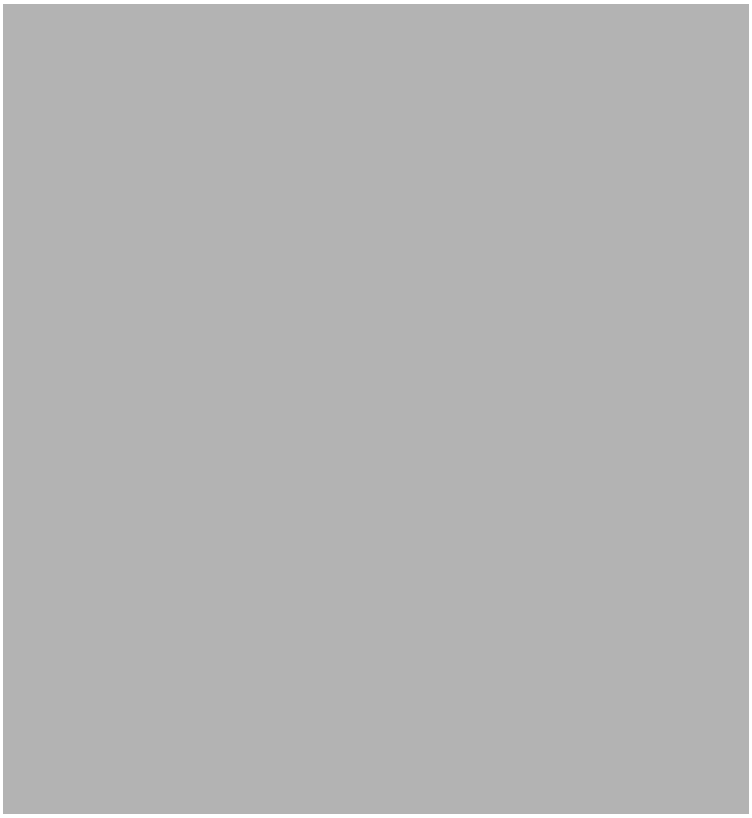


**Figure 27-43. TCD chain memory map (master node, TX mode)**

The TCD chain of the DMA Tx channel on a master node supports:

- Master to Slave: transmission of the entire frame (header + data)
- Slave to Master: transmission of the header. The data reception is controlled by the Rx channel on the master node.
- Slave to Slave: transmission of the header.

The register settings for the LINCR2 and BIDR registers for each class of LIN frame are shown in Table 27-42.

**Table 27-42. Register settings (master node, TX mode)**

| LIN frame | LINCR2 | BIDR |
|---|---|---|
| Master to Slave | DDRQ=1<br>DTRQ=0<br>HTRQ=0 | DFL = payload size<br>ID = address<br>CCS = checksum<br>DIR = 1 (TX) |

**Table 27-42. Register settings (master node, TX mode) (continued)**

| LIN frame | LINCR2 | BIDR |
|---|---|---|
| Slave to Master | DDRQ=0<br>DTRQ=0<br>HTRQ=0 | DFL = payload size<br>ID = address<br>CCS = checksum<br>DIR = 0 (RX) |
| Slave to Slave | DDRQ=1<br>DTRQ=0<br>HTRQ=0 | DFL = payload size<br>ID = address<br>CCS = checksum<br>DIR = 0 (RX) |

The concept FSM to control the DMA TX interface is shown in Figure 27-44. The DMA TX FSM will move to IDLE state immediately at next clock edge if DMATXE[0] = 0.

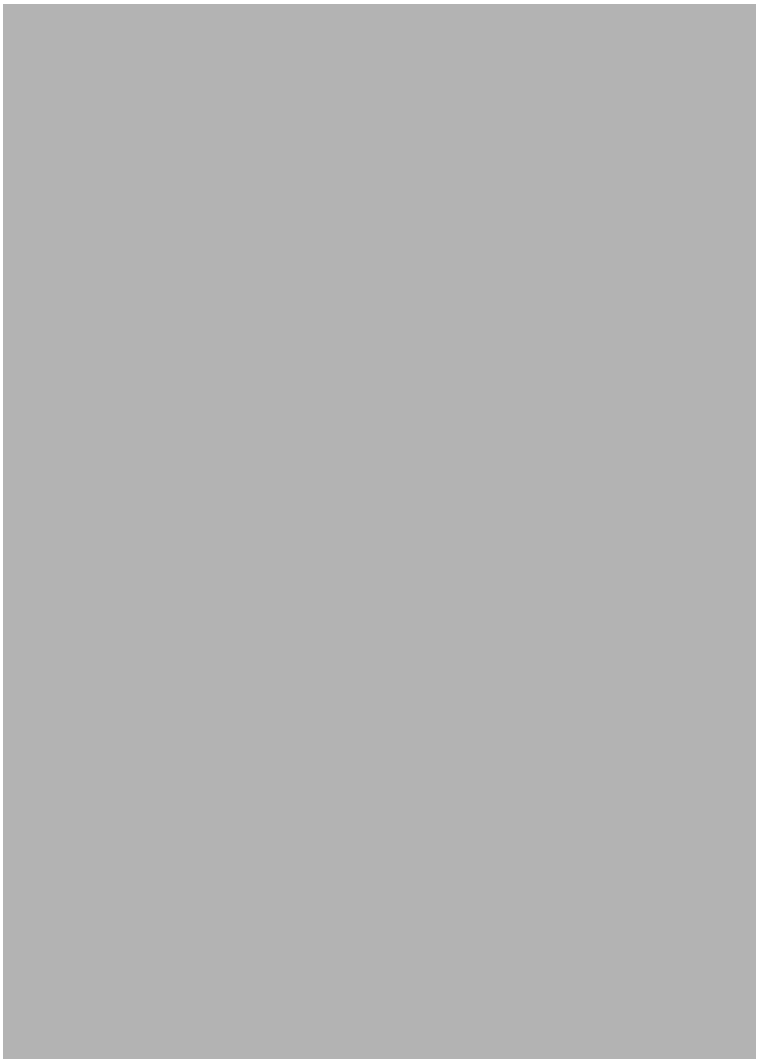


**Figure 27-44. FSM to control the DMA TX interface (master node)**

The TCD settings (word transfer) are shown in Table 27-43. All other TCD fields are equal to 0. TCD settings based on half-word or byte transfers are allowed.

**Table 27-43. TCD settings (master node, TX mode)**

| TCD field | Value | Description |
|---|---|---|
| CITER[14:0] | 1 | Single iteration for the “major” loop |
| BITER[14:0] | 1 | Single iteration for the “major” loop |
| NBYTES[31:0] | [4 + 4] + 0/4/8 = N | Data buffer is stuffed with dummy bytes if the length is not word aligned.<br>LINCR2 + BIDR + BDRL + BDRM |
| SADDR[31:0] | RAM address | |
| SOFF[15:0] | 4 | Word increment |
| SSIZE[2:0] | 2 | Word transfer |
| SLAST[31:0] | –N | |
| DADDR[31:0] | LINCR2 address | |
| DOFF[15:0] | 4 | Word increment |
| DSIZE[2:0] | 2 | Word transfer |
| DLAST_SGA[31:0] | –N | No scatter/gather processing |
| INT_MAJ | 0/1 | Interrupt disabled/enabled |
| D_REQ | 1 | Only on the last TCD of the chain. |
| START | 0 | No software request |

## 27.11.2 Master node, RX mode

On a master node in RX mode, the DMA interface requires a single RX channel. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated to the TCD chain (RAM area and LINFlexD registers) is shown in Figure 27-45.



**Figure 27-45. TCD chain memory map (master node, RX mode)**

The TCD chain of the DMA Rx channel on a master node supports Slave-to-Master reception of the data field.

The BIDR register is optionally copied into the RAM area. This BIDR field (part of FIFO data) contains the ID of each message to allow the CPU to figure out which ID was received by the LINFlexD DMA if only the “one DMA channel” setup is used.

The concept FSM to control the DMA RX interface is shown in Figure 27-46. The DMA RX FSM will move to IDLE state immediately at next clock edge if DMARXE[0]=0.

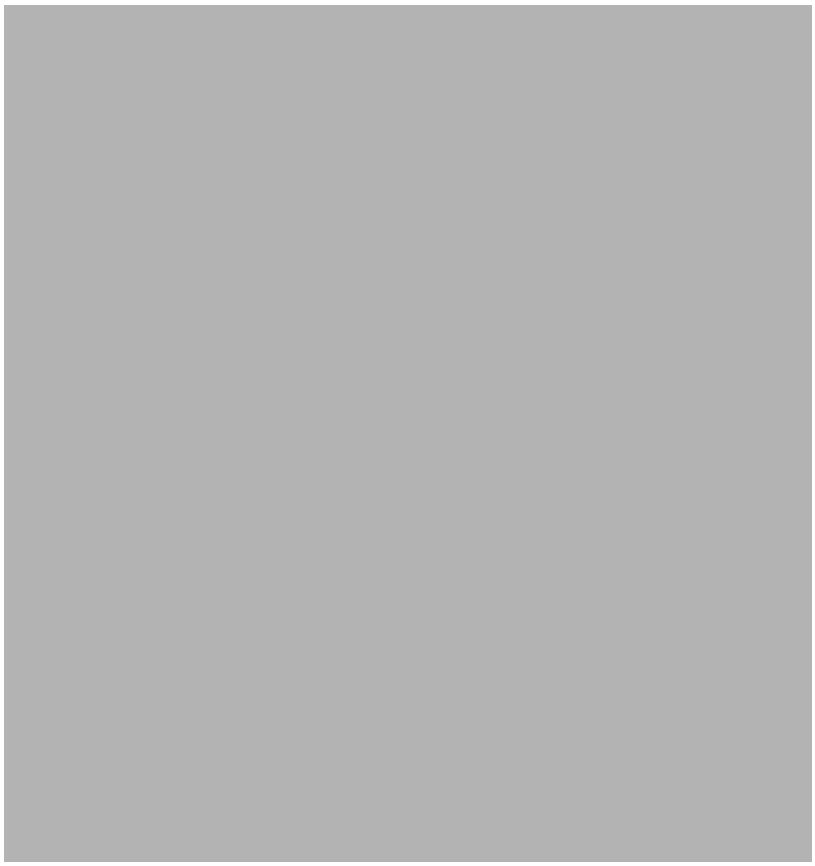


**Figure 27-46. FSM to control the DMA RX interface (master node)**

The TCD settings (word transfer) are shown in Table 27-44. All other TCD fields are equal to 0. TCD settings based on half-word or byte transfer are allowed.

**Table 27-44. TCD settings (master node, RX mode)**

| TCD field | Value | Description |
|---|---|---|
| CITER[14:0] | 1 | Single iteration for the “major” loop |
| BITER[14:0] | 1 | Single iteration for the “major” loop |
| NBYTES[31:0] | [4] + 4/8 = N | Data buffer is stuffed with dummy bytes if the length is not word aligned.<br>BIDR + BDRL + BDRM |
| SADDR[31:0] | BIDR address | — |
| SOFF[15:0] | 4 | Word increment |
| SSIZE[2:0] | 2 | Word transfer |
| SLAST[31:0] | –N | — |
| DADDR[31:0] | RAM address | — |
| DOFF[15:0] | 4 | Word increment |
| DSIZE[2:0] | 2 | Word transfer |

Table 27-44. TCD settings (master node, RX mode) (continued)

| TCD field | Value | Description |
|---|---|---|
| DLAST_SGA[31:0] | –N | No scatter/gather processing |
| INT_MAJ | 0/1 | Interrupt disabled/enabled |
| D_REQ | 1 | Only on the last TCD of the chain. |
| START | 0 | No software request |

### 27.11.3 Slave node, TX mode

On a slave node in TX mode, the DMA interface requires a DMA TX channel for each ID filter programmed in TX mode. In case a single DMA TX channel is available, a single ID field filter must be programmed in TX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated to the TCD chain (RAM area and LINFlexD registers) is shown in Figure 27-47.



Figure 27-47. TCD chain memory map (slave node, TX mode)

The TCD chain of the DMA Tx channel on a slave node supports:

- Slave to Master: transmission of the data field
- Slave to Slave: transmission of the data field

The register settings of the LINCR2, IFER, IFMR, and IFCR registers are shown in Table 27-45.

**Table 27-45. Register settings (slave node, TX mode)**

| LIN frame | LINCR2 | IFER | IFMR | IFCR |
|---|---|---|---|---|
| Slave to Master<br>or Slave to Slave | DDRQ = 0<br>DTRQ = 0<br>HTRQ = 0 | To enable an ID filter (Tx mode) for each DMA TX channel | - Identifier list mode<br>- Identifier mask mode | DFL = payload size<br>ID = address<br>CCS = checksum<br>DIR = 1(TX) |

The concept FSM to control the DMA Tx interface is shown in Figure 27-48. DMA TX FSM will move to idle state if DMATXE[*x*] = 0, where *x* = IFMI – 1.

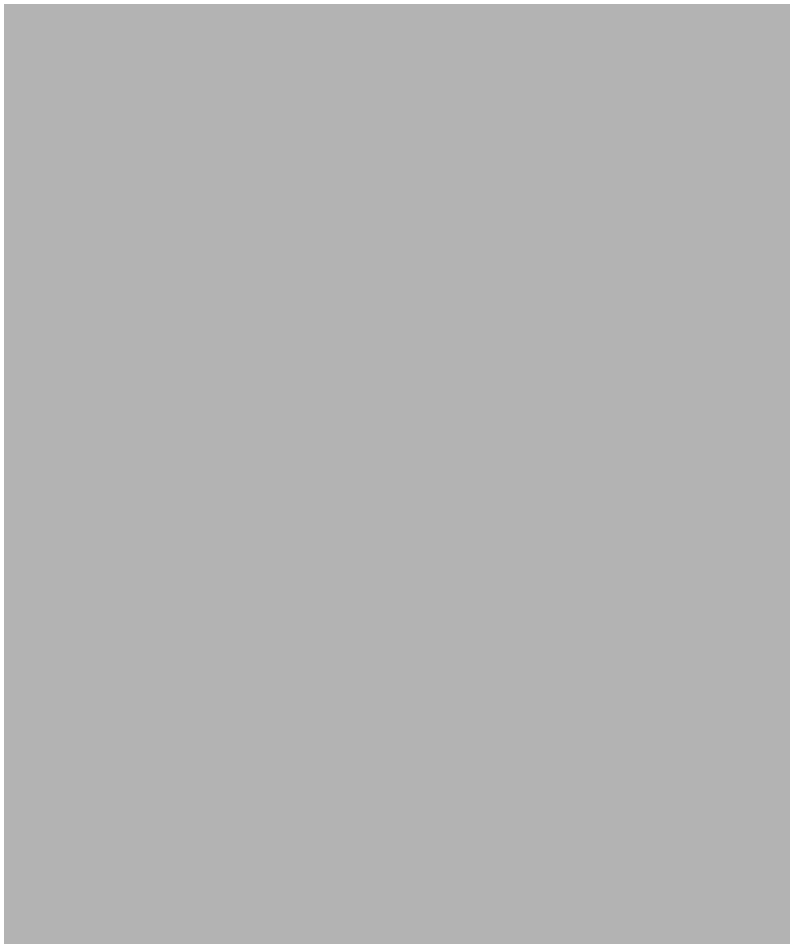


**Figure 27-48. FSM to control the DMA TX interface (slave node)**

The TCD settings (word transfer) are shown in Table 27-46. All other TCD fields are equal to 0. TCD settings based on half-word or byte transfer are allowed.

**Table 27-46. TCD settings (slave node, TX mode)**

| TCD field | Value | Description |
|---|---|---|
| CITER[14:0] | 1 | Single iteration for the “major” loop |
| BITER[14:0] | 1 | Single iteration for the “major” loop |

Table 27-46. TCD settings (slave node, TX mode) (continued)

| TCD field | Value | Description |
|---|---|---|
| NBYTES[31:0] | 4/8 = N | Data buffer is stuffed with dummy bytes if the length is not word aligned.<br>BDRL + BDRM |
| SADDR[31:0] | RAM address | |
| SOFF[15:0] | 4 | Word increment |
| SSIZE[2:0] | 2 | Word transfer |
| SLAST[31:0] | –N | |
| DADDR[31:0] | BDRL address | |
| DOFF[15:0] | 4 | Word increment |
| DSIZE[2:0] | 2 | Word transfer |
| DLAST_SGA[31:0] | –N | No scatter/gather processing |
| INT_MAJ | 0/1 | Interrupt disabled/enabled |
| D_REQ | 1 | Only on the last TCD of the chain. |
| START | 0 | No software request |

### 27.11.4 Slave node, RX mode

On a slave node in RX mode, the DMA interface requires a DMA RX channel for each ID filter programmed in RX mode. In case a single DMA RX channel is available, a single ID field filter must be programmed in RX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated to the TCD chain (RAM area and LINFlexD registers) is shown in Figure 27-49.



**Figure 27-49. TCD chain memory map (slave node, RX mode)**

The TCD chain of the DMA RX channel on a slave node supports:

- Master to Slave: reception of the data field.
- Slave to Slave: reception of the data field.

The register setting of the LINCR2, IFER, IFMR, and IFCR registers are given in Table 27-47.

**Table 27-47. Register settings (slave node, RX mode)**

| LIN frame | LINCR2 | IFER | IFMR | IFCR |
|---|---|---|---|---|
| Master to Slave or Slave to Slave | DDRQ = 0<br>DTRQ = 0<br>HTRQ = 0 | To enable an ID filter (Rx mode) for each DMA RX channel | - Identifier list mode<br>- Identifier mask mode | DFL = payload size<br>ID = address<br>CCS = checksum<br>DIR = 0 (RX) |

The concept FSM to control the DMA Rx interface is shown in Figure 27-50. DMA RX FSM will move to idle state if DMARXE[*x*] = 0 where *x* = IFMI - 1.

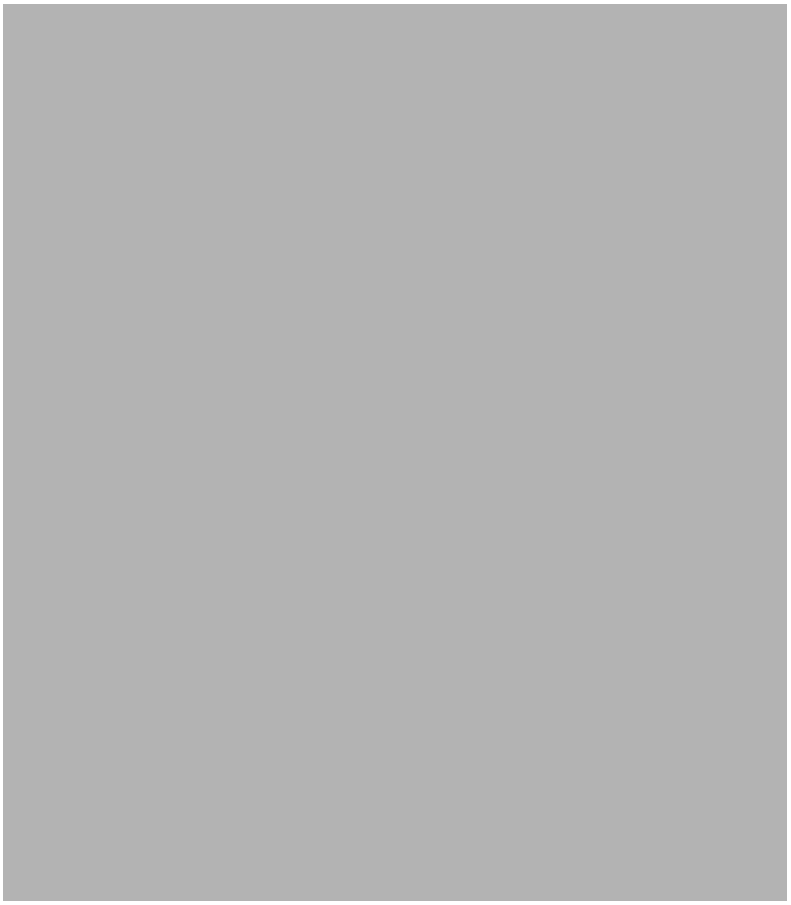


**Figure 27-50. FSM to control the DMA RX interface (slave node)**

The TCD settings (word transfer) are shown in Table 27-48. All other TCD fields = 0. TCD settings based on half-word or byte transfer are allowed.

**Table 27-48. TCD settings (slave node, RX mode)**

| TCD Field | Value | Description |
|---|---|---|
| CITER[14:0] | 1 | Single iteration for the “major” loop |
| BITER[14:0] | 1 | Single iteration for the “major” loop |
| NBYTES[31:0] | [4] + 4/8 = N | Data buffer is stuffed with dummy bytes if the length is not word aligned.<br>BIDR + BDRL + BDRM |
| SADDR[31:0] | BDRL address | |
| SOFF[15:0] | 4 | Word increment |
| SSIZE[2:0] | 2 | Word transfer |
| SLAST[31:0] | –N | |
| DADDR[31:0] | RAM address | |
| DOFF[15:0] | 4 | Word increment |

**Table 27-48. TCD settings (slave node, RX mode) (continued)**

| TCD Field | Value | Description |
|---|---|---|
| DSIZE[2:0] | 2 | Word transfer |
| DLAST_SGA[31:0] | –N | No scatter/gather processing |
| INT_MAJ | 0/1 | Interrupt disabled/enabled |
| D_REQ | 1 | Only on the last TCD of the chain. |
| START | 0 | No software request |

## 27.11.5 UART node, TX mode

In UART TX mode, the DMA interface requires a DMA TX channel. A single TCD can control the transmission of an entire Tx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is shown in Figure 27-51.



**Figure 27-51. TCD chain memory map (UART node, TX mode)**

The UART TX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the RAM to the FIFO
- Use low priority DMA channels
- Support the UART baud rate (2 Mb/s) without underrun events

The Tx FIFO size is:

- 4 bytes in 8-bit data format
- 2 half-words in 16-bit data format

A DMA request is triggered by FIFO not full (TX) status signals.

The concept FSM to control the DMA TX interface is shown in Figure 27-52. DMA TX FSM will move to idle state if DMATXE[0] = 0.

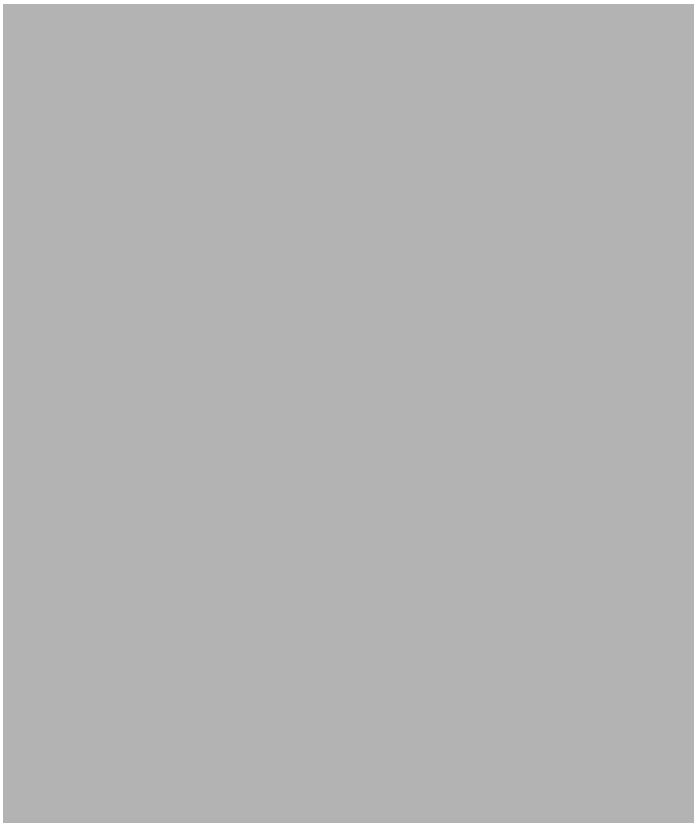


**Figure 27-52. FSM to control the DMA TX interface (UART node)**

The TCD settings (typical case) are shown in Table 27-49. All other TCD fields = 0. The minor loop transfers a single byte/half-word as soon a free entry is available in the Tx FIFO.

**Table 27-49. TCD settings (UART node, TX mode)**

| TCD Field | Value | | Description |
|---|---|---|---|
| | 8-bit data | 16-bit data | |
| CITER[14:0] | M | | Multiple iterations for the “major” loop |
| BITER[14:0] | M | | Multiple iterations for the “major” loop |
| NBYTES[31:0] | 1 | 2 | Minor loop transfer = 1 or 2 bytes |
| SADDR[31:0] | RAM address | | |
| SOFF[15:0] | 1 | 2 | Byte/Half-word increment |
| SSIZE[2:0] | 0 | 1 | Byte/Half-word transfer |
| SLAST[31:0] | -M | -M * 2 | |

**Table 27-49. TCD settings (UART node, TX mode) (continued)**

| TCD Field | Value | | Description |
|---|---|---|---|
| | 8-bit data | 16-bit data | |
| DADDR[31:0] | BDRL address | | DADDR = BDRL + 0x3 for byte transfer<br>DADDR = BDRL + 0x2 for half-word transfer |
| DOFF[15:0] | 0 | | No increment (FIFO) |
| DSIZE[2:0] | 0 | 1 | Byte/Half-word transfer |
| DLAST_SGA[31:0] | 0 | | No scatter/gather processing |
| INT_MAJ | 0/1 | | Interrupt disabled/enabled |
| D_REQ | 1 | | Only on the last TCD of the chain. |
| START | 0 | | No software request |

## 27.11.6 UART node, RX mode

In UART RX mode, the DMA interface requires a DMA RX channel. A single TCD can control the reception of an entire Rx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is shown in Figure 27-53.



**Figure 27-53. TCD chain memory map (UART node, RX mode)**

The UART RX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the FIFO to the RAM
- Use low priority DMA channels

- Support high UART baud rate (at least 2 Mb/s) without overrun events

The Rx FIFO size is:

- 4 bytes in 8-bit data format
- 2 half-words in 16-bit data format

This is sufficient because just one byte allows a reaction time of about 3.8 µs (at 2 Mbit/s), corresponding to about 450 clock cycles at 120 MHz, before the transmission is affected. A DMA request is triggered by FIFO not empty (RX) status signals.

The concept FSM to control the DMA Rx interface is shown in Figure 27-54. DMA Rx FSM will move to idle state if DMARXE[0] = 0.

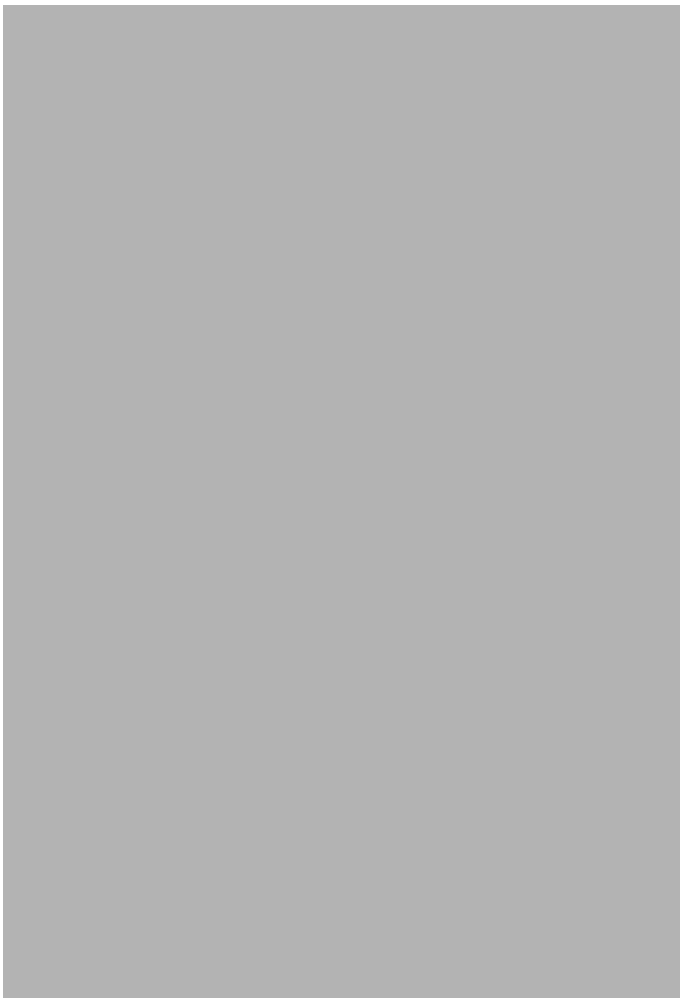


**Figure 27-54. FSM to control the DMA RX interface (UART node)**

The TCD settings (typical case) are shown in Table 27-50. All other TCD fields = 0. The minor loop transfers a single byte/half-word as soon an entry is available in the Rx FIFO. A new software reset bit is required that allows the LINFlexD FSMs to be reset in case this timeout state is reached or in any other case. Timeout counter can be re-written by software at any time to extend timeout period.

**Table 27-50. TCD settings (UART node, RX mode)**

| TCD Field | Value | | Description |
|---|---|---|---|
| | 8 bits data | 16 bits data | |
| CITER[14:0] | M | | Multiple iterations for the “major” loop |
| BITER[14:0] | M | | Multiple iterations for the “major” loop |
| NBYTES[31:0] | 1 | 2 | Minor loop transfer = 1 or 2 bytes |
| SADDR[31:0] | BDRM address | | SADDR = BDRM + 0x3 for byte transfer<br>SADDR = BDRM + 0x2 for half-word transfer |
| SOFF[15:0] | 0 | | No increment (FIFO) |
| SSIZE[2:0] | 0 | 1 | Byte/Half-word transfer |
| SLAST[31:0] | 0 | | |
| DADDR[31:0] | RAM address | | |
| DOFF[15:0] | 1 | 2 | Byte/Half-word increment |
| DSIZE[2:0] | 0 | 1 | Byte/Half-word transfer |
| DLAST_SGA[31:0] | -M | -M * 2 | No scatter/gather processing |
| INT_MAJ | 0/1 | | Interrupt disabled/enabled |
| D_REQ | 1 | | Only on the last TCD of the chain. |
| START | 0 | | No software request |

## 27.11.7 Use cases and limitations

- In LIN slave mode, the DMA capability can be used only if the ID filtering mode is activated. The number of ID filters enabled must be equal to the number of DMA channels enabled. The correspondence between channel # and ID filter is based on IFMI (identifier filter match index).
- In LIN master mode both the DMA channels (TX and RX) must be enabled in case the DMA capability is required.
- In UART mode the DMA capability can be used only if the UART Tx/Rx buffers are configured as FIFOs.
- DMA and CPU operating modes are mutually exclusive for the data/frame transfer on a UART or LIN node. Once a DMA transfer is finished the CPU can handle subsequent accesses.
- Error management must be always executed via CPU enabling the related error interrupt sources. The DMA capability does not provide support for the error management. Error management means checking status bits, handling IRQs and potentially canceling DMA transfers.
- The DMA programming model must be coherent with the TCD setting defined in this document.

## 27.12 Functional description

### 27.12.1 8-bit timeout counter

#### 27.12.1.1 LIN timeout mode

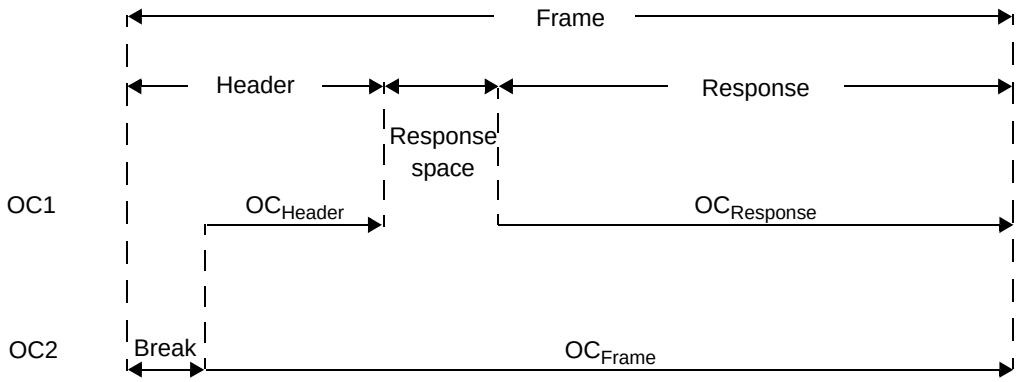



**Figure 27-55. Header and response timeout**

#### 27.12.1.2 Output compare mode

### 27.12.2 Interrupts

**Table 27-51. LINFlexD interrupt control**

| Interrupt event | Event flag bit | Enable control bit | Interrupt vector |
|---|---|---|---|
| Header Received interrupt | HRF | HRIE | RXI [1] |
| Data Transmitted interrupt | DTF | DTIE | TXI |
| Data Received interrupt | DRF | DRIE | RXI |
| Data Buffer Empty interrupt | DBEF | DBEIE | TXI |
| Data Buffer Full interrupt | DBFF | DBFIE | RXI |
| Wake-up interrupt | WUPF | WUPIE | RXI |
| LIN State interrupt [2] | LSF | LSIE | RXI |
| Buffer Overrun interrupt | BOF | BOIE | ERR |
| Framing Error interrupt | FEF | FEIE | ERR |
| Header Error interrupt | HEF | HEIE | ERR |
| Checksum Error interrupt | CEF | CEIE | ERR |
| Bit Error interrupt | BEF | BEIE | ERR |
| Output Compare interrupt | OCF | OCIE | ERR |

**Table 27-51. LINFlexD interrupt control (continued)**

| Interrupt event | Event flag bit | Enable control bit | Interrupt vector |
|---|---|---|---|
| Stuck at Zero interrupt | SZF | SZIE | ERR |

1 In Slave mode, if at least one filter is configured as TX and enabled, header received interrupt vector is RXI or TXI depending on the value of identifier received.

2 For debug and validation purposes.

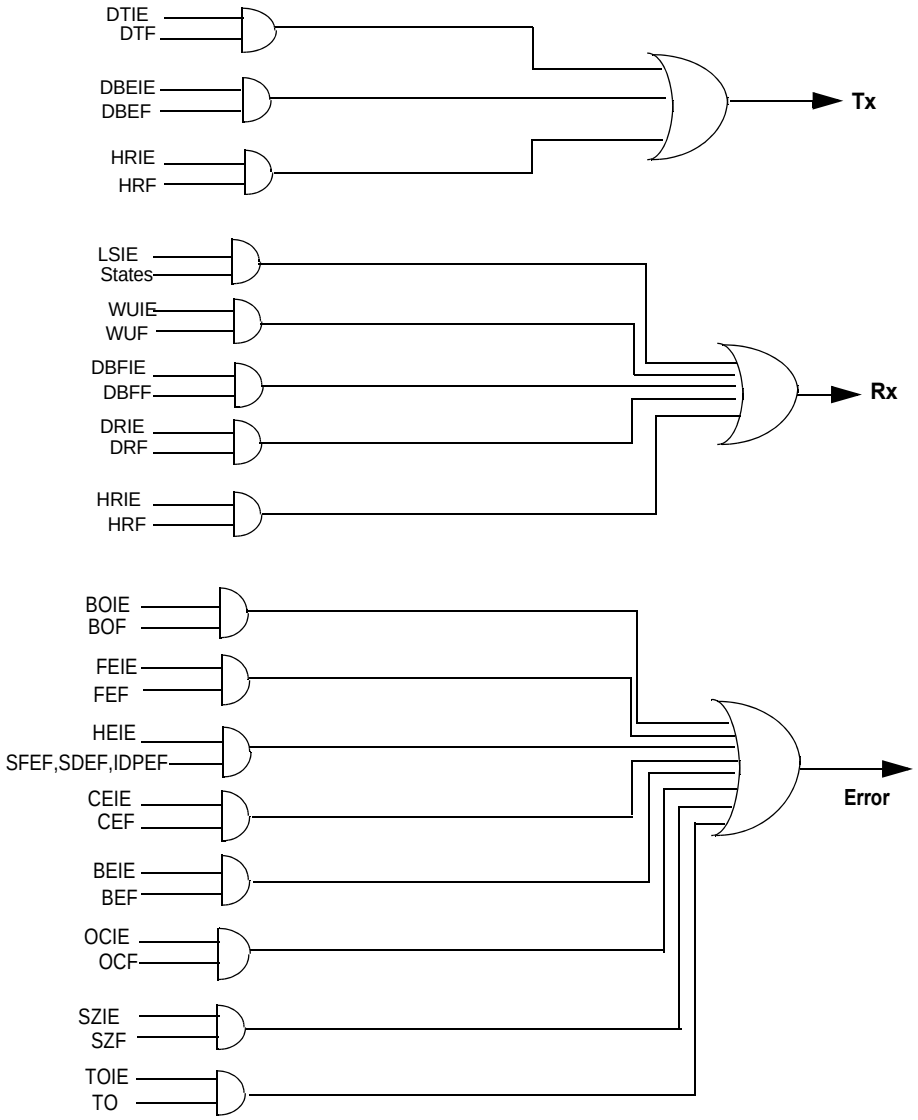



**Figure 27-56. Interrupt diagram**

## 27.12.3 Fractional baud rate generation

The baud rates for the receiver and transmitter are both set to the same value as programmed in the Mantissa (LINIBRR) and Fraction (LINFBRR) registers.

$$\text{Tx/Rx baud} = \frac{f_{ipg_clock_lin}}{(16 * LFDIV)}$$

LFDIV is an unsigned fixed point number. The 20-bit mantissa is coded in the LINIBRR register and the fraction is coded in the LINFBRR register.

The following examples show how to derive LFDIV from LINIBRR and LINFBRR register values:

**Example 27-1.**

If LINIBRR = 27d and LINFBRR = 12d, then

Mantissa (LFDIV) = 27d

Fraction (LFDIV) = 12/16 = 0.75d

Therefore LFDIV = 27.75d

**Example 27-2.**

To program LFDIV = 25.62d,

LINFBRR = 16 * 0.62 = 9.92, nearest real number 10d = Ah

LINIBRR = mantissa(25.620d) = 25d = 19h

**NOTE**

The Baud Counters are updated with the new value of the Baud Registers after a write to LINIBRR. Hence the Baud Register value must not be changed during a transaction. The LINFBRR (containing the Fraction bits) must be programmed before LINIBRR.

**NOTE**

LFDIV must be greater than or equal to 1.5d, for example, LINIBRR = 1 and LINFBRR = 8. Therefore, the maximum possible baudrate is fperiph_set_1_clk / 24.

## 27.13 Programming considerations

This section describes the various configurations in which the LINFlexD can be used.

## 27.13.1 Master node

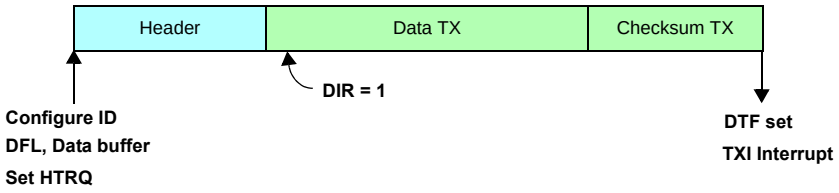



**Figure 27-57. Programming consideration: master node, transmitter**

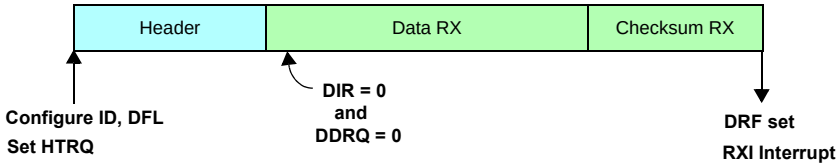
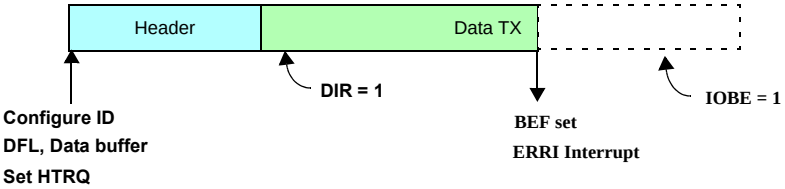



**Figure 27-58. Programming consideration: master node, receiver**



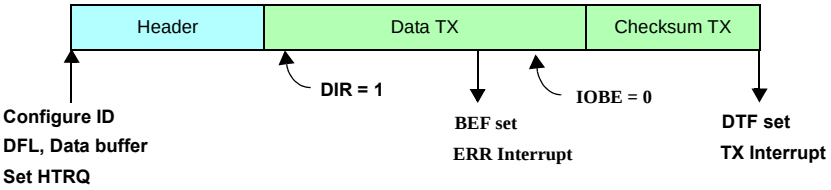




**Figure 27-59. Programming consideration: master node, transmitter, bit error**

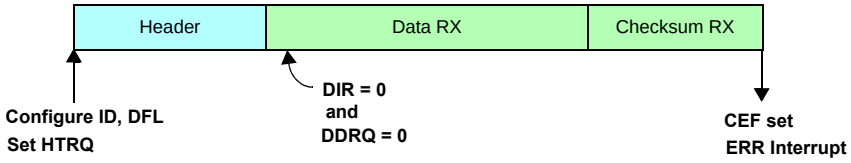



**Figure 27-60. Programming consideration: master node, receiver, checksum error**

## 27.13.2 Slave node

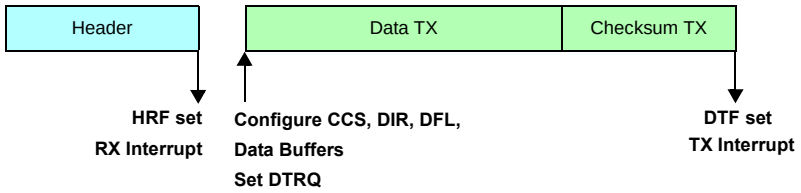



**Figure 27-61. Programming consideration: slave node, transmitter, no filters**

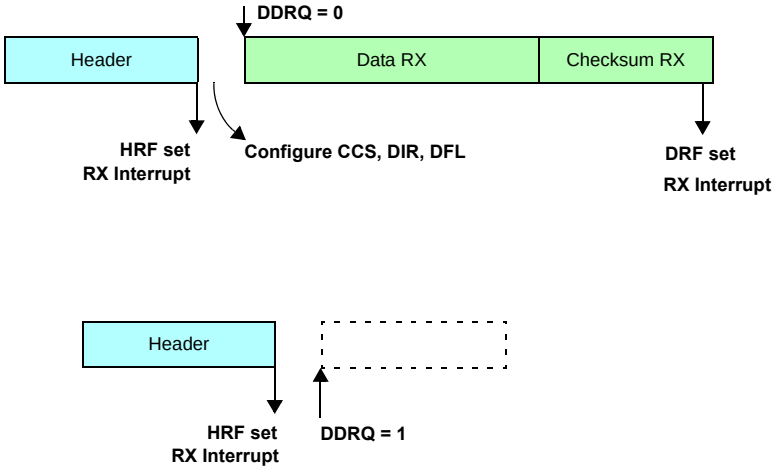



**Figure 27-62. Programming consideration: slave node, receiver, no filters**

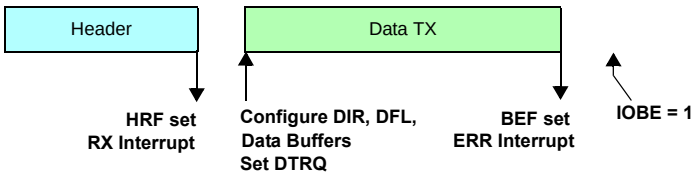



**Figure 27-63. Programming consideration: slave node, transmitter, no filters, bit error**

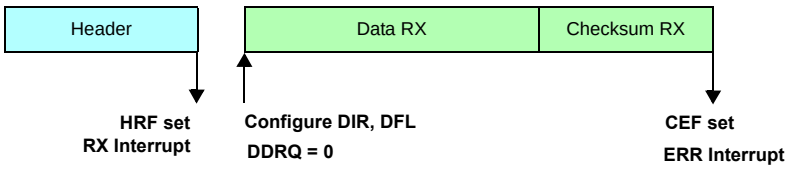
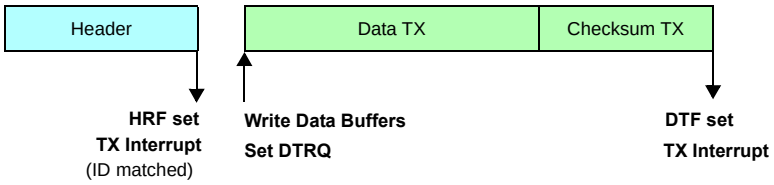



**Figure 27-64. Programming consideration: slave node, receiver, no filters, checksum error**




**Note:** This configuration can be used in case the slave never receives data (for example, as with a sensor).

**Figure 27-65. Programming consideration: slave node, at least one TX filter, BF is reset, ID matches filter**

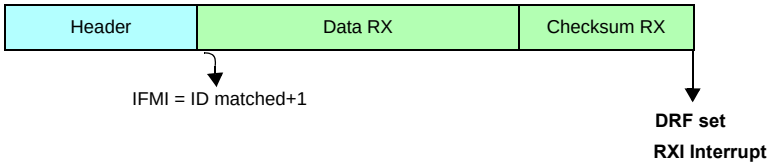



**Figure 27-66. Programming consideration: slave node, at least one RX filter, BF is reset, ID matches filter**

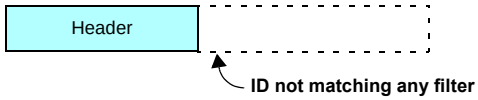
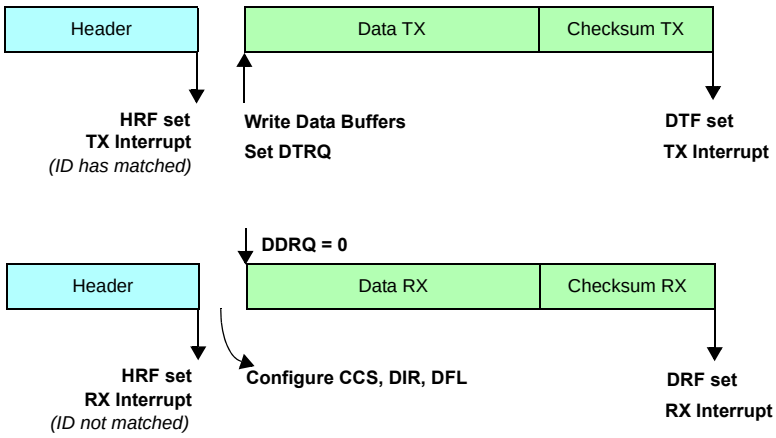



**Figure 27-67. Programming consideration: slave node, RX only, TX only, RX and TX filters, ID not matching filter, BF is reset**




**Note:** This configuration is used when:
a) All TX IDs are managed by filters
b) The number of other filters is not enough to manage all reception IDs

**Figure 27-68. Programming consideration: slave node, TX filter, BF is set**

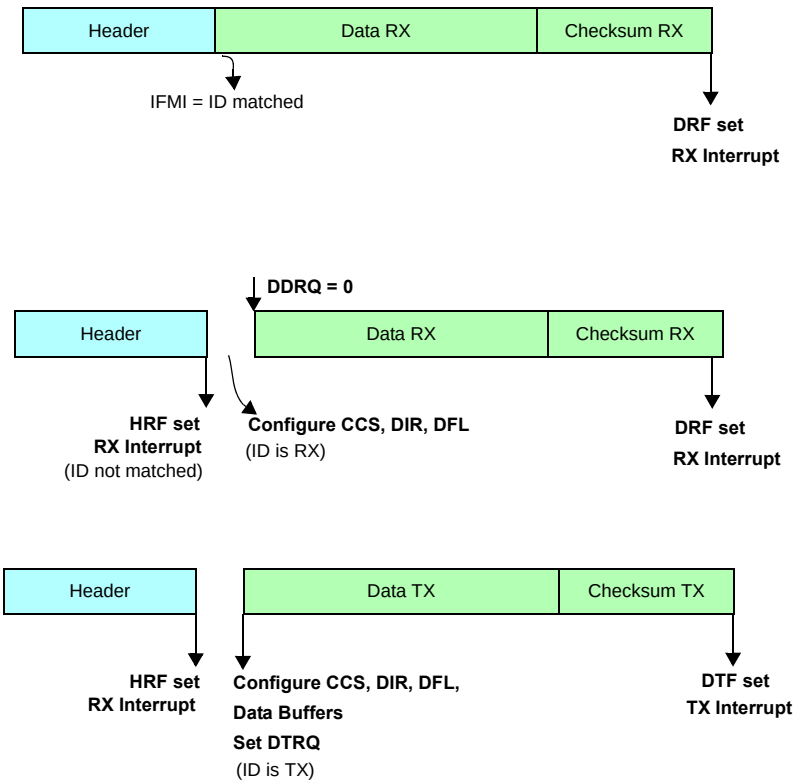
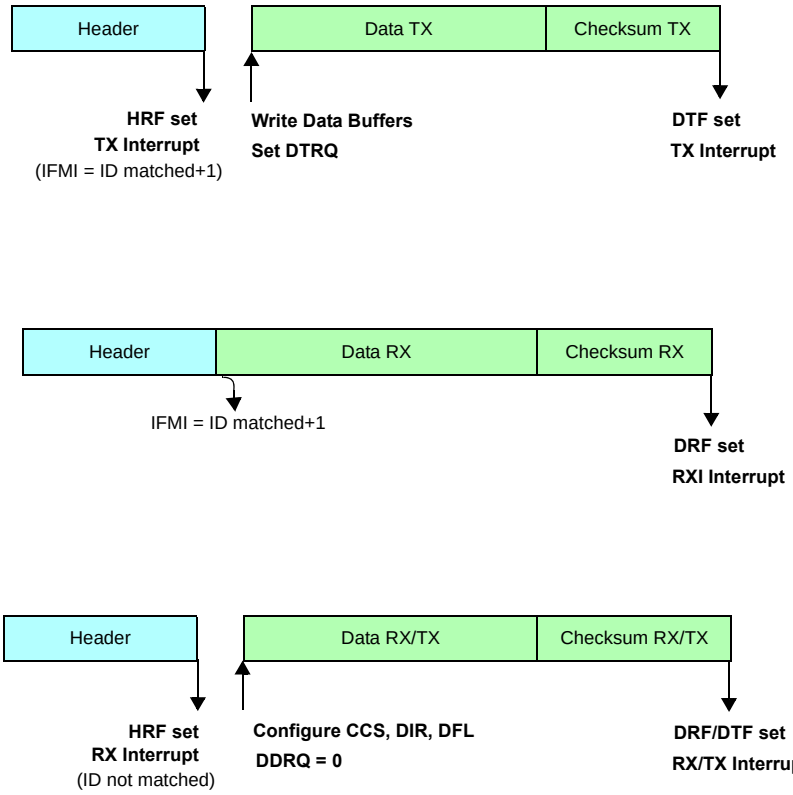



**Figure 27-69. Programming consideration: slave node, RX filter, BF is set**




**Note:** This configuration is used when:

a) The number of filters is not enough

b) Filters are used for most frequently-used IDs to reduce CPU usage

**Figure 27-70. Programming consideration: slave node, TX filter, RX filter, BF is set**

## 27.13.3 Extended frames

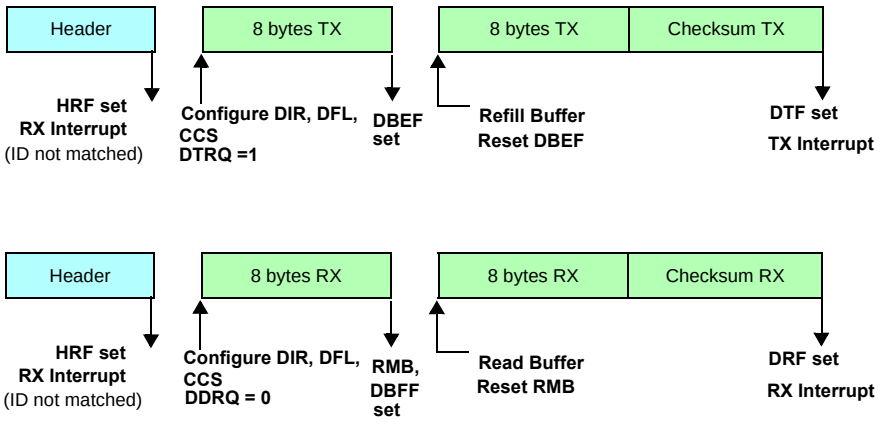



**Figure 27-71. Programming consideration: extended frames**

## 27.13.4 Timeout

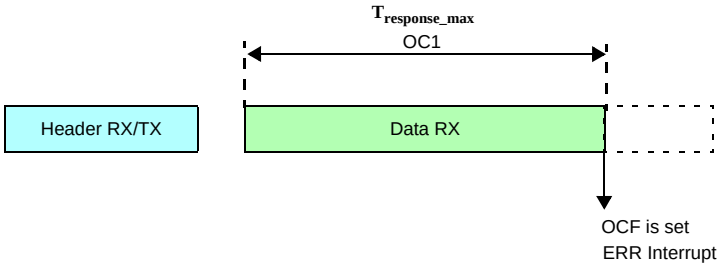



**Figure 27-72. Programming consideration: response timeout**

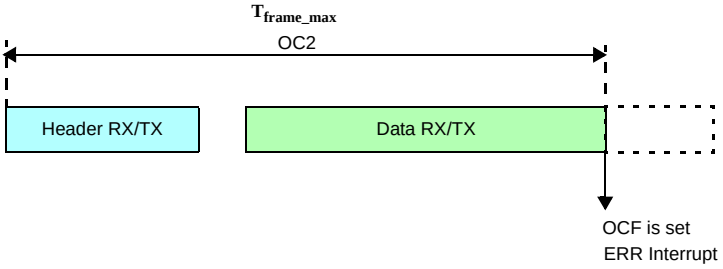



**Figure 27-73. Programming consideration: frame timeout**

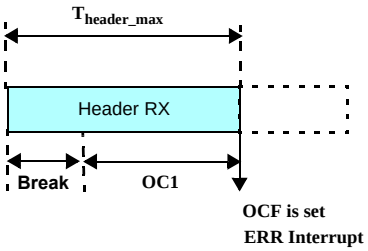



**Figure 27-74. Programming consideration: header timeout**

## 27.13.5 UART mode

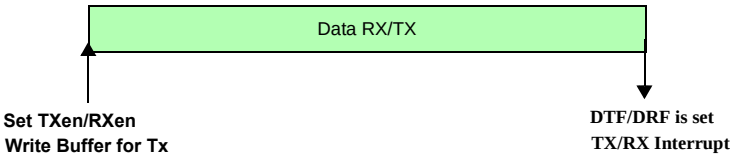



**Figure 27-75. Programming consideration: UART mode**

## 27.14 Chapter revision history

**Table 27-52. Conditional text tags**

| Tag name | Tag condition indicator | Tag use |
|---|---|---|
| **Standard tags** | | |
| cust_info | cust_info | Show in customer documentation. Used for text that may not be useful in SRS guide. |
| Editor_Notes | | Notes from editors, regarding formatting, notes to writers, and special issues or problems. |
| Footer:Confidential | | Used on master page for confidential footers/headers. |
| Footer:Preliminary | | Used for pre-release, preliminary documents. Used on master pages in footer. |
| internal_only | | Internal information only |
| MarketAssessment | | Market assessment: product in development |
| non_cust | | Information not for customer |
| Review_Q&A | | Used for asking and answering questions between writers and reviewers. |
| ReviewCopyOnly | ReviewCopyOnly | Review copy only |
| statement | | Information that is excluded from functional verification tagging |
| TBD | | To be defined |

**Table 27-53. Revision history**

| DSync version | Revision date | Author | Description of changes Internal | Description of changes External |
|---|---|---|---|---|
| From this version LINFlex3_new.fm will be maintained as LINFlex chapter. | | | | |
| 1.2 | 15 Jan 2014 | Ankur Srivastava | No changes | — |
| | | | **Rainbow RM Rev. 7 RC2** | |
| 1.3 | 18 Feb 2014 | A. Srivastava (FSL) | No changes. | — |
| | | | **Rev. 7** | |
| 1.4 | 24 Feb 2015 | A. Srivastava (FSL) | In LINTCSR register diagram (Figure 27-24), updated bit name of LTOOM to LTOM. | Editorial update |
| | | | — | |

This page is intentionally left blank.

# Chapter 28
# Memory Protection Unit (MPU)

## 28.1 Introduction

The AMBA-AHB Memory Protection Unit (MPU) provides hardware access control for all memory references generated in the device. Using preprogrammed region descriptors which define memory spaces and their associated access rights, the MPU concurrently monitors all system bus transactions and evaluates the appropriateness of each transfer. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with a protection error response. This module is commonly included as part of the platform.

### 28.1.1 Overview

The MPU module provides the following capabilities:

- Support for 16 program-visible 128-bit (4-word) region descriptors
  - Each region descriptor defines a modulo-32 byte space, aligned anywhere in memory
    - Region sizes can vary from a minimum of 32 bytes to a maximum of 4 GB
  - Two types of access control permissions defined in single descriptor word
    - Processors have separate {read, write, execute} attributes for supervisor and user accesses
    - Non-processor masters have {read, write} attributes
  - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues
  - Alternate programming model view of the access control permissions word
- Memory-mapped platform device
  - Interface to 4 slave AHB ports: flash controller (instruction port), system RAM controller, Graphics RAM (non-Z160 port), and IPS peripherals bus
    - Connections to the AHB address phase address and attributes
    - Typical location is immediately “downstream” of the platform’s crossbar switch
  - Connection to the IPS bus provides access to the MPU’s programming model

A simplified block diagram of the AHB_MPU module is shown in Figure 28-1. The AHB bus slave ports (s{0,1,2,3}_h*) are shown on the left side of the diagram, the region descriptor registers in the middle and the IPS bus interface (ips_*) on the right side. The evaluation macro contains the two magnitude comparators connected to the start and end address registers from each region descriptor (rgdn) as well as the combinational logic blocks to determine the region hit and the access protection error. For information on the details of the access evaluation macro, see Section 28.3.1, “Access evaluation macro.

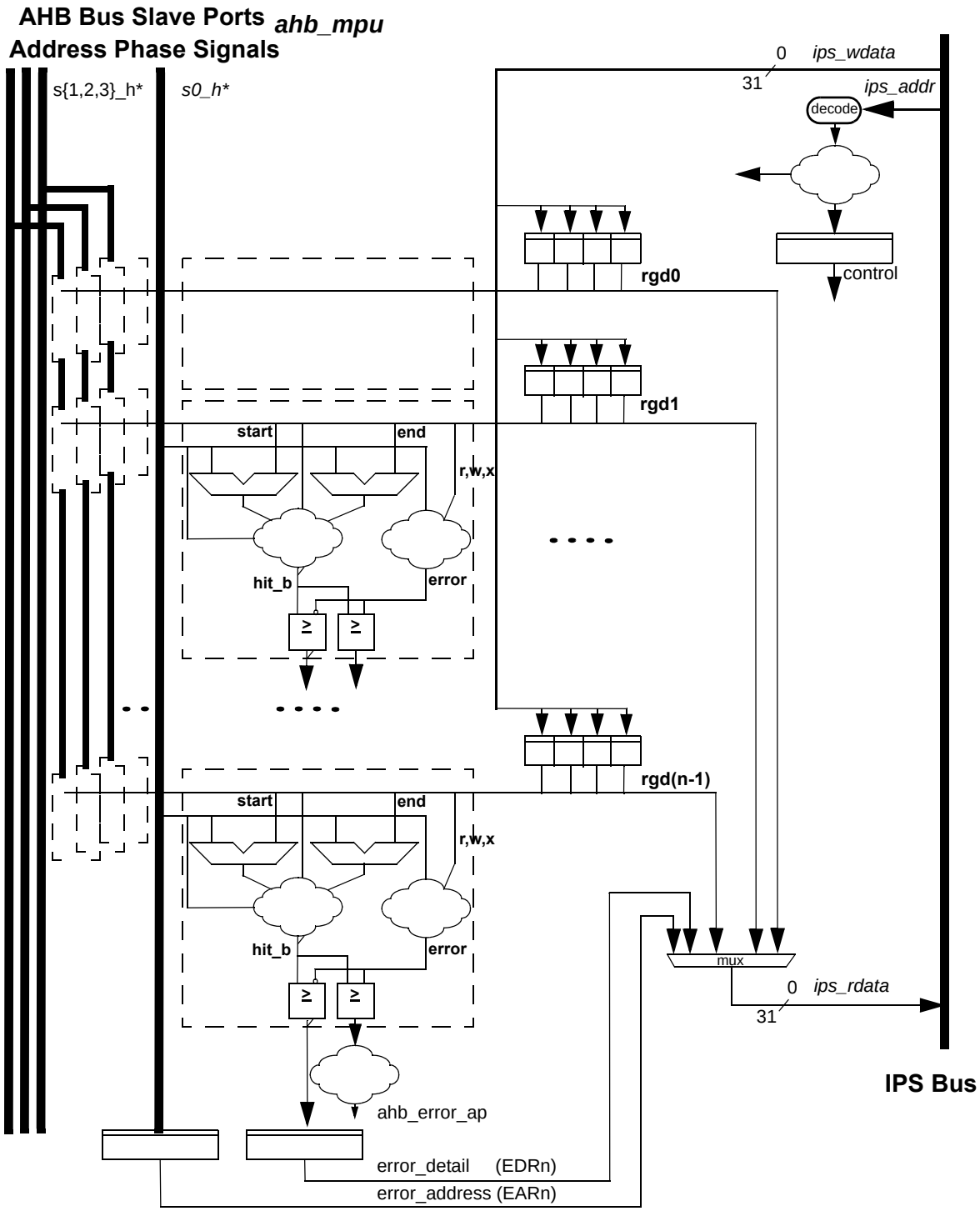



**Figure 28-1. AHB_MPU block diagram**

## 28.1.2 Features

The Memory Protection Unit implements a two-dimensional hardware array of memory region descriptors and the crossbar slave AHB ports to continuously monitor the legality of every memory reference generated by each bus master in the system. The feature set includes:

- Support for 16 memory region descriptors, each 128 bits in size
  - Specification of start and end addresses provide granularity for region sizes from 32 bytes to 4 GBytes
  - Access control definitions: 2 bus masters (processor cores) support the traditional {read, write, execute} permissions with independent definitions for supervisor and user mode accesses
  - Automatic hardware maintenance of the region descriptor valid bit removes issues associated with maintaining a coherent image of the descriptor
  - Alternate memory view of the access control word for each descriptor provides an efficient mechanism to dynamically alter only the access rights of a descriptor
  - For overlapping region descriptors, priority is given to permission granting over access denying as this approach provides more flexibility to system software. See Section 28.3.2, “Putting It All Together and AHB Error Terminations” for details and Section 28.6, “Application information” for an example.
- Support for 3 AHB slave port connections: flash controller, system ram controller and IPS peripherals bus
  - MPU hardware continuously monitors every AHB slave port access using the preprogrammed memory region descriptors
  - An access protection error is detected if a memory reference does not hit in any memory region or the reference is flagged as illegal in all memory regions where it does hit. In the event of an access error, the AHB reference is terminated with an error response and the MPU inhibits the bus cycle being sent to the targeted slave device.
  - 64-bit error registers, one for each AHB slave port, capture the last faulting address, attributes and “detail” information
- Global MPU enable/disable control bit provides a mechanism to easily load region descriptors during system startup or allow complete access rights during debug with the module disabled

## 28.1.3 Modes of operation

The MPU module does not support any special modes of operation. As a memory-mapped device located on the platform’s high-speed system bus, it responds based strictly on the memory addresses of the connected system buses. The IPS bus is used to access the MPU’s programming model and the memory protection functions are evaluated on a reference-by-reference basis using the addresses from the AHB system bus port(s).

Power dissipation is minimized when the MPU’s global enable/disable bit is cleared (MPU_CESR[VLD] = 0).

### 28.1.4 External signal description

The MPU module does not include any external interface. The MPU's internal interfaces include an IPS connection for accessing the programming model and multiple connections to the address phase signals of the platform crossbar's slave AHB ports. From a platform topology viewpoint, the MPU module appears to be directly connected "downstream" from the crossbar switch with interfaces to the AHB slave ports.

## 28.2 Memory map and register description

The MPU module provides an IPS programming model mapped to an SPP-standard on-platform 16 Kbyte space. The programming model is partitioned into three groups: control/status registers, the data structure containing the region descriptors and the alternate view of the region descriptor access control values.

The programming model can only be referenced using 32-bit (word) accesses. Attempted references using different access sizes, to undefined (reserved) addresses, or with a non-supported access type (for example, a write to a read-only register or a read of a write-only register) generate an IPS error termination.

Finally, the programming model allocates space for an MPU definition with 8 region descriptors and up to 3 AHB slave ports, like flash controller, system ram controller and IPS peripherals bus.

### 28.2.1 Memory map

The MPU programming model map is shown in Table 28-1.

**Table 28-1. MPU memory map**

| Offset address | Register name | Register description | Size (bits) | Access L | ocation |
|---|---|---|---|---|---|
| 0x0000 | MPU_CESR | MPU Control/Error Status Register | 32 | R/ partial-W | on page 1048 |
| 0x0004-0x000f | Reserved | | | | |
| 0x0010 | MPU_EAR0 | MPU Error Address Register, Slave Port 0 | 32 | R-only | on page 1049 |
| 0x0014 | MPU_EDR0 | MPU Error Detail Register, Slave Port 0 | 32 | R-only | on page 1049 |
| 0x0018 | MPU_EAR1 | MPU Error Address Register, Slave Port 1 | 32 | R-only | on page 1049 |
| 0x001c | MPU_EDR1 | MPU Error Detail Register, Slave Port 1 | 32 | R-only | on page 1049 |
| 0x0020 | MPU_EAR2 | MPU Error Address Register, Slave Port 2 | 32 | R-only | on page 1049 |
| 0x0024 | MPU_EDR2 | MPU Error Detail Register, Slave Port 2 | 32 | R-only | on page 1049 |
| 0x0028 | MPU_EAR3 | MPU Error Address Register, Slave Port 3 | 32 | R-only | on page 1049 |
| 0x002c | MPU_EDR3 | MPU Error Detail Register, Slave Port 3 | 32 | R-only | on page 1049 |
| 0x0030-0x03ff | Reserved | | | | |
| 0x0400 | MPU_RGD0 | MPU Region Descriptor 0 | 128 | R/W | on page 1050 |
| 0x0410 | MPU_RGD1 | MPU Region Descriptor 1 | 128 | R/W | on page 1050 |

**Table 28-1. MPU memory map (continued)**

| Offset address | Register name | Register description | Size (bits) | Access L | ocation |
|---|---|---|---|---|---|
| 0x0420 | MPU_RGD2 | MPU Region Descriptor 2 | 128 | R/W | on page 1050 |
| 0x0430 | MPU_RGD3 | MPU Region Descriptor 3 | 128 | R/W | on page 1050 |
| 0x0440 | MPU_RGD4 | MPU Region Descriptor 4 | 128 | R/W | on page 1050 |
| 0x0450 | MPU_RGD5 | MPU Region Descriptor 5 | 128 | R/W | on page 1050 |
| 0x0460 | MPU_RGD6 | MPU Region Descriptor 6 | 128 | R/W | on page 1050 |
| 0x0470 | MPU_RGD7 | MPU Region Descriptor 7 | 128 | R/W | on page 1050 |
| 0x0480 | MPU_RGD8 | MPU Region Descriptor 8 | 128 | R/W | on page 1050 |
| 0x0490 | MPU_RGD9 | MPU Region Descriptor 9 | 128 | R/W | on page 1050 |
| 0x04A0 | MPU_RGD10 | MPU Region Descriptor 10 | 128 | R/W | on page 1050 |
| 0x04B0 | MPU_RGD11 | MPU Region Descriptor 11 | 128 | R/W | on page 1050 |
| 0x04C0 | MPU_RGD12 | MPU Region Descriptor 12 | 128 | R/W | on page 1050 |
| 0x04D0 | MPU_RGD13 | MPU Region Descriptor 13 | 128 | R/W | on page 1050 |
| 0x04E0 | MPU_RGD14 | MPU Region Descriptor 14 | 128 | R/W | on page 1050 |
| 0x04F0 | MPU_RGD15 | MPU Region Descriptor 15 | 128 | R/W | on page 1050 |
| 0x0500–0x07FF | Reserved | | | | |
| 0x0800 | MPU_RGDAAC0 | MPU RGD Alternate Access Control 0 | 32 | R/W | on page 1055 |
| 0x0804 | MPU_RGDAAC1 | MPU RGD Alternate Access Control 1 | 32 | R/W | on page 1055 |
| 0x0808 | MPU_RGDAAC2 | MPU RGD Alternate Access Control 2 | 32 | R/W | on page 1055 |
| 0x080c | MPU_RGDAAC3 | MPU RGD Alternate Access Control 3 | 32 | R/W | on page 1055 |
| 0x0810 | MPU_RGDAAC4 | MPU RGD Alternate Access Control 4 | 32 | R/W | on page 1055 |
| 0x0814 | MPU_RGDAAC5 | MPU RGD Alternate Access Control 5 | 32 | R/W | on page 1055 |
| 0x0818 | MPU_RGDAAC6 | MPU RGD Alternate Access Control 6 | 32 | R/W | on page 1055 |
| 0x081c | MPU_RGDAAC7 | MPU RGD Alternate Access Control 7 | 32 | R/W | on page 1055 |
| 0x0820 | MPU_RGDAAC8 | MPU RGD Alternate Access Control 8 | 32 | R/W | on page 1055 |
| 0x0824 | MPU_RGDAAC9 | MPU RGD Alternate Access Control 9 | 32 | R/W | on page 1055 |
| 0x0828 | MPU_RGDAAC10 | MPU RGD Alternate Access Control 10 | 32 | R/W | on page 1055 |
| 0x082C | MPU_RGDAAC11 | MPU RGD Alternate Access Control 11 | 32 | R/W | on page 1055 |
| 0x0830 | MPU_RGDAAC12 | MPU RGD Alternate Access Control 12 | 32 | R/W | on page 1055 |
| 0x0834 | MPU_RGDAAC13 | MPU RGD Alternate Access Control 13 | 32 | R/W | on page 1055 |
| 0x0838 | MPU_RGDAAC14 | MPU RGD Alternate Access Control 14 | 32 | R/W | on page 1055 |

Table 28-1. MPU memory map (continued)

| Offset address | Register name | Register description | Size (bits) | Access L | ocation |
|---|---|---|---|---|---|
| 0x083C | MPU_RGDAAC15 | MPU RGD Alternate Access Control 15 | 32 | R/W | on page 1055 |
| 0x0840-0x3FFF | Reserved | | | | |

## 28.2.2 Register description

The following sections detail the individual registers within the MPU's programming model.

### 28.2.2.1 MPU Control/Error Status Register (MPU_CESR)

The MPU_CESR provides one byte of error status plus three bytes of configuration information. A global MPU enable/disable bit is also included in this register.

Offset MPU_Base + 0x000 Access: Read/Partial Write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SPERR | | | | | | | | 1 | 0 | 0 | 0 | HRL | | | | NSP | | | | NRGD | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | VLD |
| W | w1c | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | * | * | * | * | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 28-2. MPU Control/Error Status Register (MPU_CESR)

Table 28-2. MPU_CESR field descriptions

| Field De | scription |
|---|---|
| SPERR | Slave Port n Error, where the slave port number matches the bit number. Each bit in this field represents a flag maintained by the MPU for signaling the presence of a captured error contained in the MPU_EARn and MPU_EDRn registers. The individual bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared when the corresponding bit is written as a logical one. If another error is captured at the exact same cycle as a write of a logical one, this flag remains set. A "find first one" instruction (or equivalent) can be used to detect the presence of a captured error.<br>0 The corresponding MPU_EARn/MPU_EDRn registers do not contain a captured error.<br>1 The corresponding MPU_EARn/MPU_EDRn registers do contain a captured error. |
| HRL | Hardware Revision Level. This 4-bit read-only field specifies the MPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module. |
| NSP | Number of Slave Ports. This 4-bit read-only field specifies the number of slave ports [1-8] connected to the MPU. This field contains values of 0b0001-0b1000, depending on the device configuration. The MPC5645S has 4 slaves connected to the MPU. |

**Table 28-2. MPU_CESR field descriptions**

| Field De | scription |
|---|---|
| NRGD | Number of Region Descriptors. This 4-bit read-only field specifies the number of region descriptors implemented in the MPU. The defined encodings include:<br>0000 8 region descriptors<br>0001 12 region descriptors<br>0010 16 region descriptors (the number of descriptors on the MPC5645S) |
| VLD | Valid. This bit provides a global enable/disable for the MPU.<br>0 The MPU is disabled.<br>1 The MPU is enabled.<br>While the MPU is disabled, all accesses from all bus masters are allowed. |

### 28.2.2.2 MPU Error Address Register, Slave Port n (MPU_EARn)

When the MPU detects an access error on slave port n, the 32-bit reference address is captured in this read-only register and the corresponding bit in the MPU_CESR[SPERR] field set. Additional information about the faulting access is captured in the corresponding MPU_EDRn register at the same time. Note this register and the corresponding MPU_EDRn register contain the most recent access error; there are no hardware interlocks with the MPU_CESR[SPERR] field as the error registers are always loaded upon the occurrence of each protection violation.

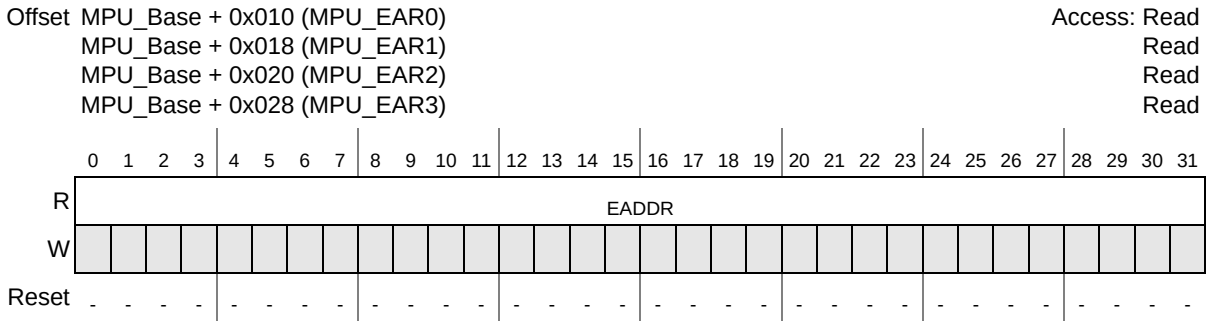



**Figure 28-3. MPU Error Address Register, Slave Port n (MPU_EARn)**

**Table 28-3. MPU_EARn Field Descriptions**

| Field De | scription |
|---|---|
| EADDR | Error Address. This read-only field is the reference address from slave port n that generated the access error. |

### 28.2.2.3 MPU Error Detail Register, Slave Port n (MPU_EDRn)

When the MPU detects an access error on slave port n, 32 bits of error detail are captured in this read-only register and the corresponding bit in the MPU_CESR[SPERR] field set. Information on the faulting address is captured in the corresponding MPU_EARn register at the same time. Note this register and the corresponding MPU_EARn register contain the most recent access error; there are no hardware interlocks with the MPU_CESR[SPERR] field as the error registers are always loaded upon the occurrence of each protection violation.

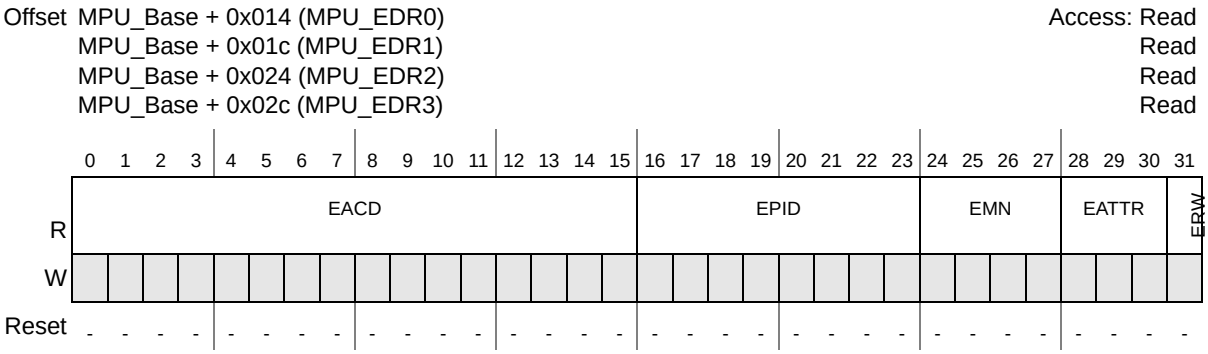



**Figure 28-4. MPU Error Detail Register, Slave Port n (MPU_EDRn)**

**Table 28-4. MPU_EDRn field descriptions**

| Field D | escription |
|---|---|
| 0–15<br>EACD | Error Access Control Detail. This 16-bit read-only field implements one bit per region descriptor and is an indication of the region descriptor hit logically anded with the access error indication. The MPU performs a reference-by-reference evaluation to determine the presence/absence of an access error. When an error is detected, the hit-qualified access control vector is captured in this field.<br><br>If the MPU_EDRn register contains a captured error and the EACD field is all zeroes, this signals an access that did not hit in any region descriptor. All non-zero EACD values signal references that hit in a region descriptor(s), but failed due to a protection error as defined by the specific set bits. If only a single EACD bit is set, then the protection error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, then the protection error was caused in an overlapping set of region descriptors. |
| 16–23<br>EPID | Error Process Identification. This 8-bit read-only field records the process identifier of the faulting reference. The process identifier is typically driven only by processor cores; for other bus masters, this field is cleared. |
| 24–27<br>EMN | Error Master Number. This 4-bit read-only field records the logical master number of the faulting reference. This field is used to determine the bus master that generated the access error. |
| 28–30<br>EATTR | Error Attributes. This 3-bit read-only field records attribute information about the faulting reference. The supported encodings are defined as:<br>0b000User mode, instruction access<br>0b001User mode, data access<br>0b010Supervisor mode, instruction access<br>0b011 Supervisor mode, data access<br>All other encodings are reserved. For non-CPU bus masters, the access attribute information is typically wired to supervisor, data (0b011). |
| 31<br>ERW | Error Read/Write. This 1-bit read-only field signals the access type (read, write) of the faulting reference.<br>0 Read<br>1 Write |

## 28.2.2.4 MPU Region Descriptor n (MPU_RGDn)

Each 128-bit (16 byte) region descriptor specifies a given memory space and the access attributes associated with that space. The descriptor definition is the very essence of the operation of the Memory Protection Unit.

The region descriptors are organized sequentially in the MPU's programming model and each of the four 32-bit words are detailed in the subsequent sections.

#### 28.2.2.4.1 MPU Region Descriptor n, Word 0 (MPU_RGDn.Word0)

The first word of the MPU region descriptor defines the 0-modulo-32 byte start address of the memory region. Writes to this word clear the region descriptor's valid bit (see Section 28.2.2.4.4, "MPU Region Descriptor n, Word 3 (MPU_RGDn.Word3)" for more information).

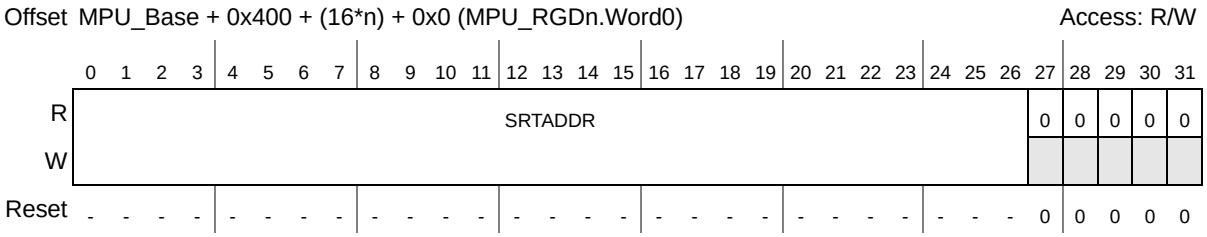



**Figure 28-5. MPU Region Descriptor, Word 0 Register (MPU_RGDn.Word0)**

**Table 28-5. MPU_RGDn.Word0 Field Descriptions**

| Field D | escription |
|---|---|
| 0–26<br>SRTADDR | Start Address. This field defines the most significant bits of the 0-modulo-32 byte start address of the memory region. |

#### 28.2.2.4.2 MPU Region Descriptor n, Word 1 (MPU_RGDn.Word1)

The second word of the MPU region descriptor defines the 31-modulo-32 byte end address of the memory region. Writes to this word clear the region descriptor's valid bit (see Section 28.2.2.4.4, "MPU Region Descriptor n, Word 3 (MPU_RGDn.Word3)" for more information).

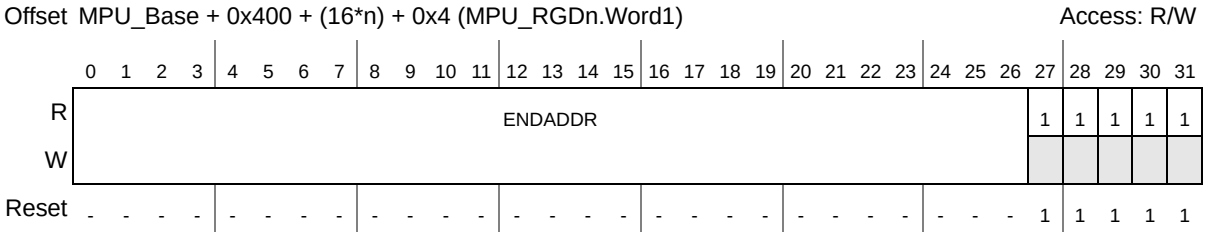



**Figure 28-6. MPU Region Descriptor, Word 1 Register (MPU_RGDn.Word1)**

**Table 28-6. MPU_RGDn.Word1 Field Descriptions**

| Field De | scription |
|---|---|
| 0–26<br>ENDADDR | End Address. This field defines the most significant bits of the 31-modulo-32 byte end address of the memory region. There are no hardware checks to verify that ENDADDR >= SRTADDR; it is software's responsibility to properly load these region descriptor fields. |

### 28.2.2.4.3 MPU Region Descriptor n, Word 2 (MPU_RGDn.Word2)

The third word of the MPU region descriptor defines the access control rights of the memory region. The access control privileges are dependent on two broad classifications of bus masters. Bus masters 0-3 are typically reserved for processor cores and the corresponding access control is a 6-bit field defining separate privilege rights for user and supervisor mode accesses as well as the optional inclusion of a process identification field within the definition. Bus masters 4-7 are typically reserved for data movement engines and their capabilities are limited to separate read and write permissions. For these fields, the bus master number refers to the logical master number defined by the Master IDs listed in Table 9-1.

For the processor privilege rights, there are three flags associated with this function: {read, write, execute}. In this context, these flags follow the traditional definition:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) instruction.
- Execute (x) permission refers to the ability to read the referenced memory address using an instruction fetch.

The evaluation logic defines the processor access type based on multiple AHB signals, as hwrite and hprot[1:0].

For non-processor data movement engines (bus masters 4-7), the evaluation logic simply uses hwrite to determine if the access is a read or write.

Writes to this word clear the region descriptor's valid bit (see Section 28.2.2.4.4, "MPU Region Descriptor n, Word 3 (MPU_RGDn.Word3)" for more information). Since it is also expected that system software may adjust only the access controls within a region descriptor (MPU_RGDn.Word2) as different tasks execute, an alternate programming view of this 32-bit entity is provided. If only the access controls are being updated, this operation should be performed by writing to MPU_RGDAACn (Alternate Access Control n) as stores to these locations do *not* affect the descriptor's valid bit.

Offset MPU_Base + 0x400 + (16*n) + 0x8 (MPU_RGDn.Word2) Access: R/W

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | M7RE | M7WE | M6RE | M6WE | M5RE | M5WE | M4RE | M4WE | M3PE | M3SM | M3UM | | | | M2PE | M2SM | M2UM | | | | M1PE | M1SM | M1UM | | | | M0PE | M0SM | M0UM | | | |
| W | | | | | | | | | | | r | w | x | | | | r | w | x | | | | r | w | x | | | | r | w | x | |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

**Figure 28-7. MPU Region Descriptor, Word 2 Register (MPU_RGDn.Word2)**

**Table 28-7. MPU_RGDn.Word2 Field Descriptions**

| Field De | scription |
|---|---|
| 0<br>M7RE | Bus master 7 read enable. If set, this flag allows bus master 7 to perform read operations. If cleared, any attempted read by bus master 7 terminates with an access error and the read is not performed. |
| 1<br>M7WE | Bus master 7 write enable. If set, this flag allows bus master 7 to perform write operations. If cleared, any attempted write by bus master 7 terminates with an access error and the write is not performed. |
| 2<br>M6RE | Bus master 6 read enable. If set, this flag allows bus master 6 to perform read operations. If cleared, any attempted read by bus master 6 terminates with an access error and the read is not performed. |
| 3<br>M6WE | Bus master 6 write enable. If set, this flag allows bus master 6 to perform write operations. If cleared, any attempted write by bus master 6 terminates with an access error and the write is not performed. |
| 4<br>M5RE | Bus master 5 read enable. If set, this flag allows bus master 5 to perform read operations. If cleared, any attempted read by bus master 5 terminates with an access error and the read is not performed. |
| 5<br>M5WE | Bus master 5 write enable. If set, this flag allows bus master 5 to perform write operations. If cleared, any attempted write by bus master 5 terminates with an access error and the write is not performed. |
| 6<br>M4RE | Bus master 4 read enable. If set, this flag allows bus master 4 to perform read operations. If cleared, any attempted read by bus master 4 terminates with an access error and the read is not performed. |
| 7<br>M4WE | Bus master 4 write enable. If set, this flag allows bus master 4 to perform write operations. If cleared, any attempted write by bus master 4 terminates with an access error and the write is not performed. |
| 8<br>M3PE | Bus master 3 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGDn.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier. |
| 9–10<br>M3SM | Bus master 3 supervisor mode access control. This 2-bit field defines the access controls for bus master 3 when operating in supervisor mode. The M3SM field is defined as:<br>0b00 r, w, x = read, write and execute allowed<br>0b01 r, –, x = read and execute allowed, but no write<br>0b10 r, w, – = read and write allowed, but no execute<br>0b11 Same access controls as that defined by M3UM for user mode |
| 11–13<br>M3UM | Bus master 3 user mode access control. This 3-bit field defines the access controls for bus master 3 when operating in user mode. The M3UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed. |
| 14<br>M2PE | Bus master 2 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGDn.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier. |
| 15–16<br>M2SM | Bus master 2 supervisor mode access control. This 2-bit field defines the access controls for bus master 2 when operating in supervisor mode. The M2SM field is defined as:<br>0b00 r, w, x = read, write and execute allowed<br>0b01 r, –, x = read and execute allowed, but no write<br>0b10 r, w, – = read and write allowed, but no execute<br>0b11 Same access controls as that defined by M2UM for user mode |
| 17–19<br>M2UM | Bus master 2 user mode access control. This 3-bit field defines the access controls for bus master 2 when operating in user mode. The M2UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed. |

**Table 28-7. MPU_RGDn.Word2 Field Descriptions**

| Field De | scription |
|---|---|
| 20<br>M1PE | Bus master 1 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGDn.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier. |
| 21–22<br>M1SM | Bus master 1 supervisor mode access control. This 2-bit field defines the access controls for bus master 1 when operating in supervisor mode. The M1SM field is defined as:<br>0b00 r, w, x = read, write and execute allowed<br>0b01 r, –, x = read and execute allowed, but no write<br>0b10 r, w, – = read and write allowed, but no execute<br>0b11 Same access controls as that defined by M1UM for user mode |
| 23–25<br>M1UM | Bus master 1 user mode access control. This 3-bit field defines the access controls for bus master 1 when operating in user mode. The M1UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed. |
| 26<br>M0PE | Bus master 0 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGDn.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier. |
| 27–28<br>M0SM | Bus master 0 supervisor mode access control. This 2-bit field defines the access controls for bus master 0 when operating in supervisor mode. The M0SM field is defined as:<br>0b00 r, w, x = read, write and execute allowed<br>0b01 r, –, x = read and execute allowed, but no write<br>0b10 r, w, – = read and write allowed, but no execute<br>0b11 Same access controls as that defined by M0UM for user mode |
| 29–31<br>M0UM | Bus master 0 user mode access control. This 3-bit field defines the access controls for bus master 0 when operating in user mode. The M0UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed. |

#### 28.2.2.4.4 MPU Region Descriptor n, Word 3 (MPU_RGDn.Word3)

The fourth word of the MPU region descriptor contains the optional process identifier and mask, plus the region descriptor's valid bit.

Since the region descriptor is a 128-bit entity, there are potential coherency issues as this structure is being updated since multiple writes are required to update the entire descriptor. Accordingly, the MPU hardware assists in the operation of the descriptor valid bit to prevent incoherent region descriptors from generating spurious access errors. In particular, it is expected that a complete update of a region descriptor is typically done with sequential writes to MPU_RGDn.Word0, then MPU_RGDn.Word1,... and finally MPU_RGDn.Word3. The MPU hardware automatically clears the valid bit on any writes to words {0,1,2} of the descriptor. Writes to this word set/clear the valid bit in a normal manner.

Since it is also expected that system software may adjust only the access controls within a region descriptor (MPU_RGDn.Word2) as different tasks execute, an alternate programming view of this 32-bit entity is provided. If only the access controls are being updated, this operation should be performed by writing to MPU_RGDAACn (Alternate Access Control n) as stores to these locations do *not* affect the descriptor's valid bit.

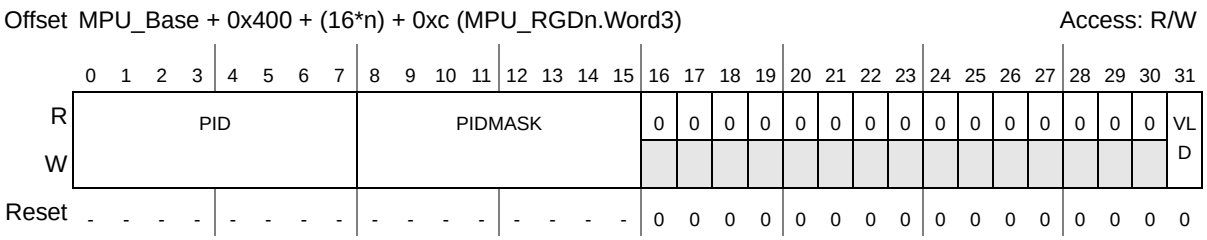



**Figure 28-8. MPU Region Descriptor, Word 3 Register (MPU_RGDn.Word3)**

**Table 28-8. MPU_RGDn.Word3 Field Descriptions**

| Field De | scription |
|---|---|
| 0–7<br>PID | Process Identifier. This 8-bit field specifies that the optional process identifier is to be included in the determination of whether the current access hits in the region descriptor. This field is combined with the PIDMASK and included in the region hit determination if MPU_RGDn.Word2[MxPE] is set. |
| 8–15<br>PIDMASK | Process Identifier Mask. This 8-bit field provides a masking capability so that multiple process identifiers can be included as part of the region hit determination. If a bit in the PIDMASK is set, then the corresponding bit of the PID is ignored in the comparison. This field is combined with the PID and included in the region hit determination if MPU_RGDn.Word2[MxPE] is set. For more information on the handling of the PID and PIDMASK, see Section 28.3.1.1, “Access Evaluation — Hit Determination. |
| 31<br>VLD | Valid. This bit signals the region descriptor is valid. Any write to MPU_RGDn.Word{0,1,2} clears this bit, while a write to MPU_RGDn.Word3 sets or clears this bit depending on bit 31 of the write operand.<br>0 Region descriptor is invalid<br>1 Region descriptor is valid |

## 28.2.2.5 MPU Region Descriptor Alternate Access Control n (MPU_RGDAACn)

As noted in Section 28.2.2.4.3, “MPU Region Descriptor n, Word 2 (MPU_RGDn.Word2)”, it is expected that since system software may adjust only the access controls within a region descriptor (MPU_RGDn.Word2) as different tasks execute, an alternate programming view of this 32-bit entity is desired. If only the access controls are being updated, this operation should be performed by writing to MPU_RGDAACn (Alternate Access Control n) as stores to these locations do not affect the descriptor’s valid bit.

The memory address therefore provides an alternate location for updating MPU_RGDn.Word2.

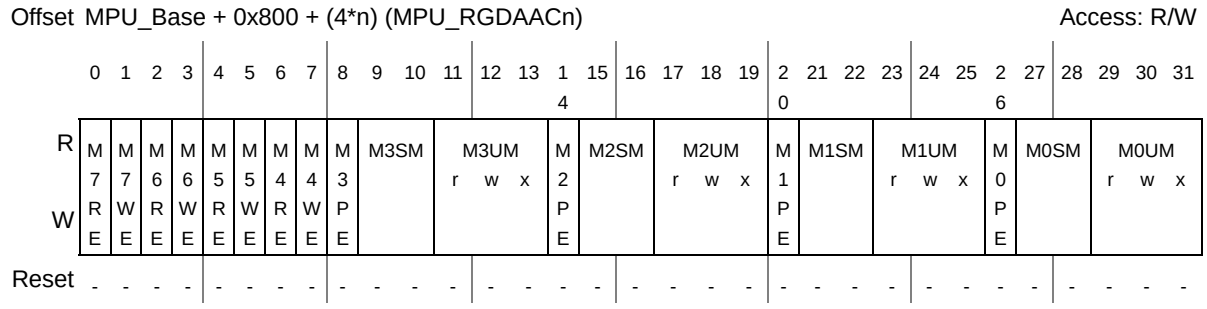



**Figure 28-9. MPU RGD Alternate Access Control n (MPU_RGDAACn)**

Since the MPU_RGDAACn register is simply another memory mapping for MPU_RGDn.Word2, the field definitions shown in Table 28-9 are identical to those presented in Table 28-7.

**Table 28-9. MPU_RGDAACn Field Descriptions**

| Field De | scription |
|---|---|
| 0<br>M7RE | Bus master 7 read enable. If set, this flag allows bus master 7 to perform read operations. If cleared, any attempted read by bus master 7 terminates with an access error and the read is not performed. |
| 1<br>M7WE | Bus master 7 write enable. If set, this flag allows bus master 7 to perform write operations. If cleared, any attempted write by bus master 7 terminates with an access error and the write is not performed. |
| 2<br>M6RE | Bus master 6 read enable. If set, this flag allows bus master 6 to perform read operations. If cleared, any attempted read by bus master 6 terminates with an access error and the read is not performed. |
| 3<br>M6WE | Bus master 6 write enable. If set, this flag allows bus master 6 to perform write operations. If cleared, any attempted write by bus master 6 terminates with an access error and the write is not performed. |
| 4<br>M5RE | Bus master 5 read enable. If set, this flag allows bus master 5 to perform read operations. If cleared, any attempted read by bus master 5 terminates with an access error and the read is not performed. |
| 5<br>M5WE | Bus master 5 write enable. If set, this flag allows bus master 5 to perform write operations. If cleared, any attempted write by bus master 5 terminates with an access error and the write is not performed. |
| 6<br>M4RE | Bus master 4 read enable. If set, this flag allows bus master 4 to perform read operations. If cleared, any attempted read by bus master 4 terminates with an access error and the read is not performed. |
| 7<br>M4WE | Bus master 4 write enable. If set, this flag allows bus master 4 to perform write operations. If cleared, any attempted write by bus master 4 terminates with an access error and the write is not performed. |
| 8<br>M3PE | Bus master 3 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGDn.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier. |
| 9–10<br>M3SM | Bus master 3 supervisor mode access control. This 2-bit field defines the access controls for bus master 3 when operating in supervisor mode. The M3SM field is defined as:<br>0b00 r, w, x = read, write and execute allowed<br>0b01 r, –, x = read and execute allowed, but no write<br>0b10 r, w, – = read and write allowed, but no execute<br>0b11 Same access controls as that defined by M3UM for user mode |
| 11–13<br>M3UM | Bus master 3 user mode access control. This 3-bit field defines the access controls for bus master 3 when operating in user mode. The M3UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed. |
| 14<br>M2PE | Bus master 2 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGDn.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier. |
| 15–16<br>M2SM | Bus master 2 supervisor mode access control. This 2-bit field defines the access controls for bus master 2 when operating in supervisor mode. The M2SM field is defined as:<br>0b00 r, w, x = read, write and execute allowed<br>0b01 r, –, x = read and execute allowed, but no write<br>0b10 r, w, – = read and write allowed, but no execute<br>0b11 Same access controls as that defined by M2UM for user mode |

Table 28-9. MPU_RGDAACn Field Descriptions (continued)

| Field De | scription |
|---|---|
| 17–19<br>M2UM | Bus master 2 user mode access control. This 3-bit field defines the access controls for bus master 2 when operating in user mode. The M2UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed. |
| 20<br>M1PE | Bus master 1 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGDn.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier. |
| 21–22<br>M1SM | Bus master 1 supervisor mode access control. This 2-bit field defines the access controls for bus master 1 when operating in supervisor mode. The M1SM field is defined as:<br>0b00 r, w, x = read, write and execute allowed<br>0b01 r, –, x = read and execute allowed, but no write<br>0b10 r, w, – = read and write allowed, but no execute<br>0b11 Same access controls as that defined by M1UM for user mode |
| 23–25<br>M1UM | Bus master 1 user mode access control. This 3-bit field defines the access controls for bus master 1 when operating in user mode. The M1UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed. |
| 26<br>M0PE | Bus master 0 process identifier enable. If set, this flag specifies that the process identifier and mask (defined in MPU_RGDn.Word3) are to be included in the region hit evaluation. If cleared, then the region hit evaluation does not include the process identifier. |
| 27–28<br>M0SM | Bus master 0 supervisor mode access control. This 2-bit field defines the access controls for bus master 0 when operating in supervisor mode. The M0SM field is defined as:<br>0b00 r, w, x = read, write and execute allowed<br>0b01 r, –, x = read and execute allowed, but no write<br>0b10 r, w, – = read and write allowed, but no execute<br>0b11 Same access controls as that defined by M0UM for user mode |
| 29–31<br>M0UM | Bus master 0 user mode access control. This 3-bit field defines the access controls for bus master 0 when operating in user mode. The M0UM field consists of three independent bits, enabling read, write and execute permissions: {r,w,x}. If set, the bit allows the given access type to occur; if cleared, an attempted access of that mode may be terminated with an access error (if not allowed by any other descriptor) and the access not performed. |

## 28.3 Functional description

In this section, the functional operation of the MPU is detailed. In particular, subsequent sections discuss the operation of the access evaluation macro as well as the handling of error-terminated AHB bus cycles.

### 28.3.1 Access evaluation macro

As previously discussed, the basic operation of the MPU is performed in the access evaluation macro, a hardware structure replicated in the two-dimensional connection matrix. As shown in Figure 28-10, the access evaluation macro inputs the AHB system bus address phase signals (AHB_ap) and the contents of a region descriptor (RGDn) and performs two major functions: region hit determination (hit_b) and detection of an access protection violation (error).

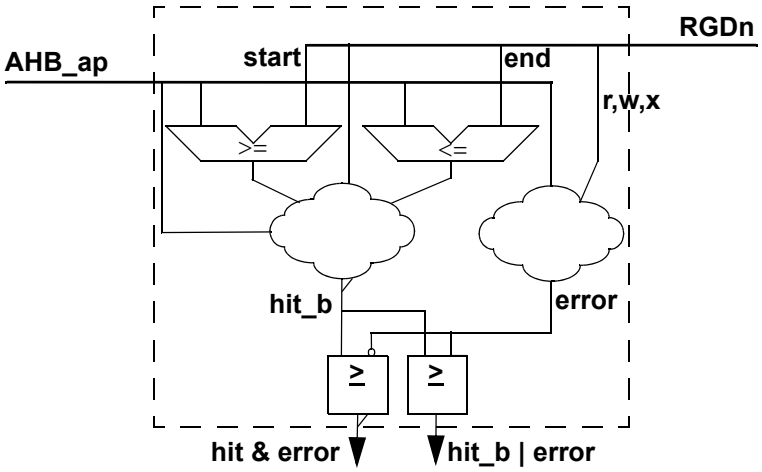



**Figure 28-10. MPU Access Evaluation Macro**

Figure 28-10 is not intended to be a schematic of the actual access evaluation macro, but rather a generalized block diagram showing the major functions included in this logic block.

### 28.3.1.1 Access Evaluation — Hit Determination

To evaluate the region hit determination, the MPU uses two magnitude comparators in conjunction with the contents of a region descriptor: the current access must be included between the region's "start" and "end" addresses and simultaneously the region's valid bit must be active.

Recall there are no hardware checks to verify that region's "end" address is greater then region's "start" address, and it is software’s responsibility to properly load appropriate values into these fields of the region descriptor.

In addition to this, the optional process identifier is examined against the region descriptor’s PID and PIDMASK fields. In order to generate the pid_hit indication: the current PID with its PIDMASK must be equal to the region's PID with its PIDMASK. Also the process identifier enable is take into account in this comparison so that the MPU forces the pid_hit term to be asserted in the case of AHB bus master does not provide its process identifier.

### 28.3.1.2 Access Evaluation — Privilege Violation Determination

While the access evaluation macro is making the region hit determination, the logic is also evaluating if the current access is allowed by the permissions defined in the region descriptor. The protection violation

logic then evaluates the access against the effective permissions using the specification shown in Table 28-10.

**Table 28-10. Protection Violation Definition**

| Description | Inputs O | | | utput |
|---|---|---|---|---|
| | eff_rgd[r] | eff_rgd[w] | eff_rgd[x] | Protection Violation? |
| inst fetch read | - | - | 0 | yes, no x permission |
| inst fetch read | - | - | 1 | no, access is allowed |
| data read | 0 | - | - | yes, no r permission |
| data read | 1 | - | - | no, access is allowed |
| data write | - | 0 | - | yes, no w permission |
| data write | - | 1 | - | no, access is allowed |

As shown in Figure 28-10, the output of the protection violation logic is the error signal.

The access evaluation macro then uses the hit_b and error signals to form two outputs. The combined (hit_b | error) signal is used to signal the current access is not allowed and (~hit_b & error) is used as the input to MPU_EDRn (error detail register) in the event of an error.

## 28.3.2 Putting It All Together and AHB Error Terminations

For each AHB slave port being monitored, the MPU performs a reduction-AND of all the individual (hit_b | error) terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports a protection error for three conditions:

1. If the access does not hit in any region descriptor, a protection error is reported.
2. If the access hits in a single region descriptor and that region signals a protection violation, then a protection error is reported.
3. If the access hits in multiple (overlapping) regions and all regions signal protection violations, then a protection error is reported.

The third condition reflects that priority is given to permission granting over access denying for overlapping regions as this approach provides more flexibility to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see Section 28.6, “Application information.

In event of a protection error, the MPU requires two distinct actions:

1. intercepts the error during the AHB address phase (first cycle out of two) and cancels the transaction before it is seen by the slave device.
2. performs the required logic functions to force the standard 2-cycle AHB error response to properly terminate the bus transaction and then provides the right values to the crossbar switch to commit the AHB transaction to other portions of the platform.

If instead the access is allowed, then the MPU simply passes all "original" AHB signals to the slave device. In this case, from functionality point of view, the MPU is fully transparent.

## 28.4 Initialization information

The reset state of MPU_CESR[VLD] disables the entire module. Recall while the MPU is disabled, all accesses from all bus masters are allowed. This state also minimizes the power dissipation of the MPU. The power dissipation of each access evaluation macro is minimized when the associated region descriptor is marked as invalid or when MPU_CESR[VLD] = 0.

Typically the appropriate number of region descriptors (MPU_RGDn) are loaded at system startup, including the setting of the MPU_RGDn.Word3[VLD] bits, before MPU_CESR[VLD] is set, enabling the module. This approach allows all the loaded region descriptors to be enabled simultaneously. Recall if a memory reference does not hit in any region descriptor, the attempted access is terminated with an error.

## 28.5 Opcode pre-fetch cycles and the execute permission

The CPU pre-fetches program-code past the current instruction to optimize performance. The code that is pre-fetched may never be executed or even be reachable (in the case of a branch), however the MPU module has no way of knowing this at the time when the pre-fetch cycles occur. Therefore such pre-fetches will result in an access violation if the opcode pre-fetch accesses a memory range in which the "x" execute access mode is not permitted. This must be taken into account when defining memory ranges without execute permission adjacent to memory used for program code. The best way to do this would be to leave some fill-bytes between the memory ranges in this case — that is, do not set the upper memory boundary to the address of the last opcode but to a following address which is several words away.

## 28.6 Application information

In an operational system, interfacing with the MPU can generally be classified into the following activities:

1. Creation of a new memory region requires loading the appropriate region descriptor into an available register location. When a new descriptor is loaded into a RGDn, it would typically be performed using four 32-bit word writes. As discussed in Section 28.2.2.4.4, “MPU Region Descriptor n, Word 3 (MPU_RGDn.Word3)”, the hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues associated with the multi-cycle descriptor writes. Deletion/removal of an existing memory region is performed simply by clearing MPU_RGDn.Word3[VLD].
2. If only the access rights for an existing region descriptor need to change, a 32-bit write to the alternate version of the access control word (MPU_RGDAACn) would typically be performed. Recall writes to the region descriptor using this alternate access control location do not affect the valid bit, so there are, by definition, no coherency issues involved with the update. The access rights associated with the memory region switch instantaneously to the new value as the IPS write completes.

3. If the region’s start and end addresses are to be changed, this would typically be performed by writing a minimum of three words of the region descriptor: MPU_RGDn.Word{0,1,3}, where the writes to Word0 and Word1 redefine the start and end addresses respectively and the write to Word3 re-enables the region descriptor valid bit. In many situations, all four words of the region descriptor would be rewritten.
4. Typically, references to the MPU’s programming model would be restricted to supervisor mode accesses from a specific processor(s), so a region descriptor would be specifically allocated for this purpose with attempted accesses from other masters or while in user mode terminated with an error.
5. When the MPU detects an access error, the current AHB bus cycle is terminated with an error response and information on the faulting reference captured in the MPU_EARn and MPU_EDRn registers. The error-terminated AHB bus cycle typically initiates some type of error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. In any event, the processor can retrieve the captured error address and detail information simply be reading the MPU_E{A,D}Rn registers. Information on which error registers contain captured fault data is signaled by MPU_CESR[SPERR].

This page is intentionally left blank.

# Chapter 29
# Mode Entry Module (MC_ME)

## 29.1 Introduction

### 29.1.1 Overview

The MC_ME controls the SoC mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

Figure 29-1 depicts the MC_ME block diagram.

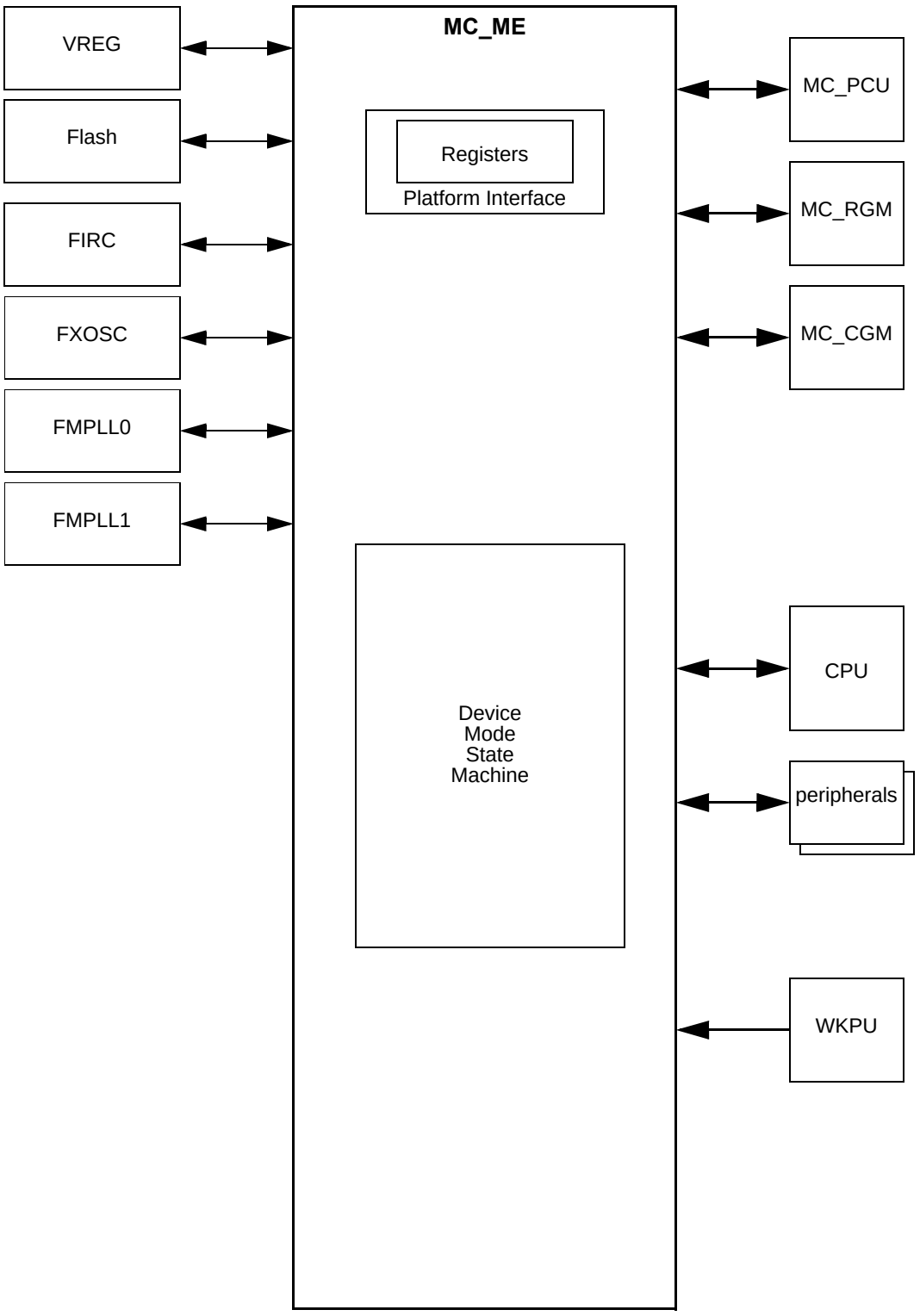



**Figure 29-1. MC_ME Block Diagram**

## 29.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the **ME_ME** register
- definition of various device mode configurations by the **ME_<*mode*>_MC** registers
- control of the actual device mode by the **ME_MCTL** register
- capture of the current mode and various resource status within the contents of the **ME_GS** register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the **ME_RUN_PC0…7**, **ME_LP_PC0…7**, and **ME_PCTL0…143** registers
- capture of current peripheral clock gated/enabled status

## 29.1.3 Modes of Operation

The MC_ME is based on several device modes corresponding to different usage models of the device. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as ***RESET***, ***DRUN***, ***SAFE***, and ***TEST***. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as ***RUN0…3***, ***HALT***, ***STOP***, and ***STANDBY*** which can be configured to meet the application requirements in terms of energy management and available processing power. The modes ***DRUN***, ***SAFE***, ***TEST***, and ***RUN0…3*** are the device software running modes.

Table 29-1 describes the MC_ME modes.

**Table 29-1. MC_ME Mode Descriptions**

| Name | Description | Entry | Exit |
|---|---|---|---|
| ***RESET*** | This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the device. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and flash modules. | system reset assertion from MC_RGM | system reset deassertion from MC_RGM |
| ***DRUN*** | This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes. BAM when present is executed in ***DRUN*** mode. | system reset deassertion from MC_RGM, software request from ***SAFE***, ***TEST*** and ***RUN0…3***, wakeup request from ***STANDBY*** | system reset assertion, ***RUN0…3***, ***TEST***, ***STANDBY*** via software, ***SAFE*** via software or hardware failure. |

Table 29-1. MC_ME Mode Descriptions (continued)

| Name | Description | Entry | Exit |
|---|---|---|---|
| ***SAFE*** | This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover. | hardware failure, software request from ***DRUN***, ***TEST***, and ***RUN0…3*** | system reset assertion, ***DRUN*** via software |
| ***TEST*** | This is a chip-wide service mode which is intended to provide a control environment for device software teting. | software request from ***DRUN*** | system reset assertion, ***DRUN*** via software |
| ***RUN0…3*** | These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other. | software request from ***DRUN*** or other ***RUN0…3***, interrupt event from ***HALT,*** interrupt or wakeup event from ***STOP*** | system reset assertion, ***SAFE*** via software or hardware failure, other ***RUN0…3*** modes, ***HALT***, ***STOP***, ***STANDBY*** via software |
| ***HALT*** | This is a reduced-activity low-power mode during which the clock to the CPU is disabled. It can be configured to switch off analog peripherals like clock sources, flash, main regulator, etc. for efficient power management at the cost of higher wakeup latency. | software request from ***RUN0…3*** | system reset assertion, ***SAFE*** on hardware failure, ***RUN0…3*** on interrupt event |
| ***STOP*** | This is an advanced low-power mode during which the clock to the CPU is disabled. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency. | software request from ***RUN0…3*** | system reset assertion, ***SAFE*** on hardware failure, ***RUN0…3*** on interrupt event or wakeup event |
| ***STANDBY*** | This is a reduced-leakage low-power mode during which power supply is cut off from most of the device. Wakeup from this mode takes a relatively long time, and content is lost or must be restored from backup. | software request from ***RUN0…3***, ***DRUN*** modes | system reset assertion, ***DRUN*** on wakeup event |

## 29.2 External Signal Description

The MC_ME has no connections to any external pins.

## 29.3 Memory Map and Register Definition

The MC_ME contains registers for:

- mode selection and status reporting
- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting

## 29.3.1 Memory map

**Table 29-2. MC_ME register description**

| Address | Name | Description | Size | Access | | Location |
|---|---|---|---|---|---|---|
| | | | | User | Supervisor | |
| 0xC3FD_C000 | ME_GS | Global Status | word | read | read | on page 1077 |
| 0xC3FD_C004 | ME_MCTL | Mode Control | word | read | read/write | on page 1079 |
| 0xC3FD_C008 | ME_ME | Mode Enable | word | read | read/write | on page 1080 |
| 0xC3FD_C00C | ME_IS | Interrupt Status | word | read | read/write | on page 1081 |
| 0xC3FD_C010 | ME_IM | Interrupt Mask | word | read | read/write | on page 1082 |
| 0xC3FD_C014 | ME_IMTS | Invalid Mode Transition Status | word | read | read/write | on page 1083 |
| 0xC3FD_C018 | ME_DMTS | Debug Mode Transition Status | word | read | read | on page 1084 |
| 0xC3FD_C020 | ME_RESET_MC | ***RESET*** Mode Configuration | word | read | read | on page 1087 |
| 0xC3FD_C024 | ME_TEST_MC | ***TEST*** Mode Configuration | word | read | read/write | on page 1087 |
| 0xC3FD_C028 | ME_SAFE_MC | ***SAFE*** Mode Configuration | word | read | read/write | on page 1088 |
| 0xC3FD_C02C | ME_DRUN_MC | ***DRUN*** Mode Configuration | word | read | read/write | on page 1088 |
| 0xC3FD_C030 | ME_RUN0_MC | ***RUN0*** Mode Configuration | word | read | read/write | on page 1089 |
| 0xC3FD_C034 | ME_RUN1_MC | ***RUN1*** Mode Configuration | word | read | read/write | on page 1089 |
| 0xC3FD_C038 | ME_RUN2_MC | ***RUN2*** Mode Configuration | word | read | read/write | on page 1089 |
| 0xC3FD_C03C | ME_RUN3_MC | ***RUN3*** Mode Configuration | word | read | read/write | on page 1089 |
| 0xC3FD_C040 | ME_HALT_MC | ***HALT*** Mode Configuration | word | read | read/write | on page 1089 |
| 0xC3FD_C048 | ME_STOP_MC | ***STOP*** Mode Configuration | word | read | read/write | on page 1090 |
| 0xC3FD_C054 | ME_STANDBY_MC | ***STANDBY*** Mode Configuration | word | read | read/write | on page 1090 |
| 0xC3FD_C060 | ME_PS0 | Peripheral Status 0 | word | read | read | on page 1092 |

**Table 29-2. MC_ME register description (continued)**

| Address | Name | Description | Size | Access | | Location |
|---|---|---|---|---|---|---|
| | | | | User | Supervisor | |
| 0xC3FD_C064 | ME_PS1 | Peripheral Status 1 | word | read | read | on page 1092 |
| 0xC3FD_C068 | ME_PS2 | Peripheral Status 2 | word | read | read | on page 1093 |
| 0xC3FD_C06C | ME_PS3 | Peripheral Status 3 | word | read | read | on page 1093 |
| 0xC3FD_C080 | ME_RUN_PC0 | Run Peripheral Configuration 0 | word | read | read/write | on page 1094 |
| 0xC3FD_C084 | ME_RUN_PC1 | Run Peripheral Configuration 1 | word | read | read/write | on page 1094 |
| ... | | | | | | |
| 0xC3FD_C09C | ME_RUN_PC7 | Run Peripheral Configuration 7 | word | read | read/write | on page 1094 |
| 0xC3FD_C0A0 | ME_LP_PC0 | Low-Power Peripheral Configuration 0 | word | read | read/write | on page 1095 |
| 0xC3FD_C0A4 | ME_LP_PC1 | Low-Power Peripheral Configuration 1 | word | read | read/write | on page 1095 |
| ... | | | | | | |
| 0xC3FD_C0BC | ME_LP_PC7 | Low-Power Peripheral Configuration 7 | word | read | read/write | on page 1095 |
| 0xC3FD_C0C4 | ME_PCTL4 | DSPI0 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0C5 | ME_PCTL5 | DSPI1 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0C6 | ME_PCTL6 | DSPI2 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0C8 | ME_PCTL8 | QUADSPI Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0D0 | ME_PCTL16 | FLEXCAN0 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0D1 | ME_PCTL17 | FLEXCAN1 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0D2 | ME_PCTL18 | FLEXCAN2 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0D7 | ME_PCTL23 | DMA_CH_MUX Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0DF | ME_PCTL31 | BAM Control | byte | read | read/write | on page 1095 |

**Table 29-2. MC_ME register description (continued)**

| Address | Name | Description | Size | Access | | Location |
|---|---|---|---|---|---|---|
| | | | | User | Supervisor | |
| 0xC3FD_C0E0 | ME_PCTL32 | ADC0 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0E5 | ME_PCTL37 | RLE Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0E9 | ME_PCTL41 | VIU Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0EA | ME_PCTL42 | DRAM Controller Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0EC | ME_PCTL44 | I2C_DMA0 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0ED | ME_PCTL45 | I2C_DMA1 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0EE | ME_PCTL46 | I2C_DMA2 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0EF | ME_PCTL47 | I2C_DMA3 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0F0 | ME_PCTL48 | LINFLEX0 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0F1 | ME_PCTL49 | LINFLEX1 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0F2 | ME_PCTL50 | LINFLEX2 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0F3 | ME_PCTL51 | LINFLEX3 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0F4 | ME_PCTL52 | GFX2D Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0F5 | ME_PCTL53 | GXG Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0F6 | ME_PCTL54 | DCULITE Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0F7 | ME_PCTL55 | DCU3 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0F8 | ME_PCTL56 | GAUGEDRIVER Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0FC | ME_PCTL60 | CANSAMPLER Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C0FE | ME_PCTL62 | SGM Control | byte | read | read/write | on page 1095 |

**Table 29-2. MC_ME register description (continued)**

| Address | Name | Description | Size | Access | | Location |
|---|---|---|---|---|---|---|
| | | | | User | Supervisor | |
| 0xC3FD_C0FF | ME_PCTL63 | TCON Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C102 | ME_PCTL66 | CFLASH0 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C104 | ME_PCTL68 | SIUL Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C108 | ME_PCTL72 | eMIOS0 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C109 | ME_PCTL73 | eMIOS1 Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C11B | ME_PCTL91 | RTC_API Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C11C | ME_PCTL92 | PIT_RTI Control | byte | read | read/write | on page 1095 |
| 0xC3FD_C128 | ME_PCTL104 | CMU0 Control | byte | read | read/write | on page 1095 |

## NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

**Table 29-3. MC_ME Memory Map**

| Address | Name | | 0 | 1 | 2 | 3 | 27 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FD_C000 | ME_GS | R | S_CURRENT_MODE | | | | S_MTRANS | S_DC | 0 | 0 | S_PDO | 0 | 0 | S_MVR | | | S_FLA | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S_FMPLL1 | S_FMPLL0 | S_FXOSC | S_FIRC | S_SYSCLK | | | |
| | | W | | | | | | | | | | | | | | | | |

**Table 29-3. MC_ME Memory Map (continued)**

| Address | Name | | 0 | 1 | 2 | 3 | 27 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FD_C004 | ME_MCTL | R | TARGET_MODE | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | W | KEY | | | | | | | | | | | | | | | |
| 0xC3FD_C008 | ME_ME | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | STANDBY | 0 | 0 | STOP | 0 | HALT | RUN3 | RUN2 | RUN1 | RUN0 | DRUN | SAFE | TEST | RESET |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C00C | ME_IS | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | I_ICONF | I_IMODE | I_SAFE | I_MTC |
| | | W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| 0xC3FD_C010 | ME_IM | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | M_ICONF | M_IMODE | M_SAFE | M_MTC |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C014 | ME_IMTS | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S_MTI | S_MRI | S_DMA | S_NMA | S_SEA |
| | | W | | | | | | | | | | | | w1c | w1c | w1c | w1c | w1c |

**Table 29-3. MC_ME Memory Map (continued)**

| Address | Name | | 0 | 1 | 2 | 3 | 27 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FD_C018 | ME_DMTS | R | PREVIOUS_MODE | | | | 0 | 0 | 0 | 0 | MPH_BUSY | 0 | 0 | PMC_PROG | CORE_DBG | 0 | 0 | SMR |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | VREG_CSRC_SC | CSRC_CSRC_SC | FIRC_SC | SCSRC_SC | SYSCLK_SW | | CFLASH_SC | CDP_PRPH_0_143 | 0 | 0 | | CDP_PRPH_96_127 | CDP_PRPH_64_95 | CDP_PRPH_32_63 | CDP_PRPH_0_31 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C01C | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FD_C020 | ME_RESET_MC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | | | FLAON | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C024 | ME_TEST_MC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | | | FLAON | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| | | W | | | | | | | | | | | | | | | | |

**Table 29-3. MC_ME Memory Map (continued)**

| Address | Name | | 0<br>16 | 1<br>17 | 2<br>18 | 3<br>19 | 27<br>20 | 5<br>21 | 6<br>22 | 7<br>23 | 8<br>24 | 9<br>25 | 10<br>26 | 11<br>27 | 12<br>28 | 13<br>29 | 14<br>30 | 15<br>31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xC3FD_C028 | ME_SAFE_MC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | | | FLAON | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C02C | ME_DRUN_MC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | | | FLAON | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C030<br>…<br>0xC3FD_C03C | ME_RUN0…3_MC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | | | FLAON | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C040 | ME_HALT_MC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | | | FLAON | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C044 | reserved | | | | | | | | | | | | | | | | | |

**Table 29-3. MC_ME Memory Map (continued)**

| Address | Name | | 0 | 1 | 2 | 3 | 27 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FD_C048 | ME_STOP_MC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | | | FLAON | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C04C … 0xC3FD_C050 | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FD_C054 | ME_STANDBY_MC | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | | | FLAON | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C058 … 0xC3FD_C05C | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FD_C060 | ME_PS0 | R | S_BAM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S_DMA_CH_MUX | 0 | 0 | 0 | 0 | S_FLEXCAN2 | S_FLEXCAN1 | S_FLEXCAN0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S_QUADSPI | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |

**Table 29-3. MC_ME Memory Map (continued)**

| Address | Name | | 0 | 1 | 2 | 3 | 27 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FD_C064 | ME_PS1 | R | S_TCON | S_SGM | 0 | 0 | 0 | 0 | 0 | S_GAUGEDRIVER | S_DCU3 | S_DCULITE | S_GXG | S_GFX2D | S_LINFLEX3 | S_LINFLEX2 | S_LINFLEX1 | S_LINFLEX0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | S_I2C_DMA3 | S_I2C_DMA2 | S_I2C_DMA1 | S_I2C_DMA0 | 0 | S_DRAM Controller | S_VIU | 0 | 0 | 0 | S_RLE | 0 | 0 | 0 | 0 | S_ADC0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C068 | ME_PS2 | R | 0 | 0 | 0 | S_PIT_RTI | S_RTC_API | S_MC_PCU | S_MC_RGM | S_MC_CGM | S_MC_ME | S_SSCM | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | S_EMIOS1 | S_EMIOS0 | 0 | 0 | S_WKPU | S_SIUL | 0 | S_CFLASH0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C06C | ME_PS3 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S_CMU0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C070 | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FD_C074 … 0xC3FD_C07C | reserved | | | | | | | | | | | | | | | | | |

**Table 29-3. MC_ME Memory Map (continued)**

| Address | Name | | 0 16 | 1 17 | 2 18 | 3 19 | 27 20 | 5 21 | 6 22 | 7 23 | 8 24 | 9 25 | 10 26 | 11 27 | 12 28 | 13 29 | 14 30 | 15 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xC3FD_C080 … 0xC3FD_C09C | ME_RUN_PC0…7 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RUN3 | RUN2 | RUN1 | RUN0 | DRUN | SAFE | TEST | RESET |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C0A0 … 0xC3FD_C0BC | ME_LP_PC0…7 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | STANDBY | 0 | 0 | STOP | 0 | HALT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C0C0 … 0xC3FD_C14C | ME_PCTL0…143 | R | 0 | DBG_F | LP_CFG | | | RUN_CFG | | | 0 | DBG_F | LP_CFG | | | RUN_CFG | | |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | DBG_F | LP_CFG | | | RUN_CFG | | | 0 | DBG_F | LP_CFG | | | RUN_CFG | | |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FD_C150 … 0xC3FD_FFFC | reserved | | | | | | | | | | | | | | | | | |

## 29.3.2 Register description

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the **ME_RUN_PC0** register may be accessed as a word at address 0xC3FD_C080, as a half-word at address 0xC3FD_C082, or as a byte at address 0xC3FD_C083.

### 29.3.2.1 Global Status Register (ME_GS)

Address 0xC3FD_C000 Access: User read, Supervisor read, Test read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | S_CURRENT_MODE | | | | S_MTRANS | S_DC | 0 | 0 | S_PDO | 0 | 0 | S_MVR | | | S_FLA | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S_FMPLL1 | S_FMPLL0 | S_FXOSC | S_FIRC | S_SYSCLK | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Figure 29-2. Global Status Register (ME_GS)**

This register contains global mode status.

**Table 29-4. Global Status Register (ME_GS) Field Descriptions**

| Field | Description |
|---|---|
| S_CURRENT_MODE | **Current device mode status**<br>0000 ***RESET***<br>0001 ***TEST***<br>0010 ***SAFE***<br>0011 ***DRUN***<br>0100 ***RUN0***<br>0101 ***RUN1***<br>0110 ***RUN2***<br>0111 ***RUN3***<br>1000 ***HALT***<br>1001 reserved<br>1010 ***STOP***<br>1011 reserved<br>1100 reserved<br>1101 ***STANDBY***<br>1110 reserved<br>1111 reserved |
| S_MTRANS | **Mode transition status**<br>0 Mode transition process is not active<br>1 Mode transition is ongoing |
| S_DC | **Device current consumption status**<br>0 Device consumption is low enough to allow powering down of main voltage regulator<br>1 Device consumption requires main voltage regulator to remain powered regardless of mode configuration |

**Table 29-4. Global Status Register (ME_GS) Field Descriptions (continued)**

| Field | Description |
|---|---|
| S_PDO | **Output power-down status** — This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off.<br>0 No automatic safe gating of I/Os used and pads power sequence driver is enabled<br>1 In ***SAFE***/***TEST*** modes, outputs of pads are forced to high impedance state and the pads power sequence driver is disabled. The inputs are level unchanged. In ***STOP*** mode, only the pad power sequence driver is disabled, but the state of the output remains functional. In ***STANDBY*** mode, the power sequence driver and all pads except those mapped on wakeup lines are not powered and therefore high impedance. Wakeup lines configuration remains unchanged |
| S_MVR | **Main voltage regulator status**<br>0 Main voltage regulator is not ready<br>1 Main voltage regulator is ready for use |
| S_FLA | Flash memory **availability status**<br>00 Flash memory is not available<br>01 Flash memory is in power-down mode<br>10 Flash memory is in low-power mode<br>11 Flash memory is in normal mode and available for use |
| S_SSCLK1 | **Secondary system clock source 1 status**<br>0 Secondary system clock source 1 is not stable<br>1 Secondary system clock source 1 is providing a stable clock |
| S_FMPLL1 | **secondary frequency modulated phase locked loop status**<br>0 secondary frequency modulated phase locked loop is not stable<br>1 secondary frequency modulated phase locked loop is providing a stable clock |
| S_FMPLL0 | **primary frequency modulated phase locked loop status**<br>0 primary frequency modulated phase locked loop is not stable<br>1 primary frequency modulated phase locked loop is providing a stable clock |
| S_FXOSC | **fast external crystal oscillator (4-16 MHz) status**<br>0 fast external crystal oscillator (4-16 MHz) is not stable<br>1 fast external crystal oscillator (4-16 MHz) is providing a stable clock |
| S_FIRC | **fast internal RC oscillator (16 MHz) status**<br>0 fast internal RC oscillator (16 MHz) is not stable<br>1 fast internal RC oscillator (16 MHz) is providing a stable clock |
| S_SYSCLK | **System clock switch status** — These bits specify the system clock currently used by the system.<br>0000 16 MHz int. RC osc.<br>0001 reserved<br>0010 reserved<br>0011 div. 4-16 MHz ext. xtal osc.<br>0100 primary PLL/2<br>0101 reserved<br>0110 reserved<br>0111 reserved<br>1000 reserved<br>1001 reserved<br>1010 reserved<br>1011 reserved<br>1100 reserved<br>1101 reserved<br>1110 reserved<br>1111 system clock is disabled |

### 29.3.2.2 Mode Control Register (ME_MCTL)

Address 0xC3FD_C004 Access: User read, Supervisor read/write, Test read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| W | KEY | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

**Figure 29-3. Mode Control Register (ME_MCTL)**

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by **ME_ME** register bits, configurations corresponding to unavailable modes are reserved and access to **ME_*<mode>*_MC** registers must respect this for successful mode requests.

**NOTE**

Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.

**Table 29-5. Mode Control Register (ME_MCTL) Field Descriptions**

| Field | Description |
|---|---|
| TARGET_MODE | **Target device mode** — These bits provide the target device mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering ***SAFE*** on hardware request. Also, while exiting from the ***HALT*** and ***STOP*** modes on hardware exit events, these are updated with the appropriate ***RUN0...3*** mode value.<br>0000 ***RESET***<br>0001 ***TEST***<br>0010 ***SAFE***<br>0011 ***DRUN***<br>0100 ***RUN0***<br>0101 ***RUN1***<br>0110 ***RUN2***<br>0111 ***RUN3***<br>1000 ***HALT***<br>1001 reserved<br>1010 ***STOP***<br>1011 reserved<br>1100 reserved<br>1101 ***STANDBY***<br>1110 reserved<br>1111 ***RESET (destructive)*** |
| KEY | **Control key** — These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key.<br>KEY:0101101011110000 (0x5AF0)<br>INVERTED KEY:1010010100001111 (0xA50F) |

### 29.3.2.3 Mode Enable Register (ME_ME)

Address 0xC3FD_C008 Access: User read, Supervisor read/write, Test read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | STANDBY | 0 | 0 | STOP | 0 | HALT | RUN3 | RUN2 | RUN1 | RUN0 | DRUN | SAFE | TEST | RESET |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**Figure 29-4. Mode Enable Register (ME_ME)**

This register allows a way to disable the device modes which are not required for a given device. ***RESET***, ***SAFE***, ***DRUN***, and ***RUN0*** modes are always enabled.

**Table 29-6. Mode Enable Register (ME_ME) Field Descriptions**

| Field | Description |
|---|---|
| STANDBY | ***STANDBY* mode enable**<br>0 ***STANDBY*** mode is disabled<br>1 ***STANDBY*** mode is enabled |
| STOP | ***STOP* mode enable**<br>0 ***STOP*** mode is disabled<br>1 ***STOP*** mode is enabled |
| HALT | ***HALT* mode enable**<br>0 ***HALT*** mode is disabled<br>1 ***HALT*** mode is enabled |
| RUN3 | ***RUN3* mode enable**<br>0 ***RUN3*** mode is disabled<br>1 ***RUN3*** mode is enabled |
| RUN2 | ***RUN2* mode enable**<br>0 ***RUN2*** mode is disabled<br>1 ***RUN2*** mode is enabled |
| RUN1 | ***RUN1* mode enable**<br>0 ***RUN1*** mode is disabled<br>1 ***RUN1*** mode is enabled |
| RUN0 | ***RUN0* mode enable**<br>0 ***RUN0*** mode is disabled<br>1 ***RUN0*** mode is enabled |
| DRUN | ***DRUN* mode enable**<br>0 ***DRUN*** mode is disabled<br>1 ***DRUN*** mode is enabled |
| SAFE | ***SAFE* mode enable**<br>0 ***SAFE*** mode is disabled<br>1 ***SAFE*** mode is enabled |

**Table 29-6. Mode Enable Register (ME_ME) Field Descriptions (continued)**

| Field | Description |
|---|---|
| TEST | ***TEST* mode enable**<br>0 ***TEST*** mode is disabled<br>1 ***TEST*** mode is enabled |
| RESET | ***RESET* mode enable**<br>0 ***RESET*** mode is disabled<br>1 ***RESET*** mode is enabled |

### 29.3.2.4 Interrupt Status Register (ME_IS)

Address 0xC3FD_C00C Access: User read, Supervisor read/write, Test read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | I_ICONF | I_IMODE | I_SAFE | I_MTC |
| W | | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-5. Interrupt Status Register (ME_IS)**

This register provides the current interrupt status.

**Table 29-7. Interrupt Status Register (ME_IS) Field Descriptions**

| Field | Description |
|---|---|
| I_ICONF_CU | Invalid mode configuration interrupt (Clock Usage) — This bit is set during a mode transition if a clock which is required to be on by an enabled peripheral is configured to be turned off. It is cleared by writing a ‘1’ to this bit.<br>0 No invalid mode configuration (clock usage) interrupt occurred<br>1 Invalid mode configuration (clock usage) interrupt is pending |
| I_ICONF | **Invalid mode configuration interrupt** — This bit is set whenever a write operation to ME_<*mode*>_MC registers with invalid mode configuration is attempted. It is cleared by writing a ‘1’ to this bit.<br>0 No invalid mode configuration interrupt occurred<br>1 Invalid mode configuration interrupt is pending |
| I_IMODE | **Invalid mode interrupt** — This bit is set whenever an invalid mode transition is requested. It is cleared by writing a ‘1’ to this bit.<br>0 No invalid mode interrupt occurred<br>1 Invalid mode interrupt is pending |

**Table 29-7. Interrupt Status Register (ME_IS) Field Descriptions (continued)**

| Field | Description |
|---|---|
| I_SAFE | ***SAFE* mode interrupt** — This bit is set whenever the device enters ***SAFE*** mode on hardware requests generated in the system. It is cleared by writing a '1' to this bit.<br>0 No ***SAFE*** mode interrupt occurred<br>1 ***SAFE*** mode interrupt is pending |
| I_MTC | **Mode transition complete interrupt** — This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a '1' to this bit. This mode transition interrupt bit will not be set while entering low-power modes ***HALT***, ***STOP***, or ***STANDBY***.<br>0 No mode transition complete interrupt occurred<br>1 Mode transition complete interrupt is pending |

### 29.3.2.5 Interrupt Mask Register (ME_IM)

Address 0xC3FD_C010 Access: User read, Supervisor read/write, Test read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | M_ICONF_CU | M_ICONF | M_IMODE | M_SAFE | M_MTC |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-6. Interrupt Mask Register (ME_IM)**

This register controls whether an event generates an interrupt or not.

**Table 29-8. Interrupt Mask Register (ME_IM) Field Descriptions**

| Field | Description |
|---|---|
| M_ICONF_CU | **Invalid mode configuration interrupt (Clock Usage) mask**<br>0 Invalid mode configuration (clock usage) interrupt is masked<br>1 Invalid mode configuration (clock usage) is enabled |
| M_ICONF | **Invalid mode configuration interrupt mask**<br>0 Invalid mode interrupt is masked<br>1 Invalid mode interrupt is enabled |
| M_IMODE | **Invalid mode interrupt mask**<br>0 Invalid mode interrupt is masked<br>1 Invalid mode interrupt is enabled |
| M_SAFE | ***SAFE* mode interrupt mask**<br>0 ***SAFE*** mode interrupt is masked<br>1 ***SAFE*** mode interrupt is enabled |
| M_MTC | **Mode transition complete interrupt mask**<br>0 Mode transition complete interrupt is masked<br>1 Mode transition complete interrupt is enabled |

### 29.3.2.6 Invalid Mode Transition Status Register (ME_IMTS)

Address 0xC3FD_C014 Access: User read, Supervisor read/write, Test read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S_MTI | S_MRI | S_DMA | S_NMA | S_SEA |
| W | | | | | | | | | | | | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-7. Invalid Mode Transition Status Register (ME_IMTS)**

This register provides the status bits for the possible causes of an invalid mode interrupt.

**Table 29-9. Invalid Mode Transition Status Register (ME_IMTS) Field Descriptions**

| Field | Description |
|---|---|
| S_MTI | **Mode Transition Illegal status** — This bit is set whenever a new mode is requested while some other mode transition process is active (S_MTRANS is ‘1’). Please refer to Section 29.4.5, “Mode Transition Interrupts for the exceptions to this behavior. It is cleared by writing a ‘1’ to this bit.<br>0 Mode transition requested is not illegal<br>1 Mode transition requested is illegal |
| S_MRI | **Mode Request Illegal status** — This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a ‘1’ to this bit.<br>0 Target mode requested is not illegal with respect to current mode<br>1 Target mode requested is illegal with respect to current mode |
| S_DMA | **Disabled Mode Access status** — This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit.<br>0 Target mode requested is not a disabled mode<br>1 Target mode requested is a disabled mode |
| S_NMA | **Non-existing Mode Access status** — This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit.<br>0 Target mode requested is an existing mode<br>1 Target mode requested is a non-existing mode |
| S_SEA | ***SAFE* Event Active status** — This bit is set whenever the device is in ***SAFE*** mode, ***SAFE*** event bit is pending and a new mode requested other than ***RESET*/*SAFE*** modes. It is cleared by writing a ‘1’ to this bit.<br>0 No new mode requested other than ***RESET*/*SAFE*** while ***SAFE*** event is pending<br>1 New mode requested other than ***RESET*/*SAFE*** while ***SAFE*** event is pending |

### 29.3.2.7 Debug Mode Transition Status Register (ME_DMTS)

Address 0xC3FD_C018 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PREVIOUS_MODE | | | | 0 | 0 | 0 | 0 | MPH_BUSY | 0 | 0 | PMC_PROG | CORE_DBG | 0 | 0 | SMR |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | VREG_CSRC_SC | CSRC_CSRC_SC | FIRC_SC | SCSRC_SC | SYSCLK_SW | | CFLASH_SC | CDP_PRPH_0_143 | 0 | 0 | 0 | 0 | CDP_PRPH_64_95 | CDP_PRPH_32_63 | CDP_PRPH_0_31 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-8. Debug Mode Transition Status Register (ME_DMTS)**

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by **ME_GS.S_MTRANS** may be taking longer than expected.

**NOTE**

The **ME_DMTS** register does not indicate whether a mode transition is ongoing. Therefore, some **ME_DMTS** bits may still be asserted after the mode transition has completed.

**Table 29-10. Debug Mode Transition Status Register (ME_DMTS) Field Descriptions**

| Field | Description |
|---|---|
| PREVIOUS_MODE | Previous **device mode** — These bits show the mode in which the device was prior to the latest change to the current mode.<br>0000 ***RESET***<br>0001 ***TEST***<br>0010 ***SAFE***<br>0011 ***DRUN***<br>0100 ***RUN0***<br>0101 ***RUN1***<br>0110 ***RUN2***<br>0111 ***RUN3***<br>1000 ***HALT***<br>1001 reserved<br>1010 ***STOP***<br>1011 reserved<br>1100 reserved<br>1101 ***STANDBY***<br>1110 reserved<br>1111 reserved |
| MPH_BUSY | MC_ME/MC_PCU Handshake Busy indicator — This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet responded. It is cleared when the MC_PCU has responded.<br>0 Handshake is not busy<br>1 Handshake is busy |
| PMC_PROG | MC_PCU Mode Change in Progress indicator — This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed.<br>0 Power-up/down transition is not in progress<br>1 Power-up/down transition is in progress |
| CORE_DBG | Processor is in Debug mode indicator — This bit is set while the processor is in debug mode.<br>0 The processor is not in debug mode<br>1 The processor is in debug mode |
| SMR | ***SAFE*** mode request from MC_RGM is active indicator — This bit is set if a hardware ***SAFE*** mode request has been triggered. It is cleared when the hardware ***SAFE*** mode request has been cleared.<br>0 A ***SAFE*** mode request is not active<br>1 A ***SAFE*** mode request is active |
| VREG_CSRC_SC | Main VREG dependent Clock Source State Change during mode transition indicator — This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change.<br>0 No state change is taking place<br>1 A state change is taking place |
| CSRC_CSRC_SC | (Other) Clock Source dependent Clock Source State Change during mode transition indicator — This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change.<br>0 No state change is taking place<br>1 A state change is taking place |
| FIRC_SC | FIRC State Change during mode transition indicator — This bit is set when the fast internal RC oscillator (16 MHz) is requested to change its power up/down state. It is cleared when the fast internal RC oscillator (16 MHz) has completed its state change.<br>0 No state change is taking place<br>1 A state change is taking place |

**Table 29-10. Debug Mode Transition Status Register (ME_DMTS) Field Descriptions (continued)**

| Field | Description |
|---|---|
| SCSRC_SC | Secondary Clock Sources State Change during mode transition indicator — This bit is set when a secondary clock source is requested to change its power up/down state. It is cleared when all secondary system clock sources have completed their state changes. (A 'secondary clock source' is a clock source other than FIRC.)<br>0 No state change is taking place<br>1 A state change is taking place |
| SYSCLK_SW | System Clock Switching pending status —<br>0 No system clock source switching is pending<br>1 A system clock source switching is pending |
| CFLASH_SC | CFLASH State Change during mode transition indicator — This bit is set when the CFLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change.<br>0 No state change is taking place<br>1 A state change is taking place |
| CDP_PRPH_0_143 | Clock Disable Process Pending status for Peripherals 0…143 — This bit is set when any peripheral has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled.<br>0 No peripheral clock disabling is pending<br>1 Clock disabling is pending for at least one peripheral |
| CDP_PRPH_96_127 | Clock Disable Process Pending status for Peripherals 96…127 — This bit is set when any peripheral appearing in **ME_PS3** has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled.<br>0 No peripheral clock disabling is pending<br>1 Clock disabling is pending for at least one peripheral |
| CDP_PRPH_64_95 | Clock Disable Process Pending status for Peripherals 64…95 — This bit is set when any peripheral appearing in **ME_PS2** has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled.<br>0 No peripheral clock disabling is pending<br>1 Clock disabling is pending for at least one peripheral |
| CDP_PRPH_32_63 | Clock Disable Process Pending status for Peripherals 32…63 — This bit is set when any peripheral appearing in **ME_PS1** has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled.<br>0 No peripheral clock disabling is pending<br>1 Clock disabling is pending for at least one peripheral |
| CDP_PRPH_0_31 | Clock Disable Process Pending status for Peripherals 0…31 — This bit is set when any peripheral appearing in **ME_PS0** has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled.<br>0 No peripheral clock disabling is pending<br>1 Clock disabling is pending for at least one peripheral |

### 29.3.2.8 *RESET* Mode Configuration Register (ME_RESET_MC)

Address 0xC3FD_C020 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | 0 | 0 | FLAON | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Figure 29-9. RESET Mode Configuration Register (ME_RESET_MC)**

This register configures system behavior during ***RESET*** mode. Please refer to Table 29-11 for details.

### 29.3.2.9 *TEST* Mode Configuration Register (ME_TEST_MC)

Address 0xC3FD_C024 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | 0 | 0 | FLAON | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Figure 29-10. TEST Mode Configuration Register (ME_TEST_MC)**

This register configures system behavior during ***TEST*** mode. Please refer to Table 29-11 for details.

**NOTE**

Byte write accesses are not allowed to this register.

### 29.3.2.10 *SAFE* Mode Configuration Register (ME_SAFE_MC)

Address 0xC3FD_C028 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | 0 | 0 | FLAON | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Figure 29-11. SAFE Mode Configuration Register (ME_SAFE_MC)**

This register configures system behavior during ***SAFE*** mode. Please refer to Table 29-11 for details.

**NOTE**

Byte write accesses are not allowed to this register.

### 29.3.2.11 *DRUN* Mode Configuration Register (ME_DRUN_MC)

Address 0xC3FD_C02C Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | 0 | 0 | FLAON | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Figure 29-12. DRUN Mode Configuration Register (ME_DRUN_MC)**

This register configures system behavior during ***DRUN*** mode. Please refer to Table 29-11 for details.

**NOTE**

Byte write accesses are not allowed to this register.

**NOTE**

The clock source and flash configuration values are retained through ***STANDBY*** mode.

### 29.3.2.12 *RUN0…3* Mode Configuration Registers (ME_RUN0…3_MC)

Address 0xC3FD_C030 - 0xC3FD_C03C — Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | 0 | 0 | FLAON | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Figure 29-13. RUN0…3 Mode Configuration Registers (ME_RUN0…3_MC)**

This register configures system behavior during ***RUN0…3*** modes. Please refer to Table 29-11 for details.

**NOTE**

Byte write accesses are not allowed to this register.

### 29.3.2.13 *HALT* Mode Configuration Register (ME_HALT_MC)

Address 0xC3FD_C040 — Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | 0 | 0 | FLAON | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Figure 29-14. HALT Mode Configuration Register (ME_HALT_MC)**

This register configures system behavior during ***HALT*** mode. Please refer to Table 29-11 for details.

**NOTE**

Byte write accesses are not allowed to this register.

### 29.3.2.14 *STOP* Mode Configuration Register (ME_STOP_MC)

Address 0xC3FD_C048 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | 0 | 0 | FLAON | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Figure 29-15. STOP Mode Configuration Register (ME_STOP_MC)**

This register configures system behavior during ***STOP*** mode. Please refer to Table 29-11 for details.

**NOTE**

Byte write accesses are not allowed to this register.

### 29.3.2.15 *STANDBY* Mode Configuration Register (ME_STANDBY_MC)

Address 0xC3FD_C054 Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO | 0 | 0 | MVRON | 0 | 0 | FLAON | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FMPLL1ON | FMPLL0ON | FXOSCON | FIRCON | SYSCLK | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

**Figure 29-16. STANDBY Mode Configuration Register (ME_STANDBY_MC)**

This register configures system behavior during ***STANDBY*** mode. Please refer to Table 29-11 for details.

**NOTE**

Byte write accesses are not allowed to this register.

**Table 29-11. Mode Configuration Registers (ME_<*mode*>_MC) Field Descriptions**

| Field | Description |
|---|---|
| PDO | **I/O output power-down control** — This bit controls the output power-down of I/Os.<br>0 No automatic safe gating of I/Os used and pads power sequence driver is enabled<br>1 In ***SAFE***/***TEST*** modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In ***STOP*** mode, only the pad power sequence driver is disabled, but the state of the output remains functional. In ***STANDBY*** mode, power sequence driver and all pads except those mapped on wakeup lines are not powered and therefore high impedance. Wakeup line configuration remains unchanged |
| MVRON | **Main voltage regulator control** — This bit specifies whether main voltage regulator is switched off or not while entering this mode.<br>0 Main voltage regulator is switched off<br>1 Main voltage regulator is switched on |
| FLAON | Flash memory **power-down control** — This bit specifies the operating mode of the code flash after entering this mode.<br>00 Reserved<br>01 Flash memory is in power-down mode<br>10 Flash memory is in low-power mode<br>11 Flash memory is in normal mode |
| FMPLL1ON | **secondary frequency modulated phase locked loop control**<br>0 secondary frequency modulated phase locked loop is switched off<br>1 secondary frequency modulated phase locked loop is switched on |
| FMPLL0ON | **primary frequency modulated phase locked loop control**<br>0 primary frequency modulated phase locked loop is switched off<br>1 primary frequency modulated phase locked loop is switched on |
| FXOSCON | **fast external crystal oscillator (4-16 MHz) control**<br>0 fast external crystal oscillator (4-16 MHz) is switched off<br>1 fast external crystal oscillator (4-16 MHz) is switched on |
| FIRCON | **fast internal RC oscillator (16 MHz) control**<br>0 fast internal RC oscillator (16 MHz) is switched off<br>1 fast internal RC oscillator (16 MHz) is switched on |
| SYSCLK | **System clock switch control** — These bits specify the system clock to be used by the system.<br>0000 16 MHz int. RC osc.<br>0001 reserved<br>0010 reserved<br>0011 div. 4-16 MHz ext. xtal osc.<br>0100 primary PLL/2<br>0101 reserved<br>0110 reserved<br>0111 reserved<br>1000 reserved<br>1001 reserved<br>1010 reserved<br>1011 reserved<br>1100 reserved<br>1101 reserved<br>1110 reserved<br>1111 system clock is disabled in ***STOP*** and ***TEST*** modes, reserved in all other modes |

### 29.3.2.16 Peripheral Status Register 0 (ME_PS0)

Address 0xC3FD_C060 Access: User read, Supervisor read, Test read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | S_BAM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S_DMA_CH_MUX | 0 | 0 | 0 | 0 | S_FLEXCAN2 | S_FLEXCAN1 | S_FLEXCAN0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S_QUADSPI | 0 | S_DSPI2 | S_DSPI1 | S_DSPI0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-17. Peripheral Status Register 0 (ME_PS0)**

This register provides the status of the peripherals. Please refer to Table 29-12 for details.

### 29.3.2.17 Peripheral Status Register 1 (ME_PS1)

Address 0xC3FD_C064 Access: User read, Supervisor read, Test read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | S_TCON | S_SGM | 0 | S_CANSAMPLER | 0 | 0 | 0 | S_GAUGEDRIVER | S_DCU3 | S_DCULITE | S_GXG | S_GFX2D | S_LINFLEX3 | S_LINFLEX2 | S_LINFLEX1 | S_LINFLEX0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | S_I2C_DMA3 | S_I2C_DMA2 | S_I2C_DMA1 | S_I2C_DMA0 | 0 | 0 | 0 | 0 | 0 | 0 | S_RLE | 0 | 0 | 0 | 0 | S_ADC0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-18. Peripheral Status Register 1 (ME_PS1)**

This register provides the status of the peripherals. Please refer to Table 29-12 for details.

### 29.3.2.18 Peripheral Status Register 2 (ME_PS2)

Address 0xC3FD_C068 Access: User read, Supervisor read, Test read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | S_PIT_RTI | S_RTC_API | S_MC_PCU | S_MC_RGM | S_MC_CGM | S_MC_ME | S_SSCM | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | S_EMIOS1 | S_EMIOS0 | 0 | 0 | S_WKPU | S_SIUL | 0 | S_CFLASH0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-19. Peripheral Status Register 2 (ME_PS2)**

This register provides the status of the peripherals. Please refer to Table 29-12 for details.

### 29.3.2.19 Peripheral Status Register 3 (ME_PS3)

Address 0xC3FD_C06C Access: User read, Supervisor read, Test read

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | S_CMU0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-20. Peripheral Status Register 3 (ME_PS3)**

This register provides the status of the peripherals. Please refer to Table 29-12 for details.

**Table 29-12. Peripheral Status Registers 0…4 (ME_PS0…4) Field Descriptions**

| Field | Description |
|---|---|
| S_<*periph*> | **Peripheral status** — These bits specify the current status of the peripherals in the system. If no peripheral is mapped on a particular position (i.e., the corresponding *MODS* bit is ‘0’), the corresponding bit is always read as ‘0’.<br>0 Peripheral is frozen<br>1 Peripheral is active |

## 29.3.2.20 Run Peripheral Configuration Registers (ME_RUN_PC0…7)

Address 0xC3FD_C080 - 0xC3FD_C09C Access: User read, Supervisor read/write, Test read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RUN3 | RUN2 | RUN1 | RUN0 | DRUN | SAFE | TEST | RESET |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 29-21. Run Peripheral Configuration Registers (ME_RUN_PC0…7)**

These registers configure eight different types of peripheral behavior during run modes.

**Table 29-13. Run Peripheral Configuration Registers (ME_RUN_PC0…7) Field Descriptions**

| Field | Description |
|---|---|
| RUN3 | **Peripheral control during *RUN3***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |
| RUN2 | **Peripheral control during *RUN2***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |
| RUN1 | **Peripheral control during *RUN1***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |
| RUN0 | **Peripheral control during *RUN0***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |
| DRUN | **Peripheral control during *DRUN***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |
| SAFE | **Peripheral control during *SAFE***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |

Table 29-13. Run Peripheral Configuration Registers (ME_RUN_PC0…7) Field Descriptions (continued)

| Field | Description |
|---|---|
| TEST | **Peripheral control during *TEST***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |
| RESET | **Peripheral control during *RESET***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |

### 29.3.2.21 Low-Power Peripheral Configuration Registers (ME_LP_PC0…7)

Address 0xC3FD_C0A0 - 0xC3FD_C0BC Access: User read, Supervisor read/write, Test read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | STANDBY | 0 | 0 | STOP | 0 | HALT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 29-22. Low-Power Peripheral Configuration Registers (ME_LP_PC0…7)

These registers configure eight different types of peripheral behavior during non-run modes.

Table 29-14. Low-Power Peripheral Configuration Registers (ME_LP_PC0…7) Field Descriptions

| Field | Description |
|---|---|
| STANDBY | **Peripheral control during *STANDBY***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |
| STOP | **Peripheral control during *STOP***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |
| HALT | **Peripheral control during *HALT***<br>0 Peripheral is frozen with clock gated<br>1 Peripheral is active |

### 29.3.2.22 Peripheral Control Registers (ME_PCTL0…143)

Address 0xC3FD_C0C0 - 0xC3FD_C14F Access: User read, Supervisor read/write, Test read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | DBG_F | LP_CFG | | | RUN_CFG | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 29-23. Peripheral Control Registers (ME_PCTL0…143)

These registers select the configurations during run and non-run modes for each peripheral.

**Table 29-15. Peripheral Control Registers (ME_PCTL0…143) Field Descriptions**

| Field | Description |
|---|---|
| DBG_F | Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode<br>0 Peripheral state depends on **RUN_CFG/LP_CFG** bits and the device mode<br>1 Peripheral is frozen if not already frozen in device modes.<br><br>**Note:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. |
| LP_CFG | **Peripheral configuration select for non-run modes** — These bits associate a configuration as defined in the **ME_LP_PC0…7** registers to the peripheral.<br>000 Selects ME_LP_PC0 configuration<br>001 Selects ME_LP_PC1 configuration<br>010 Selects ME_LP_PC2 configuration<br>011 Selects ME_LP_PC3 configuration<br>100 Selects ME_LP_PC4 configuration<br>101 Selects ME_LP_PC5 configuration<br>110 Selects ME_LP_PC6 configuration<br>111 Selects ME_LP_PC7 configuration |
| RUN_CFG | **Peripheral configuration select for run modes** — These bits associate a configuration as defined in the **ME_RUN_PC0…7** registers to the peripheral.<br>000 Selects ME_RUN_PC0 configuration<br>001 Selects ME_RUN_PC1 configuration<br>010 Selects ME_RUN_PC2 configuration<br>011 Selects ME_RUN_PC3 configuration<br>100 Selects ME_RUN_PC4 configuration<br>101 Selects ME_RUN_PC5 configuration<br>110 Selects ME_RUN_PC6 configuration<br>111 Selects ME_RUN_PC7 configuration |

**NOTE**

After modifying any of the ME_RUN_PCx, ME_LP_PCx, ME_PCTLx registers, request a mode change and wait for the mode change to be completed before entering debug mode in order to have consistent behavior on peripheral clock control process and clock status reporting in the ME_PSx registers.

## 29.4 Functional description

### 29.4.1 Mode Transition Request

The transition from one mode to another mode is normally handled by software by accessing the mode control register **ME_MCTL**. But the in case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the **ME_MCTL** register twice by writing

- the first time with the value of the key (0x5AF0) into the **KEY** bit field and the required target mode into the **TARGET_MODE** bit field,

- and the second time with the inverted value of the key (0xA50F) into the **KEY** bit field and the required target mode into the **TARGET_MODE** bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding **ME_<*mode*>_MC** register. The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the **S_CURRENT_MODE** bit field and the **S_MTRANS** bit of the global status register **ME_GS** to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please refer to Section 29.4.5, "Mode Transition Interrupts".

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as ***RUN0…3*** → ***RUN0…3***, ***DRUN*** → ***DRUN***, ***SAFE*** → ***SAFE***, and ***TEST*** → ***TEST*** are considered valid mode transition requests. As soon as the mode request is accepted as valid, the **S_MTRANS** bit is set till the status in the **ME_GS** register matches the configuration programmed in the respective **ME_<*mode*>_MC** register.

**NOTE**

It is recommended that software poll the **S_MTRANS** bit in the **ME_GS** register after requesting a transition to ***HALT***, ***STOP***, or ***STANDBY*** modes.

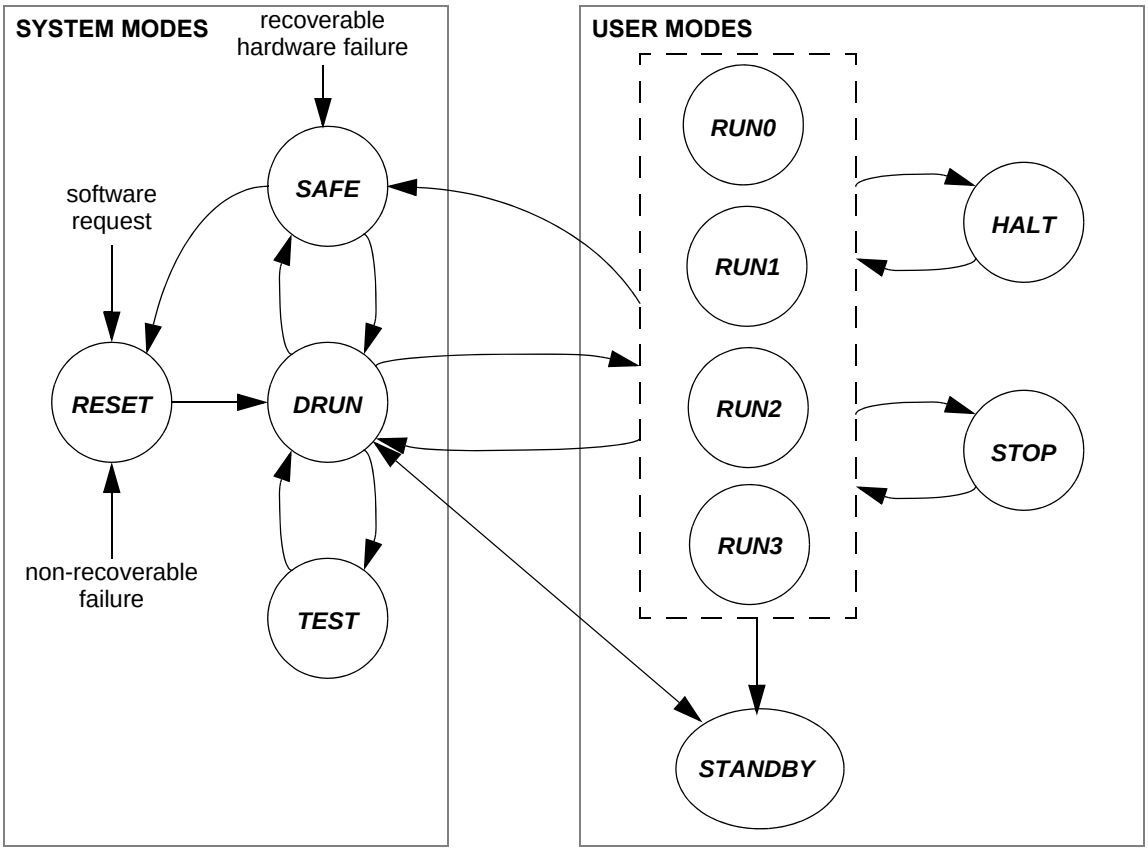



**Figure 29-24. MC_ME Mode Diagram**

## 29.4.2 Modes Details

### 29.4.2.1 *RESET* Mode

The device enters this mode on the following events:

- from ***SAFE***, ***DRUN***, ***RUN0…3***, or ***TEST*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0000”
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details)

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the **ME_RESET_MC** register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock. All power domains are made active in this mode.

### 29.4.2.2 *DRUN* Mode

The device enters this mode on the following events:

- automatically from ***RESET*** mode after completion of the reset sequence
- from ***RUN0…3***, ***SAFE***, or ***TEST*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0011”
- from the ***STANDBY*** mode after an external wakeup event or internal wakeup alarm (e.g., RTC/API event)

As soon as any of the above events has occurred, a ***DRUN*** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_DRUN_MC** register. In this mode, the flash, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the 16 MHz int. RC osc. as the system clock.

This mode is intended to be used by software

- to initialize all registers as per the system needs
- to execute small routines in a ‘ping-pong’ with the ***STANDBY*** mode

When this mode is entered from ***STANDBY*** after a wakeup event, the **ME_DRUN_MC** register content is restored to its pre-***STANDBY*** values, and the mode starts in that configuration.

All power domains are active when this mode is entered due to a system reset sequence initiated by a destructive reset event. In other cases of entry, such as the exit from ***STANDBY*** after a wakeup event, a functional reset event like an external reset or a software request from ***RUN0…3***, ***SAFE***, or ***TEST*** mode, active power domains are determined by the power configuration register **PCU_PCONF2** of the MC_PCU. All power domains except power domains #0 and #1 are configurable in this mode (see the MC_PCU chapter for details).

NOTE

Software must ensure that the code executes from RAM before changing to this mode if the flash configured to be in the low-power or power-down state in this mode.

### 29.4.2.3 *SAFE* Mode

The device enters this mode on the following events:

- from ***DRUN***, ***RUN0…3***, or ***TEST*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0010”
- from any mode except ***RESET*** due to a ***SAFE*** mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see the MC_RGM chapter for details)

As soon as any of the above events has occurred, a ***SAFE*** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_SAFE_MC** register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock. All power domains are made active in this mode.

If the ***SAFE*** mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the ***SAFE*** mode regardless of other pending requests or new requests during the mode transition. No new mode request made during a transition to the SAFE mode will cause an invalid mode interrupt.

**NOTE**

If software requests to change to the ***SAFE*** mode and then requests to change back to the parent mode before the mode transition is completed, the device’s final mode after mode transition will be the ***SAFE*** mode.

As long as a ***SAFE*** event is active, the system remains in the ***SAFE*** mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either
  - re-initialize the device via the ***DRUN*** mode, or
  - completely reset the device via the ***RESET*** mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the **PDO** bit of the **ME_SAFE_MC** register should be set. In this case, the pads’ power sequence driver cell is also disabled. The input levels remain unchanged.

### 29.4.2.4 *TEST* Mode

The device enters this mode on the following events:

- from the ***DRUN*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0001”

As soon as any of the above events has occurred, a ***TEST*** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_TEST_MC** register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole system can be stopped by programming the **SYSCLK** bit field to “1111”, and in this case, the only way to exit this mode is via a device reset.

This mode is intended to be used by software

- to execute software test routines

All power domains except power domains #0 and #1 are configurable in this mode. Active power domains are determined by the power configuration register **PCU_PCONF2** of the MC_PCU.

NOTE

Software must ensure that the code executes from RAM before changing to this mode if the flash configured to be in the low-power or power-down state in this mode.

### 29.4.2.5 *RUN0…3* Modes

The device enters one of these modes on the following events:

- from the ***DRUN***, ***SAFE***, or another ***RUN0…3*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “0100…0111”
- from the ***HALT*** mode due to an interrupt event
- from the ***STOP*** mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a ***RUN0…3*** mode transition request is generated. The mode configuration information for these modes is provided by the **ME_RUN0…3_MC** registers. In these modes, the flash, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

All power domains except power domains #0 and #1 are configurable in these modes in order to reduce leakage consumption. Active power domains are determined by the power configuration register **PCU_PCONF2** of the MC_PCU.

NOTE

Software must ensure that the code executes from RAM before changing to this mode if the flash configured to be in the low-power or power-down state in this mode.

### 29.4.2.6 *HALT* Mode

The device enters this mode on the following events:

- from one of the ***RUN0…3*** modes when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “1000”.

As soon as any of the above events has occurred, a ***HALT*** mode transition request is generated. The mode configuration information for this mode is provided by **ME_HALT_MC** register. This mode is quite configurable, and the **ME_HALT_MC** register should be programmed according to the system needs. The main voltage regulator and the flash can be put in low-power or power-down mode as needed. If there is a ***HALT*** mode request while an interrupt request is active, the transition to ***HALT*** is aborted with the resultant mode being the current mode, ***SAFE*** (on ***SAFE*** mode request), or ***DRUN*** (on reset), and an invalid mode interrupt is not generated.

This mode is intended as a first-level low-power mode with

- the CPU clock frozen
- only a few peripherals running

and to be used by software

- to wait until it is required to do something and then to react quickly (i.e., within a few system clock cycles of an interrupt event)

All power domains except power domains #0 and #1 are configurable in this mode in order to reduce leakage consumption. Active power domains are determined by the power configuration register **PCU_PCONF2** of the MC_PCU.

#### 29.4.2.7 *STOP* Mode

The device enters this mode on the following events:

- from one of the ***RUN0…3*** modes when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “1010”.

As soon as any of the above events has occurred, a ***STOP*** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_STOP_MC** register. This mode is fully configurable, and the **ME_STOP_MC** register should be programmed according to the system needs.

The main voltage regulator and the flash can be put in power-down mode as needed. If there is a ***STOP*** mode request while any interrupt or wakeup event is active, the transition to ***STOP*** is aborted with the resultant mode being the current mode, ***SAFE*** (on ***SAFE*** mode request), or ***DRUN*** (on reset), and an invalid mode interrupt is not generated.

This can be used as an advanced low-power mode with the CPU clock frozen and almost all peripherals stopped.

This mode is intended as an advanced low-power mode with

- the system clock frozen
- almost all peripherals stopped

and to be used by software

- to wait until it is required to do something with no need to react quickly (e.g., allow for system clock source to be re-started)

If the pads’ power sequence driver cell needs to be disabled while entering this mode, the **PDO** bit of the **ME_STOP_MC** register should be set. The state of the outputs is kept.

This mode can be used to stop all clock sources and thus preserve the device status. When exiting the ***STOP*** mode, the fast internal RC oscillator (16 MHz) clock is selected as the system clock until the target clock is available.

All power domains except power domains #0 and #1 are configurable in this mode in order to reduce leakage consumption. Active power domains are determined by the power configuration register **PCU_PCONF2** of the MC_PCU.

#### 29.4.2.8 *STANDBY* Mode

The device enters this mode on the following events:

- from the ***DRUN*** or one of the ***RUN0…3*** modes when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with “1101”.

As soon as any of the above events occur, a ***STANDBY*** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_STANDBY_MC** register. In this mode, the power supply is turned off for most of the device. The only parts of the device that are still powered during this mode are pads mapped on wakeup lines and power domain #0 which contains the MC_RGM, MC_PCU, WKPU, 8K RAM, RTC/API, CANSampler, SIRC, FIRC, and device and user option bits. The FIRC can be optionally switched off. This is the lowest power consumption mode possible on the device.

This mode is intended as an extreme low-power mode with

- the CPU, the flashes, and almost all peripherals and memories powered down

and to be used by software

- to wait until it is required to do something with no need to react quickly (i.e., allow for system power-up and system clock source to be re-started)

The exit sequence of this mode is similar to the reset sequence. However, in addition to booting from the default location, the device can also be configured to boot from the backup RAM (see the **RGM_STDBY** register description in the MC_RGM chapter for details). In the case of booting from backup RAM, it is also possible to keep the flashes disabled by writing “01” to the **CFLAON** and **DFLAON** fileds in the **ME_DRUN_MC** register prior to ***STANDBY*** entry. When booting from backup RAM, the core will begin executing at address 0x40000000. A 4K MMU entry will be created commencing at this address, configured to use the VLE instruction set.

If there is a ***STANDBY*** mode request while any wakeup event is active, the device mode does not change.

All power domains except power domain #0 are configurable in this mode in order to reduce leakage consumption. Active power domains are determined by the power configuration register **PCU_PCONF2** of the MC_PCU.

### 29.4.3 Mode Transition Process

The process of mode transition follows the following steps in a pre-defined manner depending on the current device mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

### 29.4.3.1 Target Mode Request

The target mode is requested by accessing the **ME_MCTL** register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the **TARGET_MODE** bit field is not updated. An optional interrupt can be generated for invalid mode requests. Refer to Section 29.4.5, “Mode Transition Interrupts for details.

In the case of mode transitions occurring because of hardware events such as a reset, a ***SAFE*** mode request, or interrupt requests and wakeup events to exit from low-power modes, the **TARGET_MODE** bit field of the **ME_MCTL** register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit **S_MTRANS** of the **ME_GS** register.

A ***RESET*** mode requested via the **ME_MCTL** register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The ***RESET*** mode request has the highest priority, and the MC_ME is kept in the ***RESET*** mode during the entire reset sequence.

The ***SAFE*** mode request has the next highest priority after reset. It can be generated either by software via the **ME_MCTL** register from all software running modes including ***DRUN***, ***RUN0…3***, and ***TEST*** or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

### 29.4.3.2 Target Mode Configuration Loading

On completion of the Target Mode Request step, the target mode configuration from the **ME_*<target mode>*_MC** register is loaded to start the resources (voltage sources, clock sources, flash, pads, etc.) control process.

An overview of resource control possibilities for each mode is shown in Table 29-16. A ‘√’ indicates that a given resource is configurable for a given mode.

**Table 29-16. MC_ME Resource Control Overview**

| Resource | Mode | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ***RESET*** | ***TEST*** | ***SAFE*** | ***DRUN*** | ***RUN0…3*** | ***HALT*** | ***STOP*** | ***STANDBY*** |
| FIRC | on | √<br>on | on | on | on | √<br>on | √<br>on | √<br>on |
| FXOSC | off | √<br>off | off | √<br>off | √<br>off | √<br>off | √<br>off | √<br>off |
| FMPLL0 | off | √<br>off | off | √<br>off | √<br>off | √<br>off | off | off |
| FMPLL1 | off | √<br>off | off | √<br>off | √<br>off | √<br>off | off | off |

Table 29-16. MC_ME Resource Control Overview (continued)

| Resource | Mode | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ***RESET*** | ***TEST*** | ***SAFE*** | ***DRUN*** | ***RUN0…3*** | ***HALT*** | ***STOP*** | ***STANDBY*** |
| CFLASH | normal | √ normal | normal | √ normal | √ normal | √ low-power | √ power-down | power-down |
| MVREG | on | on | on | on | on | √ on | √ on | off |
| PDO | off | √ off | √ on | off | off | off | √ off | on |

### 29.4.3.3 Peripheral Clocks Disable

On completion of the Target Mode Request step, the MC_ME requests each peripheral to enter its stop mode when:

- the peripheral is configured to be disabled via the target mode, the peripheral configuration registers **ME_RUN_PC0…7** and **ME_LP_PC0…7**, and the peripheral control registers **ME_PCTL0…143**

**WARNING**

The MC_ME does not automatically request peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. Therefore, it is software's responsibility to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

**NOTE**

If any clock source needs to be cut-off, then the peripherals which are running on this source, need to be switched to some other clock source or should be switched off.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a ***SAFE*** mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The ***SAFE*** mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please refer to Section 29.4.6, "Peripheral Clock Gating" for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the device enters the ***SAFE*** mode.

### 29.4.3.4 Processor Low-Power Mode Entry

If, on completion of the Peripheral Clocks Disable step, the mode transition is to the ***HALT*** mode, the MC_ME requests the processor to enter its halted state. The processor acknowledges its halt state request after completing all outstanding bus transactions.

If, on completion of the Peripheral Clocks Disable step, the mode transition is to the ***STOP*** or ***STANDBY*** mode, the MC_ME requests the processor to enter its stopped state. The processor acknowledges its stop state request after completing all outstanding bus transactions.

### 29.4.3.5 Processor and System Memory Clock Disable

If, on completion of the Processor Low-Power Mode Entry step, the mode transition is to the ***HALT***, ***STOP***, or ***STANDBY*** mode and the processor is in its appropriate halted or stopped state, the MC_ME disables the processor and system memory clocks to achieve further power saving.

The clocks to the processor and system memory are unaffected while transitioning between software running modes such as ***DRUN***, ***RUN0…3***, and ***SAFE***.

**WARNING**

Clocks to the whole device including the processor and system memory can be disabled in ***TEST*** mode.

### 29.4.3.6 Clock sources (main voltage regulator independent) switch-on

On completion of the Processor Low-Power Mode Entry, the MC_ME controls all clock sources that affect the system clock based on the **<*clock source*>ON** bits of the **ME_<*current mode*>_MC** and **ME_<*target mode*>_MC** registers. The following system clock sources are switched on at this step:

- The fast internal RC oscillator (16 MHz)
- The fast external crystal oscillator (4–16 MHz)
- The secondary FMPLL

NOTE

Clock sources which need the main voltage regulator to be stable are not controlled by this step.

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the **S_<*clock source*>** bits of **ME_GS** register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

### 29.4.3.7 Main Voltage Regulator Switch-On

On completion of the Target Mode Request step, if the main voltage regulator needs to be switched on from its off state based on the **MVRON** bit of the **ME_<*current mode*>_MC** and **ME_<*target mode*>_MC**

registers, the MC_ME requests the MC_PCU to power-up the regulator and waits for the output voltage stable status in order to update the **S_MVR** bit of the **ME_GS** register.

This step is required only during the exit of the low-power modes ***HALT*** and ***STOP***. In this step, the fast internal RC oscillator (16 MHz) is switched on regardless of the target mode configuration, as the main voltage regulator requires the 16 MHz int. RC osc. during power-up in order to generate the voltage status.

During the ***STANDBY*** exit sequence, the MC_PCU alone manages the power-up of the main voltage regulator, and the MC_ME is kept in ***RESET*** or shut off (depending on the power domain #1 status).

**NOTE**

Software Guideline: ME_DRUN_CSRCON[2] should not be configured, to keep PLL ON in DRUN on standby exit.

#### 29.4.3.8 Flash Module Switch-On

On completion of the Main Voltage Regulator Switch-On step, if the flash needs to be switched to normal mode from its low-power or power-down mode based on the **FLAON** bit field of the **ME_<*current mode*>_MC** and **ME_<*target mode*>_MC** registers, the MC_ME requests the flash to exit from its low-power/power-down mode. When the flash available for access, the **S_FLA** bit field of the **ME_GS** register updated to “11” by hardware.

If the main regulator is also off in device low-power modes, then during the exit sequence, the flash is kept in its low-power state and is switched on only when the Main Voltage Regulator Switch-On process has completed.

**WARNING**

It is illegal to switch the flash from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

#### 29.4.3.9 Clock Sources (Main Voltage Regulator Dependent) Switch-On

On completion of the Clock sources (main voltage regulator independent) switch-on and Main Voltage Regulator Switch-On, the MC_ME controls all clock sources, which need the main voltage regulator to be on, based on the **<*clock source*>ON** bits of the **ME_<*current mode*>_MC** and **ME_<*target mode*>_MC** registers. The following clock sources are switched on at this step:

#### 29.4.3.10 Power Domain #2 Switch-On

On completion of the Main Voltage Regulator Switch-On step, the MC_ME indicates a mode change to the MC_PCU. The MC_PCU then determines whether a power-up sequence is required for power domain #2. Only after the MC_PCU has executed all required power-ups does the MC_ME complete the mode transition.

#### 29.4.3.11 Pad Outputs-On

On completion of the Main Voltage Regulator Switch-On step, if the **PDO** bit of the **ME_<*target mode*>_MC** register is cleared, then

- all pad outputs are enabled to return to their previous state
- the I/O pads power sequence driver is switched on

#### 29.4.3.12 Peripheral Clocks Enable

Based on the current and target device modes, the peripheral configuration registers **ME_RUN_PC0…7**, **ME_LP_PC0…7**, and the peripheral control registers **ME_PCTL0…143**, the MC_ME enables the clocks for selected modules as required. This step is executed only after the Main Voltage Regulator Switch-On process is completed.

Also, if a mode change translates to a power up of one or more power domains, the MC_PCU indicates the MC_ME after completing the power-up sequence upon which the MC_ME may assert the peripheral clock enables of the peripherals residing in those power domains.

#### 29.4.3.13 Processor and Memory Clock Enable

If the mode transition is from any of the low-power modes ***HALT*** or ***STOP*** to ***RUN0…3***, the clocks to the processor and system memory are enabled. The process of enabling these clocks is executed only after the Flash Module Switch-On process is completed.

#### 29.4.3.14 Processor Low-Power Mode Exit

If the mode transition is from any of the low-power modes ***HALT***, ***STOP***, or ***STANDBY*** to ***RUN0…3***, the MC_ME requests the processor to exit from its halted or stopped state. This step is executed only after the Processor and Memory Clock Enable process is completed.

#### 29.4.3.15 System Clock Switching

Based on the **SYSCLK** bit field of the **ME_<*current mode*>_MC** and **ME_<*target mode*>_MC** registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the 16 MHz int. RC osc. takes effect only after the **S_FIRC** bit of the **ME_GS** register is set by hardware (i.e., the fast internal RC oscillator (16 MHz) has stabilized).
- The target clock configuration for the div. 4-16 MHz ext. xtal osc. takes effect only after the **S_FXOSC** bit of the **ME_GS** register is set by hardware (i.e the fast external crystal oscillator (4-16 MHz) has stabilized).
- The target clock configuration for the primary PLL/2 takes effect only after the **S_FMPLL0** bit of the **ME_GS** register is set by hardware (i.e., the primary frequency modulated phase locked loop has stabilized).

- If the clock is to be disabled, the **SYSCLK** bit field should be programmed with “1111”. This is possible only in the ***STOP*** and ***TEST*** modes. In the ***STANDBY*** mode, the clock configuration is fixed, and the system clock is automatically forced to ‘0’.

The current system clock configuration can be observed by reading the **S_SYSCLK** bit field of the **ME_GS** register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the Peripheral Clocks Disable process has completed in order not to change the system clock frequency before peripherals close their internal activities

An overview of system clock source selection possibilities for each mode is shown in Table 29-17. A ‘√’ indicates that a given clock source is selectable for a given mode.

**Table 29-17. MC_ME System Clock Selection Overview**

| System Clock Source | Mode | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ***RESET*** | ***TEST*** | ***SAFE*** | ***DRUN*** | ***RUN0…3*** | ***HALT*** | ***STOP*** | ***STANDBY*** |
| 16 MHz int. RC osc. | √ (default) | √ (default) | √ (default) | √ (default) | √ (default) | √ (default) | √ (default) | |
| div. 4-16 MHz ext. xtal osc. | | √ | | √ | √ | √ | √ | |
| primary PLL/2 | | √ | | √ | √ | √ | | |
| system clock is disabled | | √[1] | | | | | √ | √ (default) |

[1] disabling the system clock during ***TEST*** mode will require a reset in order to exit ***TEST*** mode

### 29.4.3.16 Power Domain #2 Switch-Off

Based on the device mode and the MC_PCU’s power configuration register **PCU_PCONF2**, the power domain #2 is controlled by the MC_PCU.

If a mode change translates to a power-down of the power domain, then the MC_PCU starts the power-down sequence. The MC_PCU acknowledges the completion of the power-down sequence with respect to the new mode, and the MC_ME uses this information to update the mode transition status. This step is executed only after the Peripheral Clocks Disable process has completed.

### 29.4.3.17 Pad Switch-Off

If the **PDO** bit of the **ME_*<target mode>*_MC** register is ‘1’ then

- the outputs of the pads are forced to the high impedance state if the target mode is ***SAFE*** or ***TEST***

- I/O pads power sequence driver is switched off if the target mode is one of ***SAFE***, ***TEST***, or ***STOP*** modes

In ***STANDBY*** mode, the power sequence driver and all pads except the external reset and those mapped on wakeup lines are not powered and therefore high impedance. The wakeup line configuration remains unchanged.

This step is executed only after the Peripheral Clocks Disable process has completed.

### 29.4.3.18 Clock Sources Switch-Off

Based on the device mode and the **<*clock source*>ON** bits of the **ME_<*mode*>_MC** registers, if a given clock source is to be switched off, the MC_ME requests the clock source to power down and updates its availability status bit **S_<*clock source*>** of the **ME_GS** register to '0'. The following clock sources switched off at this step:

This step is executed only after the System Clock Switching process has completed.

### 29.4.3.19 Flash Switch-Off

Based on the **FLAON** bit field of the **ME_<*current mode*>_MC** and **ME_<*target mode*>_MC** registers, if the flash is to be put in its low-power or power-down mode, the MC_ME requests the flash to enter the corresponding power mode and waits for the flash to acknowledge. The exact power mode status of the flash is updated in the **S_FLA** bit field of the **ME_GS** register. This step is executed only when the Processor and System Memory Clock Disable process has completed.

### 29.4.3.20 Main Voltage Regulator Switch-Off

Based on the **MVRON** bit of the **ME_<*current mode*>_MC** and **ME_<*target mode*>_MC** registers, if the main voltage regulator is to be switched off, the MC_ME requests it to power down and clears the availability status bit **S_MVR** of the **ME_GS** register.

This step is required only during the entry of low-power modes like ***HALT*** and ***STOP***. This step is executed only after completing the following processes:

- Clock Sources Switch-Off
- Flash Switch-Off
- the device consumption is less than the pre-defined threshold value (i.e., the **S_DC** bit of the **ME_GS** register is '0').

If the target mode is ***STANDBY***, the main voltage regulator is not switched off by the MC_ME and the ***STANDBY*** request is asserted after the above processes have completed upon which the MC_PCU takes control of the main regulator. As the MC_PCU needs the 16 MHz int. RC osc., the fast internal RC oscillator (16 MHz) remains active until all the ***STANDBY*** steps are executed by the MC_PCU after which it may be switched off depending on the **FIRCON** bit of the **ME_STANDBY_MC** register.

### 29.4.3.21 Current Mode Update

The current mode status bit field **S_CURRENT_MODE** of the **ME_GS** register is updated with the target mode bit field **TARGET_MODE** of the **ME_MCTL** register when:

- all the updated status bits in the **ME_GS** register match the configuration specified in the **ME_<*target mode*>_MC** register
- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (halt/stop) entry and exit processes are finished

Software can monitor the mode transition status by reading the **S_MTRANS** bit of the **ME_GS** register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the **ME_DMTS** register can indicate which process is still in progress.

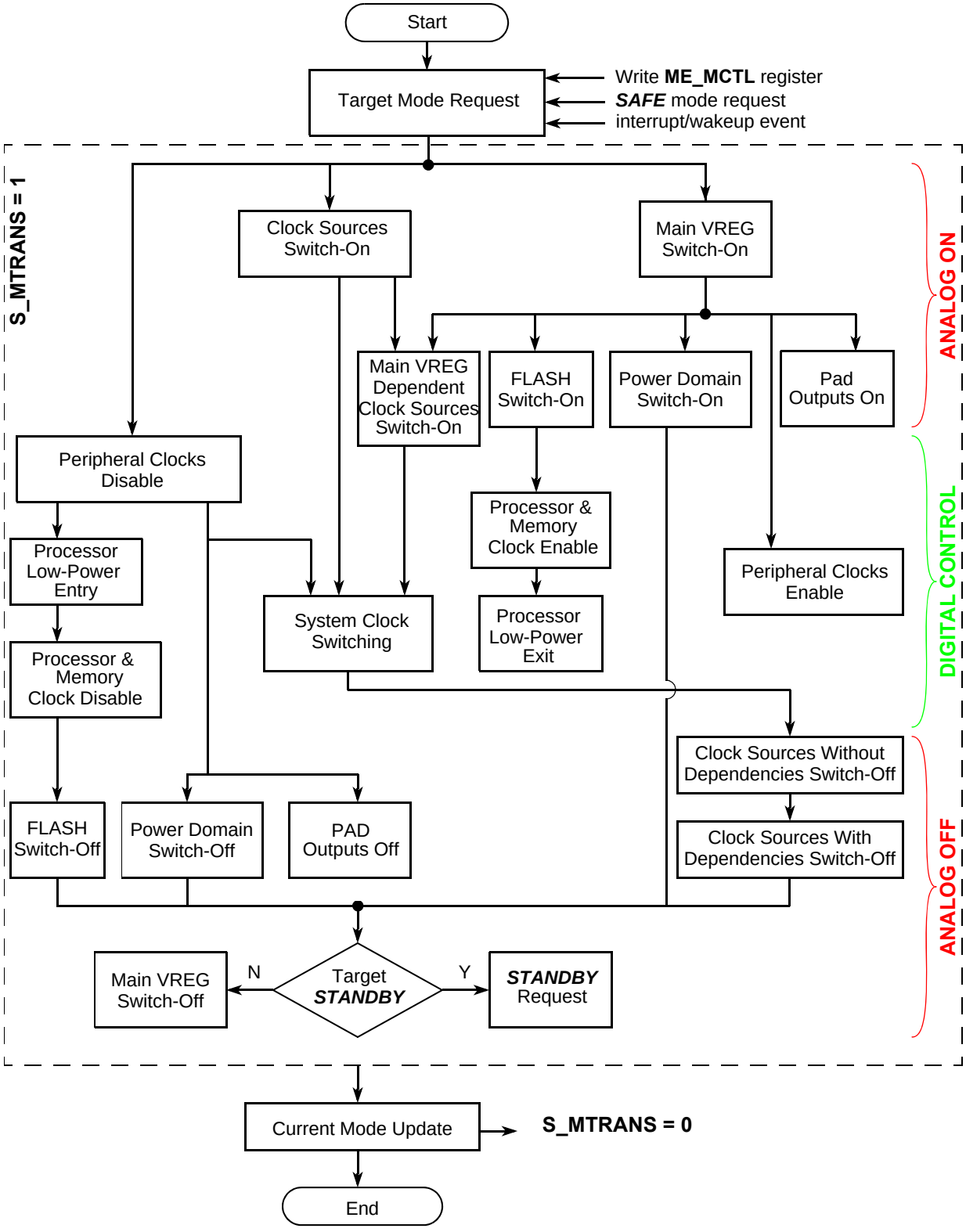



**Figure 29-25. MC_ME Transition Diagram**

### 29.4.4 Protection of Mode Configuration Registers

While programming the mode configuration registers **ME_<*mode*>_MC**, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- The FIRC must be enabled if it is to be set as the system clock. Likewise, the FIRC cannot be disabled whilst it is selected as the system clock.
- If the div. 4-16 MHz ext. xtal osc. clock is selected as the system clock, OSC must be on.
- If the primary PLL/2 clock is selected as the system clock, PLL and FXOSC must be on.

NOTE

Software must ensure that clock sources with dependencies other than those mentioned above are swithced on as needed. There is no automatic protection mechanism to check this in the MC_ME.

- Configuration “00” for the **FLAON** bit field is reserved.
- MVREG must be on if any of the following is active:
  — CFLASH
- System clock configurations marked as ‘reserved’ may not be selected.
- Configuration “1111” for the **SYSCLK** bit field is allowed only for the ***STOP*** and ***TEST*** modes, and only in this case may all system clock sources be turned off.

**WARNING**

If the system clock is stopped during ***TEST*** mode, the device can exit only via a system reset.

### 29.4.5 Mode Transition Interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a ***SAFE*** mode transition not due to a software request, and indicating when a mode transition has completed.

#### 29.4.5.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the **ME_<*mode*>_MC** registers violating the protection rules mentioned in the Section 29.4.4, “Protection of Mode Configuration Registers, the interrupt pending bit **I_ICONF** of the **ME_IS** register is set and an interrupt request is generated if the mask bit **M_ICONF** of **ME_IM** register is ‘1’.

#### 29.4.5.2 Invalid Mode Transition Interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the ***SAFE*** mode and the ***SAFE*** mode request from MC_RGM is active, and if the target mode requested is other than ***RESET*** or ***SAFE***, then this new mode request is considered to be invalid, and the **S_SEA** bit of the **ME_IMTS** register is set.

- If the **TARGET_MODE** bit field of the **ME_MCTL** register is written with a value different from the specified mode values (i.e., a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the **S_NMA** bit of the **ME_IMTS** register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the **ME_MCTL** register.
- If some of the device modes are disabled as programmed in the **ME_ME** register, their respective configurations are considered reserved, and any access to the **ME_MCTL** register with those values results in an invalid mode transition request. When such a disabled mode is requested, the **S_DMA** bit of the **ME_IMTS** register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the **ME_MCTL** register.
- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit **S_MRI** of the **ME_IMTS** register is set. This condition is detected only when the proper key mechanism is followed while writing the **ME_MCTL** register. Otherwise, the write operation is ignored.
- If further new mode requests occur while a mode transition is in progress (the **S_MTRANS** bit of the **ME_GS** register is '1'), the mode transition illegal status bit **S_MTI** of the **ME_IMTS** register is set. This condition is detected only when the proper key mechanism is followed while writing the **ME_MCTL** register. Otherwise, the write operation is ignored.

**NOTE**

As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the **ME_IMTS** register in the order from highest to lowest is **S_SEA**, **S_NMA**, **S_DMA**, **S_MRI**, and **S_MTI**.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the ***RESET*** or ***SAFE*** mode irrespective of the mode transition status.
- As the exit of ***HALT*** and ***STOP*** modes depends on the interrupts of the system which can occur at any instant, these requests to return to ***RUN0…3*** modes are always valid.
- In order to avoid any unwanted lockup of the device modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined 'reasonable' amount of time for whatever reason. The parent mode is the device mode before a valid mode request was made.
- Self-transition requests (e.g., ***RUN0*** → ***RUN0***) are not considered as invalid even when the mode transition process is active (i.e., **S_MTRANS** is '1'). During the low-power mode exit process, if the system is not able to enter the respective ***RUN0…3*** mode properly (i.e., all status bits of the **ME_GS** register match with configuration bits in the **ME_<*mode*>_MC** register), then software can only request the ***SAFE*** or ***RESET*** mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit **I_IMODE** of the **ME_IS** register is set, and an interrupt request is generated if the mask bit **M_IMODE** of the **ME_IM** register is '1'.

#### 29.4.5.3 *SAFE* Mode Transition Interrupt

Whenever the system enters the ***SAFE*** mode as a result of a ***SAFE*** mode request from the MC_RGM due to a hardware failure, the interrupt pending bit **I_SAFE** of the **ME_IS** register is set, and an interrupt is generated if the mask bit **M_SAFE** of **ME_IM** register is '1'.

The ***SAFE*** mode interrupt pending bit can be cleared only when the ***SAFE*** mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a ***SAFE*** mode request). If the system is already in ***SAFE*** mode, any new ***SAFE*** mode request by the MC_RGM also sets the interrupt pending bit **I_SAFE**. However, the ***SAFE*** mode interrupt pending bit is not set when the ***SAFE*** mode is entered by a software request (i.e., programming of **ME_MCTL** register).

#### 29.4.5.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (i.e., the **S_MTRANS** bit of **ME_GS** register transits from '1' to '0'), the interrupt pending bit **I_MTC** of the **ME_IS** register is set, and an interrupt request is generated if the mask bit **M_MTC** of the **ME_IM** register is '1'. The interrupt bit **I_MTC** is not set when entering low-power modes ***HALT*** and ***STOP*** in order to avoid the same event requesting the immediate exit of these low-power modes.

### 29.4.6 Peripheral Clock Gating

During all device modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers **ME_RUN_PC0…7** are chosen only during the software running modes ***DRUN***, ***TEST***, ***SAFE***, and ***RUN0…3***. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the **RUN_CFG** bit field of the **ME_PCTL0…143** registers.

The low-power peripheral configuration registers **ME_LP_PC0…7** are chosen only during the low-power modes ***HALT***, ***STOP***, and ***STANDBY***. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the **LP_CFG** bit field of the **ME_PCTL0…143** registers.

Any modifications to the **ME_RUN_PC0…7**, **ME_LP_PC0…7**, and **ME_PCTL0…143** registers do not affect the clock gating behavior until a new mode transition request is generated.

Whenever the processor enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the **DBG_F** bit of the associated **ME_PCTL0…143** register is set. Otherwise, the peripheral clock gating status depends on the **RUN_CFG** and **LP_CFG** bits. Any further modifications of the **ME_RUN_PC0…7**, **ME_LP_PC0…7**, and **ME_PCTL0…143** registers during a debug session will take affect immediately without requiring any new mode request.

### 29.4.7 Application Example

Figure 29-26 shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

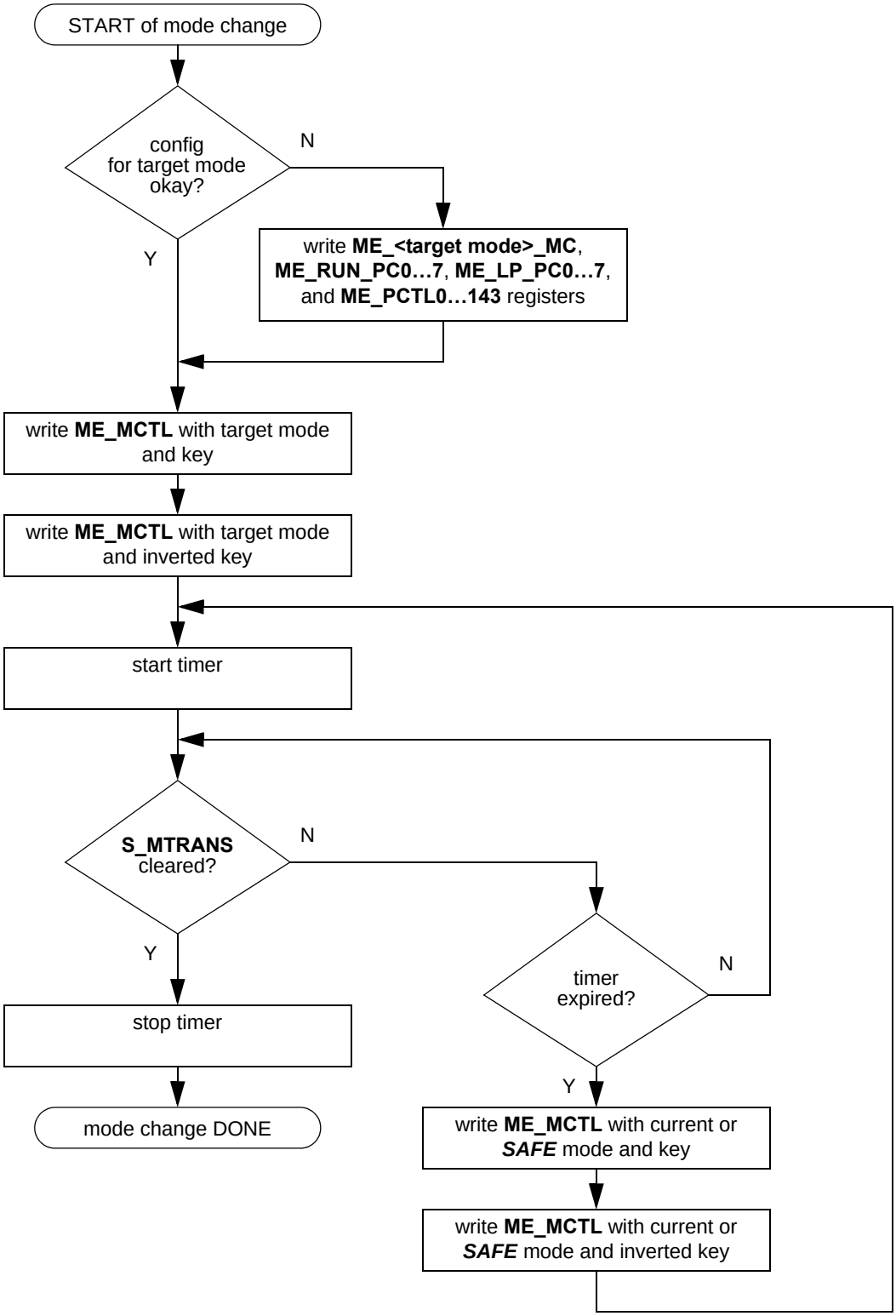



**Figure 29-26. MC_ME Application Example Flow Diagram**

# Chapter 30
# Nexus Development Interface (NDI)

## 30.1 Introduction

The Nexus Development Interface (NDI) block provides real-time development support capabilities for the MPC5645S MCU in compliance with the IEEE-ISTO 5001-2003 standard. This development support is supplied for MCUs without requiring external address and data pins for internal visibility.

The NDI block is an integration of several individual Nexus blocks that are selected to provide the development support interface for MPC5645S.

The NDI block interfaces to the e200z4d, and internal buses to provide development support as per the IEEE-ISTO 5001-2003 standard. The development support provided includes program trace, watchpoint messaging, ownership trace, watchpoint triggering, processor overrun control, run-time access to the MCU's internal memory map, and access to the e200z4d internal registers during halt, via the JTAG port.

## 30.2 Block diagram

Figure 30-1 shows a functional block diagram of the NDI.

A simplified block diagram of the NDI illustrates the functionality and interdependence of major blocks (see Figure 30-2) and how the individual Nexus blocks are combined to form the NDI.

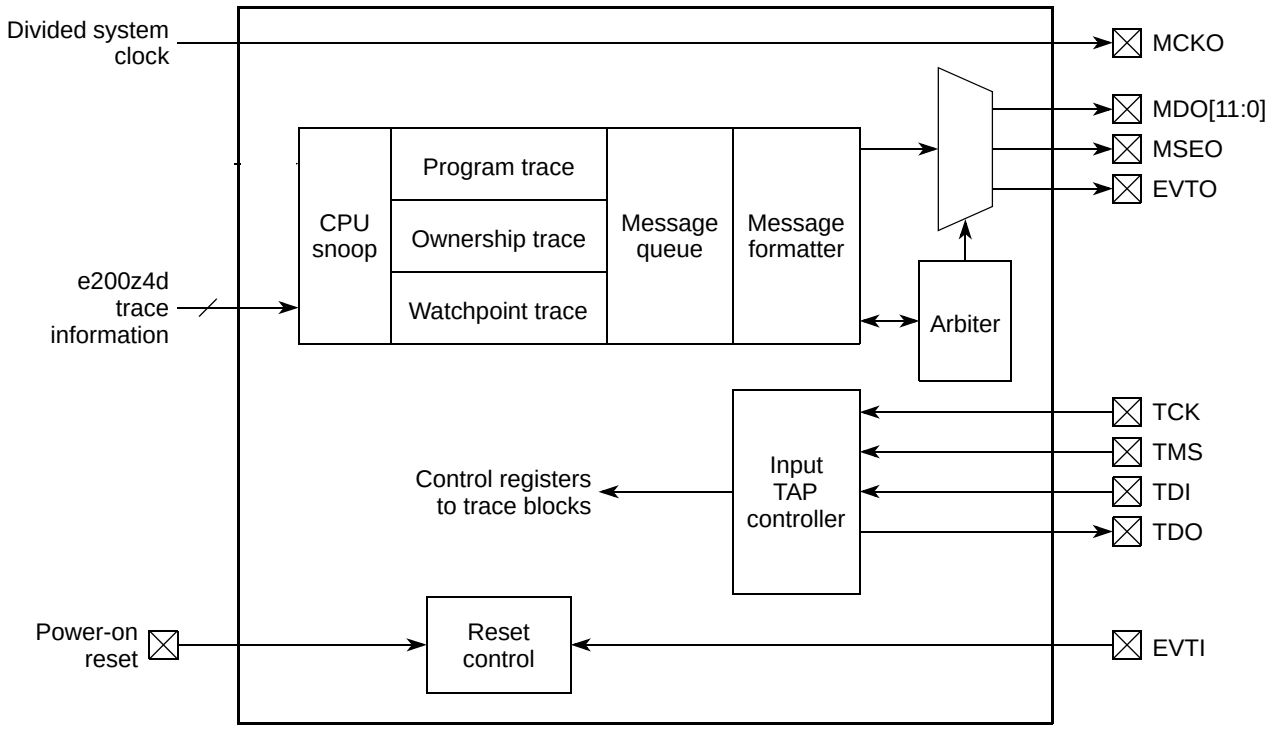



**Figure 30-1. NDI block diagram**

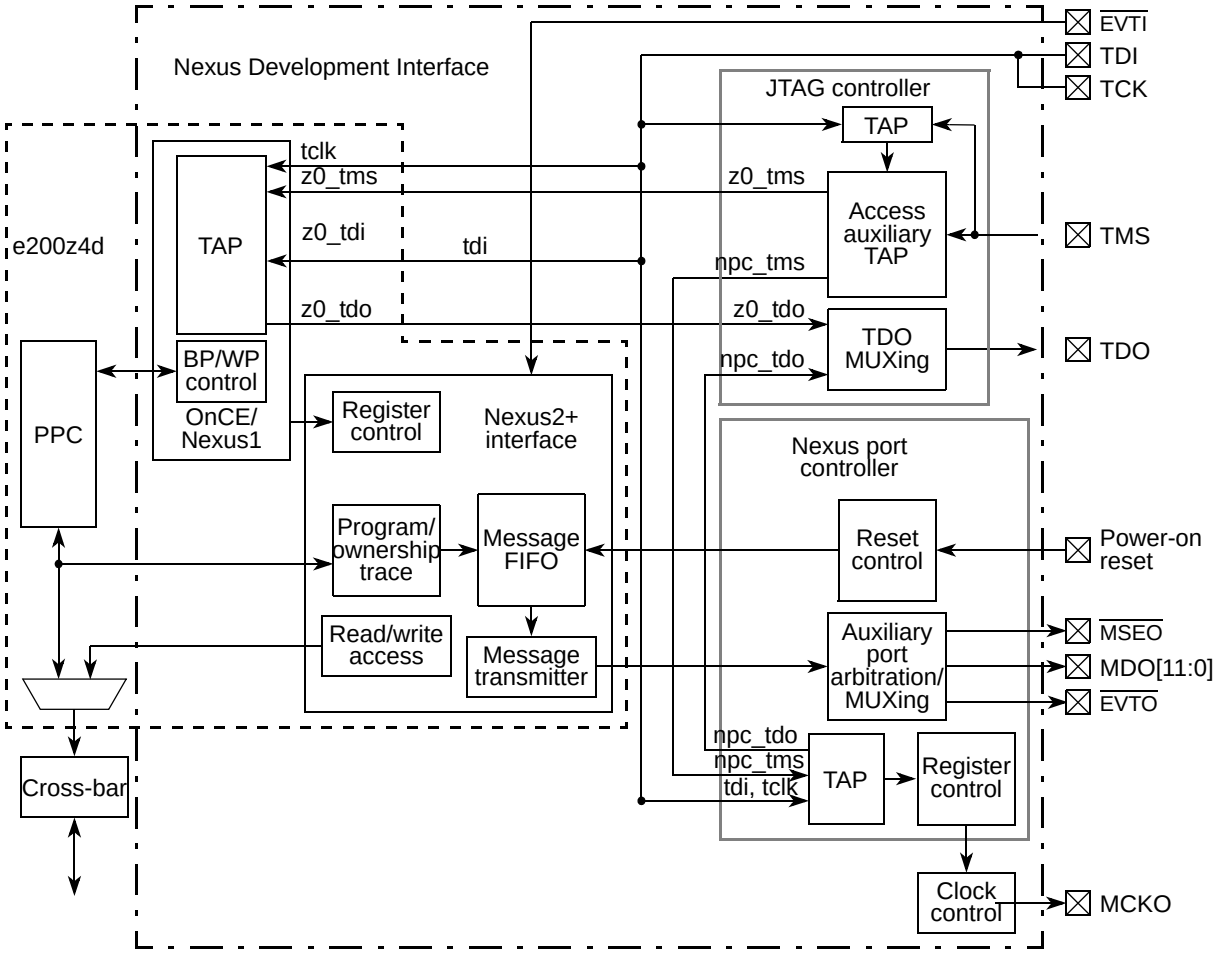



**Figure 30-2. NDI Implementation block diagram**

## 30.3 Features

The NDI module of the MPC5645S is compliant with Class 3 of the IEEE-ISTO 5001-2003 standard, with additional Class 4 features available.The following features are implemented:

- Program trace via branch trace messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Ownership trace via ownership trace messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An ownership trace message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Watchpoint messaging via the auxiliary pins.
- Watchpoint trigger enable of program trace messaging.
- Auxiliary interface for higher data input/output.
  - — Four message data out pins.

- One message start/end out pins (MSEO).
- One watchpoint event pin (EVTO).
- One event-in pin (EVTI).
- One message clock out pin (MCKO).
- Four pin JTAG port (TDI, TDO, TMS, and TCK).

- Registers for program trace, ownership trace, and watchpoint trigger.
- All features controllable and configurable via the JTAG port.
- Run-time access to the on-chip memory map via the Nexus read/write access protocol. This allows for enhanced download/upload capabilities.
- All features are independently configurable and controllable via the IEEE 1149.1 I/O port.
- The NDI block reset is controlled with JCOMP, power-on reset, and the TAP state machine. All these sources are independent of system reset.
- Support for internal censorship mode to prevent external access to flash memory contents when censorship is enabled.

**NOTE**

If the e200z4d core has executed a wait instruction, then the Nexus2+ controller clocks are gated off. While the core is in this state, it is not be possible to perform Nexus read/write operations.

## 30.4 Modes of operation

The NDI block is in reset when the TAP controller state machine is in the TEST-LOGIC-RESET state. The TEST-LOGIC-RESET state is entered on the assertion of the power-on reset signal or through state machine transitions controlled by TMS. Ownership of the TAP is achieved by loading the appropriate enable instruction for the desired Nexus client in the JTAGC controller (JTAGC) block.

The NPC transitions out of the reset state immediately following negation of power-on reset.

### 30.4.1 Nexus reset

In Nexus reset mode, the following actions occur:

- Register values default back to their reset values.
- The message queues are marked as empty.
- The auxiliary output port pins are negated if the NDI controls the pads.
- The TDO output buffer is disabled if the NDI has control of the TAP.
- The TDI, TMS, and TCK inputs are ignored.
- The NDI block indicates to the MCU that it is not using the auxiliary output port. This indication can be used to three-state the output pins or use them for another function.

### 30.4.2 Operating mode

In full-port mode, all available MDO pins are used to transmit messages. All trace features are enabled or can be enabled by writing the configuration registers via the JTAG port. Four MDO pins are available in full-port mode.

#### 30.4.2.1 Disabled-Port Mode

In disabled-port mode, message transmission is disabled. Any debug feature that generates messages can not be used. The primary features available are class 1 features and read/write access.

#### 30.4.2.2 Censored Mode

The NDI supports internal flash censorship mode by preventing the transmission of trace messages and Nexus access to memory-mapped resources when censorship is enabled.

#### 30.4.2.3 Stop Mode

Stop mode logic is implemented in the Nexus port controller (NPC). When a request is made to enter stop mode, the NDI block completes monitoring of any pending bus transaction, transmits all messages already queued, and acknowledges the stop request. After the acknowledgment, the system clock input are shut off by the clock driver on the device. While the clocks are shut off, the development tool cannot access NDI registers via the JTAG port.

## 30.5 External signal description

All the signals are available in the 416 TEPBGA without any multiplexing scheme. Refer to Chapter 3, “Signal Description,” for details.

### 30.5.1 Nexus Signal Reset States

**Table 30-1. NDI Signal Reset State**

| Name Fu | nction | Nexus Reset State | Pull |
|---|---|---|---|
| EVTI | Event-in pin | — | Up |
| $\overline{\text{EVTO}}$ | Event-out pin | 0b1 | — |
| MCKO | Message clock out pin | 0b0 | — |
| MDO[11:0] | Message data out pins | 0 | — |
| MSEO | Message start/end out pin | 0b1 | — |

## 30.6 Memory map and register description

The NDI block contains no memory-mapped registers. Nexus registers are accessed by a development tool via the JTAG port using a client-select value and a register index. OnCE registers are accessed by loading the appropriate value in the RS[0:6] field of the OnCE command register (OCMD) via the JTAG port.

## 30.6.1 Nexus Debug Interface Registers

Table 30-2 shows the NDI registers by client select and index values. OnCE register addressing is documented in Chapter 24, "IEEE 1149.1 Test Access Port Controller (JTAGC)."

**Table 30-2. Nexus Debug Interface Registers**

| Client Select | Index | Register |
|---|---|---|
| **Client-Independent Registers** | | |
| 0bxxxx | 0 | Device ID (DID)[1] |
| 0bxxxx | 127 | Port configuration register (PCR)[1] |
| **e200z4d Control/Status Registers** | | |
| 0b0000 | 2 | e200z4d development control1 (DC1) |
| 0b0000 | 3 | e200z4d development control2 (DC2) |
| 0b0000 | 4 | e200z4d development status (DS) |
| 0b0000 | 7 | Read/write access control/status (RWCS) |
| 0b0000 | 9 | Read/write access address (RWA) |
| 0b0000 | 10 | Read/write access data (RWD) |
| 0b0000 | 11 | e200z4d watchpoint trigger (PPC_WT) |

1 Implemented in NPC block. All other registers implemented in e200z4d Nexus3 block.

## 30.6.2 Register Description

This section lists the NDI registers and describes the registers and their bit fields.

### 30.6.2.1 Nexus Device ID Register (DID)

The NPC device identification register, shown in Figure 30-3, allows the part revision number, design center, part identification number, and manufacturer identity code of the device to be determined through the auxiliary output port, and serially through TDO. This register is read-only.

Reg Index: 0 Access: User read only

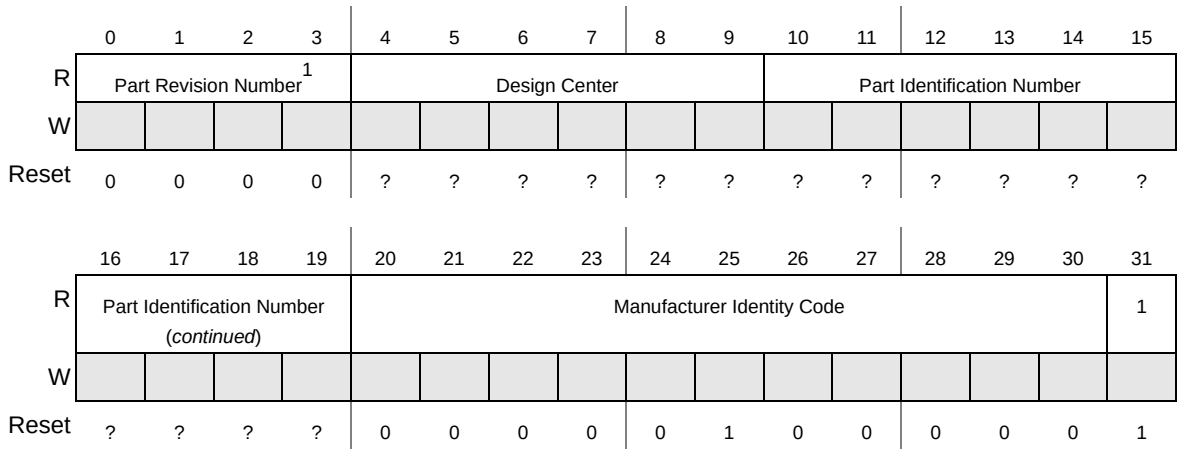



**Figure 30-3. Nexus Device ID Register (DID)**

1 Part Revision Number default value is 0x0 for the device's initial mask set and changes for each mask set revision.

**Table 30-3. DID Field Descriptions**

| Field Description | |
|---|---|
| 0–3<br>PRN | Part Revision Number. Contains the revision number of the part. This field changes with each revision of the device or module. |
| 4–9<br>DC | Design center. |
| 10–19<br>PIN | Part identification number. Contains the part number of the device. |
| 20–30<br>MIC | Manufacturer identity code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID: 0x20. |
| 31 | Fixed Per JTAG 1149.1. Always set to 1. |

### 30.6.2.2 Port Configuration Register (PCR)

The PCR is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NDI is enabled.

The PCR register may be rewritten by the debug tool subsequent to the enabling of the NPC for low power debug support. In this case, the debug tool may set and clear the LP_DBG_EN, SLEEP_SYNC, and STOP_SYNC bits, but must preserve the original state of the remaining bits in the register.

**NOTE**

The mode or clock division must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

**NOTE**

For any write to be possible in npc PCR register, TCK frequency should be less than the IP clock frequency.

Reg Index: 127 Access: User read/write

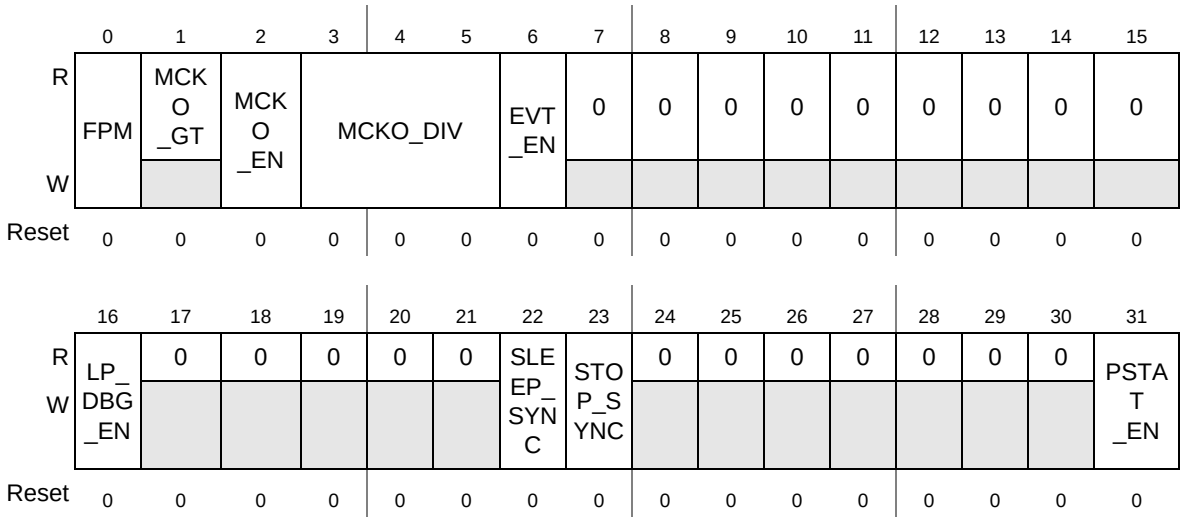



**Figure 30-4. Port Configuration Register (PCR)**

**Table 30-4. PCR field descriptions**

| Field De | scription |
|---|---|
| FPM | Full Port Mode.The value of the FPM bit determines if the auxiliary output port uses the full MDO port or a reduced MDO port to transmit messages.<br>0 A subset of MDO pins are used to transmit messages.<br>1 All MDO pins are used to transmit messages. |
| MCKO_EN | MCKO Enable. This bit enables the MCKO clock to run. When enabled, the frequency of MCKO is determined by the MCKO_DIV field.<br>0 MCKO clock is driven to zero.<br>1 MCKO clock is enabled. |
| MCKO_DIV | MCKO Division Factor. The value of this signal determines the frequency of MCKO relative to the system clock frequency when MCKO_EN is asserted. In this table, SYS_CLK represents the system clock frequency. |

| MCKO_DIV[2:0] M | CKO Frequency |
|---|---|
| 0b000 | SYSCLK÷1 |
| 0b001 | SYSCLK÷2 |
| 0b010 | Reserved |
| 0b011 | SYS_CLK÷4 |
| 0b100 | Reserved |
| 0b101 | Reserved |
| 0b110 | Reserved |
| 0b111 | SYS_CLK÷8 |

**Table 30-4. PCR field descriptions (continued)**

| Field De | scription |
|---|---|
| EVT_EN | EVTO/EVTI Enable. This bit enables the EVTO/EVTI port functions.<br>0 EVTO/EVTI port disabled.<br>1 EVTO/EVTI port enabled. |
| LP_DBG_EN | Low Power Debug Enable. The LP_DBG_EN bit enables debug functionality to support entry and exit from low power sleep and stop modes.<br>0 Low power debug disabled.<br>1 Low power debug enabled.<br>**Note:** See also Section 30.8.1, “Relationship between TCK and system clock frequency.” |
| SLEEP_ SYNC | Sleep Mode Synchronization. The SLEEP_SYNC bit is used to synchronize the entry into sleep mode between the device and debug tool. The device sets this bit before a pending entry into sleep mode. After reading SLEEP_SYNC as set, the debug tool then clears SLEEP_SYNC to acknowledge to the device that it may enter into sleep mode.<br>0 Sleep mode entry acknowledge.<br>1 Sleep mode entry pending.<br>**Note:** See also Section 30.8.1, “Relationship between TCK and system clock frequency.” |
| STOP_SYNC | Stop Mode Synchronization. The STOP_SYNC bit is used to synchronize the entry into stop mode between the device and debug tool. The device sets this bit before a pending entry into stop mode. After reading STOP_SYNC as set, the debug tool then clears STOP_SYNC to acknowledge to the device that it may enter into stop mode.<br>0 Stop mode entry acknowledge.<br>1 Stop mode entry pending |
| PSTAT_EN | Processor Status Mode Enable |

### 30.6.2.3 Development Control Register 1, 2 (DC1, DC2)

The development control registers are used to control the basic development features of the Nexus module. Figure 30-5 shows development control register 1 and Table 30-5 describes the register’s fields.

Nexus Reg: 0x0002 — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OPC | MCK_DIV | | EOC | | 0 | PTM | WEN | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | OVC | | | EIC | | TM | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 30-5. Development Control Register 1 (DC1)**

**Table 30-5. DC1 Field Descriptions**

| Field Des | cription |
|---|---|
| 0<br>OPC[1] | Output Port Mode Control.<br>0 Reduced-port mode configuration (2 MDO pins).<br>1 Full-port mode configuration (all MDO pins). |
| 1–2<br>MCK_DIV[1:0][1] | MCKO Clock Divide Ratio (see note 1).<br>00 MCKO is 1x processor clock freq.<br>01 MCKO is 1/2x processor clock freq.<br>10 MCKO is 1/4x processor clock freq.<br>11 MCKO is 1/8x processor clock freq. |
| 3–4<br>EOC[1:0] | EVTO Control.<br>00 EVTO upon occurrence of watchpoints (configured in DC2).<br>01 EVTO upon entry into debug mode.<br>10 EVTO upon timestamping event.<br>11 Reserved. |
| 5 | Reserved. |
| 6<br>PTM | Program Trace Method.<br>0 Program trace uses traditional branch messages.<br>1 Program trace uses branch history messages. |
| 7<br>WEN | Watchpoint Trace Enable.<br>0 Watchpoint messaging disabled.<br>1 Watchpoint messaging enabled. |
| 8–23 | Reserved. |
| 24–26<br>OVC[2:0] | Overrun Control.<br>000 Generate overrun messages.<br>001–010 Reserved.<br>011 Delay processor for BTM / DTM / OTM overruns.<br>1XX Reserved. |
| 27–28<br>EIC[1:0] | EVTI Control.<br>00 EVTI is used for synchronization (program trace/ data trace).<br>01 EVTI is used for debug request.<br>1X Reserved. |
| 29–31<br>TM[2:0] | Trace Mode. Any or all of the TM bits may set, enabling one or more traces.<br>000 No trace.<br>1XX Program trace enabled.<br>X1X Data trace enabled (not supported mode)<br>XX1 Ownership trace enabled. |

1 The output port mode control bit (OPC) and MCKO divide bits (MCK_DIV) are shown for clarity. These functions are controlled globally by the NPC port control register (PCR). These bits are writable in the PCR but have no effect.

Development control register 2 is shown in Figure 30-6 and its fields are described in Table 30-6.

Nexus Reg: 0x0003 Access: User read/write

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

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

**Figure 30-6. Development Control Register 2 (DC2)**

**Table 30-6. DC2 Field Descriptions**

| Field De | scription |
|---|---|
| 0–7<br>EWC[7:0] | EVTO Watchpoint Configuration. Any or all of the bits in EWC may be set to configure the EVTO watchpoint.<br>00000000 No Watchpoints trigger EVTO<br>1XXXXXXX Watchpoint #0 (IAC1 from Nexus1) triggers EVTO.<br>X1XXXXXX Watchpoint #1 (IAC2 from Nexus1) triggers EVTO.<br>XX1XXXXX Watchpoint #2 (IAC3 from Nexus1) triggers EVTO.<br>XXX1XXXX Watchpoint #3 (IAC4 from Nexus1) triggers EVTO.<br>XXXX1XXX Watchpoint #4 (DAC1 from Nexus1) triggers EVTO.<br>XXXXX1XX Watchpoint #5 (DAC2 from Nexus1) triggers EVTO.<br>XXXXXX1X Watchpoint #6 (DCNT1 from Nexus1) triggers EVTO.<br>XXXXXXX1 Watchpoint #7 (DCNT2 from Nexus1) triggers EVTO. |
| 8–31 | Reserved. |

**NOTE**

The EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence for the EWC bits to have any effect.

### 30.6.2.4 Development Status Register (DS)

The development status register is used to report system debug status. When debug mode is entered or exited, or a CPU-defined low-power mode is entered, a debug status message is transmitted with DS[31:24]. The external tool can read this register at any time.

Nexus Reg: 0x0004 Access: User read only

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DBG | LPS | | | LPC | | CHK | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 30-7. Development Status Register (DS)**

**Table 30-7. DS Field Descriptions**

| Field De | scription |
|---|---|
| 0<br>DBG | CPU Debug Mode Status.<br>0 CPU not in debug mode.<br>1 CPU in debug mode. |
| 1–3<br>LPS | System Low Power Status<br>000 Normal (run) mode<br>xx1 Doze mode<br>x1x Nap mode<br>1xx Sleep mode |
| 4–5<br>LPC[1:0] | CPU Low-Power Mode Status.<br>00 Normal (run) mode.<br>01 CPU in halted state.<br>10 CPU in stopped state.<br>11 Reserved. |
| 6<br>CHK | CPU Checkstop Status.<br>0 CPU not in checkstop state.<br>1 CPU in checkstop state. |
| 7–31 | Reserved. |

### 30.6.2.5 Read/Write Access Control/Status (RWCS)

The read write access control/status register provides control for read/write access. Read/write access provides DMA-like access to memory-mapped resources on the system bus while the processor is halted or during runtime. The RWCS register also provides read/write access status information as shown in Table 30-9.

Nexus Reg: 0x0007 — Access: User read/write

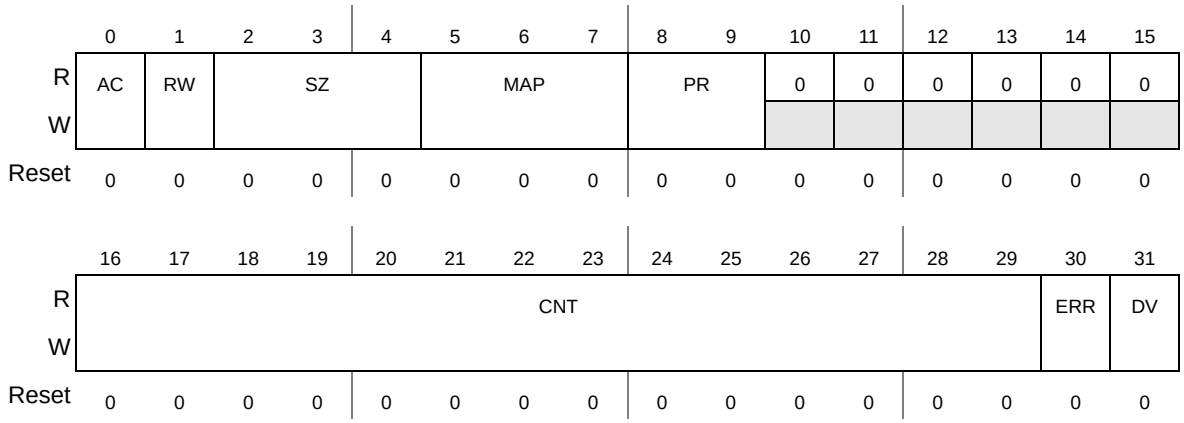



**Figure 30-8. Read/Write Access Control/Status Register (RWCS)**

**Table 30-8. RWCS Field Description**

| Field De | scription |
|---|---|
| 0<br>AC | Access Control.<br>0 End access.<br>1 Start access. |
| 1<br>RW | Read/Write Select.<br>0 Read access.<br>1 Write access. |
| 2–4<br>SZ[2:0] | Word Size.<br>000 8-bit (byte.)<br>001 16-bit (halfword).<br>010 32-bit (word).<br>011 64-bit (doubleword—only in burst mode).<br>100–111 Reserved. |
| 5–7<br>MAP[2:0] | MAP Select.<br>000 Primary memory map.<br>001-111 Reserved. |
| 8–9<br>PR[1:0] | Read/Write Access Priority.<br>00 Lowest access priority.<br>01 Reserved (default to lowest priority).<br>10 Reserved (default to lowest priority).<br>11 Highest access priority. |
| 10–15 | Reserved. |
| 16–31<br>CNT[13:0] | Access Control Count. Number of accesses of word size SZ. |
| 30<br>ERR | Read/Write Access Error. See Table 30-9. |
| 31<br>DV | Read/Write Access Data Valid. See Table 30-9. |

Table 30-9 details the status bit encodings.

**Table 30-9. Read/Write Access Status Bit Encoding**

| Read Action | Write Action | ERR | DV |
|---|---|---|---|
| Read access has not completed | Write access completed without error | 0 | 0 |
| Read access error has occurred | Write access error has occurred | 1 | 0 |
| Read access completed without error | Write access has not completed | 0 | 1 |
| Not allowed | Not allowed | 1 | 1 |

### 30.6.2.6 Read/Write Access Address (RWA)

The read/write access address register provides the system bus address to be accessed when initiating a read or a write access.

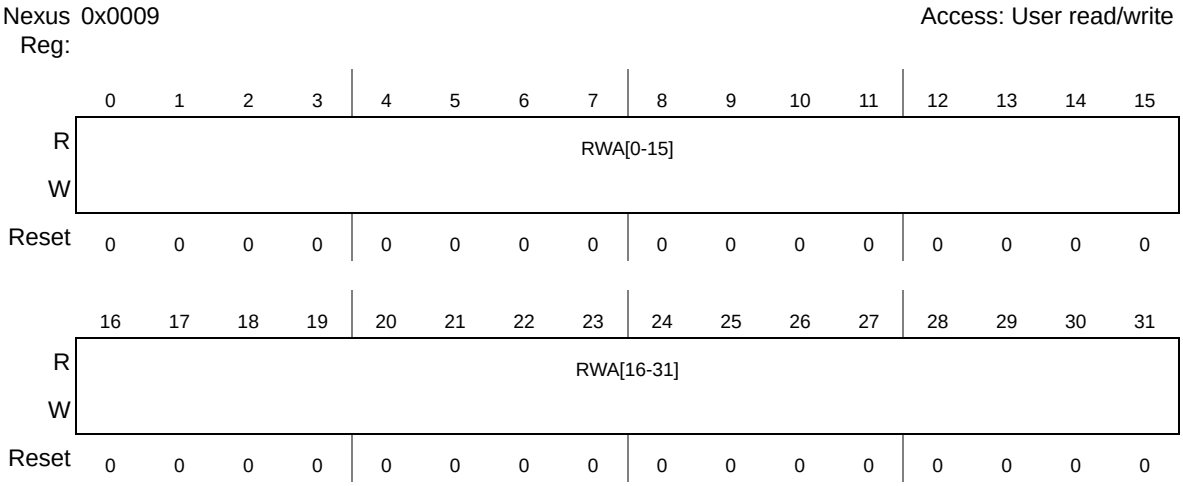



**Figure 30-9. Read/Write Access Address Register (RWA)**

### 30.6.2.7 Read/Write Access Data (RWD)

The read/write access data register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

Nexus Reg: 0x000A — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RWD[0-15] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RWD[16-31] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 30-10. Read/Write Access Data Register (RWD)**

### 30.6.2.8 Watchpoint Trigger Register (WT)

The watchpoint trigger register allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control program and/or data trace enable and disable. The WT bits can be used to produce an address-related window for triggering trace messages.

Nexus Reg: 0x000B — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PTS | | | | PTE | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 30-11. Watchpoint Trigger Register (WT)**

Table 30-10 details the watchpoint trigger register fields.

**Table 30-10. WT Field Descriptions**

| Field De | scription |
|---|---|
| 0–2<br>PTS[2:0] | Program Trace Start Control.<br>000 Trigger disabled.<br>001 Use watchpoint #0 (IAC1 from Nexus1).<br>010 Use watchpoint #1 (IAC2 from Nexus1).<br>011 Use watchpoint #2 (IAC3 from Nexus1).<br>100 Use watchpoint #3 (IAC4 from Nexus1).<br>101 Use watchpoint #4 (DAC1 from Nexus1).<br>110 Use watchpoint #5 (DAC2 from Nexus1).<br>111 Use watchpoint #6 or #7 (DCNT1 or DCNT2 from Nexus1). |
| 3–5<br>PTE[2:0] | Program Trace End Control.<br>000 Trigger disabled.<br>001 Use watchpoint #0 (IAC1 from Nexus1).<br>010 Use watchpoint #1 (IAC2 from Nexus1).<br>011 Use watchpoint #2 (IAC3 from Nexus1).<br>100 Use watchpoint #3 (IAC4 from Nexus1).<br>101 Use watchpoint #4 (DAC1 from Nexus1).<br>110 Use watchpoint #5 (DAC2 from Nexus1).<br>111 Use watchpoint #6 or #7 (DCNT1 or DCNT2 from Nexus1). |
| 12–31 | Reserved. |

## 30.7 Functional Description

The NDI block is implemented by integrating the following blocks on this device:

- Nexus e200z4d development interface (OnCE and Nexus3 subblocks)
- Nexus port controller (NPC) Block
- NPC_HNDSHK module

### 30.7.1 NPC_HNDSHK module

This module enables debug entry/exit across low power modes(Stop, Halt, standby) .

The NPC_HNDSHK supports:

- Setting and clearing of the NPC PCR sync bit on low-power mode entry and exit
- Putting the CPU into debug mode on low-power mode exit
- Generating a falling edge on the JTAG TDO pad on low-power mode exit

On HALT, STOP, or STANDBY mode entry, the MC_ME asserts the lp_mode_entry_req input after the clock disable process has completed and before the processor enters its halted or stopped state. The mode transition will then not proceed until the lp_mode_entry_ack output has been asserted. The notification to the debugger of a low-power mode entry consists of setting the low-power mode handshake bit in the port control register (read by the debugger) via the lp_sync_in output. The debugger acknowledges that the transition into a low-power mode may proceed by clearing the low-power mode handshake bit in the port control register (written by the debugger), which results in the deassertion of the lp_sync_out input.

After entry into low power mode the TDO pad becomes high impedance. On exit from low power mode it is forced to 0, so for correct operation the TDO pin should be pulled up while in low power modes.

On HALT or STOP mode exit, the MC_ME asserts the lp_mode_exit_req input after ensuring that the regulator and memories are in normal mode and before the processor exits its halted or stopped state. The mode transition will then not proceed until the lp_mode_exit_ack output has been asserted. The MC_RGM asserts the exit_from_standby input when executing a reset sequence due to a STANDBY exit. The reset sequence will then not complete until the lp_mode_exit_ack output has been asserted.

The notification to the debugger of a low-power mode exit consists of driving the TDO pad to `0'. The debugger acknowledges that the transition from a low-power mode can continue by setting the low-power mode sync bit in the port control register (written by debugger), which results in the assertion of the lp_sync_out input.

**NOTE**

The debugger clock multiplexer may not guarantee glitch free switching. Therefore, TCK should be disabled from when the debugger clears the sync bit in ENTRY_CLR until the debugger senses the falling edge of TDO in TDO_SET.

## 30.7.2 Enabling Nexus Clients for TAP Access

After the conditions have been met to bring the NDI out of the reset state, the loading of a specific instruction in the JTAG controller (JTAGC) block is required to grant the NDI ownership of the TAP. Each Nexus client has its own JTAGC instruction opcode for ownership of the TAP, granting that client the means to read/write its registers. The JTAGC instruction opcode for each Nexus client is shown in Table 30-11. After the JTAGC opcode for a client has been loaded, the client is enabled by loading its NEXUS-ENABLE instruction. The NEXUS-ENABLE instruction opcode for each Nexus client is listed in Table 30-12. Opcodes for all other instructions supported by Nexus clients can be found in the relevant sections of this chapter.

**Table 30-11. JTAGC Instruction Opcodes to Enable Nexus Clients**

| JTAGC Instruction | Opcode | Description |
|---|---|---|
| ACCESS_AUX_TAP_NPC | 10000 | Enables access to the NPC TAP controller. |
| ACCESS_AUX_TAP_ONCE | 10001 | Enables access to the e200z4d TAP controller. |

**Table 30-12. Nexus Client JTAG Instructions**

| Instruction D | escription | Opcode |
|---|---|---|
| **NPC JTAG Instruction Opcodes** | | |
| NEXUS_ENABLE | Opcode for NPC Nexus ENABLE instruction (4-bits) | 0x0 |
| BYPASS | Opcode for the NPC BYPASS instruction (4-bits) | 0xF |
| **e200z0 OnCE JTAG Instruction Opcodes[1]** | | |

**Table 30-12. Nexus Client JTAG Instructions**

| Instruction D | escription | Opcode |
|---|---|---|
| NEXUS2_ACCESS | Opcode for e200z4d OnCE Nexus ENABLE instruction (10-bits) | 0x7C |
| BYPASS | Opcode for the e200z4d OnCE BYPASS instruction (10-bits) | 0x7F |

1 See the e200z4d core reference manual for a complete list of available OnCE instructions.

## 30.7.3 Configuring the NDI for Nexus Messaging

The NDI is placed in disabled mode upon exit of reset. If message transmission via the auxiliary port is desired, a write to the port configuration register (PCR) located in the NPC is then required to enable the NDI and select the mode of operation. Asserting MCKO_EN in the PCR places the NDI in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively. When writing to the PCR, the PCR LSB must be written to a logic zero. Setting the LSB of the PCR enables factory debug mode and prevents the transmission of Nexus messages.

Table 30-13 describes the NDI configuration options.

**Table 30-13. NDI Configuration Options**

| JCOMP Asserted | MCKO_EN bit of the Port Configuration Register | FPM bit of the Port Configuration Register | Configuration |
|---|---|---|---|
| No | X | X | Reset |
| Yes | 0 | X | Disabled |
| Yes | 1 | 1 | Full-port mode |
| Yes | 1 | 0 | Reduced-port mode |

## 30.7.4 Programmable MCKO Frequency

MCKO is an output clock to the development tools used for the timing of MSEO and MDO pin functions. MCKO is derived from the system clock, and its frequency is determined by the value of the MCKO_DIV field in the port configuration register (PCR) located in the NPC. Possible operating frequencies include one-quarter and one-eighth system clock speed.

Figure 30-14 shows the MCKO_DIV encodings. In this table, SYS_CLK represents the system clock frequency. The default value selected if a reserved encoding is programmed is SYS_CLK÷2

**NOTE**

On MPC5645S, the pad type used for the Nexus3 signals will not support the default SYSCLK÷2 and SYSCLK÷1 setting, so the user must change the MCKO frequency to be not faster than SYSCLK÷4.

**Table 30-14. MCKO_DIV Values**

| MCKO_DIV[2:0] MCKO | Frequency |
|---|---|
| 0b000 | SYSCLK÷1 |
| 0b001 | SYSCLK÷2 |
| 0b010 | Reserved |
| 0b011 | SYS_CLK÷4 |
| 0b100 | Reserved |
| 0b101 | Reserved |
| 0b110 | Reserved |
| 0b111 | SYS_CLK÷8 |

## 30.7.5 Nexus Messaging

Most of the messages transmitted by the NDI include a SRC field. This field is used to identify which source generated the message. Table 30-15 shows the values used for the SRC field by the different clients on the MPC5645S. These values are specific to the MPC5645S. The size of the SRC field in transmitted messages is 4 bits. This value is also specific to the MPC5645S.

**Table 30-15. SRC Packet Encodings**

| SRC[3:0] MPC564 | 5S Client |
|---|---|
| 0b0000 | e200z04d |
| All other combinations | Reserved |

## 30.7.6 EVTO Sharing

The NPC block controls sharing of the EVTO output between all Nexus clients that generate an EVTO signal. The sharing mechanism is a logical AND of all incoming EVTO signals from Nexus blocks, thereby asserting EVTO whenever any block drives its EVTO. When there is no active MCKO, such as in disabled mode, the NPC drives EVTO for two system clock periods. EVTO sharing is active as long as the NDI is not in reset.

## 30.7.7 Debug Mode Control

On MPC5645S, program breaks can be requested either by using the EVTI pin as a break request, or when a Nexus event is triggered.

### 30.7.7.1 EVTI Generated Break Request

To use the EVTI pin as a debug request, the EIC field in the e200z4d Nexus3 Development Control Register 1 (DC1[4:3]) must be set to configure the EVTI input as a debug request.

### 30.7.8 Nexus Reset Control

The JCOMP input that is used as the primary reset signal for the NPC is also used by the NPC to generate a single-bit reset signal for other Nexus blocks. The single bit reset signal functions much like the IEEE 1149.1-2001 defined $\overline{\text{TRST}}$ signal but has a default value of disabled (JCOMP is pulled low during reset) The IEEE 1149.1-2001 defines $\overline{\text{TRST}}$ to be pulled up (enabled) by default.

## 30.8 Initialization / application information

### 30.8.1 Relationship between TCK and system clock frequency

The JTAG clock (TCK) typically operates at a frequency well below the system clock frequency, as specified in the *MPC5645S Microcontroller Data Sheet*. In some cases, however, such as low power mode (if the device supports low power modes), the system clock frequency may be lowered significantly from the normal operating range. If the system clock frequency is reduced below the frequency of TCK, it will no longer be possible to communicate with the Nexus Port Controller Port Configuration Register (NPC_PCR).

Therefore, if the tool needs to update the NPC_PCR Low Power Debug Enable (NPC[PCR[LP_DBG_EN]) or Low Power Synchronization bits (NPC[PCR[LP_SYNC], the TCK clock frequency must be lowered.

This page is intentionally left blank.

# Chapter 31
# OpenVG Graphics Accelerator (GFX2D)

## 31.1 Introduction

The OpenVG Graphics Accelerator (GFX2D) is an embedded processor targeting the OpenVG 1.1 graphics API and feature set. The GFX2D has a rich, but well-chosen set of features, with emphasis being on very high image quality and low memory bandwidth consumption.

The GFX2D has a geometry engine unit for operations on primitives such as lines or curves specified with absolute or relative coordinates in a user specified system. It also has 2D bitmap acceleration unit for graphics operations such as bit-block transfers, block fill, dithering and raster operations. A pipe-lined unit calculates color and alpha values for pixels from pixel coordinates and alpha according to the required gradient or texturing mode. A separate anti-aliasing polygon rasterizer is connected to the 2D graphics acceleration unit for vector graphics rendering acceleration.

### 31.1.1 Block Diagram

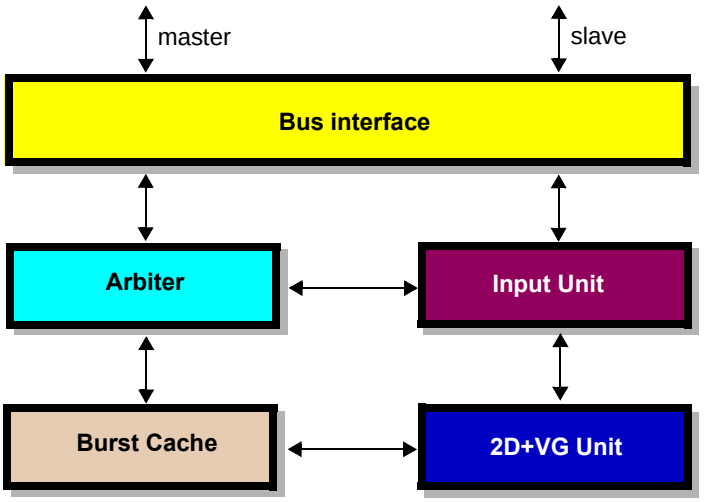



**Figure 31-1. GFX2D Block Diagram**

## 31.1.2 Features

The following describes the functional features of the graphics processor.

### 31.1.2.1 Frame buffer

- Frame buffer sizes supported up-to 2048x2048
- ARGB4444, RGB565, ARGB1555, ARGB5551, ARGB8888 frame buffer modes
- Programmable ARGB order in frame buffer: ARGB, BGRA, ABGR, RGBA
- Linear and block-based (4x4 pixels) frame buffer modes
- Fast buffer clears:
  - Write large bursts of buffer clear data to the system memory from the bus interface unit without the aid of the 2D/VG rendering pipeline
- Support for OpenVG render to Image

### 31.1.2.2 2D bitmap graphics (separate 2D unit)

- Parallel operation with the 3D pipeline, independent command input
- BitBlt (surface-to-surface copy)
  - Format conversion from monochrome/ARGB/YUV to ARGB during BitBlt
- Block fill
- Internal 32-bit color precision
- Source bitmap format:
  - 1/4/8-bit monochrome
  - ARGB4444, RGB565, ARGB1555, ARGB5551, ARGB8888
  - Programmable ARGB order: ARGB, BGRA, ABGR, RGBA
  - Packed YUV 4:2:2 formats (FOURCC codes YUY2, UYVY, YVYU), two pixels per 32 bits of data
  - 1-bit bitmap maps to foreground and background colors
  - 4-bit bitmap is optionally gamma corrected to 8-bit alpha values and can be combined with foreground color to draw anti-aliased fonts
- Destination bitmap format:
  - ARGB4444, RGB565, ARGB1555, ARGB5551, ARGB8888, B8, A8, AB88
  - Programmable ARGB order: ARGB, BGRA, ABGR, RGBA
- Supports three source bitmaps for separate mask/pattern/alpha bitmap support plus reading destination for ROP, blend and color key operations
- Supports masking source coordinates for wrapping patterns
- Supports ROP4 (ROP3 with separate ROPs for masked and unmasked pixels) logical operations
- Supports inverting mask and alpha values from source
- Supports destination rotation by 0/90/180/270 degrees
- Supports programmable blending with optional alpha inverse premultiply

- Supports per pixel and constant alpha with optional modulation by source color alpha for OpenVG alpha masking
- Supports color keying by source and destination colors, with optional ignoring of alpha channel
- Supports one scissor rectangle for destination coordinates
- Dithering (ordered)
- Color component masking
- RGB reads and writes
- Non-power of two source and destination bitmap sizes supported (stride must be a multiple of 32-bits)
- BitBlt with scaling implemented with the 3D rendering pipeline, bilinear filtering with texture lookups, programmable filter kernels possible with the programmable Pixel processor

#### 31.1.2.3 Vector graphics

- Parallel operation with the 3D pipeline, independent command input
- Rasterization of convex and concave polygons with anti-aliasing
- Efficient native polygon rendering (no tessellation to triangles)
- Non-zero and odd-even fill rules
- Primitives supported:
  - Polygons
  - OpenVG path primitives (except Elliptical Arcs): Horizontal/vertical lines, generic lines, curves, smooth curves, moveto, path closing
  - Curve types supported: cubic and quadratic Bézier
  - Strokes with thickness, joints and end caps, unlimited stroke thickness
  - Special case handling of singularities for thick strokes
  - Supports paths with a maximum of 256 crossings along a horizontal or a vertical line
- Input coordinates:
  - Absolute and relative coordinate input in floating point
  - Fixed-point (byte, short, int) and floating-point coordinate input - 0.8, 0.16, 16.16 formats
  - Little- and Big-endian support separately selectable for command stream and data.
- Geometry
  - User to surface transform for vertices and stroke shape
  - Hardware curve tessellation
  - Adjustable accuracy for curve and round cap splitting
  - OpenVG/SVG join types: Miter (with miter limit), round, bevel
  - OpenVG/SVG cap types: Butt, round, square
- Pixel processing:
  - Programmable gradient and texturing processor
  - Linear and radial gradients (with focal point)

- Perspective texture mapping with filtering
- Two textures supported
- sRGB and pre-multiply support for textures
- 16-sample anti-aliasing
- 4x RGSS AA (Rotated Grid Super Sample)
- Per-pixel alpha-masking
- Maximum texture size: 1024x1024 pixels

- Vector graphics rendering system CPU load:
  - Display list generation during path creation ñ commands and vertices are stored to an internal format/buffer, no format conversion is performed
  - Filling or stroking a path only requires a few register writes to start the operation in hardware
  - Display lists are transferred to the vector graphics rasterizer using DMA without CPU interaction

## 31.2 External Signal Description

The GFX2D does not include any external signals.

## 31.3 Memory Map and Register Definition

This section provides a detailed description of the GFX2D external registers that are mapped to aperture.

The GFX2D 16K space memory map is shown in Table 31-1. The address of each register is given as an offset to the GFX2D base address.

**Table 31-1. GFX2D Memory Map**

| Offset | Register | Access | Reset Value | Location |
|---|---|---|---|---|
| 0x000 | G12_COMMANDSTREAM—Command Stream Input | W | 0x0000_0000 | on page 1141 |
| 0x3fc | G12_MMUCOMMANDSTREAM—MMU Command Stream Input | W | 0x0000_0000 | on page 1142 |
| 0x400 | G12_REVISION—Revision | R | 0x0000_0000 | on page 1142 |
| 0x410 | G12_SYSSTATUS—System Status | R | 0x0000_0000 | on page 1143 |
| 0x418 | G12_IRQSTATUS—Interrupt Status | R/W | 0x0000_0000 | on page 1143 |
| 0x438 | G12_IRQENABLE—Interrupt Enable | R/W | 0x0000_0000 | on page 1144 |
| 0x4e0 | G12_IRQ_ACTIVE_CNT—Active Interrupt Counters | R | 0x0000_0000 | on page 1144 |
| 0x508 | G12_CLOCKEN—Clock Enable | R/W | 0x0000_000F | on page 1145 |
| 0x510 | MMU_READ_ADDR—MMU Read Address | W | 0x0000_0000 | on page 1145 |
| 0x518 | MMU_READ_DATA—MMU Read Data | R | 0x0000_00FF | on page 1146 |
| 0x7c0 | G12_FIFOFREE—FIFO Free | R | 0x0000_0000 | on page 1146 |

### 31.3.1 G12_COMMANDSTREAM

Offset 0x000 Access: User write

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

**Figure 31-2. Command Stream input (G12_COMMANDSTREAM)**

**Table 31-2. Command Stream input (G12_COMMANDSTREAM) Field Descriptions**

| Field | Description |
|---|---|
| 31–0 | Command stream data |

## 31.3.2 G12_MMUCOMMANDSTREAM

Offset 0x3fc — Access: User write

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

**Figure 31-3. MMU Command Stream input (G12_MMUCOMMANDSTREAM)**

**Table 31-3. MMU Command Stream input (G12_MMUCOMMANDSTREAM) Field Descriptions**

| Field | Description |
|---|---|
| 31–0 | MMU Command stream data |

## 31.3.3 G12_REVISION

Offset 0x400 — Access: User read

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

**Figure 31-4. Revision (G12_REVISION)**

**Table 31-4. Revision (G12_REVISION) Field Descriptions**

| Field | Description |
|---|---|
| 31–8 | Reserved |
| 7–4 | Major revision ID |
| 3–0 | Minor revision ID |

## 31.3.4 G12_SYSSTATUS

Offset 0x410 Access: User read

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

**Figure 31-5. System Status (G12_SYSSTATUS)**

**Table 31-5. System Status (G12_SYSSTATUS) Field Descriptions**

| Field | Description |
|---|---|
| 31–1 | Reserved |
| 0 | Internal reset state |

## 31.3.5 G12_IRQSTATUS

Offset 0x418 Access: User read/write

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

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FBC | FIFO | G2D | MH |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 31-6. Interrupt Status (G12_IRQSTATUS)**

**Table 31-6. Interrupt Status (G12_IRQSTATUS) Field Description**

| Field | Description |
|---|---|
| 31–4 | Reserved. |
| 3 | FBC done interrupt status |
| 2 | FIFO full error interrupt status |
| 1 | 2D idle request interrupt status |
| 0 | MH AXI error / MMU page fault interrupt status |

## 31.3.6 G12_IRQENABLE

Offset 0x438 Access: User read/write

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

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FBC | FIFO | G2D | MH |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 31-7. Interrupt Enable (G12_IRQENABLE)**

**Table 31-7. Interrupt Enable (G12_IRQENABLE) Field Description**

| Field | Description |
|---|---|
| 31–4 | Reserved. |
| 3 | FBC done interrupt enable |
| 2 | FIFO full error interrupt enable |
| 1 | 2D idle request interrupt enable |
| 0 | MH AXI error / MMU page fault interrupt enable |

## 31.3.7 G12_IRQ_ACTIVE_CNT

Offset 0x4e0 Access: User read

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FBC | | | | | | | | ERROR | | | | | | | | G2D | | | | | | | | MH | | | | | | | |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 31-8. Active interrupt counters (G12_IRQ_ACTIVE_CNT)**

**Table 31-8. Active interrupt counters (G12_IRQ_ACTIVE_CNT) Field Descriptions**

| Field | Description |
|---|---|
| 31–24 | FBC done interrupt counter |
| 23–16 | FIFO full error interrupt counter |
| 15–8 | 2D idle request interrupt counter |
| 7–0 | MH AXI error / MMU page fault interrupt counter |

## 31.3.8 G12_CLOCKEN

Offset 0x508 Access: User read/write

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

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RESERVED | | | VG_ L1L2 | G2D_ VGL3 | BCA CHE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

**Figure 31-9. Clock Enable (G12_CLOCKEN)**

**Table 31-9. Clock Enable (G12_CLOCKEN) Field Description**

| Field | Description |
|---|---|
| 31–6 | Reserved |
| 5–3 | Legacy |
| 2 | Enable VG level 1 and level 2 clock |
| 1 | Enable 2D and VG level 3 clock |
| 0 | Enable burst cache clock |

## 31.3.9 MMU_READ_ADDR

Offset 0x510 Access: User write

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

**Figure 31-10. MMU Read Address (MMU_READ_ADDR)**

**Table 31-10. MMU Read Address (MMU_READ_ADDR) Field Descriptions**

| Field | Description |
|---|---|
| 31–15 | Reserved |
| 14–0 | Register Address |

## 31.3.10 MMU_READ_DATA

Offset 0x518 Access: User read

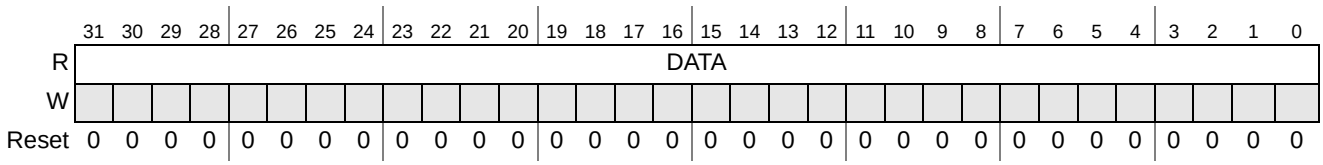



**Figure 31-11. MMU Read Data (MMU_READ_DATA)**

**Table 31-11. MMU Read Data (MMU_READ_DATA) Field Descriptions**

| Field | Description |
|---|---|
| 31–0 | Register Data |

## 31.3.11 G12_FIFOFREE

Offset 0x7c0 Access: User read

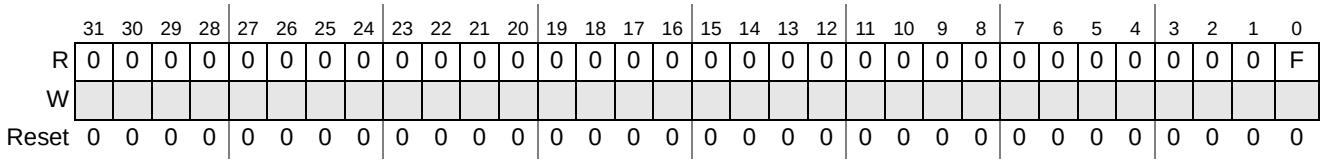



**Figure 31-12. FIFO Free (G12_FIFOFREE)**

**Table 31-12. FIFO Free (G12_FIFOFREE) Field Descriptions**

| Field | Description |
|---|---|
| 31–1 | Reserved |
| 0 | Free space in input fifo |

## 31.4 Command Stream registers

This section provides a detailed description of the GFX2D internal registers that are only accessible via G12_COMMANDSTREAM.

**Table 31-13. Command Stream Registers**

| Offset | Register | Access | Reset Value | Section/Page |
|---|---|---|---|---|
| 0x0 | G2D_BASE0 | W | 0x0000_0000 | 31.4.1/31-1154 |
| 0x1 | G2D_CFG0 | | | 31.4.2/31-1154 |
| 0x2 | G2D_BASE1 | | | 31.4.1/31-1154 |
| 0x3 | G2D_CFG1 | | | 31.4.2/31-1154 |
| 0x4 | G2D_BASE2 | | | 31.4.1/31-1154 |
| 0x5 | G2D_CFG2 | | | 31.4.2/31-1154 |
| 0x6 | G2D_BASE3 | | | 31.4.1/31-1154 |
| 0x7 | G2D_CFG3 | | | 31.4.2/31-1154 |
| 0x8 | G2D_SCISSORX | | | 31.4.3/31-1155 |
| 0x9 | G2D_SCISSORY | | | 31.4.4/31-1155 |
| 0xa | G2D_FOREGROUND | | | 31.4.5/31-1155 |
| 0xb | G2D_BACKGROUND | | | 31.4.5/31-1155 |
| 0xc | G2D_ALPHABLEND | | | 31.4.6/31-1155 |
| 0xd | G2D_ROP | | | 31.4.7/31-1156 |
| 0xe | G2D_CONFIG | | | 31.4.8/31-1157 |
| 0xf | G2D_INPUT | | | 31.4.9/31-1158 |
| 0x10 | G2D_MASK | | | 31.4.10/31-1158 |
| 0x11 | G2D_BLENDERCFG | | | 31.4.11/31-1158 |
| 0x14 | G2D_BLEND_A0 | | | 31.4.12/31-1159 |
| 0x15 | G2D_BLEND_A1 | | | 31.4.12/31-1159 |
| 0x16 | G2D_BLEND_A2 | | | 31.4.12/31-1159 |

Table 31-13. Command Stream Registers (continued)

| Offset | Register | Access | Reset Value | Section/Page |
|---|---|---|---|---|
| 0x1f | G2D_BLEND_C7 | W | 0x0000_0000 | 31.4.12/31-1159 |
| 0x20 | VGV1_VTX0 | | | 31.4.13/31-1160 |
| 0x21 | VGV1_VTX1 | | | 31.4.13/31-1160 |
| 0x22 | VGV1_TILEOFS | | | 31.4.14/31-1160 |
| 0x23 | VGV1_FILL | | | 31.4.15/31-1161 |
| 0x24 | VGV1_SCISSORX | | | 31.4.16/31-1161 |
| 0x25 | VGV1_SCISSORY | | | 31.4.17/31-1161 |
| 0x27 | VGV1_CFG1 | | | 31.4.18/31-1161 |
| 0x28 | VGV1_CFG2 | | | 31.4.19/31-1162 |
| 0x29 | VGV1_DIRTYBASE | | | 31.4.20/31-1162 |
| 0x2a | VGV1_CBASE1 | | | 31.4.21/31-1162 |
| 0x2b | VGV1_UBASE2 | | | 31.4.22/31-1162 |
| 0x40 | VGV2_C1X | | | 31.4.23/31-1163 |
| 0x41 | VGV2_C1Y | | | 31.4.24/31-1163 |
| 0x42 | VGV2_C2X | | | 31.4.25/31-1163 |
| 0x43 | VGV2_C2Y | | | 31.4.25/31-1163 |
| 0x44 | VGV2_C3X | | | 31.4.26/31-1163 |
| 0x45 | VGV2_C3Y | | | 31.4.26/31-1163 |
| 0x46 | VGV2_C4X | | | 31.4.27/31-1163 |
| 0x47 | VGV2_C4Y | | | 31.4.27/31-1163 |
| 0x48 | VGV2_C1XREL | | | 31.4.28/31-1164 |

Table 31-13. Command Stream Registers (continued)

| Offset | Register | Access | Reset Value | Section/Page |
|---|---|---|---|---|
| 0x55 | VGV2_XFYA | W | 0x0000_0000 | 31.4.34/31-1165 |
| 0x56 | VGV2_XFSTXX | | | 31.4.35/31-1165 |
| 0x57 | VGV2_XFSTYX | | | 31.4.36/31-1165 |
| 0x58 | VGV2_XFSTXY | | | 31.4.37/31-1166 |
| 0x59 | VGV2_XFSTYY | | | 31.4.38/31-1166 |
| 0x5a | VGV2_BBOXMINX | | | 31.4.39/31-1166 |
| 0x5b | VGV2_BBOXMINY | | | 31.4.39/31-1166 |
| 0x5c | VGV2_BBOXMAXX | | | 31.4.40/31-1166 |
| 0x5d | VGV2_BBOXMAXY | | | 31.4.40/31-1166 |
| 0x5e | VGV2_SCALE | | | 31.4.41/31-1166 |
| 0x5f | VGV2_BIAS | | | 31.4.42/31-1167 |
| 0x60 | VGV2_ACCURACY | | | 31.4.43/31-1167 |
| 0x61 | VGV2_THINRADIUS | | | 31.4.44/31-1167 |
| 0x62 | VGV2_ARCCOS | | | 31.4.45/31-1167 |
| 0x63 | VGV2_ARCSIN | | | 31.4.46/31-1167 |
| 0x64 | VGV2_ARCTAN | | | 31.4.47/31-1168 |
| 0x65 | VGV2_RADIUS | | | 31.4.48/31-1168 |
| 0x66 | VGV2_MITER | | | 31.4.49/31-1168 |
| 0x68 | VGV2_CLIP | | | 31.4.50/31-1168 |
| 0x6e | VGV2_MODE | | | 31.4.51/31-1169 |
| 0x6f | VGV2_ACTION | | | 31.4.52/31-1170 |

Table 31-13. Command Stream Registers (continued)

| Offset | Register | Access | Reset Value | Section/Page |
|---|---|---|---|---|
| 0x7c | VGV3_WRITERAW | W | 0x0000_0000 | 31.4.61/31-1173 |
| 0x7d | VGV3_WRITEDMI | | | 31.4.62/31-1173 |
| 0x7f | VGV3_LAST | | | 31.4.63/31-1173 |
| 0x84 | FBC_BASE | | | 31.4.64/31-1174 |
| 0x86 | FBC_DATA | | | 31.4.65/31-1174 |
| 0x88 | FBC_WIDTH | | | 31.4.66/31-1174 |
| 0x8a | FBC_HEIGHT | | | 31.4.67/31-1174 |
| 0x8c | FBC_STRIDE | | | 31.4.68/31-1174 |
| 0x8e | FBC_START | | | 31.4.69/31-1175 |
| 0xb0 | G2D_CONST0 | | | 31.4.70/31-1175 |
| 0xb1 | G2D_CONST1 | | | 31.4.70/31-1175 |
| 0xb2 | G2D_CONST2 | | | 31.4.70/31-1175 |
| 0xb3 | G2D_CONST3 | | | 31.4.70/31-1175 |
| 0xb4 | G2D_CONST4 | | | 31.4.70/31-1175 |
| 0xb5 | G2D_CONST5 | | | 31.4.70/31-1175 |
| 0xb6 | G2D_CONST6 | | | 31.4.70/31-1175 |
| 0xb7 | G2D_CONST7 | | | 31.4.70/31-1175 |
| 0xc0 | GRADW_CONST0 | | | 31.4.71/31-1175 |
| 0xc1 | GRADW_CONST1 | | | 31.4.71/31-1175 |
| 0xc2 | GRADW_CONST2 | | | 31.4.71/31-1175 |
| 0xc3 | GRADW_CONST3 | | | 31.4.71/31-1175 |

Table 31-13. Command Stream Registers (continued)

| Offset | Register | Access | Reset Value | Section/Page |
|---|---|---|---|---|
| 0xd4 | GRADW_BORDERCOLOR | W | 0x0000_0000 | 31.4.76/31-1177 |
| 0xe0 | GRADW_INST0 | | | 31.4.77/31-1178 |
| 0xe1 | GRADW_INST1 | | | 31.4.77/31-1178 |
| 0xe2 | GRADW_INST2 | | | 31.4.77/31-1178 |
| 0xe3 | GRADW_INST3 | | | 31.4.77/31-1178 |
| 0xe4 | GRADW_INST4 | | | 31.4.77/31-1178 |
| 0xe5 | GRADW_INST5 | | | 31.4.77/31-1178 |
| 0xe6 | GRADW_INST6 | | | 31.4.77/31-1178 |
| 0xe7 | GRADW_INST7 | | | 31.4.77/31-1178 |
| 0xf0 | G2D_XY | | | 31.4.78/31-1178 |
| 0xf1 | G2D_WIDTHHEIGHT | | | 31.4.79/31-1179 |
| 0xf2 | G2D_SXY | | | 31.4.80/31-1179 |
| 0xf3 | G2D_SXY2 | | | 31.4.80/31-1179 |
| 0xf4 | G2D_VGSPAN | | | 31.4.81/31-1179 |
| 0xfe | G2D_IDLE | | | 31.4.82/31-1179 |
| 0xff | G2D_COLOR | | | 31.4.83/31-1180 |

### 31.4.1 G2D_BASE0-3

Offset 0x00 (G2D_BASE0) Access: User write
0x02 (G2D_BASE1)
0x04 (G2D_BASE3)
0x06 (G2D_BASE3)

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Base address for frame buffer |

### 31.4.2 G2D_CFG0-3

Offset 0x01 (G2D_CFG0) Access: User write
0x03 (G2D_CFG1)
0x05 (G2D_CFG3)
0x07 (G2D_CFG3)

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23 | 0x0 | STRIDEDESIGN. Stride sign bit |
| 22 | 0x0 | SWAPBITS. Swap order of bits or nibbles in bytes |
| 21 | 0x0 | SWAPRB. Swap red and blue components after conversion |
| 20 | 0x0 | SWAPALL. ARGB -> BGRA swap |
| 19 | 0x0 | SWAPBYTES. Swap read bytes in 32-bit value before conversion |
| 18 | 0x0 | SWAPWORDS. Swap read words in 32-bit value before conversion |
| 17 | 0x0 | SRGB. Surface is in sRGB format |
| 16 | 0x0 | TILED |
| 15–12 | 0x0 | FORMAT<br>0x0 G2D_1 (foreground and background)<br>0x1 G2D_1BW (black and white)<br>0x2 G2D_4<br>0x3 G2D_8 (blue when written)<br>0x4 G2D_4444<br>0x5 G2D_1555<br>0x6 G2D_0565<br>0x7 G2D_8888<br>0x8 G2D_YUY2<br>0x9 G2D_UYVY<br>0xA G2D_YVYU<br>0xB G2D_4444_RGBA<br>0xC G2D_5551_RGBA<br>0xD G2D_8888_RGBA<br>0xE G2D_A8<br>0xF G2D_88 (alpha and blue) |
| 11–0 | 0x0 | STRIDE |

### 31.4.3 G2D_SCISSORX

Offset 0x08 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–22 | 0x0 | Reserved |
| 21–11 | 0x0 | RIGHT |
| 10–0 | 0x0 | LEFT |

### 31.4.4 G2D_SCISSORY

Offset 0x09 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–22 | 0x0 | Reserved |
| 21–11 | 0x0 | BOTTOM |
| 10–0 | 0x0 | TOP |

### 31.4.5 G2D_FOREGROUND, G2D_BACKGROUND

Offset 0x0A (G2D_FOREGROUND)
0x0B (G2D_BACKGROUND) Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Color in 32-bit ARGB format |

### 31.4.6 G2D_ALPHABLEND

Offset: 0x0C Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–16 | 0x0 | Reserved |
| 15 | 0x0 | MASKTOALPHA. Route mask to alpha |
| 14 | 0x0 | PREMULTIPLYDST. Premultiply destination color |
| 13 | 0x0 | INVERTMASK. Invert mask value |
| 12 | 0x0 | MODULATE.Modulate mask to alpha |
| 11 | 0x0 | OPTIMIZE. Optimize reads based on color alpha value before ROP |
| 10 | 0x0 | INVERT. Invert alpha value |
| 9 | 0x0 | CONSTANT. Use constant alpha value |

Offset: 0x0C Access: User write

| Field | Reset | Description |
|---|---|---|
| 8 | 0x0 | OBS_ENABLE. Obsolete: Enable alpha blend, enable blender instead |
| 7–0 | 0x0 | ALPHA. Constant alpha value |

### 31.4.7 G2D_ROP

Offset: 0x0D Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–16 | 0x0 | Reserved |
| 15–8 | 0x0 | Raster Operation (ROP) data type MASK |
| 7 | 0x0 | Raster Operation (ROP) data type DST_SRC_PAT |
| 6 | 0x0 | Raster Operation (ROP) data type SRC_PAT |
| 5 | 0x0 | Raster Operation (ROP) data type DST_PAT |
| 4 | 0x0 | Raster Operation (ROP) data type SRC_DST |
| 3 | 0x0 | Raster Operation (ROP) data type PAT |
| 2 | 0x0 | Raster Operation (ROP) data type SRC |
| 1 | 0x0 | Raster Operation (ROP) data type DST |
| 0 | 0x0 | Raster Operation (ROP) data type NOT |

## 31.4.8 G2D_CONFIG

Offset: 0x0E Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–20 | 0x0 | Reserved |
| 19 | 0x0 | NOPROTECT. Don't use protect for destination reads |
| 18 | 0x0 | NOLASTPIXEL. Don't draw last pixel of a line (for polylines) |
| 17 | 0x0 | PALMLINES.Draw lines according to PalmOS rules (default WinCE) |
| 16 | 0x0 | ALPHATEX. Take only alpha from texture 0, apply color from input |
| 15 | 0x0 | AMASK. Mask writes to alpha component, value 0 writes, 1 doesn't. |
| 14 | 0x0 | RMASK. Mask writes to red component, value 0 writes, 1 doesn't. |
| 13 | 0x0 | GMASK. Mask writes to green component, value 0 writes, 1 doesn't. |
| 12 | 0x0 | BMASK. Mask writes to blue component, value 0 writes, 1 doesn't. |
| 11 | 0x0 | WRITESRGB. Write sRGB color |
| 10 | 0x0 | DITHER. Dither the output |
| 9 | 0x0 | IGNORECKALPHA. Ignore alpha in color key compare |
| 8 | 0x0 | OBS_GAMMA. Obsolete: Gamma correct 4-bit bitmap, use SRGB instead |
| 7–6 | 0x0 | ROTATE. Destination rotate |
| 5 | 0x0 | DSTCK. Destination color key |
| 4 | 0x0 | SRCCK. Source color key |
| 3 | 0x0 | SRC3. Read source 3 |
| 2 | 0x0 | SRC2. Read source 2 |
| 1 | 0x0 | SRC1. Read source 1 |
| 0 | 0x0 | DST. Read destination |

## 31.4.9 G2D_INPUT

Offset: 0x0F Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–6 | 0x0 | Reserved |
| 5 | 0x0 | LINEMODE. Draw lines instead of rectangles |
| 4 | 0x0 | VGMODE. Input is assumed to be generated by V1 and uses slightly different protocol |
| 3 | 0x0 | COPYCOORD. Copy destination coordinates to unsent source coordinates too |
| 2 | 0x0 | SCOORD2. Source coordinate 2 in input (for pattern) |
| 1 | 0x0 | SCOORD1. Source coordinate 1 in input |
| 0 | 0x0 | COLOR. Constant color field in input |

## 31.4.10 G2D_MASK

Offset: 0x10 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–12 | 0x0 | XMASK. Mask for second source X coordinate |
| 11–0 | 0x0 | YMASK. Mask for second source Y coordinate |

## 31.4.11 G2D_BLENDERCFG

Offset: 0x11 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–9 | 0x0 | Reserved |
| 8 | 0x0 | NOMASK. Don't perform mask blending |
| 7 | 0x0 | OBS_DIVALPHA. Obsolete: Take divider from TEMP0 alpha instead of TEMP1 |
| 6 | 0x0 | OOALPHA. One Over Alpha: Divide TEMP0 by TEMP1 after blend program (inverse premultiply) |
| 5 | 0x0 | ENABLE. Enable blender |
| 4–3 | 0x0 | ALPHAPASSES. Number of alpha passes |
| 2–0 | 0x0 | PASSES. Number of passes |

## 31.4.12 G2D_BLEND_A0-3, GD2_BLEND_C0-7

Offset 0x14 (G2D_BLEND_A0)
0x15 (G2D_BLEND_A1)
0x16 (G2D_BLEND_A2)
0x17 (G2D_BLEND_A3)
0x18 (G2D_BLEND_C0)
0x19 (G2D_BLEND_C1)
0x1A (G2D_BLEND_C2)
0x1B (G2D_BLEND_C3)
0x1C (G2D_BLEND_C4)
0x1D (G2D_BLEND_C5)
0x1E (G2D_BLEND_C6)
0x1F (G2D_BLEND_C7)

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31 | 0x0 | CONST_D. Constant used for SRC D |
| 30 | 0x0 | CONST_C. Constant used for SRC C |
| 29 | 0x0 | CONST_B. Constant used for SRC B |
| 28 | 0x0 | CONST_A. Constant used for SRC A |
| 27–25 | 0x0 | SRC_D. Source D register. SeeTable 31-14. |
| 24–22 | 0x0 | SRC_C. Source C register. SeeTable 31-14. |
| 21–19 | 0x0 | SRC_B. Source B register. SeeTable 31-14. |
| 18–16 | 0x0 | SRC_A. Source A register. SeeTable 31-14. |
| 15 | 0x0 | INV_D. Invert (1-x) argument D |
| 14 | 0x0 | INV_C. Invert (1-x) argument C |
| 13 | 0x0 | INV_B. Invert (1-x) argument B |
| 12 | 0x0 | INV_A. Invert (1-x) argument A |
| 11 | 0x0 | AR_D. Alpha replicate argument D |
| 10 | 0x0 | AR_C. Alpha replicate argument C |
| 9 | 0x0 | AR_B. Alpha replicate argument B |
| 8 | 0x0 | AR_A. Alpha replicate argument A |
| 7–6 | 0x0 | DST_C. C*D result destination register SeeTable 31-15. |
| 5–4 | 0x0 | DST_B. A*B result destination register SeeTable 31-15. |
| 3–2 | 0x0 | DST_A. A*B op C*D result destination register SeeTable 31-15. |
| 1–0 | 0x0 | OPERATION.<br>0x0 G2D_BLENDOP_ADD<br>0x1 G2D_BLENDOP_SUB<br>0x2 G2D_BLENDOP_MIN<br>0x3 G2D_BLENDOP_MAX |

**Table 31-14. Blend Source**

| Value | Blend Source |
|---|---|
| 0x0 | G2D_BLENDSRC_ZERO (one with invert) |
| 0x1 | G2D_BLENDSRC_SOURCE (paint) |
| 0x2 | G2D_BLENDSRC_DESTINATION |
| 0x3 | G2D_BLENDSRC_IMAGE (second texture) |
| 0x4 | G2D_BLENDSRC_TEMP0 |
| 0x5 | G2D_BLENDSRC_TEMP1 |
| 0x6 | G2D_BLENDSRC_TEMP2 |
| 0x7 | G2D_BLENDSRC_MASK (mask value with coverage replicated to all channels) |

**Table 31-15. Blend Destination**

| Value | Blend Source |
|---|---|
| 0x0 | G2D_BLENDSRC_IGNORE (ignore destination) |
| 0x1 | G2D_BLENDDST_TEMP0 |
| 0x2 | G2D_BLENDDST_TEMP1 |
| 0x3 | G2D_BLENDDST_TEMP2 |

## 31.4.13 VGV1_VTX0-1

Offset 0x20 (VGV1_VTX0)
0x21 (VGV1_VTX1)

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–16 | 0x0 | Vertex 0(1) Y coordinate (S12.4) |
| 15–0 | 0x0 | Vertex 0(1) X coordinate (S12.4) |

## 31.4.14 VGV1_TILEOFS

Offset: 0x22

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–25 | 0x0 | Reserved |
| 24 | 0x0 | The leftmost tile |
| 23–12 | 0x0 | Tile Y offset (U12) |
| 11–0 | 0x0 | Tile X offset (U12) |

## 31.4.15 VGV1_FILL

Offset: 0x23 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–1 | 0x0 | Reserved |
| 0 | 0x0 | INHERIT. Inherit the winding counters to the next tile |

## 31.4.16 VGV1_SCISSORX

Offset: 0x24 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–27 | 0x0 | Reserved |
| 26–16 | 0x0 | RIGHT. Scissor right edge |
| 15–11 | 0x0 | Padding |
| 10–0 | 0x0 | LEFT. Scissor left edge |

## 31.4.17 VGV1_SCISSORY

Offset: 0x25 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–27 | 0x0 | Reserved |
| 26–16 | 0x0 | BOTTOM. Scissor bottom edge |
| 15–11 | 0x0 | Padding |
| 10–0 | 0x0 | TOP. Scissor top edge |

## 31.4.18 VGV1_CFG1

Offset: 0x27 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–1 | 0x0 | Reserved |
| 0 | 0x0 | WINDRULE. Polygon fill rule (0=non-zero, 1=odd-even) |

## 31.4.19 VGV1_CFG2

Offset: 0x28 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–2 | 0x0 | Reserved |
| 1–0 | 0x0 | AAMODE.<br>0x0 none<br>0x1 16x edge AA<br>0x2 flipquad |

## 31.4.20 VGV1_DIRTYBASE

Offset: 0x29 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Dirty buffer base address |

## 31.4.21 VGV1_CBASE1

Offset: 0x2A Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Compressed buffer base address |

## 31.4.22 VGV1_UBASE2

Offset: 0x2B Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Uncompressed buffer base address |

## 31.4.23 VGV2_C1X

Offset: 0x40 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Implements HLINE by modifying C3X keeping C3Y unchanged. Float 1.6.17, typical range -1e9..1e9. |

## 31.4.24 VGV2_C1Y

Offset: 0x41 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Implements VLINE by modifying C3Y keeping C3X unchanged. Float 1.6.17, typical range -1e9..1e9. |

## 31.4.25 VGV2_C2X, VGV2_C2Y

Offset 0x42 (VGV2_C2X)
0x43 (VGV2_C2Y) Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | First controlpoint for cubic. Float 1.6.17, typical range -1e9..1e9. |

## 31.4.26 VGV2_C3X, VGV2_C3Y

Offset 0x44 (VGV2_C3X)
0x45 (VGV2_C3Y) Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Second controlpoint for cubic, first controlpoint for quadratic. Float 1.6.17, typical range -1e9..1e9. |

## 31.4.27 VGV2_C4X, VGV2_C4Y

Offset 0x46 (VGV2_C4X)
0x47 (VGV2_C4Y) Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | End point (also moveto point). Float 1.6.17, typical range -1e9..1e9. |

## 31.4.28 VGV2_C(1-4)XREL, VGV2_C(1-4)YREL

Offset 0x48 (VGV2_C1XREL)
0x49 (VGV2_C1YREL)
0x4A (VGV2_C2XREL)
0x4B (VGV2_C2YREL)
0x4C (VGV2_C3XREL)
0x4D (VGV2_C3YREL)
0x4E (VGV2_C4XREL)
0x4F (VGV2_C4YREL)

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Relative to last primitive end. Float 1.6.17, typical range -1e9..1e9. |

## 31.4.29 VGV2_XFXX

Offset: 0x50

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | User to surface transform matrix [r0,c0] (see Usage). Float 1.6.17, typical range -100..100. |

## 31.4.30 VGV2_XFYX

Offset: 0x51

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | User to surface transform matrix [r1,c0] (see Usage). Float 1.6.17, typical range -100..100. |

## 31.4.31 VGV2_XFXY

Offset: 0x52

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | User to surface transform matrix [r0,c1] (see Usage). Float 1.6.17, typical range -100..100. |

## 31.4.32 VGV2_XFYY

Offset: 0x53 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | User to surface transform matrix [r1,c1] (see Usage). Float 1.6.17, typical range -100..100. |

## 31.4.33 VGV2_XFXA

Offset: 0x54 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | User to surface transform matrix [r0,c2] (see Usage). Float 1.6.17, typical range -10000..10000. |

## 31.4.34 VGV2_XFYA

Offset: 0x55 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | User to surface transform matrix [r1,c2] (see Usage). Float 1.6.17, typical range -10000..10000. |

## 31.4.35 VGV2_XFSTXX

Offset: 0x56 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Stroke matrix [r0,c0] (see Usage). Float 1.6.17, typical range -100..100. |

## 31.4.36 VGV2_XFSTYX

Offset: 0x57 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Stroke matrix [r1,c0] (see Usage). Float 1.6.17, typical range -100..100. |

## 31.4.37 VGV2_XFSTXY

Offset: 0x58 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Stroke matrix [r0,c1] (see Usage). Float 1.6.17, typical range -100..100. |

## 31.4.38 VGV2_XFSTYY

Offset: 0x59 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Stroke matrix [r1,c1] (see Usage). Float 1.6.17, typical range -100..100. |

## 31.4.39 VGV2_BBOXMINX, VGV2_BBOXMINY

Offset 0x5A (VGV2_BBOXMINX)
0x5B (VGV2_BBOXMINY) Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Bounding box minimum x(y) (origo at matrix origo). Float 1.6.17, typical range -10000..10000 |

## 31.4.40 VGV2_BBOXMAXX, VGV2_BBOXMAXY

Offset 0x5C (VGV2_BBOXMAXX)
0x5D (VGV2_BBOXMAXY) Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Bounding box maximum x(y) (origo at matrix origo). Float 1.6.17, typical range -10000..10000 |

## 31.4.41 VGV2_SCALE

Offset: 0x5E Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Scale for coord=incoord*scale+bias. Float 1.6.17, typical range 0..1, normally 1 |

## 31.4.42 VGV2_BIAS

Offset: 0x5F Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Bias for coord=incoord*scale+bias. Float 1.6.17, typical range -1..1, normally 0 |

## 31.4.43 VGV2_ACCURACY

Offset: 0x60 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Accuracy goal in pixels for curve splitting. Float 1.6.17, typical range 0.25..1.0. |

## 31.4.44 VGV2_THINRADIUS

Offset: 0x61 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Screen space radius limit for strokes considered to be thin, typical range 0..1.0, normally 0. |

## 31.4.45 VGV2_ARCCOS

Offset: 0x62 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Round cap smoothness (cos(roundangle), see Usage). Float 1.6.17, typical range 0.5..1. |

## 31.4.46 VGV2_ARCSIN

Offset: 0x63 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Round cap smoothness (sin(roundangle), see Usage). Float 1.6.17, typical range -0.5..0.5. |

## 31.4.47 VGV2_ARCTAN

Offset: 0x64 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Round cap smoothness (tan(roundangle), see Usage). Float 1.6.17, typical range 0.1..1.0. |

## 31.4.48 VGV2_RADIUS

Offset: 0x65 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Stroke radius (in user space). Float 1.6.17, typical range 0.1..10000. |

## 31.4.49 VGV2_MITER

Offset: 0x66 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Miter limit (given as cosine of angle, see Usage). Float 1.6.17, typical range 0..1. |

## 31.4.50 VGV2_CLIP

Offset: 0x68 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–0 | 0x0 | Distance of clip edge from origo. Float 1.6.17, typical range 10..100000, normally 1024. |

## 31.4.51 VGV2_MODE

Offset: 0x6E

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–18 | 0x0 | EXPONENTADD. Add value to output coordinate exponents before float_to_int conversion. Nonzero values here scale the output coordinates and thereby the typical ranges in float registers above are modified. |
| 17 | 0x0 | SIMPLECLIP. Disable some clip optimizations (in case they don't work in some cases). |
| 16 | 0x0 | SIMPLESTROKE. Disable complex stroking cases (sometimes faster but less robust). Currently only disables CROSSED segments, may be expanded in the future. |
| 15 | 0x0 | SYMMETRICJOINS. Symmetric two sided joins |
| 14 | 0x0 | DROPOTHER. Drop primitives on others than left sides of bounding box |
| 13 | 0x0 | DROPLEFT. Drop primitives left of bounding box |
| 12 | 0x0 | OPENFILL. Allow open fill paths |
| 11 | 0x0 | NODOTS. Disable drawing of dots for zerolenth segments |
| 10 | 0x0 | FULLSPLIT. Always split curves up to maxsplit limit |
| 9 | 0x0 | STROKESPLIT. Use stroke splitting mode (should be enabled when STROKE is enabled) |
| 8 | 0x0 | STROKE. Enable stroke mode |
| 7–6 | 0x0 | JOIN. Stroke join mode<br>0x0 Miter joins (both sides extended towards intersection. If angle is too small (compared to STMITER register) the miter is converted into a BEVEL.<br>0x1 Round joins (smoothness depends on ARCSIN/ARCCOS registers)<br>0x2 Bevel joins (ends of both sides are connected with a single line) |
| 5–4 | 0x0 | CAP. Stroke cap mode<br>0x0 Butt caps (straight line overlappin starting point)<br>0x1 Round caps (smoothness depends on ARCSIN/ARCCOS registers)<br>0x2 Square caps (square centered on starting point) |
| 3–0 | 0x0 | MAXSPLIT. Limit of number of recursive splits for curves |

## 31.4.52 VGV2_ACTION

Offset: 0x6F Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–4 | 0x0 | Reserved |
| 3–0 | 0x0 | Triggers a drawing action using current coordinates<br>0x0 End previous path<br>0x1 End previous path, C1=C4, start new open subpath<br>0x2 End previous path, C1=C4, start new closed subpath<br>0x3 Line C1,C4<br>0x4 Cubic C1,C2,C3,C4<br>0x5 Quadratic C1,C3,C4<br>0x6 Smooth cubic C1,C4<br>0x7 Smooth quadratic C1,C3,C4<br>0x8 Half lineto C4=pos, C3=normal.<br>0x9 Moveto open + half lineto C4=pos, C3=normal.<br>0xA Moveto closed + half lineto C4=pos, C3=normal.<br>0xB End previous path, C1=C4, move but do not start a subpath<br>0xF End previous path and block following regwrites until all lines sent |

## 31.4.53 VGV3_CONTROL

Offset: 0x70 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–24 | 0x0 | Reserved |
| 23–21 | 0x0 | DMIWAITBUF. If set: pause stream reading until the given frame buffer (one hot encoded) is free |
| 20 | 0x0 | V0SYNC. If set: pause stream reading until sync signal received from V0 |
| 19 | 0x0 | BCFLUSH. If set: flush burst cache (full flush). Should be used when the command buffer(s) added with this control write use old memory that may potentially be in cache. Can also be used to trigger flushes when textures etc are changed. |
| 18 | 0x0 | WRITE. If set: write current stream address to location given by VGV3_WRITEADDR (can be used for polling state) |
| 17 | 0x0 | ABORT. If set: abort stream reading immediately (should not be used in normal operation) |
| 16 | 0x0 | PAUSE. If set: pause stream reading after next marked packet |
| 15–12 | 0x0 | DMIWAITCHMASK. Selects DMI channels which are checked for free buffer when non-zero DMIWAIT-BUF is given |
| 11–0 | 0x0 | MARKADD. Increment mark counter by this value. Mark counter tells how many undread marked packets are in the stream. Stream reading continues as long as mark counter is positive (and reading is not paused). |

## 31.4.54 VGV3_MODE

Offset: 0x71 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–5 | 0x0 | Reserved |
| 4 | 0x0 | DMIRESET. If set, reset state of DMI interface (should not be needed) |
| 3 | 0x0 | DMIPAUSETYPE. If set, also input regwrites are blocked (in addition to stream reading) when V0SYNC or DMIWAITBUF is set in VGV3_CONTROL (normally allways set to 0) |
| 2 | 0x0 | WRITEFLUSH. If set memory writes are done straight to L2 to make sure they arrive to memory as soon as possible. |
| 1 | 0x0 | UNUSED |
| 0 | 0x0 | FLIPENDIAN. Flip dwords read from memory from big-endian to little-endian |

## 31.4.55 VGV3_WRITEADDR

Offset: 0x72 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Address where all status writes happen. Normally a single shared address used by the driver should be sufficient, but it is also possible to change this address in the data stream. |

## 31.4.56 VGV3_WRITE

Offset: 0x73 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Write this value to VGV3_WRITEADDR. This can be used to report to the CPU how far in the stream reading has progressed |

## 31.4.57 VGV3_WRITEIFPAUSED

Offset: 0x74 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Write this value to VGV3_WRITEADDR if reading is paused. This should be the last register write in a packet so that pause/mark counter state has been updated for the packet before the write condition compare happens. Every marked packet should probably have this conditional write so that the CPU can detect when a pause request has been completed. |

## 31.4.58 VGV3_NEXTADDR

Offset: 0x75 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Address of sub-stream packet to call (see VGV3_NEXTCMD register) |

## 31.4.59 VGV3_NEXTCMD

Offset: 0x76 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–28 | 0x0 | Reserved |
| 27–16 | 0x0 | CALLCOUNT. Length of sub-stream packet in call in dwords. |
| 15 | 0x0 | MARK. If set: mark counter is decreased by one when this packet ends. If it becomes zero reading is paused. |
| 14–12 | 0x0 | NEXTCMD. Action to perform after this packet.<br>0x0 Continue reading at current address, COUNT gives size of next packet.<br>0x1 Jump to CALLADDR, COUNT gives size of next packet.<br>0x2 First call a sub-stream at CALLADDR for CALLCOUNT dwords. Then perform a continue.<br>0x3 Not supported.<br>0x4 Not supported.<br>0x5 Abort reading. This ends the stream. Normally stream can just be paused (or automatically pauses at the end) which avoids any data being lost. |
| 11–0 | 0x0 | COUNT. Length of next packet in dwords. |

## 31.4.60 VGV3_VGBYPASS

Offset: 0x77 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–1 | 0x0 | Reserved |
| 0 | 0x0 | When set, bypass V1 and V2, i.e. use only V3+2D (when V1+V2 cloks disabled). |

## 31.4.61 VGV3_WRITES8, VGV3_WRITES16, VGV3_WRITES32, VGV3_WRITEF32, VGV3_WRITERAW

Offset 0x78 (VGV3_WRITES8)
0x79 (VGV3_WRITES16)
0x7A (VGV3_WRITES32)
0x7B (VGV3_WRITEF32)
0x7C (VGV3_WRITERAW)

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–27 | 0x0 | Reserved |
| 26–24 | 0x0 | FORMAT. Lower 3 bits of address specifies the format (and each format effectively maps to a different VGV3_WRITE register).<br>0x0 Signed 8 bit data (4 writes per data dword) => VGV2-float<br>0x1 Signed 16 bit data (2 writes per data dword) => VGV2-float<br>0x2 Signed 32 bit data => VGV2-float<br>0x3 IEEE 32-bit floating point => VGV2-float<br>0x4 No converson |
| 23–20 | 0x0 | ACTION. If nonzero: enable action mode where this value is sent to VGV2_ACTION register at end of each loop. |
| 19–16 | 0x0 | LOOP. If nonzero: enable looping mode where addr is reset after every count writes. |
| 15–8 | 0x0 | COUNT. Count of how many register writes to do. |
| 7–0 | 0x0 | ADDR. Start register address for burst |

## 31.4.62 VGV3_WRITEDMI

Offset: 0x7D

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–7 | 0x0 | Reserved |
| 6–4 | 0x0 | BUFFER. Set the frame buffer BUFFER as done in DMI outputs |
| 3–0 | 0x0 | CHANMASK. Frame buffer 'BUFFER' will be displayed on masked DMI channels |

## 31.4.63 VGV3_LAST

Offset: 0x7F

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–1 | 0x0 | Reserved |
| 0 | 0x0 | Just a dummy register used to check if address is in VGV3 range |

### 31.4.64 FBC_BASE

Offset: 0x84 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Fast Buffer Clears Base Address |

### 31.4.65 FBC_DATA

Offset: 0x86 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | 32-bit clear value |

### 31.4.66 FBC_WIDTH

Offset: 0x88 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–11 | 0x0 | Reserved |
| 10–0 | 0x0 | Width (internal value is value + 1) |

### 31.4.67 FBC_HEIGHT

Offset: 0x8A Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–11 | 0x0 | Reserved |
| 10–0 | 0x0 | Height (internal value is value + 1) |

### 31.4.68 FBC_STRIDE

Offset: 0x8C Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–11 | 0x0 | Reserved |
| 10–0 | 0x0 | Stride (internal value is value + 1) |

## 31.4.69 FBC_START

Offset: 0x8E

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–1 | 0x0 | Reserved |
| 0 | 0x0 | DUMMY. Just write anything |

## 31.4.70 G2D_CONST0-7

Offset 0xB0 (G2D_CONST0)
0xB1 (G2D_CONST1)
0xB2 (G2D_CONST2)
0xB3 (G2D_CONST3)
0xB4 (G2D_CONST4)
0xB5 (G2D_CONST5)
0xB6 (G2D_CONST6)
0xB7 (G2D_CONST7)

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Blender Constant in 32-bit ARGB format |

## 31.4.71 GRADW_CONST0-B

Offset 0xC0 (GRADW_CONST0)
0xC1 (GRADW_CONST1)
0xC2 (GRADW_CONST2)
0xC3 (GRADW_CONST3)
0xC4 (GRADW_CONST4)
0xC5 (GRADW_CONST5)
0xC6 (GRADW_CONST6)
0xC7 (GRADW_CONST7)
0xC8 (GRADW_CONST8)
0xC9 (GRADW_CONST9)
0xCA (GRADW_CONSTA)
0xCB (GRADW_CONSTB)

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–16 | 0x0 | Reserved |
| 15–0 | 0x0 | Constant value in 1.5.10 floating point |

## 31.4.72 G2D_GRADIENT

Offset: 0xD0 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–9 | 0x0 | Reserved |
| 8 | 0x0 | SEL. Select register window for gradient and texture unit |
| 7 | 0x0 | ENABLE2.Enable second texture |
| 6 | 0x0 | ENABLE. Enable gradient and texturing operation |
| 5–3 | 0x0 | INSTRUCTIONS2. Number of instructions for pass 2 - 1 |
| 2–0 | 0x0 | INSTRUCTIONS. Number of instructions for pass 1 - 1 |

## 31.4.73 GRADW_TEXCFG

Offset: 0xD1 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–30 | 0x0 | Reserved |
| 29 | 0x0 | SWAPBITS. Swap order of bits or nibbles in bytes |
| 28 | 0x0 | TEX2D. 2D texture |
| 27 | 0x0 | SWAPRB. Swap red and blue components after conversion |
| 26 | 0x0 | SWAPALL. ARGB -> BGRA swap |
| 25 | 0x0 | SWAPBYTES. Swap read bytes in 32-bit value before conversion |
| 24 | 0x0 | SWAPWORDS. Swap read words in 32-bit value before conversion |
| 23 | 0x0 | PREMULTIPLY. Premultiply colors with alpha |
| 22 | 0x0 | SRGB. Texture is in sRGB format |
| 21 | 0x0 | BILIN. |
| 20–19 | 0x0 | WRAPV.<br>0x0 CLAMP<br>0x1 REPEAT<br>0x2 MIRROR<br>0x3 BORDER |
| 18–17 | 0x0 | WRAPU.<br>0x0 CLAMP<br>0x1 REPEAT<br>0x2 MIRROR<br>0x3 BORDER |
| 16 | 0x0 | TILED. |

Offset: 0xD1 Access: User write

| Field | Reset | Description |
|---|---|---|
| 15–12 | 0x0 | FORMAT<br>0x0 G2D_1 (foreground and background)<br>0x1 G2D_1BW (black and white)<br>0x2 G2D_4<br>0x3 G2D_8 (blue when written)<br>0x4 G2D_4444<br>0x5 G2D_1555<br>0x6 G2D_0565<br>0x7 G2D_8888<br>0x8 G2D_YUY2<br>0x9 G2D_UYVY<br>0xA G2D_YVYU<br>0xB G2D_4444_RGBA<br>0xC G2D_5551_RGBA<br>0xD G2D_8888_RGBA<br>0xE G2D_A8<br>0xF G2D_88 (alpha and blue) |
| 11–0 | 0x0 | STRIDE. |

## 31.4.74 GRADW_TEXSIZE

Offset: 0xD2 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–22 | 0x0 | Reserved |
| 21–11 | 0x0 | Texture height |
| 10–0 | 0x0 | Texture width |

## 31.4.75 GRADW_TEXBASE

Offset: 0xD3 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Texture base address |

## 31.4.76 GRADW_BORDERCOLOR

Offset: 0xD4 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Texture border color in 32-bit ARGB format |

## 31.4.77 GRADW_INST0-7

Offset 0xE0 (GRADW_INST0)
0xE1 (GRADW_INST1)
0xE2 (GRADW_INST2)
0xE3 (GRADW_INST3)
0xE4 (GRADW_INST4)
0xE5 (GRADW_INST5)
0xE6 (GRADW_INST6)
0xE7 (GRADW_INST7)

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31 | 0x0 | Reserved |
| 30–29 | 0x0 | Opcode.<br>0x0 DOT<br>0x1 RCP<br>0x2 SQRTMUL<br>0x3 SQRTADD |
| 28–25 | 0x0 | Destination, 0xxx = temp, 100x = output |
| 24–20 | 0x0 | Source A, 1xxxx = constant, 00xxx = temp, 0100x = output |
| 19–15 | 0x0 | Source B, 1xxxx = constant, 00xxx = temp, 0100x = output |
| 14–10 | 0x0 | Source C, 1xxxx = constant, 00xxx = temp, 0100x = output |
| 9–5 | 0x0 | Source D, 1xxxx = constant, 00xxx = temp, 0100x = output |
| 4–0 | 0x0 | Source E, 1xxxx = constant, 00xxx = temp, 0100x = output |

## 31.4.78 G2D_XY

Offset: 0xF0

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–28 | 0x0 | Reserved |
| 27–16 | 0x0 | Primitve X |
| 15–12 | 0x0 | Padding |
| 11–0 | 0x0 | Primitive Y |

## 31.4.79 G2D_WIDTHHEIGHT

Offset: 0xF1 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–28 | 0x0 | Reserved |
| 27–16 | 0x0 | Primitve width |
| 15–12 | 0x0 | Padding |
| 11–0 | 0x0 | Primitive height |

## 31.4.80 G2D_SXY, G2D_SXY2

Offset 0xF2 (G2D_SXY)
0xF3 (G2D_SXY2)
Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–27 | 0x0 | Reserved |
| 26–16 | 0x0 | Primitve source X |
| 15–11 | 0x0 | Padding |
| 10–0 | 0x0 | Primitive source Y |

## 31.4.81 G2D_VGSPAN

Offset: 0xF4 Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–20 | 0x0 | Reserved |
| 19–16 | 0x0 | Coverage value |
| 15–12 | 0x0 | Padding |
| 11–0 | 0x0 | Span width |

## 31.4.82 G2D_IDLE

Offset: 0xFE Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–3 | 0x0 | Reserved |
| 2 | 0x0 | Send signal to V3 after flush |
| 1 | 0x0 | Flush burst cache |
| 0 | 0x0 | Send IRQ after flush |

## 31.4.83 G2D_COLOR

Offset: 0xFF

Access: User write

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Color in 32-bit ARGB format |

## 31.5 MMU Command Stream registers

This section provides a detailed description of the GFX2D internal registers that are only accessible via G12_MMUCOMMANDSTREAM.

**Table 31-16. MMU Command Stream Registers**

| Offset | Register | Access | Reset Value | Section/Page |
|---|---|---|---|---|
| 0x040 | MH_MMU_CONFIG—MMU Configuration | R/W | 0x0000_0000 | 31.5.1/31-1183 |
| 0x041 | MH_MMU_VA_RANGE—MMU Virtual Base Address and Range | R/W | 0x0000_0000 | 31.5.2/31-1183 |
| 0x042 | MH_MMU_PT_BASE—MMU Page Table Base Address | R/W | 0x0000_0000 | 31.5.3/31-1184 |
| 0x043 | MH_MMU_PAGE_FAULT—MMU Page Fault | R | 0x0000_0000 | 31.5.4/31-1184 |
| 0x044 | MH_MMU_TRAN_ERROR—MMU Transaction Error Address | R/W | 0x0000_0000 | 31.5.5/31-1184 |
| 0x045 | MH_MMU_INVALIDATE—MMU Invalidate | W | 0x0000_0000 | 31.5.6/31-1185 |
| 0x046 | MH_MMU_MPU_BASE—MPU Base Address | R/W | 0x0000_0000 | 31.5.7/31-1185 |
| 0x047 | MH_MMU_MPU_END—MPU End Address | R/W | 0x0000_0000 | 31.5.8/31-1185 |
| 0xa40 | MH_ARBITER_CONFIG—Arbiter Configuration | R/W | 0x07C8_6590 | 31.5.9/31-1186 |
| 0xa41 | MH_CLNT_AXI_ID_REUSE—Client AXI IDs | R/W | 0x0000_1243 | 31.5.10/31-1187 |
| 0xa42 | MH_INTERRUPT_MASK—Interrupt Mask | R/W | 0x0000_0000 | 31.5.11/31-1187 |
| 0xa43 | MH_INTERRUPT_STATUS—Interrupt Status | R/W | 0x0000_0000 | 31.5.12/31-1187 |
| 0xa44 | MH_INTERRUPT_CLEAR—Interrupt Clear | W | 0x0000_0000 | 31.5.13/31-1188 |
| 0xa45 | MH_AXI_ERROR—AXI Read/Write Error Status | R | 0x0000_0000 | 31.5.14/31-1188 |
| 0xa46 | MH_PERFCOUNTER0_SELECT—Performance Counter 0 Select | R/W | 0x0000_00FF | 31.5.15/31-1189 |
| 0xa47 | MH_PERFCOUNTER0_CONFIG—Performance Counter 0 Config | R/W | 0x0000_0000 | 31.5.16/31-1189 |
| 0xa48 | MH_PERFCOUNTER0_LOW—Performance Counter 0 Lower 32 bits | R | 0x0000_0000 | 31.5.17/31-1189 |
| 0xa49 | MH_PERFCOUNTER0_HI—Performance Counter 0 Upper 16 bits | R | 0x0000_0000 | 31.5.18/31-1189 |

**Table 31-16. MMU Command Stream Registers**

| Offset | Register | Access | Reset Value | Section/Page |
|---|---|---|---|---|
| 0xa4a | MH_PERFCOUNTER1_SELECT—Performance Counter 1 Select | R/W | 0x0000_00FF | 31.5.19/31-1190 |
| 0xa4b | MH_PERFCOUNTER1_CONFIG—Performance Counter 1 Config | R/W | 0x0000_0000 | 31.5.20/31-1190 |
| 0xa4c | MH_PERFCOUNTER1_LOW—Performance Counter 1 Lower 32 bits | R | 0x0000_0000 | 31.5.21/31-1190 |
| 0xa4d | MH_PERFCOUNTER1_HI—Performance Counter 1 Upper 16 bits | R | 0x0000_0000 | 31.5.22/31-1190 |
| 0xa4e | MH_DEBUG_CTRL—Debug Control | R/W | 0x0000_0000 | 31.5.23/31-1191 |
| 0xa4f | MH_DEBUG_DATA—Debug Data | R | 0x0000_0000 | 31.5.24/31-1191 |
| 0xa50 | MH_AXI_HALT_CONTROL—AXI Halt Control | R/W | 0x0000_0000 | 31.5.25/31-1191 |

## 31.5.1 MH_MMU_CONFIG

Offset: 0x040 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–26 | 0x0 | Reserved |
| 25–24 | 0x0 | Specifies PAw client behavior for MMU lookups. See Table 31-17. |
| 23–22 | 0x0 | Specifies TCr client behavior for MMU lookups. See Table 31-17. |
| 21–20 | 0x0 | Specifies VGTr1 client behavior for MMU lookups. See Table 31-17. |
| 19–18 | 0x0 | Specifies VGTr0 client behavior for MMU lookups. See Table 31-17. |
| 17–16 | 0x0 | Specifies CPr4 client behavior for MMU lookups. See Table 31-17. |
| 15–14 | 0x0 | Specifies CPr3 client behavior for MMU lookups. See Table 31-17. |
| 13–12 | 0x0 | Specifies CPr2 client behavior for MMU lookups. See Table 31-17. |
| 11–10 | 0x0 | Specifies CPr1 client behavior for MMU lookups. See Table 31-17. |
| 9–8 | 0x0 | Specifies CPr0 client behavior for MMU lookups. See Table 31-17. |
| 7–6 | 0x0 | Specifies CPw client behavior for MMU lookups. See Table 31-17. |
| 5–4 | 0x0 | Specifies RBw client behavior for MMU lookups. See Table 31-17. |
| 3–2 | 0x0 | Padding |
| 1 | 0x0 | SPLIT_MODE_ENABLE. Reserves 8 translation buffer entries for use by the TC client |
| 0 | 0x0 | MMU_ENABLE. Enables MMU; if disabled all MMU checks are bypassed and PA = VA |

**Table 31-17. MMU Behavior**

| Value | Description |
|---|---|
| 0x0 | Never translate, PA = VA |
| 0x1 | Translate if VA is in VA range, otherwise PA = VA |
| 0x2 | Only translate if VA is in VA range, else page fault |

## 31.5.2 MH_MMU_VA_RANGE

Offset: 0x041 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–12 | 0x0 | Virtual Base address (VA_BASE[31:12]) aligned to a 4kB page boundary (VA_BASE[11:0]=0). |
| 11–0 | 0x0 | Number of 64 KB regions mapped, up to 256MB |

### 31.5.3 MH_MMU_PT_BASE

Offset: 0x042 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–12 | 0x0 | Page Table Base address (PT_BASE[31:12]) aligned to a 4kB page boundary (PT_BASE[11:0]=0). |
| 11–0 | 0x0 | Alignment, zeroed. |

### 31.5.4 MH_MMU_PAGE_FAULT

Offset: 0x043 Access: User read

| Field | Reset | Description |
|---|---|---|
| 31–12 | 0x0 | VA[31:12] of lookup that caused the page fault |
| 11 | 0x0 | Page table entry not valid for write operation |
| 10 | 0x0 | Page table entry not valid for read operation |
| 9 | 0x0 | Client VA not in VA range |
| 8 | 0x0 | Client PA not in MPU range |
| 7 | 0x0 | Padding bit |
| 6–4 | 0x0 | AXI ID of lookup that caused the page fault |
| 3–2 | 0x0 | Reports programmed client behavior bits at the time of page fault. See Table 31-17. |
| 1 | 0x0 | Operation Type, 0 = read, 1 = write |
| 0 | 0x0 | Page fault occured |

### 31.5.5 MH_MMU_TRAN_ERROR

Offset: 0x044 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–5 | 0x0 | Address location (TRAN_ERROR[31:5]) used for dummy reads or writes in the event of a page fault, aligned on a 32-bit boundary (TRAN_ERROR[4:0]=0). |
| 14–0 | 0x0 | Alignment, zeroed. |

## 31.5.6 MH_MMU_INVALIDATE

Offset: 0x045 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–2 | 0x0 | Reserved |
| 1 | 0x0 | INVALIDATE_ALL. If SPLIT_MODE_ENABLE is set, clears all non-TC tag valid bits; else clears all tag valid bits.<br>Read:<br>0 Invalidate not in progress.<br>1 Invalidate in progress, awaiting sPTE fetch to complete.<br>Write:<br>0 No effect, only cleared by hardware<br>1 Initiate invalidate process. |
| 0 | 0x0 | INVALIDATE_TC. If SPLIT_MODE_ENABLE is set, clears all TC tag valid bits; else does nothing.<br>Read:<br>0 Invalidate not in progress.<br>1 Invalidate in progress, awaiting sPTE fetch to complete.<br>Write:<br>0 No effect, only cleared by hardware<br>1 Initiate invalidate process. |

## 31.5.7 MH_MMU_MPU_BASE

Offset: 0x046 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–12 | 0x0 | Memory Protection Unit Base address (MPU_BASE[31:12]) aligned to a 4kB page boundary (MPU_BASE[11:0]=0). |
| 11–0 | 0x0 | Alignment, zeroed. |

## 31.5.8 MH_MMU_MPU_END

Offset: 0x047 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–12 | 0x0 | Memory Protection Unit End address (MPU_END[31:12]) aligned to a 4kB page boundary (MPU_END[11:0]=0). |
| 11–0 | 0x0 | Alignment, zeroed. |

## 31.5.9 MH_ARBITER_CONFIG

Offset: 0xa40 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–27 | 0x0 | Reserved. |
| 26 | 0x1 | PA_CLNT_ENABLE. Enables PA client requests |
| 25 | 0x1 | RB_CLNT_ENABLE. Enables RB client requests |
| 24 | 0x1 | TC_CLNT_ENABLE. Enables TC client requests |
| 23 | 0x1 | VGT_CLNT_ENABLE. Enables VGT client requests |
| 22 | 0x1 | CP_CLNT_ENABLE. Enables CP client requests |
| 21–16 | 0x8 | When IN_FLIGHT_LIMIT_ENABLE is enabled, specifies the maximum number of outstanding requests allowed by MH (0 = no limit) |
| 15 | 0x0 | IN_FLIGHT_LIMIT_ENABLE. Enables limit of outstanding requests |
| 14 | 0x1 | TC_ARB_HOLD_ENABLE. Holds arbiter for 4 cycles when TC wins arbitration, same page and bank requests exist and TCD is full |
| 13 | 0x1 | TC_REORDER_ENABLE. Enables TC reordering queue. Requires L1_ARB_ENABLE to be set also |
| 12–10 | 0x1 | SDRAM page size--RBC or BRC formats assumed for SDRAM configuration<br>000 0.5 KB<br>001 1 KB<br>010 2 KB<br>011 4 KB<br>100 8 (4) KB<br>101 16 (4) KB |
| 9 | 0x0 | L2_ARB_CONTROL. Selects algorithm for level 2 arbitration<br>0 Least recently used client<br>1 Fixed priority: [CP->PA->VGT->TC->RB] |
| 8 | 0x1 | L1_ARB_HOLD_ENABLE. Selects behavior when arbiter is back pressured<br>0 Forces re-arbitration.<br>1 Holds arbiter on current winner. |
| 7 | 0x1 | L1_ARB_ENABLE. Enables level 1 arbitration for same page and bank addresses |
| 6 | 0x0 | SAME_PAGE_GRANULARITY. Specifies granularity of SAME_PAGE_LIMIT<br>0 granularity = 2<br>1 granularity = 16 |
| 5–0 | 0x10 | When L1_ARB_ENABLE is enabled, specifies the number of same page requests allowed before re-arbitration: for SAME_PAGE_GRANULARITY = 0: same page requests = [SAME_PAGE_LIMIT x 2] + 1 for SAME_PAGE_GRANULARITY = 1: same page requests = [SAME_PAGE_LIMIT x 16] + 1 (0 = no limit) |

## 31.5.10 MH_CLNT_AXI_ID_REUSE

Offset: 0xa41 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–15 | 0x0 | Reserved |
| 14–12 | 0x1 | PAW_ID. PA write client AXI ID |
| 11 | 0x0 | Padding bit |
| 10–8 | 0x2 | MMUR_ID. MMU read client AXI ID |
| 7 | 0x0 | Padding bit |
| 6–4 | 0x4 | RBW_ID. RB write client AXI ID |
| 3 | 0x0 | Padding bit |
| 2–0 | 0x3 | CPW_ID. CP write sub-client AXI ID |

## 31.5.11 MH_INTERRUPT_MASK

Offset: 0xa42 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–3 | 0x0 | Reserved |
| 2 | 0x0 | Enables MMU page fault interrupt |
| 1 | 0x0 | Enables AXI Write error interrupt |
| 0 | 0x0 | Enables AXI Read error interrupt |

## 31.5.12 MH_INTERRUPT_STATUS

Offset: 0xa43 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–3 | 0x0 | Reserved |
| 2 | 0x0 | MMU page fault interrupt status |
| 1 | 0x0 | AXI Write error interrupt status |
| 0 | 0x0 | AXI Read error interrupt status |

## 31.5.13 MH_INTERRUPT_CLEAR

Offset: 0xa44 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–3 | 0x0 | Reserved |
| 2 | 0x0 | Clears MMU page fault interrupt |
| 1 | 0x0 | Clears AXI Write error interrupt |
| 0 | 0x0 | Clears AXI Read error interrupt |

## 31.5.14 MH_AXI_ERROR

Offset: 0x045 Access: User read

| Field | Reset | Description |
|---|---|---|
| 31–8 | 0x0 | Reserved. |
| 7 | 0x0 | AXI_WRITE_ERROR. AXI Write error |
| 6–4 | 0x0 | AXI_WRITE_ID. AXI ID of write error:<br>CPw = MH_CLNT_AXI_ID_REUSE.CPw_ID<br>RBw = MH_CLNT_AXI_ID_REUSE.RBw_ID<br>PAw = MH_CLNT_AXI_ID_REUSE.PAw_ID |
| 3 | 0x0 | AXI_READ_ERROR. AXI Read error |
| 2–0 | 0x0 | AXI_READ_ID. AXI ID of read error:<br>000 CPr0<br>001 CPr1<br>010 CPr2<br>011 CPr3<br>100 CPr4<br>101 VGTr0<br>110 VGTr1<br>111 TCr<br>MMUr = MH_CLNT_AXI_ID_REUSE.MMUr_ID |

## 31.5.15 MH_PERFCOUNTER0_SELECT

Offset: 0xa46 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–8 | 0x0 | Reserved |
| 7–0 | 0xFF | Counter Select. See Table 31-18. |

## 31.5.16 MH_PERFCOUNTER0_CONFIG

Offset: 0xa47 Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–8 | 0x0 | Reserved |
| 7–0 | 0x0 | N value used in Nth access and N entry measurements only. |

## 31.5.17 MH_PERFCOUNTER0_LOW

Offset: 0xa48 Access: User read

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Lower 32 bits of performance count |

## 31.5.18 MH_PERFCOUNTER0_HI

Offset: 0xa49 Access: User read

| Field | Reset | Description |
|---|---|---|
| 31–16 | 0x0 | Reserved |
| 15–0 | 0x0 | Upper 16 bits of performance count |

## 31.5.19 MH_PERFCOUNTER1_SELECT

Offset: 0xa4a

Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–8 | 0x0 | Reserved |
| 7–0 | 0xFF | Counter Select. See Table 31-18. |

## 31.5.20 MH_PERFCOUNTER1_CONFIG

Offset: 0xa4b

Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–8 | 0x0 | Reserved |
| 7–0 | 0x0 | N value used in Nth access and N entry measurements only. |

## 31.5.21 MH_PERFCOUNTER1_LOW

Offset: 0xa4c

Access: User read

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Lower 32 bits of performance count |

## 31.5.22 MH_PERFCOUNTER1_HI

Offset: 0xa4d

Access: User read

| Field | Reset | Description |
|---|---|---|
| 31–16 | 0x0 | Reserved |
| 15–0 | 0x0 | Upper 16 bits of performance count |

## 31.5.23 MH_DEBUG_CTRL

Offset: 0xa4e

Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–6 | 0x0 | Reserved |
| 5–0 | 0x0 | Debug data index |

## 31.5.24 MH_DEBUG_DATA

Offset: 0xa4f

Access: User read

| Field | Reset | Description |
|---|---|---|
| 31–0 | 0x0 | Debug data |

## 31.5.25 MH_AXI_HALT_CONTROL

Offset: 0xa50

Access: User read/write

| Field | Reset | Description |
|---|---|---|
| 31–1 | 0x0 | Reserved |
| 0 | 0x0 | AXI_HALT.<br>Read:<br>0 AXI Halt Not Acknowledged.<br>1 AXI Halt Acknowledged.<br>Write:<br>0 No effect. Cleared by reset. Must wait for AXI halt acknowledgment before initiating reset.<br>1 Initiate AXI Halt Request. |

## 31.5.26 Performance Counters

**Table 31-18. Performance Counters**

| Select Value | Counter Name |
|---|---|
| 0x00 | CP_R0_REQUESTS |
| 0x01 | CP_R1_REQUESTS |
| 0x02 | CP_R2_REQUESTS |
| 0x03 | CP_R3_REQUESTS |
| 0x04 | CP_R4_REQUESTS |
| 0x05 | CP_TOTAL_READ_REQUESTS |
| 0x06 | CP_W_16B_REQUESTS |
| 0x07 | CP_W_32B_REQUESTS |
| 0x08 | CP_TOTAL_WRITE_REQUESTS |
| 0x09 | CP_TOTAL_REQUESTS |
| 0x0A | CP_DATA_BYTES_WRITTEN |
| 0x0B | CP_WRITE_CLEAN_RESPONSES |
| 0x0C | CP_R0_READ_BURSTS_RECEIVED |
| 0x0D | CP_R1_READ_BURSTS_RECEIVED |
| 0x0E | CP_R2_READ_BURSTS_RECEIVED |
| 0x0F | CP_R3_READ_BURSTS_RECEIVED |
| 0x10 | CP_R4_READ_BURSTS_RECEIVED |
| 0x11 | CP_TOTAL_READ_BURSTS_RECEIVED |
| 0x12 | CP_R0_DATA_BEATS_READ |
| 0x13 | CP_R1_DATA_BEATS_READ |
| 0x14 | CP_R2_DATA_BEATS_READ |
| 0x15 | CP_R3_DATA_BEATS_READ |
| 0x16 | CP_R4_DATA_BEATS_READ |
| 0x17 | CP_TOTAL_DATA_BEATS_READ |
| 0x18 | VGT_R0_REQUESTS |
| 0x19 | VGT_R1_REQUESTS |
| 0x1A | VGT_TOTAL_REQUESTS |
| 0x1B | VGT_R0_READ_BURSTS_RECEIVED |
| 0x1C | VGT_R1_READ_BURSTS_RECEIVED |
| 0x1D | VGT_TOTAL_READ_BURSTS_RECEIVED |
| 0x1E | VGT_R0_DATA_BEATS_READ |
| 0x1F | VGT_R1_DATA_BEATS_READ |
| 0x20 | VGT_TOTAL_DATA_BEATS_READ |
| 0x21 | TC_16B_REQUESTS |
| 0x22 | TC_32B_REQUESTS |
| 0x23 | TC_TOTAL_REQUESTS |
| 0x24 | TC_ROQ_REQUESTS |
| 0x25 | TC_INFO_SENT |
| 0x26 | TC_READ_BURSTS_RECEIVED |
| 0x27 | TC_DATA_BEATS_READ |

**Table 31-18. Performance Counters (continued)**

| Select Value | Counter Name |
|---|---|
| 0x28 | TCD_BURSTS_READ |
| 0x29 | RB_REQUESTS |
| 0x2A | RB_DATA_BYTES_WRITTEN |
| 0x2B | RB_WRITE_CLEAN_RESPONSES |
| 0x2C | PA_REQUESTS |
| 0x2D | PA_DATA_BYTES_WRITTEN |
| 0x2E | PA_WRITE_CLEAN_RESPONSES |
| 0x2F–0x36 | AXI_READ_REQUESTS_ID_0-7 |
| 0x37 | AXI_TOTAL_READ_REQUESTS |
| 0x38–0x3F | AXI_READ_REQUEST_DATA_BEATS_ID_0-7 |
| 0x40 | AXI_TOTAL_READ_REQUEST_DATA_BEATS |
| 0x41–0x48 | AXI_WRITE_REQUESTS_ID_0-7 |
| 0x49 | AXI_TOTAL_WRITE_REQUESTS |
| 0x4A–0x51 | AXI_TOTAL_REQUESTS_ID_0-7 |
| 0x52 | AXI_TOTAL_REQUESTS |
| 0x53–0x5A | AXI_READ_CHANNEL_BURSTS_ID_0-7 |
| 0x5B | AXI_READ_CHANNEL_TOTAL_BURSTS |
| 0x5C–0x63 | AXI_READ_CHANNEL_DATA_BEATS_READ_ID_0-7 |
| 0x64 | AXI_READ_CHANNEL_TOTAL_DATA_BEATS_READ |
| 0x65–0x6C | AXI_WRITE_CHANNEL_BURSTS_ID_0-7 |
| 0x6D | AXI_WRITE_CHANNEL_TOTAL_BURSTS |
| 0x6E–0x75 | AXI_WRITE_CHANNEL_DATA_BYTES_WRITTEN_ID_0-7 |
| 0x76 | AXI_WRITE_CHANNEL_TOTAL_DATA_BYTES_WRITTEN |
| 0x77–0x7E | AXI_WRITE_RESPONSE_CHANNEL_RESPONSES_ID_0-7 |
| 0x7F | AXI_WRITE_RESPONSE_CHANNEL_TOTAL_RESPONSES |
| 0x80 | TOTAL_MMU_MISSES |
| 0x81 | MMU_READ_MISSES |
| 0x82 | MMU_WRITE_MISSES |
| 0x83 | TOTAL_MMU_HITS |
| 0x84 | MMU_READ_HITS |
| 0x85 | MMU_WRITE_HITS |
| 0x86 | SPLIT_MODE_TC_HITS |
| 0x87 | SPLIT_MODE_TC_MISSES |
| 0x88 | SPLIT_MODE_NON_TC_HITS |
| 0x89 | SPLIT_MODE_NON_TC_MISSES |
| 0x8A | STALL_AWAITING_TLB_MISS_FETCH |
| 0x8B | MMU_TLB_MISS_READ_BURSTS_RECEIVED |
| 0x8C | MMU_TLB_MISS_DATA_BEATS_READ |
| 0x8D | CP_CYCLES_HELD_OFF |
| 0x8E | VGT_CYCLES_HELD_OFF |
| 0x8F | TC_CYCLES_HELD_OFF |

**Table 31-18. Performance Counters (continued)**

| Select Value | Counter Name |
|---|---|
| 0x90 | TC_ROQ_CYCLES_HELD_OFF |
| 0x91 | TC_CYCLES_HELD_OFF_TCD_FULL |
| 0x92 | RB_CYCLES_HELD_OFF |
| 0x93 | PA_CYCLES_HELD_OFF |
| 0x94 | TOTAL_CYCLES_ANY_CLNT_HELD_OFF |
| 0x95 | TLB_MISS_CYCLES_HELD_OFF |
| 0x96 | AXI_READ_REQUEST_HELD_OFF |
| 0x97 | AXI_WRITE_REQUEST_HELD_OFF |
| 0x98 | AXI_REQUEST_HELD_OFF |
| 0x99 | AXI_REQUEST_HELD_OFF_INFLIGHT_LIMIT |
| 0x9A | AXI_WRITE_DATA_HELD_OFF |
| 0x9B | CP_SAME_PAGE_BANK_REQUESTS |
| 0x9C | VGT_SAME_PAGE_BANK_REQUESTS |
| 0x9D | TC_SAME_PAGE_BANK_REQUESTS |
| 0x9E | TC_ARB_HOLD_SAME_PAGE_BANK_REQUESTS |
| 0x9F | RB_SAME_PAGE_BANK_REQUESTS |
| 0xA0 | TOTAL_SAME_PAGE_BANK_REQUESTS |
| 0xA1 | CP_SAME_PAGE_BANK_REQUESTS_KILLED_FAIRNESS_LIMIT |
| 0xA2 | VGT_SAME_PAGE_BANK_REQUESTS_KILLED_FAIRNESS_LIMIT |
| 0xA3 | TC_SAME_PAGE_BANK_REQUESTS_KILLED_FAIRNESS_LIMIT |
| 0xA4 | RB_SAME_PAGE_BANK_REQUESTS_KILLED_FAIRNESS_LIMIT |
| 0xA5 | TOTAL_SAME_PAGE_BANK_KILLED_FAIRNESS_LIMIT |
| 0xA6 | TOTAL_MH_READ_REQUESTS |
| 0xA7 | TOTAL_MH_WRITE_REQUESTS |
| 0xA8 | TOTAL_MH_REQUESTS |
| 0xA9 | MH_BUSY |
| 0xAA | CP_NTH_ACCESS_SAME_PAGE_BANK_SEQUENCE |
| 0xAB | VGT_NTH_ACCESS_SAME_PAGE_BANK_SEQUENCE |
| 0xAC | TC_NTH_ACCESS_SAME_PAGE_BANK_SEQUENCE |
| 0xAD | RB_NTH_ACCESS_SAME_PAGE_BANK_SEQUENCE |
| 0xAE | TC_ROQ_N_VALID_ENTRIES |
| 0xAF | ARQ_N_ENTRIES |
| 0xB0 | WDB_N_ENTRIES |
| 0xB1 | MH_READ_LATENCY_START |
| 0xB2 | MH_READ_LATENCY_POST |
| 0xB3 | MC_TOTAL_READ_REQUESTS |
| 0xB4 | ELAPSED_CLK_CYCLES |
| 0xFF | NONE (defaut value, not counting) |

## 31.6 Functional Description

Figure 31-1 shows the GFX2D top level block diagram.

### 31.6.1 Bus interface

The bus interface has a 32-bit AHB slave port and a 64-bit AXI master port.

### 31.6.2 Input unit

This unit handles the command stream input through the slave interface, status register reads, and external interrupt generation with programmable interrupt handler. It connects to the slave port of the bus interface unit where the CPU can feed in commands to the graphics processor.

### 31.6.3 Arbiter

The arbiter sits between the rendering core and the bus interface, handling prioritized arbitration of memory reads and writes.

### 31.6.4 Burst cache

The burst cache handles the main memory traffic for the rendering pipeline. It is composed of 256-bit blocks. The burst cache intrinsically works with pixel groups and caches recently used pixels. It also stops pixels proceeding in the pixel pipeline, if the same memory location already has a pending write coming from the pixel pipeline. This protection is done separately for the various 2D and VG buffers.

### 31.6.5 2D+VG unit

This unit handles all 2D-bitmap related operations. It has a rectangle rasterizer, which generates the source and destination X and Y coordinates. The rasterizer can also operate in right-to-left and/or bottom-to-top direction.

Raster operation (ROP) is supported between the source, pattern/mask and destination data. The data is then written to the destination bitmap. Separate ROPs can be used for masked and unmasked pixels (ROP4). The color can also be alpha blended with the destination with either per pixel or constant alpha value.

This unit also contains a vector graphics rasterizer, which is capable of rasterizing anti-aliased complex polygons.

## 31.6.6 2D+VG unit level architecture

Figure 31-13 shows the architecture of the 2D+VG unit.

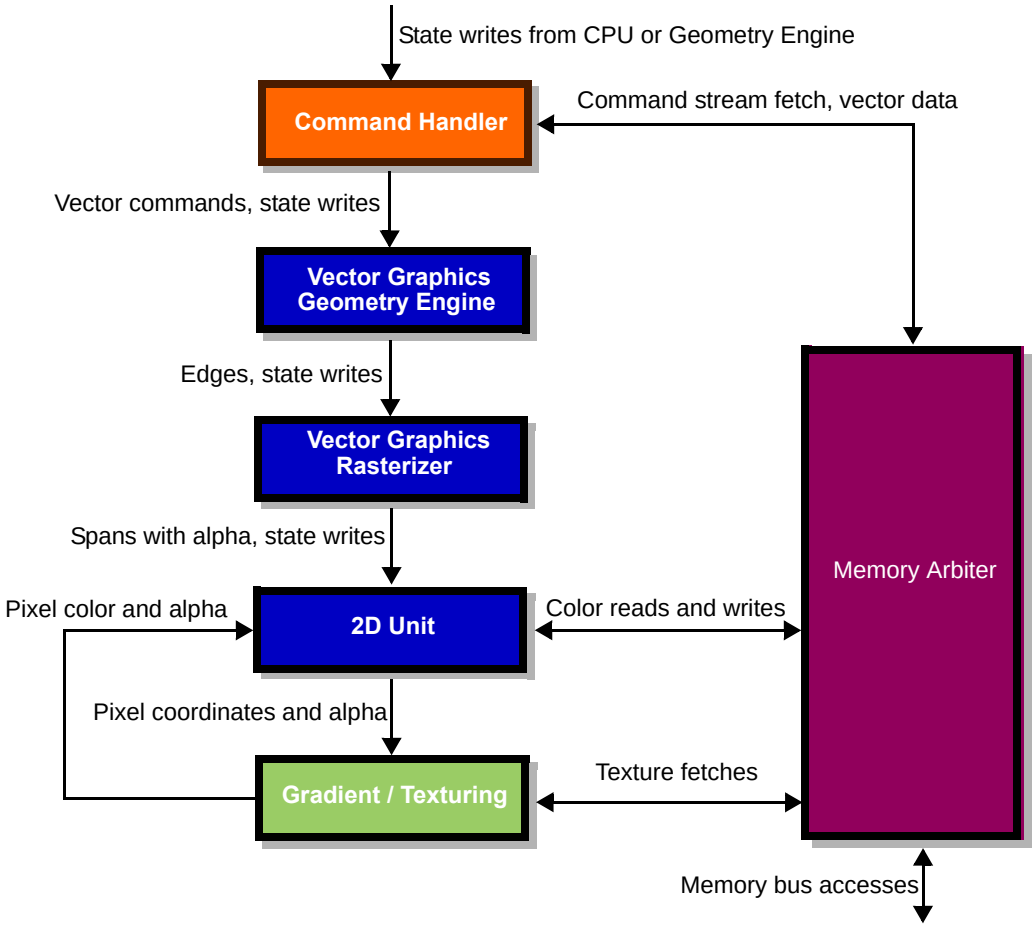



**Figure 31-13. 2D+VG Architecture**

### 31.6.6.1 Command Handler

The purpose of the Command Handler is to read command streams from memory and send them onwards to the vector graphics Geometry Engine and other units. These streams can be used like display lists in 3D graphics. The unit supports three data streams from which register addresses and data can be read in various formats. Multiple streams can also be used to implement subroutines in the display list. In addition the unit has a stream marker mechanism that can be used to extend an already executing command list and to report progress status back to the CPU through memory writes. Finally the unit can perform conditional register writes base on feedback information from the Geometry Engine. This can be used to perform bounding box tests and hardware based tiling.

#### 31.6.6.2 Geometry Engine

The vector graphics Geometry Engine performs geometry operations on primitives. Input primitives can be lines or curves (cubic/quadratic) specified with absolute or relative coordinates in a user specified coordinate system. Curves are first split into line segments according to a given accuracy goal. After this the resulting line path can optionally be converted into a thick stroke with possibly rounded joins and caps. The Geometry Engine can also perform bounding box tests where results are relayed back to the Command Handler.

#### 31.6.6.3 Rasterizer

The vector graphics Rasterizer handles rasterizing arbitrary vector graphic shapes to spans for the 2D unit.

#### 31.6.6.4 2D unit

The 2D unit performs block based 2D operations like fills and blits. It also manages drawing spans from the vector graphics rasterizer which are internally handled as one pixel height rectangles. If required, it also pipes the pixels through the gradient and texturing unit.

#### 31.6.6.5 Gradient and texturing unit

The gradient and texturing unit calculates color and alpha values for pixels from pixel coordinates and alpha according to the required gradient or texturing mode. Internally the unit is a very simple CPU with at maximum 8 instructions per pixel and a flexible ALU to perform all necessary operations like dot products, square root and division. The texturing part handles fetching textures from memory and filtering them.

This page is intentionally left blank.

# Chapter 32
# Periodic Interrupt Timer (PIT)

## 32.1 Introduction

The PIT is an array of timers that can be used to raise interrupts and trigger DMA channels.

This device has one PIT module with eight Timer Channels (PIT channels 0 through 7). These are connected to the Trigger input 0 through 7 of the DMA_MUX.

Figure 32-1 shows the PIT block diagram.

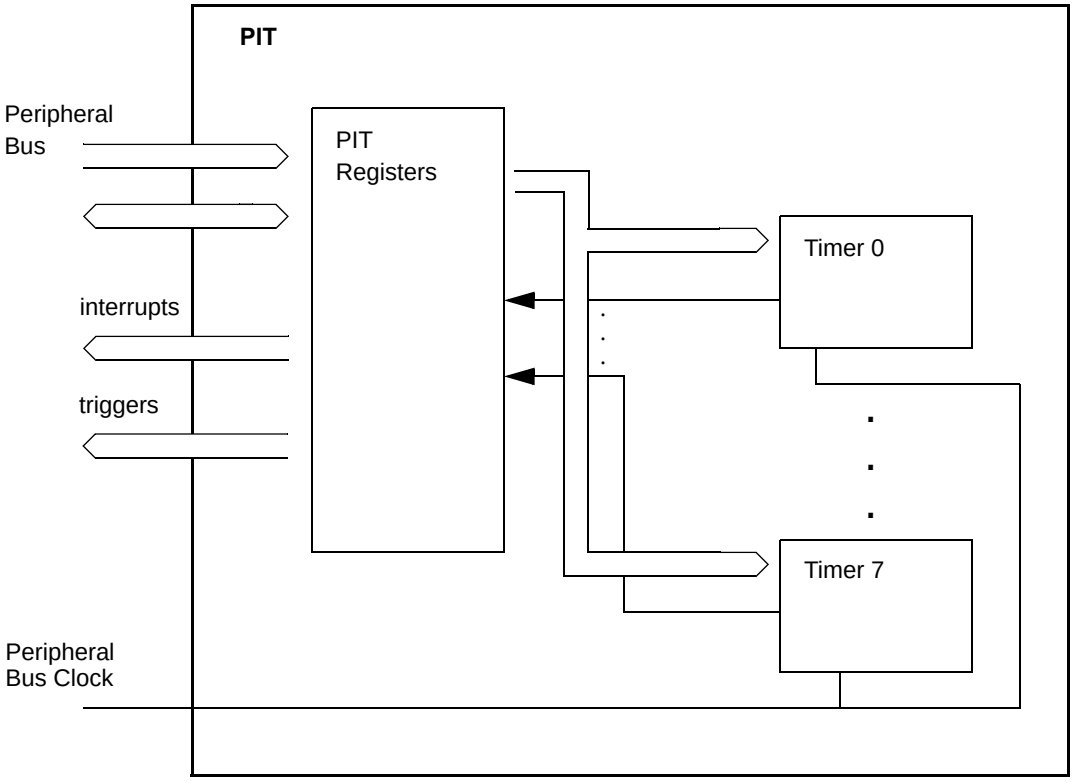



**Figure 32-1. PIT block diagram**

The main features of this block are:

- Timers can generate DMA trigger pulses
- Timers can generate interrupts
- All interrupts are maskable
- Independent timeout periods for each timer

## 32.2 Signal description

The PIT module has no external pins.

## 32.3 Memory map and register description

This section provides a detailed description of all registers accessible in the PIT module.

### 32.3.1 Memory map

Table 32-1 gives an overview on all PIT registers.

**Table 32-1. PIT memory map**

| Address Offset | Use | Location |
|---|---|---|
| 0x000 | PIT Module Control Register | on page 1201 |
| 0x004–0x0FC | Reserved | |
| 0x100–0x10C | Timer Channel 0 | See Table 32-2 |
| 0x110–0x11C | Timer Channel 1 | |
| 0x120–0x12C | Timer Channel 2 | |
| 0x130–0x13C | Timer Channel 3 | |
| 0x140–0x14C | Timer Channel 4 | |
| 0x150–0x15C | Timer Channel 5 | |
| 0x160–0x16C | Timer Channel 6 | |
| 0x170–0x17C | Timer Channel 7 | |
| 0x0180–0x1FC | Reserved | |

**Table 32-2. Timer Channel n**

| Address Offset | Use | Location |
|---|---|---|
| channel + 0x00 | Timer Load Value Register | on page 1201 |
| channel + 0x04 | Current Timer Value Register | on page 1202 |
| channel + 0x08 | Timer Control Register | on page 1203 |
| channel + 0x0C | Timer Flag Register | on page 1204 |

**NOTE**

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level.

Reserved registers will read as 0, writes will have no effect.

### 32.3.2 Register descriptions

This section describes in address order all the PIT registers and their individual bits.

### 32.3.2.1 PIT Module Control Register (PITMCR)

This register controls whether the timer clocks should be enabled and whether the timers should run in debug mode.

Offset 0x000 Access: Read/Write

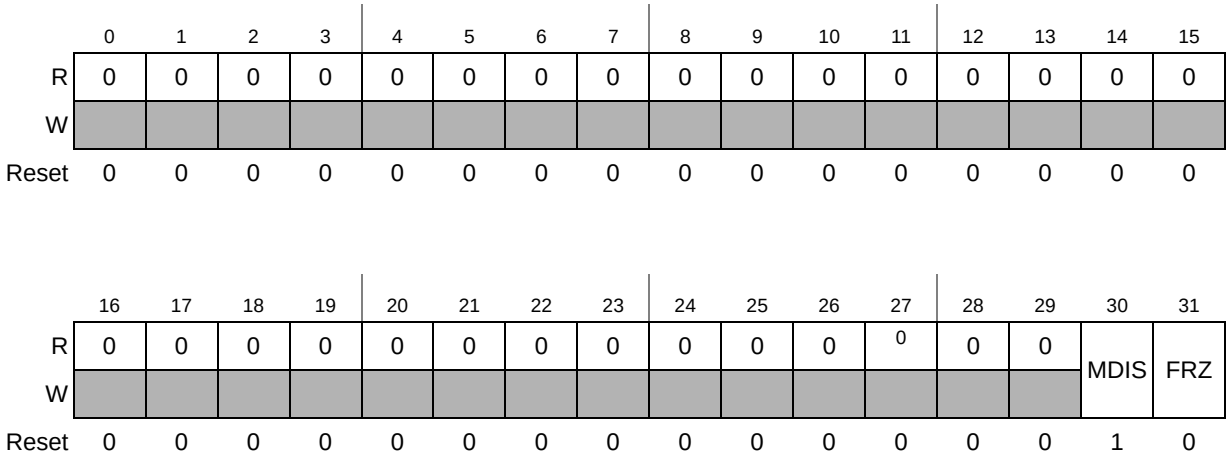



**Figure 32-2. PIT Module Control Registers (PITMCR)**

**Table 32-3. PITMCR Field Descriptions**

| Field | Description |
|---|---|
| MDIS | Module Disable. This is used to disable the module clock. This bit should be enabled before any other setup is done.<br>0 Clock for PIT Timers is enabled<br>1 Clock for PIT Timers is disabled (default) |
| FRZ | Freeze. Allows the timers to be stopped when the device enters debug mode.<br>0 = Timers continue to run in debug mode.<br>1 = Timers are stopped in debug mode. |

### 32.3.2.2 Timer Load Value Register (LDVAL)

These registers select the timeout period for the timer interrupts.

Offset channel_base + 0x00 Access: Read/Write

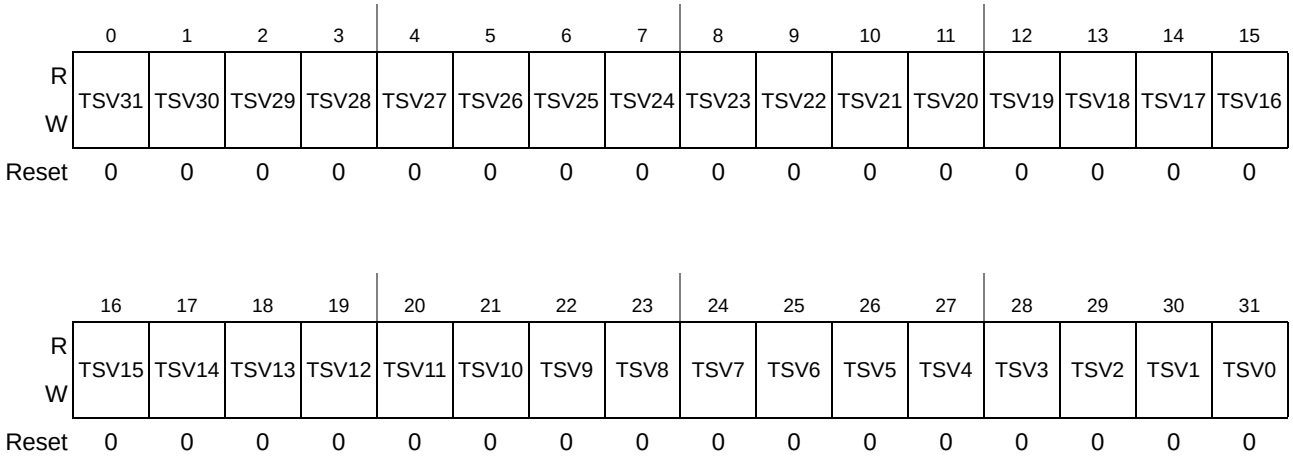



**Figure 32-3. Timer Load Value Register (LDVAL)**

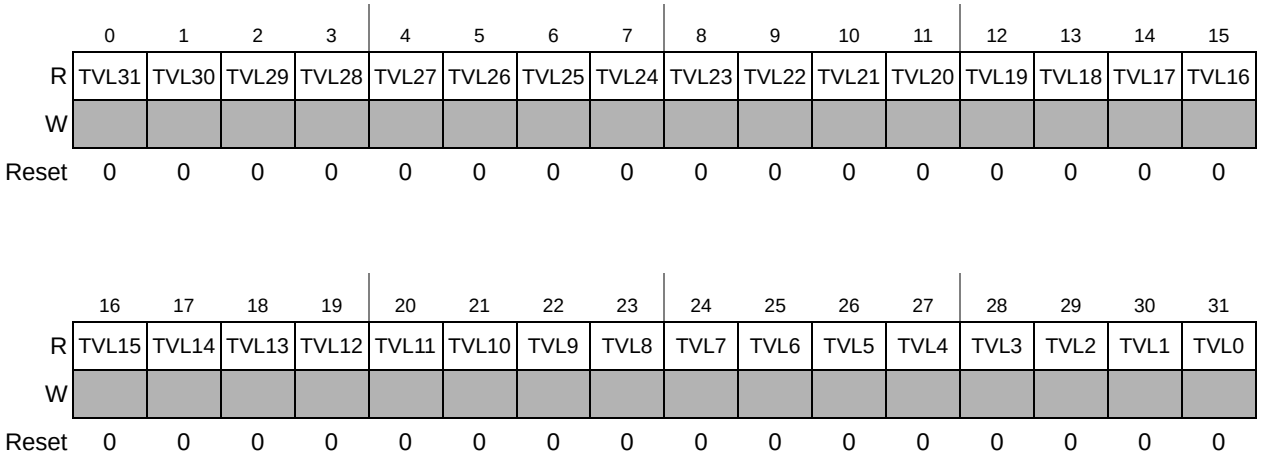
**Table 32-4. LDVAL Field Descriptions**

| Field | Description |
|---|---|
| TSV*n* | Time Start Value Bits. These bits set the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer, instead the value will be loaded once the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again (see Figure 32-8). |

### 32.3.2.3 Current Timer Value Register (CVAL)

These registers indicate the current timer position.

Offset channel_base + 0x04 Access: Read/Write



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TVL31 | TVL30 | TVL29 | TVL28 | TVL27 | TVL26 | TVL25 | TVL24 | TVL23 | TVL22 | TVL21 | TVL20 | TVL19 | TVL18 | TVL17 | TVL16 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TVL15 | TVL14 | TVL13 | TVL12 | TVL11 | TVL10 | TVL9 | TVL8 | TVL7 | TVL6 | TVL5 | TVL4 | TVL3 | TVL2 | TVL1 | TVL0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 32-4. Current Timer Value Register (CVAL)**

**Table 32-5. CVAL Field Descriptions**

| Field | Description |
|---|---|
| TVL*n* | Current Timer Value. These bits represent the current timer value. Note that the timer uses a downcounter.<br>NOTE: The timer values will be frozen in Debug mode if the FRZ bit is set in the PIT Module Control Register (see Figure 32-2) |

### 32.3.2.4 Timer Control Register (TCTRL)

These register contain the control bits for each timer.

Offset channel_base + 0x08 Access: Read/Write

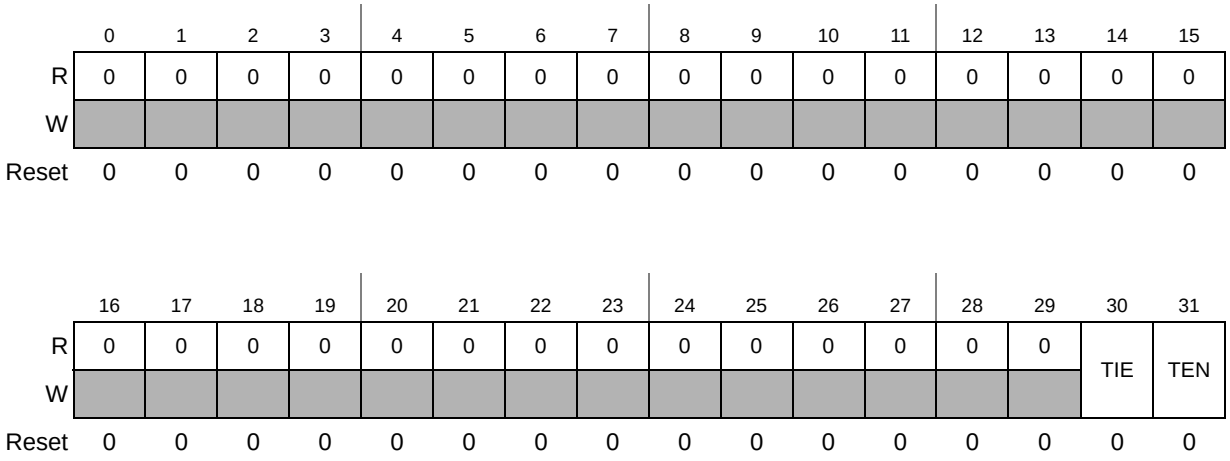



**Figure 32-5. Timer Control Register (TCTRL)**

**Table 32-6. TCTRL Field Descriptions**

| Field | Description |
|---|---|
| TIE | Timer Interrupt Enable Bit.<br>0 Interrupt requests from Timer x are disabled<br>1 Interrupt will be requested whenever TIF is set<br>When an interrupt is pending (TIF set), enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TIF flag must be cleared first. |
| TEN | Timer Enable Bit.<br>0 Timer will be disabled<br>1 Timer will be active |

### 32.3.2.5 Timer Flag Register (TFLG)

These registers hold the PIT interrupt flags.

Offset channel_base + 0x0C Access: Read/Write

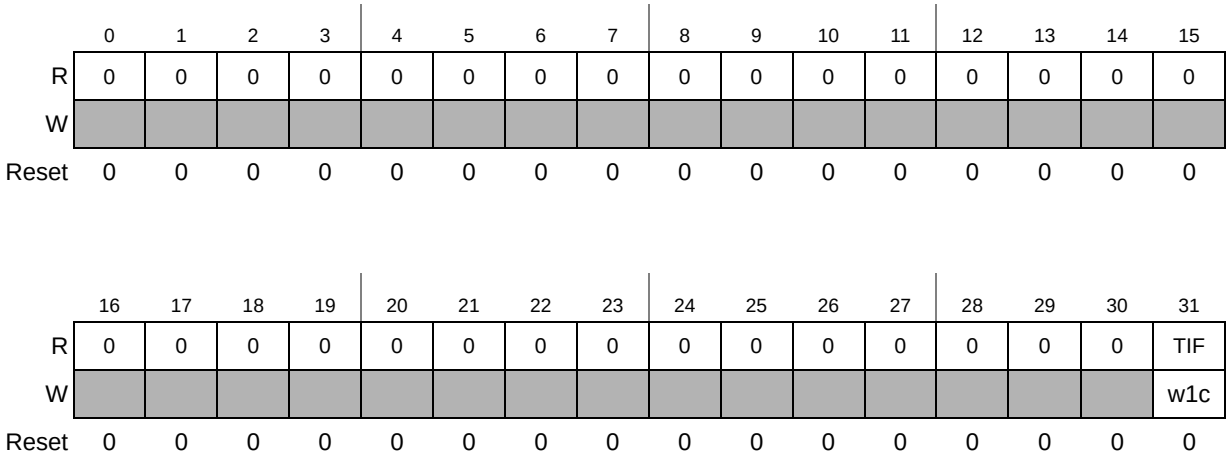



**Figure 32-6. Timer Flag Register (TFLG)**

**Table 32-7. TFLG Field Descriptions**

| Field | Description |
|---|---|
| TIF | Time Interrupt Flag. TIF is set to 1 at the end of the timer period.This flag can be cleared only by writing it with a 1. Writing a 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request.<br>0 Time-out has not yet occurred<br>1 Time-out has occurred |

## 32.4 Functional description

### 32.4.1 General

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses as well as to generate interrupts, each interrupt will be available on a separate interrupt line.

#### 32.4.1.1 Timers

The timers generate triggers at periodic intervals, when enabled. They load their start values, as specified in their LDVAL registers, then count down until they reach 0. Then they load their respective start value again. Each time a timer reaches 0, it will generate a trigger pulse, and set the interrupt flag.

All interrupts can be enabled or masked (by setting the TIE bits in the TCTRL registers). A new interrupt can be generated only after the previous one is cleared.

If desired, the current counter value of the timer can be read via the CVAL registers.

The counter period can be restarted, by first disabling, then enabling the timer with the TEN bit (see Figure 32-7).

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value and then enabling the timer again (see Figure 32-8).

It is also possible to change the counter period without restarting the timer by writing the LDVAL register with the new load value. This value will then be loaded after the next trigger event (see Figure 32-9).

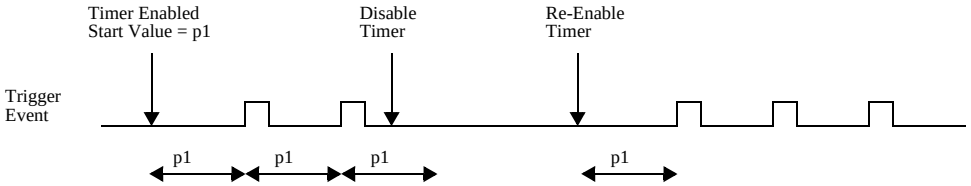



**Figure 32-7. Stopping and Starting a Timer**

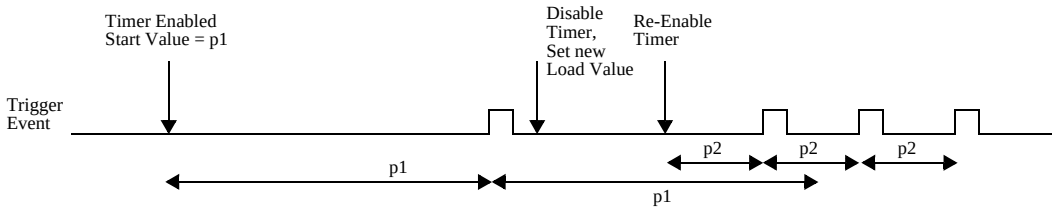



**Figure 32-8. Modifying Running Timer Period**

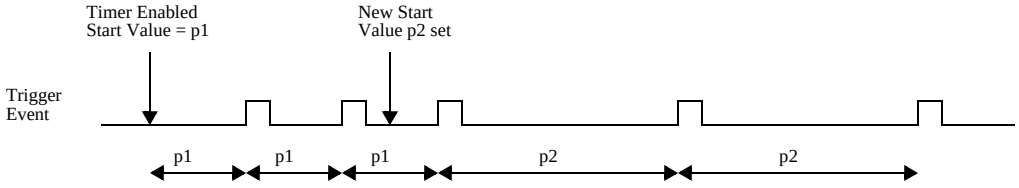



**Figure 32-9. Dynamically Setting a New Load Value**

### 32.4.1.2 Debug Mode

In Debug Mode the timers will be frozen - this is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system (e.g. the timer values) and then continue the operation.

## 32.4.2 Interrupts

All of the timers support interrupt generation. Refer to the MCU specifications for related vector addresses and priorities.

Timer interrupts can be disabled by setting the TIE bits to zero. The timer interrupt flags (TIF) are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to that TIF bit.

## 32.5 Initialization and Application Information

### 32.5.1 Example Configuration

In the example configuration:

- the PIT clock has a frequency of 50 MHz
- timer 1 shall create an interrupt every 5.12 ms
- timer 3 shall create a trigger event every 30 ms

First the PIT module needs to be activated by writing a 0 to the MDIS bit in the PITCTRL register.

The 50 MHz clock frequency equates to a clock period of 20 ns . Timer 1 needs to trigger every 5.12 ms/20 ns = 256000 cycles and timer 3 every 30 ms/20 ns = 1500000 cycles. The value for the LDVAL register trigger would be calculated as (period / clock period) -1.

This means that LDVAL1 with 0003E7FF hex and LDVAL3 with 0016E35F hex.

The interrupt for Timer 1 is enabled by setting TIE in the TCTRL1 register. The timer is started by writing a 1 to bit TEN in the TCTRL1 register.

Timer 3 shall be used only for triggering. Therefore Timer 3 is started by writing a 1 to bit TEN in the TCTRL3 register, bit TIE stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_CTRL = 0x00;

// RTI
PIT_RTI_LDVAL = 0x004C4B3F; // setup RTI for 5000000 cycles
PIT_RTI_TCTRL = PIT_TIE; // let RTI generate interrupts
PIT_RTI_TCTRL |= PIT_TEN; // start RTI

// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start timer 1

// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3for 1500000 cycles
PIT_TCTRL3 = TEN; // start timer 3
```

This page is intentionally left blank.

# Chapter 33
# Peripheral Bridge (PBRIDGE)

## 33.1 Introduction

The PBRIDGE is the interface between the system bus and on-chip peripherals. It has a hard-wired configuration and cannot be re-configured in software.

### 33.1.1 Overview

MPC5645S devices have one PBRIDGE, which provides an interface between the system bus and all lower bandwidth peripherals. Accesses that fall within the address space of the PBRIDGE are decoded to provide individual module selects for peripheral devices on the slave bus interface.

### 33.1.2 Features

The following list summarizes the key features of the PBRIDGE.

- Supports the slave interface signals. This interface is only meant for slave peripherals.
- Supports 32-bit slave peripherals. (Byte, halfword, and word reads and writes are supported to each.)

## 33.2 Functional description

The PBRIDGE serves as an interface between a system bus and the peripheral (slave) bus. It functions as a protocol translator. Accesses that fall within the address space of the PBRIDGE are decoded to provide individual module selects for peripheral devices on the slave bus interface.

### 33.2.1 Access support

Aligned 32-bit word accesses, halfword accesses, and byte accesses are supported for the peripherals. Peripheral registers must not be misaligned, although no explicit checking is performed by the PBRIDGE.

**NOTE**

Data accesses that cross a 32-bit boundary are not supported.

#### 33.2.1.1 Peripheral write buffering

Buffered writes are not supported by the MPC5645S PBRIDGE.

#### 33.2.1.2 Read cycles

Two-clock read accesses are possible with the PBRIDGE when the requested access size is 32-bits or smaller, and is not misaligned across a 32-bit boundary.

#### 33.2.1.3 Write cycles

Three-clock write accesses are possible with the PBRIDGE when the requested access size is 32-bits or smaller. Misaligned writes that cross a 32-bit boundary are not supported.

### 33.2.2 General operation

Slave peripherals are modules that contain readable/writable control and status registers. The system bus master reads and writes these registers through the PBRIDGE. The PBRIDGE generates module enables, the module address, transfer attributes, byte enables, and write data as inputs to the slave peripherals. The PBRIDGE captures read data from the slave interface and drives it on the system bus.

The PBRIDGE occupies a 64 MB portion of the address space. The register maps of the slave peripherals are located on 16-KB boundaries. Each slave peripheral is allocated one 16-KB block of the memory map, and is activated by one of the module enables from the PBRIDGE.

The PBRIDGE is responsible for indicating to slave peripherals if an access is in supervisor or user mode.

# Chapter 34
# Power Control Unit (MC_PCU)

## 34.1 Introduction

### 34.1.1 Overview

The power control unit (MC_PCU) is used to reduce the overall SoC power consumption. Power can be saved by disconnecting parts of the SoC from the power supply via a power switching device. The SoC is grouped into multiple parts having this capability which are called “power domains”.

When a power domain is disconnected from the supply, the power consumption is reduced to zero in that domain. Any status information of such a power domain is lost. When re-connecting a power domain to the supply voltage, the domain draws an increased current until the power domain reaches its operational voltage.

Power domains are controlled on a device mode basis. For each mode, software can configure whether a power domain is connected to the supply voltage (power-up state) or disconnected (power-down state). Maximum power saving is reached by entering the ***STANDBY*** mode.

On each mode change request, the MC_PCU evaluates the power domain settings in the power domain configuration registers and initiates a power-down or a power-up sequence for each individual power domain. The power-up/down sequences are handled by finite state machines to ensure a smooth and safe transition from one power state to the other.

Exiting the ***STANDBY*** mode can only be done via a system wakeup event as all power domains other than power domain #0 are in the power-down state.

In addition, the MC_PCU acts as a bridge for mapping the VREG peripheral to the MC_PCU address space.

Figure 34-1 depicts the MC_PCU block diagram.

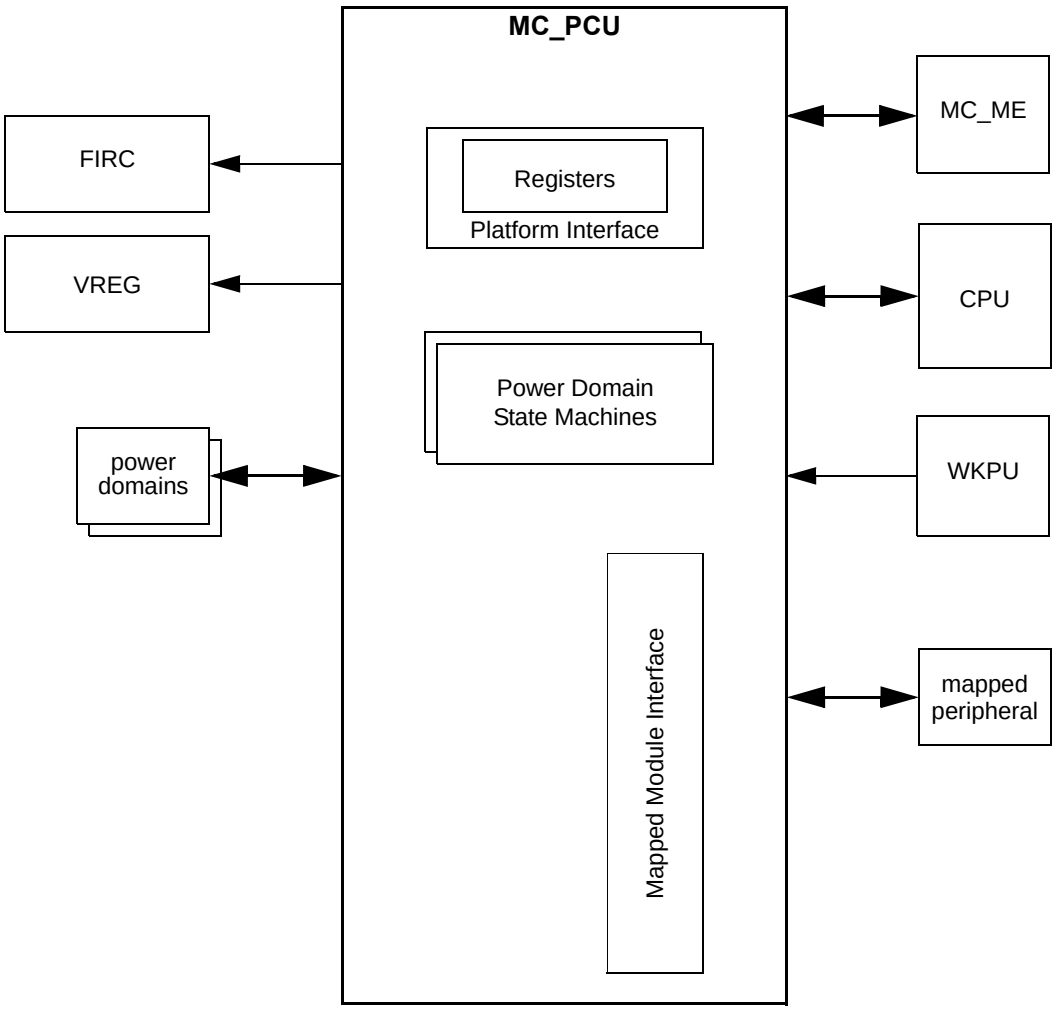



**Figure 34-1. MC_PCU block diagram**

### 34.1.2 Features

The MC_PCU includes the following features:

- support for 3 power domains
- support for device modes ***RESET***, ***DRUN***, ***SAFE***, ***TEST***, ***RUN0…3***, ***HALT***, ***STOP***, and ***STANDBY*** (for further mode details, please see the MC_ME chapter)
- power states updating on each mode change and on system wakeup
- a handshake mechanism for power state changes thus guaranteeing operable voltage
- maps the VREG registers to the MC_PCU address space

## 34.2 External Signal Description

The MC_PCU has no connections to any external pins.

## 34.3 Memory map and register definition

### 34.3.1 Memory Map

**Table 34-1. MC_PCU register description**

| Address | Name | Description | Size | Access | | Location |
|---|---|---|---|---|---|---|
| | | | | User | Supervisor | |
| 0xC3FE_8000 | PCU_PCONF0 | Power Domain #0 Configuration | word | read | read | on page 1215 |
| 0xC3FE_8004 | PCU_PCONF1 | Power Domain #1 Configuration | word | read | read | on page 1216 |
| 0xC3FE_8008 | PCU_PCONF2 | Power Domain #2 Configuration | word | read | read/write | on page 1217 |
| 0xC3FE_8040 | PCU_PSTAT | Power Domain Status Register | word | read | read | on page 1217 |

**NOTE**

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

**Table 34-2. MC_PCU memory map**

| Address | Name | | 0 | 1 | 2 | 3 | 27 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FE_8000 | PCU_PCONF0 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | STBY | 0 | 0 | STOP | 0 | HALT | RUN3 | RUN2 | RUN1 | RUN0 | DRUN | SAFE | TEST | RST |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_8004 | PCU_PCONF1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | STBY | 0 | 0 | STOP | 0 | HALT | RUN3 | RUN2 | RUN1 | RUN0 | DRUN | SAFE | TEST | RST |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_8008 | PCU_PCONF2 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | STBY | 0 | 0 | STOP | 0 | HALT | RUN3 | RUN2 | RUN1 | RUN0 | DRUN | SAFE | TEST | RST |
| | | W | | | | | | | | | | | | | | | | |

**Table 34-2. MC_PCU memory map (continued)**

| Address | Name | | 0<br>16 | 1<br>17 | 2<br>18 | 3<br>19 | 27<br>20 | 5<br>21 | 6<br>22 | 7<br>23 | 8<br>24 | 9<br>25 | 10<br>26 | 11<br>27 | 12<br>28 | 13<br>29 | 14<br>30 | 15<br>31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xC3FE_800C<br>…<br>0xC3FE_803C | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FE_8040 | PCU_PSTAT | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| | | R | | | | | | | | | | | | | | PD2 | PD1 | PD0 |
| | | W | | | | | | | | | | | | | | | | |
| 0x044<br>…<br>0x07C | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FE_8080<br>…<br>0xC3FE_80FC | VREG registers | | | | | | | | | | | | | | | | | |
| 0xC3FE_8100<br>…<br>0xC3FE_BFFC | reserved | | | | | | | | | | | | | | | | | |

## 34.3.2 Register Descriptions

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the **PD0** field of the **PCU_PSTAT** register may be accessed as a word at address 0xC3FE_8040, as a half-word at address 0xC3FE_8042, or as a byte at address 0xC3FE_8043.

### 34.3.2.1 Power Domain #0 Configuration Register (PCU_PCONF0)

Address 0xC3FE_8000 Access: User read, Supervisor read, Test read

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | STBY | 0 | 0 | STOP | 0 | HALT | RUN3 | RUN2 | RUN1 | RUN0 | DRUN | SAFE | TEST | RST |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 34-2. Power Domain #0 Configuration Register (PCU_PCONF0)**

This register defines for power domain #0 whether it is on or off in each device mode. As power domain #0 is the always-on power domain (and includes the MC_PCU), none of its bits are programmable. This register is available for completeness reasons.

**Table 34-3. Power Domain Configuration Register Field Descriptions**

| Field | Description |
|---|---|
| RST | Power domain control during ***RESET*** mode<br>0 Power domain off<br>1 Power domain on |
| TEST | Power domain control during ***TEST*** mode<br>0 Power domain off<br>1 Power domain on |
| SAFE | Power domain control during ***SAFE*** mode<br>0 Power domain off<br>1 Power domain on |
| DRUN | Power domain control during ***DRUN*** mode<br>0 Power domain off<br>1 Power domain on |
| RUN0 | Power domain control during ***RUN0*** mode<br>0 Power domain off<br>1 Power domain on |
| RUN1 | Power domain control during ***RUN1*** mode<br>0 Power domain off<br>1 Power domain on |
| RUN2 | Power domain control during ***RUN2*** mode<br>0 Power domain off<br>1 Power domain on |
| RUN3 | Power domain control during ***RUN3*** mode<br>0 Power domain off<br>1 Power domain on |

**Table 34-3. Power Domain Configuration Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| HALT | Power domain control during ***HALT*** mode<br>0 Power domain off<br>1 Power domain on |
| STOP | Power domain control during ***STOP*** mode<br>0 Power domain off<br>1 Power domain on |
| STBY | Power domain control during ***STANDBY*** mode<br>0 Power domain off<br>1 Power domain on |

### 34.3.2.2 Power Domain #1 Configuration Register (PCU_PCONF1)

Address 0xC3FE_8004 Access: User read, Supervisor read, Test read

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | STBY | 0 | 0 | STOP | 0 | HALT | RUN3 | RUN2 | RUN1 | RUN0 | DRUN | SAFE | TEST | RST |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 34-3. Power Domain #1 Configuration Register (PCU_PCONF1)**

This register defines for power domain #1 whether it is on or off in each device mode. The bit field description is the same as in Table 34-3. As the platform, clock generation, and mode control reside in power domain #1, this power domain is only powered down during the ***STANDBY*** mode. Therefore, none of the bits is programmable. This register is available for completeness reasons.

The difference between **PCU_PCONF0** and **PCU_PCONF1** is the reset value of the **STBY** bit: During the ***STANDBY*** mode, power domain #1 is disconnected from the power supply, and therefore **PCU_PCONF1.STBY** is always ‘0’. Power domain #0 is always on, and therefore **PCU_PCONF0.STBY** is ‘1’.

For further details about ***STANDBY*** mode, please refer to Section 34.4.4.2, “STANDBY Mode Transition.

### 34.3.2.3 Power Domain #2 Configuration Register (PCU_PCONF2)

Address 0xC3FE_8008 Access: User read, Supervisor read/write, Test read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | STBY | 0 | 0 | STOP | 0 | HALT | RUN3 | RUN2 | RUN1 | RUN0 | DRUN | SAFE | TEST | RST |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 34-4. Power Domain #2 Configuration Register (PCU_PCONF2)**

This register defines for power domain #2 whether it is on or off in each device mode. The bit field description is the same as in Table 34-3.

### 34.3.2.4 Power Domain Status Register (PCU_PSTAT)

Address 0xC3FE_8040 Access: User read, Supervisor read, Test read

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | PD2 | PD1 | PD0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

**Figure 34-5. Power Domain Status Register (PCU_PSTAT)**

This register reflects the power status of all available power domains.

**Table 34-4. Power Domain Status Register (PCU_PSTAT) Field Descriptions**

| Field | Description |
|---|---|
| PD*n* | Power status for power domain #*n*<br>0 Power domain is inoperable<br>1 Power domain is operable |

## 34.4 Functional Description

### 34.4.1 General

The MC_PCU controls all available power domains on a device mode basis. The **PCU_PCONFn** registers specify during which system/user modes a power domain is powered up. The power state for each individual power domain is reflected by the bits in the **PCU_PSTAT** register.

On a mode change, the MC_PCU evaluates which power domain(s) must change power state. The power state is controlled by a state machine (FSM) for each individual power domain (see Figure 34-1) which ensures a clean and safe state transition.

### 34.4.2 Reset / Power-On Reset

After any reset, the SoC will transition to the ***RESET*** mode during which all power domains are powered up (see the MC_ME chapter). Once the reset sequence has been completed, the ***DRUN*** mode is entered and software can begin the MC_PCU configuration.

### 34.4.3 MC_PCU Configuration

Per default, all power domains are powered in all modes other than ***STANDBY***. Software can change the configuration for each power domain on a mode basis by programming the **PCU_PCONFn** registers.

Each power domain which is powered down is held in a reset state. Read/write accesses to peripherals in those power domains will result in a transfer error.

### 34.4.4 Mode Transitions

On a mode change requested by the MC_ME, the MC_PCU evaluates the power configurations for all power domains. It compares the settings in the **PCU_PCONFn** registers for the new mode with the settings for the current mode. If the configuration for a power domain differs between the modes, a power state change request is generated. These requests are handled by a finite state machine to ensure a smooth and safe transition from one power state to another.

#### 34.4.4.1 *DRUN*, *SAFE*, *TEST*, *RUN0…3*, *HALT*, and *STOP* Mode Transition

The ***DRUN***, ***SAFE***, ***TEST***, ***RUN0…3***, ***HALT***, and ***STOP*** modes allow an increased power saving. The level of power saving is software-controllable via the settings in the **PCU_PCONFn** registers for power domain #2 onwards. The settings for power domains #0 and #1 can not be changed. Therefore, power domains #0 and #1 remain connected to the power supply for all modes beside ***STANDBY***.

Figure 34-6 shows an example for a mode transition from ***RUN0*** to ***HALT*** and back, which will result in power domain #2 being powered down during the ***HALT*** mode. In this case, **PCU_PCONF2.HALT** is programmed to be ‘0’.

When the MC_PCU receives the mode change request to ***HALT*** mode, it starts its power-down phase. During the power-down phase, clocks are disabled and the reset is asserted resulting in a loss of all information for this power domain.

Then the power domain is disconnected from the power supply (power-down state).

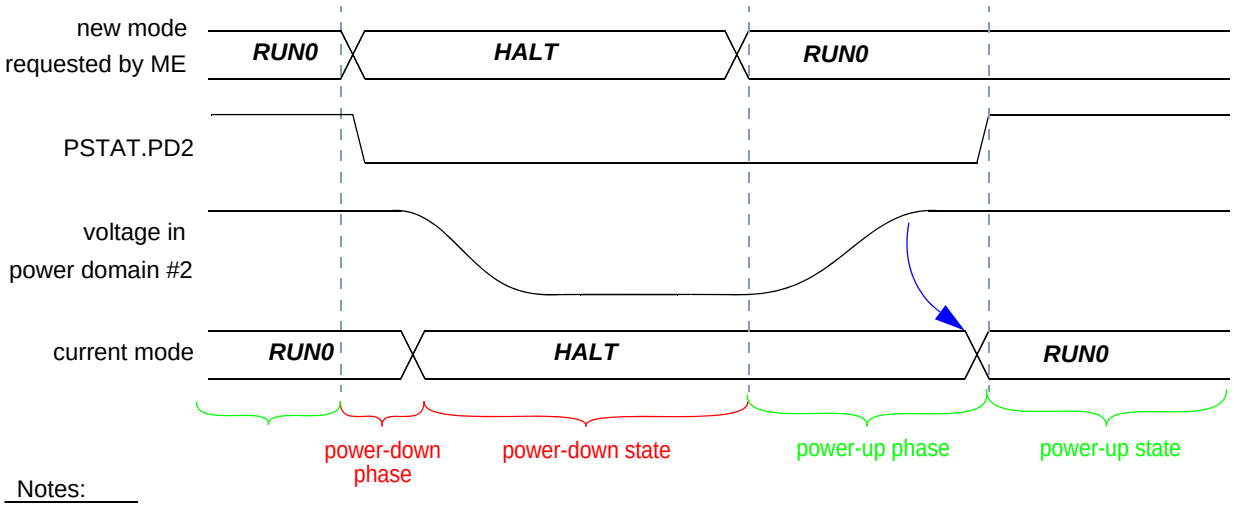



**Figure 34-6. MC_PCU Events During Power Sequences (non-*STANDBY* mode)**

When the MC_PCU receives a mode change request to ***RUN0***, it starts its power-up phase if **PCU_PCONF2.RUN0** is '1'. The power domain is re-connected to the power supply, and the voltage in power domain #2 will increase slowly. Once the voltage of power domain #2 is within an operable range, its clocks are enabled, and its resets are deasserted (power-up state).

**NOTE**

It is possible that, due to a mode change, power-up is requested before a power domain completed its power-down sequence. In this case, the information in that power domain is lost.

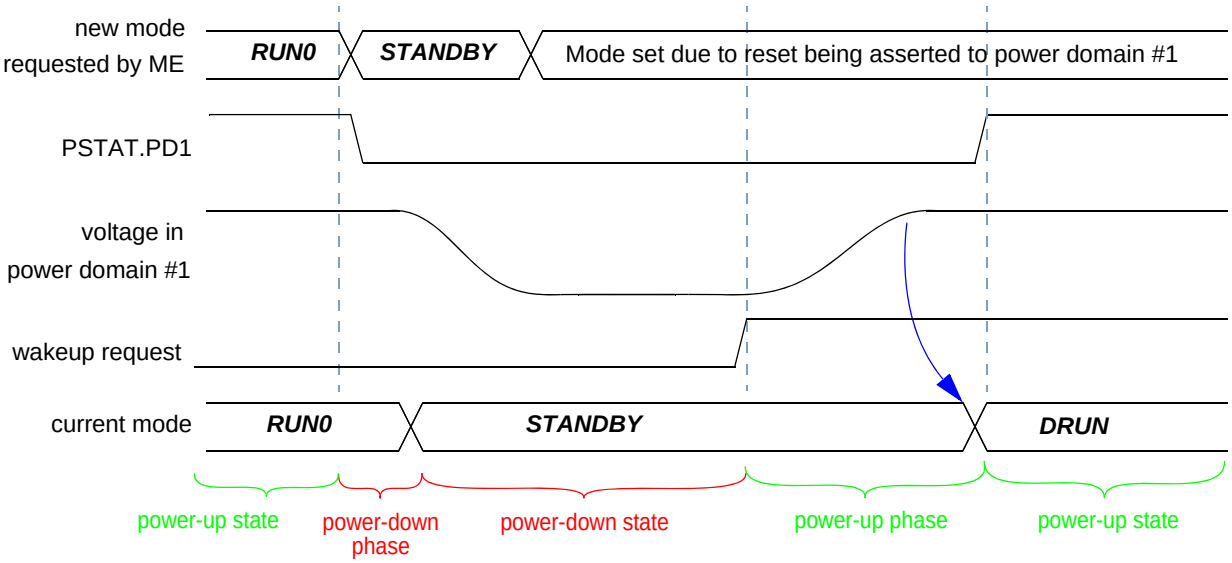
### 34.4.4.2 *STANDBY* Mode Transition

***STANDBY*** offers the maximum power saving. The level of power saving is software-controllable via the settings in the **PCU_PCONFn** registers for power domain #2 onwards. Power domain #0 stays connected to the power supply while power domain #1 is disconnected from the power supply. Amongst others power domain #1 contains the platform and the MC_ME. Therefore this mode differs from all other user/system modes.

Once ***STANDBY*** is entered it can only be left via a system wakeup. On exiting the ***STANDBY*** mode, all power domains are powered up according to the settings in the **PCU_PCONFn** registers, and the ***DRUN*** mode is entered. In ***DRUN*** mode, at least power domains #0 and #1 are powered.

Figure 34-7 shows an example for a mode transition from ***RUN0*** to ***STANDBY*** to ***DRUN***. All power domains which have **PCU_PCONFn.STBY** cleared will enter power-down phase. In this example only power domain #1 will be disabled during ***STANDBY*** mode.

When the MC_PCU receives the mode change request to ***STANDBY*** mode it starts the power down phase for power domain #1. During the power down phase, clocks are disabled and reset is asserted resulting in a loss of all information for this power domain. Then the power domain is disconnected from the power supply (power-down state).




Notes:

Not drawn to scale; PCONF1.RUN0 = 1; PCONF1.STBY = 0

**Figure 34-7. MC_PCU Events During Power Sequences (*STANDBY* mode)**

When the MC_PCU receives a system wakeup request, it starts the power-up phase. The power domain is re-connected to the power supply and the voltage in power domain #1 will increase slowly. Once the voltage is in an operable range, clocks are enabled and the reset is be deasserted (power-up state).

**NOTE**

It is possible that due to a wakeup request, power-up is requested before a power domain completed its power-down sequence. In this case, the information in that power domain is lost.

### 34.4.4.3 Power Saving for Memories During *STANDBY* Mode

All memories which are not powered down during ***STANDBY*** mode automatically enter a power saving state. No software configuration is required to enable this power saving state. While a memory is residing in this state an increased power saving is achieved. Data in the memories is retained.

## 34.5 Initialization Information

To initialize the MC_PCU, the registers **PCU_PCONF2…** should be programmed. After programming is done, those registers should no longer be changed.

## 34.6 Application Information

### 34.6.1 *STANDBY* Mode Considerations

***STANDBY*** offers maximum power saving possibility. But power is only saved during the time a power domain is disconnected from the supply. Increased power is required when a power domain is re-connected to the power supply. Additional power is required during restoring the information (e.g., in the platform). Care should be taken that the time during which the SoC is operating in ***STANDBY*** mode is significantly longer than the required time for restoring the information.

This page is intentionally left blank.

# Chapter 35
# Quad Serial Peripheral Interface (QuadSPI)

## 35.1 Introduction

The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to one single or two external serial flash devices, each with up to four bidirectional data lines.

### 35.1.1 Features

The QuadSPI supports the following features:

- Compatible with Winbond and Spansion as vendors of SPI Serial Flash devices.
- Single, dual and quad mode of operation.
- Supports both 24-bit and 32-bit addresses.
- Two identical serial flash devices can be connected and accessed in parallel for data read operations, forming one (virtual) flash memory with doubled readout bandwidth.
- DMA support to read RX Buffer data via AMBA AHB bus (64 bit width interface) or IP registers space (32 bit access).
- Inner loop size of DMA access can be configured.
- In total 14 interrupt conditions are mapped to 5 different interrupt lines (see Table 35-27)
- Memory mapped read access to connected flash devices.
  - Supports flash devices of up to 128 MB in size
  - Appropriate command sequence for flash read triggered automatically by read access
- Automatic divide by 2 of the serial flash device clock for commands not supporting the full frequency range.
- Supports both 24 and 32 bit address mode.

Additionally, the module supports a Stop Mode for power-saving purposes. This is independent from any low power modes on the external QuadSPI memory device.

### 35.1.2 Block Diagram

Figure 35-1 is a block diagram of the Quad Serial Peripheral Interface (QuadSPI) module.

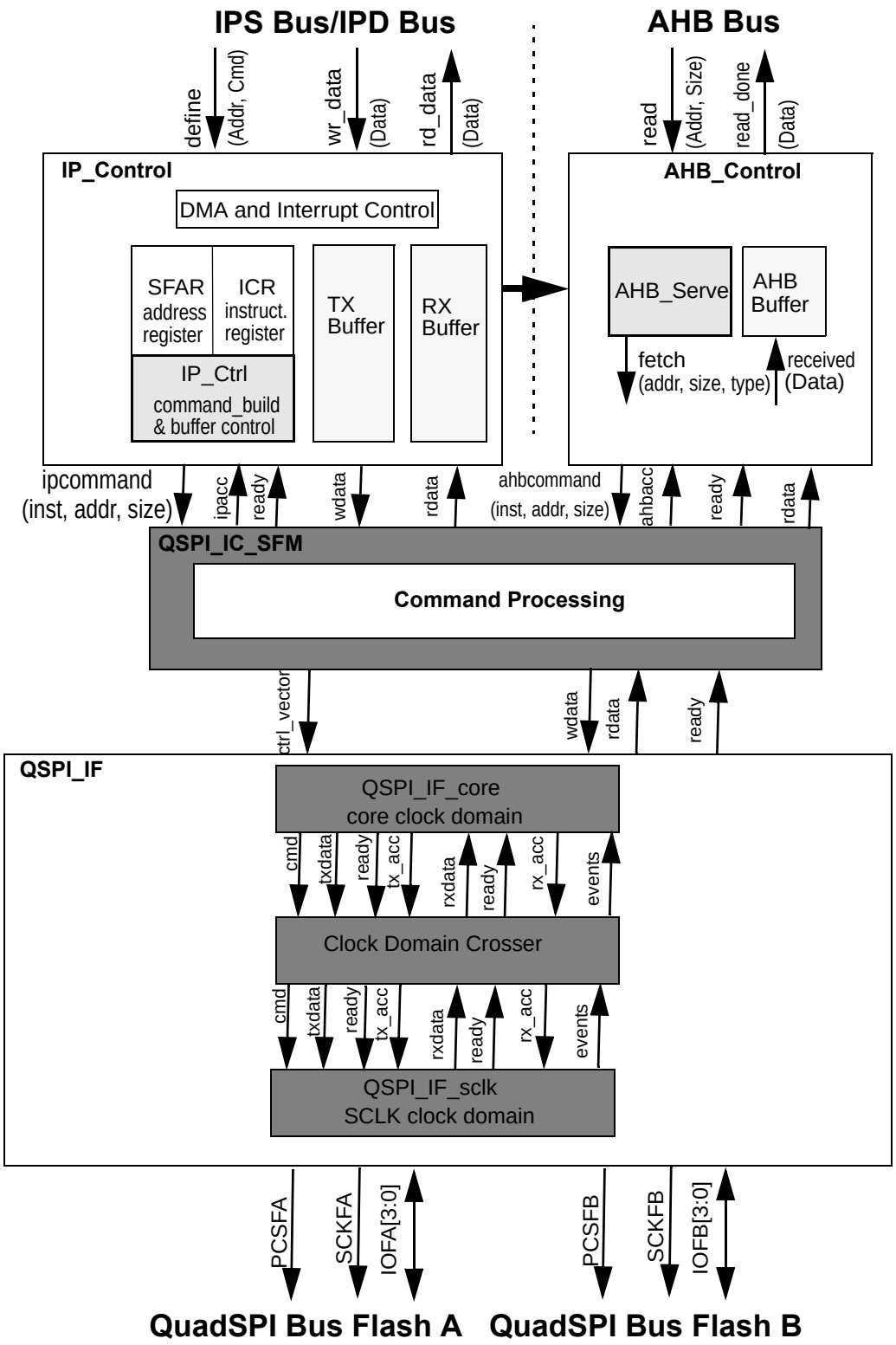



**Figure 35-1. QuadSPI block diagram**

## 35.1.3 QuadSPI modes of operation

This sections provides information about the modes in which the QuadSPI module can be used.

### 35.1.3.1 Normal Mode

In this mode one or two external serial flash memory device can be accessed. Further details about this mode of operation can be found in chapter Section 35.5.3, “Normal Mode“”.

### 35.1.3.2 Module Disable Mode

The Module Disable Mode is used for power management of the device containing the QuadSPI module, it is controlled by signals external to the QuadSPI. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in the Module Disable Mode. See Section 35.5.4.2, “Module Disable Mode“”.

### 35.1.3.3 Stop Mode

The Stop Mode is also used for power management. When a request is made to enter Stop Mode, the QuadSPI block completes the action currently processed. Then the request is acknowledged.

## 35.1.4 Glossary for QuadSPI module

**Table 35-1. Glossary**

| Term | Definition |
|---|---|
| AHB Command | An AHB Command is a SFM Command triggered by a read access to the address range belonging to the memory mapped access defined in Table 35-23. Refer to Section 35.6.2.2, “AHB Commands“” for details. |
| Asserted | A signal that is asserted is in its active state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one. |
| Clear | To clear a bit or bits means to establish logic level zero on the bit or bits. |
| Clock Phase | Determines when the data should be sampled relative to the active edge of SCK |
| Clock Polarity | Determines the idle state of the SCK signal. |
| Drain | To remove entries from a FIFO by software or hardware. |
| Endianess | Byte Ordering scheme. |
| Field | Two or more register bits grouped together. |
| Fill | To add entries to a FIFO by software or hardware. |
| Host | Refers to another functional block in the device containing the QuadSPI module |
| Instruction Code | 8 bits defining the type of command to be executed. |
| IP Command | A IP Command is a SFM Command triggered by writing into the QSPI_MCR[IC] field. |
| Logic level one | The voltage that corresponds to Boolean true (1) state. |
| Logic level zero | The voltage that corresponds to Boolean false (0) state. |

Table 35-1. Glossary (continued)

| Term | Definition |
|---|---|
| Negated | A signal that is negated is in its inactive state. An active low signal changes from logic level ‘0’ to logic level ‘1’ when negated, and an active high signal changes from logic level ‘1’ to logic level ‘0’. |
| Set | To set a bit or bits means to establish logic level one on the bit or bits. |
| RX Buffer PUSH Event | Addition of valid entries into the RX Buffer. In the default case each Buffer PUSH Event adds 2 entries to the RX Buffer since the interface to the serial clock domain is 64 bits in width. Depending from the number of bytes read from the serial flash device it is possible for the very last Buffer PUSH Event that only one entry is added.<br>The QSPI_RBSR[RDBFL] field is incremented by the number of entries added to the RX Buffer. |
| RX Buffer POP Event | Removal of valid entries from the RX Buffer. Each Buffer POP Event removes (QSPI_RBCT[WMRK] + 1) valid entries from the buffer. The QSPI_RBSR[RDBFL] field is decremented by the same number and the QSPI_RBSR[RDCTR] field is incremented accordingly. |
| Individual Flash Mode | Access to a single, individual serial flash device. Refer to Section 35.5.1, “Serial Flash Access Schemes“” for details. |
| Parallel Flash Mode | Read access to two serial flash devices attached to the QuadSPI module in parallel. Refer to Section 35.5.1, “Serial Flash Access Schemes“” for details. |
| SFM Command | Serial Flash Memory Command. A SFM command consists of an instruction code and all other parameters (e.g. size or mode bytes) needed for that specific instruction code. Triggering a command either initiates a transaction on the external serial flash or results in an error. Refer to Table 35-36 for details. |
| Single/Dual/Quad Instructions | Depending from the serial flash device connected to the QuadSPI module there will be instructions using a different number of data lines.<br>• Single Single line I/O with one data out and one data in line to/from the serial flash device<br>• Dual Dual line I/O with 2 bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI module<br>• Quad Quad line I/O with 4 bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI module |
| Transaction | A transaction consists of all flags, data and signals in either direction to execute a command for an attached serial flash device. It is a combination of chip select, sclk, instruction code, address, mode- and/or dummy bytes, transmit and/or receive data. |

## 35.2 External Signal Description

### 35.2.1 Overview

Table 35-2 lists the signals of the external signals belonging to the QuadSPI module in conjunction with the different modes of operation:

Table 35-2. Signal Properties

| Signal Name | Function | Direction |
|---|---|---|
| PCSFA | Peripheral Chip Select Flash A | Output |
| PCSFB | Peripheral Chip Select Flash B | Output |
| SCKFA | Serial Clock Flash A | Output |

Table 35-2. Signal Properties

| Signal Name | Function | Direction |
|---|---|---|
| SCKFB | Serial Clock Flash B | Output |
| IOFA[3:0] | Serial I/O Flash A | Bidir |
| IOFB[3:0] | Serial I/O Flash B | Bidir |

## 35.2.2 Detailed Signal Description

The following paragraphs describe the function of the signals given in Table 35-2 in more detail. Only the modes relevant to the specific signal are mentioned according to Table 35-2.

### 35.2.2.1 PCSFA - Peripheral Chip Select Flash A

This signal is the chip select for the serial flash device A.

### 35.2.2.2 PCSFB - Peripheral Chip Select Flash B

This signal is the chip select for the serial flash device B.

### 35.2.2.3 SCKFA — Serial Clock Flash A

This signal is the serial clock output to the serial flash device A.

### 35.2.2.4 SCKFB — Serial Clock Flash B

This signal is the serial clock output to the serial flash device B.

### 35.2.2.5 IOFA[3:0] - Data IO Flash A

These signals are the data I/O lines to/from the serial flash device A. Refer to Section 35.2.3, “Driving of External Signals“” for details about the signal drive and timing behavior. Note that the signal pins of the serial flash device may change their function according to the SFM Command executed, leaving them as control inputs when Single and Dual Instructions are executed. The QuadSPI module drives these signals high During the execution of Single and Dual Instructions.

### 35.2.2.6 IOFA[3:0] - Data IO Flash B

These signals are the data I/O lines to/from the serial flash device B. Refer to Section 35.2.3, “Driving of External Signals“” for details about the signal drive and timing behavior. Note that the signal pins of the serial flash device may change their function according to the SFM Command executed, leaving them as control inputs when Single and Dual Instructions are executed. The QuadSPI module drives these signals high During the execution of Single and Dual Instructions.

### 35.2.3 Driving of External Signals

The different phases of serial flash access scheme are shown in Figure 35-2 below:

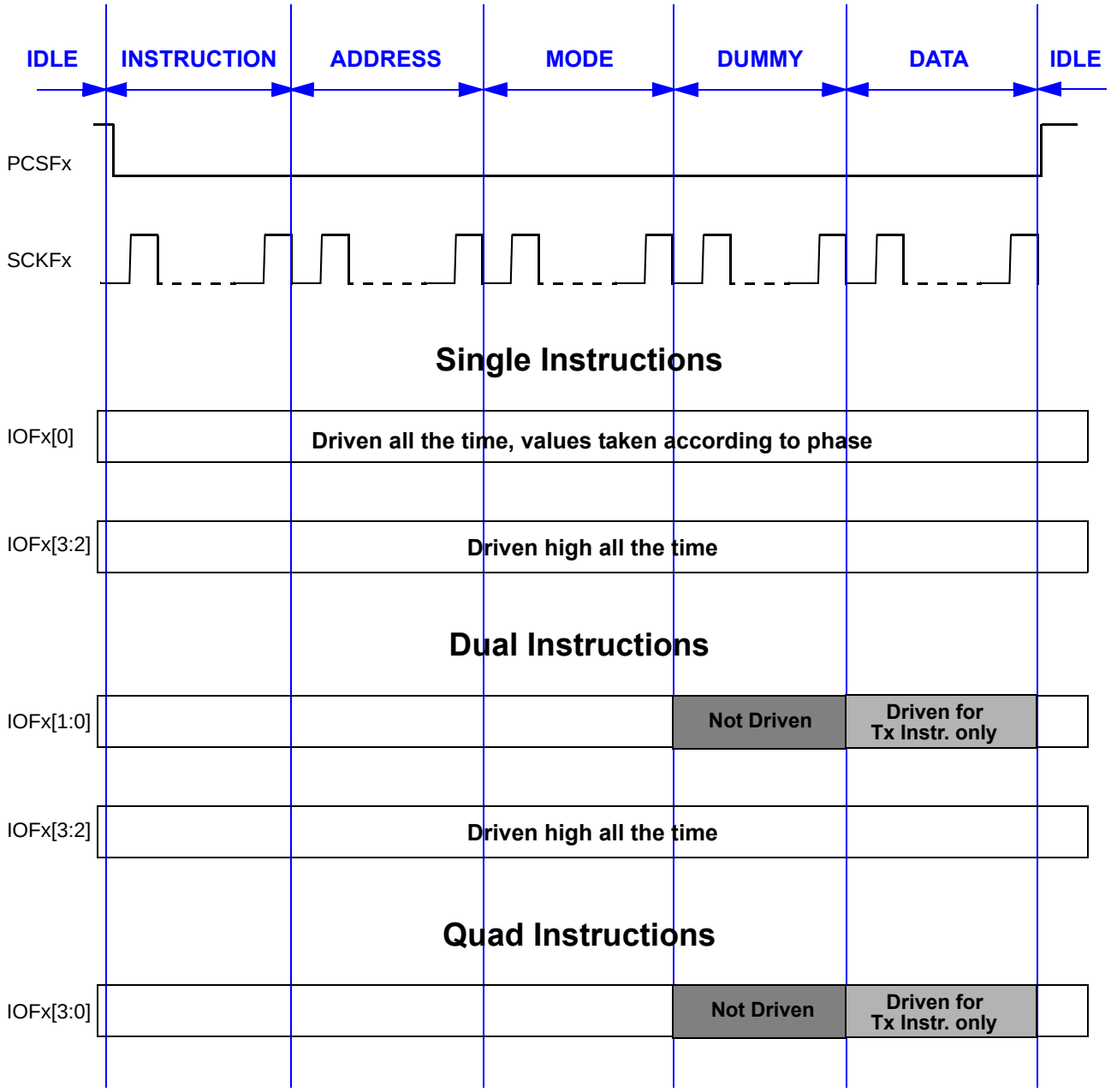



**Figure 35-2. Serial Flash Access Scheme**

The different phases and the I/O driving characteristics of the QuadSPI module are characterized in the following way:

- IDLE: Serial flash device not selected. No interaction with the serial flash device. All IOFx signals driven.

- INSTRUCTION: Serial flash device selected. The instruction is sent to the serial flash device. All IOFx signals are driven.
- ADDRESS: Serial Flash Address is sent to the device. All IOFx signals are driven. Note that this phase is not applicable for all SFM Commands.
- MODE: Mode bytes are sent to the serial flash device. All IOFx signals are driven. Note that this phase is not applicable for all SFM Commands.
- DUMMY: Dummy clocks are provided to the serial flash device. Refer to Figure 35-2 for the IOFx signals driven. The actual data lines required for the SFM Command executed are not driven for data read commands. Note that this phase is not applicable for all SFM Commands.
- DATA: Serial flash data are sent to or received from the serial flash device. Refer to Figure 35-2 for the IOFx signals driven. The actual data lines required for the SFM Command executed are not driven for data read commands. Note that this phase is not applicable for all SFM Commands.

The PCSFx and SCKFx signals are driven permanently throughout all the phases.

In Individual Flash Mode this applies to the selected flash device. In Parallel Flash Mode this applies to both serial flash devices simultaneously.

## 35.3 Memory map and register definition

### 35.3.1 Memory map

Table 35-3 shows the QuadSPI memory map.

**Table 35-3. QuadSPI memory map**

| Address | Register name | Location |
|---|---|---|
| QSPI_BASE+0x000 | Module Configuration Register (QSPI_MCR) | on page 1232 |
| QSPI_BASE+0x004 | Latency Configuration Register (QSPI_LCR) | on page 1234 |
| QSPI_BASE+0x004 - QSPI_BASE+0x0FC | Reserved | |
| QSPI_BASE+0x100 | Serial Flash Address Register (QSPI_SFAR) | on page 1235 |
| QSPI_BASE+0x104 | Instruction Code Register (QSPI_ICR) | on page 1236 |
| QSPI_BASE+0x108 | Sampling Register (QSPI_SMPR) | on page 1237 |
| QSPI_BASE+0x10C | RX Buffer Status Register (QSPI_RBSR) | on page 1238 |
| QSPI_BASE+0x110 | RX Buffer Control Register (QSPI_RBCT) | on page 1239 |
| QSPI_BASE+0x114 – QSPI_BASE+0x14C | Reserved | |
| QSPI_BASE+0x150 | TX Buffer Status Register (QSPI_TBSR) | on page 1239 |
| QSPI_BASE+0x154 | TX Buffer Data Register (QSPI_TBDR) | on page 1240 |
| QSPI_BASE+0x158 | AMBA Control Register (QSPI_ACR) | on page 1241 |

Table 35-3. QuadSPI memory map (continued)

| Address | Register name | Location |
|---|---|---|
| QSPI_BASE+0x15C | Status Register (QSPI_SFMSR) | on page 1242 |
| QSPI_BASE+0x160 | Flag Register (QSPI_SFMFR) | on page 1243 |
| QSPI_BASE+0x164 | Interrupt and DMA Request Select and Enable Register (QSPI_SFMRSER) | on page 1245 |
| QSPI_BASE+0x168 – QSPI_BASE+0x1FC | Reserved | |
| QSPI_BASE+0x200 – QSPI_BASE+0x27C | RX Buffer Data Registers 0–31 (QSPI_RBDR0–QSPI_RBDR31) | on page 1247 |

## 35.3.2 Serial Flash Address Assignment

In Table 35-4 below it is noted how the different access modes are related to the address specified for the next SFM Command. Note that this address assignment is valid for IP Commands as well as AHB Command. For IP Commands it is the value to be programmed into the QSPI_SFAR register, for AHB Commands it is the address accessed via the AHB bus.

Table 35-4. Serial Flash Address Assignment

| Address | Access Mode |
|---|---|
| 0x7000_0000 ... 0x77FF_FFFF (128 MB) | Individual Flash Mode - Flash A only |
| 0x7800_0000 ... 0x7FFF_FFFF (128 MB) | Individual Flash Mode - Flash B only |
| 0x8000_0000 ... 0X8FFF_FFFF (256 MB) | Parallel Flash Mode - Flash A and B in parallel |

**NOTE**

Any read access to non-implemented addresses will provide undefined results.

The address space available for each single flash device in 24-bit addressing mode is limited to 3 bytes corresponding to 16 MB address space. In 32-bit addressing mode this is extended to 4bytes, allowing the maximum of 128 MB to be addressed. It is within the responsibility of the user not to exceed beyond the physically available addresses of the serial flash attached to the QuadSPI module.

Parallel Flash Mode is valid only for commands related to data read from the serial flash. Any IP Command other than data read in Parallel Flash Mode will result in the assertion of the QSPI_SFMFR[IUEF] flag and any AHB Command other than data read in Parallel Flash Mode will result in the assertion of the QSPI_SFMFR[ABCEF] flag, see Table 35-45 and Table 35-57 for the related commands.

In the Individual Flash Modes the 3 address bytes available for the flash address are determined by SFADR[8:31] as given in the table above.

In Parallel Flash Mode both flashes are read with the same starting address of 3 bytes in size. This address is derived from SFADR[7:30] as given in the table above. The LSB of the SFADR field is used to select the appropriate bits of both flash devices to combine the byte corresponding to the selected address.

### 35.3.3 AMBA Bus Register Memory Map

**Table 35-5. QuadSPI AMBA Bus Memory Map**

| Address | Register Name |
|---|---|
| Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A | |
| 0x7000_0000 ... 0x77FF_FFFF (128 MB) | Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A<br>Refer to Section 35.3.5.2, “Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A“” for details and to Table 35-29 and Table 35-33 for information about the byte ordering. |
| Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B | |
| 0x7800_0000 ... 0x7FFF_FFFF (128 MB) | Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B<br>Refer to Section 35.3.5.3, “Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B“” for details and to Table 35-29 and Table 35-33 for information about the byte ordering. |
| Memory Mapped Serial Flash Data - Parallel Flash Mode | |
| 0x8000_0000 ... 0X8FFF_FFFF (256 MB) | Memory Mapped Serial Flash Data - Parallel Flash Mode<br>Refer to Section 35.3.5.4, “Memory Mapped Serial Flash Data - Parallel Flash Mode“” for details and to Table 35-32 and Table 35-33 for information about the byte ordering. |
| AHB RX data buffer (QSPI_ARDB0 to QSPI_ARDB31) | |
| 0x9000_0000 ...(128*4 Byte) 0x9000_01FF | AHB RX data buffer (QSPI_ARDB0 to QSPI_ARDB31)<br>Address 0x9000_0000 is QSPI_ARDB0. Refer to Table 35-29 and Table 35-31 for information about the byte ordering. |

## 35.3.4 Register descriptions

### 35.3.4.1 Register write access

This section describes the write access restriction terms that apply to all registers.

#### 35.3.4.1.1 Register write access restriction

For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in Table 35-6. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any notification. The values of the bits or fields are not changed.

The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled.

**Table 35-6. Register Write Access Restrictions**

| Condition | Description |
|---|---|
| Anytime | No write access restriction. |
| Disabled Mode | Write access only if the module is in *Module Disable Mode*. |
| Normal Mode | Write access only if the module is in *Normal Mode*. |

#### 35.3.4.1.2 Register Write Access Requirements

All registers can be accessed with 8-bit, 16-bit and 32-bit wide operations. For some of the registers, at least a 16/32-bit wide write access is required to ensure correct operation. This write access requirement is stated in the detailed register description for each register affected

### 35.3.4.2 Module Configuration Register (QSPI_MCR)

The QSPI_MCR holds configuration data associated with QuadSPI operation.

Address: QSPI_BASE + 0x000

Write: Bits 0-15 Disabled Mode
All other fields: Anytime

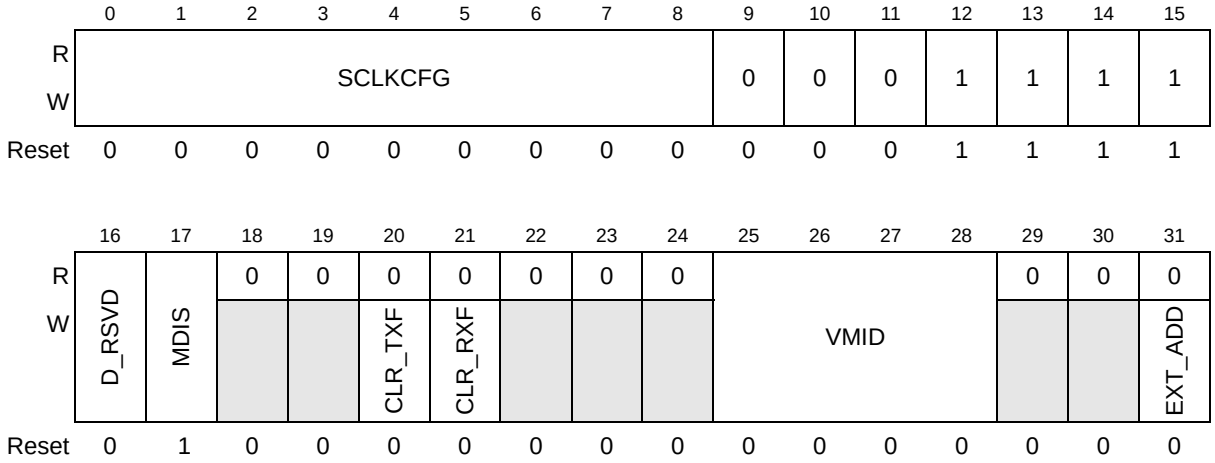



**Figure 35-3. Module Configuration Register (QSPI_MCR)**

**CAUTION**

The upper 16 bits of this register are writable in Disabled mode. Be sure that these are always kept at the reset value indicated in Figure 35-3.

**Table 35-7. QSPI_MCR Field Descriptions**

| Field | Description |
|---|---|
| SCLKCFG | Serial Clock Configuration. This field configures a divider external to the QuadSPI module which divides the clock provided by the SoC before it is fed into the serial flash clock domain.<br>0x00 Divide by 1<br>0x02 Divide by 2<br>0x04 Divide by 3<br>....<br>0x0E Divide by 8<br>All others: Reserved |
| D_RSVD | Reserved bit.<br>This bit is writable but should be kept as value 0. |
| MDIS | Module Disable. The MDIS bit allows the clock to the non-memory mapped logic in the QuadSPI to be stopped, putting the QuadSPI in a software controlled power-saving state. See Section 35.5.4.2, "Module Disable Mode" for more information.<br>0 Enable QuadSPI clocks.<br>1 Allow external logic to disable QuadSPI clocks. |
| CLR_TXF | Clear TX FIFO/Buffer. Invalidate the TX Buffer content.<br>0 No action<br>1 Read and write pointers of the RX Buffer are reset to 0. QSPI_TBSR[TRCTR] is reset to 0. |
| CLR_RXF | Clear RX FIFO. Invalidate the RX Buffer.<br>0 No action<br>1 Read and write pointers of the RX Buffer are reset to 0. QSPI_RBSR[RDBFL] is reset to 0. |

**Table 35-7. QSPI_MCR Field Descriptions (continued)**

| Field | Description |
|---|---|
| VMID | VMID — Vendor Model ID. This field applies to Flash A and Flash B.<br>0000 Reserved<br>0001 Winbond<br>0010 Spansion<br>0011 Macronix<br>0100 Numonyx<br>others Reserved |
| EXT_ADD[1] | Extended memory address modes.<br>0 Default mode, 24-bit addressing.<br>1 32-bit addressing. |

[1] This bit is set to enable the QSPI for 32-bit addressing mode. The respective external serial flash memory should also be independently enabled for accepting 32-bit addresses.

### 35.3.4.3 Latency Configuration Register (QSPI_LCR)

The Latency Configuration Register is used to insert a variable number of dummy cycles during command execution. The dummy cycle insertion is dependent on memory vendor (refer to vendor's memory specification guide), frequency of serial flash clock (SCK) and type of command (refer to QSPI latency support, Table 35-9).

Address: QSPI_BASE + 0x004 Write: Anytime

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | NDC | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

**Figure 35-4. Latency Configuration Register (QSPI_LCR)**

**Table 35-8. QSPI_LCR Field Descriptions**

| Field | Description |
|---|---|
| NDC[1] | Number of dummy cycles to be inserted during command execution.<br>0x00 No dummy cycle<br>0x01 1 dummy cycle<br>0x02 2 dummy cycles<br>0x03 3 dummy cycles<br>0x04 4 dummy cycles<br>....<br>0x3E 62 dummy cycles<br>0x3F 63 dummy cycles |

1 Refer to Data sheet of Memory vendors.

**Table 35-9. QSPI latency support**

| Memory Vendor | Supported Commands |
|---|---|
| Winbond | None |
| Spansion | 0B, 3B, 6B, BB, EB |
| Macronix | All |
| Numonyx | All |

### 35.3.4.4 Serial Flash Address Register (QSPI_SFAR)

The Serial Flash Address Register contains the address for the next IP command. Bits 23 to 0 (bits 26 to 0 when MCR.EXT_ADD = 1) are used for the addressing of the flash device itself. Additional bits are used to specify the access mode of the next IP command. Refer to Table 35-4 for the mapping between the access mode and the QSPI_SFAR content and to Section 35.5.3, “Normal Mode“” for details about the command triggering and command execution.

Address: QSPI_BASE + 0x100 Write: QSPI_SFMSR[IP_ACC] = 0

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SFADR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SFADR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 35-5. Serial Flash Address Register (QSPI_SFAR)**

**Table 35-10. QSPI_SFAR Field Descriptions**

| Field | Description |
|---|---|
| SFADR | Serial Flash Address, register content is used as byte address for all following IP Commands. |

Refer to Table 35-11 below for the address assignment related to the different access modes.

**Table 35-11. SFADR Address Assignment**

| SFADR | Serial Flash Byte Address - related to the first buffer entry[1] | Serial Flash Device |
|---|---|---|
| 0x7000_0000 | 0x00_0000 - 0x00_0003 | Flash A<br>Refer to Section 35.5.3.4.1, “Byte Ordering in Individual Flash Mode“” |
| ... | ... | |
| 0x77FF_FFFC | 0x7FFF_FFFC - 0x7FFF_FFFF | |
| 0x7800_0000 | 0x0000_0000 - 0x0000_0003 | Flash B<br>Refer to Section 35.5.3.4.1, “Byte Ordering in Individual Flash Mode“” |
| ... | ... | |
| 0x7FFF_FFFC | 0x7FFF_FFFC - 0x7FFF_FFFF | |
| 0x8000_0000 | 0x0000_0000 - 0x00_0001 | Flash B<br>Refer to Section 35.5.3.4.2, “Byte Ordering in Parallel Flash Mode“” |
| ... | ... | |
| 0x8FFF_FFFF | 0x8FFF_FFFE - 0x8FFF_FFFF | |

1 The address specified in the SFADR field determines the byte appearing at QSPI_RBDR0[0:7]. Subsequent bytes appear according to the byte ordering given above.

### 35.3.4.5 Instruction Code Register (QSPI_ICR)

The Instruction Code Register consists of the generic instruction code (IC) and an additional parameter section (ICO). This contains additional options to parameterize the command as shown in Table 35-43.

If the IC field is written successfully a new command to the external serial flash device is started with that instruction code if this code is supported by the module (see Section 35.8, “Serial Flash Devices“”).

Refer to Section 35.5.3.1, “Issuing SFM Commands“” for further details about the triggering of IP Commands.

Address: QSPI_BASE + 0x104

Write: ICO: QSPI_SFMSR[IP_ACC] = 0
IC: QSPI_SFMSR[IP_ACC] = 0
and QSPI_SFMSR[AHB_ACC] = 0

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ICO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ICO | | | | | | | | IC | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 35-6. Instruction Code Register (QSPI_ICR)**

**Table 35-12. QSPI_ICR Field Descriptions**

| Field | Description |
|---|---|
| ICO | Instruction Code Options, additional parameters for the IC instruction described below. Meaning of the individual bits vary for each instruction code and vendor, detailed description in Table 35-43. |
| IC | Write access: Instruction Code of the SFM command to be executed next.<br>Read access: Instruction Code of the last SFM command successfully written.<br>Upon writing this byte a new command sequence is started to the external serial flash device. |

### 35.3.4.6 Sampling Register (QSPI_SMPR)

The Sampling Register allows configuration of the scheme how the incoming data from an external serial flash device are sampled in the QuadSPI module.

Address: QSPI_BASE + 0x108

Write: Disabled Mode

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FSDLY | FSPHS | 0 | 0 | HSDLY | HSPHS | HSENA |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 35-7. Sampling Register (QSPI_SMPR)**

**Table 35-13. QSPI_SMPR Field Descriptions**

| Field | Description |
|---|---|
| FSDLY | Full Speed Delay selection. Select the delay w.r.t. the reference edge for the sample point valid for full speed commands:<br>0: One clock cycle delay<br>1: Two clock cycles delay |
| FSPHS | Full Speed Phase selection. Select the edge of the sampling clock valid for full speed commands:<br>0: Select sampling at non-inverted clock<br>1: Select sampling at inverted clock |
| HSDLY | Half Speed Delay selection. **Only relevant when HSENA bit is set**. Select the delay w.r.t. the reference edge for the sample point valid for half speed commands:<br>0: One clock cycle delay<br>1: Two clock cycles delay |
| HSPHS | Half Speed Phase selection. **Only relevant when HSENA bit is set**. Select the edge of the sampling clock valid for half speed commands:<br>0: Select sampling at non-inverted clock<br>1: Select sampling at inverted clock |
| HSENA | Half Speed serial flash clock Enable:<br>This bit enables the divide by 2 of the clock to the external serial flash device for specific commands. Refer to Section 35.8.6, “Serial Flash Clock Frequency Limitations“” for details.<br>0: Disable divide by 2 of serial flash clock for half speed commands<br>1: Enable divide by 2 of serial flash clock for half speed commands |

### 35.3.4.7 RX Buffer Status Register (QSPI_RBSR)

This register contains information related to the receive data buffer.

Address: QSPI_BASE + 0x10C

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

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

**Figure 35-8. RX Buffer Status Register (QSPI_RBSR)**

**Table 35-14. QSPI_RBSR Field Descriptions**

| Field | Description |
|---|---|
| RDCTR | Read Counter, indicates how many entries of 4 byte have been removed from the RX Buffer.<br>It is incremented by the number (QSPI_RBCT[WMRK] + 1) on RX Buffer POP event. For further details please refer to Section 35.3.5.5, “AHB RX data buffer (QSPI_ARDB0 to QSPI_ARDB31)“” and Section 35.5.3.3.2, “Data Transfer from the QuadSPI Module Internal Buffers“”. |
| RDBFL | RX Buffer Fill Level, indicates how many entries of 4 bytes are still available in the RX Buffer. |

## 35.3.4.8 RX Buffer Control Register (QSPI_RBCT)

This register contains control data related to the receive data buffer.

Address: QSPI_BASE + 0x110 Write: QSPI_SFMSR[IP_ACC] = 0

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

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

**Figure 35-9. RX Buffer Control Register (QSPI_RBCT)**

**Table 35-15. QSPI_RBCT Field Descriptions**

| Field | Description |
|---|---|
| RXBRD | RX Buffer Readout: This bit specifies the access scheme for the RX Buffer readout.<br>0 RX Buffer content is read using the AHB Bus registers QSPI_ARDB0 to QSPI_ARDB31.<br>For details refer to Section 35.6.3, “Exclusive Access to Serial Flash for AHB Commands“”.<br>1 RX Buffer content is read using the IP Bus registers QSPI_RBDR0 to QSPI_RBDR31. |
| WMRK | RX Buffer Watermark: This field determines when the readout action of the RX Buffer is triggered. When the number of valid entries in the RX Buffer exceeds the number given by the WMRK field the QSPI_SFMFR[RXWE] flag is asserted.<br>For details refer to Section 35.6.7, “DMA Usage“”. |

## 35.3.4.9 TX Buffer Status Register (QSPI_TBSR)

This register contains information related to the transmit data buffer.

Address: QSPI_BASE + 0x150

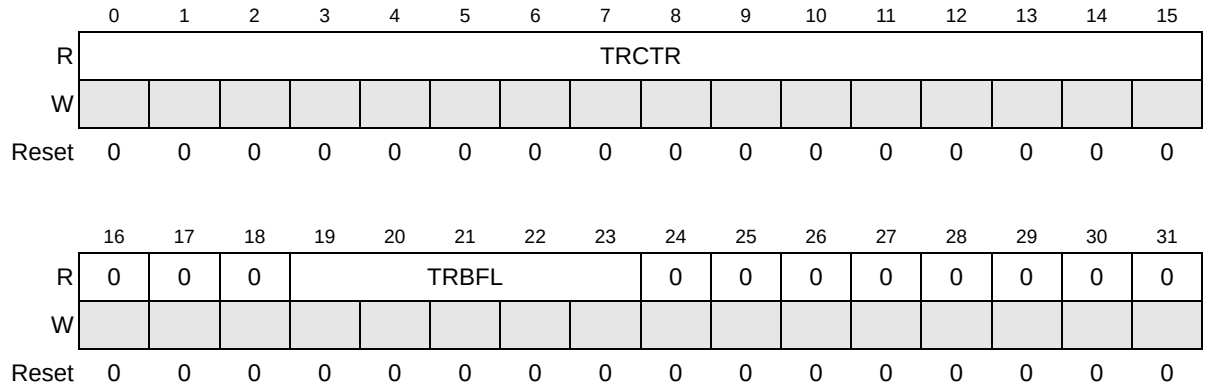



**Figure 35-10. TX Buffer Status Register (QSPI_TBSR)**

**Table 35-16. QSPI_TBSR Field Descriptions**

| Field | Description |
|---|---|
| TRCTR | Transmit Counter. This field indicates how many entries of 4 bytes have been written into the TX Buffer by host accesses. It is reset to 0 when a 1 is written into the QSPI_MCR[CLR_TXF] bit. It is incremented on each write access to the QSPI_TBDR register when another word has been pushed onto the TX Buffer. When it is not cleared the TRCTR field wraps around to 0. Refer to Section 35.3.4.10, "TX Buffer Data Register (QSPI_TBDR)"" for details. |
| TRBFL | TX Buffer Fill Level. The TRBFL field contains the number of entries of 4 bytes each available in the TX Buffer for the QuadSPI module to transmit to the serial flash device. |

## 35.3.4.10 TX Buffer Data Register (QSPI_TBDR)

The QSPI_TBDR register provides access to the circular TX Buffer. This buffer provides the data written into it as write data for the page programming commands to the serial flash device. Refer to Table 35-29 for the byte ordering scheme.

Address: QSPI_BASE + 0x154

Write: QSPI_SFMSR[TXFULL] = 0
32-bit write access required

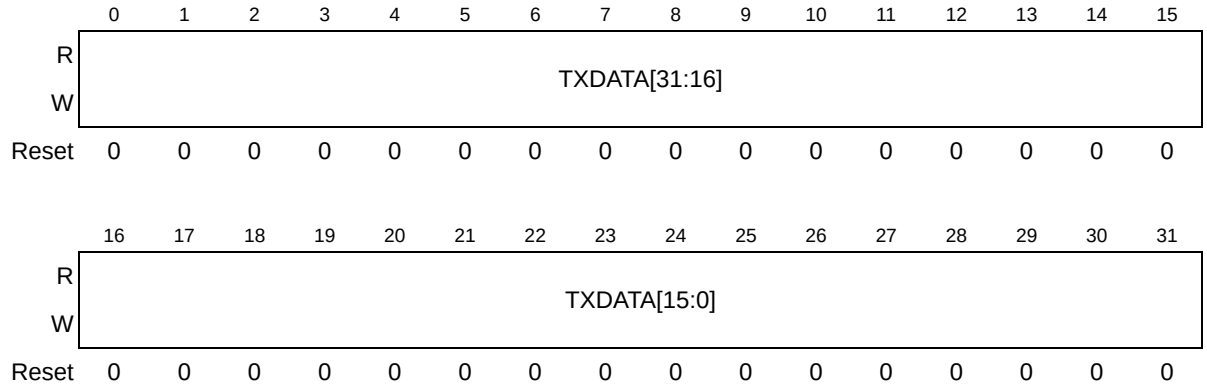



**Figure 35-11. TX Buffer (QSPI_TBDR)**

**Table 35-17. QSPI_TBDR Field Descriptions**

| Field | Description |
|---|---|
| TXDATA | TX Data<br>On write access the data are written into the next available entry of the TX Buffer and the QPSI_TBSR[TRBFL] field is updated accordingly.<br>On read access the last data written are read. |

### 35.3.4.11 AMBA Control Register (QSPI_ACR)

The QSPI_ACR register defines the command that is used for following AHB commands. The read commands allowed are given in 35.8, “Serial Flash Devices”. The execution of the command itself is triggered by an AHB read to the address range assigned to the memory mapped serial flash data. For further details refer to Section 35.6.4, “Command arbitration“”.

Address: QSPI_BASE + 0x158 Write: Anytime

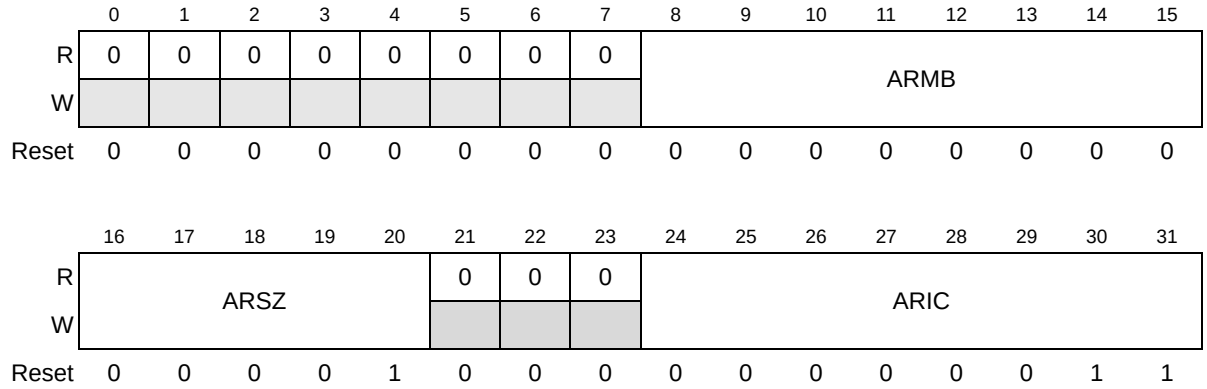



**Figure 35-12. AMBA Control Register (QSPI_ACR)**

**Table 35-18. QSPI_ACR Field Descriptions**

| Field | Description |
|---|---|
| ARMB | AMBA Read Mode Byte. Instruction code option for ARIC for Continuous Mode. Refer to Table 35-43, M7-M0 and Section 35.6.6, “Continuous Mode Commands“”. |
| ARSZ | AMBA Read Size. Specifies the number of bytes which are read from the external serial flash device into the AHB Buffer for any starting address not found in the AHB Buffer. The total number of bytes to be read by the resulting AHB Command is the value in ARSZ * 8.<br>Individual Flash Mode: Legal values for ARSZ are in the range from 1 to 16, resulting in 8 up to 128 bytes read from the (single) external serial flash devices.<br>Parallel Flash Mode: Legal values for ARSZ are in the range from 1 to 16, resulting in 8 up to 128 bytes read from both external serial flash devices. Note that the actual number of bytes read from each flash device is half the total number of bytes. An attempt to write 0 is ignored, instead the reset value is programmed into this field. |
| ARIC | AMBA Read Instruction Code. Selects the read command to be used for any read access to the external serial flash device.<br>The reset value of the ARIC field is the READ_DATA command with a size of 4 bytes. |

### 35.3.4.12 Status Register (QSPI_SFMSR)

The QSPI_SFMSR register provides all available status information about SFM command execution and arbitration, the RX Buffer and TX Buffer and the AHB Buffer.

Address: QSPI_BASE + 0x15C

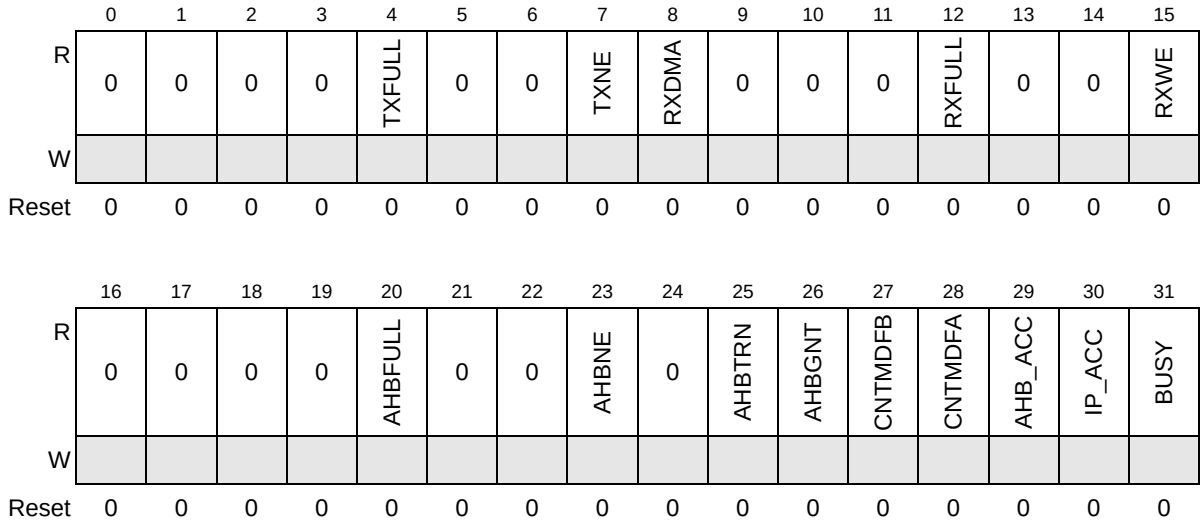



**Figure 35-13. Status Register (QSPI_SFMSR)**

**Table 35-19. QSPI_SFMSR Field Descriptions**

| Field | Description |
|---|---|
| **TX Buffer Related Status Information** | |
| TXFULL | TX Buffer Full: Asserted when no more data can be stored. |
| TXNE | TX Buffer Not Empty: Asserted when TX Buffer contains data. |
| **RX Buffer Related Status Information** | |
| RXDMA | RX Buffer DMA: Asserted when RX Buffer read out via DMA is active. |
| RXFULL | RX Buffer Full: Asserted when the RX Buffer is full, i.e. that QSPI_RBSR[RDBFL] field is equal to 32. |
| RXWE | RX Buffer Watermark Exceeded: Asserted when the number of valid entries in the RX Buffer exceeds the number given in the QSPI_RBCT[WMRK] field. |
| **AHB Buffer and AHB Access Related Status Information** | |
| AHBFULL | AHB Buffer Full: Asserted when AHB buffer is full. |
| AHBNE | AHB Buffer Not Empty: Asserted when AHB buffer contains data. |
| AHBTRN | AHB Access Transaction pending: Asserted when there is a pending request on the AHB interface. Refer to the AMBA specification for details. |
| AHBGNT | AHB Command priority Granted: Asserted when another module has been granted priority of AHB Commands against IP Commands. For details refer to Section 35.6.4, “Command arbitration“”. |
| **SFM Command Related Status Information** | |

**Table 35-19. QSPI_SFMSR Field Descriptions *(continued)***

| Field | Description |
|---|---|
| CNTMDFB | Continuos Mode Bit Flash B: This bit is updated when a SFM Command is triggered.<br>0 The SFM Command triggered does not belong to the subset of Continuous Mode commands.<br>1 The SFM Command triggered does belong to the subset of Continuous Mode commands.<br>Refer to Section 35.6.6, “Continuous Mode Commands“” for details. |
| CNTMDFA | Continuos Mode Bit Flash A: This bit is updated when a SFM Command is triggered.<br>0 The SFM Command triggered does not belong to the subset of Continuous Mode commands.<br>1 The SFM Command triggered does belong to the subset of Continuous Mode commands.<br>Refer to Section 35.6.6, “Continuous Mode Commands“” for details. |
| AHB_ACC | AHB Access: Asserted when the transaction currently executed was initiated by AHB bus. |
| IP_ACC | IP Access: Asserted when transaction currently executed was initiated by IP bus. |
| BUSY | Module Busy: Asserted when module is currently busy handling a transaction to an external flash device. |

### 35.3.4.13 Flag Register (QSPI_SFMFR)

The QSPI_SFMFR register provides all available flags about SFM command execution and arbitration which may serve as source for the generation of interrupt service requests. Note that the error flags in this register do not relate directly to the execution of the transaction in the serial flash device itself but only to the behavior and conditions visible in the QuadSPI module.

Address: QSPI_BASE + 0x160 Write: Enabled Mode

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | TBFF | TBUF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RBOF | RBDF |
| W | | | | | w1c | w1c | | | | | | | | | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | ABCEF | ABMEF | ABOF | IUEF | ICEF | IMEF | 0 | IPAEF | IPIEF | 0 | IPGEF | 0 | 0 | 0 | TFF |
| W | | w1c | w1c | w1c | w1c | w1c | w1c | | w1c | w1c | | w1c | | | | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 35-14. Flag Register (QSPI_SFMFR)**

**Table 35-20. QSPI_SFMFR Field Descriptions**

| Field | Description |
|---|---|
| **TX Buffer Related Flags** | |
| TBFF | TX Buffer Fill Flag: Set when the TX Buffer is not full. Refer to Section 35.5.3.6, “TX Buffer Operation“” for details. |

**Table 35-20. QSPI_SFMFR Field Descriptions *(continued)***

| Field | Description |
|---|---|
| TBUF | TX Buffer Underrun Flag: Set when FSM QSPI_IF tried to pull data although TX Buffer was empty. The IP Command leading to the TX Buffer underrun is continued (data sent to the serial flash device are undefined). The application must clear the TX Buffer in response to this event by writing a a into the QSPI_MCR[CLR_TXF] bit. |
| **RX Buffer Related Flags** | |
| RBOF | RX Buffer Overflow Flag: Set when not all the data read from the serial flash device could be pushed into the RX Buffer.<br>The IP Command leading to this condition is continued until the number of bytes according to the QSPI_IC[ICO] field has been read from the serial flash device.<br>The content of the RX Buffer is not changed. |
| RBDF | RX Buffer Drain Flag: Will be set if the QSPI_SFMSR[RXWE] status bit is asserted.<br>Writing 1 into this bit triggers one of the following actions:<br>• If the RX Buffer is has up to QSPI_RBCT[WMRK] valid entries the flag is cleared.<br>• If the RX Buffer has more than QSPI_RBCT[WMRK] valid entries and the QSPI_SFMRSER[RBDDE] bit is not set (flag driven mode) a RX Buffer POP event is triggered. The flag remains set if the RX Buffer contains more than QSPI_RBCT[WMRK] valid entries after the RX Buffer POP event is finished. The flag is cleared if the RX Buffer contains less than or equal to QSPI_RBCT[WMRK] valid entries after the RX Buffer POP event is finished.<br>Refer to Section 35.5.3.5.2, “Receive Buffer Drain Interrupt or DMA Request“” for details. |
| **AHB Command and AHB Buffer Related Flags** | |
| ABCEF | AHB Command Error Flag: Set when the execution of an AHB Command is started with an invalid AHB Command programmed into the QSPI_ACR[ARIC] field[1].<br>No communication with the serial flash device is initiated by the QuadSPI module.<br>The AHB bus request which triggered this command is answered with an ERROR response. |
| ABMEF | AHB Command Mode Error Flag: Set when the execution of a valid AHB Command is started and one of the following condition occurs:<br>• Mode Bit Collision[2] is detected<br>• Mode Bit Error[3] is detected.<br>No communication with the serial flash device is initiated by the QuadSPI module.<br>The AHB bus request which triggered this command is answered with an ERROR response. |
| ABOF | AHB Buffer Overflow Flag: Set when the module attempted to push data onto the AHB Buffer that exceeded the size of the AHB Buffer. This condition can occur only if the QSPI_ACR[ARSZ] field is programmed incorrectly.<br>The AHB Command leading to this condition is continued until the number of entries according to the QSPI_ACR[ARSZ] field has been read from the serial flash device.<br>The content of the AHB Buffer is not changed. |
| **IP Command Related Flags** | |
| IUEF | IP Command Usage Error Flag: Set when in Parallel Flash Mode the execution of an IP Command is started and the QSPI_ICR[IC] field does not contain a data read command. Refer to Table 35-45 and Table 35-57 for the related commands.<br>No communication with the serial flash device is initiated by the QuadSPI module. |
| ICEF | IP Command Error Flag: Set when the execution of an IP Command is started and the QSPI_ICR[IC] field contains an invalid command.<br>No communication with the serial flash device is initiated by the QuadSPI module. |

**Table 35-20. QSPI_SFMFR Field Descriptions *(continued)***

| Field | Description |
|---|---|
| IMEF | IP Command Mode Error Flag: Set when the execution of a valid IP Command is started and one of the following condition occurs:<br>• Mode Bit Collision[2] is detected<br>• Mode Bit Error[3] is detected.<br>No communication with the serial flash device is initiated by the QuadSPI module. |
| **Command Arbitration and Execution Related Flags** | |
| IPAEF | IP Command Trigger during AHB Access Error Flag. Set when the following condition occurs:<br>• A write access occurs to the QSPI_ICR[IC] field and the QSPI_SFMFR[AHB_ACC] bit is set. Any command leading to the assertion of the IPAEF flag is ignored |
| IPIEF | IP Command Trigger could not be executed Error Flag. Set when the QSPI_SFMSR[IP_ACC] bit is set and any of the following conditions occurs:<br>• Write access to the QSPI_ICR register. Any command leading to the assertion of the IPIEF flag is ignored<br>• Write access to the QSPI_SFAR register.<br>• Write access to the QSPI_RBCT register. |
| IPGEF | IP Command Trigger during AHB Grant Error Flag: Set when the following condition occurs:<br>• A write access occurs to the QSPI_ICR[IC] field and the QSPI_SFMSR[AHBGNT] bit is set. Any command leading to the assertion of the IPGEF flag is ignored |
| TFF | IP Command Transaction Finished Flag: Set when the QuadSPI module has finished a running IP Command. If an error occurred the related error flags are valid at the latest in the same clock cycle when the TFF flag is asserted. |

1 Invalid AHB Command means that the QSPI_ACR[ARIC] field is programmed to a command not reflecting data read. Refer to Table 35-45 and Table 35-57 for the related commands.

2 Refer to Section 35.6.6, “Continuous Mode Commands“” for the description of a Mode Bit Collision.

3 When two serial flash devices are accessed simultaneously in Parallel Flash Mode both of the related Continuous Mode bits QSPI_SFMSR[CNTMDFA] and QSPI_SFMSR[CNTMDFB] must match with the command. If this is not the case a Mode Bit Error is detected.

Additional details about the error flags contained in that register can be found in Table 35-36.

### 35.3.4.14 Interrupt and DMA Request Select and Enable Register (QSPI_SFMRSER)

The QSPI_SFMRSER register provides enables and selectors for the interrupts in the QuadSPI module.

Address: QSPI_BASE + 0x164 Write: Anytime

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | TBFIE | TBUIE | 0 | 0 | 0 | 0 | RBDDE | 0 | 0 | 0 | RBOIE | RBDIE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | ABCEIE | ABMEIE | ABOIE | IUEIE | ICEIE | IMEIE | 0 | IPAEIE | IPIEIE | 0 | IPGEIE | 0 | 0 | 0 | TFIE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 35-15. Interrupt and DMA Request Select and Enable Register (QSPI_SFMRSER)**

**Table 35-21. QSPI_SFMRSER Field Descriptions**

| Field | Description |
|---|---|
| TBFIE | TX Buffer Fill Interrupt Enable |
| TBUIE | TX Buffer Underrun Interrupt Enable |
| RBDDE | RX Buffer Drain DMA Enable: Enables generation of DMA requests for RX Buffer Drain. When this bit is set DMA requests via the ipd_req_rfdf line are generated as long as the QSPI_SFMSR[RXWE] status bit is set.<br>0 No DMA request will be generated<br>1 DMA request will be generated |
| RBOIE | RX Buffer Overflow Interrupt Enable |
| RBDIE | RX Buffer Drain Interrupt Enable: Enables generation of IRQ requests for RX Buffer Drain. When this bit is set the ipi_int_rfdf line is asserted as long as the QSPI_SFMSR[RBDF] flag is set.<br>0 No RBDF interrupt will be generated<br>1 RBDF Interrupt will be generated |
| ABCEIE | AHB Command Error Interrupt Enable |
| ABMEIE | AHB Mode Error Interrupt Enable |
| ABOIE | AHB Buffer Overflow Interrupt Enable |
| IUEIE | IP Command Usage Error Interrupt Enable |
| ICEIE | IP Command Error Interrupt Enable |
| IMEIE | IP Command Mode Error Interrupt Enable |
| IPAEIE | IP Command Trigger during AHB Access Error Interrupt Enable |
| IPIEIE | IP Command Trigger during IP Access Error Interrupt Enable |
| IPGEIE | IP Command Trigger during AHB Grant Error Interrupt Enable |
| TFIE | Transaction Finished Interrupt Enable |

**NOTE**

Each flag of the QSPI_SFMFR register enabled as source for an interrupt prevents the QuadSPI module from entering Stop Mode or Module Disable Mode when this flag is set. Refer to Section 35.5.4, “Power Saving Features“” for details.

### 35.3.4.15 RX Buffer Data Registers 0–31 (QSPI_RBDR0–QSPI_RBDR31)

The QSPI_RBDR registers provide access to the individual entries in the RX Buffer. Refer to Table 35-29 for the byte ordering scheme.

Address: QSPI_BASE + 0x200 (QSPI_RBDR0)
...
QSPI_BASE + 0x27C (QSPI_RBDR31)

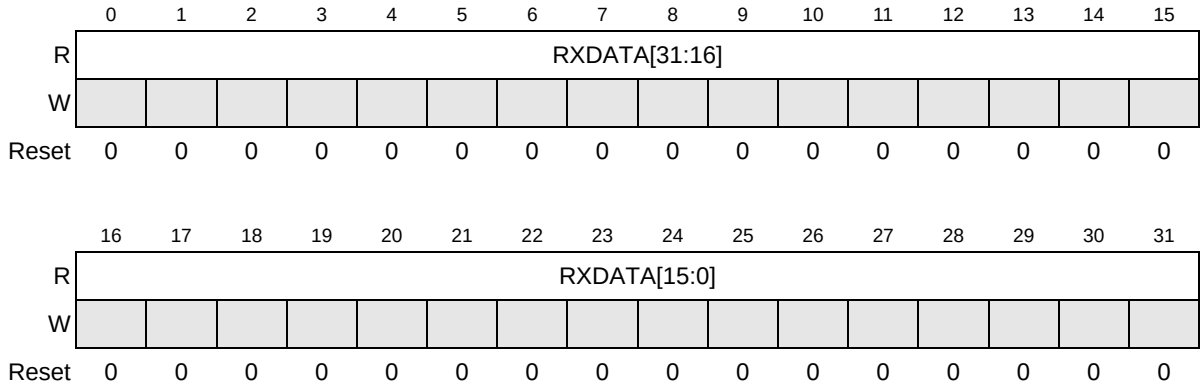



**Figure 35-16. RX Buffer Data Registers 0–31 (QSPI_RBDR0–QSPI_RBDR31)**

**Table 35-22. QSPI_RBDR field descriptions**

| Field | Description |
|---|---|
| RXDATA | RX Data. The RXDATA field contains the data associated with the related RX Buffer entry. Data format and byte ordering is given in Section 35.5.3.4, “Byte Ordering of Serial Flash Read Data“”. |

QSPI_RBDR0 corresponds to the actual position of the read pointer within the RX Buffer. The number of valid entries available depends on the number of valid buffer entries available in the RX Buffer.

Example 1, RX Buffer filled completely with 32 words: In this case the address range for valid read access extends from QSPI_RBDR0 to QSPI_RBDR31.

Example 2, RX Buffer filled with 5 valid words: RX Buffer fill level QSPI_RBSR[RDBFL] is 5. In this case an access to QSPI_RBDR4 provides the last valid entry.

Any access beyond the range of valid RX Buffer entries provides undefined results.

### 35.3.5 AHB Bus Register Memory Map Descriptions

This chapter contains definitions of registers in the AMBA address space.

#### 35.3.5.1 AHB Bus Access Considerations

It has to be noted that all logic in the QuadSPI module implementing the AHB Bus access is related to read the content of an external serial flash device. Therefore the following restrictions apply to the QuadSPI module w.r.t. accesses to the AHB bus:

- Any write access is answered with the ERROR condition according to the AMBA AHB Specification. No write occurs.
- Any AHB Command resulting in the assertion of the QSPI_SFMFR[ABCEF] or the QSPI_SFMFR[ABMEF] flags is answered with the ERROR condition according to the AMBA_AHB specification. The resulting AHB Command is ignored.
- AHB Bus access types fully supported are NONSEQ and BUSY.
- AHB access type SEQ is treated in the same way like NONSEQ. Refer to the AMBA AHB Specification for further details.

#### 35.3.5.2 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A

Starting with address 0x7000_0000 the content of the first external serial flash devices is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 corresponds to bus address 0x7000_0000 with increasing order. Refer to Table 35-23 below for the address mapping. The byte ordering for 32 bit access is given in Table 35-29 and for 64 bit read access the byte ordering is given in Table 35-33.

**Table 35-23. Memory Mapped Individual Flash Mode - Flash A Address Scheme**

| Memory Mapped Address 32 Bit Access | Memory Mapped Address 64 Bit Access | Serial Flash Byte Address |
|---|---|---|
| 0x7000_0000 | 0x7000_0000 | 0x000_0000 - 0x000_0003 |
| 0x7000_0004 | | 0x000_0004 - 0x000_0007 |
| ... | ... | ... |
| 0x77FF_FFF8 | 0x77FF_FFF8 | 0x7FF_FFF8 - 0x7FF_FFFB |
| 0x77FF_FFFC | | 0x7FF_FFFC - 0x7FF_FFFF |

The available address range depends from the size of the external serial flash device. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Section 35.5.3.3, “Flash Read“”.

#### 35.3.5.3 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B

Starting with address 0x7800_0000 the content of the first external serial flash devices is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 corresponds to bus address 0x7800_0000 with increasing order. Refer to Table 35-24 below for the address

mapping. The byte ordering for 32 bit access is given in Table 35-29 and for 64 bit read access the byte ordering is given in Table 35-33.

**Table 35-24. Memory Mapped Individual Flash Mode - Flash B Address Scheme**

| Memory Mapped Address 32 Bit Access | Memory Mapped Address 64 Bit Access | Serial Flash Byte Address |
|---|---|---|
| 0x07800_0000 | 0x7800_0000 | 0x000_0000 - 0x000_0003 |
| 0x07800_0004 | | 0x000_0004 - 0x000_0007 |
| ... | ... | ... |
| 0x07FFF_FFF8 | 0x7FFF_FFF8 | 0x7FF_FFF8 - 0x7FF_FFFB |
| 0x07FFF_FFFC | | 0x7FF_FFFC - 0x7FF_FFFF |

The available address range depends from the size of the external serial flash device. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Section 35.5.3.3, “Flash Read“”.

### 35.3.5.4 Memory Mapped Serial Flash Data - Parallel Flash Mode

Starting with address 0x8000_0000 the content of both (identical) external serial flash devices is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 of both flash devices corresponds to bus address

Read from bus address 0x8000_0000 provides bits [7:4] of both serial flash devices and read from bus address 0x8000_0001 provides bits [3:0] of both flash devices. Refer to Table 35-25 below for the address mapping. The byte ordering for 32 bit access is given in Table 35-31 and for 64 bit read access the byte ordering is given in Table 35-33.

**Table 35-25. Memory Mapped Parallel Flash Mode Address Scheme**

| Memory Mapped Address 32 Bit Access | Memory Mapped Address 64 Bit Access | Serial Flash A Byte Address | Serial Flash B Byte Address |
|---|---|---|---|
| 0x8000_0000 | 0x8000_0000 | 0x000_0000 - 0x000_0001 | 0x000_0000 - 0x000_0001 |
| 0x8000_0004 | | 0x000_0002 - 0x000_0003 | 0x000_0002 - 0x000_0003 |
| 0x8000_0008 | 0x8000_0008 | 0x000_0004 - 0x000_0005 | 0x000_0004 - 0x000_0005 |
| 0x8000_000C | | 0x000_0006 - 0x000_0007 | 0x000_0006 - 0x000_0007 |
| ... | ... | ... | ... |

**Table 35-25. Memory Mapped Parallel Flash Mode Address Scheme (continued)**

| Memory Mapped Address 32 Bit Access | Memory Mapped Address 64 Bit Access | Serial Flash A Byte Address | Serial Flash B Byte Address |
|---|---|---|---|
| 0x8FFF_FFF8 | 0x8FFF_FFF8 | 0x7FF_FFFC - 0x7FF_FFFD | 0x7FF_FFFC - 0x7FF_FFFD |
| 0x8FFF_FFFC | | 0x7FF_FFFE - 0x7FF_FFFF | 0x7FF_FFFE - 0x7FF_FFFF |

The usable space depends from the size of the external serial flash devices. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Section 35.5.3.3, “Flash Read“”.

### 35.3.5.5 AHB RX data buffer (QSPI_ARDB0 to QSPI_ARDB31)

The AHB RX Data Buffer register 0 to 31 can be used to read the buffer content of the RX Buffer from successive addresses. QSPI_ARDB0 corresponds to the RX Buffer register entry corresponding to the current value of the read pointer with increasing order.

The increment of the read pointer depends from the access scheme (DMA or flag-driven). Refer to Section 35.5.3.3.2, “Data Transfer from the QuadSPI Module Internal Buffers“” topic **RX Buffer, data read via register interface** and **AHB read** for the description of successive accesses to the RX Buffer content. Refer also to Section 35.5.3.4, “Byte Ordering of Serial Flash Read Data“” for the byte ordering scheme.

Address:
0x9000_0000 for QSPI_ARDB0
0x9000_007C for QSPI_ARDB31

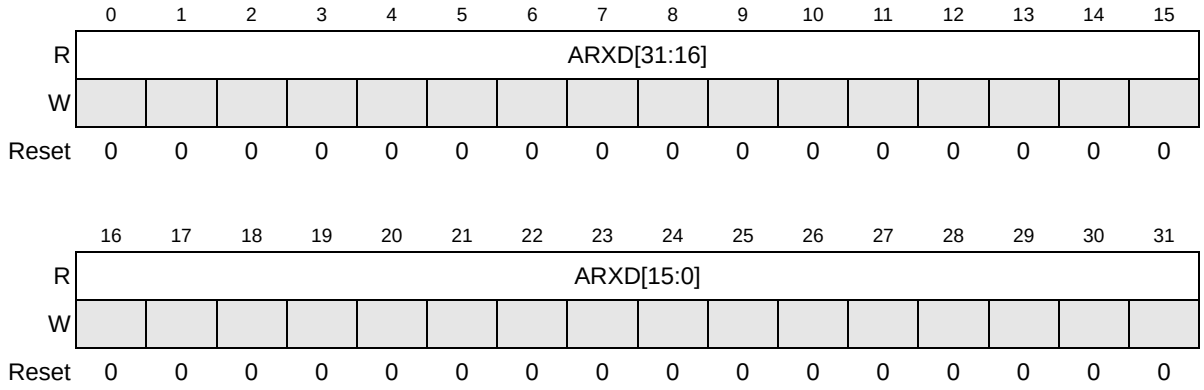



**Figure 35-17. AHB RX data buffer (QSPI_ARDB0 to QSPI_ARDB31)**

**Table 35-26. QSPI_ARDB Field Descriptions**

| Field | Description |
|---|---|
| ARXD | AMBA provided RX Buffer Data.<br>Byte order (endianess) is identical to the RX Buffer Data Registers. |

Valid address range accessible in the QSPI_ARDBn range depends on the number of valid buffer entries available in the RX Buffer.

- Example 1, RX Buffer filled completely with 32 words: In this case the address range for valid read access extends from QSPI_ARDB0 to QSPI_ARDB31.
- Example 2, RX Buffer filled with 5 valid words, RX Buffer fill level QSPI_RBSR[RDBFL] is 5. In this case an access to QSPI_ARDB4 provides the last valid entry.

## 35.4 Interrupt Signals

The interrupt request lines of the QuadSPI module are mapped to the internal flags according to Table 35-27 below:

**Table 35-27. Assignment of interrupt request lines**

| IRQ/DMA line | QSPI_SFMFR flag | Interrupt description |
|---|---|---|
| ipi_int_tfff | TBFF | TX Buffer Fill |
| ipi_int_tcf | TFF | IP Command Transaction Finished |
| ipi_int_rfdf | RBDF | RX Buffer Drain |
| ipi_int_overrun | | Buffer Overflow/Underrun Error<br>Logical OR from: |
| | RBOF | RX Buffer Overrun |
| | TBUF | TX Buffer Underrun |
| | ABOF | AHB Buffer Overflow |
| ipi_int_cerr | | Serial Flash Command Error<br>Logical OR from: |
| | IPAEF | IP access while AHB busy Error |
| | IPIEF | IP Command could not be triggered Error |
| | IPGEF | IP access while AHB Grant Error |
| | IUEF | IP Command Usage Error |
| ipi_int_ored | DLPFF, TBFF, TFF, ILLINE, RBDF, RBOF, TBUF, ABSEF, ABOF, IPAEF, IPIEF, IPGEF, IUEF | Logical OR from all the QSPI_SFMFR flags mentioned |

## 35.5 Functional Description

### 35.5.1 Serial Flash Access Schemes

The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to one single or two external serial flash devices, each with up to 4 bidirectional data lines. Depending from the serial flash devices attached to the QuadSPI module the following access schemes are possible:

**Table 35-28. Access Schemes for Serial Flash Data Access**

| Access Scheme | One Flash Device on Port A | One Flash Device on Port B | Two identical Flash Devices connected on Port A and Port B |
|---|---|---|---|
| Individual Flash Mode: Access to Flash A | Yes | N/a | Yes |
| Individual Flash Mode: Access to Flash B | N/a | Yes | Yes |
| Parallel Flash Mode: Read from Flash A and Flash B | N/a | N/a | Yes |

**NOTE**

If two flash devices are accessed in Parallel Flash Mode they are accessed with identical control signals. Special alignment on per-flash basis is **not** possible. It is within the responsibility of the application to ensure that the identical signals are applicable to both flash devices.

In parallel Flash Mode both external serial flash devices appear logically as one single memory doubled in size w.r.t. one individual flash device.

If two different flash devices are attached they can be operated only in Individual Flash Mode.

In the Parallel Flash Mode only data read commands are supported. Any other IP Command will result in an error condition signalled by the assertion of the QSPI_SFMFR[IUEF] flag and any other AHB Command will result in the assertion of the QSPI_SFMFR[ABCEF] flag.

In the Individual Flash Mode all supported commands are available.

Unless explicitly noted all the following descriptions relate to the Individual Flash Mode.

### 35.5.2 Modes of Operation

Refer to Section 35.1.3, “QuadSPI modes of operation“ for an overview over the possible operational modes of the QuadSPI block.

- Normal Mode can be used for write or read accesses to an external serial flash device.
  - — Serial Flash Write: Data can be programmed into the flash of the serial flash device. Refer to Section 35.5.3.2, “Flash Programming“” for further details.

— Serial Flash Read: Read the contents of the serial flash device. Two separate read channels are available via RX Buffer and AHB Buffer, see Section 35.5.3.3, “Flash Read“”.

- Stop Mode: The mode is used for power management. When a request is made to enter Stop Mode, the QuadSPI block acknowledges the request and completes the SFM Command in progress, then the system clocks to the QuadSPI block may be shut off, see Section 35.5.4.1, “Stop Mode“.
- Module Disable Mode: The mode is used for power management. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in Module Disable Mode.The module enters the mode by setting QSPI_MCR[MDIS] or when a request is asserted by an external controller. See Section 35.5.4.2, “Module Disable Mode“,” for more details.

## 35.5.3 Normal Mode

This mode is used to allow communication with an external serial flash device. Compared to the standard SPI protocol, this communication method uses up to 4 bidirectional data lines operating at high data rates. The communication to the external serial flash device consists of an instruction code and optional address, mode, dummy and data transfers. The flexible programmable core engine described below is immune to a wide variety of command/protocol differences in the serial flash devices provided by various flash vendors.

### 35.5.3.1 Issuing SFM Commands

Each access to the external device follows the same sequence:

1. The user must provide the required components of a SFM command to the QuadSPI module.
2. From these components the complete transaction is built. The transaction starts and the status bit QSPI_SFMSR[BUSY] is set.
3. Communication with the external serial flash device is started and the transaction is executed.
4. When the transaction is finished (all transmit- and receive operations with the external serial flash device are finished) the status bit QSPI_SFMSR[BUSY] is reset. In case of an IP Command the QSPI_SFMFR[TFF] flag is asserted.

Further details are given in below in Section 35.5.3.2, “Flash Programming“” and Section 35.5.3.3, “Flash Read“”.

Note that there are 2 different ways to trigger the processing of SFM Commands in the QuadSPI module.

#### 35.5.3.1.1 IP Commands

For IP Commands the required components need to be written into the following registers:

- Read address of the serial flash into QSPI_SFAR, refer to Section 35.3.4.4, “Serial Flash Address Register (QSPI_SFAR)“. For IP Commands not related to specific addresses the base address of the related flash need to be programmed.
- Instruction code options belonging to the IP Command into the QSPI_ICR[ICO] field.
- Instruction code belonging to the IP Command into the QSPI_ICR[IC] field.

Note that the write into the QSPI_ICR[IC] field must be the last step of the sequence. It is possible to combine both fields of the QSPI_ICR into one single write. Refer to Section 35.3.4.5, “Instruction Code Register (QSPI_ICR)“” for details.

Note that there are some conditions were no IP Command is executed after writing the QSPI_ICR[IC] field and the write operation itself is ignored. They are described in Section 35.6.4, “Command arbitration“”.

#### 35.5.3.1.2 AHB Commands

Note that the required components of the AHB commands are located in different registers w.r.t. the IP Commands. They need to be written into the QSPI_ACR register like described in Section 35.3.4.11, “AMBA Control Register (QSPI_ACR)“”.

The AHB Command itself is triggered by a read access of the host into the memory mapped serial flash data, like described in Section 35.3.5.2, “Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A“”.

Again the possible error conditions are described in Section 35.6.4, “Command arbitration“”.

### 35.5.3.2 Flash Programming

In all cases the memory sector to be written needs to be erased first. The programming sequence itself is then initiated in the following way:

1. Check that the TX Buffer is empty. If the QSPI_SFMSR[TXNE] bit is set the TX Buffer must be cleared by writing 1 into the QSPI_MCR[CLR_TXF] bit.
2. Program the address related to the command in the QSPI_SFAR register. Optionally one can clear the QSPI_TBSR[TRCTR] field by writing 1 into QSPI_MCR[CLR_TXF].
3. Provide initial data for the program command into the circular buffer via register TX Buffer Data Register (QSPI_TBDR). At least one word of data must be written into the TX Buffer.
4. Program the required instruction code options (i.e. size of data) into the QSPI_ICR[ICO] register.
5. Trigger the IP Command to program the serial flash device by writing the instruction code into the QSPI_ICR[IC] register.
6. Depending from the amount of data required step 3 must be repeated until all the required data have been written into the QSPI_TBDR register. At any time the QSPI_TBDR[TRCTR] field can be read to check how many words have been written actually into the TX Buffer.

Steps 4 and 5 may be executed together.

Upon writing the QSPI_ICR[IC] field (refer to step 5) the QuadSPI module will start to execute the command by transferring instruction code, address and then data to the external device. The data are fetched from the TX Buffer. It consists of 15 entries with 32-bit and is organized as a circular FIFO, whose read pointer is incremented after each fetch. When all data are transmitted, the QuadSPI module will return from ‘busy’ to ‘idle’. However, this is not true for the external device since the internal programming is still ongoing. It is up to the user to monitor the relevant status information available from the serial flash device and to ensure that the programming is finished properly.

### 35.5.3.3 Flash Read

Host access to the data stored in the external serial flash device is done in two steps: First the data must be read into the internal buffers and in the second step these internal buffers can be read by the host.

#### 35.5.3.3.1 Reading Serial Flash Data into the QuadSPI Module

Read access to the external serial flash device can be triggered in two different ways:

- **IP Command Read**: For **reading flash data into the RX Buffer** the user must provide the required components of the related SFM command, including the selection of the flash device and the access mode, to the QSPI_SFAR and the QSPI_ICR registers. All available read commands supported by the external serial flash are possible.

  Optionally it is possible to clear the RX Buffer pointer prior to triggering the IP Command by writing a 1 into the QSPI_MCR[CLR_RXF] bit.

  From these inputs the complete transaction is built when the QSPI_ICR[IC] field is written. The transaction related to the read access starts and the requested number of bytes is fetched from the external serial flash device into the RX Buffer. Since the read access is triggered by an IP command the IP_ACC status bit is set driving in turn the BUSY bit (both are located in the QSPI_SFMSR).

  The communication with the external serial flash is stopped when the specified number of bytes has been read (successful completion of the transaction).
- **AHB Command Read**: For **reading flash data into the AHB Buffer** the user must setup a read access to the address range were the external serial flash devices are mapped to by programming the QSPI_ACR register *with the requested data not already available in the AHB Buffer.* Flash device selection and access mode are determined by the address accessed in the AHB address space associated to the QuadSPI module, refer to Section 35.3.5.2, “Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A“”, Section 35.3.5.3, “Memory Mapped Serial Flash Data - Individual Flash Mode on Flash B“” and Section 35.3.5.4, “Memory Mapped Serial Flash Data - Parallel Flash Mode“”.

  On each AHB read access to the memory mapped area the valid data in the AHB Buffer are checked against the address requested in the actual read. When the AHB read request can’t be served from the content of the AHB Buffer the complete transaction to access the external serial flash device is built from the QSPI_ACR register contents and started. The requested number of buffer entries defined in the QSPI_ACR[ARSZ] field is then fetched from the external serial flash device into the internal AHB Buffer. Since the read access is triggered via the AHB bus the QSPI_SFMSR[AHB_ACC] status bit is set driving in turn the QSPI_SFMSR[BUSY] bit until the transaction is finished. The communication with the external serial flash is stopped when the specified number of entries has been filled.

  Basically the AHB buffer behaves similar to a cache memory with a size of one single line.

#### 35.5.3.3.2 Data Transfer from the QuadSPI Module Internal Buffers

The data read out from the external serial flash device by the QuadSPI module are stored in the internal buffers. Depending from the buffer to which the data from the external serial flash has been loaded there are several different ways to access these data in the internal buffers. Refer to Figure 35-1 for details about the two available buffers, the RX Buffer and the AHB Buffer, in the QuadSPI module:

- The **RX Buffer** is implemented as FIFO of depth 32 entries of 4 bytes. Its content is accessible in two different address areas both referring to the identical data.
  - In the IPS address space in the area associated to QSPI_RBDR0 to QSPI_RBDR31.
  - In the AHB address space in the area associated to QSPI_ARDB0 to QSPI_ARDB31. Two successive entries are accessed with one single 64 bit AHB read operation.

  RX Buffer operation can be summarized as follows: The QSPI_RBCT[WMRK] field determines at which fill level the RXWE bit is asserted and how many entries are removed from the RX Buffer on each Buffer POP operation. So the QSPI_SFMSR[RXWE] bit indicates that the configured number of data entries is available in the RX Buffer and the QSPI_RBSR[RDBFL] field indicates how many valid entries are available in total. Note that the first entry (QSPI_RBDR0 or QSPI_ARDB0) always corresponds to the first valid entry in the RX Buffer.

  Further details can be found inSection 35.3.4.15, “RX Buffer Data Registers 0–31 (QSPI_RBDR0–QSPI_RBDR31)“” and in Section 35.3.5.5, “AHB RX data buffer (QSPI_ARDB0 to QSPI_ARDB31)“”.
- **Flag-based Data Read of the RX Buffer** is done by polling the QSPI_SFMFR[RXWE] bit. When it is asserted the valid entries can be read either via the IPS address space (QSPI_RBDRn) or the AHB address space (QSPI_ARDBn). A Buffer POP operation must be triggered by the application by writing a 1 into the QSPI_SFMFR[RBDF] bit.
- **DMA controlled Data Read of the RX Buffer** is done by using the DMA capabilities of the QuadSPI and the device containing the QuadSPI module. The application must ensure that the DMA controller of the related device is programmed appropriately like it is described in Section 35.6.7, “DMA Usage“”.

  DMA controlled read out is triggered fully automatically by the assertion of the QSPI_SFMFR[RXWE] bit. The related Buffer POP operation is also handled completely inside the QuadSPI module. Like in the case above accessing the RX Buffer content either on QSPI_RBDRn or QSPI_ARDBn related addresses is equivalent.
- **AHB Buffer data read via memory mapped access**: This kind of access is done by reading one of the addresses assigned to the external serial flash device(s) within the range given in Table 35-5 *under the condition that the data requested are already present in the AHB Buffer or it is currently read from the serial flash device*. If this is not the case a memory mapped read of the AHB Buffer is triggered like described above). As long as the requested data are already available in the AHB Buffer they are provided to the host. The host can read the available data out of the AHB Buffer in any order.

  If the address requested by the current read is the one currently fetched by the QuadSPI module from the serial flash the execution of the current command remains running with the AHB read access stalled. As soon as the data from the requested address have been read by the QuadSPI module the AHB read access is served. So it’s possible to run sequential read from the AHB buffer at arbitrary speed without the need to monitor any information about the availability of the data. Nevertheless this access scheme stalls the AHB bus for the time required to read the data from the serial flash device.

  As long as the host restricts its accesses to the data already in the buffer and the data currently fetched from the serial flash it is possible to run the host read from the AHB Buffer in parallel to the serial flash read into the AHB Buffer.

### 35.5.3.4 Byte Ordering of Serial Flash Read Data

In this paragraph, the byte ordering of the serial flash data is given. The basic scheme is that the **first** byte read out of the serial flash device - which is addressed by the QSPI_SFAR[SFADR] field - corresponds to bit position QSPI_RBDR0[0:7] register for IP Command read. In contrast to that for AHB Command read the bytes are always positioned according to the byte ordering of the AHB bus.

#### 35.5.3.4.1 Byte Ordering in Individual Flash Mode

Table 35-29 below gives the byte ordering scheme how the byte oriented data space of the serial flash device is mapped into one single 32 bit entry of the RX Buffer or the AHB Buffer. The table is valid within the following context:

- Flash A or Flash B in Individual Flash Mode
- All AHB data read commands with access size of 32 bit

**Table 35-29. Byte Ordering in Individual Flash Mode**

| Serial Flash Byte Number | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Buffer Entry Bit Position [31:0] (32 Bit data width) | [7:0] | [15:8] | [23:16] | [31:24] |

**NOTE**

For IP Commands the read size can be given in number of bytes. If this number is not a multiple of 4 the last buffer entry is not completely filled with the missing higher numbered bytes at undefined values.

For AHB Command read starting from an address not aligned to 32 bit boundaries the requested bytes are given at the appropriate positions according to the AMBA AHB specification.

#### 35.5.3.4.2 Byte Ordering in Parallel Flash Mode

In Parallel Flash Mode each byte is combined out of 2 half bytes which are read in parallel from the two serial flash devices. Table 35-30 below shows how the flash content is separated into the half bytes and Table 35-31 shows how the half bytes are assembled to the content of the QSPI_RBDR0 register.

**Table 35-30. Serial Flash Device Half Byte Ordering**

| Serial Flash Device Byte # | Flash A Bit Position | | Flash B Bit Position | |
|---|---|---|---|---|
| | [7:4] | [3:0] | [7:4] | [3:0] |
| 0 | fah0 | fal0 | fbh0 | fbl0 |
| 1 | fah1 | fal1 | fbh1 | fbl1 |
| 2 | fah2 | fal2 | fbh2 | fbl2 |
| 3 | fah3 | fal3 | fbh3 | fbl3 |
| 4 | fah4 | fal4 | fbh4 | fbl4 |
| 5 | fah5 | fal5 | fbh5 | fbl5 |

**Table 35-30. Serial Flash Device Half Byte Ordering**

| Serial Flash Device Byte # | Flash A Bit Position | | Flash B Bit Position | |
|---|---|---|---|---|
| | [7:4] | [3:0] | [7:4] | [3:0] |
| 6 | fah6 | fal6 | fbh6 | fbl6 |
| 7 | fah7 | fal7 | fbh7 | fbl7 |
| 8 | fah8 | fal8 | fbh8 | fbl8 |

The table entry naming reflects the half byte positioning in the serial flash devices:

- <**fa**>h0 means **Flash A**, <fb>h0 means Flash B.
- fa<**h**>0 means half byte in **high position**, fa<l>o means half byte in low position.
- fah<**0**> means **physical byte address 0** in the serial flash device, fal<1> means physical byte address 1 in the serial flash device.

**Table 35-31. Byte Ordering in Parallel Flash Mode - RX Buffer**

| QSPI_SFAR[SFADR] set to 0x000_0000 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **QSPI_RBDR0 QSPI_ARDB0** | fah0 | fbh0 | fal0 | fbl0 | fah1 | fbh1 | fal1 | fbl1 |
| **QSPI_RBDR1 QSPI_ARDB1** | fah2 | fbh2 | fal2 | fbl2 | fah3 | fbh3 | fal3 | fbl3 |
| **QSPI_SFAR[SFADR] set to 0x000_0001** | | | | | | | | |
| **QSPI_RBDR0 QSPI_ARDB0** | fah1 | fbh1 | fal1 | fbl1 | fah2 | fbh2 | fal2 | fbl2 |
| **QSPI_RBDR1 QSPI_ARDB1** | fah3 | fbh3 | fal3 | fbl3 | fah4 | fbh4 | fal4 | fbl4 |

## NOTE

For IP Commands the read size can be given in number of bytes. If this number is not a multiple of 4 the last buffer entry is not completely filled with the missing higher numbered bytes at undefined values.

**Table 35-32. Byte Ordering in Parallel Flash Mode - AHB Buffer**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **AHB Address 0x8000_0000 (32 Bit Access)** | fah0 | fbh0 | fal0 | fbl0 | fah1 | fbh1 | fal1 | fbl1 |
| **AHB Address 0x8000_0004 (32 Bit Access)** | fah2 | fbh2 | fal2 | fbl2 | fah3 | fbh3 | fal3 | fbl3 |

**NOTE**

For AHB Command read starting from an address not aligned to 32 bit boundaries or AHB access size smaller than 32 bit the requested bytes are given at the appropriate positions according to the AMBA AHB specification.

#### 35.5.3.4.3 Buffer Entry Ordering for 64 Bit Read Access

For read access via the AHB interface 64 bit access is possible. Each 64 bit access reads 2 32 bit entries simultaneously. The ordering of these 32 bit entries within the 64 bit word is given in Table 35-33 below:

**Table 35-33. 64 Bit Read Access Buffer Entry Ordering**

| AHB Read Data Bit Position [63:0] | [31:0] | [63:32] |
|---|---|---|
| Buffer Entry # | Even (0, 2, 4, ...) | Odd (1, 3, 5, ...) |

### 35.5.3.5 Normal Mode Interrupt and DMA Requests

**Table 35-34. IF SCLK Input and Output Signals**

| Signal | I/O | Width | Comment |
|---|---|---|---|
| new_command | I | 1 | request for new transaction to external serial flash |
| ctrl_vector | I | 90 | parameters for new transaction |
| cont_mode | I | 1 | indicates if the external serial flash device is in continous mode ‘1’ = yes |
| sfm_samp_params | I | 5 | controls delay and phase when to sample data sent from serial flash |
| cutil_rdy | O | 1 | state machine ready to accept a new command |
| rdata_valid | O | 1 | newly read data word is available |
| rdata | O | 33 | 32 bit rdata + indicator bit for real rdata |
| tx_underrun | O | 1 | transaction had to be terminated prematurely due to insufficient tx data |
| rx_overrun | O | 1 | transaction had to be terminated prematurely due to RX Buffer overflow |
| cont_mode_sent | O | 1 | control byte for continous mode has been sent |
| tr_complete | O | 1 | transaction complete |

The QuadSPI module has 10 different flags that can only generate interrupt requests and one flag that can generate interrupt as well as DMA requests. Table 35-35 lists the eight conditions. Note that the flags mentioned in the table relate to the Flag Register (QSPI_SFMFR).

**Table 35-35. Interrupt and DMA Request Conditions**

| Condition | Flag (QSPI_SPISR) | DMA |
|---|---|---|
| TX Buffer Fill | TBFF | |
| TX Buffer Underrun | TBUF | |

**Table 35-35. Interrupt and DMA Request Conditions**

| Condition | Flag (QSPI_SPISR) | DMA |
|---|---|---|
| RX Buffer Drain | RBDF | X |
| RX Buffer Overflow | RBOF | |
| AHB Buffer Overflow | ABOF | |
| AHB Command Error | ABCEF | |
| AHB Command Mode Error | ABMEF | |
| IP Command Usage Error | IUEF | |
| IP Command Error | ICEF | |
| IP Command Mode Error | IMEF | |
| IP Command Trigger during AHB Access Error | IPAEF | |
| IP Command Trigger could not be executed Error | IPIEF | |
| IP Access during AHB Grant Error | IPGEF | |
| IP Command related Transaction Finished | TFF | |

Each condition has a flag bit in the QSPI_SFMFR and a Request Enable bit in the QSPI_SFMRSER. The RX Buffer Drain Flag (RBDF) has separate enable bits for generating IRQ and DMA requests. Note that not each single flag is represented by an individual IRQ line.

#### 35.5.3.5.1 Transmit Buffer Fill Interrupt Request

The Transmit Buffer Fill IRQ indicates that the TX Buffer can accept new data. It is asserted if the QSPI_SFMFR[TBFF] flag is asserted and if the corresponding enable bit (QSPI_SFMRSER[TBFIE]) is set. Refer to Section 35.5.3.6, “TX Buffer Operation“” for details about the assertion of the QSPI_SFMFR[TBFF] flag.

#### 35.5.3.5.2 Receive Buffer Drain Interrupt or DMA Request

The Receive Buffer Drain IRQ derived from the QSPI_SFMFR[RBDF] flag indicates that the RX Buffer of the QuadSPI module has data available from the serial flash device to be read by the host. It remains set as long as the QSPI_RBSR[RXWE] bit is set. The QSPI_SFMRSER[RBDIE] bit enables the related IRQ.

Aside from the IRQ it is possible to handle RX Buffer drain by DMA. If the QSPI_SFMRSER[RBDDE] bit is set each write of the module into the RX Buffer triggers a DMA request. The application must set the environment appropriately (e.g. the DMA controller) for the DMA transfers.

#### 35.5.3.5.3 Buffer Overflow/Underrun Interrupt Request

The Buffer Overflow/Underrun IRQ is a combination of the following flags (all located in the QSPI_SPIFR register with the related enable bits in the QSPI_SPIRSER register):

- TBUF - TX Buffer Underrun, enabled by TBU_IE
- RBOF - RX Buffer Overflow, enabled by RBO_IE

- ABOF - AHB Buffer Overflow, enabled by ABO_IE

The Transmit Buffer Underrun indicates that an underrun condition in the TX Buffer has occurred. It is generated when the TX Buffer is empty, a transfer to the serial flash is initiated and the QSPI_SPIRSER[TFUF_IE] bit is set.

The Receive Buffer Overflow indicates that an overflow condition in the RX Buffer has occurred. It is generated when the RX Buffer is full, an additional read transfer attempts to write into the RX Buffer and the QSPI_SFMRSER[RBO_IE] bit is set.

The AHB Buffer Overflow indicates that an overflow condition in the AHB Buffer has occurred. It is generated when the AHB Buffer is full, an additional read transfer attempts to write into the AHB Buffer and the QSPI_SFMRSER[ABO_IE] bit is set.

The data from the transfers that generated the individual overflow conditions are ignored.

#### 35.5.3.5.4 Serial Flash Command Error Interrupt Request

The IPAEF, IPIEF, IPGEF, ABCEF, ABMEF, IUEF, ICEF and IMEF flags in the QSPI_SFMSR and the related interrupt enable bits in the QSPI_SFMRSER determine the assertion of the interrupt line.

#### 35.5.3.5.5 Transaction Finished Interrupt Request

The IP Command Transaction Finished IRQ indicates the completion of the current IP Command. It is masked by the QSPI_SFMSR[TF_IE] bit.

### 35.5.3.6 TX Buffer Operation

The TX Buffer provides the data used for page programming. For proper operation it is required to provide at least one entry in the TX Buffer prior to starting the execution of the page programming command. The application must ensure that the required number of data bytes is written into the TX Buffer fast enough as long as the command is executed without a TX Buffer overflow or underrun.

The QuadSPI module sets the QSPI_SFMFR[TBFF] flag initially and subsequently as long as the TX Buffer can accept more data to be written into.

When the QuadSPI module tries to pull data out of an empty TX Buffer the TX Buffer underrun is signalled by the QSPI_SFMFR[TBUF] flag. The current IP Command leading to the underrun condition is continued until the specified number of bytes has been sent to the serial flash device. When the SFM Command has been finished the QSPI_SFMFR[TBFF] flag is asserted.

The TX Buffer overflow isn’t signalled explicitly, but the TX Buffer fill level can be monitored by the QSPI_TBSR[TRBFL] field.

Refer to Section 35.3.4.9, “TX Buffer Status Register (QSPI_TBSR)“” and Section 35.3.4.13, “Flag Register (QSPI_SFMFR)“” for details about the TX Buffer related registers and to Section 35.5.3.5.1, “Transmit Buffer Fill Interrupt Request“” for details about the usage of the associated interrupt.

#### 35.5.3.7 Address Scheme

Earlier serial flash memories supported only 24-bit address space hence restricting the maximum memory size of the serial flash as 16 MB. The new memory specification supports 32-bit addressing in three different ways. The enhanced QuadSPI supports two types of 32-bit addressing mode, in addition to legacy 24-bit address mode.

##### 35.5.3.7.1 Extended Address Mode

In this mode, the legacy 24-bit commands are converted to accept 32-bit address commands. Both memory as well as QuadSPI shouldd be configured independently for 32-bit address mode. By default, the QuadSPI is in 24-bit legacy address mode. Each of the memory vendors have different way of enabling this mode (Refer to the memory specification from memory vendors). For example, the command B7h sent to Macronix flash will enable it for 32-bit address mode.

##### 35.5.3.7.2 Extended Address Register

In this mode, the upper 8-bit of the 32-bit address is provided by the Extended address register in the memory itself. The memory provides a specific register, which is updated according to the address to be accessed. This effectively converts the legacy 24-bit address command into 32-bit address commands. The memories greater in size than 16 MB, consists of banks of 16 MB. The 8-bit written in the extended address register effectively enables a bank. For example, in Spansion memory, when the extended address register is updated with a value of 0x01 with the help of the command 17h, it will open Bank1 of the memory. The consequent 24-bit addresscommands will lead to Bank1. The extended address register needs to be update with respective value for access to other banks. This way effectively converts the legacy 24-bit address command into 32-bit address commands.

### 35.5.4 Power Saving Features

The QuadSPI supports three power-saving strategies:

- Stop Mode
- Module Disable Mode - Clock gating of non-memory mapped logic
- Clock gating of slave bus signals and clock to memory-mapped logic

Like all power saving features the Stop Mode requires logic external to the QuadSPI module for power management and clock gating control. Figure 35-18 shows an example on how the QuadSPI power saving features can be used:

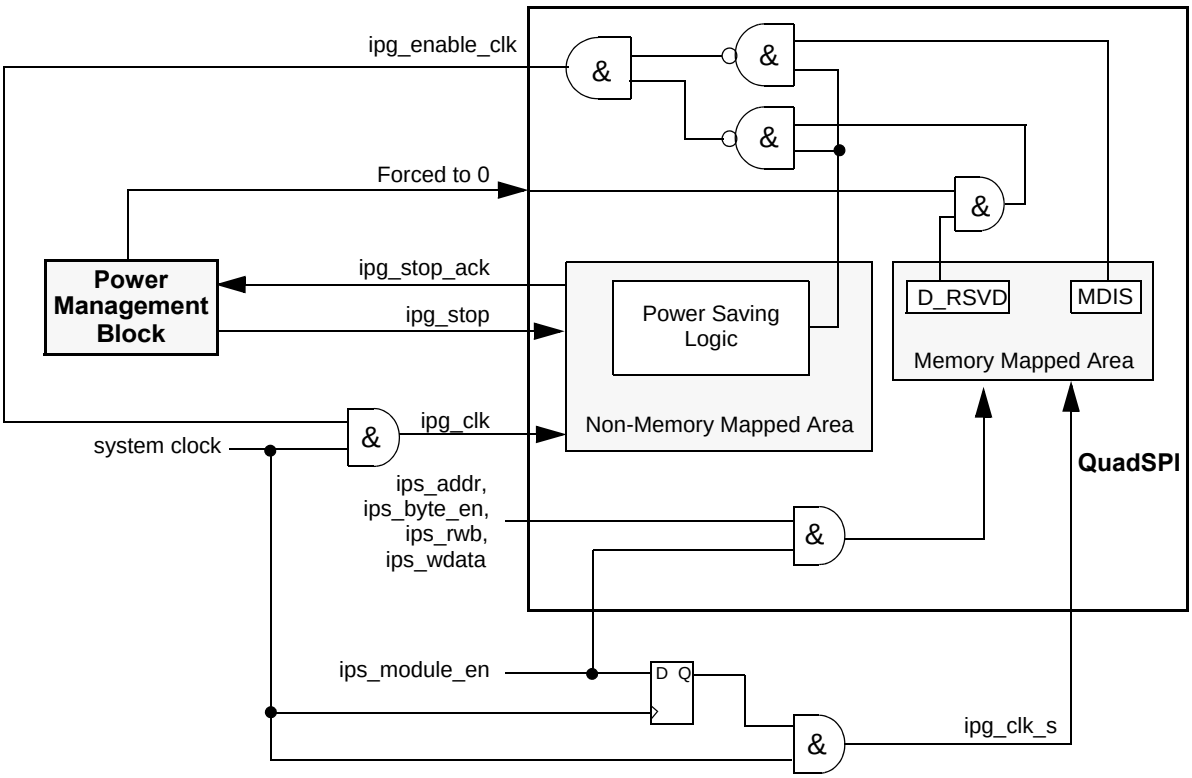



**Figure 35-18. power Saving Mode Concept of QuadSPI module**

### 35.5.4.1 Stop Mode

The QuadSPI has a dedicated low power mode managed by the Mode Entry module. When the Mode Entry module requests the QuadSPI module to enter low power mode, the QuadSPI block waits until the following conditions are met before completing the mode transition:

- QSPI_SFMSR[BUSY] = 0 and
- QSPI_SFMSR[AHBTRN] = 0 and
- QSPI_RBSR[RDBFL] = 0 and
- QSPI_SFMSR[RXDMA] = 0
- None of the flags in the QSPI_SFMFR register enabled as interrupts is set

The conditions given above ensure that there is no SFM Command currently executed, all the data read into the RX Buffer from the serial flash have been fetched by the application. It is also ensured that no current AHB access, no active DMA nor any enabled interrupt is pending

Note that it is not visible to the application whether the module has already negated the ipg_enable_clk.

While the clocks are shut off, the QuadSPI memory-mapped logic is not accessible. Certain read or write operations have a different effect when the QuadSPI is in the Stop Mode. In the Stop Mode not all of the status and flag bits of the QuadSPI module are updated and writing to them will have no effect. Interrupt and DMA request signals cannot be cleared while in the Stop Mode.

Note that there is a time where it is illegal to issue a new SFM Command. This time starts, due to the internal processing pipeline, 2 clock cycles prior to raising the request to go into Stop Mode. This time ends with leaving the Stop Mode.

### 35.5.4.2 Module Disable Mode

Module Disable Mode is a block-specific mode that the QuadSPI can enter to save power. There are two possibilities to request entering the Module Disable Mode:

- Host software can initiate the Module Disable Mode by writing a ‘1’ to the MDIS bit in the QSPI_MCR.
- The Module Disable Mode can also be initiated by hardware. A power management block can initiate Module Disable Mode by asserting the ipg_doze signal while the DOZE bit in the QSPI_MCR is asserted.

When a request is encountered to enter the Module Disable Mode the QuadSPI negates ipg_enable_clk when it is ready to enter the Module Disable Mode.

The condition to enter the Module Disable Mode is reached when:

- QSPI_SFMSR[BUSY] = 0 and
- QSPI_SFMSR[AHBTRN] = 0 and
- QSPI_RBSR[RDBFL] = 0
- QSPI_SFMSR[RXDMA] = 0
- None of the flags in the QSPI_SFMFR register enabled as interrupts is set

The conditions given above ensure that there is no SFM Command currently executed, all the data read into the RX Buffer from the serial flash have been fetched by the application. It is also ensured that no current AHB access, no active DMA nor any enabled interrupt is pending

Note that it is not visible to the application whether the module has already negated the ipg_enable_clk.

If implemented, the ipg_enable_clk signal can stop the clock to the non-memory mapped logic. When ipg_enable_clk is negated and the ipg_clk is stopped the QuadSPI is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the QuadSPI is in the Module Disable Mode. In the Module Disable Mode not all of the status and flag bits of the QuadSPI module are updated and writing to them will have no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable Mode.

Note that there is a time where it is illegal to issue a new SFM Command. This time starts, due to the internal processing pipeline, 2 clock cycles prior to raising the request to enter the Module Disable Mode. This time ends with leaving the Module Disable Mode.

### 35.5.4.3 Leaving Power Saving Modes

In the Stop Mode and the Module Disable Mode the clocks to the QuadSPI module are switched off by external circuitry. Note that after the QuadSPI module has left these power saving modes and has returned to Normal Mode the execution of the first SFM Command is deferred until the clock to drive that part of the module related to the serial flash device is available. Depending from the point in time when the first

SFM Command is triggered the actual execution of that command will start with a slight delay w.r.t. the re-enabling of the clock signal.

#### 35.5.4.4 Slave Bus Signal Gating

The QuadSPI's module enable signal is used to gate slave bus signals such as address, byte enable, read/write and data. This prevents toggling slave bus signals from propagating through parts of the QuadSPI's combinational logic and consuming power unless it is a QuadSPI access. The module enable signal can also be used to gate the clock (ipg_clk_s) to the memory-mapped logic.

## 35.6 Initialization/Application Information

### 35.6.1 Power Up and Reset

Note that the serial flash devices connected to the QuadSPI module may require special voltage characteristics of their inputs during power up or reset. It is within the responsibility of the application to ensure this.

### 35.6.2 Available Status/Flag Information

This paragraph gives an overview over the different status and flag information available and their interdependencies for different use cases. Related registers are QSPI_SFMSR and QSPI_SFMFR. Refer to the related descriptions how to set up the QuadSPI module appropriately.

#### 35.6.2.1 IP Commands

Refer to Section 35.3.4.5, "Instruction Code Register (QSPI_ICR)"" for additional details not explicitly covered in this paragraph.

##### 35.6.2.1.1 IP Commands - Normal Operation

Writing the QSPI_ICR[IC] field triggers the execution of a new IP Command. Given that this is a legal command the QSPI_SFMSR[IPACC] and the QSPI_SFMSR[BUSY] bits are asserted simultaneously immediately after the execution is started.

When the instruction on the serial flash device has been finished these bits are de-asserted and the QSPI_SFMFR[TFF] flag is set.

##### 35.6.2.1.2 IP Commands - Error Situations

Refer to Table 35-36 below.

#### 35.6.2.2 AHB Commands

Refer to Section 35.5.3.3.1, "Reading Serial Flash Data into the QuadSPI Module"" for additional details not explicitly covered in this paragraph.

#### 35.6.2.2.1 AHB Commands - Normal Operation

Memory mapped read access to a serial flash address not covered in the AHB Buffer triggers the execution of an AHB Command. Given that this is a legal command the QSPI_SFMSR[AHBACC] and the QSPI_SFMSR[BUSY] bits are asserted simultaneously immediately after the execution is started.

When the instruction on the serial flash device has been finished these bits are de-asserted.

#### 35.6.2.2.2 IP Commands - Error Situations

Refer to Table 35-36 below.

### 35.6.2.3 Overview of error flags

Table 35-36 below gives an overview of the different error flags in the QSPI_SFMFR register and additional error-related details.

**Table 35-36. Overview of QSPI_SFMFR error flags**

| Error Category | Error Flag in QSPI_SFMFR | Command Execution on Serial Flash Device TFF Behavior | Description |
|---|---|---|---|
| **Command Arbitration Errors** | IPIEF | no -> TFF not asserted in conjunction with that command | Attempt to trigger IP Command not be executed.<br>• IP Command already running, write attempt to QSPI_IC register.<br>• IP Command running, attempt to write QSPI_SFAR register.<br>• IP Command running, attempt to write QSPI_RBCT register. |
| | IPAEF | | • AHB Command already running, another IP Command could not be executed.<br>• AHB Command already running, write attempt to QSPI_ICR[IC] field. |
| | IPGEF | | • Exclusive access to the serial flash granted for AHB Commands, write attempt to QSPI_ICR[IC] field. |
| **AHB Command Errors** | ABCEF | no -> TFF not asserted in conjunction with that command | • AHB Command Error |
| | ABMEF | | • AHB Command Mode Error[1] |
| **IP Command Errors** | IUEF | | • IP Command Usage Error |
| | ICEF | | • IP Command Error |
| | IMEF | | • IP Command Mode Error[1] |
| **Buffer Related Errors** | RBOF | yes -> TFF is asserted on completion | • RX Buffer Overrun |
| | TBUF | | • TX Buffer Underrun |

[1] Refer to Section 35.6.6, “Continuous Mode Commands“” for the description of a Mode Bit Collision

Note that only the buffer related errors are related to a transaction on the external serial flash. All the other errors do not trigger an actual transaction.

#### 35.6.2.4 IP bus and AHB access command collisions

There are two flags related to this topic, the QSPI_SFMFR[IPAEF] and QSPI_SFMFR[IPIEF]. See Section 35.5.3.3.1, “Reading Serial Flash Data into the QuadSPI Module“” for a description of the flags itself and to Section 35.6.4, “Command arbitration“” for details about possible command collisions.

### 35.6.3 Exclusive Access to Serial Flash for AHB Commands

It is possible that several masters need to access the serial flash device connected to the QuadSPI module separately, one master by triggering IP Commands and reading the RX Buffer and the other masters by triggering AHB Commands. To avoid command collisions resulting in excessive latencies the QuadSPI module implements a request-handshake mechanism between the master triggering AHB Commands and the QuadSPI module allowing this specific master to request exclusive access to the serial flash device for AHB Commands. If this exclusive access is granted the execution of IP Commands is blocked. This resolves command collisions and excessive times where the AHB interface may be blocked.

If this capability is used in the device there is additional status and flag information available related to this mechanism. The QSPI_SFMSR[AHBGNT] bit reflects the module-internal state that the exclusive access mentioned above is granted, any attempt to trigger an IP Command is rejected and results in the assertion of the QSPI_SFMFR[IPGEF] flag. Refer to the descriptions of the related bit and flag for details.

It is within the responsibility of the application to set up the master using this mechanism appropriately, if used incorrectly no IP Commands at all can be triggered.

Two different cases can be distinguished:

#### 35.6.3.1 RX Buffer Read via QuadSPI_ARDB Registers

In this case all masters share the AHB bus for RX Buffer as well as for AHB Buffer read. In this case the access to the AHB interface by the master triggering AHB Commands must be deferred until any pending IP Command has been finished **and** the RX Buffer readout has been finished as well. This is the conservative use case, corresponding to the reset value 0 of the QuadSPI_RBCT[RXBRD] bit.

In this case the QuadSPI_SFMSR[AHBGNT] bit is asserted not earlier than any running IP Command has been finished (QuadSPI_SFMSR[IP_ACC] is 0), the RX Buffer has been read out completely (QuadSPI_RBSR[RDBFL] equal to 0) and no DMA read is pending (QuadSPI_SFMSR[RXDMA] equal to 0.

#### 35.6.3.2 RX Buffer Read via QuadSPI_RDBR Registers

This is the preferred use case. It is not possible that a pending AHB bus access triggered by an AHB Command stalls the AHB bus and blocks the RX Buffer readout since the RX Buffer is read via the IP bus based registers QuadSPI_RBDR0 to QuadSPI_RBDR31.

For this case it is recommended to program the QuadSPI_RBCT[RXBRD] bit to 1. The QuadSPI_SFMSR[AHBGNT] bit is asserted immediately after any running IP Command has been finished (QuadSPI_SFMSR[IP_ACC] is 0), allowing the master triggering AHB Commands to trigger AHB Commands as soon as possible without the need to wait for the RX Buffer readout to be finished.

### 35.6.4 Command arbitration

In case of overlapping commands the arbitration scheme is described in the following paragraphs under the assumption that the priority mechanism described in Section 35.6.3, “Exclusive Access to Serial Flash for AHB Commands“” is **not** used:

- During the execution of an IP Command the running IP Command can’t be terminated by issuing another IP Command or AHB Command. The QSPI_SFMFR[IPIEF] flag is asserted when the host tries to write into the QSPI_ICR register. When the host triggers an AHB Command (refer to Section 35.5.3.3.1, “Reading Serial Flash Data into the QuadSPI Module“” for details) this command is stalled until the currently running IP Command is finished.
- During the execution of an AHB Command the running AHB Command can’t be terminated by issuing an IP Command. The command is ignored and the QSPI_SFMFR[IPAEF] flag is asserted. Refer to Section 35.3.4.13, “Flag Register (QSPI_SFMFR)“” for the description of these flags.

  When another AHB Command is triggered the address of the memory mapped access is considered. If the requested address is currently read from the serial flash device the running command is continued. If this is not the case the currently running command is terminated and another AHB Command related to the requested address is executed. Refer to Section 35.5.3.3.1, “Reading Serial Flash Data into the QuadSPI Module“” for further details.

In case of coinciding commands the IP Command is triggered and the AHB Command is stalled until the IP Command has been finished (QSPI_SFMSR[IP_ACC] has been deasserted).

The IP Commands ignored in case of command collision will not result in the assertion of the QSPI_SFMFR[TFF] flag.

### 35.6.5 Flash Device Selection

Regardless of the SFM Command (IP or AHB) the access mode is selected by specifying the 32 bit address value for the following SFM Command.

For IP Commands the access mode is selected with the address programmed into the QSPI_SFAR register. Refer to Section 35.3.4.4, “Serial Flash Address Register (QSPI_SFAR)“” for details.

For AHB Commands the access mode is determined by the memory mapped address which is accessed Refer to Section 35.3.3, “AMBA Bus Register Memory Map“” for details.

### 35.6.6 Continuous Mode Commands

In the command set of the serial flash devices there are commands which use a special access mode of the attached serial flash device. Once this mode is enabled in the serial flash device there are only certain SFM Commands allowed until this mode is explicitly disabled. Refer to the serial flash device specification for further details.

Note that it is an error if a SFM Command triggered does not correspond to the related status bits QSPI_SFMSR[CNTMDFA/B]. In this case a Mode Bit Collision is detected associated with the QSPI_SFMFR[IMEF] or the QSPI_SFMFR[ABMEF] flag, depending from the SFM Command triggered.

## 35.6.7 DMA Usage

For the complete description of the DMA module refer to the related chapter. In this paragraph only the details specific to the DMA usage related to the QuadSPI module are given.

### 35.6.7.1 DMA Usage in Normal Mode

#### 35.6.7.1.1 Bandwidth Considerations

Careful consideration of the throughput rate of the entire chain (serial flash -> AHB bus/IP bus -> DMA controller) involved in the read data process is essential for proper operation. Such analysis must take into account not only the datarate provided by the serial flash but also the datarate of the AHB bus and the performance of the DMA controller in reading data from the RX Buffer.

Two figures must match for proper operation, that means that the data rate provided by the serial flash device must not exceed the average RX Buffer readout data rate. Otherwise, the longer this state persists, a RX Buffer overflow will result.

AHB Bus Side:

The total number of bus cycles for each DMA Minor Loop completion is added from the following components:

- Overhead for each minor loop, given by DMA controller: 10 cycles
- Overhead due to clock domain crossing: 2 cycles
- Number of bus clock cycles required for 8 bytes (64 bit read size): 2 cycles (read/write sequence of DMA controller)

Note that the size of the minor loop is determined by the size of the QSPI_RBCT[WMRK] field, therefore the overhead given above distributes among (QSPI_RBCT[WMRK] + 1)/2 read accesses of 64 bit each.

Table 35-37 below gives some examples for typical use cases:

**Table 35-37. Access Duration Examples - Bus Clock Side**

| QSPI_RBCT[WMRK] Setting | # of Bytes per DMA Loop[1] | # of Bus Clock Cycles for DMA Minor Loop | Time Duration of DMA Minor Loop for 120 MHz Bus Clock Frequency |
|---|---|---|---|
| 0 | 4 | 12 + 2 = 14 | ~117 ns |
| 1 | 8 | 12 + 2 = 14 | ~117 ns |
| 3 | 16 | 12 + 4 = 16 | ~133 ns |

**Table 35-37. Access Duration Examples - Bus Clock Side**

| QSPI_RBCT[WMRK] Setting | # of Bytes per DMA Loop[1] | # of Bus Clock Cycles for DMA Minor Loop | Time Duration of DMA Minor Loop for 120 MHz Bus Clock Frequency |
|---|---|---|---|
| 7 | 32 | 12 + 8 = 20 | ~167 ns |
| 11 | 48 | 12 + 12 = 24 | ~200 ns |

1 'DMA Loop' means one Minor Loop Completion which is equivalent to one Major Loop Iteration

Serial Flash Device Side:

The number of serial flash clock cycles can be determined in the following way:

- Number of serial flash clock cycles required to read 4 bytes, corresponding to one RX Buffer entry (setup of command and address not considered): 4 cycles for Quad Mode Instructions in Parallel Flash Mode, 8 cycles for Quad Mode Instructions in Individual Flash Mode, 16/32 cycles for Dual/Single Instructions.
- Overhead due to clock domain crossing: 1 cycle

Table 35-38 below lists the number of clock cycles required to read the data from serial flash corresponding to the different setting of the QSPI_RBCT[WMRK] field:

**Table 35-38. Access Duration Examples - Serial Flash Clock Side**

| QSPI_RBCT[WMRK] Setting | # of Bytes per DMA Loop[1] | # of SCKFx Cycles for 48 MHz SCKFx | | | | Time Duration of SerialFlash Data Readout for 48 MHz SCKFx Frequency | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | IFM[2] Single | IFM Dual | IFM Quad | PFM[3] Quad | IFM Single | IFM Dual | IFM Quad | PFM Quad |
| 0 | 4 | 33 | 17 | 9 | 5 | ~688 ns | ~354 ns | ~188 ns | ~104 ns |
| 1 | 8 | 65 | 33 | 17 | 9 | ~1.354 µs | ~688 ns | ~354 ns | ~188 ns |
| 3 | 16 | 129 | 65 | 33 | 17 | ~2.688 µs | ~1.354 µs | ~688 ns | ~354 ns |
| 7 | 32 | 257 | 129 | 65 | 33 | ~5.354 µs | ~2.688 µs | ~1.354 µs | ~688 ns |
| 11 | 48 | 385 | 193 | 97 | 49 | ~8.021 µs | ~4.021 µs | ~2.021 µs | ~1.021 µs |

1 'DMA Loop' means one Minor Loop Completion which is equivalent to one Major Loop Iteration

2 Individual Flash Mode

3 Parallel Flash Mode

From the examples given in Table 35-37 and Table 35-38 above it can be seen that, dependent from the relationship between the frequencies of the bus clock and the serial flash clock, there are settings possible where the serial flash provides the read data faster than the AHB bus can read out the RX Buffer. In the tables above it is the case of reading the serial flash in Parallel Flash Mode with quad instructions and QSPI_RBCT[WMRK] set to 0. In this case the RX Buffer is filled over time, to avoid the RX Buffer overrun the IP Command used to read from flash need to specify the data size small enough.

Note also that side effects like load from other masters on the AHB bus are not considered.

## 35.7 Byte Ordering - Endianess

The internal byte orientation of the QuadSPI module is big endian (BE). This means that the high order bits of the associated data vectors are associated with low order address positions.

The byte ordering is according to the following example:

### 35.7.1 Programming Flash Data

CPU write instructions to the QSPI_TBDR register like

(1) Write QSPI_TBDR -> 0x01_02_03_04

(2) Write QSPI_TBDR -> 0x05_06_07_08

result in the following content of the TX Buffer:

**Table 0-2 Example of QuadSPI TX Buffer**

| TX Buffer Entry | Content |
|---|---|
| 0 | 32'h01_02_03_04 |
| 1 | 32'h05_06_07_08 |

Programming the TX Buffer into the external serial flash device results in the following byte order to be sent to the serial flash:

01...02...03...04...05...06...07...08

### 35.7.2 Reading Flash Data into the RX Buffer

Reading the content from the same address provides the following sequence of bytes, identical to the write case:

01...02...03...04...05...06...07...08

This results in the RX Buffer filled with:

**Table 35-39. Resulting RX Buffer Content**

| RX Buffer Entry | Content |
|---|---|
| 0 | 32'h01_02_03_04 |
| 1 | 32'h05_06_07_08 |

#### 35.7.2.1 Readout of the RX Buffer via QSPI_RBDRn

The RX Buffer content appears at CPU read access via the IP SkyBlue interface in the following order:

(1) Read QSPI_RBDR0 <- 0x01_02_03_04

(2) Read QSPI_RBDR1 <- 0x05_06_07_08

#### 35.7.2.2 Readout of the RX Buffer via ARDBn

The RX Buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

(1a): 32 Bit Access: Read QSPI_ARDB0 <- 0x01_02_03_04

(2a): 32 Bit Access: Read QSPI_ARDB1 <- 0x05_06_07_08

(1b/2b): 64 Bit Access: Read QSPI_ARDB0 <- 0x01_02_03_04_05_06_07_08

### 35.7.3 Reading Flash Data into the AHB Buffer

Reading the content from the same address as it was written to provides the following sequence of bytes, identical to the write case:

01...02...03...04...05...06...07...08

This results in the AHB Buffer filled with:

**Table 35-40. Resulting AHB Buffer Content**

| AHB Buffer Entry | Content |
|---|---|
| 0 | 64'h01_02_03_04_05_06_07_08 |

#### 35.7.3.1 Readout of the AHB Buffer via Memory Mapped Read

The AHB Buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

(1a): 32 Bit Read Access: <- 0x01_02_03_04

(2a): 32 Bit Read Access:  <- 0x05_06_07_08

(1/2): 64 Bit Read Access: <- 0x01_02_03_04_05_06_07_08

## 35.8 Serial Flash Devices

Currently flash memory devices with serial flash bus are developed by several vendors. Most standard commands currently have the same instruction code for all vendors, some commands are however unique for one vendor. The currently supported list of instruction codes and the required instruction code options are provided in this section.

### 35.8.1 Supported Instruction Codes in Winbond Devices

**Table 35-41. Winbond Instruction Codes**

| Command | Instruction Code | Required Input Data / Parameters | | | Output Data |
|---|---|---|---|---|---|
| | | Options QSPI_ICR[ICO] | Address [QSPI_SFAR] | Data [TX Buffer] | Status / Data [RX Buffer] |
| Write Enable | 06h | | | | |
| Write Disable | 04h | | | | |
| Read Status Reg1 | 05h | size | | | S7 - S0 |
| Read Status Reg2 | 35h | size | | | S15 - S8 |
| Write Status Reg | 01h | size | | S15 - S0 | |
| Page Program | 02h | size | A23 - A0 | size * (D7 - D0)[1] | |
| Quad Page Program | 32h | size | A23 - A0 | size * (D7 - D0) | |
| 32k Block Erase | 52h | | A23 - A0 | | |
| 64k Block Erase | D8h | | A23 - A0 | | |
| Sector Erase | 20h | | A23 - A0 | | |
| Chip Erase | C7h/60h | | | | |
| Erase Suspend | 75h | | | | |
| Erase Resume | 7Ah | | | | |
| Power down | B9h | | | | |
| High Performance Mode | A3h | | | | |
| Read Data | 03h[2] | size | A23 - A0 | | size * (D7 - D0)[1] |
| Fast Read | 0Bh | size | A23 - A0 | | size * (D7 - D0) |
| Fast Read Dual Output | 3Bh | size | A23 - A0 | | size * (D7 - D0) |
| Fast Read Dual I/O | BBh | M7 - M0, size | A23 - A0 | | size * (D7 - D0) |
| Fast Read Quad Output | 6Bh | size | A23 - A0 | | size * (D7 - D0) |
| Fast Read Quad I/O | EBh | M7 - M0, size | A23 - A0 | | size * (D7 - D0) |
| Octal Word Read Quad I/O | E3h | M7 - M0, size | A23 - A0 | | size * (D7 - D0) |
| (Continuos Read) Mode Bit Reset | FFh | (FFh), size | | | |

**Table 35-41. Winbond Instruction Codes (continued)**

| Command | Instruction Code | Required Input Data / Parameters | | | Output Data |
|---|---|---|---|---|---|
| | | Options QSPI_ICR[ICO] | Address [QSPI_SFAR] | Data [TX Buffer] | Status / Data [RX Buffer] |
| Release Power Down/HighPerf.Mode (optional): read device ID | ABh | (opt.): read size | | | (opt.): ID7 - ID0 |
| Read Manufacturer/Device ID | 90h | size | A23 - A0 = 0h/1h | | ManID7 - ManID0 DevID7 - Dev0 |
| Read Unique ID | 4Bh | size | | | UniqID63 - UniqID0 |
| Read JEDEC ID | 9Fh | size | | | ManID7 - ManID0, MemID7 - Mem0, CapID7 - CapID0 |

[1]Denotes that 'size'-times one byte is transferred on the serial flash data bus. Total number of bytes must be provided in the TX Buffer or can be read from the RX Buffer

[2]According to Winbond documentation this command only supports a maximum clock speed of 50 MHz. If the Serial Flash is operated at a higher clock frequency, the clock frequency for this command must be decreased. Refer to Section 35.8.5, "Serial Flash Clock Frequency Limitations"" for details.

The QSPI_MCR[ICO] field is mapped to the different command options belonging to the complete SFM Command which is then sent to the serial flash. This is shown in detail in the following tables. Table 35-42 shows the commands usable in all flash access modes, Table 35-43 the commands specific to the Individual Flash Modes and Table 35-44 the commands specific to the Parallel Flash Mode. All size information is given in number of bytes. Note that the mapping reflects the bits which are actually sent to the single serial flash device selected in the Individual Flash Mode or which are sent to each of the two serial flash devices in the Parallel Flash Mode.

**Table 35-42. ICO field on Winbond Devices - All Flash Access Modes**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 01h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 02h | Size (bytes to be written) | | | | | | | | | | | | | | | | | | | | | | | |
| 05h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 32h | Size (bytes to be written) | | | | | | | | | | | | | | | | | | | | | | | |
| 35h | | | | | | | | | | | | | | | Size | | | | | | | | | |

**Table 35-42. ICO field on Winbond Devices - All Flash Access Modes (continued)**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 03h | Individual Flash Access: Refer to Table 35-43<br>Parallel Flash Access: Refer to Table 35-44 | | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| BBh | | | | | | | | | | | | | | | | | | | | | | | | |
| 6Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| E3h | | | | | | | | | | | | | | | | | | | | | | | | |
| EBh | | | | | | | | | | | | | | | | | | | | | | | | |
| ABh | | | | | | | | | | | | | | | read[1] | | | | | | | | | |
| 90h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 4Bh | | | | | | | | | | | | | Size (bytes) | | | | | | | | | | | |
| 9Fh | | | | | | | | | | | | | | | Size | | | | | | | | | |
| FFh | | | | | | | | | | | | | | | | Size[2] | | | | | | | | |

[1] 'read' bit controls if 8 SCLK clocks are added to read the device ID, equivalent to Size = 1 (1byte to read)
In any case 24 dummy bits are transmitted after the instruction code to allow reading.

[2] To reset continuous read mode while in Dual I/O operations, set Size to 1. To reset the continuous read mode while in Quad I/O operations set Size to 0.

**Table 35-43. ICO field on Winbond Devices - Individual Flash Modes**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 03h | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| BBh | Size (bytes to be read) | | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |
| 6Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| E3h | Size (bytes to be read) | | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |
| EBh | Size (bytes to be read) | | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |

In the Parallel Flash Mode the QSPI_ICR[ICO] field is no longer related to the size read from each individual flash device. Instead the size to be read is related to the (virtual) combined flash memory. Therefore the amount of data that is read from each single flash device is half the amount of data specified in the QSPI_ICR[ICO] field.

**Table 35-44. ICO field on Winbond Devices - Parallel Flash Mode**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 03h | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| BBh | | Size (bytes to be read) | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |
| 6Bh | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| E3h | | Size (bytes to be read) | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |
| EBh | | Size (bytes to be read) | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |

## 35.8.2 Instruction Codes in Spansion Devices

**Table 35-45. Spansion Instruction Codes**

| Command | Instruction Code | Required Input Data / Parameters | | | Output Data |
|---|---|---|---|---|---|
| | | Options QSPI_ICR[ICO] | Address [QSPI_SFAD] | Data [TX Buffer] | Status / Data [RX Buffer] |
| Write Enable | 06h | | | | |
| Write Disable | 04h | | | | |
| Read Status Reg | 05h | size | | | S7 - S0 [8] |
| Read Config Reg | 35h | size | | | S15 - S8 [8] |
| Write Reg | 01h | size | | S15 - S0 | |
| Reset Erase and prog fail flag | 30h | size | | S15 - S0 | |
| Page Program | 02h | size | A23 - A0 | size * (D7 - D0)[2] | |
| Quad Page Program | 32h | size | A23 - A0 | size * (D7 - D0) | |
| 16kbit = 4KB Sector Erase | 20h | | A23 - A0 | | |
| 32kbit = 8KB Block Erase | 40h | | A23 - A0 | | |
| 64KB Block Erase | D8h | | A23 - A0 | | |
| Bulk Erase | C7h / 60h | | | | |
| Deep Power Down | B9h | | | | |
| Read Data | 03h[3] | size | A23 - A0 | | size * (D7 - D0)[1] |
| Fast Read | 0Bh | size | A23 - A0 | | size * (D7 - D0) |
| Dual Output Read | 3Bh | size | A23 - A0 | | size * (D7 - D0) |

**Table 35-45. Spansion Instruction Codes (continued)**

| Command | Instruction Code | Required Input Data / Parameters | | | Output Data |
|---|---|---|---|---|---|
| | | Options QSPI_ICR[ICO] | Address [QSPI_SFAD] [1] | Data [TX Buffer] | Status / Data [RX Buffer] |
| Quad Output Read | 6Bh | size | A23 - A0 | | size * (D7 - D0) |
| Dual I/O High Performance Read | BBh | M7 - M0, size | A23 - A0 | | size * (D7 - D0) |
| Quad I/O High Performance Read | EBh | M7 - M0, size | A23 - A0 | | size * (D7 - D0) |
| Read Manufacturer/Device ID | 90h | size | A23 - A0 [24] = 0h / 1h | | ManID7 - ManID0, DevID7 - Dev0 |
| Read JEDEC ID | 9Fh | size | | | ManID7 - ManID0, MemID7 - MemID0, CapID7 - CapID0 |
| Multi I/O Performance Mode | A3h | | | | |
| Release Power Down/HighPerf.Mode optional: read Electronic Signature | ABh | (opt.): read size | | | (opt.): Sig79 - Sig0 |
| Read Status Register | 07h | | | | S7 - S0 (8) |
| Autoboot Register Read | 14h | | | | S31 - S0 (32) |
| Autoboot Register Write | 15h | | | S31 - S0 (32) | |
| Bank Address Register Read | 16h | | | | S7 - S0 (8) |
| Bank Address Register Write | 17h | | | S7 - S0 (8) | |
| ASP Read | 2Bh | | | | S15 - S0 (16) |
| ASP Program | 2Fh | | | S15 - S0 (16) | |
| Program Suspend | 85h | | | | |
| Program Resume | 8Ah | | | | |
| Write PPB Lock Register | A6h | | | S7 - S0 (8) | |
| Read PPB Lock Register | A7h | | | | S7 - S0 (8) |
| DYB Read | E0h | | A31 - A0 | | S7 - S0 (8) |
| DYB Write | E1h | | A31 - A0 | S7 - S0 (8) | |
| PPB Read | E2h | | A31 - A0 | | S7 - S0 (8) |
| PPB Write | E3h | | A31 - A0 | S7 - S0 (8) | |
| PPB Erase | E4h | | | | |
| Password Read | E7h | | | | S63 - S0 (64) |
| Password Program | E8h | | | S63 - S0 (64) | |
| Password Unlock | E9h | | | S63 - S0 (64) | |
| Software Reset | F0h | | | | |

1 The address bits are decided by the mode serial flash is working. For 32-bit extended address the bits used are A26 - A0, and for default 24-bit address mode the bits used are A23 - A0.

2 Denotes that 'size'-times one byte is transferred on the serial flash data bus. Total number of bytes must be provided in the TX Buffer or can be read from the RX Buffer.

3 According to Spansion documentation this command only supports a maximum clock speed of 40 MHz.

The QSPI_MCR[ICO] field is mapped to the different command options belonging to the complete SFM Command which is then sent to the serial flash. This is shown in detail in the following tables. Table 35-46 shows the commands usable in all flash access modes, Table 35-47 the commands specific to the Individual Flash Modes, and Table 35-48 the commands specific to the Parallel Flash Mode. All size information is given in number of bytes. Note that the mapping reflects the bits which are actually sent to the single serial flash device selected in the Individual Flash Mode or which are sent to each of the two serial flash devices in the Parallel Flash Mode.

**Table 35-46. ICO Field on Spansion Devices - All Flash Access Modes**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 01h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 02h | Size (bytes to be written) | | | | | | | | | | | | | | | | | | | | | | | |
| 05h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 32h | Size (bytes to be written) | | | | | | | | | | | | | | | | | | | | | | | |
| 35h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 03h | Individual Flash Access: Refer to Table 35-47<br>Parallel Flash Access: Refer to Table 35-48 | | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| BBh | | | | | | | | | | | | | | | | | | | | | | | | |
| 6Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| EBh | | | | | | | | | | | | | | | | | | | | | | | | |
| 90h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 9Fh | | | | | | | | | | Size | | | | | | | | | | | | | | |
| ABh | | | | | | | | | | read[1] | | | | | | | | | | | | | | |
| 07h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 14h | | | | | | | | | | | | | | Size | | | | | | | | | | |
| 15h | | | | | | | | | | | | | | Size | | | | | | | | | | |
| 16h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 17h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 2Bh | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 2Fh | | | | | | | | | | | | | | | Size | | | | | | | | | |

**Table 35-46. ICO Field on Spansion Devices - All Flash Access Modes (continued)**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| A6h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| A7h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| E0h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| E1h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| E2h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| E3h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| E7h | | | | | | | | | | Size | | | | | | | | | | | | | | |
| E8h | | | | | | | | | | Size | | | | | | | | | | | | | | |
| E9h | | | | | | | | | | Size | | | | | | | | | | | | | | |

1 'read' bit controls if up to 80 clocks are added to read the electronic signature, equivalent to Size = 10 (10bytes to read). In any case 24 dummy bits are transmitted after the instruction code to allow reading.

**Table 35-47. ICO Field on Spansion Devices - Individual Flash Mode**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 03h | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| BBh | Size (bytes to be read) | | | | | | | | | | | | | | | | **M7 - M0** | | | | | | | |
| 6Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| EBh | Size (bytes to be read) | | | | | | | | | | | | | | | | **M7 - M0** | | | | | | | |

Since both serial flash devices are treated logically as one single device doubled in size the amount of data that must be read from each single flash device is half the amount of data specified in the QSPI_ICR[ICO] field.

**Table 35-48. ICO Field on Spansion Devices - Parallel Flash Mode**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 03h | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |

**Table 35-48. ICO Field on Spansion Devices - Parallel Flash Mode (continued)**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 3Bh | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| BBh | | Size (bytes to be read) | | | | | | | | | | | | | | | **M7 - M0** | | | | | | | |
| 6Bh | x | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| EBh | | Size (bytes to be read) | | | | | | | | | | | | | | | **M7 - M0** | | | | | | | |

## 35.8.3 Instruction Codes in Macronix Devices

**Table 35-49. Macronix Instruction Codes**

| Command | Instruction Code | Required Input Data / Parameters | | | Output Data |
|---|---|---|---|---|---|
| | | Options QSPI_ICR[ICO] | Address [QSPI_SFAD] | Data [TX Buffer] | Status / Data [RX Buffer] |
| Write Enable | 06h | | | | |
| Write Disable | 04h | | | | |
| Read Status Reg | 05h | size | | | S7 - S0 [8] |
| Write Reg | 01h | size | | S15 - S0 | |
| Read Erase and prog fail flag | 30h | size | | S15 - S0 | |
| Page Program | 02h | size | A23 - A0 | size * (D7 - D0)[2] | |
| 4KB Sector Erase | 20h | | A23 - A0 | | |
| Block Erase | 52h / D8h | | A23 - A0 | | |
| Chip Erase | C7h / 60h | | | | |
| Deep Power Down | B9h | | | | |
| Read Data | 03h | size | A23 - A0 | | size * (D7 - D0)[1] |
| Fast Read | 0Bh | size | A23 - A0 | | size * (D7 - D0) |
| Dual Read Mode Sequence | 3Bh | size | A23 - A0 | | size * (D7 - D0) |
| Quad Read Mode | 6Bh | size | A23 - A0 | | size * (D7 - D0) |
| Dual I/O High Performance Read | BBh | M7 - M0, size | A23 - A0 | | size * (D7 - D0) |
| Quad I/O High Performance Read | EBh | M7 - M0, size | A23 - A0 | | size * (D7 - D0) |
| Read Electronix Manufacturer and Device ID | 90h / DFh / EFh | size | A23 - A0[24] = 0h / 1h | | ManID7 - ManID0, MemID7 - MemID0, CapID7 - CapID0 |

**Table 35-49. Macronix Instruction Codes (continued)**

| Command | Instruction Code | Required Input Data / Parameters | | | Output Data |
|---|---|---|---|---|---|
| | | Options QSPI_ICR[ICO] | Address [QSPI_SFAD][1] | Data [TX Buffer] | Status / Data [RX Buffer] |
| Read JEDEC ID | 9Fh | size | | | ManID7 - ManID0, MemID7 - MemID0, CapID7 - CapID0 |
| High Performance Enable Mode | A3h | | | | |
| Read Electronic Signature | ABh | (opt): read size | | | (opt.): Sig79 - Sig0 |
| Read Security Register | 2Bh | | | | S7 - S0 [8] |
| Write Security Register | 2Fh | | | S7 - S0 [8] | |
| Exit 4-byte Mode | E9h | | | | |
| Clear SR Fail Flag | 30h | | | | |
| Single Block Lock Protection | 36h | | A23 - A0 | | |
| Quad Page Program | 38h | | A23 - A0 | | size * (D7 - D0) |
| Single Block UnLock Protection | 39h | | A23 - A0 | | |
| Read Block Protection Lock Status | 3Ch | | A23 - A0 | | |
| Read DMC | 5Ah | | A23 - A0 | | size * (D7 - D0) |
| Write Protection Selection | 68h | | | | |
| Enable SO to output RY/BY | 70h | | | | |
| Gang Block Lock | 7Eh | | | | |
| Disable SO to output RY/BY | 80h | | | | |
| Gang Block UnLock | 98h | | | | |
| Continuous Mode | ADh | | A23 - A0 | S15 - S0 [16] | |
| Enter Secured OTP | B1h | | | | |
| Enter 4byte Mode | B7h | | | | |
| Exit Secured OTP | C1h | | | | |

1 The address bits are decided by the mode serial flash is working. For 32-bit extended address the bits used are A26 - A0, and for default 24-bit address mode the bits used are A23 - A0.

2 Denotes that ‘size’- times one byte is transferred on the serial flash data bus. Total number of bytes must be provided in the TX Buffer or can be read from the RX Buffer.

The QSPI_MCR[ICO] field is mapped to the different command options belonging to the complete SFM Command which is then sent to the serial flash. This is shown in detail in the following tables. Table 35-50 shows the commands usable in all flash access modes, Table 35-51 the commands specific to the Individual Flash Modes, and Table 35-52 the commands specific to the Parallel Flash Mode. All size information is given in number of bytes. Note that the mapping reflects the bits which are actually sent to

the single serial flash device selected in the Individual Flash Mode or which are sent to each of the two serial flash devices in the Parallel Flash Mode.

**Table 35-50. ICO Field on Macronix Devices - All Flash Access Modes**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 01h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 02h | Size (bytes to be written) | | | | | | | | | | | | | | | | | | | | | | | |
| 05h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 32h | Size (bytes to be written) | | | | | | | | | | | | | | | | | | | | | | | |
| 35h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 03h | Individual Flash Access: Refer to Table 35-51<br>Parallel Flash Access: Refer to Table 35-52 | | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| BBh | | | | | | | | | | | | | | | | | | | | | | | | |
| 6Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| EBh | | | | | | | | | | | | | | | | | | | | | | | | |
| 90h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 9Fh | | | | | | | | | | Size | | | | | | | | | | | | | | |
| ABh | | | | | | | | | | read[1] | | | | | | | | | | | | | | |
| 07h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 2Bh | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 2Fh | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 38h | Size (bytes to be written) | | | | | | | | | | | | | | | | | | | | | | | |
| 3Ch | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 5Ah | | | | | | | | | | | | | | | Size | | | | | | | | | |
| ADh | | | | | | | | | | | | | | | Size | | | | | | | | | |

1 'read' bit controls if up to 80 clocks are added to read the electronic signature, equivalent to Size = 10 (10bytes to read). In any case 24 dummy bits are transmitted after the instruction code to allow reading.

**Table 35-51. ICO Field on Macronix Devices - Individual Flash Mode**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 03h | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |

Table 35-51. ICO Field on Macronix Devices - Individual Flash Mode (continued)

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| BBh | Size (bytes to be read) | | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |
| 6Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| EBh | Size (bytes to be read) | | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |

Since both serial flash devices are treated logically as one single device doubled in size, the amount of data that must be read from each single flash device is half the amount of data specified in the QSPI_ICR[ICO] field.

Table 35-52. ICO Field on Macronix Devices - Parallel Flash Mode

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 03h | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| BBh | | Size (bytes to be read) | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |
| 6Bh | x | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| EBh | | Size (bytes to be read) | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |

## 35.8.4 Instruction Codes in Numonyx Devices

Table 35-53. Numonyx Instruction Codes

| Command | Instruction Code | Required Input Data / Parameters | | | Output Data |
|---|---|---|---|---|---|
| | | Options QSPI_ICR[ICO] | Address [QSPI_SFAD][1] | Data [TX Buffer] | Status / Data [RX Buffer] |
| Write Enable | 06h | | | | |
| Write Disable | 04h | | | | |
| Read Status Register | 05h | size | | | S7 - S0 [8] |
| Write Status Register | 01h | size | | S15 - S0 | |
| Page Program | 02h | size | A23 - A0 | size * (D7 - D0)[2] | |
| Sub Sector Erase | 20h | | A23 - A0 | | |
| Block Erase | D8h | | A23 - A0 | | |

**Table 35-53. Numonyx Instruction Codes (continued)**

| Command | Instruction Code | Required Input Data / Parameters | | | Output Data |
|---|---|---|---|---|---|
| | | Options QSPI_ICR[ICO] | Address [QSPI_SFAD] | Data [TX Buffer] | Status / Data [RX Buffer] |
| Bulk Erase | C7h | | | | |
| Read Data | 03h | size | A23 - A0 | | size * (D7 - D0)[1] |
| Fast Read | 0Bh | size | A23 - A0 | | size * (D7 - D0) |
| Dual Read Mode Sequence | 3Bh | size | A23 - A0 | | size * (D7 - D0) |
| Quad Read Mode | 6Bh | size | A23 - A0 | | size * (D7 - D0) |
| Dual I/O High Performance Read | BBh | M7 - M0, size | A23 - A0 | | size * (D7 - D0) |
| Quad I/O High Performance Read | EBh | M7 - M0, size | A23 - A0 | | size * (D7 - D0) |
| Read JEDEC ID | 9Fh | size | | | ManID7 - ManID0, MemID7 - MemID0, CapID7 - CapID0 |
| Exit Extended Address Mode | E9h | | | | |
| Read Flag Status Register | 70h | | | | S7 - S0 [8] |
| Write NV configuration Register | B1h | | | S15 - S0 [16] | |
| Read NV configuration Register | B5h | | | | S15 - S0 [16] |
| Enter Extended Address Mode | B7h | | | | |
| Quad Command Page Program | 12h | | A23 - A0 | | size * (D7 - D0) |
| Quad Input Extended Fast program | 32h | | A23 - A0 | | size * (D7 - D0) |
| OTP Program | 42h | | A23 - A0 | | size * (D7 - D0) |
| Read OTP | 4Bh | | A23 - A0 | | size * (D7 - D0) |
| Clear Status Flag Register | 50h | | | | |
| Write Volatile Enhanced Configuration | 61h | | | S7 - S0 [8] | |
| Read Volatile Enhanced Configuration | 65h | | | | S7 - S0 [8] |
| Erase Suspend | 75h | | | | |
| Erase Resume | 7Ah | | | | |
| Write Volatile Configuration Register | 81h | | | S7 - S0 [8] | |
| Read Volatile Configuration Register | 85h | | | | S7 - S0 [8] |
| Dual Input Fast Program | A2h | | | size * (D7 - D0) | |
| Multiple I/O Read Identification | AFh | | | | S23 - S0 [24] |

**Table 35-53. Numonyx Instruction Codes (continued)**

| Command | Instruction Code | Required Input Data / Parameters | | | Output Data |
|---|---|---|---|---|---|
| | | Options QSPI_ICR[ICO] | Address [QSPI_SFAD][1] | Data [TX Buffer] | Status / Data [RX Buffer] |
| Quad Output Fast Read | BCh | | A31-A0 | size * (D7 - D0) | |
| Write Extended Address Register | C5h | | | S7 - S0 [8] | |
| Read Extended Address Register | C8h | | | | S7 - S0 [8] |
| Dual Input Extended Fash Program | D2h | | A23 - A0 | size * (D7 - D0) | |
| Write to Lock Register | E5h | | A23 - A0 | S7 - S0 [8] | |

1 The address bits are decided by the mode serial flash is working. For 32-bit extended address the bits used are A26 - A0, and for default 24-bit address mode the bits used are A23 - A0.

2 Denotes that 'size'- times one byte is transferred on the serial flash data bus. Total number of bytes must be provided in the TX Buffer or can be read from the RX Buffer.

The QSPI_MCR[ICO] field is mapped to the different command options belonging to the complete SFM Command which is then sent to the serial flash. This is shown in detail in the following tables. Table 35-54 shows the commands usable in all flash access modes, Table 35-55 the commands specific to the Individual Flash Modes, and Table 35-56 the commands specific to the Parallel Flash Mode. All size information is given in number of bytes. Note that the mapping reflects the bits which are actually sent to the single serial flash device selected in the Individual Flash Mode or which are sent to each of the two serial flash devices in the Parallel Flash Mode.

**Table 35-54. ICO Field on Numonyx Devices - All Flash Access Modes**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 01h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 02h | Size (bytes to be written) | | | | | | | | | | | | | | | | | | | | | | | |
| 05h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 32h | Size (bytes to be written) | | | | | | | | | | | | | | | | | | | | | | | |
| 35h | | | | | | | | | | | | | | | Size | | | | | | | | | |
| 03h | Individual Flash Access: Refer to Table 35-55<br>Parallel Flash Access: Refer toTable 35-56 | | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| BBh | | | | | | | | | | | | | | | | | | | | | | | | |
| 6Bh | | | | | | | | | | | | | | | | | | | | | | | | |
| EBh | | | | | | | | | | | | | | | | | | | | | | | | |
| 90h | | | | | | | | | | | | | | | Size | | | | | | | | | |

**Table 35-54. ICO Field on Numonyx Devices - All Flash Access Modes (continued)**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 9Fh | | | | | | | | | | Size | | | | | | | | | | | | | | |
| ABh | | | | | | | | | | read[1] | | | | | | | | | | | | | | |

1 'read' bit controls if up to 80 clocks are added to read the electronic signature, equivalent to Size = 10 (10bytes to read). In any case 24 dummy bits are transmitted after the instruction code to allow reading.

**Table 35-55. ICO Field on Numonyx Devices - Individual Flash Mode**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 03h | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| BBh | Size (bytes to be read) | | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |
| 6Bh | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | | |
| EBh | Size (bytes to be read) | | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |

Since both serial flash devices are treated logically as one single device doubled in size, the amount of data that must be read from each single flash device is half the amount of data specified in the QSPI_ICR[ICO] field.

**Table 35-56. ICO Field on Numonyx Devices - Parallel Flash Mode**

| Instruction Code | QSPI_ICR[ICO] | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 03h | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| 0Bh | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| 3Bh | | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| BBh | | Size (bytes to be read) | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |
| 6Bh | x | Size (bytes to be read) | | | | | | | | | | | | | | | | | | | | | | |
| EBh | | Size (bytes to be read) | | | | | | | | | | | | | | | M7 - M0 | | | | | | | |

## 35.8.5 Serial Flash Clock Frequency Limitations

Certain commands of the Winbond serial flash devices are limited in the frequency applied to the serial flash device on command execution. To allow for higher clock speeds for the remaining commands the serial flash device clock can be divided by 2 (half speed) when executing such a command limited in frequency by setting the QSPI_SMPR[HSENA] bit. Refer to Table 35-41 for the commands affected.

## 35.9 Internal Sampling of Serial Flash Input Data

Depending from the actual implementation there is a delay between the internal clocking in the QuadSPI module and the external serial flash device. This means that the incoming data from the serial flash appear this delay later in time at the QuadSPI sampling logic w.r.t. internal reference clock. Refer to Figure 35-19 for an overview of this scheme.

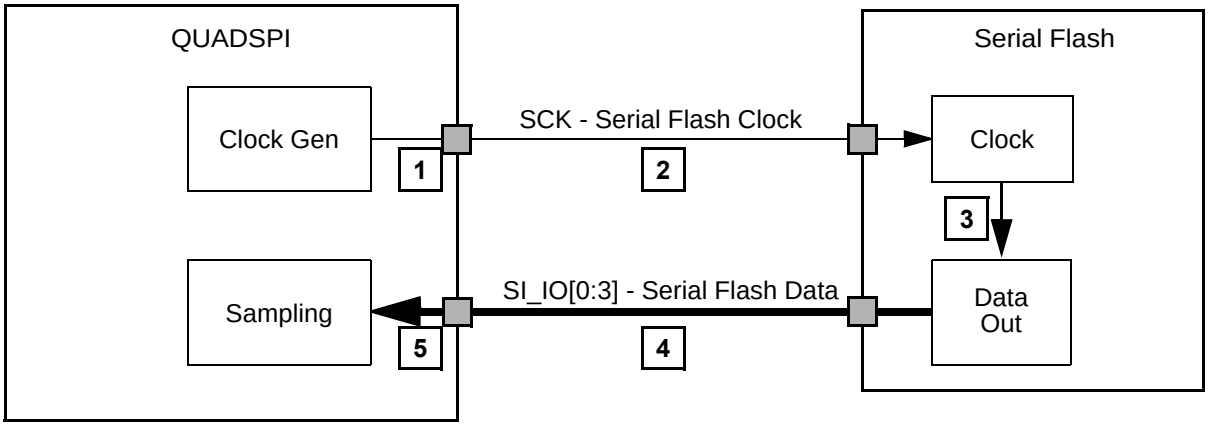



**Figure 35-19. Serial Flash Sampling Clock Overview**

**NOTE**

The arrival of the serial flash data in the sampling stage of the QuadSPI module are given in fig Figure 35-20 below. Note that the amount of the total delay $t_{Del,total}$ is very specific to the characteristics of the actual implementation.

Note also that the serial flash device clock SCK is inverted w.r.t. the QuadSPI internal reference clock.

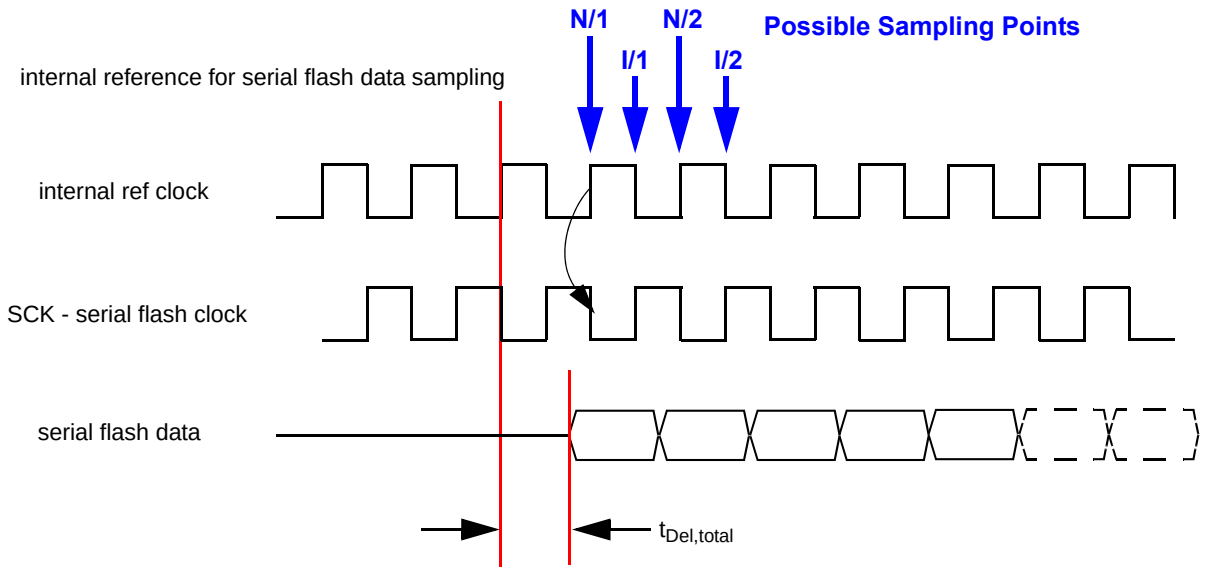



**Figure 35-20. Serial Flash Sampling Clock Timing**

The rising edge of the internal reference clock is taken as timing reference for the data output of the serial flash. After a time of $t_{Del,total}$ the data arrive at the internal sampling stage of the QuadSPI module. According to Figure 35-19 the following parts of the delay chain contribute to $t_{Del,total}$:

1. Output delay of the serial flash clock output of the device containing the QuadSPI module
2. Wire delay of application/PCB from the device containing the QuadSPI module to the external serial flash device
3. Clock to data out delay of the external serial flash device, including input and output delays
4. Wire delay of application/PCB from the external serial flash device to the device containing the QuadSPI module
5. Input delay belonging to the data in input

The possible points in time for the sampling of the incoming data are denoted as N/1, I/1, N/2 and I/2 above. The sampling point relevant for the internal sampling is configured in the QSPI_SMPR register, refer to Section 35.3.4.6, “Sampling Register (QSPI_SMPR)“” for details. Note that the falling edges of the reference clock are not actually used, instead the inverted clock is used for sampling at these positions. Table 35-57 below gives an overview of the available configurations for the commands running at regular (full) speed:

**Table 35-57. Sampling Configuration**

| Sampling Point | Description | Delay [FSDLY] [HSDLY] | Phase [FSPHS] [HSPHS] | QSPI_SMPR for Full Speed Setting[1] |
|---|---|---|---|---|
| N/1 | sampling with non-inverted clock, 1 sample delay | 0 | 0 | 0x0000000x |
| I/1 | sampling with inverted clock, 1 sample delay | 0 | 1 | 0x0000002x |

**Table 35-57. Sampling Configuration (continued)**

| Sampling Point | Description | Delay [FSDLY] [HSDLY] | Phase [FSPHS] [HSPHS] | QSPI_SMPR for Full Speed Setting[1] |
|---|---|---|---|---|
| N/2 | sampling with non-inverted clock, 2 samples delay | 1 | 0 | 0x0000004x |
| I/2 | sampling with inverted clock, 2 samples delay | 1 | 1 | 0x0000006x |

[1] 'x' is not considered here

Depending from the actual delay and the serial flash clock frequency the appropriate sampling point can be chosen. The following remarks should be considered when selecting the appropriate setting:

- Theoretically there should be 2 settings possible to capture the correct data since the serial flash output is valid for 1 clock cycle, disregarding rise and fall times and timing uncertainties.
- Depending from the timing uncertainties it may turn out in actual applications that only one possible sample positions remains. This is subject to careful consideration depending from the actual implementation.
- The delay $t_{Del,total}$ is an absolute size to shift the point in time when the serial flash date get valid at the QuadSPI input.
- For decreasing frequency of the serial flash clock the distance between the edges increases. So for large differences in the frequency the required setting may change.
- For commands running at half of the regular serial flash clock (QSPI_SMPR[HSENA] bit set) the sampling point must be figured separately to allow for the compensation of the absolute shift in time w.r.t. the sample-relative setting in the QSPI_SPMPR register.

This page is intentionally left blank.

# Chapter 36
# Real-Time Clock (RTC/API)

## 36.1 Overview

The RTC is a free running counter used for time keeping applications. The RTC may be configured to generate an interrupt at a predefined interval independent of the mode of operation (run mode or low power mode). If in a low power mode when the RTC interval is reached, the RTC will first generate a wakeup and then assert the interrupt request. The RTC also supports an autonomous periodic interrupt (API) function used to generate a periodic wakeup request to exit a low power mode or an interrupt request.

## 36.2 Features

Features of the RTC include:

- 4 selectable counter clock sources
  - 4–16 MHz FXOSC (OscA)
  - 128 kHz SIRC
  - 32 kHz SXOSC (OscB)
  - 16 MHz FIRC
- Optional 512 prescaler and optional 32 prescaler
- 32-bit counter
  - supports times up to 1.5 months with 1 ms resolution
  - runs in all modes of operation
  - reset when disabled by software and by POR
- 12-bit compare value to support interrupt intervals of 1s up to greater than 1 hr with 1s resolution
- RTC compare value changeable while counter is running
- RTC status and control register are reset only by POR
- Autonomous periodic interrupt (API)
  - 10-bit compare value to support wakeup intervals of 1.0 ms to 1 s
  - compare value changeable while counter is running
- Configurable interrupt for RTC match, API match, and RTC rollover
- Configurable wakeup event for RTC match, API match, and RTC rollover

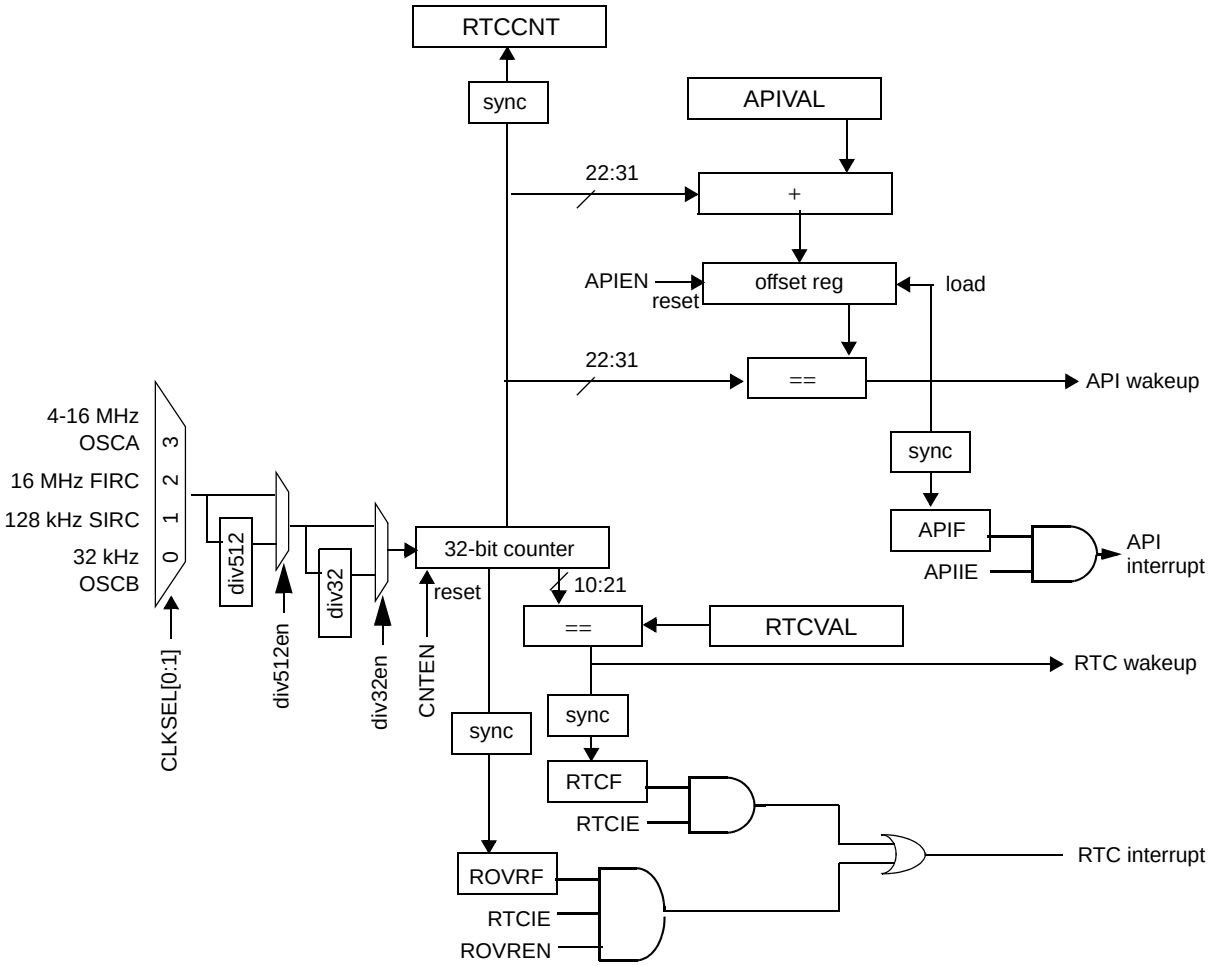



**Figure 36-1. RTC/API block diagram**

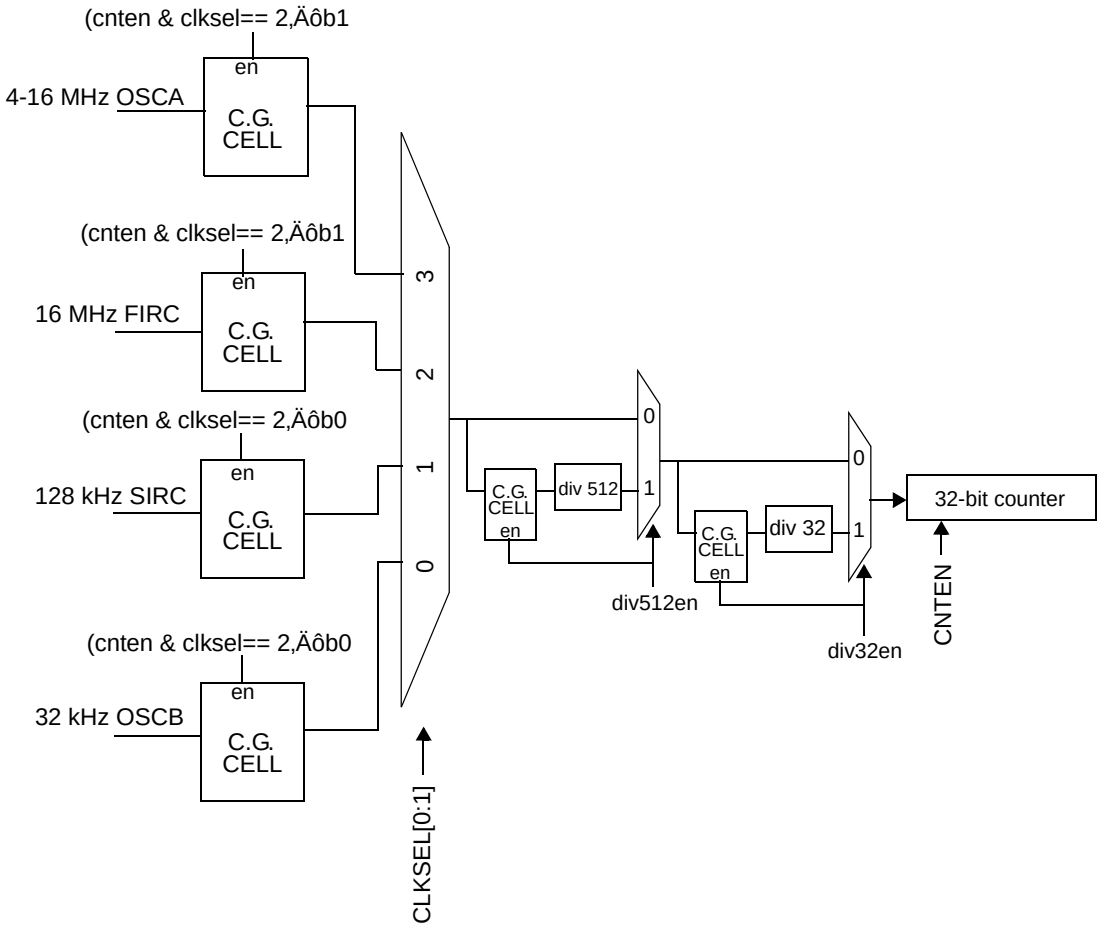



**Figure 36-2. Clock gating for RTC clocks**

## 36.3 Device-specific information

For this device:

- FXOSC, SXOSC, FIRC, and SIRC clocks are provided as counter clocks for the RTC. Default clock on reset is SIRC divided by 4.
- The RTC will be reset on destructive reset, with the exception of software watchdog reset.
- The RTC provides a configurable divider by 512 to be optionally used when FIRC source is selected.

## 36.4 Modes of operation

### 36.4.1 Functional mode

There are two functional modes of operation for the RTC: normal operation and low power mode. In normal operation, all RTC registers can read or written and the input isolation is disabled. The RTC/API

and associated interrupts are optionally enabled. In low power mode, the bus interface is disabled and the input isolation is enabled. The RTC/API is enabled if enabled prior to entry into low power mode.

### 36.4.2 Debug mode

On entering into the debug mode the RTC counter freezes on the last valid count if the RTCC[FRZEN] is set. On exit from debug mode counter continues from the frozen value.

## 36.5 Memory map and register descriptions

**Table 36-1. Memory map**

| Address offset | Register | Location |
|---|---|---|
| 0x0 | RTC Supervisor Control Register (RTCSUPV) | on page 1294 |
| 0x4 | RTC Control Register (RTCC) | on page 1294 |
| 0x8 | RTC Status Register (RTCS) | on page 1296 |
| 0xC | RTC Counter Register (RTCCNT) | on page 1297 |

### 36.5.1 RTC Supervisor Control Register (RTCSUPV)

The RTCSUPV register contains the SUPV bit which determines whether other registers are accessible in supervisor mode or user mode.

**NOTE**

RTCSUPV register is accessible only in supervisor mode.

Offset: RTC_BASE + 0x0000

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

**Figure 36-3. RTC Supervisor Control Register (RTCSUPV)**

**Table 36-2. RTCSUPV Register Bit/Field Descriptions**

| Field | Description |
|---|---|
| 0<br>SUPV | RTC Supervisor Bit<br>0 All registers are accessible in both user as well as supervisor mode.<br>1 All other registers are accessible in supervisor mode only. |

### 36.5.2 RTC Control Register (RTCC)

The RTCC register contains:

- RTC counter enable
- RTC interrupt enable
- RTC clock source select
- RTC compare value
- API enable
- API interrupt enable
- API compare value

Offset RTC_BASE + 0x0004 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CNTEN | RTCIE | FRZEN | ROVREN | RTCVAL[ | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | APIEN | APIIE | CLKSEL | | DIV512EN | DIV32EN | APIVAL | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 36-4. RTC Control Register (RTCC)**

**Table 36-3. RTCC field descriptions**

| Field | Description |
|---|---|
| CNTEN | Counter Enable<br>The CNTEN bit enables the RTC counter. Making CNTEN bit 1'b0 has the effect of asynchronously resetting (synchronous reset negation) all the RTC logic. This allows for the RTC configuration and clock source selection to be updated without causing synchronization issues.<br>1 Counter enabled<br>0 Counter disabled |
| RTCIE | RTC Interrupt Enable<br>The RTCIE bit enables interrupts requests to the system if RTCF is asserted.<br>1 RTC interrupts enabled<br>0 RTC interrupts disabled |
| FRZEN | Freeze Enable Bit<br>The counter freezes on entering the debug mode (as the ipg_debug is detected active) on the last valid count value if the FRZEN bit is set. After coming of the debug mode counter starts from the frozen value.<br>0 Counter does not freeze in debug mode.<br>1 Counter freezes in debug mode. |

**Table 36-3. RTCC field descriptions (continued)**

| Field | Description |
|---|---|
| ROVREN | Counter Roll Over Interrupt Enable<br>The ROVREN bit enables interrupt requests when the RTC has rolled over from 0xffff_ffff to 0x0000_0000. The RTCIE bit must also be set in order to generate an interrupt from a counter rollover.<br>1 RTC rollover interrupt enabled<br>0 RTC rollover interrupt disabled |
| RTCVAL | RTC Compare Value<br>The RTCVAL bits are compared to bits 10:21 of the RTC counter and if match sets RTCF. RTCVAL may only be updated when CNTEN is 0.<br>**Note:** RTCVAL should not be set to 0. |
| APIEN | Autonomous Periodic Interrupt Enable<br>The APIEN bit enables the autonomous periodic interrupt function.<br>1 API enabled<br>0 API disabled |
| APIIE | API Interrupt Enable<br>The APIIE bit enables interrupts requests to the system if APIF is asserted.<br>1 API interrupts enabled<br>0 API interrupts disabled |
| CLKSEL | Clock Select<br>The CLKSEL bits select the clock source for the RTC. CLKSEL may only be updated when CNTEN is 0. The user should ensure that oscillator is enabled before selecting it as a clock source for RTC.<br>00 SXOSC_clk_divided<br>01 SIRC_divided<br>10 IRC_fast_divided<br>11 FXOSC_clk_divided |
| DIV512EN | Divide by 512 enable<br>The DIV512EN bit enables the 512 clock divider. DIV512EN may only be updated when CNTEN is 0.<br>0 Divide by 512 is disabled.<br>1 Divide by 512 is enabled. |
| DIV32EN | Divide by 32 enable<br>The DIV32EN bit enables the 32 clock divider. DIV32EN may only be updated when CNTEN is 0.<br>0 Divide by 32 is disabled.<br>1 Divide by 32 is enabled. |
| APIVAL | API Compare Value<br>The APIVAL bits are added as an offset to the current RTC counter value which is then compared against the RTC counter. When there is a match, an API wakeup event occurs and if APIIE is set, an interrupt is also raised to the core. APIVAL can only be updated when APIEN is 0.<br>**Note:** API functionality starts only when APIVAL is non-zero. The first API interrupt takes two more cycles because of synchronization of APIVAL to the RTC clock and API_VAL+1 cycles for the subsequent occurrences. After that, interrupts are periodic in nature. The minimum supported value of APIVAL is 4. This is due to synchronization issues. |

## 36.5.3 RTC Status Register (RTCS)

The RTCS register contains:

- RTC interrupt flag
- API interrupt flag
- ROLLOVR Flag

Offset RTC_BASE + 0x0008 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | RTCF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | w1c | | | | | | | | | | | | | |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | APIF | 0 | 0 | ROVRF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | w1c | | | w1c | | | | | | | | | | |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 36-5. RTC Status Register (RTCS)**

**Table 36-4. RTCS field descriptions**

| Field | Description |
|---|---|
| RTCF | RTC Interrupt Flag<br>The RTCF bit indicates that the RTC counter has reached the counter value matching RTCVAL. RTCF is cleared by writing a 1 to RTCF. Writing a 0 to RTCF has no effect.<br>1 RTC interrupt<br>0 No RTC interrupt<br>**Note:** RTCF is not set if RTCVAL is 12'b0. |
| APIF | API Interrupt Flag<br>The APIF bit indicates that the RTC counter has reached the counter value matching API offset value. APIF is cleared by writing a 1 to APIF. Writing a 0 to APIF has no effect.<br>1 API interrupt<br>0 No API interrupt<br>Note: The periodic interrupt comes after APIVAL[0:9] + 1'b1 RTC counts |
| ROVRF | Counter Roll Over Interrupt Flag<br>The ROVRF bit indicates that the RTC has rolled over from 0xffff_ffff to 0x0000_0000. ROVRF is cleared by writing a 1 to ROVRF.<br>1 RTC has rolled over.<br>0 RTC has not rolled over. |

## 36.5.4 RTC Counter Register (RTCCNT)

The RTCCNT register contains the current value of the RTC counter.

Offset: RTC_BASE + 0x000C

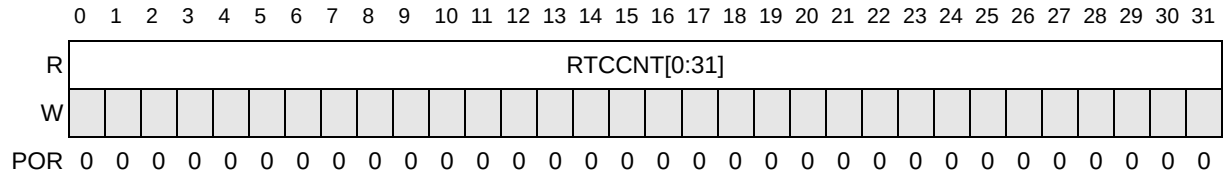



**Figure 36-6. RTC Counter Register (RTCCNT)**

**Table 36-5. RTCCNT field descriptions**

| Field | Description |
|---|---|
| RTCCNT | RTC Counter Value<br>Due to the clock synchronization, the RTCCNT value may actually represent a previous counter value. |

## 36.6 RTC functional description

The RTC consists of a 32-bit free running counter enabled with the RTCC[CNTEN] bit (CNTEN when negated asynchronously resets the counter and synchronously enables the counter when enabled). The value of the counter may be read via the RTCCNT register. Note that due to the clock synchronization, the RTCCNT value may actually represent a previous counter value. The difference between the counter and the read value depends on ratio of counter clock and ipg_clk. Maximum possible difference between the two is 6 count values.

The clock source to the counter is selected with the RTCC[CLKSEL] field, which gives four options for clocking the RTC/API. The four clock sources are assumed to be two 16 MHz sources, one 32 kHz source and one 128 kHz source. The output of the clock mux can be optionally divided by combination of 512 and 32 to give a 1 ms RTC/API count period for different clock sources. Note that the RTCC[CNTEN] bit must be disabled when the RTC/API clock source is switched.

When the counter value for counter bits 10:21 match the 12-bit value in the RTCC[RTCVAL] field, then the RTCS[RTCF] interrupt flag bit is set (after proper clock synchronization). If the RTCC[RTCIE] interrupt enable bit is set, then the RTC interrupt request is generated. The RTC supports interrupt requests in the range of 1s to 4096s (> 1 hr.) with a 1s resolution. The RTCC[RTCVAL] field may only be updated when the RTCC[CNTEN] bit is cleared to disable the counter. If there is a match while in low power mode then the RTC will first generate a wakeup request to force a wakeup to run mode, then the RTCF flag will be set.

A rollover interrupt can be generated when the RTC transitions from a count of 0xFFFF_FFFF to 0x0000_0000. The rollover flag is enabled by setting the RTCC[ROVREN] bit. An interrupt request is generated for an RTC counter rollover when both the RTCC[ROVREN] and RTCC[RTCIE] bits are set.

All the flags and counter values are synchronized with ipg_clk. It is assumed that ipg_clk frequency is always more than or equal to the rtc_clk used to run the counter.

## 36.7 API functional description

Setting RTCC[APIEN] bit enables the autonomous interrupt function. The 10-bit RTCC[APIVAL] field selects the time interval for triggering an interrupt and/or wakeup event. Since the RTC is a free running counter, the APIVAL is added to the current count to calculate an offset. When the counter reaches (offset count + 1), a interrupt and/or wakeup request is generated. Then the offset value is recalculated and again retriggers a new request when the new value is reached. APIVAL may only be updated when APIEN is disabled. When a compare is reached, the RTCS[APIF] interrupt flag bit is set (after proper clock synchronization). If the RTCC[APIIE] interrupt enable bit is set, then the API interrupt request is generated. If there is a match while in low power mode, then the API will first generate a wakeup request to force a wakeup into normal operation, then the APIF flag will be set.

This page is intentionally left blank.

# Chapter 37
# Reset Generation Module (MC_RGM)

## 37.1 Introduction

### 37.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the device. It provides a register interface and the reset sequencer. The different registers are available to monitor and control the device reset sequence. The reset sequencer is a state machine which controls the different phases (***PHASE0***, ***PHASE1***, ***PHASE2***, ***PHASE3***, and ***IDLE***) of the reset sequence and control the reset signals generated in the system.

Flash should never be powered down while going into a short functional Reset sequence.

Figure 37-1 depicts the MC_RGM block diagram.

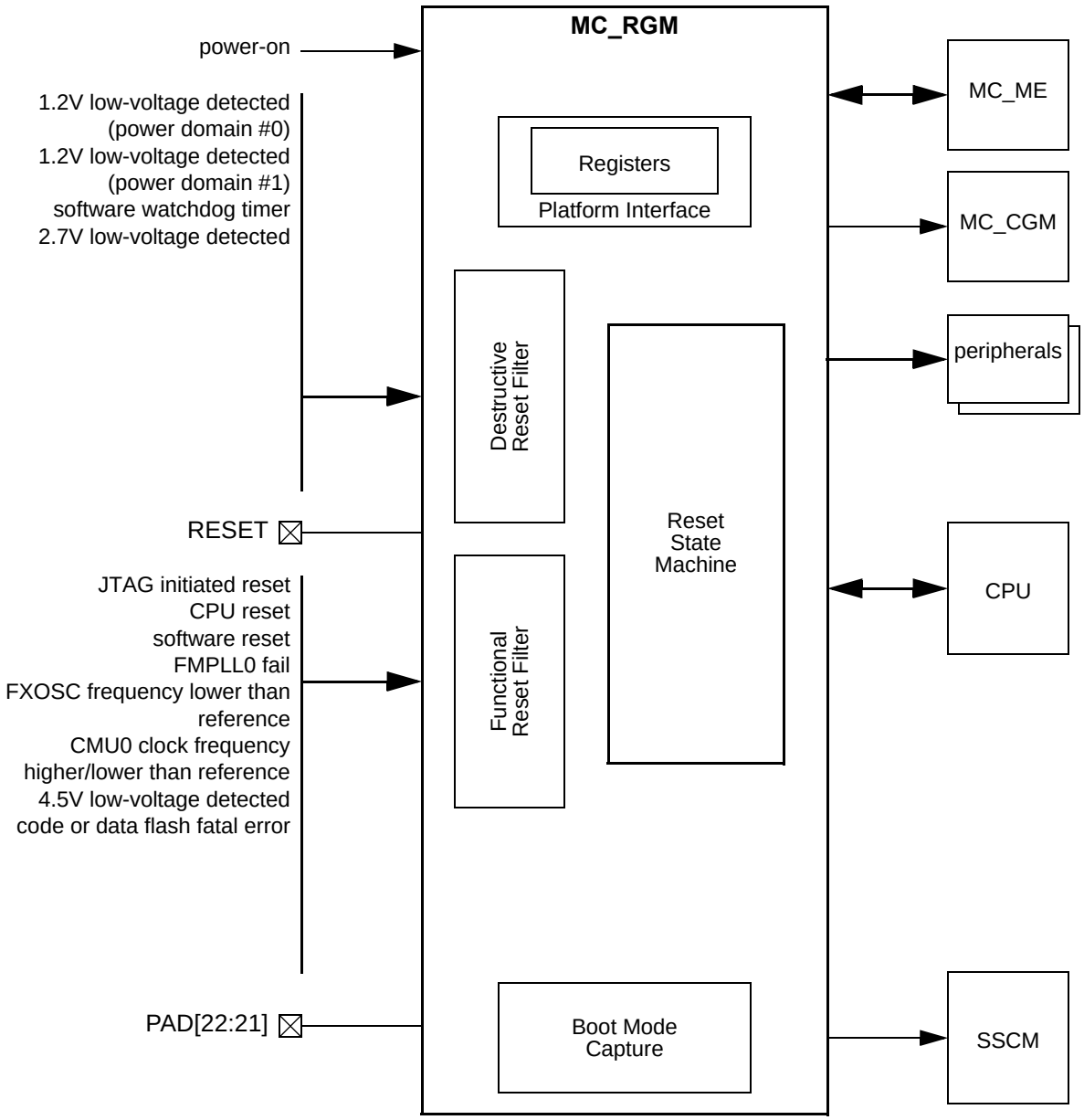



**Figure 37-1. MC_RGM block diagram**

## 37.1.2 Features

The MC_RGM contains the functionality for the following features:

- ‘destructive’ resets management
- ‘functional’ resets management
- signalling of reset events after each reset sequence (reset status flags)
- conversion of reset events to ***SAFE*** mode or interrupt request events (for further mode details, please see the MC_ME chapter)

- short reset sequence configuration
- bidirectional reset behavior configuration
- selection of alternate boot via the backup RAM on ***STANDBY*** mode exit (for further mode details, please see the MC_ME chapter)
- boot mode capture on RESET deassertion

### 37.1.3 Modes of operation

The different reset sources are organized into two families: ‘destructive’ and ‘functional’.

- A ‘destructive’ reset source is associated with an event related to a critical - usually hardware - error or dysfunction. When a ‘destructive’ reset event occurs, the full reset sequence is applied to the device starting from ***PHASE0***. This resets the full device ensuring a safe start-up state for both digital and analog modules. ‘Destructive’ resets are
  - power-on reset
  - 1.2V low-voltage detected (power domain #0)
  - 1.2V low-voltage detected (power domain #1)
  - software watchdog timer
  - 2.7V low-voltage detected
- A ‘functional’ reset source is associated with an event related to a less-critical - usually non-hardware - error or dysfunction. When a ‘functional’ reset event occurs, a partial reset sequence is applied to the device starting from ***PHASE1***. In this case, most digital modules are reset normally, while analog modules or specific digital modules’ (e.g. debug modules, flash modules) state is preserved. ‘Functional’ resets are
  - external reset
  - JTAG initiated reset
  - CPU reset
  - software reset
  - checkstop reset
  - FMPLL0 fail
  - FXOSC frequency lower than reference
  - CMU0 clock frequency higher/lower than reference
  - 4.5V low-voltage detected
  - FMPLL1 fail fatal error

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e. ***PHASEn*** states). Each phase is associated with a particular device reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The device reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the ***IDLE*** phase. During this entire process, the MC_ME state machine is held in ***RESET*** mode. Only at the end of the reset sequence, when the ***IDLE*** phase is reached, does the MC_ME enter the ***DRUN*** mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a ***SAFE*** mode request issued to the MC_ME or to an interrupt issued to the CPU (see Section 37.3.1.4, “Destructive Event Reset Disable Register (RGM_DERD) and Section 37.3.1.6, “Destructive Event Alternate Request Register (RGM_DEAR) for ‘destructive’ resets and Section 37.3.1.3, “Functional Event Reset Disable Register (RGM_FERD) and Section 37.3.1.5, “Functional Event Alternate Request Register (RGM_FEAR) for ‘functional’ resets).

## 37.2 External Signal Description

The MC_RGM interfaces to the bidirectional reset pin RESET and the boot mode pins PAD[22:21].

## 37.3 Memory Map and Register Definition

**Table 37-1. MC_RGM Register Description**

| Address | Name | Description | Location |
|---|---|---|---|
| 0xC3FE_4000 | RGM_FES | Functional Event Status | on page 1307 |
| 0xC3FE_4002 | RGM_DES | Destructive Event Status | on page 1308 |
| 0xC3FE_4004 | RGM_FERD | Functional Event Reset Disable | on page 1309 |
| 0xC3FE_4006 | RGM_DERD | Destructive Event Reset Disable | on page 1311 |
| 0xC3FE_4010 | RGM_FEAR | Functional Event Alternate Request | on page 1312 |
| 0xC3FE_4012 | RGM_DEAR | Destructive Event Alternate Request | on page 1313 |
| 0xC3FE_4018 | RGM_FESS | Functional Event Short Sequence | on page 1314 |
| 0xC3FE_401A | RGM_STDBY | ***STANDBY*** Reset Sequence | on page 1315 |
| 0xC3FE_401C | RGM_FBRE | Functional Bidirectional Reset Enable | on page 1316 |

**NOTE**

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

**Table 37-2. MC_RGM Memory Map**

| Address | Name | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FE_4000 | RGM_FES / RGM_DES | R | F_EXR | 0 | 0 | 0 | 0 | 0 | 0 | F_FMPLL1 | F_LVD45 | F_CMU0_FHL | F_CMU0_OLR | F_FMPLL0 | 0 | F_SOFT | F_CORE | F_JTAG |
| | | W | w1c | | | | | | | w1c | w1c | w1c | w1c | w1c | | w1c | w1c | w1c |
| | | R | F_POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | F_LVD27 | F_SWT | F_LVD12_PD1 | F_LVD12_PD0 |
| | | W | w1c | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| 0xC3FE_4004 | RGM_FERD / RGM_DERD | R | D_EXR | 0 | 0 | 0 | 0 | 0 | 0 | D_FMPLL1 | D_LVD45 | D_CMU0_FHL | D_CMU0_OLR | D_FMPLL0 | 0 | D_SOFT | D_CORE | D_JTAG |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | D_LVD27 | D_SWT | D_LVD12_PD1 | D_LVD12_PD0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_4008 … 0xC3FE_400C | reserved | | | | | | | | | | | | | | | | | |
| 0xC3FE_4010 | RGM_FEAR / RGM_DEAR | R | AR_EXR | 0 | 0 | 0 | 0 | 0 | 0 | AR_FMPLL1 | AR_LVD45 | AR_CMU0_FHL | AR_CMU0_OLR | AR_FMPLL0 | 0 | AR_SOFT | AR_CORE | AR_JTAG |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | AR_LVD27 | AR_SWT | AR_LVD12_PD1 | AR_LVD12_PD0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_4014 | reserved | | | | | | | | | | | | | | | | | |

**Table 37-2. MC_RGM Memory Map (continued)**

| Address | Name | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0xC3FE_4018 | RGM_FESS / RGM_STDBY | R | SS_EXR | 0 | 0 | 0 | 0 | 0 | 0 | SS_FLASH | SS_LVD45 | SS_CMU0_FHL | SS_CMU0_OLR | SS_FMPLL0 | 0 | SS_SOFT | SS_CORE | SS_JTAG |
| | | W | | | | | | | | | | | | | | | | |
| | | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BOOT_FROM_BKP_RAM | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_401C | RGM_FBRE | R | BE_EXR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BE_LVD45 | BE_CMU0_FHL | BE_CMU0_OLR | BE_FMPLL0 | | BE_SOFT | BE_CORE | BE_JTAG |
| | | W | | | | | | | | | | | | | | | | |
| 0xC3FE_4020 … 0xC3FE_7FFC | reserved | | | | | | | | | | | | | | | | | |

## 37.3.1 Register Descriptions

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the **RGM_STDBY** register may be accessed as a word at address 0xC3FE_4018, as a half-word at address 0xC3FE_401A, or as a byte at address 0xC3FE_401B.

### 37.3.1.1 Functional Event Status Register (RGM_FES)

Address 0xC3FE_4000 Access: Supervisor read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | F_EXR | 0 | 0 | 0 | 0 | 0 | 0 | F_FMPLL1 | F_LVD45 | F_CMU0_FHL | F_CMU0_OLR | F_FMPLL0 | 0 | F_SOFT | F_CORE | F_JTAG |
| W | w1c | | | | | | | w1c | w1c | w1c | w1c | w1c | | w1c | w1c | w1c |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 37-2. Functional Event Status Register (RGM_FES)**

This register contains the status of the last asserted functional reset sources. It can be accessed in read/write on either supervisor mode or test mode. Register bits are cleared on write ‘1’.

**Table 37-3. Functional Event Status Register (RGM_FES) Field Descriptions**

| Field | Description |
|---|---|
| F_EXR | **Flag for External Reset**<br>0 No external reset event has occurred since either the last clear or the last destructive reset assertion<br>1 An external reset event has occurred |
| F_FMPLL1 | **Flag for FMPLL1 fail**<br>0 No FMPLL1 fail event has occurred since either the last clear or the last destructive reset assertion<br>1 A FMPLL1 fail event has occurred |
| F_LVD45 | **Flag for 4.5V low-voltage detected**<br>0 No 4.5V low-voltage detected event has occurred since either the last clear or the last destructive reset assertion<br>1 A 4.5V low-voltage detected event has occurred |
| F_CMU0_FHL | **Flag for CMU0 clock frequency higher/lower than reference**<br>0 No CMU0 clock frequency higher/lower than reference event has occurred since either the last clear or the last destructive reset assertion<br>1 A CMU0 clock frequency higher/lower than reference event has occurred |
| F_CMU0_OLR | **Flag for FXOSC frequency lower than reference**<br>0 No FXOSC frequency lower than reference event has occurred since either the last clear or the last destructive reset assertion<br>1 A FXOSC frequency lower than reference event has occurred |
| F_FMPLL0 | **Flag for FMPLL0 fail**<br>0 No FMPLL0 fail event has occurred since either the last clear or the last destructive reset assertion<br>1 A FMPLL0 fail event has occurred |
| F_SOFT | **Flag for software reset**<br>0 No software reset event has occurred since either the last clear or the last destructive reset assertion<br>1 A software reset event has occurred |

**Table 37-3. Functional Event Status Register (RGM_FES) Field Descriptions (continued)**

| Field | Description |
|---|---|
| F_CORE | **Flag for CPU reset**<br>0 No CPU reset event has occurred since either the last clear or the last destructive reset assertion<br>1 A CPU reset event has occurred; this event can only be asserted when the RST field of the DBCR0 register is set by an external debugger. Please refer to chapter “Debug Support” of the core reference manual for more details. |
| F_JTAG | **Flag for JTAG initiated reset**<br>0 No JTAG initiated reset event has occurred since either the last clear or the last destructive reset assertion<br>1 A JTAG initiated reset event has occurred |

### 37.3.1.2 Destructive Event Status Register (RGM_DES)

Address 0xC3FE_4002 Access: Supervisor read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | F_POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | F_LVD27 | F_SWT | F_LVD12_PD1 | F_LVD12_PD0 |
| W | w1c | | | | | | | | | | | | w1c | w1c | w1c | w1c |
| POR | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 37-3. Destructive Event Status Register (RGM_DES)**

This register contains the status of the last asserted destructive reset sources. It can be accessed in read/write on either supervisor mode or test mode. Register bits are cleared on write ‘1’.

**Table 37-4. Destructive Event Status Register (RGM_DES) Field Descriptions**

| Field | Description |
|---|---|
| F_POR | **Flag for Power-On reset**<br>0 No power-on event has occurred since the last clear (due to either a software clear or a low-voltage detection)<br>1 A power-on event has occurred |
| F_LVD27 | **Flag for 2.7V low-voltage detected**<br>0 No 2.7V low-voltage detected event has occurred since either the last clear or the last power-on reset assertion<br>1 A 2.7V low-voltage detected event has occurred |
| F_SWT | **Flag for software watchdog timer**<br>0 No software watchdog timer event has occurred since either the last clear or the last power-on reset assertion<br>1 A software watchdog timer event has occurred |

**Table 37-4. Destructive Event Status Register (RGM_DES) Field Descriptions (continued)**

| Field | Description |
|---|---|
| F_LVD12_PD1 | **Flag for 1.2V low-voltage detected (power domain #1)**<br>0 No 1.2V low-voltage detected (power domain #1) event has occurred since either the last clear or the last power-on reset assertion<br>1 A 1.2V low-voltage detected (power domain #1) event has occurred |
| F_LVD12_PD0 | **Flag for 1.2V low-voltage detected (power domain #0)**<br>0 No 1.2V low-voltage detected (power domain #0) event has occurred since either the last clear or the last power-on reset assertion<br>1 A 1.2V low-voltage detected (power domain #0) event has occurred |

**NOTE**

The **F_POR** flag is automatically cleared on a 1.2V low-voltage detected (power domain #0 or #1) or a 2.7V low-voltage detected (VREG). This means that if the power-up sequence is not monotonic (i.e the voltage rises and then drops enough to trigger a low-voltage detection), the **F_POR** flag may not be set but instead the **F_LVD12_PD0**, **F_LVD12_PD1**, or **F_LVD27_VREG** flag is set on exiting the reset sequence. Therefore, if the **F_POR**, **F_LVD12_PD0**, **F_LVD12_PD1**, or **F_LVD27_VREG** flags are set on reset exit, software should interpret the reset cause as power-on.

**NOTE**

In contrast to all other reset sources, the 1.2V low-voltage detected (power domain #0) event is captured on its deassertion. Therefore, the status bit **F_LVD12_PD0** is also asserted on the reset's deassertion. In case an alternate event is selected, the ***SAFE*** mode or interrupt request are similarly asserted on the reset's deassertion.

### 37.3.1.3 Functional Event Reset Disable Register (RGM_FERD)

Address 0xC3FE_4004 Access: Supervisor read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | D_EXR | 0 | 0 | 0 | 0 | 0 | 0 | D_FMPLL1 | D_LVD45 | D_CMU0_FHL | D_CMU0_OLR | D_FMPLL0 | 0 | D_SOFT | D_CORE | D_JTAG |
| W | | | | | | | | | | | | | | | | |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 37-4. Functional Event Reset Disable Register (RGM_FERD)**

This register provides dedicated bits to disable functional reset sources.When a functional reset source is disabled, the associated functional event will trigger either a ***SAFE*** mode request or an interrupt request (see Section 37.3.1.5, "Functional Event Alternate Request Register (RGM_FEAR)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after power-on reset.

**Table 37-5. Functional Event Reset Disable Register (RGM_FERD) Field Descriptions**

| Field | Description |
|---|---|
| D_EXR | **Disable External Reset**<br>0 An external reset event triggers a reset sequence<br>1 An external reset event generates a ***SAFE*** mode request |
| D_FMPLL1 | **Disable FMPLL1 fail fatal error**<br>0 An FMPLL1 fail fatal error event triggers a reset sequence<br>1 An FMPLL1 fail fatal error event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_FEAR[AR_FMPLL1]** |
| D_LVD45 | **Disable 4.5V low-voltage detected**<br>0 A 4.5V low-voltage detected event triggers a reset sequence<br>1 A 4.5V low-voltage detected event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_FEAR.AR_LVD45** |
| D_CMU0_FHL | **Disable CMU0 clock frequency higher/lower than reference**<br>0 A CMU0 clock frequency higher/lower than reference event triggers a reset sequence<br>1 A CMU0 clock frequency higher/lower than reference event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_FEAR.AR_CMU0_FHL** |
| D_CMU0_OLR | **Disable FXOSC frequency lower than reference**<br>0 A FXOSC frequency lower than reference event triggers a reset sequence<br>1 A FXOSC frequency lower than reference event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_FEAR.AR_CMU0_OLR** |
| D_FMPLL0 | **Disable FMPLL0 fail**<br>0 A FMPLL0 fail event triggers a reset sequence<br>1 A FMPLL0 fail event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_FEAR.AR_FMPLL0** |
| D_SOFT | **Disable software reset**<br>0 A software reset event triggers a reset sequence<br>1 A software reset event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_FEAR.AR_SOFT** |
| D_CORE | **Disable CPU reset**<br>0 A CPU reset event triggers a reset sequence<br>1 A CPU reset event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_FEAR.AR_CORE** |
| D_JTAG | **Disable JTAG initiated reset**<br>0 A JTAG initiated reset event triggers a reset sequence<br>1 A JTAG initiated reset event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_FEAR.AR_JTAG** |

### 37.3.1.4 Destructive Event Reset Disable Register (RGM_DERD)

Address 0xC3FE_4006 Access: Supervisor read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | D_LVD27 | D_SWT | D_LVD12_PD1 | D_LVD12_PD0 |
| W | | | | | | | | | | | | | | | | |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 37-5. Destructive Event Reset Disable Register (RGM_DERD)**

This register provides dedicated bits to disable particular destructive reset sources. When a destructive reset source is disabled, the associated destructive event will trigger either a safe mode request or an interrupt request (see Section 37.3.1.6, “Destructive Event Alternate Request Register (RGM_DEAR)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after power-on reset.

**Table 37-6. Destructive Event Reset Disable Register (RGM_DERD) Field Descriptions**

| Field | Description |
|---|---|
| D_LVD27 | **Disable 2.7V low-voltage detected**<br>0 A 2.7V low-voltage detected event triggers a reset sequence<br>1 A 2.7V low-voltage detected event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_DEAR.AR_LVD27** |
| D_SWT | **Disable software watchdog timer**<br>0 A software watchdog timer event triggers a reset sequence<br>1 A software watchdog timer event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_DEAR.ep** |
| D_LVD12_PD1 | **Disable 1.2V low-voltage detected (power domain #1)**<br>0 A 1.2V low-voltage detected (power domain #1) event triggers a reset sequence<br>1 A 1.2V low-voltage detected (power domain #1) event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_DEAR.AR_LVD12_PD1** |
| D_LVD12_PD0 | **Disable 1.2V low-voltage detected (power domain #0)**<br>0 A 1.2V low-voltage detected (power domain #0) event triggers a reset sequence<br>1 A 1.2V low-voltage detected (power domain #0) event generates either a ***SAFE*** mode or an interrupt request depending on the value of **RGM_DEAR.AR_LVD12_PD0** |

### 37.3.1.5 Functional Event Alternate Request Register (RGM_FEAR)

Address 0xC3FE_4010 Access: Supervisor read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | AR_EXR | 0 | 0 | 0 | 0 | 0 | 0 | AR_FMPLL1 | AR_LVD45 | AR_CMU0_FHL | AR_CMU0_OLR | AR_FMPLL0 | | AR_SOFT | AR_CORE | AR_JTAG |
| W | | | | | | | | | | | | | | | | |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 37-6. Functional Event Alternate Request Register (RGM_FEAR)**

This register defines an alternate request to be generated when a reset on a functional event has been disabled. The alternate request can be either a ***SAFE*** mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

**Table 37-7. Functional Event Alternate Request Register (RGM_FEAR) Field Descriptions**

| Field | Description |
|---|---|
| AR_EXR | **Alternate Request for External Reset**<br>0 Generate a ***SAFE*** mode request on an **external reset** event if the reset is disabled<br>1 Generate an interrupt request on an **external reset** event if the reset is disabled |
| AR_FMPLL1 | **Alternate Request for FMPLL1 fail fatal error**<br>0 Generate a ***SAFE*** mode request on an FMPLL1 fail fatal error event if the reset is disabled<br>1 Generate an interrupt request on an FMPLL1 fail fatal error event if the reset is disabled |
| AR_LVD45 | **Alternate Request for 4.5V low-voltage detected**<br>0 Generate a ***SAFE*** mode request on a 4.5V low-voltage detected event if the reset is disabled<br>1 Generate an interrupt request on a 4.5V low-voltage detected event if the reset is disabled |
| AR_CMU0_FHL | **Alternate Request for CMU0 clock frequency higher/lower than reference**<br>0 Generate a ***SAFE*** mode request on a CMU0 clock frequency higher/lower than reference event if the reset is disabled<br>1 Generate an interrupt request on a CMU0 clock frequency higher/lower than reference event if the reset is disabled |
| AR_CMU0_OLR | **Alternate Request for FXOSC frequency lower than reference**<br>0 Generate a ***SAFE*** mode request on a FXOSC frequency lower than reference event if the reset is disabled<br>1 Generate an interrupt request on a FXOSC frequency lower than reference event if the reset is disabled<br>Note: For the case when RGM_FERD[D_CMU0_OLR] = 1 & RGM_FEAR[AR_CMU0_OLR] = 1 a RGM interrupt will not be generated for a FXOSC failure when the System Clock = FXOSC as there will be no System Clock to execute the interrupt service routine. However, the interrupt service routine will be executed if the FXOSC recovers at some point. The recommended use case for this feature is when the System Clock = FIRC or FMPLL. |
| AR_FMPLL0 | **Alternate Request for FMPLL0 fail**<br>0 Generate a ***SAFE*** mode request on a FMPLL0 fail event if the reset is disabled<br>1 Generate an interrupt request on a FMPLL0 fail event if the reset is disabled |
| AR_SOFT | **Alternate Request for software reset**<br>0 Generate a ***SAFE*** mode request on a software reset event if the reset is disabled<br>1 Generate an interrupt request on a software reset event if the reset is disabled |

Table 37-7. Functional Event Alternate Request Register (RGM_FEAR) Field Descriptions (continued)

| Field | Description |
|---|---|
| AR_CORE | **Alternate Request for CPU reset**<br>0 Generate a ***SAFE*** mode request on a CPU reset event if the reset is disabled<br>1 Generate an interrupt request on a CPU reset event if the reset is disabled |
| AR_JTAG | **Alternate Request for JTAG initiated reset**<br>0 Generate a ***SAFE*** mode request on a JTAG initiated reset event if the reset is disabled<br>1 Generate an interrupt request on a JTAG initiated reset event if the reset is disabled |

### 37.3.1.6 Destructive Event Alternate Request Register (RGM_DEAR)

Address 0xC3FE_4012 Access: Supervisor read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | AR_LVD27 | AR_SWT | AR_LVD12_PD1 | AR_LVD12_PD0 |
| W | | | | | | | | | | | | | | | | |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 37-7. Destructive Event Alternate Request Register (RGM_DEAR)

This register defines an alternate request to be generated when a reset on a destructive event has been disabled. The alternate request can be either a ***SAFE*** mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Table 37-8. Destructive Event Alternate Request Register (RGM_DEAR) Field Descriptions

| Field | Description |
|---|---|
| AR_LVD27 | **Alternate Request for 2.7V low-voltage detected**<br>0 Generate a ***SAFE*** mode request on a 2.7V low-voltage detected event if the reset is disabled<br>1 Generate an interrupt request on a 2.7V low-voltage detected event if the reset is disabled |
| AR_SWT | **Alternate Request for software watchdog timer**<br>0 Generate a ***SAFE*** mode request on a software watchdog timer event if the reset is disabled<br>1 Generate an interrupt request on a software watchdog timer event if the reset is disabled |
| AR_LVD12_PD1 | **Alternate Request for 1.2V low-voltage detected (power domain #1)**<br>0 Generate a ***SAFE*** mode request on a 1.2V low-voltage detected (power domain #1) event if the reset is disabled<br>1 Generate an interrupt request on a 1.2V low-voltage detected (power domain #1) event if the reset is disabled |
| AR_LVD12_PD0 | **Alternate Request for 1.2V low-voltage detected (power domain #0)**<br>0 Generate a ***SAFE*** mode request on a 1.2V low-voltage detected (power domain #0) event if the reset is disabled<br>1 Generate an interrupt request on a 1.2V low-voltage detected (power domain #0) event if the reset is disabled |

### 37.3.1.7 Functional Event Short Sequence Register (RGM_FESS)

Address 0xC3FE_4018 Access: Supervisor read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SS_EXR | 0 | 0 | 0 | 0 | 0 | 0 | SS_FLASH | SS_LVD45 | SS_CMU0_FHL | SS_CMU0_OLR | SS_FMPLL0 | 0 | SS_SOFT | SS_CORE | SS_JTAG |
| W | | | | | | | | | | | | | | | | |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 37-8. Functional Event Short Sequence Register (RGM_FESS)**

This register defines which reset sequence will be done when a functional reset sequence is triggered.The functional reset sequence can either start from ***PHASE1*** or from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***.

**NOTE**

This could be useful for fast reset sequence, for example to skip flash reset. However, short sequence resets should not be used when in a mode with Flash configured to power-down state. In this case a full PHASE1 reset is required in order to power up the Flash - therefore FESS should be set to 0x0.

**NOTE**

If any functional event is defined to perform a short reset sequence then it will not also assert the external reset pin even if that option is selected in the RGM_FBRE register. Short reset sequences can be useful to allow a fast reset sequence, for example to skip flash reset.

It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

**Table 37-9. Functional Event Short Sequence Register (RGM_FESS) Field Descriptions**

| Field | Description |
|---|---|
| SS_EXR | **Short Sequence for External Reset**<br>0 The reset sequence triggered by an **external** reset event will start from ***PHASE1***<br>1 The reset sequence triggered by an **external** reset event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2*** |
| SS_FLASH | **Short Sequence for code or data flash fatal error**<br>0 The reset sequence triggered by a code or data flash fatal error event will start from ***PHASE1***<br>1 The reset sequence triggered by a code or data flash fatal error event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2*** |
| SS_LVD45 | **Short Sequence for 4.5V low-voltage detected**<br>0 The reset sequence triggered by a 4.5V low-voltage detected event will start from ***PHASE1***<br>1 The reset sequence triggered by a 4.5V low-voltage detected event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2*** |

**Table 37-9. Functional Event Short Sequence Register (RGM_FESS) Field Descriptions (continued)**

| Field | Description |
|---|---|
| SS_CMU0_FHL | **Short Sequence for CMU0 clock frequency higher/lower than reference**<br>0 The reset sequence triggered by a CMU0 clock frequency higher/lower than reference event will start from ***PHASE1***<br>1 The reset sequence triggered by a CMU0 clock frequency higher/lower than reference event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2*** |
| SS_CMU0_OLR | **Short Sequence for FXOSC frequency lower than reference**<br>0 The reset sequence triggered by a FXOSC frequency lower than reference event will start from ***PHASE1***<br>1 The reset sequence triggered by a FXOSC frequency lower than reference event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2*** |
| SS_FMPLL0 | **Short Sequence for FMPLL0 fail**<br>0 The reset sequence triggered by a FMPLL0 fail event will start from ***PHASE1***<br>1 The reset sequence triggered by a FMPLL0 fail event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2*** |
| SS_SOFT | **Short Sequence for software reset**<br>0 The reset sequence triggered by a software reset event will start from ***PHASE1***<br>1 The reset sequence triggered by a software reset event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2*** |
| SS_CORE | **Short Sequence for CPU reset**<br>0 The reset sequence triggered by a CPU reset event will start from ***PHASE1***<br>1 The reset sequence triggered by a CPU reset event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2*** |
| SS_JTAG | **Short Sequence for JTAG initiated reset**<br>0 The reset sequence triggered by a JTAG initiated reset event will start from ***PHASE1***<br>1 The reset sequence triggered by a JTAG initiated reset event will start from ***PHASE3***, skipping ***PHASE1*** and ***PHASE2*** |

### 37.3.1.8 *STANDBY* Reset Sequence Register (RGM_STDBY)

Address 0xC3FE_401A Access: Supervisor read/write

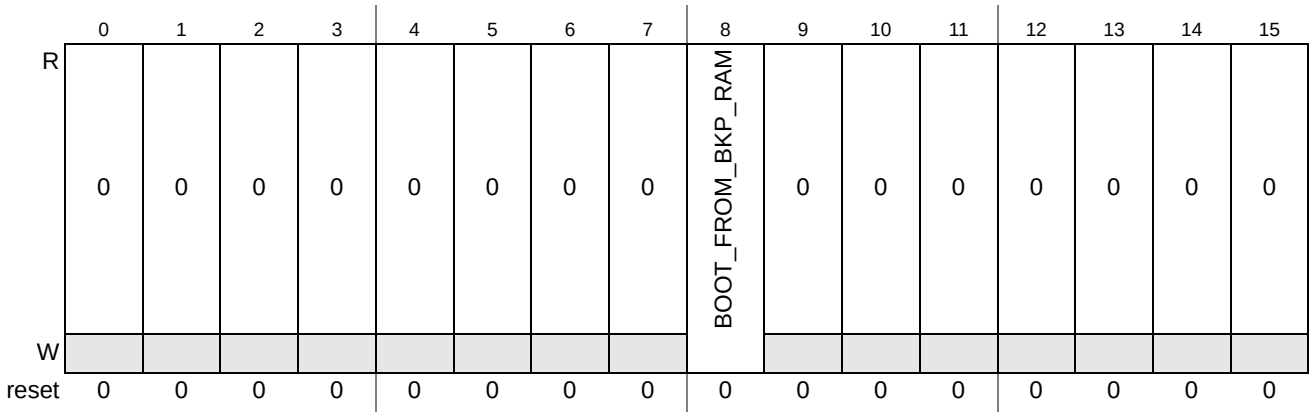



**Figure 37-9. STANDBY Reset Sequence Register (RGM_STDBY)**

This register defines reset sequence to be applied on ***STANDBY*** mode exit. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

**Table 37-10. STANDBY Reset Sequence Register (RGM_STDBY) Field Descriptions**

| Field | Description |
|---|---|
| BOOT_FROM_BKP_RAM | **Boot from Backup RAM indicator** — This bit indicates whether the system will boot from backup RAM or flash out of ***STANDBY*** exit.<br>0 Boot from flash on ***STANDBY*** exit<br>1 Boot from backup RAM on ***STANDBY*** exit |

**NOTE**

This register is reset on any enabled ‘destructive’ or ‘functional’ reset event.

### 37.3.1.9 Functional Bidirectional Reset Enable Register (RGM_FBRE)

Address 0xC3FE_401C Access: Supervisor read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BE_EXR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BE_LVD45 | BE_CMU0_FHL | BE_CMU0_OLR | BE_FMPLL0 | 0 | BE_SOFT | BE_CORE | BE_JTAG |
| W | | | | | | | | | | | | | | | | |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 37-10. Functional Bidirectional Reset Enable Register (RGM_FBRE)**

This register enables the generation of an external reset on functional reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

**Table 37-11. Functional Bidirectional Reset Enable Register (RGM_FBRE) Field Descriptions**

| Field | Description |
|---|---|
| BE_EXR | **Bidirectional Reset Enable for External Reset**<br>0 RESET is asserted on an **external** reset event if the reset is enabled<br>1 RESET is not asserted on an **external** reset event |
| BE_LVD45 | **Bidirectional Reset Enable for 4.5V low-voltage detected**<br>0 RESET is asserted on a 4.5V low-voltage detected event if the reset is enabled<br>1 RESET is not asserted on a 4.5V low-voltage detected event |
| BE_CMU0_FHL | **Bidirectional Reset Enable for CMU0 clock frequency higher/lower than reference**<br>0 RESET is asserted on a CMU0 clock frequency higher/lower than reference event if the reset is enabled<br>1 RESET is not asserted on a CMU0 clock frequency higher/lower than reference event |
| BE_CMU0_OLR | **Bidirectional Reset Enable for FXOSC frequency lower than reference**<br>0 RESET is asserted on a FXOSC frequency lower than reference event if the reset is enabled<br>1 RESET is not asserted on a FXOSC frequency lower than reference event |
| BE_FMPLL0 | **Bidirectional Reset Enable for FMPLL0 fail**<br>0 RESET is asserted on a FMPLL0 fail event if the reset is enabled<br>1 RESET is not asserted on a FMPLL0 fail event |
| BE_SOFT | **Bidirectional Reset Enable for soft reset**<br>0 RESET is asserted on a soft reset event if the reset is enabled<br>1 RESET is not asserted on a soft reset event |

Table 37-11. Functional Bidirectional Reset Enable Register (RGM_FBRE) Field Descriptions (continued)

| Field | Description |
|---|---|
| BE_CORE | **Bidirectional Reset Enable for CPU reset**<br>0 RESET is asserted on a CPU reset event if the reset is enabled<br>1 RESET is not asserted on a CPU reset event |
| BE_JTAG | **Bidirectional Reset Enable for JTAG initiated reset**<br>0 RESET is asserted on a JTAG initiated reset event if the reset is enabled<br>1 RESET is not asserted on a JTAG initiated reset event |

## 37.4 Functional description

### 37.4.1 Reset state machine

The main role of MC_RGM is the generation of the reset sequence which ensures that the correct parts of the device are reset based on the reset source event. This is summarized in Table 37-12.

Table 37-12. MC_RGM reset implications

| Source | What gets reset | External reset assertion | Boot mode capture |
|---|---|---|---|
| power-on reset | all | yes | yes |
| 'destructive' resets | all except MC_RGM and RTC/API | yes | yes |
| external reset | all (unless an alternate request is enabled) except MC_RGM, RTC/API and Nexus/TAP | yes | yes |
| 'functional' resets | all (unless an alternate request is enabled) except MC_RGM, RTC/API and Nexus/TAP | programmable[1] | programmable[2] |
| shortened 'functional' resets[3] | all except MC_RGM, RTC/API, Nexus/TAP, CFLASH, and SSCM | programmable[1] | programmable[2] |

1 the assertion of the external reset is controlled via the **RGM_FBRE** register

2 the boot mode is captured if the external reset is asserted

3 the short sequence is enabled via the **RGM_FESS** register

**NOTE**

JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in Figure 37-11.

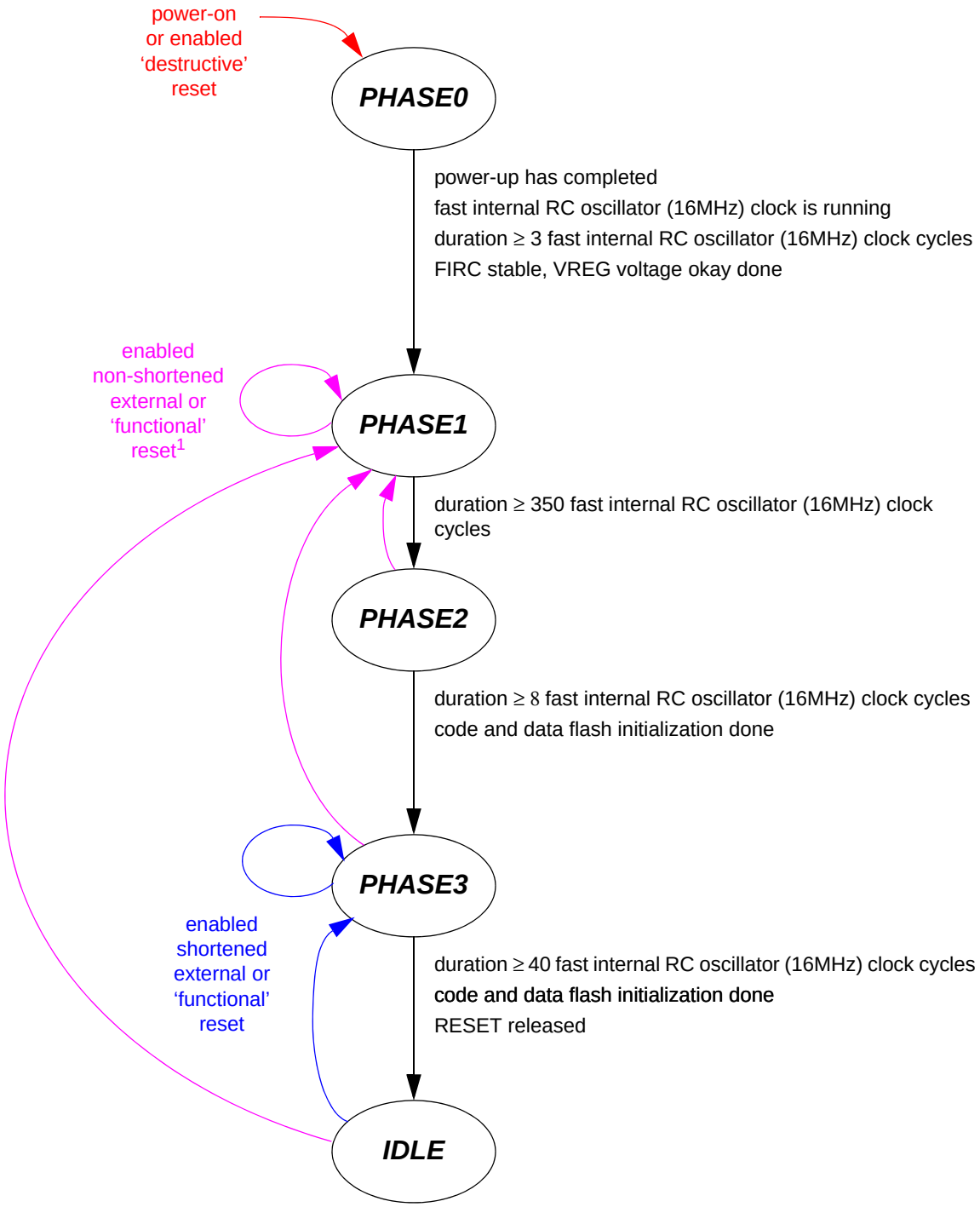



**Figure 37-11. MC_RGM State Machine**

### *37.4.1.1 PHASE0* Phase

This phase is entered immediately from any phase on a power-on or enabled 'destructive' reset event. The reset state machine exits ***PHASE0*** and enters ***PHASE1*** on verification of the following:

- power-up has completed
- fast internal RC oscillator (16MHz) clock is running
- all enabled ‘destructive’ resets have been processed
- all processes that need to be done in ***PHASE0*** are completed
  - FIRC stable, VREG voltage okay
- a minimum of 3 fast internal RC oscillator (16MHz) clock cycles have elapsed since power-up completion and the last enabled ‘destructive’ reset event

#### 37.4.1.2 *PHASE1* Phase

This phase is entered either on exit from ***PHASE0*** or immediately from ***PHASE2***, ***PHASE3***, or ***IDLE*** on a non-masked external or ‘functional’ reset event if it has not been configured to trigger a ‘short’ sequence. The reset state machine exits ***PHASE1*** and enters ***PHASE2*** on verification of the following:

- all enabled, non-shortened ‘functional’ resets have been processed
- a minimum of 350 fast internal RC oscillator (16MHz) clock cycles have elapsed since the last enabled external or non-shortened ‘functional’ reset event

#### 37.4.1.3 *PHASE2* Phase

This phase is entered on exit from ***PHASE1***. The reset state machine exits ***PHASE2*** and enters ***PHASE3*** on verification of the following:

- all processes that need to be done in ***PHASE2*** are completed
  - code and data flash initialization
- a minimum of 8 fast internal RC oscillator (16MHz) clock cycles have elapsed since entering ***PHASE2***

#### 37.4.1.4 *PHASE3* Phase

This phase is a entered either on exit from ***PHASE2*** or immediately from ***IDLE*** on an enabled, shortened ‘functional’ reset event. The reset state machine exits ***PHASE3*** and enters ***IDLE*** on verification of the following:

- all processes that need to be done in ***PHASE3*** are completed
  - code and data flash initialization
- a minimum of 40 fast internal RC oscillator (16MHz) clock cycles have elapsed since the last enabled, shortened ‘functional’ reset event

#### 37.4.1.5 *IDLE* Phase

This is the final phase and is entered on exit from ***PHASE3***. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

### 37.4.2 Destructive Resets

A ‘destructive’ reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed.

The status flag associated with a given ‘destructive’ reset event (**RGM_DES.F_*<destructive reset>*** bit) is set when the ‘destructive’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

The ‘destructive’ reset can be optionally disabled by writing bit **RGM_DERD.D_*<destructive reset>***.

**NOTE**

The **RGM_DERD** register can be written only once between two power-on reset events.

The device’s low-voltage detector threshold ensures that, when 1.2V low-voltage detected (power domain #0) is enabled, the supply is sufficient to have the destructive event correctly propagated through the digital logic. Therefore, if a given ‘destructive’ reset is enabled, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. However, if the given ‘destructive’ reset is disabled and the voltage goes below the digital functional threshold, functionality can no longer be ensured, and the reset may or may not be asserted.

An enabled destructive reset will trigger a reset sequence starting from the beginning of ***PHASE0***.

### 37.4.3 External Reset

The MC_RGM manages the external reset coming from RESET. The detection of a falling edge on RESET will start the reset sequence from the beginning of ***PHASE1***.

The status flag associated with the external reset falling edge event (**RGM_FES.F_EXR** bit) is set when the external reset is asserted and the power-on reset is not asserted.

The external reset can optionally be disabled by writing bit **RGM_FERD.D_EXR**.

**NOTE**

The **RGM_FERD** register can be written only once between two power-on reset events.

An enabled external reset will normally trigger a reset sequence starting from the beginning of ***PHASE1***. Nevertheless, the **RGM_FESS** register enables the further configuring of the reset sequence triggered by the external reset. When **RGM_FESS.SS_EXR** is set, the external reset will trigger a reset sequence starting directly from the beginning of ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***. This can be useful especially when an external reset should not reset the flash.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- a power-on reset
- a ‘destructive’ reset event

- an external reset event
- a ‘functional’ reset event configured via the **RGM_FBRE** register to assert the external reset

In this case, the external reset is asserted until the end of ***PHASE3***.

### 37.4.4 Functional Resets

A ‘functional’ reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given ‘functional’ reset event (**RGM_FES.F_*<functional reset>*** bit) is set when the ‘functional’ reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software’s responsibility to determine which reset source is the most critical for the application.

The ‘functional’ reset can be optionally disabled by software writing bit **RGM_FERD.D_*<functional reset>***.

**NOTE**

The **RGM_FERD** register can be written only once between two power-on reset events.

An enabled functional reset will normally trigger a reset sequence starting from the beginning of ***PHASE1***. Nevertheless, the **RGM_FESS** register enables the further configuring of the reset sequence triggered by a functional reset. When **RGM_FESS.SS_*<functional reset>*** is set, the associated ‘functional’ reset will trigger a reset sequence starting directly from the beginning of ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***. This can be useful especially in case a functional reset should not reset the flash module.

### 37.4.5 *STANDBY* Entry Sequence

***STANDBY*** mode can be entered only when the MC_RGM is in ***IDLE***. On ***STANDBY*** entry, the MC_RGM moves to ***PHASE1***. The minimum duration counter in ***PHASE1*** does not start until ***STANDBY*** mode is exited. On entry to ***PHASE1*** due to ***STANDBY*** mode entry, the resets for all power domains except power domain #0 are asserted. During this time, RESET is not asserted as the external reset can act as a wakeup for the device.

There is an option to keep the flash inaccessible and in low-power mode on ***STANDBY*** exit by configuring the ***DRUN*** mode before ***STANDBY*** entry so that the flash is in power-down or low-power mode. If the flash is to be inaccessible, the ***PHASE2*** and ***PHASE3*** states do not wait for the flash to complete initialization before exiting, and the reset to the flash remains asserted.

See the MC_ME chapter for details on the ***STANDBY*** and ***DRUN*** modes.

### 37.4.6 Alternate Event Generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for each reset

source event (except the power-on reset event) to be converted from a reset to either a ***SAFE*** mode request issued to the MC_ME or to an interrupt request issued to the CPU.

Alternate event selection for a given reset source is made via the **RGM_F/DERD** and **RGM_F/DEAR** registers as shown in Table 37-13.

**Table 37-13. MC_RGM Alternate Event Selection**

| RGM_F/DERD Bit Value | RGM_F/DEAR Bit Value | Generated Event |
|---|---|---|
| 0 | X | reset |
| 1 | 0 | ***SAFE*** mode request |
| 1 | 1 | interrupt request |

The alternate event is cleared by deasserting the source of the request (i.e. at the reset source that caused the alternate request) and also clearing the appropriate **RGM_F/DES** status bit.

**NOTE**

Alternate requests (***SAFE*** mode as well as interrupt requests) are generated asynchronously.

**NOTE**

If a masked ‘destructive’ reset event which is configured to generate a ***SAFE*** mode/interrupt request occurs during ***PHASE0***, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME. The same is true for masked ‘functional’ reset events during ***PHASE1***.

## 37.4.7 Boot Mode Capturing

The MC_RGM provides sampling of the boot mode PAD[22:21] for use by the system to determine the boot mode. This sampling is done five fast internal RC oscillator (16MHz) clock cycles before the rising edge of RESET. The result of the sampling is then provided to the system. For each bit, a value of ‘1’ is produced only if each of the oldest three of the five samples have the value ‘1’, otherwise a value of ‘0’ is produced.

**NOTE**

In order to ensure that the boot mode is correctly captured, the application needs to apply the valid boot mode value to the device at least five fast internal RC oscillator (16MHz) clock periods before the external reset deassertion crosses the $V_{IH}$ threshold.

**NOTE**

RESET can be low as a consequence of the internal reset generation. This will force re-sampling of the boot mode pins.

# Chapter 38
# Run-Length Encoding Decoder (RLE_DEC)

## 38.1 Introduction

The RLE_DEC module is used to decode data that has been compressed using a Run Length Encoding (RLE) scheme. It has input and output FIFO buffers directly connected to the crossbar switch and requires the CPU or DMA to push in the encoded data and then extract the decoded result. The module configuration is optimized for decoding data stored in a two-dimensional image format but can also be used to extract data stored as a linear array.

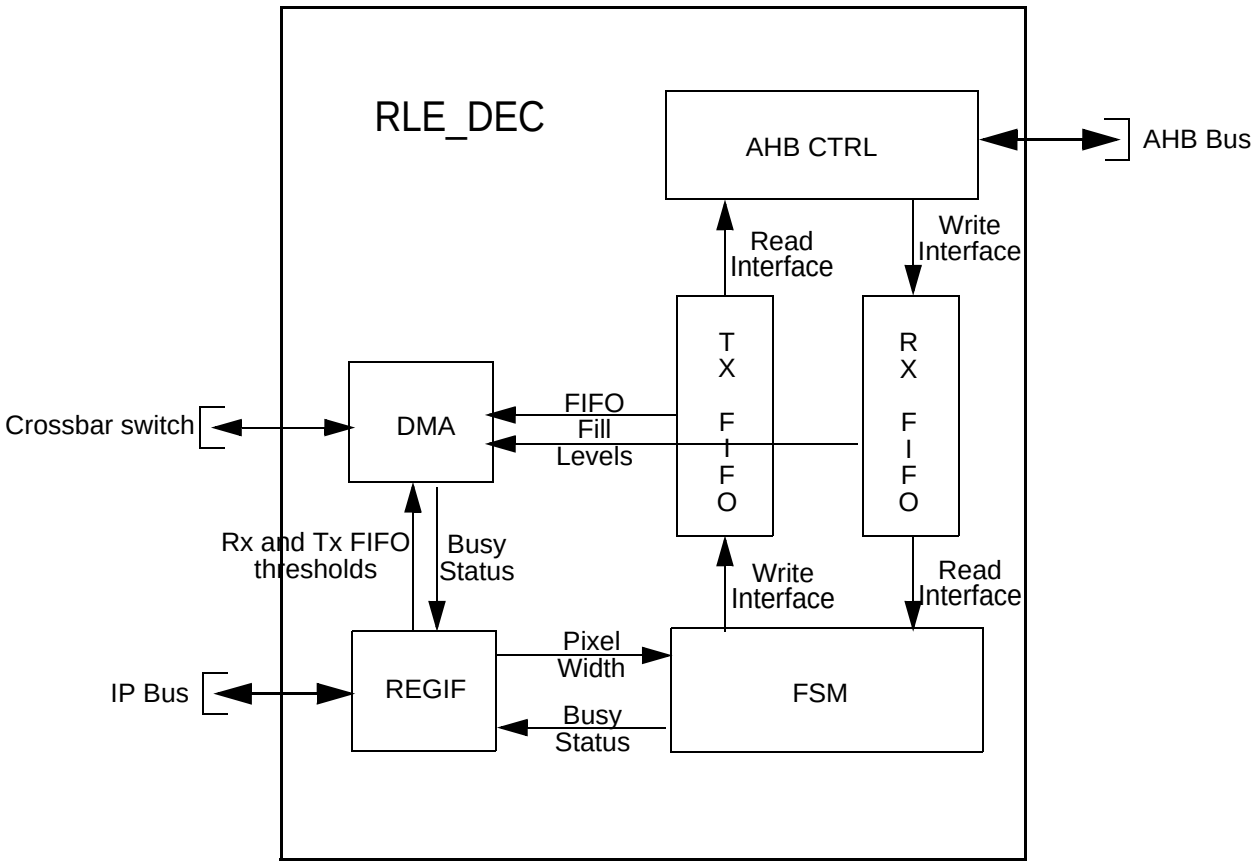



**Figure 38-1. RLE_DEC block diagram**

### 38.1.1 Overview

Figure 38-1 is a block diagram of the Run-Length Encoding Decoder (RLE_DEC) module. The module has two independent interfaces that are memory mapped into the device:

- Crossbar switch interface for moving data into and out of the module

- IP bus interface for configuring the module

The crossbar interface appears in the device memory map as two FIFOs. Writes to the Rx FIFO will cause the RLE_DEC to begin decoding operations. The RLE_DECs Finite State Machine (FSM) removes data from the Rx FIFO, decodes it and places the output into the Tx FIFO. The decoded data appears in the Tx FIFO and is removed from this FIFO after it is read by a crossbar master such as the CPU or eDMA. The DMA operations require the use of two different DMA channels.

The module is configured and its status monitored using registers on the IP bus that appear as locations in the module register memory area. This configuration includes information on the size of the image (or data set) and the size of the individual pixels (or data elements) in the image. The RLE_DEC uses this information to determine when an operation is complete For this reason the RLE_DEC will typically be reconfigured for each operation since the size of each compressed image will be different.

The module can raise interrupts to indicate errors and when an operation is complete.

## 38.1.2 Features

The RLE_DEC supports the following features:

- Lossless decompression
- Pixel formats supported: 8bpp, 16bpp, 24bpp and 32bpp (Programmable)
- AHB mapped Rx FIFO (8x8 bytes deep) with DMA support.
- AHB mapped Tx FIFO (8x8 bytes deep) with DMA support.
- Programmable fill levels of read and write buffers for initiating burst transfers.
- Partial Image Decode feature, wherein only a portion of the decoded image is given as output. (See Section 38.5.3, “Image coordinates’ example,” for details).
- Support for Stop Mode for power-saving purposes

## 38.1.3 RLE_DEC modes of operation

### 38.1.3.1 Normal Mode

In this mode, the RLE_DEC block performs the normal ‘Run Length Encoding’ Decode operation. Further details about this mode of operation can be found in chapter 38.5.5, “Normal mode”.

### 38.1.3.2 Module Disable Mode

The Module Disable Mode is used for power management of the device containing the RLE_DEC module, it is controlled by signals external to the RLE_DEC. The clock to the non-memory mapped logic in the RLE_DEC can be stopped while in the Module Disable Mode. See chapter 38.5.6.1, “Module Disable Mode”.

#### 38.1.3.3 Stop Mode

Stop Mode is used for power management by the system mode management. When a request is made to enter Stop Mode, the RLE_DEC block completes the action currently processed. Then the request is acknowledged.

## 38.2 External signal description

RLE_DEC has no external signals.

## 38.3 Interrupt and DMA request signals

The interrupt and DMA request lines of the RLE_DEC module are mapped to the internal flags in the DEC_RLE_ISR register.

## 38.4 Memory map and register definition

### 38.4.1 Memory map

Table 38-1 shows the RLE_DEC memory map.

**Table 38-1. RLE_DEC memory map**

| Address offset | Register | Location |
|---|---|---|
| 0x00 | Module Configuration Register (RLE_DEC_MCR) | on page 1326 |
| 0x04 | Image Configuration Register (RLE_DEC_ICR) | on page 1327 |
| 0x08 | Compressed Image Size Register (RLE_DEC_CISR) | on page 1328 |
| 0x0C | Decompressed Image Coordinates register (RLE_DEC_DICR) | on page 1328 |
| 0x10 | Status Register (RLE_DEC_SR) | on page 1329 |
| 0x14 | Interrupt Request Status Register (RLE_DEC_ISR) | on page 1329 |
| 0x18 | Interrupt Request Enable Register (RLE_DEC_RIER) | on page 1330 |
| 0x1C | Start Pixel Coordinate Register of Image (RLE_DEC_SPCR) | on page 1331 |
| 0x20 | End Pixel Coordinate Register of Image (RLE_DEC_EPCR) | on page 1332 |

### 38.4.2 AMBA Bus Register Memory Map

Table 38-2 shows the Rx and Tx FIFOs of RLE_DEC mapped to Crossbar switch (AHB) interface.

**Table 38-2. RLE_DEC AMBA Bus Memory Map**

| Address | Register Name |
|---|---|
| 0x9000_4000<br>to<br>0x9000_403F | Rx FIFO address range:- The AHB master can read from and write into these addresses (For Debug purposes only). The FIFO operation is performed only when the access is made to the below mentioned Memory Mapped Area for Rx FIFO. |
| 0x9000_4040<br>to<br>0x9000_407F | Tx FIFO address range:- The AHB master can read from and write into these addresses (For Debug purposes only). The FIFO operation is performed only when the access is made to the below mentioned Memory Mapped Area for Tx FIFO. |
| 0x9000_4080<br>to<br>0x9000_40BF | Memory Mapped Area for Rx FIFO. AHB can only Write into this address. A write into this address range results in a write to the Rx FIFO wherever the write pointer points. If Rx FIFO is full, AHB write request will result in HREADY low, unless space is there for the write operation.<br>(Please Note: The write(burst or single write) should start from address 0x9000_4080 only.) |
| 0x9000_40C0<br>to<br>0x9000_40FF | Memory Mapped Area for Tx FIFO. AHB can only Read from this address. A read from this address range results in a read from the top of the Tx FIFO. If Tx FIFO is empty, AHB Read request will result in HREADY low, unless some data is available for reading.<br>(Please Note: The read(burst or single read) should start from address 0x9000_40C0 only.) |

## 38.4.3 Register descriptions

### 38.4.3.1 Module Configuration Register (RLE_DEC_MCR)

The RLE_DEC_MCR holds configuration data associated with RLE_DEC operation. This register can be accessed with 8-bit, 16-bit and 32-bit wide operations.

Address: RLE_DEC_BASE + 0x000 Write:Anytime

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX_FIFO_THRESHOLD | | | | | | | | TX_FIFO_THRESHOLD | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | TXFFEN | 0 | MDIS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 38-2. Module Configuration Register (RLE_DEC_MCR)**

**Table 38-3. RLE_DEC_MCR Field Descriptions**

| Field | Description |
|---|---|
| RX_FIFO_THRESHOLD | Rx FIFO threshold: This field determines how much space should be available for writing into the Rx Buffer until the write action is triggered (through DMA). When the number of bytes of available space in RX FIFO exceeds the number given by this field the RLE_DEC_FR[RFFR] flag is asserted. |
| TX_FIFO_THRESHOLD | Tx FIFO threshold: This field determines how many entries must be read into the TX FIFO until the readout action is triggered. When the number of valid entries in the TX FIFO exceeds the number given by this field the RLE_DEC_FR[TFDR] flag is asserted. |
| TXFFEN | Tx FIFO Flush Enable. This bit configures the operation of the RLE Decoder in the case when the image decoding is complete and the data remaining in the Tx FIFO is below the value of TX_FIFO_THRESHOLD.<br>0 Trigger the DMA action only when TxFIFO has more data than TX_FIFO_THRESHOLD.<br>1 Trigger the DMA action until the Tx FIFO is empty even if it is less than TX_FIFO_THRESHOLD. |
| MDIS | Module Disable. The MDIS bit allows the clock to the non-memory mapped logic in the RLE_DEC to be stopped, putting the RLE_DEC in a software controlled power-saving state. See Section 38.5.6.1, "Module Disable Mode" for more information.<br>0 Enable RLE_DEC clocks.<br>1 Allow external logic to disable RLE_DEC clocks. |

### 38.4.3.2 Image Configuration Register (RLE_DEC_ICR)

The RLE_DEC_ICR holds configuration data associated with configuration of image parameters. This register can be accessed with 8-bit, 16-bit and 32-bit wide operations.

Address: RLE_DEC_BASE + 0x004 Write:Anytime

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

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

**Figure 38-3. Image Configuration Register (RLE_DEC_ICR)**

**Table 38-4. RLE_DEC_ICR field descriptions**

| Field | Description |
|---|---|
| WIDTH | 00 -- Pixel Width is 08-bits; 01 -- Pixel Width is 16-bits;<br>10 -- Pixel Width is 24-bits; 11 -- Pixel Width is 32-bits. |

### 38.4.3.3 Compressed Image Size Register (RLE_DEC_CISR)

The RLE_DEC_CISR holds the size of the compressed image. This register can be accessed with 8-bit, 16-bit and 32-bit wide operations.

Address: RLE_DEC_BASE + 0x008 Write:Anytime

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

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

**Figure 38-4. Compressed Image Size Register (RLE_DEC_CISR)**

**Table 38-5. RLE_DEC_CISR field descriptions**

| Field | Description |
|---|---|
| SIZE | This is the byte-size of compressed image, that will be given as an input to the RLE_DEC. This should be programmed before the Module is enabled using MDIS. |

### 38.4.3.4 Decompressed Image Coordinates Register (RLE_DEC_DICR)

The RLE_DEC_DICR holds the final coordinates of the decompressed image. Decompression is done until this pixel reached. Please refer to section “Image Coordinates’ Description”. This register can be accessed with 8-bit, 16-bit and 32-bit wide operations.

Address: RLE_DEC_BASE + 0x00C Write:Anytime

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

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

**Figure 38-5. Decompressed Image Coordinates Register (RLE_DEC_DICR)**

**Table 38-6. RLE_DEC_DICR field descriptions**

| Field | Description |
|---|---|
| X | This is the X Coordinate of the final pixel of the decompressed image, that will be given as the output of the RLE DECODER. This should be programmed before the Module is enabled using MDIS. This field has a minimum value of 1. |
| Y | This is the Y Coordinate of the final pixel of the decompressed image, that will be given as the output of the RLE DECODER. This should be programmed before the Module is enabled using MDIS. This field has a minimum value of 1. |

### 38.4.3.5 Status Register (RLE_DEC_SR)

The RLE_DEC_SR register provides the status information about the TX FIFO and RX FIFO.

Address: RLE_DEC_BASE + 0x0010

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RXFREE | | | | | | | | TXFILL | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 38-6. Status Register (RLE_DEC_SR)**

**Table 38-7. RLE_DEC_SR Field Descriptions**

| Field | Description |
|---|---|
| RXFREE | This status gives the amount of free space in bytes, available in Rx FIFO to write. |
| TXFILL | This status gives the amount of data in bytes, available in Tx FIFO to read. |

### 38.4.3.6 Interrupt Request Status Register (RLE_DEC_ISR)

The RLE_DEC_ISR register is the Interrupt Status Register for RLE DECODER. The interrupts are asserted upon associated events. They are deasserted upon write to this register.

Address: RLE_DEC_BASE + 0x0014 Reset upon write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RXDIF | TXDIF | RXUIF | TXUIF | RXFIF | TXFIF | RXEIF | TXEIF |
| W | | | | | | | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 38-7. Interrupt Request Status Register (RLE_DEC_ISR)**

**Table 38-8. RLE_DEC_ISR field descriptions**

| Field | Description |
|---|---|
| RXDIF | Rx Image Done. Is asserted when the amount of received data has equalled the programmed “Compressed Image Size Register” |
| TXDIF | Tx Image Done. Is asserted when coordinates of decompressed image cross End Pixel Coordinates which are defined in “End Pixel Coordinate Register” |
| RXUIF | Rx FIFO has space. Asserted when Rx FIFO has more space than Rx FIFO Threshold. (DMA request for input data.) |
| TXUIF | Tx FIFO has data. Asserted when Tx FIFO has more data than Tx FIFO Threshold. (DMA request for output data.) |
| RXFIF | Rx FIFO Full: Asserted when Rx FIFO has no more space left for data and a write was attempted on it. |
| TXFIF | Tx FIFO Full: Asserted when Tx FIFO has no more space left for data and a write was attempted on it. |
| RXEIF | Rx FIFO Empty: Asserted when Rx FIFO has no data and a read was attempted from it. |
| TXEIF | Tx FIFO Empty: Asserted when Tx FIFO has no data and a read was attempted from it. |

### 38.4.3.7 Interrupt Request Enable Register (RLE_DEC_RIER)

The RLE_DEC_RIER register provides enables for the interrupts in the RLE_DEC module.

Address: RLE_DEC_BASE + 0x0018 Write: Anytime

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RXDIE | TXDIE | RXUIE | TXUIE | RXFIE | TXFIE | RXEIE | TXEIE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 38-8. Interrupt Request Enable Register (RLE_DEC_RIER)**

**Table 38-9. RLE_DEC_RIER field descriptions**

| Field | Description |
|---|---|
| RXDIE | Interrupt Enable for Rx Image Done Interrupt |
| TXDIE | Interrupt Enable for Tx Image Done Interrupt |
| RXUIE | Interrupt Enable for Rx FIFO has space Interrupt |
| TXUIE | Interrupt Enable for Tx FIFO has data Interrupt |
| RXFIE | Interrupt Enable for Rx FIFO full Interrupt |
| TXFIE | Interrupt Enable for Tx FIFO full Interrupt |
| RXEIE | Interrupt Enable for Rx FIFO empty Interrupt |
| TXEIE | Interrupt Enable for Tx FIFO empty Interrupt |

### 38.4.3.8 Start Pixel Coordinate Register of Image (RLE_DEC_SPCR)

The RLE_DEC_SPCR holds the start coordinates of the decompressed image. This register can be accessed with 8-bit, 16-bit and 32-bit wide operations.

Address: RLE_DEC_BASE + 0x001C Write:Anytime

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | X | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Y | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 38-9. Start Pixel Coordinate Register of Image (RLE_DEC_SPCR)**

**Table 38-10. RLE_DEC_SPCR Field Descriptions**

| Field | Description |
|---|---|
| X | This is the X Coordinate of the first pixel of the decompressed image, that will be given as the output of the RLE DECODER. See Section 38.5.3, “Image coordinates’ example”. This should be programmed before the Module is enabled using MDIS. This field has a minimum value of 1. |
| Y | This is the Y Coordinate of the final pixel of the decompressed image, that will be given as the output of the RLE DECODER. See Section 38.5.3, “Image coordinates’ example”. This should be programmed before the Module is enabled using MDIS. This field has a minimum value of 1. |

### 38.4.3.9 End Pixel Coordinate Register of Image (RLE_DEC_EPCR)

The RLE_DEC_SPCR holds the end coordinates of the decompressed image of interest. See Section 38.5.3, “Image coordinates’ example”. This register can be accessed with 8-bit, 16-bit and 32-bit wide operations.

Address: RLE_DEC_BASE + 0x0020 Write:Anytime

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | X | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Y | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 38-10. End Pixel Coordinate Register of Image (RLE_DEC_EPCR)**

Table 38-11. RLE_DEC_EPCR Field Descriptions

| Field | Description |
|---|---|
| X | This is the X Coordinate of the last pixel of the decompressed image of interset, that will be given as the output of the RLE DECODER. See Section 38.5.3, “Image coordinates’ example”. This should be programmed before the Module is enabled using MDIS. This field has a minimum value of 1. |
| Y | This is the Y Coordinate of the last pixel of the decompressed image of interset, that will be given as the output of the RLE DECODER. See Section 38.5.3, “Image coordinates’ example”. This should be programmed before the Module is enabled using MDIS. This field has a minimum value of 1. |

## 38.4.4 Crossbar switch memory map descriptions

### 38.4.4.1 Rx FIFO address range

The size of the Rx FIFO is 8-bytes x 8-bytes. The Rx FIFO address range is 0x9000_4000 to 0x9000_4038. A Crossbar switch can read from and write into these addresses (For Debug purposes only). The FIFO operation is performed only when the access is made to the below mentioned Memory Mapped Area for Rx FIFO.

### 38.4.4.2 Tx FIFO address range

The size of the Tx FIFO is 8-bytes x 8-bytes. The Tx FIFO address range is 0x9000_4040 to 0x9000_4078. A Crossbar switch can read from and write into these addresses (For Debug purposes only). The FIFO operation is performed only when the access is made to the below mentioned Memory Mapped Area for Tx FIFO.

### 38.4.4.3 Memory mapped Rx FIFO

The contents of the Rx FIFO are mapped to 0x9000_4080. Writing RLE compressed data into the Rx FIFO increments the write-pointer based on the size of the written data.

The RLE decode process can be automated by using the eDMA. Each time the free space of Rx FIFO exceeds a certain level (programmable), a (Rx FIFO) DMA request is asserted. When enabled the eDMA can be used to write RLE compressed data into the Rx FIFO.

A read request to Rx FIFO results in an error.

### 38.4.4.4 Memory mapped Tx FIFO

The contents of the Tx FIFO are mapped to 0x9000_40C0. The RLE_DEC module writes decompressed data into the Tx FIFO.

The RLE decode process can be automated by using the eDMA. Each time the contents of the Tx FIFO exceeds a certain fill-level (programmable), a (Tx FIFO) DMA request is asserted. When enabled the eDMA can be used to read the decompressed data from the Tx FIFO.

A write request to Tx FIFO results in an error.

## 38.5 Functional description

### 38.5.1 RLE encoding format

The expected encoding is as follows:

- The first data byte in the encoded image is a command byte (CMD[7:0])
- The ms bit (CMD[7]) indicates if the following bytes are raw or compressed pixels. One pixel can be 8-bit, 16-bit, 24-bit or 32-bit wide, depending on the PIXEL_WIDTH configuration.
- The remaining 7 command bits (CMD[6:0]), specify the number of raw or compressed pixels that follow the command byte. The count is offset by 1 such that value of 0 means one pixel follows.
- For compressed pixels (CMD[7] = 1), only one pixel follows the command byte. This pixel is repeated count+1 times. A new command follows after CMD+(1*{Pixel width}) pixels.
- For raw pixels (CMD[7] = 0), count+1 number of pixels follow the command byte and these are passed to the Tx FIFO as is.
- If there is more data to decode then a new command follows after CMD+({count+1}*{Pixel width}) bytes. This encoding continues until the whole image is decoded

### 38.5.2 RLE decoding process

The recommended process to decode the data is as follows:

- The RLE_DEC module is disabled by default (MDIS=1).
- Before enabling the device, program the compressed image size register (RLE_DEC_CISR) and decompressed image size register (RLE_DEC_DICR) for the image. The decoder expects to read compressed-image-size amount of bytes from the Rx FIFO and expects to decode and write decompressed-image-size amount of bytes into the Tx FIFO.
- When less than the full image is to be decoded then configure the Start Pixel Coordinates Register (RLE_DEC_SPCR) and the End Pixel Coordinates Register (RLE_DEC_EPCR).
- Configure the WIDTH of the pixels in the image (RLE_DEC_ICR).
- Enable the RLE_DEC DMA Rx and Tx channels in the DMAMUX module and connect these to channels in the eDMA. Configure the selected eDMA channels such that the compressed image is written into the Rx FIFO an the decompressed image is copy from the Tx FIFO.
- Enable the RLE_DEC by setting MDIS=0.
- When the decoding is complete the TXDIF flag is set (RLE_DEC_ISR register). This indicates that the RLE process is complete but depending on the size of the image and the value of the RLE_DEC_MCR[TXFFEN] bit there may be some decoded pixels that have not been copied from the Tx FIFO because its threshold has not been reached.
- If the TXFFEN bit is not set to automatically flush the Tx FIFO then trigger an eDMA transfer under software control until the Tx FIFO is empty (TXFILL=0 in RLE_DEC_SR register).

- Set MDIS=1 to disable the module. The module needs to be disabled at the end of image decompression and enabled again only after the parameters of the new image have been programmed.

### 38.5.3 Image coordinates' example

Figure 38-11 shows the definition of the coordinates used in RLE_DEC to define the decompressed image.

Let the decompressed image be of size 10-pixels x 10-pixels. The start coordinate is the coordinate at the top-left corner, (1,1), i.e. X=1 and Y=1. The last coordinate is bottom-right coordinate, (10,10) in this case. So, we see that the image size can be described by the bottom-right coordinate of the image. This pixel coordinate is described in the register "Decompressed Image Coordinate Register".

If only a part of the complete image is to be given as output, then "Start Pixel Coordinate Register" and "End Pixel Coordinate Register" should be programmed appropriately. For example, in the figure shown below, if the gray area is the image of interest and it is only required to output only that part of the decompressed image, then the "Start Pixel Coordinate Register" should be programmed as (X=3, Y=4) and "End Pixel Coordinate Register" should be programmed as (X=7, Y=7).

If however, the complete decompressed image is to be output, then the "Start Pixel Coordinate Register" should be programmed as (X=1, Y=1) and "End Pixel Coordinate Register" should be programmed as (X=10, Y=10), i.e. equal to "Decompressed Image Coordinate Register".

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| (1,1) | (2,1) | (3,1) | (4,1) | (5,1) | (6,1) | (7,1) | (8,1) | (9,1) | (10,1) |
| (1,2) | | | | | | | | | |
| (1,3) | | | | | | | | | |
| (1,4) | | (3,4) | | | | | | | |
| (1,5) | | | | | | | | | |
| (1,6) | | | | | | | | | |
| (1,7) | | | | | | (7,7) | | | |
| (1,8) | | | | | | | | | |
| (1,9) | | | | | | | | | |
| (1,10) | | | | | | | | | (10,10) |

**Figure 38-11. Decompressed 10x10 pixel image**

### 38.5.4 Modes of operation

The possible operational modes of the RLE_DEC block are:

- Normal Mode: This is used for normal RLE decompression of data received from AHB interface. The module enters this mode by deasserting RLE_DEC_MCR[MDIS].
- Stop Mode: The mode used by the system when changing modes if the RLE is no longer required. When the system requests Stop Mode, the RLE_DEC block completes the execution of the current command and acknowledges the request. The system then removes clocks to the RLE_DEC block.

- Module Disable Mode: The mode is used for power management. The clock to the non-memory mapped logic in the RLE_DEC can be stopped while in Module Disable Mode.The module enters the mode by setting RLE_DEC_MCR[MDIS].

### 38.5.5 Normal mode

In normal mode, decompression of data is performed using RLE scheme. The module enters this mode by deasserting RLE_DEC_MCR[MDIS].

Whenever the Rx FIFO has some data, the state machine is triggered. The decoding operation starts according to the RLE decoding scheme described above. Whenever Tx FIFO is not full and Rx FIFO has got data to transmit, the decoding is done.

After completing the decode of one image, the Normal mode should be exited. For decoding the data of next image, the re-entry to Normal mode should be done only after the new image parameters have been programmed.

### 38.5.6 Power-saving features

The RLE_DEC supports two power-saving strategies:

- Stop Mode
- Module Disable Mode

#### 38.5.6.1 Module Disable Mode

The RLE_DEC block is in Module Disable Mode by default. It is exited to Normal-mode by de-asserting the MDIS bit in RLE_DEC_MCR register.

Host software can initiate the Module Disable Mode by writing a ‘1’ to the MDIS bit. When a request is encountered to enter the Module Disable Mode the RLE_DEC negates clock-enable when it is ready to enter the Module Disable Mode.

If implemented, the Clock-enable signal can stop the clock to the non-memory mapped logic. When clock-enable is negated, the RLE_DEC is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the RLE_DEC is in the Module Disable Mode. DMA request signals cannot be cleared while in the Module Disable Mode.

Note that issuing a new AHB request is illegal in Normal Mode during the time starting with raising the request to enter Module Disable Mode and ending with leaving the Module Disable Mode.

# Chapter 39
# Sound Generator Module (SGM)

## 39.1 Introduction

The SGM is a 4-channel Sound Generator supporting autonomous audio note generation, mono linear PCM data playback, mixing and amplitude control. Output data from the SGM mixer in the form of stereo 16-bit PCM samples can be fed to an external audio DAC using its I2S interface or to an internal PWM for output to an external LPF.

## 39.2 Features

The SGM provides the following features:

- Four independent audio channels with dedicated FIFO buffers
- Four channel audio mixer featuring left and right channel outputs
- Individual channel volume control
  - Volume resolution 8 bits (256 volume settings)
- Input data formats supported : 8, 12, 14 and 16 bit
- Mono 16-bit PWM output to drive external LPF and speaker/buzzer.
- I2S interface
- Choice of operating clocks to allow flexible selection of audio sample rates
- Module disable for power saving when SGM is not in use

The SGM channels support two modes of operation:

- Direct digital synthesis (DDS) mode
  - FIFO acts as a memory buffer containing waveform samples (wavetable)
  - Programmable sample rate
  - Configurable ASR envelope with 8-bit volume control
    - Linear increment (Attack) and decrement (Release) with programmable amplitude
    - Exponential increment (attack) and decrement (Release) with programmable amplitude
    - Programmable sustain time
  - Programmable number of Note pulses 1 to 65536 with unlimited repetition option
  - Automatic inter-note time control
  - Sound start/stop function to start or stop sound generation immediately
  - Buffered configuration registers
- Wave mode
  - Input FIFO for each channel to buffer incoming Wave file audio sample stream.
    - DMA request associated with each channel.
    - Programmable FIFO watermark

- Automatic Wave Duration Control
- Automatic Dead-Time Control
- Repeat mode with programmable repeat number

- Ability for each channel to operate in Wave mode or DDS mode independently of other channels
  - Synchronisation option for multiple channels in Wave mode

Interrupts are supported as follows:

- DDS mode:
  - At the end of autonomous increment/decrement
  - At the end of programmed sequence of Note pulses
  - On reaching configurable amplitude level during autonomous increment/decrement
- Wave mode:
  - FIFO full / empty / watermark

## 39.3 Device-specific configuration

The I2S and PWM outputs are multiplexed together on this device, and the active output is determined by the SGMCTL[PWME] bit. See Chapter 3, “Signal Description,” for details for pin assignments.

## 39.4 Block diagram

A block diagram of the SGM module is shown in Figure 39-1.

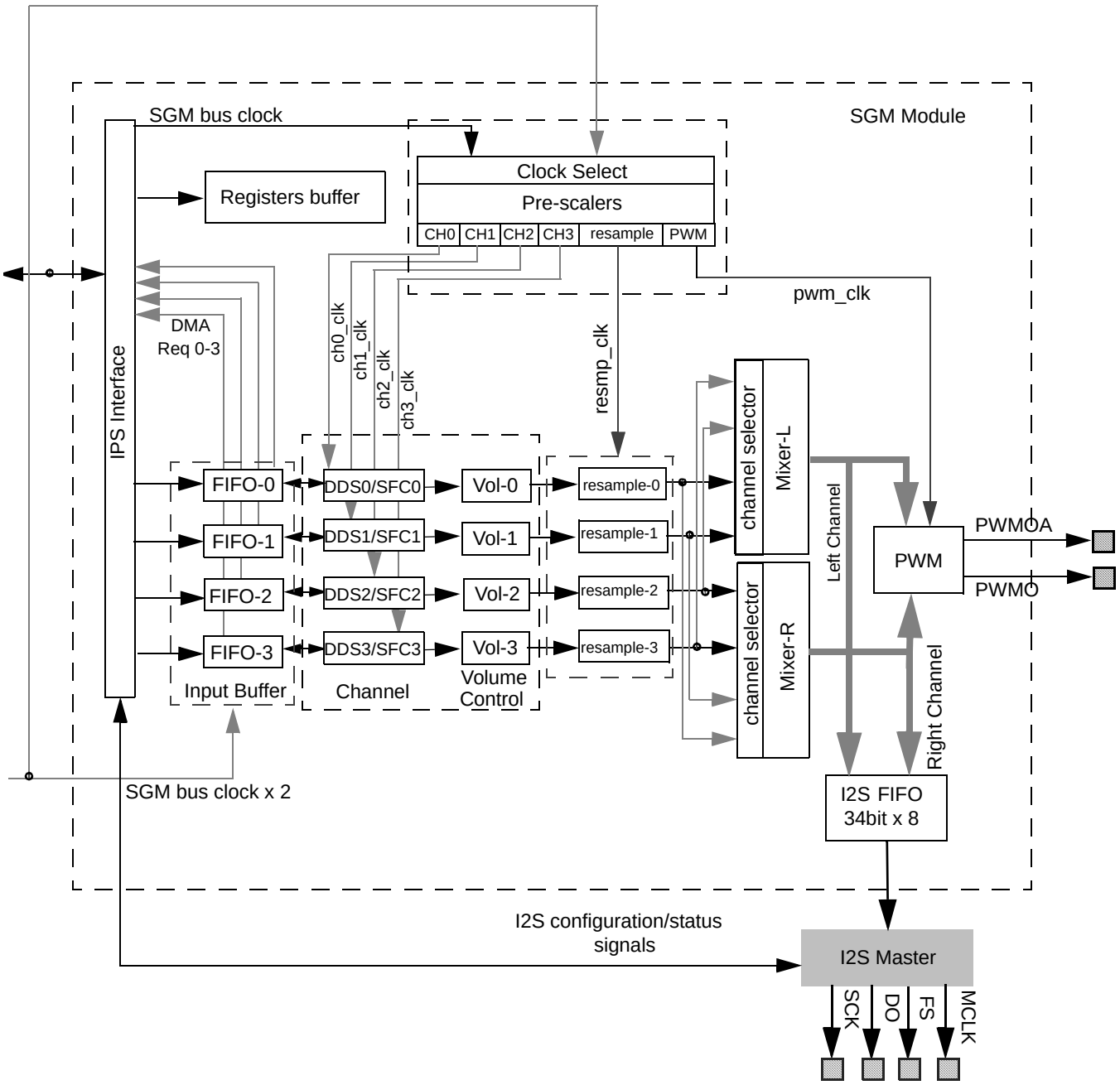



**Figure 39-1. High Level Block Diagram of SGM**

## 39.5 External signal description

**Table 39-1. SGM external interface signals**

| Signal | Description | I/O | Type | Voltage Range | width |
|---|---|---|---|---|---|
| PWM External Signals | | | | | |
| PWMO | Either PWM Block sample rate clock or composite signal that can be filtered, amplified and fed to a buzzer or loudspeaker (SGMCTL.OS = 1). | O | logic | [0,VDD] | 1 |
| PWMOA | PWM Block pwm signal output signal. | O | logic | [0,VDD] | 1 |
| I2S External signals | | | | | |
| SCK | Clock out (audio data driven on this clock) | O | logic | [0,VDD] | 1 |
| DO | Audio serial Data Out | O | logic | [0,VDD] | 1 |
| FS | Frame Sync | O | logic | [0,VDD] | 1 |
| MCLK | MCLK (Auxiliary Clock) output enable | O | logic | [0,VDD] | 1 |

## 39.6 Memory map and register definition

This section describes the SGM memory map and register definition.

### 39.6.1 Memory map

The SGM module contains a set of control and status registers located between SGM register base + 0x0000 and 0x00FC (TBD).

**Table 39-2. SGM memory map**

| Address | Register | Width | Access | Location |
|---|---|---|---|---|
| $BASE+0x0000 | SGMCTL | 32 | Read/Write | on page 1342 |
| $BASE+0x0004 | SGMCFG | 32 | Read/Write | on page 1344 |
| $BASE+0x0008 | CLKRSP | 32 | Read/Write | on page 1347 |
| $BASE+0x000C | CLKCH3 | 32 | Read/Write | on page 1348 |
| $BASE+0x0010 | DDSCH3 | 32 | Read/Write | on page 1348 |
| $BASE+0x0014 | ECRACH3 | 32 | Read/Write | on page 1349 |
| $BASE+0x0018 | ECRRCH3 | 32 | Read/Write | on page 1350 |
| $BASE+0x001C | ECRSCH3 | 32 | Read/Write | on page 1351 |
| $BASE+0x0020 | NTCH3 | 32 | Read/Write | on page 1352 |
| $BASE+0x0024 | TPCCH3 | 32 | Read/Write | on page 1352 |
| $BASE+0x0028 | PTCCH3 | 32 | Read/Write | on page 1353 |

Table 39-2. SGM memory map (continued)

| Address | Register | Width | Access | Location |
|---|---|---|---|---|
| $BASE+0x002C | DTCCH3 | 32 | Read/Write | on page 1354 |
| $BASE+0x0030 | RNCCH3 | 32 | Read/Write | on page 1354 |
| $BASE+0x0034 | CLKCH2 | 32 | Read/Write | on page 1348 |
| $BASE+0x0038 | DDSCH2 | 32 | Read/Write | on page 1348 |
| $BASE+0x003C | ECRACH2 | 32 | Read/Write | on page 1349 |
| $BASE+0x0040 | ECRRCH2 | 32 | Read/Write | on page 1350 |
| $BASE+0x0044 | ECRSCH2 | 32 | Read/Write | on page 1351 |
| $BASE+0x0048 | NTCH2 | 32 | Read/Write | on page 1352 |
| $BASE+0x004C | TPCCH2 | 32 | Read/Write | on page 1352 |
| $BASE+0x0050 | PTCCH2 | 32 | Read/Write | on page 1353 |
| $BASE+0x0054 | DTCCH2 | 32 | Read/Write | on page 1354 |
| $BASE+0x0058 | RNCCH2 | 32 | Read/Write | on page 1354 |
| $BASE+0x005C | CLKCH1 | 32 | Read/Write | on page 1348 |
| $BASE+0x0060 | DDSCH1 | 32 | Read/Write | on page 1348 |
| $BASE+0x0064 | ECRACH1 | 32 | Read/Write | on page 1349 |
| $BASE+0x0068 | ECRRCH1 | 32 | Read/Write | on page 1350 |
| $BASE+0x006C | ECRSCH1 | 32 | Read/Write | on page 1351 |
| $BASE+0x0070 | NTCH1 | 32 | Read/Write | on page 1352 |
| $BASE+0x0074 | TPCCH1 | 32 | Read/Write | on page 1352 |
| $BASE+0x0078 | PTCCH1 | 32 | Read/Write | on page 1353 |
| $BASE+0x007C | DTCCH1 | 32 | Read/Write | on page 1354 |
| $BASE+0x0080 | RNCCH1 | 32 | Read/Write | on page 1354 |
| $BASE+0x0084 | CLKCH0 | 32 | Read/Write | on page 1348 |
| $BASE+0x0088 | DDSCH0 | 32 | Read/Write | on page 1348 |
| $BASE+0x008C | ECRACH0 | 32 | Read/Write | on page 1349 |
| $BASE+0x0090 | ECRRCH0 | 32 | Read/Write | on page 1350 |
| $BASE+0x0094 | ECRSCH0 | 32 | Read/Write | on page 1351 |
| $BASE+0x0098 | NTCH0 | 32 | Read/Write | on page 1352 |
| $BASE+0x009C | TPCCH0 | 32 | Read/Write | on page 1352 |
| $BASE+0x00A0 | PTCCH0 | 32 | Read/Write | on page 1353 |
| $BASE+0x00A4 | DTCCH0 | 32 | Read/Write | on page 1354 |
| $BASE+0x00A8 | RNCCH0 | 32 | Read/Write | on page 1354 |
| $BASE+0x00AC | VCRWAV | 32 | Read/Write | on page 1355 |

Table 39-2. SGM memory map (continued)

| Address | Register | Width | Access | Location |
|---|---|---|---|---|
| $BASE+0x00B0 | SGMTOCR | 32 | Read/Write | on page 1356 |
| $BASE+0x00B4 | MIXCR | 32 | Read/Write | on page 1356 |
| $BASE+0x00B8 | CLKPWM | 32 | Read/Write | on page 1357 |
| $BASE+0x00BC | PWMCR | 32 | Read/Write | on page 1358 |
| $BASE+0x00C0 | DFIFO1 | 32 | Write | on page 1358 |
| $BASE+0x00C4 | DFIFO2 | 32 | Write | on page 1359 |
| $BASE+0x00C8 | FIFOWM | 32 | Read/Write | on page 1360 |
| $BASE+0x00CC | FIFORP | 32 | Read/Write | on page 1360 |
| $BASE+0x00D0 | FIFOWP | 32 | Read/Write | on page 1361 |
| $BASE+0x00D4 | SGMST | 32 | Read | on page 1361 |
| $BASE+0x00D8 | SGMICFD | 32 | Read/Write | on page 1363 |
| $BASE+0x00DC | SGMIC | 32 | Read/Write | on page 1365 |
| $BASE+0x00E0 | SGMISFD | 32 | Read/Write | on page 1367 |
| $BASE+0x00E4 | SGMIS | 32 | Read/Write | on page 1368 |
| $BASE+0x00E8 | I2SEN | 32 | Read/Write | on page 1370 |
| $BASE+0x00EC | I2SCTL | 32 | Read/Write | on page 1371 |
| $BASE+0x00F0 | I2SDFC | 32 | Read/Write | on page 1372 |
| $BASE+0x00F4 | I2SPRS | 32 | Read/Write | on page 1374 |
| $BASE+0x00F8 | I2SINTC | 32 | Read/Write | on page 1374 |
| $BASE+0x00FC | I2SST | 32 | Read/Write | on page 1375 |

## 39.6.2 Register descriptions

### 39.6.2.1 SGM Control Register (SGMCTL)

The SGMCTL register controls the function of SGM.

SGM Register Base + 0x0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SOG CH3 | SOG CH2 | SOG CH1 | SOG CH0 | 0 | 0 | 0 | 0 | MDIS | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MOD CH3 | MOD CH2 | MOD CH1 | MOD CH0 | 0 | 0 | 0 | 0 | TOE | WAV CLKS | 0 | 0 | 0 | PWM E | PWM CHS | OS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-2. SGM Control Register (SGMCTL)**

**Table 39-3. SGMCTL Register Description**

| Field | Description |
|---|---|
| 31<br>SOGCH3 | **Start Of Generation of Channel 3.** Control the start/Stop of the Channel 3.<br>0 Stop the channel sound generation.<br>1 Start the channel sound generation. |
| 30<br>SOGCH2 | **Start Of Generation of Channel 2.** Control the start/Stop of the Channel 2.<br>0 Stop the channel sound generation.<br>1 Start the channel sound generation. |
| 29<br>SOGCH1 | **Start Of Generation of Channel 1.** Control the start/Stop of the Channel 1.<br>0 Stop the channel sound generation.<br>1 Start the channel sound generation. |
| 28<br>SOGCH0 | **Start Of Generation of Channel 0.** Control the start/Stop of the Channel 0.<br>0 Stop the channel sound generation.<br>1 Start the channel sound generation. |
| 27-24 | **Reserved.** |
| 23<br>MDIS | **Module Disable.** Force the SGM into a power-down mode by disabling the module clock.<br>0 Clock input switched on.<br>1 Clock input switched off. Force SGM enter low-power.<br>Note: To enable the SGM, this bit needs to be cleared before configuring other registers of SGM. |
| 22-16 | **Reserved.** |
| 15<br>MODCH3 | **Channel 3 mode Selection.**<br>0 Wave mode<br>1 DDS mode |
| 14<br>MODCH2 | **Channel 2 mode Selection.**<br>0 Wave mode<br>1 DDS mode |
| 13<br>MODCH1 | **Channel 1 mode Selection.**<br>0 Wave mode<br>1 DDS mode |
| 12<br>MODCH0 | **Channel 0 mode Selection.**<br>0 Wave mode<br>1 DDS mode |

**Table 39-3. SGMCTL Register Description (continued)**

| Field | Description |
|---|---|
| 11-8 | **Reserved.** |
| 7<br>TOE | **Time Out Enable.**<br>0 Disable Timeout<br>1 Enable Timeout |
| 6<br>WAVCLKS | **Wave mode Clock Selection.** Select the resample clock as channel clock for all Wave Mode channels.<br>0 Use the individual channel clock.<br>1 Select the resample clock as channel clock for all channels which are in Wave mode |
| 5-3 | **Reserved.** |
| 2<br>PWME | **PWM Output Select.** Select the PWM or I2S output.<br>0 Select the I2S output<br>1 Select the PWM output. |
| 1<br>PWMCHS | **PWM Channel Select.** Select the data source for PWM duty cycle.<br>0 Select the output data of Mixer Left as the duty cycle of PWM<br>1 Select the output data of Mixer Right as the duty cycle of PWM |
| 0<br>OS | **SGM Output Selection.** Control separate or mixed “frequency” and “volume” outputs for PWM.<br>0 Separate outputs.<br>1 Mixed outputs |

### 39.6.2.2 SGM Configuration Register (SGMCFG)

The SGMCFG register determines the configuration of SGM.

SGM Register Base + 0x0004

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ATKF CH3 | ATKF CH2 | ATKF CH1 | ATKF CH0 | RELF CH3 | RELF CH2 | RELF CH1 | RELF CH0 | NOP ECH3 | NOP ECH2 | NOP ECH1 | NOP ECH0 | TPCE CH3 | TPCE CH2 | TPCE CH1 | TPCE CH0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RPTE CH3 | RPTE CH2 | RPTE CH1 | RPTE CH0 | 0 | 0 | 0 | 0 | SFCH3 | | SFCH2 | | SFCH1 | | SFCH0 | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-3. SGM Configuration Register (SGMCFG)**

**Table 39-4. SGMCFG Register Description**

| Field | Description |
|---|---|
| 31<br>ATKFCH3 | **Envelope Attack style for Channel 3.** This bit controls the type of interpolation used in the attack phase for channel 3 in DDS mode.<br>0 linear interpolation<br>1 exponential interpolation |
| 30<br>ATKFCH2 | **Envelope Attack style for Channel 2.** This bit controls the type of interpolation used in the attack phase for channel 2 in DDS mode.<br>0 linear interpolation<br>1 exponential interpolation |
| 29<br>ATKFCH1 | **Envelope Attack style for Channel 1.** This bit controls the type of interpolation used in the attack phase for channel 1 in DDS mode.<br>0 linear interpolation<br>1 exponential interpolation |
| 28<br>ATKFCH0 | **Envelope Attack style for Channel 0.** This bit controls the type of interpolation used in the attack phase for channel 0 in DDS mode.<br>0 linear interpolation<br>1 exponential interpolation |
| 27<br>RELFCH3 | **Envelope Release style for Channel 3.** This bit controls the type of interpolation used in the release phase for channel 3 in DDS mode.<br>0 linear interpolation<br>1 exponential interpolation |
| 26<br>RELFCH2 | **Envelope Release style for Channel 2.** This bit controls the type of interpolation used in the release phase for channel 2 in DDS mode.<br>0 linear interpolation<br>1 exponential interpolation |
| 25<br>RELFCH1 | **Envelope Release style for Channel 1.** This bit controls the type of interpolation used in the release phase for channel 1 in DDS mode.<br>0 linear interpolation<br>1 exponential interpolation |
| 24<br>RELFCH0 | **Envelope Release style for Channel 0.** This bit controls the type of interpolation used in the release for channel 0 in DDS mode.<br>0 linear interpolation<br>1 exponential interpolation |
| 23<br>NOPECH3 | **Envelope No-Output phase enable for Channel 3.** In DDS mode this bit controls the inter-note no-output phase.<br>0 Disable the no-output phase for channel 3 in DDS mode.<br>1 Enable the no-output phase for channel 3 in DDS mode. |
| 22<br>NOPECH2 | **Envelope No-Output phase enable for Envelope for Channel 2.** In DDS mode this bit controls the inter-note no-output phase.<br>0 Disable the no-output phase for channel 2 in DDS mode.<br>1 Enable the no-output phase for channel 2 in DDS mode. |
| 21<br>NOPECH1 | **Envelope No-Output phase enable for Envelope for Channel 1.** In DDS mode this bit controls the inter-note no-output phase.<br>0 Disable the no-output phase for channel 1 in DDS mode.<br>1 Enable the no-output phase for channel 1 in DDS mode. |
| 20<br>NOPECH0 | **Envelope No-Output phase enable for Envelope for Channel 0.** In DDS mode this bit controls the inter-note no-output phase.<br>0 Disable the no-output phase for channel 0 in DDS mode.<br>1 Enable the no-output phase for channel 0 in DDS mode. |

| Field | Description |
|---|---|
| 19<br>TPCECH3 | **Target note Pulse Counter Enable For Channel 3.** In DDS mode this bit controls the note pulses counter.<br>0 Disable the pulse counter for channel 3 in DDS mode.<br>1 Enable the pulse counter for channel 3 in DDS mode. |
| 18<br>TPCECH2 | **Target note Pulse Counter Enable For Channel 2.** In DDS mode this bit controls the note pulse counter.<br>0 Disable the pulse counter for channel 2 in DDS mode.<br>1 Enable the pulse counter for channel 2 in DDS mode. |
| 17<br>TPCECH1 | **Target note Pulse Counter Enable For Channel 1.** In DDS mode this bit controls the note pulse counter.<br>0 Disable the pulse counter for channel 1 in DDS mode.<br>1 Enable the pulse counter for channel 1 in DDS mode. |
| 16<br>TPCECH0 | **Target note Pulse Counter Enable For Channel 0.** In DDS mode this bit controls the note pulse counter.<br>0 Disable the note pulse counter for channel 0 in DDS mode.<br>1 Enable the note pulse counter for channel 0 in DDS mode. |
| 15<br>RPTECH3 | **Repeat Mode enable for Channel 3.** In Wave mode, this bit controls the playback mode for channel 3.<br>0 Disable repeat mode for channel 3 in wave mode.<br>1 Enable repeat mode for channel 3 in wave mode. |
| 14<br>RPTECH2 | **Repeat Mode enable for Channel 2.** In Wave mode, this bit controls the playback mode for channel 2.<br>0 Disable repeat mode for channel 2 in wave mode.<br>1 Enable repeat mode for channel 2 in wave mode. |
| 13<br>RPTECH1 | **Repeat Mode enable for Channel 1.** In Wave mode, this bit controls the playback mode for channel 1.<br>0 Disable repeat mode for channel 1 in wave mode.<br>1 Enable repeat mode for channel 1 in wave mode. |
| 12<br>RPTECH0 | **Repeat Mode enable for Channel 0.** In Wave mode, this bit controls the playback mode for channel 0.<br>0 Disable repeat mode for channel 0 in wave mode.<br>1 Enable repeat mode for channel 0 in wave mode. |
| 11-8 | **Reserved**. |
| 7-6<br>SFCH3 | **Sample Format for Channel 3.** Defines the PCM data format for Channel 3.<br>00 16-bit<br>01 14-bit<br>10 12-bit<br>11 8-bit |
| 5-4<br>SFCH2 | **Sample Format for Channel 2.** Defines the PCM data format for Channel 2.<br>00 16-bit<br>01 14-bit<br>10 12-bit<br>11 8-bit |
| 3-2<br>SFCH1 | **Sample Format for Channel 1.** Defines the PCM data format for Channel 1.<br>00 16-bit<br>01 14-bit<br>10 12-bit<br>11 8-bit |
| 1-0<br>SFCH0 | **Sample Format for Channel 0.** Defines the PCM data format for Channel 0.<br>00 16-bit<br>01 14-bit<br>10 12-bit<br>11 8-bit |

### 39.6.2.3 Clock Configuration Register for Resampler (CLKRSP)

The CLKRSP register controls the resampling clock.

SGM Register Base + 0x0008

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

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

**Figure 39-4. Clock Configuration Register for Resampler (CLKRSP)**

**Table 39-5. CLKRSP Register Description**

| Field | Description |
|---|---|
| 31-30<br>CLKS | **Clock Source Selection Resampler.** See Table 39-6 for details. |
| 29-24 | **Reserved.** |
| 18-16<br>PRSR | **Prescaler Range of Resampler.** These three bits determine the range of the prescaler clock, as shown in Table 39-7 |
| 15-0<br>PRSM | **Prescaler Modulus Select.** These 16 bits control the prescale modulus counter in the clock generator. The clock divide ratio is (PRSM+1). A value of 0 disables the generated clock. |

**Table 39-6. Clock Source Selection**

| CLKS[1:0] | Selected Clock Source |
|---|---|
| 0x00 | IPS Clock |
| 0x01 | Sample clock 1 |
| 0x02 | Sample Clock 2 |
| 0x03 | System Clock |

**Table 39-7. Prescaler Range**

| PRSR[2] | PRSR[1] | PRSR[0] | Prescaled Clock |
|---|---|---|---|
| 0 | 0 | 0 | Source clock |
| 0 | 0 | 1 | Source clock / 2 |
| 0 | 1 | 0 | Source clock / 4 |
| 0 | 1 | 1 | Source clock / 8 |
| 1 | 0 | 0 | Source clock / 16 |
| 1 | 0 | 1 | Source clock / 32 |
| 1 | 1 | 0 | Source clock / 64 |

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | Source clock / 128 |

**NOTE:**

The channel configuration registers are described for channel 3 - CLKCH3(offset 0x000C) to RNCCH3 (offset 0x0030). The registers for the other three channels are identical except that they have the offsets shown below:

For channel 2, CLKCH2(offset 0x0034) to RNCCH2(offset 0x0058).

For channel 1, CLKCH1(offset 0x005C) to RNCCH1(offset 0x0080).

For channel 0, CLKCH0(offset 0x084) to RNCCH0(offset 0x00A8).

### 39.6.2.4 Clock Configuration Register for Channel 3 (CLKCH3)

The CLKCH3 register controls the clock for channel 3.

SGM Register Base + 0x000C (Channel 3)
SGM Register Base + 0x0034 (Channel 2)
SGM Register Base + 0x005C (Channel 1)
SGM Register Base + 0x0084 (Channel 0)

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

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

**Figure 39-5. Clock Configuration Register for Channel 3 (CLKCH3)**

**Table 39-8. CLKCH3 Register Description**

| Field | Description |
|---|---|
| 31-30 CLKS | **Clock Source Selection of Channel 3.** See Table 39-6 for details. |
| 29-24 | **Reserved.** |
| 18-16 PRSR | **Prescaler Range of Channel 3.** These three bits determine the range of the prescaler clock, as shown in Table 39-7 |
| 15-0 PRSM | **Prescaler Modulus Select.** These 16 bits control the prescale modulus counter in the clock generator. The clock divide ratio is (PRSM+1). A value of 0 disables the generated clock. |

### 39.6.2.5 DDS Configuration Register for Channel 3 (DDSCH3)

DDSCH3 contains the accumulator increment value for Channel 3 DDS mode.

SGM Register Base + 0x0010 (Channel 3)
SGM Register Base + 0x0038 (Channel 2)
SGM Register Base + 0x0060 (Channel 1)
SGM Register Base + 0x0088 (Channel 0)

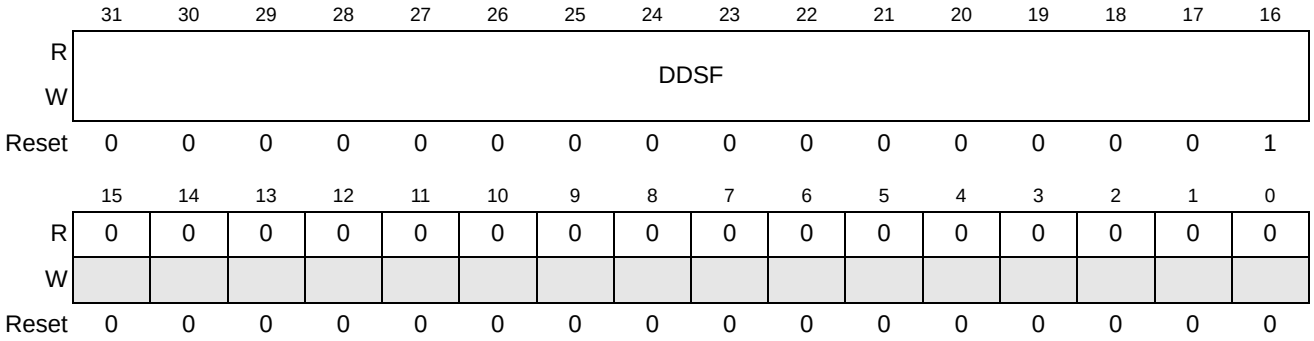



**Figure 39-6. DDS Configuration Register for Channel 3 (DDSCH3)**

**Table 39-9. DDSCH3 Register Description**

| Field | Description |
|---|---|
| 31-16<br>DDSF | **Accumulator increment value for DDS Channel 3.**<br>DDSF is added to the wavetable memory pointer every clock cycle. It therefore controls the frequency of the sound. If DDSF is set to all 0's, no sound will generated. |
| 15-0 | **Reserved.** |

### 39.6.2.6 Envelope Configuration Register of Attack Phase for Channel 3 (ECRACH3)

The ECRACH3 register controls the envelope shape for the Attack phase of Channel 3.

SGM Register Base + 0x0014 (Channel 3)
SGM Register Base + 0x003C (Channel 2)
SGM Register Base + 0x0064 (Channel 1)
SGM Register Base + 0x008C (Channel 0)

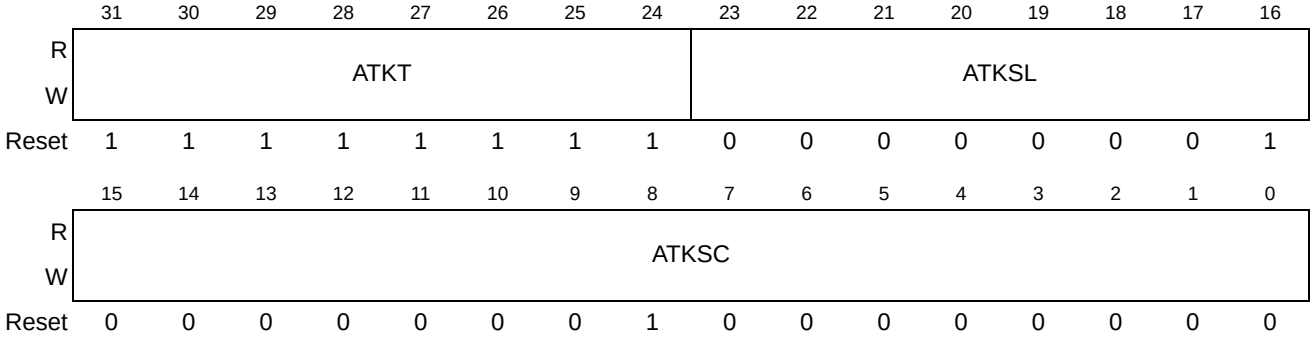



**Figure 39-7. Envelope Configuration Register of Attack Phase for Channel 3 (ECRACH3)**

**Table 39-10. ECRACH3 Register Description**

| Field | Description |
|---|---|
| 31-24<br>ATKT | **Target Volume of Attack Phase.** These bits control the final amplitude of the attack phase. |
| 23-16<br>ATKSL | **Attack Step Size.** These bits control the step height of the volume increment for the envelope if linear interpolation is selected. For each step, ATKSL is added to the existing amplitude. |
| 15-0<br>ATKSC | **Attack Step duration count.** These bits control the duration of an amplitude increment step for the ASR envelope. The duration counter is clocked by the channel clock. When the value of the step counter reaches ATKSC+1 the next step will be taken. |

The duration of the entire attack phase can be calculated with the method below:

- The duration of one step: $D_{step} = T_{chanel\ clock}*(ATKSC +1)$
- The total number of steps for linear interpolation: $N_{step} = ceil(ATKT/ATKSL) - 1$
- The total number of steps for exponential interpolation: $N_{step} = ceil(log((ATKT+1),2))-1$
- The entire duration of the attack phase : $D_{atk} = D_{step} * N_{step}$
- The minimum attack phase occurs when ATKSL = 0xFF and ATKSC = 0

**NOTE**

The function ceil(x) returns the smallest integer no less than x.

The function log(x,2) returns the binary logarithm of x.

### 39.6.2.7 Envelope Configuration Register of Release Phase for Channel 3 (ECRRCH3)

The ECRRCH3 register controls the envelope shape for the Release phase of Channel 3.

SGM Register Base + 0x0018 (Channel 3)
SGM Register Base + 0x0040 (Channel 2)
SGM Register Base + 0x0068 (Channel 1)
SGM Register Base + 0x0090 (Channel 0)

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RELT | | | | | | | | RELSL | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RELSC | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-8. Envelope Configuration Register of Release Phase for Channel 3 (ECRRCH3)**

**Table 39-11. ECRRCH3 Register Description**

| Field | Description |
|---|---|
| 31-24<br>RELT | **Target Volume of Release Phase.** These bits control the final amplitude of the release phase**.** |
| 23-16<br>RELSL | **Release Step Length.** These bits control the step height of the volume decrement for the envelope if linear interpolation is selected. For each step, RELSL is subtracted from the existing amplitude.. |
| 15-0<br>RELSC | **Release Step duration Count.** These bits control the duration of an amplitude decrement step for the ASR envelope. The duration counter is clocked by the channel clock. When the value of the step counter reaches RELSC+1 the next step will be taken. |

The duration of the entire release phase can be calculated with the method below:

- The duration of one step: $D_{step} = T_{chanel\ clock}*(RELSC +1)$
- The total number of steps for linear interpolation: $N_{step}$ = ceil((ATKT-RELT)/RELSL)
- The total number of steps for exponential interpolation: $N_{step}$=ceil(log((ATKT/RELT),2))
- The entire duration of Release Phase : $D_{rel} = D_{step} * N_{step}$

### 39.6.2.8 Envelope Configuration Register of sustain Timing for Channel 3 (ECRSCH3)

ERSCH3 controls the timing of the envelope sustain phase of Channel 3.

SGM Register Base + 0x001C (Channel 3)
SGM Register Base + 0x0044 (Channel 2)
SGM Register Base + 0x006C (Channel 1)
SGM Register Base + 0x0094 (Channel 0)

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

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

**Figure 39-9. Envelope Timing Configuration Register of sustain timing for Channel 3 (ECRSCH3)**

**Table 39-12. ECRSCH3 Register Description**

| Field | Description |
|---|---|
| 31-24 | **Reserved.** |
| 23-0<br>SUST | **sustain Timing.** These bits control the timing of sustain phase of the envelope for channel 3. |

The duration of the Sustain phase can be calculated with the method below:

- The entire duration of the Sustain Phase : $D_{sust} = T_{chanel\ clock}*(SUST +1)$

So the entire duration of one note is : $D_{note} = D_{atk} + D_{rel} + D_{sust}$

### 39.6.2.9 Inter-Note No-Output Phase Timing for Channel 3 (NTCH3)

NTCH3 controls the timing of inter-note No-Output phase of Channel 3.

SGM Register Base + 0x0020 (Channel 3)
SGM Register Base + 0x0048 (Channel 2)
SGM Register Base + 0x0070 (Channel 1)
SGM Register Base + 0x0098 (Channel 0)

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

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

**Figure 39-10. Inter-Note No-Output phase timing for Channel 3 (NTCH3)**

**Table 39-13. NTCH3 Register Description**

| Field | Description |
|---|---|
| 31-24 | **Reserved.** |
| 23-0 NOPT | **No-Output Timing.** These bits control the timing of No-Output phase of the channel 3. |

The duration of the no-output phase can be calculated with the method below:

- The entire duration of Attack Phase : $D_{nop} = T_{chanel\ clock}*(NOPT +1)$

### 39.6.2.10 Target Note Pulse Count for Channel 3 (TPCCH3)

TPCCH3 controls the target note pulse number for Channel 3 in DDS mode.

SGM Register Base + 0x0024 (Channel 3)
SGM Register Base + 0x004C (Channel 2)
SGM Register Base + 0x0074 (Channel 1)
SGM Register Base + 0x009C (Channel 0)

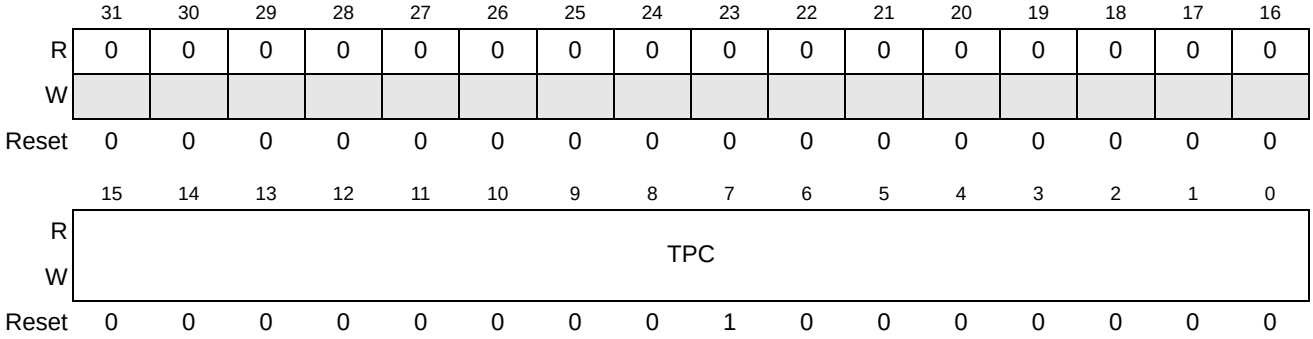



**Figure 39-11. Target note Pulse Number for Channel 3 (TPCCH3)**

**Table 39-14. TPCCH3 Register Description**

| Field | Description |
|---|---|
| 31-16 | **Reserved**. |
| 14-0 TPC | **Target note Pulse Count.** These bits control the number of note pulses in DDS mode for channel 3. When the pulses number reaches TPC Channel 3 enters the IDLE state. If the TPC is set to 0, the note generation will be endless until the SGMTCL.SOGCH3 de-asserts. |

## 39.6.2.11 PlayBack Timing Configuration Register for Channel 3(PTCCH3)

PTCCH3 controls the length of Playback for Channel 3 in Wave mode.

SGM Register Base + 0x0028 (Channel 3)
SGM Register Base + 0x0050 (Channel 2)
SGM Register Base + 0x0078 (Channel 1)
SGM Register Base + 0x00A0 (Channel 0)

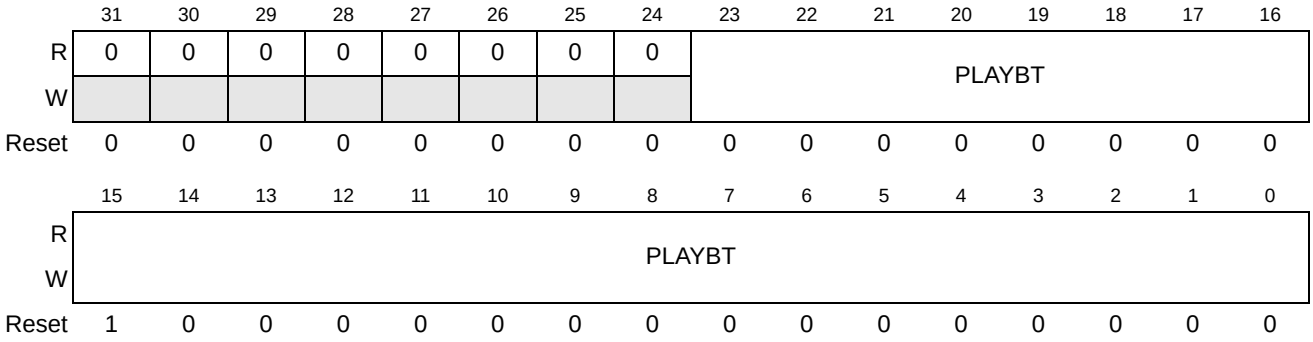



**Figure 39-12. PlayBack Timing Configuration Register for channel 3(PTCCH3)**

**Table 39-15. PBTCCH3 register Description**

| Field | Description |
|---|---|
| 22-0 PLAYBT | **PlayBack Time.** These bits control the length of playback for channel 3 in Wave mode. The channel fetches and plays (PLAYBT+1) samples for each wave repetition. |

### 39.6.2.12 Dead Time Configuration Register for Channel 3(DTCCH3)

DTCCH3 controls the timing of the Deadtime between wave repetitions for Channel 3 in Wave mode.

SGM Register Base + 0x002C (Channel 3)
SGM Register Base + 0x0054 (Channel 2)
SGM Register Base + 0x007C (Channel 1)
SGM Register Base + 0x00A4 (Channel 0)

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

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

**Figure 39-13. Dead Time Configuration Register for channel 3(DTCCH3)**

**Table 39-16. DTCCH3 register Description**

| Field | Description |
|---|---|
| 22-0 DEADT | **Dead Time.** These bits control the timing of Deadtime for channel 3 in Wave mode. |

### 39.6.2.13 Repeat Number Configuration Register for Channel 3(RNCCH3)

RNCCH3 controls the repetition number for Channel 3 in Wave mode.

SGM Register Base + 0x0030 (Channel 3)
SGM Register Base + 0x0058 (Channel 2)
SGM Register Base + 0x0080 (Channel 1)
SGM Register Base + 0x00A8 (Channel 0)

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

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

**Figure 39-14. Repeat Number Configuration Register for channel 3(RNCCH3)**

**Table 39-17. RPTN register Description**

| Field | Description |
|---|---|
| 9-0<br>RPTN | **Repeat Number.** These bits control the number of repetitions for channel 3 in Wave mode when the repeat mode is selected. Setting RPTN to 0 causes the wave to repeat endlessly. |

## 39.6.2.14 Volume Control Register for Wave mode (VCRWAV)

VCRWAV controls the volume of all 4 channels in Wave mode.

SGM Register Base + 0x00AC

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VOLCH3 | | | | | | | | VOLCH2 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VOLCH1 | | | | | | | | VOLCH0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 39-15. Volume Control Register for Wave mode (VCRWAV)**

Table 39-18. VCRWAV register Description

| Field | Description |
|---|---|
| 31-24 VOLCH3 | **Volume control for Channel 3 in Wave mode.** |
| 23-16 VOLCH2 | **Volume control for Channel 2 in Wave mode.** |
| 16-8 VOLCH1 | **Volume control for Channel 1 in Wave mode.** |
| 7-0 VOLCH0 | **Volume control for Channel 0 in Wave mode.** |

### 39.6.2.15 SGM TimeOut Count Register (SGMTOCR)

SGMTOCR controls the sound playback timeout.

SGM Register Base + 0x00B0

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TIMEOUT | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TIMEOUT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-16. SGM TimeOut Count Register (SGMTOCR)**

Table 39-19. SGMTOCR register Description

| Field | Description |
|---|---|
| 31-0 TIMTOUT | **Timeout Timing Count.** These bits control the time out timing when the SGM is playing a sound. |

### 39.6.2.16 Mixer Configuration Register (MIXCR)

MIXCR controls the channel selection for the left mixer and right mixer.

SGM Register Base + 0x00B4

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CHSL | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

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

**Figure 39-17. Mixer Configuration Register (MIXCR)**

**Table 39-20. MIXCR Register Description**

| Field | Description |
|---|---|
| 19-16<br>CHSL | **Channel Select for left mixer.**<br>• CHSL[3]=0 : channel 3 is not selected for left mixer<br>• CHSL[3]=1 : channel 3 is selected for the left mixer |
| 3-0<br>CHSR | **Channel Select for right mixer.**<br>• CHSR[3]=0 : channel 3 is not selected for the right mixer<br>• CHSR[3]=1 : channel 3 is selected for the right mixer |

## 39.6.2.17 Clock Configuration Register for PWM (CLKPWM)

The CLKPWM register determines the PWM clock.

SGM Register Base + 0x00B8

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

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

**Figure 39-18. Clock Configuration Register for PWM (CLKPWM)**

**Table 39-21. CLKPWM Register Description**

| Field | Description |
|---|---|
| 31-30<br>CLKS | **Clock Source Selection Resampler.** See Table 39-6 for details. |
| 29-24 | **Reserved.** |

| Field | Description |
|---|---|
| 18-16 PRSR | **Prescaler Range of Resampler.** These three bits determine the range of the prescaler clock, as shown in Table 39-7 |
| 15-0 PRSM | **Prescaler Modulus Select.** These 16 bits control the prescale modulus counter in the clock generator. The clock divide ratio is (PRSM+1). A value of 0 disables the generated clock. |

### 39.6.2.18 PWM Configuration Register (PWMCR)

PWMCR controls the PWM frequency and TON frequency.

SGM Register Base + 0x00BC

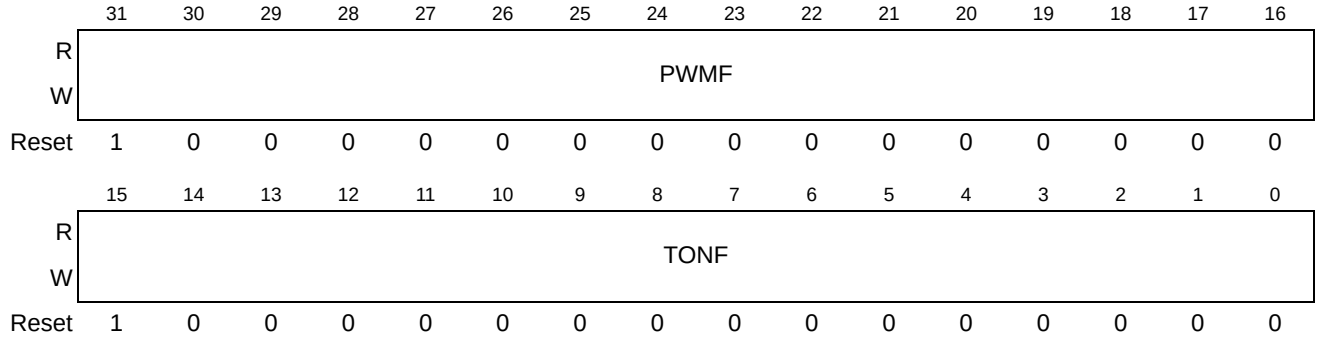



**Figure 39-19. PWM Configuration Register (PWMCR)**

**Table 39-22. MIXCR Register Description**

| Field | Description |
|---|---|
| 31-16 PWMF | **PWM Frequency.** These bits control the PWM frequency and also define the maximum amplitude (100% duty cycle) od the PWM. A 100% duty cycle (continually high) will be generated if the value of the MIXed data is higher than PWMF |
| 15-0 TONF | **Ton Frequency.** These bits determine the PWM duty cycle change frequency. The divider is clocked by pwm_clk. The divide ratio is 2xTONFxPWMF. |

### 39.6.2.19 Data FIFO Register 1(DFIFO1)

DFIFO1 provides the Data FIFO interface for Channel 3 and Channel 2.

SGM Register Base + 0x00C0

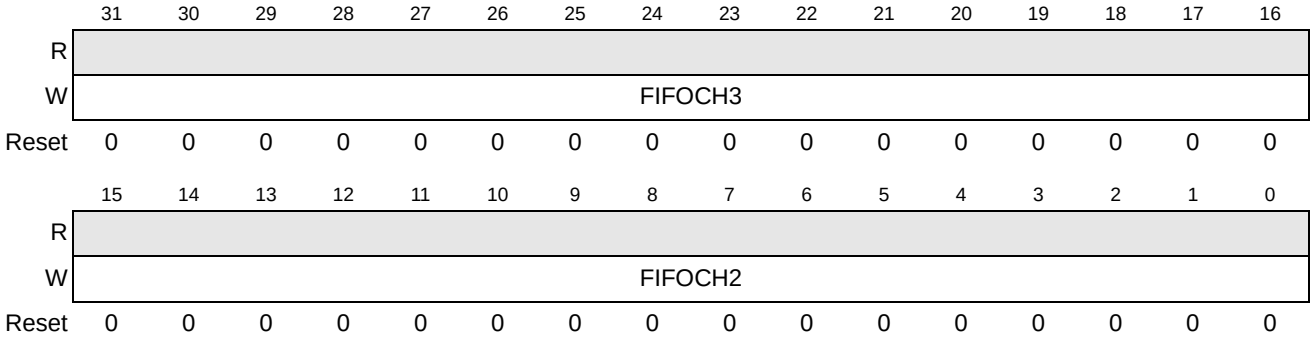



**Figure 39-20. PWM Configuration Register (PWMCR)**

**Table 39-23. Data FIFO Register 1**

| Field | Description |
|---|---|
| 31-16<br>FIFOCH3 | **Data FIFO for Channel 3.**<br>Writes to this register will be stored in the FIFO and increment the FIFO write pointer. |
| 15-0<br>FIFOCH2 | **Data FIFO for Channel 2.**<br>Writes to this register will be stored in the FIFO and increment the FIFO write pointer. |

## 39.6.2.20 Data FIFO Register 2(DFIFO2)

DFIFO2 provides the Data FIFO interface for Channel 1 and Channel 0.

SGM Register Base + 0x00C4

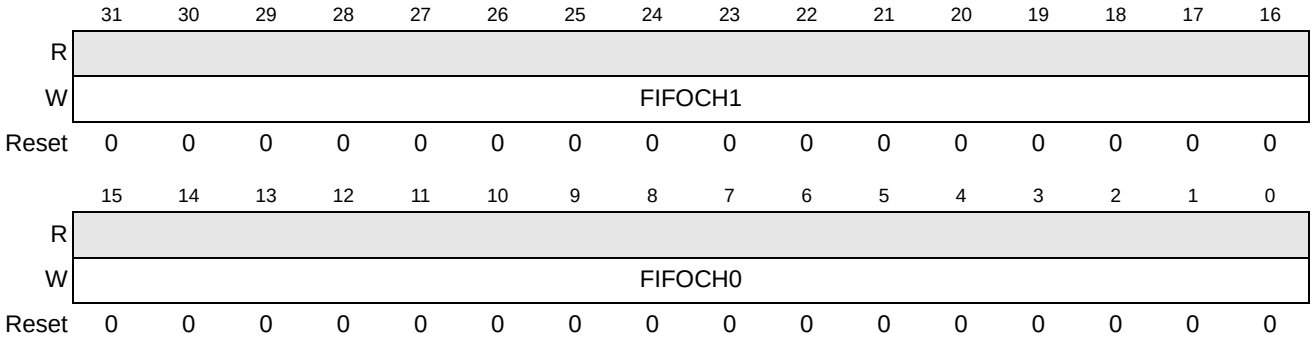



**Figure 39-21. Data FIFO Register (DFIFO2)**

**Table 39-24. Data FIFO Register 2**

| Field | Description |
|---|---|
| 31-16<br>FIFOCH1 | **Data FIFO for Channel 1.**<br>Writes to this register will be stored in the FIFO and increment the FIFO write pointer. |
| 15-0<br>FIFOCH0 | **Data FIFO for Channel 0.**<br>Writes to this register will be stored in the FIFO and increment the FIFO write pointer. |

### 39.6.2.21 FIFO WaterMark (FIFOWM)

The FIFOWM register configures the watermark of the FIFOs.

SGM Register Base + 0x00C8

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WMCH3 | | | | | | | | WMCH2 | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WMCH1 | | | | | | | | WMCH0 | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-22. FIFO WaterMark Register (FIFOWM)**

**Table 39-25. FIFO WaterMark Register 2**

| Field | Description |
|---|---|
| 31-24<br>WMCH3 | **FIFO WaterMark for Channel 3.** |
| 23-16<br>WMCH2 | **FIFO WaterMark for Channel 2.** |
| 16-8<br>WMCH1 | **FIFO WaterMark for Channel 1.** |
| 7-0<br>WMCH0 | **FIFO WaterMark for Channel 0.** |

### 39.6.2.22 FIFO Read Pointer (FIFORP)

The FIFORP register contains the current FIFO read pointers. Any write to FRPCHx will clear/flush the FIFO of the channel, including the read/write pointer and all the FIFO flags. Any 32-bit write to the FIFORP will clear/flush the entire datapath/pipeline of the SGM.

SGM Register Base + 0x00CC

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FRPCH3 | | | | | | | | FRPCH2 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FRPCH1 | | | | | | | | FRPCH0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-23. FIFO Read Pointer (FIFORP)**

Table 39-26. FIFO Read Pointer

| Field | Description |
|---|---|
| 31-24<br>FRPCH3 | **FIFO Read Pointer for Channel 3.**<br>Any write to this byte will clear/flush the FIFO of channel 3. |
| 23-16<br>FRPCH2 | **FIFO Read Pointer for Channel 2.**<br>Any write to this byte will clear/flush the FIFO of channel 2. |
| 16-8<br>FRPCH1 | **FIFO Read Pointer for Channel 1.**<br>Any write to this byte will clear/flush the FIFO of channel 1. |
| 7-0<br>FRPCH0 | **FIFO Read Pointer for Channel 0.**<br>Any write to this byte will clear/flush the FIFO of channel 0. |

### 39.6.2.23 FIFO Write Pointer (FIFOWP)

The FIFOWP contains the current FIFO write pointers. Any write to FWPCHx will clear/flush the FIFO and datapath of this channel, including the read/write pointer and all the FIFO flags. Any 32-bit write to the FIFOWP will clear/flush the entire datapath/pipeline of the SGM.

SGM Register Base + 0x00D0

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FWPCH3 | | | | | | | | FWPCH2 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FWPCH1 | | | | | | | | FWPCH0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 39-24. FIFO Write Pointer (FIFOWP)

Table 39-27. FIFO Write Pointer

| Field | Description |
|---|---|
| 31-24<br>FWPCH3 | **FIFO write pointer for Channel 3.**<br>Any write to this byte will clear/flush the FIFO of channel 3. |
| 23-16<br>FWPCH2 | **FIFO write pointer for Channel 2.**<br>Any write to this byte will clear/flush the FIFO of channel 2. |
| 16-8<br>FWPCH1 | **FIFO write pointer for Channel 1.**<br>Any write to this byte will clear/flush the FIFO of channel 1. |
| 7-0<br>FWPCH0 | **FIFO write pointer for Channel 0.**<br>Any write to this byte will clear/flush the FIFO of channel 0. |

### 39.6.2.24 SGM Status Register (SGMST)

The SGMST register indicates the current operating status of the SGM.

SGM Register Base + 0x00D4

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | FLSCH3 | STATCH3 | | | 0 | 0 | 0 | 0 | FLSCH2 | STATCH2 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | FLSCH1 | STATCH1 | | | 0 | 0 | 0 | 0 | FLSCH0 | STATCH0 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-25. FIFO Write Pointer (FIFOWP)**

**Table 39-28. FIFO Write Pointer**

| Field | Description |
|---|---|
| 27<br>FLSCH3 | **FIFO and DataPath of channel 3 is being flushed.**<br>0: FIFO and DataPath of channel 3 is ready<br>1: FIFO and DataPath of channel 3 is being flushed<br>If and only if the FIFO and DataPath of channel 3 is ready(FLSCH3=0), is it allowed to fill the FIFO and start sound generation on channel 3 (set SGMCTL.SOGCH3). Otherwise it will lead to unpredictable consequences. |
| 26-24<br>STATCH3 | **Operation Status of Channel 3.**<br>These bits indicate the current state of Channel 3. See Table 39-40 and Table 39-41 for details of these states.<br>000: IDLE<br>001: PLAYB<br>010: DEADT<br>011: ATTK<br>100: SUST<br>101: RELS<br>110: NOPT<br>111: N/A<br>If and only the operation status is IDLE (STATCH3=000), is it allowed to start sound generation on channel 3 (set SGMCTL.SOGCH3). Otherwise it will lead to unpredictable consequences. |
| 19<br>FLSCH2 | **FIFO and DataPath of channel 2 is being flushed.**<br>0: FIFO and DataPath of channel 2 is ready<br>1: FIFO and DataPath of channel 2 is being flushed<br>If and only if the FIFO and DataPath of channel 2 is ready(FLSCH2=0), is it allowed to fill the FIFO and start sound generation on channel 2(set SGMCTL.SOGCH2). Otherwise it will lead to unpredictable consequences. |

| Field | Description |
|---|---|
| 18-16<br>STATCH2 | **Operation Status of Channel 2.**<br>These bits indicate the current state of Channel 2. See Table 39-40 and Table 39-41 for details of these states.<br>000: IDLE<br>001: PLAYB<br>010: DEADT<br>011: ATTK<br>100: SUST<br>101: RELS<br>110: NOPT<br>111: N/A<br>If and only if the operation status is IDLE (STATCH2=000), is it allowed to start sound generation on channel 2 (set SGMCTL.SOGCH2). Otherwise it will lead to unpredictable consequences. |
| 11<br>FLSCH1 | **FIFO and DataPath of channel 1 is being flushed.**<br>0: FIFO and DataPath of channel 1 is ready<br>1: FIFO and DataPath of channel 1 is being flushed<br>If and only if the FIFO and DataPath of channel 1 is ready(FLSCH1=0), is it allowed to fill the FIFO and start sound generation on channel 1(set SGMCTL.SOGCH1). Otherwise it will lead to unpredictable consequences. |
| 10-8<br>STATCH1 | **Operation Status of Channel 1.**<br>These bits indicate the current state of Channel 1. See Table 39-40 and Table 39-41 for details of these states.<br>000: IDLE<br>001: PLAYB<br>010: DEADT<br>011: ATTK<br>100: SUST<br>101: RELS<br>110: NOPT<br>111: N/A<br>If and only if the operation status is IDLE (STATCH1=000), is it allowed to start sound generation on channel 1 (set SGMCTL.SOGCH1). Otherwise it will lead to unpredictable consequences. |
| 3<br>FLSCH0 | **FIFO and DataPath of channel 0 is being flushed.**<br>0: FIFO and DataPath of channel 0 is ready<br>1: FIFO and DataPath of channel 0 is being flushed<br>If and only if the FIFO and DataPath of channel 0 is ready(FLSCH0=0), is it allowed to fill the FIFO and start sound generation on channel 0 (set SGMCTL.SOGCH0). Otherwise it will lead to unpredictable consequences. |
| 2-0<br>STATCH0 | **Operation Status of Channel 0.**<br>These bits indicate the current state of Channel 0. See Table 39-40 and Table 39-41 for details of these states.<br>000: IDLE<br>001: PLAYB<br>010: DEADT<br>011: ATTK<br>100: SUST<br>101: RELS<br>110: NOPT<br>111: N/A<br>If and only if the operation status is IDLE (STATCH0=000), is it allowed to start sound generation on channel 0 (set SGMCTL.SOGCH0). Otherwise it will lead to unpredictable consequences. |

### 39.6.2.25 SGM Interrupt Control Register for FIFO and DMA (SGMICFD)

The SGMICFD register enables and controls the FIFO interrupt and DMA request functions.

SGM Register Base + 0x00D8

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FLDE CH3 | FLDE CH2 | FLDE CH1 | FLDE CH0 | 0 | 0 | 0 | 0 | FLIE CH3 | FLIE CH2 | FLIE CH1 | FLIE CH0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FFIE CH3 | FFIE CH2 | FFIE CH1 | FFIE CH0 | FEIE CH3 | FEIE CH2 | FEIE CH1 | FEIE CH0 | FOIE CH3 | FOIE CH2 | FOIE CH1 | FOIE CH0 | FUIE CH3 | FUIE CH2 | FUIE CH1 | FUIE CH0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-26. SGM Interrupt Control Register for DMA and FIFO (SGMICDF)**

**Table 39-29. SGM Interrupt Control Register for DMA and FIFO**

| Field | Description |
|---|---|
| 31<br>FLDECH3 | **FIFO level DMA Request Enable for Channel 3.**<br>Enables a DMA service request when the FIFO level is below the watermark of channel 3. |
| 30<br>FLDECH2 | **FIFO level DMA Request Enable for Channel 2.**<br>Enables a DMA service request when the FIFO level is below the watermark of channel 2. |
| 29<br>FLDECH1 | **FIFO level DMA Request Enable for Channel 1.**<br>Enables a DMA service request when the FIFO level is below the watermark of channel 1. |
| 28<br>FLDECH0 | **FIFO level DMA Request Enable for Channel 0.**<br>Enables a DMA service request when the FIFO level is below the watermark of channel 0. |
| 27-24 | **Reserved.** |
| 23<br>FLIECH3 | **FIFO level interrupt Request Enable for Channel 3.**<br>Enables an interrupt request when the FIFO level is below the watermark of channel 3. |
| 22<br>FLIECH2 | **FIFO level interrupt Request Enable for Channel 2.**<br>Enables the interrupt request when the FIFO level is below the watermark of channel 2. |
| 21<br>FLIECH1 | **FIFO level interrupt Request Enable for Channel 1.**<br>Enables the interrupt request when the FIFO level is below the watermark of channel 1. |
| 20<br>FLIECH0 | **FIFO level interrupt Request Enable for Channel 0.**<br>Enables the interrupt request when the FIFO level is below the watermark of channel 0. |
| 19-16 | Reserve**d.** |
| 15<br>FFIECH3 | FIFO Full Interrupt Enable for Channel 3. |
| 14<br>FFIECH2 | FIFO Full Interrupt Enable for Channel 2. |
| 13<br>FFIECH1 | FIFO Full Interrupt Enable for Channel 1. |
| 12<br>FFIECH0 | FIFO Full Interrupt Enable for Channel 0. |
| 11<br>FEIECH3 | FIFO Empty Interrupt Enable for Channel 3. |

| Field | Description |
|---|---|
| 10<br>FEIECH2 | FIFO Empty Interrupt Enable for Channel 2. |
| 9<br>FEIECH1 | FIFO Empty Interrupt Enable for Channel 1. |
| 8<br>FEIECH0 | FIFO Empty Interrupt Enable for Channel 0. |
| 7<br>FOIECH3 | FIFO Overflow Interrupt Enable for Channel 3. |
| 6<br>FOIECH2 | FIFO Overflow Interrupt Enable for Channel 2. |
| 5<br>FOIECH1 | FIFO Overflow Interrupt Enable for Channel 1. |
| 4<br>FOIECH0 | FIFO Overflow Interrupt Enable for Channel 0. |
| 3<br>FUIECH3 | FIFO Underflow Interrupt Enable for Channel 3. |
| 2<br>FUIECH2 | FIFO Underflow Interrupt Enable for Channel 2. |
| 1<br>FUIECH1 | FIFO Underflow Interrupt Enable for Channel 1. |
| 0<br>FUIECH0 | FIFO Underflow Interrupt Enable for Channel 0. |

### 39.6.2.26 SGM Interrupt Control Register (SGMIC)

The SGMIC register enables and controls the interrupts associated with each sound channel.

SGM Register Base + 0x00DC

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TOIE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDIE CH3 | PDIE CH2 | PDIE CH1 | PDIE CH0 | RDIE CH3 | RDIE CH2 | RDIE CH1 | RDIE CH0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EORI ECH3 | EORI ECH2 | EORI ECH1 | EORI ECH0 | EOAI ECH3 | EOAI ECH2 | EOAI ECH1 | EOAI ECH0 | EONI ECH3 | EONI ECH2 | EONI ECH1 | EONI ECH0 | PCIE CH3 | PCIE CH2 | PCIE CH1 | PCIE CH0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-27. SGM Interrupt Control Register (SGMIC)**

**Table 39-30. SGM Interrupt Control Register**

| Field | Description |
|---|---|
| 31<br>TOIE | TimeOut Interrupt Enable for SGM. |
| 30-24 | Reserved. |
| 23<br>PDIECH3 | Playback Duration Interrupt Enable for Channel 3 in Wave mode. |
| 22<br>PDIECH2 | Playback Duration Interrupt Enable for Channel 2 in Wave mode. |
| 21<br>PDIECH1 | Playback Duration Interrupt Enable for Channel 1 in Wave mode. |
| 20<br>PDIECH0 | Playback Duration Interrupt Enable for Channel 0 in Wave mode. |
| 19<br>RDIECH3 | Repeat Duration Interrupt Enable for Channel 3 in Wave mode. |
| 18<br>RDIECH2 | Repeat Duration Interrupt Enable for Channel 2 in Wave mode. |
| 17<br>RDIECH1 | Repeat Duration Interrupt Enable for Channel 1 in Wave mode. |
| 16<br>RDIECH0 | Repeat Duration Interrupt Enable for Channel 0 in Wave mode. |
| 15<br>EORIECH3 | Enable the Interrupt of reaching the end of Release Phase in DDS mode for channel 3. |
| 14<br>EORIECH2 | Enable the Interrupt of reaching the end of Release Phase in DDS mode for channel 2. |
| 13<br>EORIECH1 | Enable the Interrupt of reaching the end of Release Phase in DDS mode for channel 1. |
| 12<br>EORIECH0 | Enable the Interrupt of reaching the end of Release Phase in DDS mode for channel 0. |
| 11<br>EOAIECH3 | Enable the Interrupt of reaching the end of Attack Phase in DDS mode for channel 3. |
| 10<br>EOAIECH2 | Enable the Interrupt of reaching the end of Attack Phase in DDS mode for channel 2. |
| 9<br>EOAIECH1 | Enable the Interrupt of reaching the end of Attack Phase in DDS mode for channel 1. |
| 8<br>EOAIECH0 | Enable the Interrupt of reaching the end of Attack Phase in DDS mode for channel 0. |
| 7<br>EONIECH3 | Enable the Interrupt at the end of inter-note no-output phase for Channel 3 in DDS mode. |
| 6<br>EONIECH2 | Enable the Interrupt at the end of inter-note no-output phase for Channel 2 in DDS mode. |
| 5<br>EONIECH1 | Enable the Interrupt at the end of inter-note no-output phase for Channel 1 in DDS mode. |

| Field | Description |
|---|---|
| 4<br>EONIECH0 | Enable the Interrupt at the end of inter-Note no-output phase for Channel 0 in DDS mode. |
| 3<br>PCIECH3 | Enable the Interrupt when reaching the Target number of note Pulses Count for Channel 3 in DDS mode. |
| 2<br>PCIECH2 | Enable the Interrupt when reaching the Target number of note Pulses Count for Channel 2 in DDS mode. |
| 1<br>PCIECH1 | Enable the Interrupt when reaching the Target number of note Pulses Count for Channel 1 in DDS mode. |
| 0<br>PCIECH0 | Enable the Interrupt when reaching the Target number of note Pulses Count for Channel 0 in DDS mode. |

### 39.6.2.27 SGM Interrupt Status Register for FIFO and DMA (SGMISFD)

The SGMISFD register contains the DMA and FIFO interrupt status of SGM.

SGM Register Base + 0x00E0

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FLFCH3 | FLFCH2 | FLFCH1 | FLFCH0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FFFCH3 | FFFCH2 | FFFCH1 | FFFCH0 | FEFCH3 | FEFCH2 | FEFCH1 | FEFCH0 | FOFCH3 | FOFCH2 | FOFCH1 | FOFCH0 | FUFCH3 | FUFCH2 | FUFCH1 | FUFCH0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-28. SGM Interrupt Status Register for FIFO and DMA(SGMISFD)**

**Table 39-31. SGM Interrupt Status Register for FIFO and DMA**

| Field | Description |
|---|---|
| 31-24 | **Reserved.** |
| 23<br>FLFCH3 | **FIFO programmable level Flag for Channel 3.**<br>Note the DMA clears this flag when servicing a DMA request |
| 22<br>FLFCH2 | **FIFO programmable level Flag for Channel 2.**<br>Note the DMA clears this flag when servicing a DMA request |
| 21<br>FLFCH1 | **FIFO programmable level Flag for Channel 1.**<br>Note the DMA clears this flag when servicing a DMA request |
| 20<br>FLFCH0 | **FIFO programmable level Flag for Channel 0.**<br>Note the DMA clears this flag when servicing a DMA request |
| 19-16 | Reserved. |

| Field | Description |
|---|---|
| 15<br>FFFCH3 | FIFO Full Flag for Channel 3. |
| 14<br>FFFCH3 | FIFO Full Flag for Channel 2. |
| 13<br>FFFCH3 | FIFO Full Flag for Channel 1. |
| 12<br>FFFCH3 | FIFO Full Flag for Channel 0. |
| 11<br>FEFCH3 | FIFO Empty Flag for Channel 3. |
| 10<br>FEFCH2 | FIFO Empty Flag for Channel 2. |
| 9<br>FEFCH1 | FIFO Empty Flag for Channel 1. |
| 8<br>FEFCH0 | FIFO Empty Flag for Channel 0. |
| 7<br>FOFCH3 | FIFO Overflow Flag for Channel 3. |
| 6<br>FOFCH2 | FIFO Overflow Flag for Channel 2. |
| 5<br>FOFCH1 | FIFO Overflow Flag for Channel 1. |
| 4<br>FOFCH0 | FIFO Overflow Flag for Channel 0. |
| 3<br>FUFCH3 | FIFO Underflow Flag for Channel 3. |
| 2<br>FUFCH2 | FIFO Underflow Flag for Channel 2. |
| 1<br>FUFCH1 | FIFO Underflow Flag for Channel 1. |
| 4<br>FUFCH0 | FIFO Underflow Flag for Channel 0. |

### 39.6.2.28 SGM Interrupt Status Register (SGMIS)

The SGMIS register contains the status of the interrupts associated with each sound channel.

SGM Register Base + 0x00E4

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TOF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDFCH3 | PDFCH2 | PDFCH1 | PDFCH0 | RDFCH3 | RDFCH2 | RDFCH1 | RDFCH0 |
| W | w1c | | | | | | | | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EORFCH3 | EORFCH2 | EORFCH1 | EORFCH0 | EOAFCH3 | EOAFCH2 | EOAFCH1 | EOAFCH0 | EONFCH3 | EONFCH2 | EONFCH1 | EONFCH0 | PCFCH3 | PCFCH2 | PCFCH1 | PCFCH0 |
| W | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-29. SGM Interrupt Status Register (SGMIS)**

**Table 39-32. SGM Interrupt Status Register**

| Field | Description |
|---|---|
| 31<br>TOF | Timeout Flag. |
| 30-24 | Reserved. |
| 23<br>PDFCH3 | Playback Duration Over status for Channel 3 in Wave mode. |
| 22<br>PDFCH2 | Playback Duration Over status for Channel 2 in Wave mode. |
| 21<br>PDFCH1 | Playback Duration Over status for Channel 1 in Wave mode. |
| 20<br>PDFCH0 | Playback Duration Over status for Channel 0 in Wave mode. |
| 19<br>RDFCH3 | Repeat Duration over Flag for Channel 3 in Wave mode. |
| 18<br>RDFCH2 | Repeat Duration over Flag for Channel 2 in Wave mode. |
| 17<br>RDFCH1 | Repeat Duration over Flag for Channel 1 in Wave mode. |
| 16<br>RDFCH0 | Repeat Duration over Flag for Channel 0 in Wave mode. |
| 15<br>EORFCH3 | Flag of reaching the end of at Release Phase in DDS mode for channel 3. |
| 14<br>EORFCH2 | Flag of reaching the end of at Release Phase in DDS mode for channel 2. |
| 13<br>EORFCH1 | Flag of reaching the end of at Release Phase in DDS mode for channel 1. |
| 12<br>EORFCH0 | Flag of reaching the end of at Release Phase in DDS mode for channel 0. |

| Field | Description |
|---|---|
| 11<br>EOAFCH3 | Flag of reaching the end of at Attack Phase in DDS mode for channel 3. |
| 10<br>EOAFCH2 | Flag of reaching the end of at Attack Phase in DDS mode for channel 2. |
| 9<br>EOAFCH1 | Flag of reaching the end of at Attack Phase in DDS mode for channel 1. |
| 8<br>EOAFCH0 | Flag of reaching the end of at Attack Phase in DDS mode for channel 0. |
| 7<br>EONFCH3 | Flag of reaching the end of the inter-note no-output phase for Channel 3 in DDS mode. |
| 6<br>EONFCH2 | Flag of reaching the end of the inter-note no-output phase for Channel 2 in DDS mode. |
| 5<br>EONFCH1 | Flag of reaching the end of the inter-note no-output phase for Channel 1 in DDS mode. |
| 4<br>EONFCH0 | Flag of reaching the end of the inter-note no-output phase for Channel 0 in DDS mode. |
| 3<br>PCFCH3 | Flag of reaching the Target number of note Pulses Count for Channel 3 in DDS mode. |
| 2<br>PCFCH2 | Flag of reaching the Target number of note Pulses Count for Channel 2 in DDS mode. |
| 1<br>PCFCH1 | Flag of reaching the Target number of note Pulses Count for Channel 1 in DDS mode. |
| 0<br>PCFCH0 | Flag of reaching the Target number of note Pulses Count for Channel 0 in DDS mode. |

### 39.6.2.29 I2S Enable Register (I2SEN)

The I2SEN register enables the I2S operations

SGM Register Base + 0x00E8

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

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SRST | 0 | 0 | 0 | IEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-30. I2S Enable Register (I2SEN)**

**Table 39-33. I2S Enable Register**

| Field | Description |
|---|---|
| 31-5 | **Reserved.** |
| 4<br>SRST | **I2S module Soft Reset.**<br>Writing 1 to this bit soft resets the module logic. Memory mapped registers are not be affected.<br>0: Normal function<br>1: Soft Reset asserted |
| 3-1 | **Reserved.** |
| 0<br>IEN | **I2S enable.**<br>0: I2S interface disabled<br>1: 2S interface enabled |

## 39.6.2.30 I2S Control Register (I2SCTL)

The I2SCTL register controls I2S operations

SGM Register Base + 0x00EC

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

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FIFOTH | | 0 | 0 | 0 | 0 | ACEN | ACSEL | CHSEL | 0 | PM | | 0 | 0 | PSYNC | POL |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-31. I2S control register (I2SCTL)**

**Table 39-34. I2S Control Register**

| Field | Description |
|---|---|
| 31-15 | **Reserved.** |
| 15-14<br>FIFOTH | **I2S FIFO threshold level.**<br>After accumulating the number of data specified by this number, the I2S starts sending data out ofthe interface.<br>00: Threshold is 1, start as soon as data is received from SGM<br>01: Threshold is 2, start after 2 data units are received from SGM<br>10: Threshold is 3, start after 3 data units are received from SGM<br>11: Threshold is 4, start after 4 data units are received from SGM |
| 13-10 | **Reserved.** |
| 9<br>ACEN | Auxiliary Clock (MCLK).<br>0 Auxiliary clock disabled.<br>1 Auxiliary clock enabled. |
| 8<br>ACSEL | **Auxiliary Clock (MCLK) Selection.**<br>0: Auxiliary clock is 256 times the sampling clock<br>1: Auxiliary clock is 512 times the sampling clock |

| Field | Description |
|---|---|
| 7<br>CHSEL | **Channel selection for mono PCM mode.**<br>0: Left mixer data will be selected for I2S<br>1: Right mixer data will be selected for I2S |
| 6 | **Reserved.** |
| 5-4<br>PM | **Protocol Mode.**<br>00: Philips Protocol mode (stereo)<br>01: MSB Justify mode (stereo)<br>10: LSB Justify mode (stereo)<br>11:  PCM Mode (mono) |
| 3-2 | **Reserved.** |
| 1<br>PSYNC | **Active edge of Frame Sync**<br>0: Frame Sync is active high<br>1: Frame Sync is active low |
| 0<br>POL | **Clock polarity control for data transfer**<br>0: Data & Frame sync clocked on rising edge<br>1: Data & Frame sync clocked on falling edge |

### 39.6.2.31 I2S Output Data Format Control Register (I2SDFC)

The I2SDFC register controls the I2S output data format.

SGM Register Base + 0x00F0

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

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | MSBF | | 0 | BSO | | 0 | 0 | ODF | | 0 | 0 | 0 | IDF |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-32. I2S Output Data Format Control Register (I2SDFC)**

**Table 39-35.  I2S Output Data Format Control Register**

| Field | Description |
|---|---|
| 31-21 | **Reserved.** |
| 20-16<br>FSL | **Frame Sync Length.**<br>These bits specify the frame sync length for PCM mode. The actual number of Frame synch length FSL+1, so the default value is 1. |
| 15-13 | **Reserved.** |

| Field | Description |
|---|---|
| 12<br>MSBF | **MSB First.**<br>This bit represents what data is to be sent first as output on I2S interface. For formats where bit stuffing is applicable, this option will be applied only after the bit stuffing is already done<br>0: Send most significant bit out first<br>1: Send least significant bit out first |
| 9-8<br>BSO | **Bit Stuffing Option.**<br>This option is only used when the output data format is bigger than the input data format. In this case additional data is stuffed to the outgoing data.<br>000: Append 0's after Least Significant Bit (LSB)<br>001: Append most mignificant bytes (MSB) after LSB i.e repeat data<br>010: Prepend Sign Bit - most significant bit (MSB) before MSB<br>011: Prepend 0's before MSB<br>Others: Reserved<br>*Note: These bits have different implications for different protocol modes (PM). Table 39-36 shows the details.* |
| 7-6 | **Reserved.** |
| 5-4<br>ODF | **Output data format**<br>00: 16 bits per channel<br>01: 32 bits per channel<br>10: 24 bits per channel<br>11: Reserved |
| 3-1 | **Reserved.** |
| 0<br>IDF | **Input data format i.e bits per channel**<br>0: 16 bit per channel<br>1: Reserved |

**Table 39-36. Output Protocol Mode (Formats) supported by I2S**

| Protocol Mode | Interpretation |
|---|---|
| Philips Mode | **Data is always Most Significant Bit Justified.**<br>**000**: Append 0's after Least Significant Bit (LSB)<br>**001**: Append most significant bytes(MSB) after LSB i.e repeat data<br>**010**: Prepend sign bit - most significant bit (MSB) before MSB<br>**011**: Prepend 0's before MSB<br>**Others**: Reserved |
| MSB Justify Mode | **Data is always most significant bit justified.**<br>**000**: Append 0's after least significant bit (LSB)<br>**001**: Append most significant bytes(MSB) after LSB i.e repeat data<br>**Others**: Reserved |
| LSB Justify Mode | **Data is always least significant bit justified.**<br>**000**: Append 0's after least significant bit (LSB)<br>*Note: Reserved for LSB justified mode*<br>**001**: Append most significant bytes(MSB) after LSB i.e repeat data<br>*Note: Reserved for LSB justified mode*<br>**010**: Prepend sign bit - most significant bit (MSB) before MSB<br>**011**: Prepend 0's before MSB<br>**100**: Prepend appropriate most significant bytes (MSB) before MSB i.e repeat data<br>**Others**: Reserved |

**Table 39-36. Output Protocol Mode (Formats) supported by I2S (continued)**

| Protocol Mode | Interpretation |
|---|---|
| PCM Mode | **Data is always most significant bit justified.**<br>**000**: Append 0's after least significant bit (LSB)<br>**001**: Append most significant bytes (MSB) after LSB i.e repeat data<br>**010**: Prepend sign bit - most significant bit (MSB) before MSB<br>**011**: Prepend 0's before MSB<br>**100**: Prepend appropriate most significant bytes(MSB) before MSB i.e repeat data<br>**Others**: Reserved |

### 39.6.2.32 I2S Clock Prescaler Register (I2SPRS)

The I2SPRS register controls the clock prescaler values that govern the bit and auxiliary clock frequencies.

SGM Register Base + 0x00F4

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ACPSCLR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | I2SPSCLR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-33. I2S Clock Prescaler Register (I2SPRS)**

**Table 39-37. I2S Clock Prescaler Register**

| Field | Description |
|---|---|
| [31:16]<br>ACPSCLR | **Auxiliary clock pre-scaler.**<br>This value needs to be synchronised with the SGM prescaler value and the prescaler value programmed for on I2SPSCLR after accounting for data stuffing and data repetition on the Left and Right channels. |
| 15-0<br>I2SPSCLR | **I2S Clock pre-scaler.**<br>These bits specify the divide factor for the system clock to achieve the bit clock. This value should be synchronised with the prescaler values programmed for SGM after accounting for data stuffing and data repetition on the Left and Right Channels. |

### 39.6.2.33 I2S Interrupt Control Register (I2SINTC)

The I2SINTC register controls the interrupts of I2S interface.

SGM Register Base + 0x00F8

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TSFEN | 0 | 0 | 0 | 0 | 0 | FOFE | FUFE | 0 | 0 | 0 | 0 | 0 | CPLEN | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 39-34. I2S Interrupt Control Register (I2SINTC)**

**Table 39-38. I2S Enable Register**

| Field | Description |
|---|---|
| 31<br>TSFEN | **I2S transfer error interrupt enable.** |
| 31-26 | **Reserved.** |
| 25<br>FOFE | **I2S FIFO Overflow enable.** |
| 24<br>FUFE | **I2S FIFO Underflow enable.** |
| 18<br>CPLEN | **I2S complete interrupt enable.** |
| 15-0 | **Reserved.** |

## 39.6.2.34 I2S Status Register (I2SST)

The I2SINTS register shows the status of the I2S interface.

SGM Register Base + 0x00FC

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TSFE | 0 | 0 | 0 | 0 | 0 | FOF | FUF | 0 | 0 | 0 | 0 | 0 | CPL | 0 | 0 |
| W | w1c | | | | | | | | | | | | | w1c | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FIFOST | | | | 0 | 0 | 0 | BSY |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 39-35. I2S Status Register (I2SST)**

Table 39-39. I2SST field descriptions

| Field | Description |
|---|---|
| 31<br>TSFE | **I2S transfer error.**<br>If the I2S FIFO is empty but the I2S requests data then transfer error interrupt will be raised. |
| 31-26 | **Reserved.** |
| 25<br>FOF | **I2S FIFO Overflow.**<br>Interrupt indicating the FIFO overflow. The I2S soft reset can clear this flag. |
| 24<br>FUF | **I2S FIFO Underflow.**<br>Interrupt indicating the FIFO underflow. The I2S soft reset can clear this flag. |
| 18<br>CPL | **I2S complete**<br>Interrupt indicating all data is transferred after receiving end of sequence from the SGM. |
| 15-8 | **Reserved.** |
| 7-4<br>FIFOST | **FIFO Status.**<br>This field specifies the amount of data currently residing in the I2S FIFO. Cleared on soft reset, or when the I2S FIFO gets emptied. |
| 3-1 | **Reserved.** |
| 0<br>BSY | **I2S Busy.**<br>The I2S interface is busy. Cleared on soft reset or when module completes outputing data. |

## 39.7 Functional description

The Sound Generator Module creates sound by clocking PCM sound samples out of up to four sound source channels, mixing these together and sending the output to either a mono PWM output or to a stereo I2S interface. The sound channels can operate in one of two modes depending on whether a wavetable (DDS mode) or complete PCM stream (wave) is required. DDS mode includes a volume envelope that shapes the playback of the wavetable. The volume in Wave mode is determined by the volume set in the mixer for that channel.

### 39.7.1 Wave mode

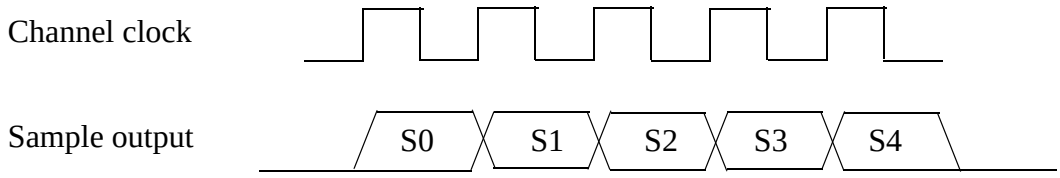
In Wave mode, the PCM data is stored in memory external to the SGM (for example RAM or flash). Each sample is fetched using the eDMA then fed to the SGM which promotes it to 16 bits per sample using its Sample Format Converter (SFC). Samples should be in two’s complement signed format. The channel FIFO is used to buffer the incoming data and then deliver the samples at the required clock rate to the mixer. As a sample is fetched from the FIFO and played by the channel its location in the FIFO will be replaced by a newer incoming sample.

There are two options for playback:

- Single Shot Mode. In this mode the channel “plays” the entire source wave once. The duration of the playback is controlled by the Playback Timing Configuration register (Section 39.6.2.11, “PlayBack Timing Configuration Register for Channel 3(PTCCH3)) which determines the number

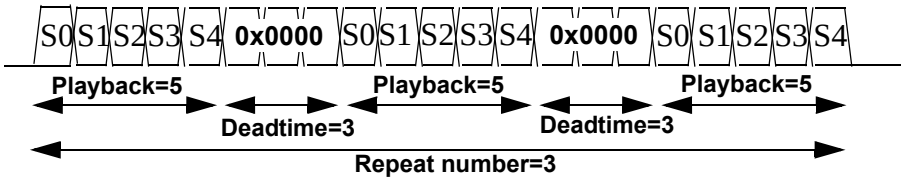
of samples in the source wave. See Figure • for a simple example where the source wave consists of 5 samples.

**Figure 39-36. Single shot mode playback**




- Repeat Mode. In this mode the channel “plays” the entire source wave then waits for a deadtime (Section 39.6.2.12, “Dead Time Configuration Register for Channel 3(DTCCH3)) during which it produces samples of value 0 then “plays” the entire wave again. The number of repeats is programmable in the Repeat Number Register (Section 39.6.2.13, “Repeat Number Configuration Register for Channel 3(RNCCH3)). In the last iteration, the deadtime duration is removed and the repeat finishes at the end of playback duration. Endless repetition can be produced by setting the repeat number to all 0. See Figure  for an example where the five sample source wave is repeated three times.

**Figure 39-37. Repeat mode playback**

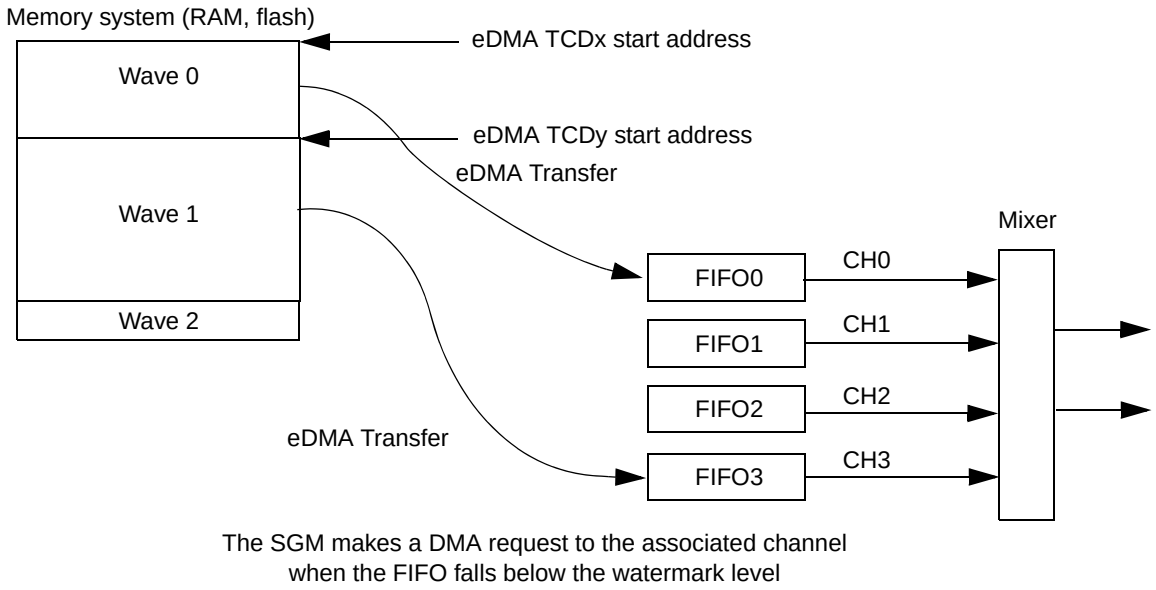



The input FIFO (256×16 bit) for each channel is used to buffer the incoming PCM audio sample stream. See Figure 39-38. The status of the FIFO such as overflow and underflow is reported in the FIFO and DMA status register (Section 39.6.2.27, “SGM Interrupt Status Register for FIFO and DMA (SGMISFD)) which can also raise interrupts if required. Configure the interrupt requirements using the FIFO and DMA interrupt control register (Section 39.6.2.25, “SGM Interrupt Control Register for FIFO and DMA (SGMICFD)).

**NOTE**

For correct operation the eDMA Transfer Descriptor must match the playback length of the wave.

**Figure 39-38. PCM Sample Flow for Wave mode**




### 39.7.1.1 State Machine of SGM channel in Wave mode

The SGM manages the operation of the channel using a state machine. The current state of the machine for each channel is found in the SGMST register. See Figure  and Table 39-40 for details of the state machine operation.

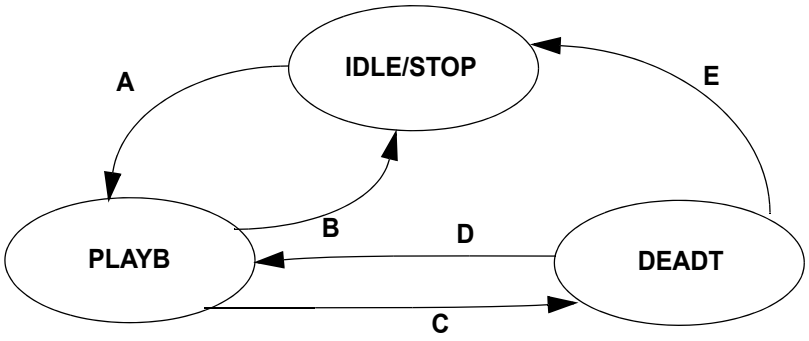



**Figure 39-39. Wave mode State Machine - by Function**

**Table 39-40. Wave mode State Machine**

| Source State | Destination State | Transfer Condition | Description |
|---|---|---|---|
| IDLE/STOP | PLAYB | A | If SOG is asserted, samples will fetched from FIFO and the playback starts. |

Table 39-40. Wave mode State Machine (continued)

| | | | |
|---|---|---|---|
| PLAYB | IDLE/STOP | B | If SOG is deasserted, the playback will be stopped immediately.<br>Alternatively, when the playback ends and no-repeat mode is selected, the playback will be stopped. |
| PLAYB | DEADT | C | If repeat mode is selected then when playback ends at the end of the current repeat, the SGM will enter the deadtime phase and give samples of 0x0000. |
| DEADT | PLAYB | D | If repeat mode is selected then when the deadtime phase ends and the channel has not reached the target number of repeats then another playback will begin. |
| DEADT | IDLE | E | If SOG is deasserted, the deadtime will be stopped immediately.<br>Alternatively, when the deadtime ends and the channel reaches the target number of repeat then the playback will stop. |

## 39.7.2 DDS mode

In DDS mode the sound is not generated by playing back a full wave stream, instead it is synthesized from a wavetable. This section explains the operation of DDS mode.

### 39.7.2.1 DDS Concept

A sound can be described by 3 different attributes: pitch, volume and timbre.

- The pitch depends on the frequency of the wave.
- The volume depends on the amplitude of the wave.
- The timbre depends on the shape of the wave. In general, each instrument has its individual timbre, e.g. piano, guitar, violin, drum, voice.

Direct Digital Synthesis relies on the fact that the timbre of a sound does not vary greatly for relatively large changes in frequency and so a single table containing samples that represent the timbre of the sound may be played back at different frequencies rather than having to store individual waves for each required pitch. The characteristic envelope of the sound may change as the pitch changes. For example higher pitched sounds may have a faster attack and shorter sustain than lower pitches. DDS mode accounts for this difference by allowing the user to configure the volume envelope for the wavetable independently of the samples in the table. The envelope has three phases:

- Attack: the volume is increasing from 0 or the previous final volume value
- Sustain: the volume remains constant
- Release: the volume is decreasing to a final value

Each phase is configurable for duration and for the phases where the volume is changing the type of change and rate of change is controllable.

### 39.7.2.2 Wavetable initialization

In DDS mode**,** the wavetable needs to be loaded into the channel FIFO by the system (typically the eDMA or CPU). The FIFO then works as a local memory buffer which the DDS uses to synthesize the sound. Unlike Wave mode the samples remain in memory until the wavetable is replaced.

### 39.7.2.3 Generating the tone

It is unlikely that a complete tone of suitable duration may be contained in the wavetable, therefore each tone is generated by playing the samples in the wavetable continually until the end of the volume envelope is reached. The rate at which the samples are fed from the wavetable depends on the clock for the channel and the increment size for the wavetable address counter (Section 39.6.2.5, “DDS Configuration Register for Channel 3 (DDSCH3)) and clock for each channel. The increment size affects the sequence of the samples as well as the rate at which new samples are selected from the wavetable so it generally remains fixed for a given tone. The pitch of the tone is therefore typically changed by increasing or decreasing the clock to the channel. Since the channel clock also affects the envelope timing it is usual to change the envelope parameters when changing the pitch of the tone.

The duration of each tone is determined by the volume envelope. The attack phase controls the maximum amplitude of the sound, the sustain phase determines how long it stays at that amplitude and the release phase controls the final amplitude of the sound. The attack and release phases are both programmable for length through the rate at which they change the amplitude and how much change is made at a time.

For the attack phase the attack configuration register (Section 39.6.2.6, “Envelope Configuration Register of Attack Phase for Channel 3 (ECRACH3)) and the SGM configuration register (Section 39.6.2.2, “SGM Configuration Register (SGMCFG)) control the behavior. The attack may use linear or exponential interpolation as configured in SGMCFG. For linear mode the channel clock increments a counter that causes an increase in amplitude of ECRACHn.ATKSL when it reaches a value of ECRACHn.ATKSC. When the output reaches the amplitude specified by ECRACHn.ATKT the attack phase is over. For exponential mode the step size is controlled by the ATKT bitfield such that each step increases in magnitude in an exponential fashion.

In the sustain phase the duration is simply determined by the number of channel clock ticks specified by the SUST bitfiled of the sustain configuration register (Section 39.6.2.8, “Envelope Configuration Register of sustain Timing for Channel 3 (ECRSCH3)).

For the release phase the attack configuration register (Section 39.6.2.7, “Envelope Configuration Register of Release Phase for Channel 3 (ECRRCH3)) and the SGM configuration register (Section 39.6.2.2, “SGM Configuration Register (SGMCFG)) control the behavior. The release may use linear or exponential interpolation as configured in SGMCFG. For linear mode the channel clock increments a counter that causes a decrease in amplitude of ECRRCHn.RELSL when it reaches a value of ECRRCHn.RELSC. When the output reaches the amplitude specified by ECRRCHn.RELT the release phase is over. For exponential mode the step size is controlled by the RELT bitfield such that each step decreases in magnitude in an exponential fashion. See Figure  and Figure

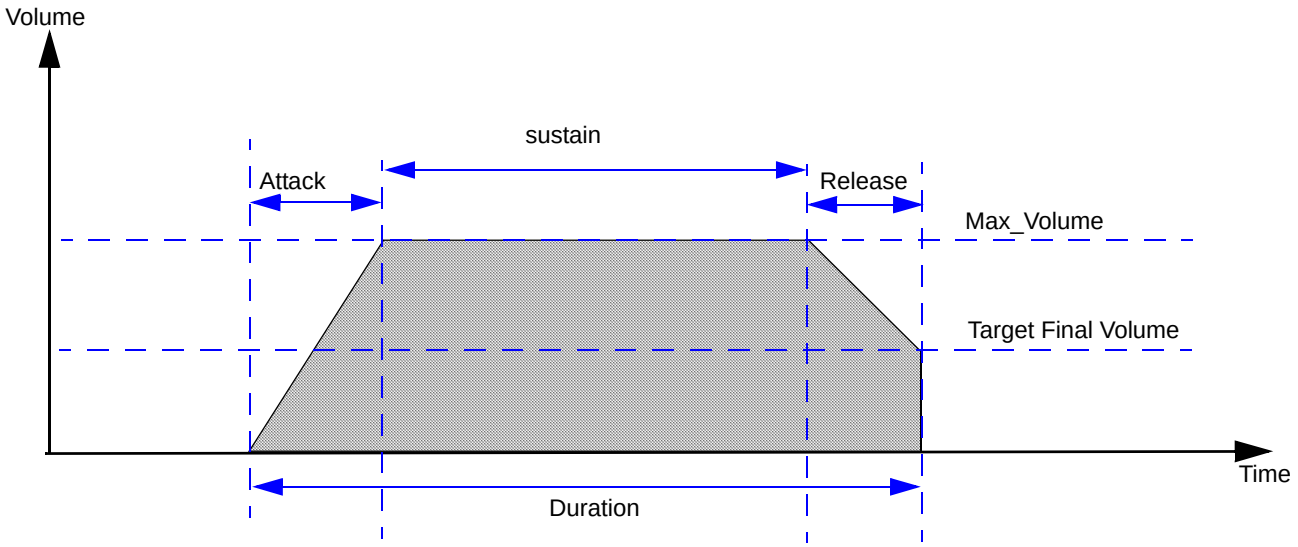



**Figure 39-40. ASR Envelope**

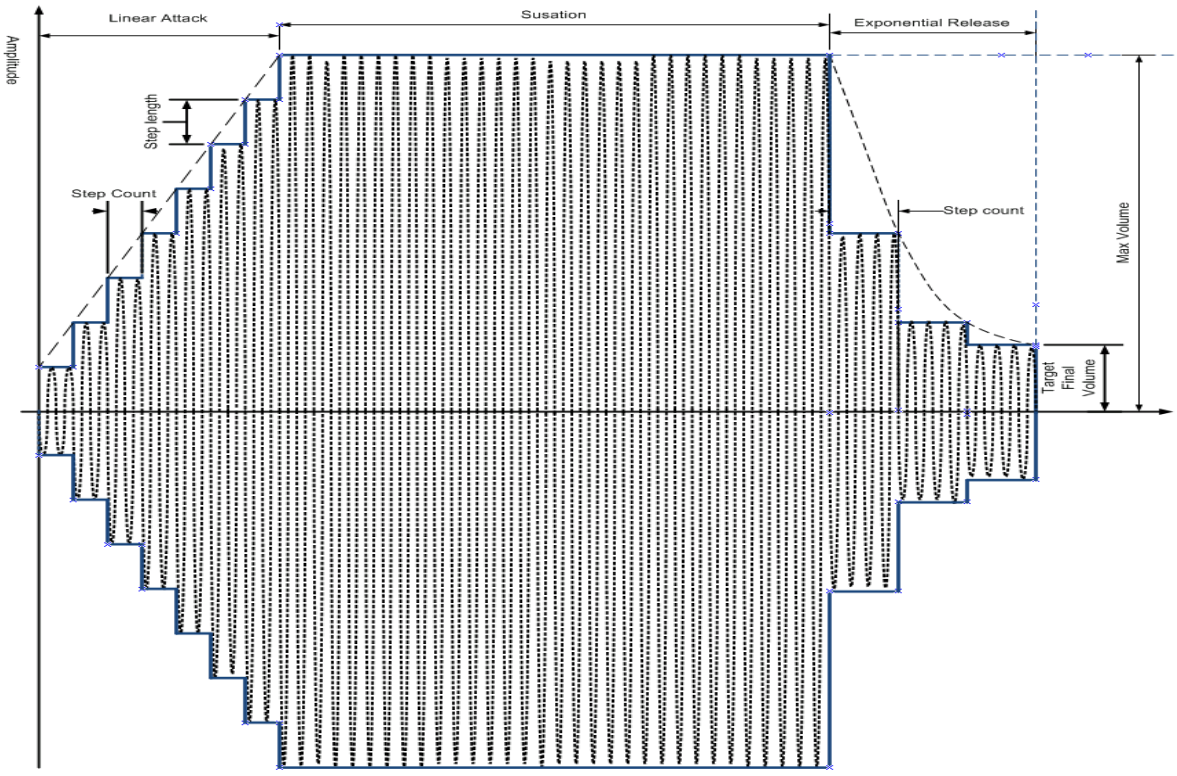



**Figure 39-41. ASR envelope with linear Attack and exponential Release**

The channel allows repetition of the volume envelope (Section 39.6.2.10, “Target Note Pulse Count for Channel 3 (TPCCH3)) and the insertion of a delay between repetitions of the wavetable (Section 39.6.2.9, “Inter-Note No-Output Phase Timing for Channel 3 (NTCH3)).

Figure  gives an example of the DDS mode in action.

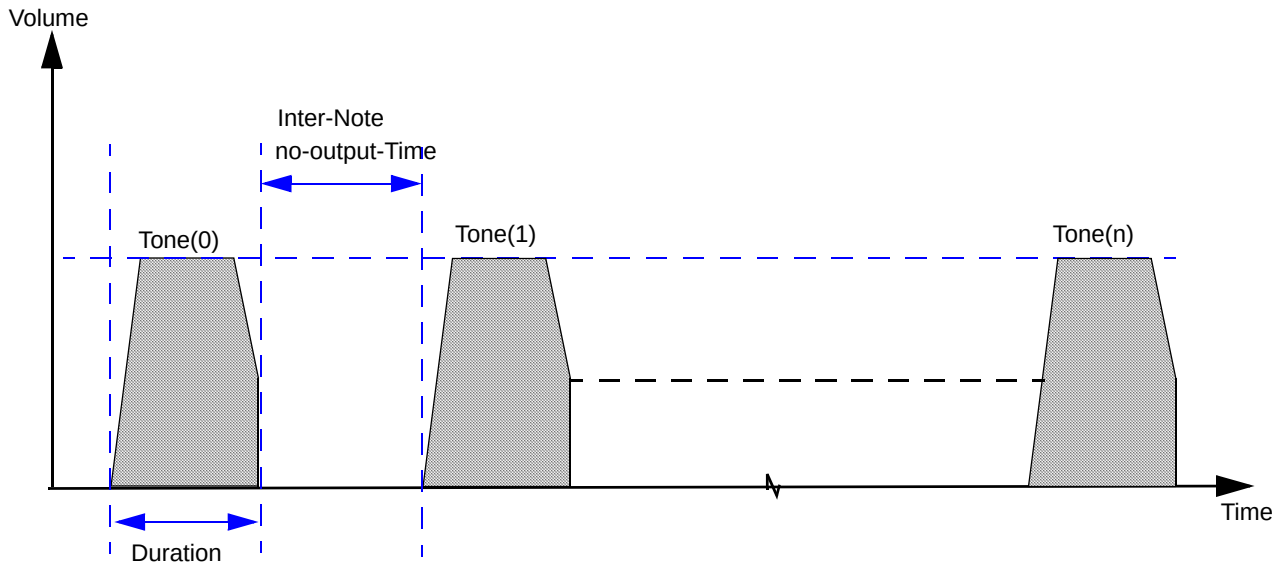



**Figure 39-42. DDS Mode for one Channel**

### 39.7.2.4 Updating the configuration buffers

In DDS mode, the configuration registers for DDS mode, SGMCFG (upper word), DDSCH3/2/1/0, ECRACH3/2/1/0, ECRRCH3/2/1/0, ECRSCH3/2/1/0, NTCH3/2/1/0, TPCCH3/2/1/0, are equipped with hardware buffers.

If sound generation hasn’t started, the buffer values will be updated from the configuration registers every IPS clock cycle.

When a note is being generated, the values of the configuration registers are copied to the buffers when the following sequence is detected:

1. Configure the 1st note before starting the generation.
1. Setting the start of generation bit (SOGCHx bit in SGMCTL).
2. Buffers are updated at next IPS clock.
3. Synchronous reload occurs :
    - at the end of Release phase if the no-output phase insertion is disabled.
    - at the end of no-output phase, if the no-output phase insertion is enabled.

Up to the next reload, configuration registers and buffers can hold different values. The following figure shows an example (not to scale).

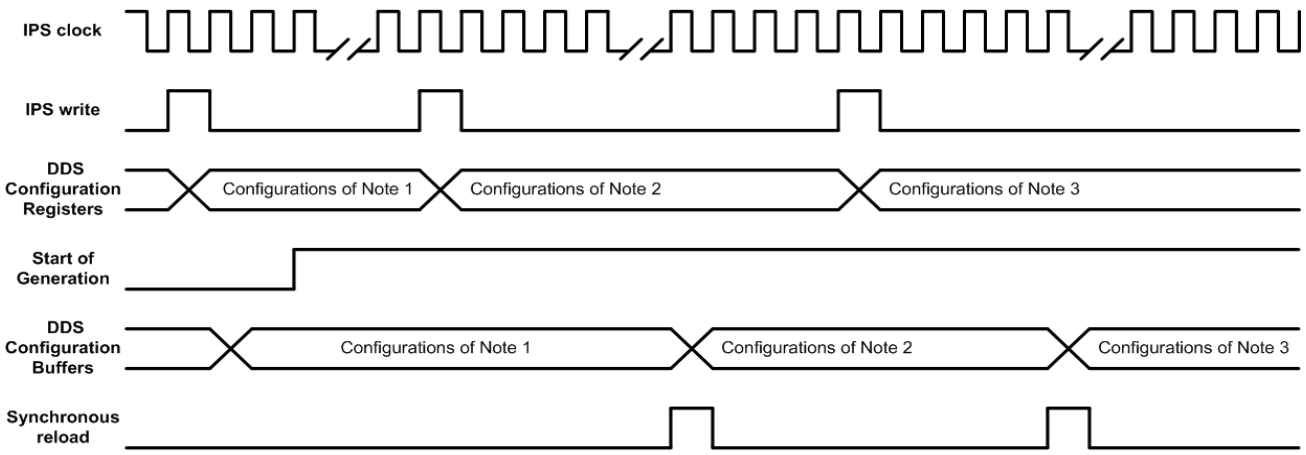



**Figure 39-43. Updating the DDS configuration buffer**

### 39.7.2.5 State Machine of SGM channel in DDS mode

The SGM manages the operation of the channel using a state machine. The current state of the machine for each channel is found in the SGMST register. See Figure  and Table 39-41 for details of the state machine operation.

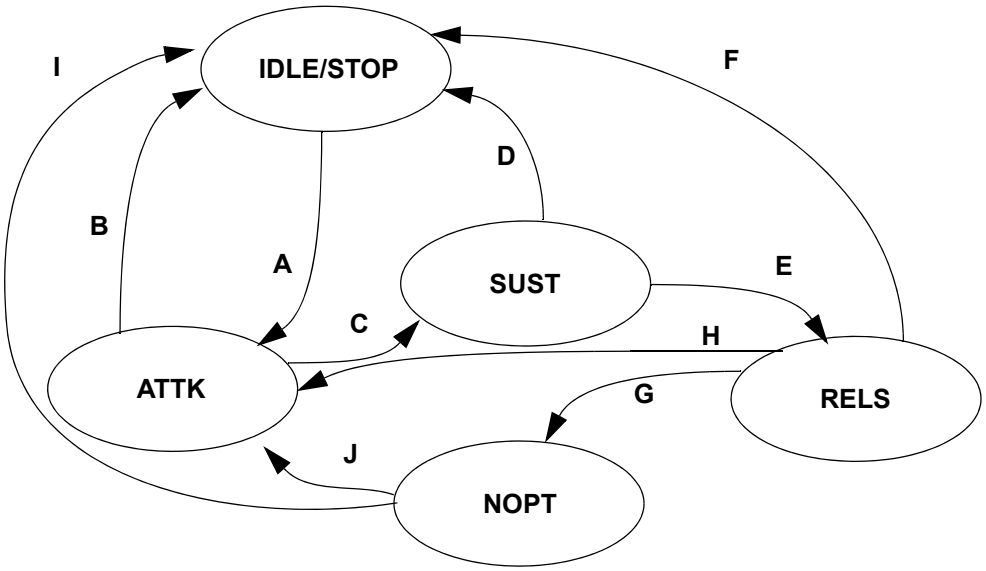



**Figure 39-44. DDS-Mode State Machine - by Function**

**Table 39-41. DDS mode State Machine**

| Source State | Destination State | Transfer Condition | Description |
|---|---|---|---|
| IDLE/STOP | ATTK | A | If the SOG is asserted, the ASR envelope will enter the Attack phase. |

Table 39-41. DDS mode State Machine (continued)

| Source State | Destination State | Transfer Condition | Description |
|---|---|---|---|
| ATTK | IDLE/STOP | B | The generation of the sound can be stopped by deasserting the SOG anytime. |
| ATTK | SUST | C | If the ASR envelope reach the target volume of Attack phase, Sustain phase will be entered. |
| SUST | IDLE/STOP | D | The generation of the sound can be stopped by deasserting the SOG anytime. |
| SUST | RELS | E | When the target sustain timing is reached, the Release phase will be entered. |
| RELS | IDLE/STOP | F | The generation of the sound can be stopped by deasserting the SOG anytime.Or if the no-output phase is disabled, the note pulse count is enabled and the target note pulse number is reached, the IDLE phase will be entered. |
| RELS | NOP | G | If the ASR envelope reach the target volume of Release phase and the no-output phase is enabled, the no-output phase will be entered. |
| RELS | ATTK | H | If the ASR envelope reach the target volume of Release phase and the no-output phase is disabled, the no-output phase will be entered. |
| NOPT | IDLE/STOP | I | The generation of the sound can be stopped by deasserting the SOG anytime.<br>Or when the target note pulse number is reached (if the target note pulse count is enabled), the generation ends. |
| NOPT | ATTK | J | When the target no-output timing is reached, the Attack phase will be entered if the target note pulse number isn't reached or disabled. |

## 39.7.3 SGM architecture

The SGM consists of a clock configuration logic, IPS Interface, four identical sound channels, a re-sampling block, mixer with volume control, PWM output and I2S interface. Refer to  for a block diagram of the SGM. The following sections discuss each in turn.

## 39.7.4 SGM clocking

The SGM uses the following clocks:

Source clocks:

- System clock
- SGM module clock (maximum frequency half the system clock)

Generated clocks:

- Resampling clock -- Used to resample the data from all the 4 channels with the same expected sample rate.

**NOTE**

In Wave mode, the PCM data from 4 channels should have the same sample rates. Use the SGMCTL[WAVCLKS] bit to select the resample clock as the channel clock of all wave-mode channels. In DDS mode, the reference clock for each DDS can be different. Before mixing the channels together, they should be re-sampled by the expected sample rate.

- ch0_clk - Used to clock the channel 0.
- ch1_clk - Used to clock the channel 1.
- ch2_clk - Used to clock the channel 2.
- ch3_clk - Used to clock the channel 3.
- pwm_clk - Used to clock the PWM block.
- ton_clk - Used to clock the rate of change of mixer output with PWM.

## 39.7.5 Channel controller

There are four channels available in the SGM. Each channel consists of:

- A DDS controller with an ASR envelope controller
- A Sample Format Converter (SFC)
- Volume control logic

### 39.7.5.1 DDS Controller

The DDS controller produces DDS (wavetable synthesis) using an accumulator, the channel FIFO and a volume control managed by the envelop controller. See

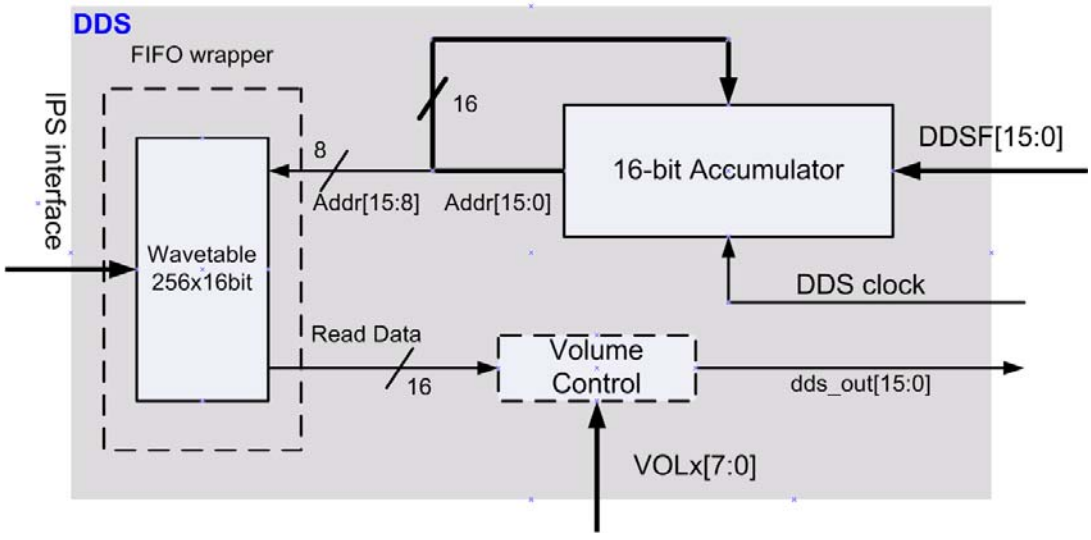



**Figure 39-45. DDS Architecture in SGM**

The 16-bit accumulator is incremented by the channel clock and generates the address to read the samples from the FIFO. Only the upper 8 bits are used to select the address to the FIFO. The DDSF value provides the increment of the 16-bit accumulator and so can be used to determine both the frequency of change of sample as well as the sequence of samples:

- A value of 0x0100 will output sequential samples from the wavetable at a rate of one per DDS clock cycle
- A value of 0x0200 will output every second sample from the wavetable at a rate of one per DDS clock cycle
- A value of 0x0001 will output sequential samples from the wavetable at a rate of one every 256 DDS clock cycles
- A value of 0x00C0 will output the sequence sample 0, sample 1, sample 2, sample 3, sample 3, sample 4, sample 5, sample 6, sample 7, sample 7 and so on at the DDS clock rate

Since only the upper 8 bits of the accumulator are used to address the wavetable, the sample rate is:

$$F_{ddsout} = \frac{F_{ddsclk}}{2^{16}} \times DDSF$$

The repetition frequency of the wavetable contents is given by:

**fwave = Fddsclk/DDSF** ***Eqn. 39-1***

### 39.7.5.2 ASR Envelope Controller

The SGM provides a simple, configurable ASR envelope, which is a simplified ADSR envelope. See Figure . The envelope controller controls the volume of each sample output from the DDS Controller.

### 39.7.5.3 Sample Format Converter

In Wave mode, linear mono PCM data coded in 8-bit, 12-bit and 16-bit are supported.

The Sample Format Converter (SFC) converts data coded in 8 or 12 bit into 16-bit linear PCM data (represented by two’s complement).

### 39.7.5.4 Volume Control

The Volume Controller is used to adjust the volume of the sound. For each channel, the volume can be controlled individually. A fixed-point multiplier is applied to each channel to compute the required sample value for the resampler and mixer.

For Wave mode the volume is directly controlled by the volume control register for each channel. For DDS mode, the volume is controlled by the ASR envelope controller.

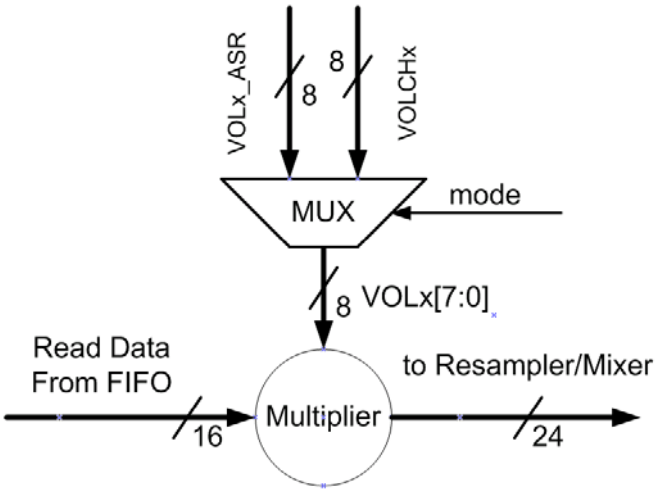



**Figure 39-46. Volume Controller for One Channel**

## 39.7.6 Re-sampling Block

The re-sampling block is used to unify the sample rates for all channels before entering the mixers. For DDS mode, different sample clocks can be used for each channel and the re-sampling clock can be used to choose the current sample value on each channel as its output. For Wave mode, the algorithm for SRC(Sample Rate Conversion) would be very complicated with linear/cubic interpolation and therefore all wave mode channels must share a sample clock.

## 39.7.7 Mixer

The mixers are used to mix the data from the four channels together and generate the PCM outputs for the external audio DAC and/or PWM (as the duty). Two mixers, right mixer and left mixer, are provided to generate the PCM data for 2 output channels: Channel-R and Channel-L.

Each mixer can be configured to mix any 1 to 4 of all 4 input channels together. Consider, for example, channel-0 and channel-1 in Wave mode and channel-2 and channel-3 in DDS mode. The left mixer can be configured to mix channel-0 and channel-3 together while the right mixer can be configured to mix channel-1and channel-2 together.

The mixer implements an arithmetic add for all the 4 input data (represented by 2' s complement) and a division with the rule below:

If 1 channel is selected,  division by 1;

If 2 channels are selected, division by 2;

If 3channels are selected, division by 4;

If 4 channels are selected, division by 4;

### 39.7.8 I2S Interface

It is possible to select either PWM or I2S options for the output of the mixed sample data. The output option chosen does not affect the hardware or process for creating the mixed sound. This section describes the I2S block.

#### 39.7.8.1 Features

The I2S interface has the following features:

- Synchronous Master mode support only
- Output sample support
  - 16-bit per channel
  - 32-bit per channel
  - 16-bit encapsulated in 32-bit
- Same data can be repeated on the both left and right channels
- Supported protocol modes
  - Philips (stereo)
  - I2S MSB Justify (stereo)
  - I2S LSB Justify
  - PCM (mono)
- For PCM mode
  - Only 16-bit input audio data is supported
  - Output data supported format is 16-bit or 16-bit extended 32-bit
  - Programmable frame sync width of 1 to up to 13-bit clocks i.e. support for long and short frame synchro
- For extended bit formats, following data repetition options are supported
  - Append sign bits (For LSB justified modes)
  - Repeat MSBits (for MSB justified modes)
  - Stuff 0's (all extended modes)
  - Stuff 1's (all extended modes)
- Additional clock output port support with programmable frequency of
  - 256 × Frame clock
  - 512 × Frame clock
- Programmable polarity for clock, data and frame synchronisation signals
- 16-bit pre-scaler to support different SCK clock frequency
- 8 word deep (32-bit wide) fifo for buffering audio data output data
- Programmable FIFO thresholds to start I2S data transfer

- FIFO underrun and transfer complete interrupts

### 39.7.8.2 Clock Choices

When no dedicated clock sources are provided for the sample rate of 44.1/22.05/11.025 kHz and 48/24/8KHz on this device, the solution below is used to minimize the sample rate variation.

- SGM prescalers will be programmed in such a way, that its sampling frequency is always slightly higher than I2S
- Once enabled, I2S will keep asking for data from SGM. I2S maintains the actual sample rate control, so its prescaler should be accurately programmed
- An Underrun (I2S needing data, but SGM not able to process that fast) will never happen

### 39.7.8.3 I2S Block Diagram

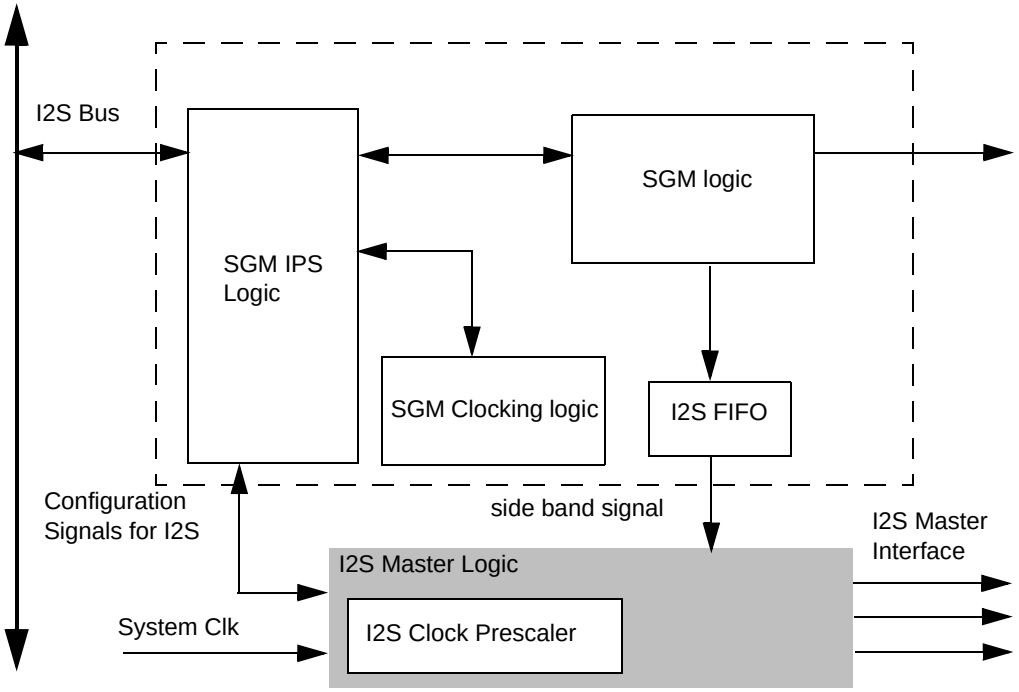



**Figure 39-47. I2S block diagram**

### 39.7.8.4 Supported Protocol modes

This section describes the I2S modes supported by the interface. Configuration of the protocol modes are done using the control and output data format registers (Section 39.6.2.30, “I2S Control Register (I2SCTL) and Section 39.6.2.31, “I2S Output Data Format Control Register (I2SDFC)).

### 39.7.8.4.1 Philips Mode

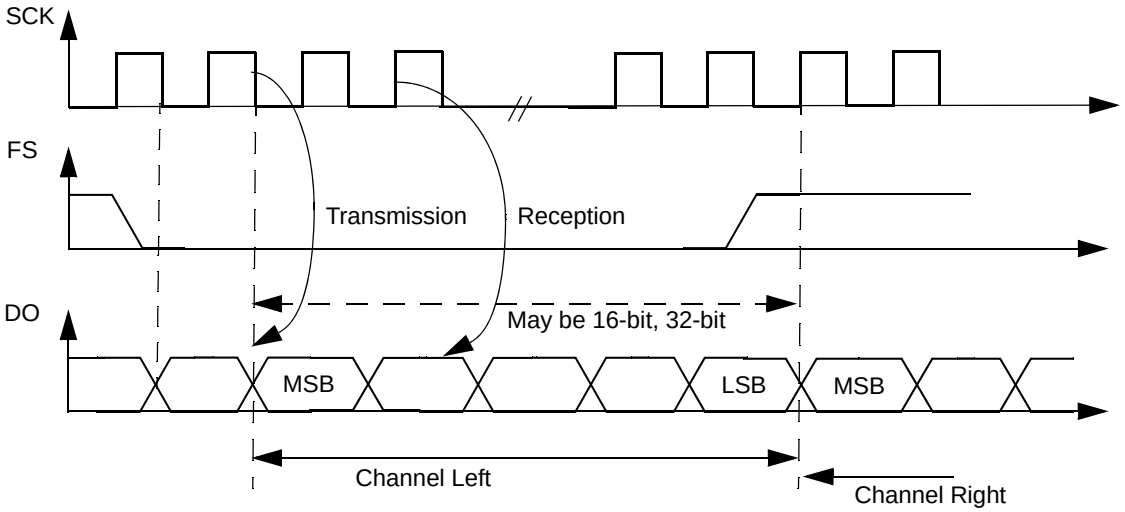



**Figure 39-48. Philips Mode 16/32-bit**

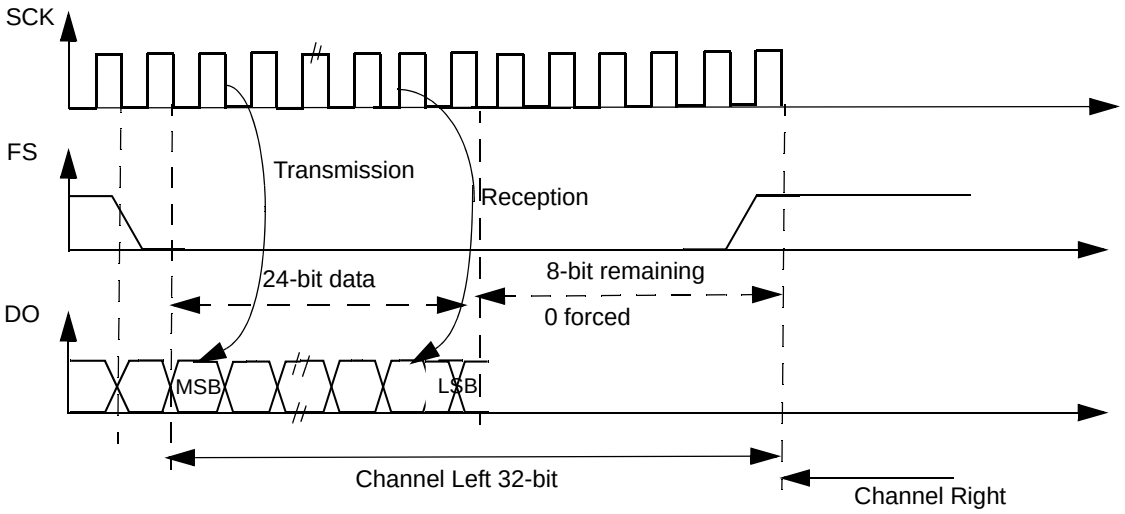



**Figure 39-49. Philips Mode 24-bit**

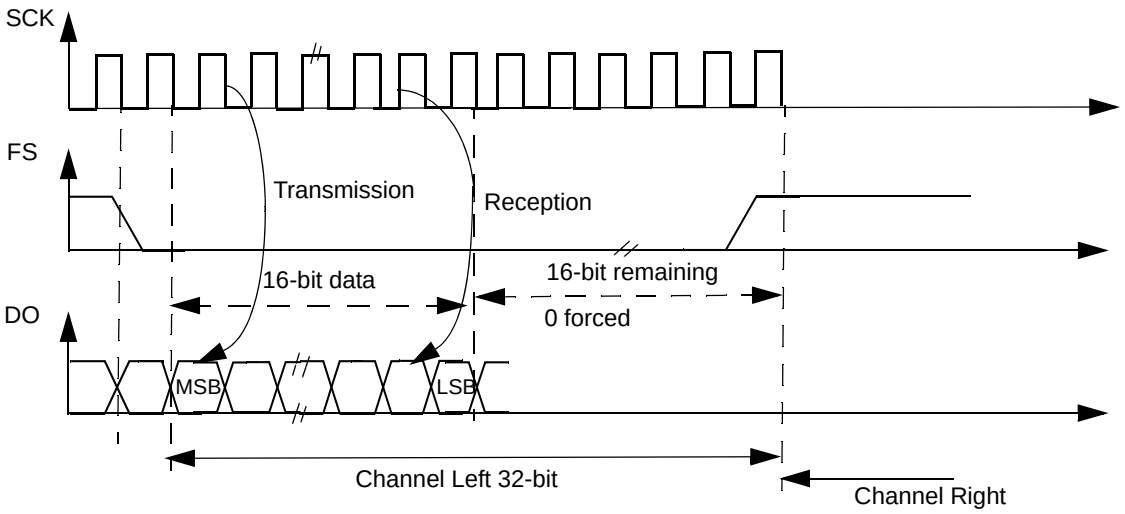



**Figure 39-50. Philips Mode 16-bit extended to 32-bit**

### 39.7.8.4.2 MSB Justify Mode

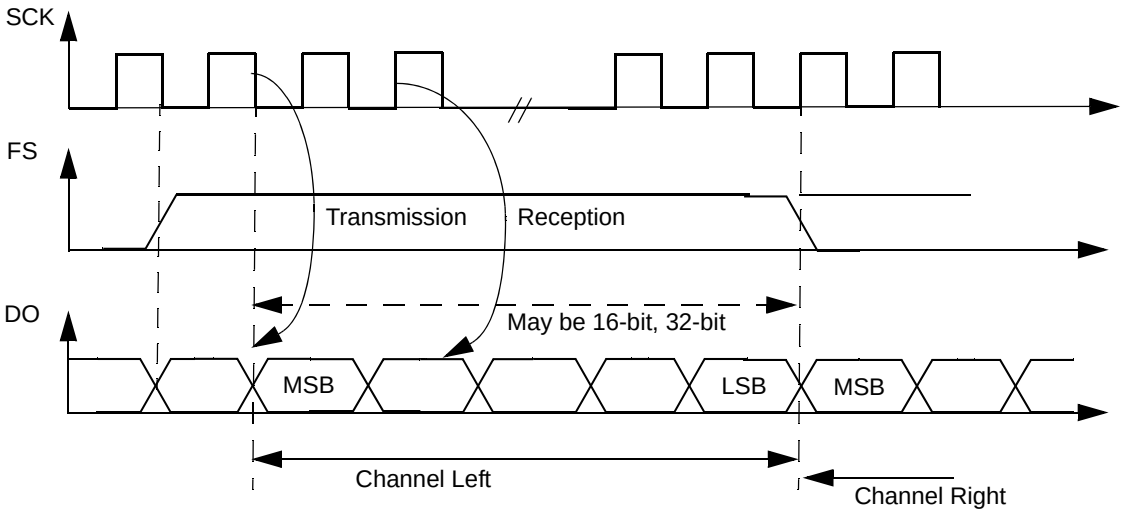



**Figure 39-51. 16/32 bit**

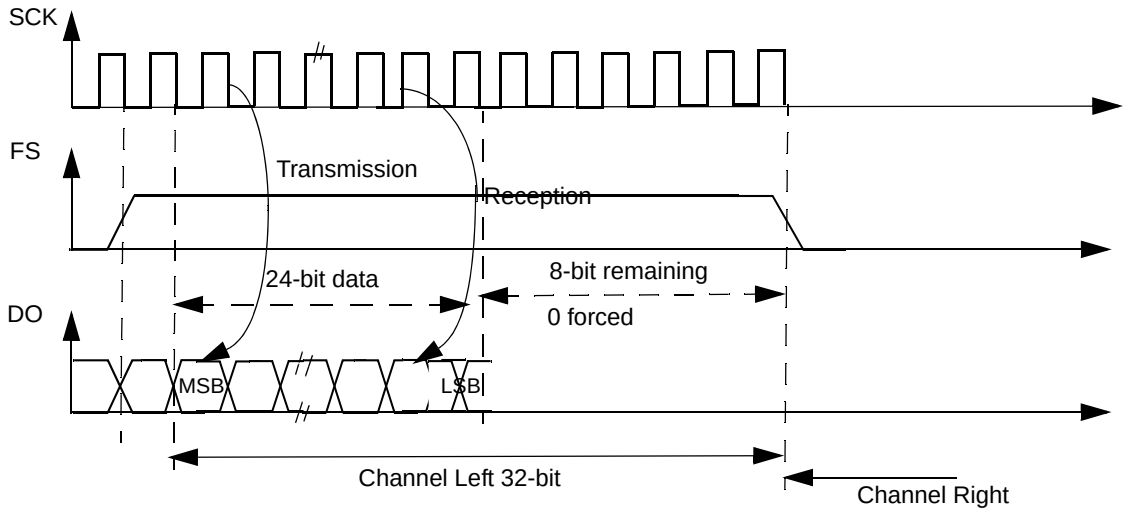



**Figure 39-52. 24-bit extended to 32-bit**

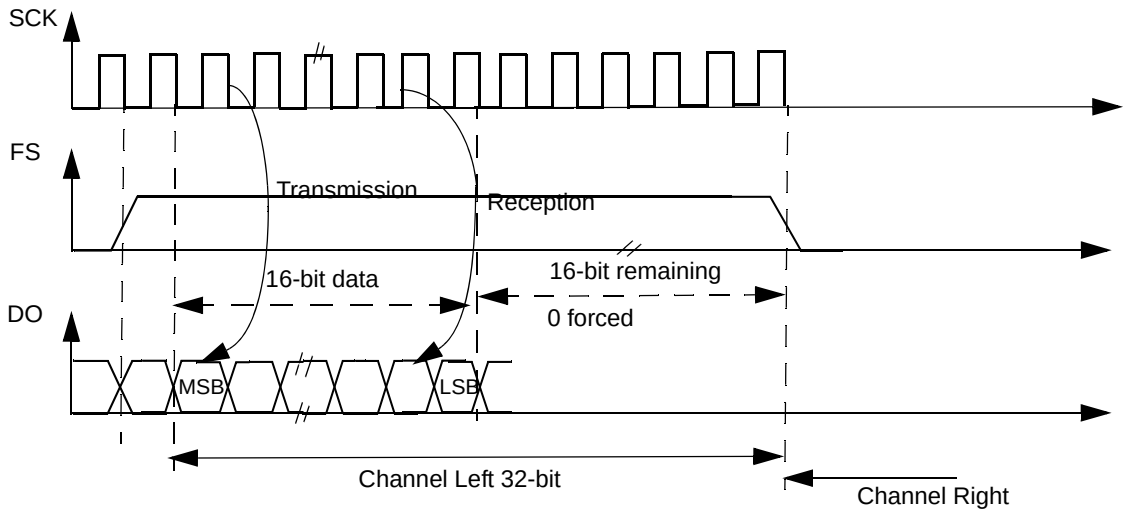



**Figure 39-53. 16-bit extended to 32-bit**

### 39.7.8.4.3 LSB Justify Mode

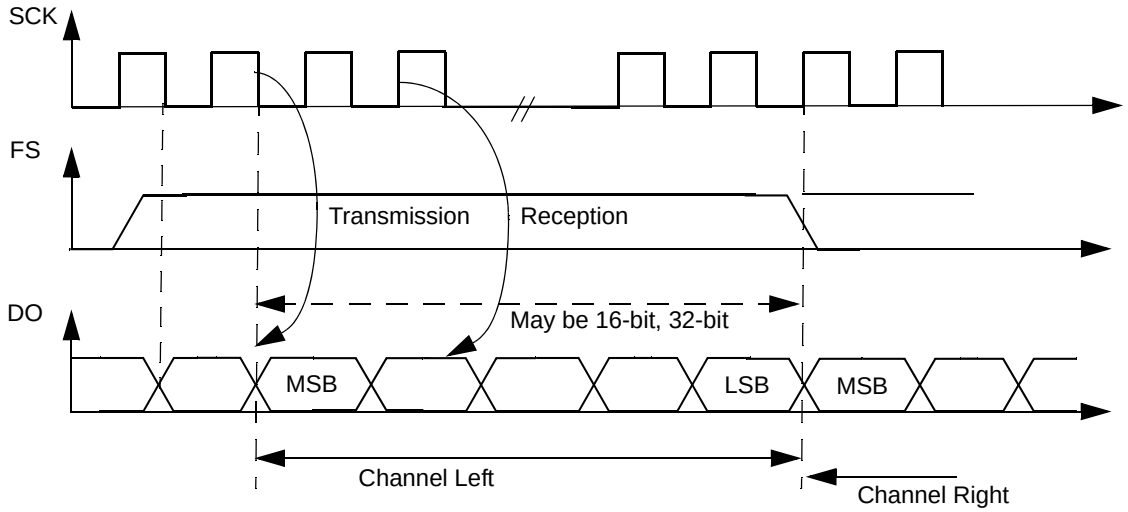



**Figure 39-54. 16/32bit**

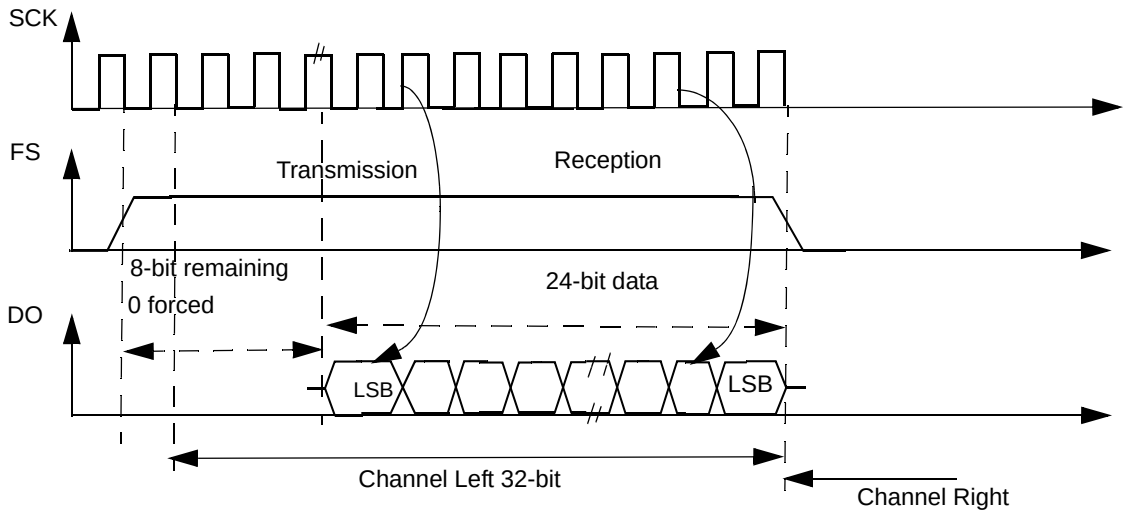



**Figure 39-55. 24-bit extended to 32-bit**

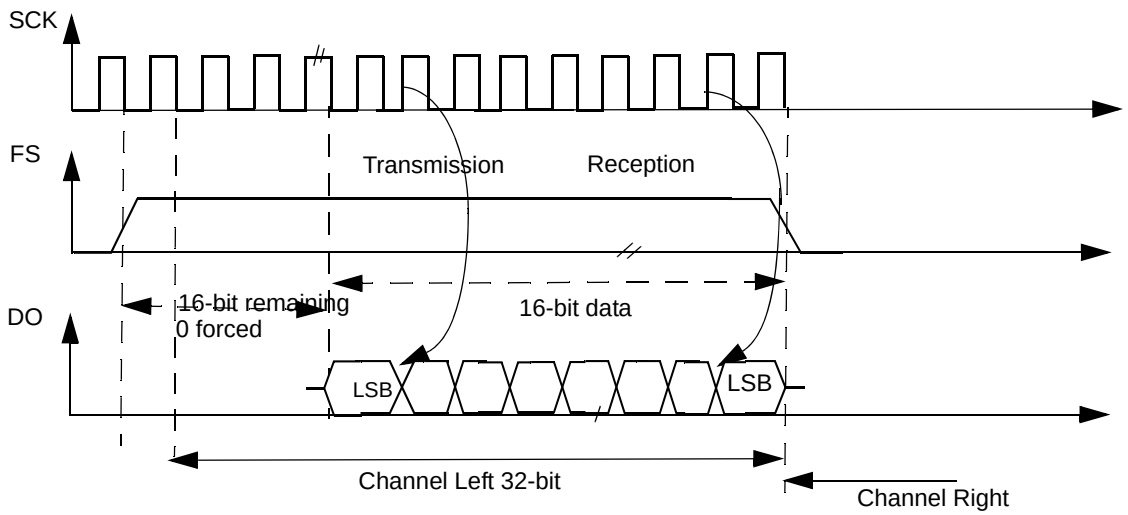



**Figure 39-56. 16-bit extended to 32-bit**

### 39.7.8.4.4 PCM Mode

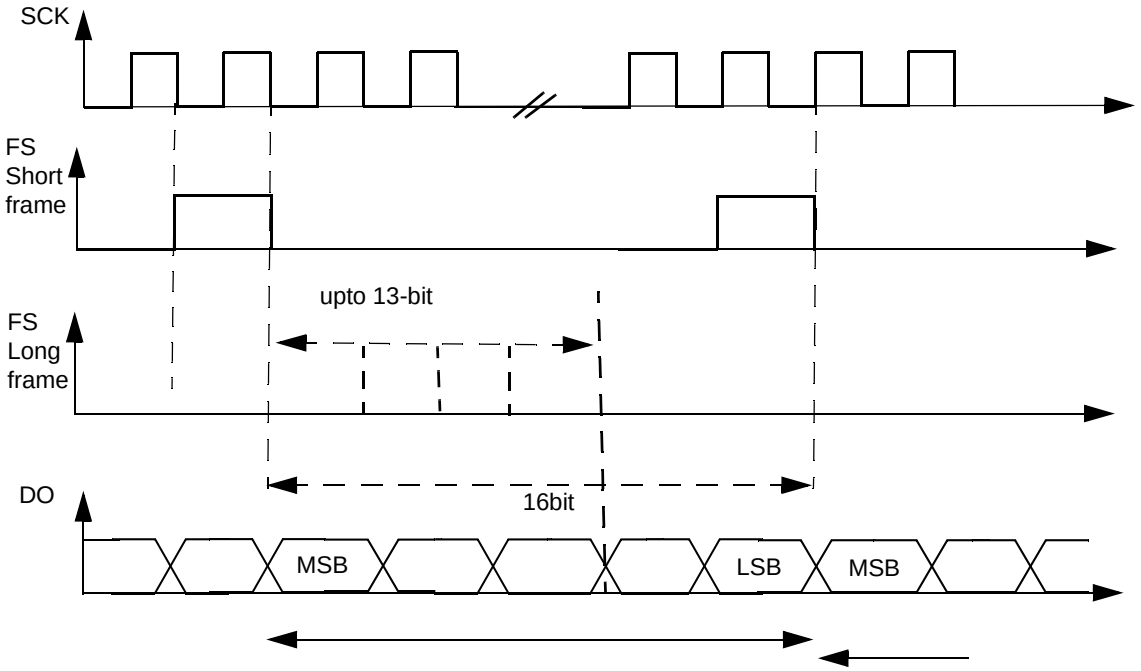



**Figure 39-57. 16-bit extended to 32-bit**

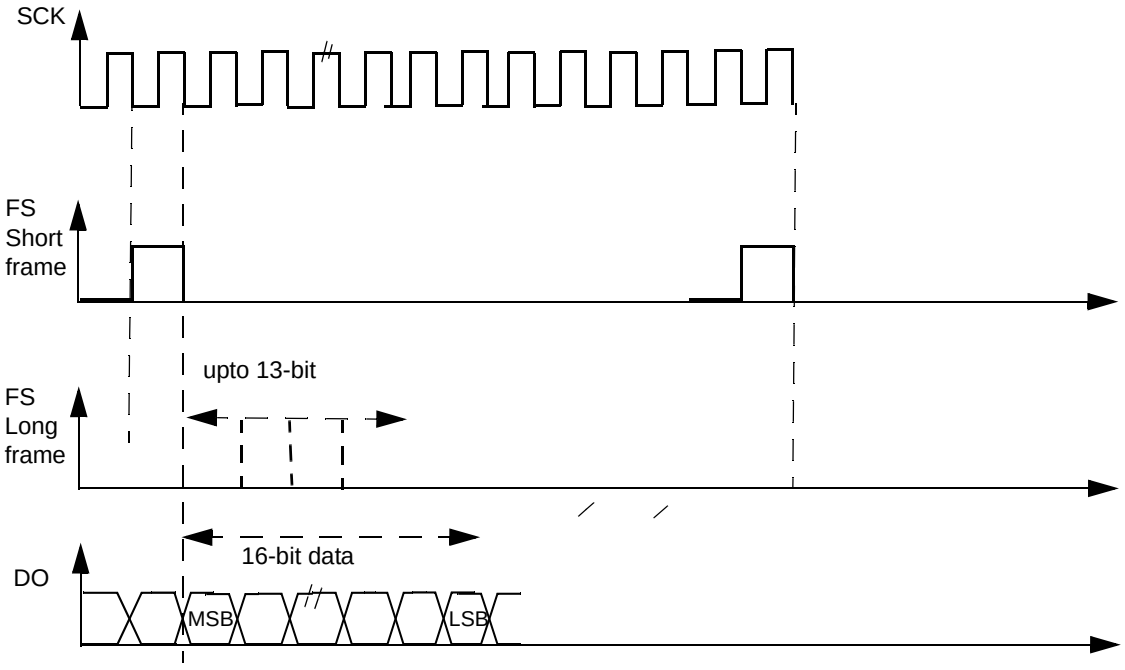



**Figure 39-58. 32-bit**

## 39.7.9 PWM Output

As an alternative to the I2S interface it is possible to have the output of one of the mixer channels act as the source of the duty cycle of a PWM. The duty cycle of the PWM changes with each output sample and the output signal can be filtered and integrated to provide a low cost DAC.

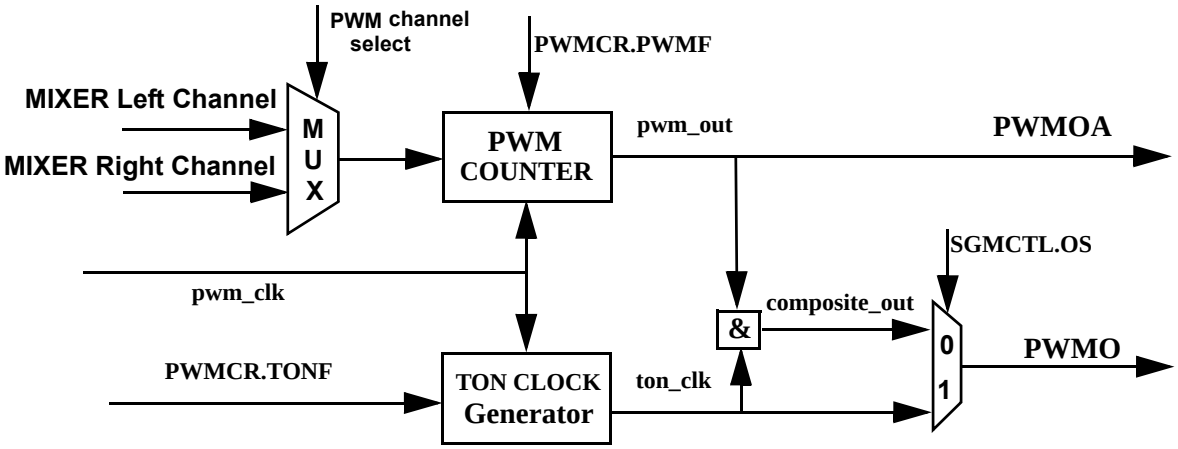



**Figure 39-59. PWM Block**

The PWM output is designed to provide a simple digital output that can be filtered to reproduce the analog output of either one of the mixer channels. Reproducing the highest quality sound is not possible for the PWM due to the very high PWM frequencies that would be required. Therefore for most applications the PWM is limited to a lower quality sound output. When the PWM output is enabled the I2S is disabled.

The mixed data values from either the left mixer or the right mixer are used as the count of the duty cycle of the PWM (PWMCHS bit in Section 39.6.2.2, “SGM Configuration Register (SGMCFG)). The total length of the PWM count is defined by the PWMCR[PWMF] field and the frequency of the PWM is defined by the PWM clock (Section 39.6.2.18, “PWM Configuration Register (PWMCR)).

A 100% duty cycle (continually high) will be generated if the value of the mixed data is higher than the PWMF count. The amplitude of the filtered output is determined by the duty cycle of the PWM. The frequency of the output is decided by the channel sample rate.

The PWM frequency is generated by PWMF and the PWM clock. It can be calculated as:

$$f_{PWM} = \frac{f_{pwmclk}}{([PWMF] + 1)}$$

The duty cycle of PWM signal is calculated as follows:

- If [Mixer_out] > [PWMF]: Duty cycle =100%
- If 0 ⩽ [Mixer_out] ⩽ [PWMF]: Duty cycle = [Mixer_out] / ([PWMF]+1)

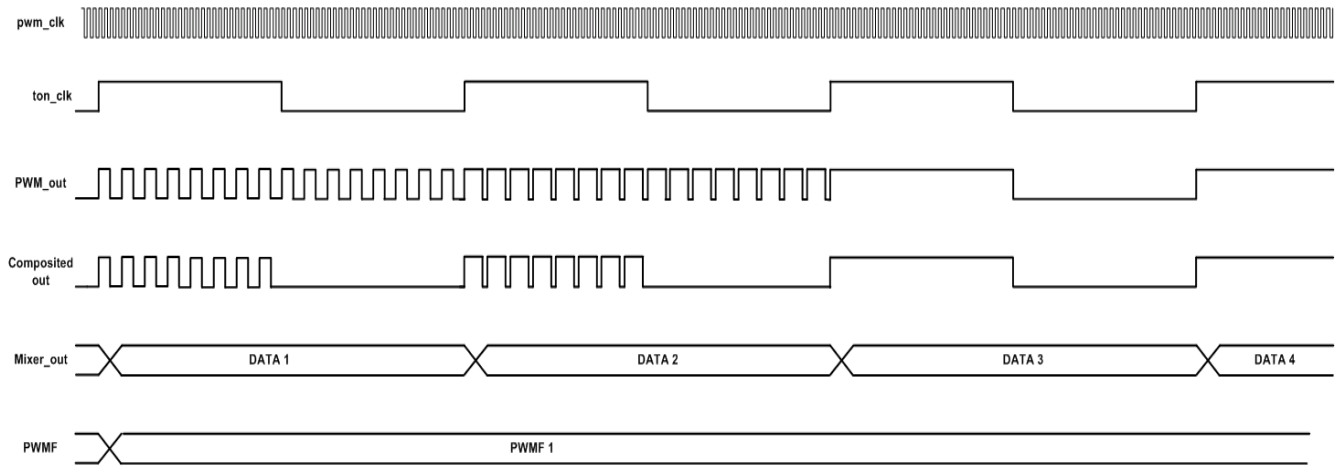



**Figure 39-60. Generation of the PWM Composite output signal**

## 39.8 Interrupts and DMA

All of the SGM interrupt sources are ANDed together into a single interrupt vector. There are interrupt sources related to the operation of each channel, the FIFO and DMA status and the I2S interface.

The channel interrupts and DMA are as configured and recorded in the SGM interrupt registers (Section 39.6.2.26, “SGM Interrupt Control Register (SGMIC) and Section 39.6.2.28, “SGM Interrupt Status Register (SGMIS)).

Enable the DMA for Wave mode channels using the SGMICDF[FLDECHn] bits. The status flags SGMISDF[FLFCHn] indicate that a DMA request has been made. Note that the DMA will clear this bit when it acknowledges the request.

There is a general timeout flag to indicate expiration of timeout counter. The counter is enabled using the SGMCTL[TOE] bit and the interrupt control flag SGMIC[TOIE].

Interrupt sources when channels are in Wave mode:

- Playback duration over (end of each playback cycle)
- Repeat duration over

Interrupt sources when channels are in DDS mode:

- End of release phase
- End of attack phase
- End of inter-note no-output phase
- End of pulse count

The FIFO and DMA interrupts are configured and recorded in the DMA and FIFO interrupt registers (Section 39.6.2.25, “SGM Interrupt Control Register for FIFO and DMA (SGMICFD) and Section 39.6.2.27, “SGM Interrupt Status Register for FIFO and DMA (SGMISFD))

The FIFO interrupts are as follows:

- FIFO Programmable Level Flag (note that this flag is automatically cleared by an active DMA)
- FIFO full flag
- FIFO empy flag
- FIFO underflow flag
- FIFO overflow flag

## 39.9 Initialization and application information

This section provides some use cases to give some basic direction to a user on how to initialize and configure the SGM module.

### 39.9.1 Wave Mode Use Cases

The typical use cases for this function are:

- Speech (single shot mode)
- Alarm sound (Repeat mode)

#### 39.9.1.1 Speech

##### 39.9.1.1.1 Description

The application requires to playback a PCM wave which can have a duration up to 10s (5s typical). The quality can go up to 48kHz (22kHz typical) sampling rate and the samples can be up to 16-bit (12-bit typical).

According to these requirements, a mono linear PCM file of about 1MByte (about 170kByte typical) need to be stored in the system and the SGM will have to fetch each of the 480,000 samples (110,000 typical) stored in the wave and send them out to the speaker/amplifier at the sampling frequency rate.

##### 39.9.1.1.2 SGM functionality

The only information the SGM needs to know are

- the number of samples
- the sample data width
- the sampling rate

The number of samples and the sampling data width will be used by the channel to issue all the DMA requests and fill the input FIFO. The FIFO control is transparent to the user. The number of samples, the sampling data width and the sampling rate will be used by the mixer to consume the data in the FIFO.

The user will have to program the DMA TCD to fetch the data from the address of the wave in the system memory and enable the DMA channel.

### 39.9.1.2 Alarm sound

#### 39.9.1.2.1 Description

The application will playback a repetitive sound with a blank period in between.

The duration of a single sound can go up to 5s (typical 2.5s), the duration of the blank period should be the same, sampling rate and sample width are identical to the voice test case.

The duration of the repetition can be or not programmable in number of cycles (1 cycle being the sound+the blanking time).

As an example, we can imagine an indicator noise.

#### 39.9.1.2.2 SGM functionality

The user programs the SGM sample rate, the sample width and the number of samples for the sound and the duration of the deadtime. The user also programs the SGM to have it as free running (repeating until the user will stop it) or with a limited duration (repeat number).

The user will program inside the DMA the address of the sound in the system memory and enable the channel.

**Table 39-42. Work flow for Alarm Sound in WAV mode**

| CPU (main) | CPU(ISR) | SGM |
|---|---|---|
| 0. Clear the SGMCTL.MDIS to enable SGM | | |
| 1. Configure the DMA descriptor to move the PCM data from memory to FIFO | | |
| 2. Check the SGMST. If the FIFO of current channel is ready(FLSCHx=0), configure the FIFO watermark. Then enable the DMA request | | |
| | | DMA controller will move the PCM data from memory to SGM FIFO |
| 3. Configure the SGM<br>• Clocking/ Wave mode<br>• Enable the Repeat mode, configure the repeat number and Sample Format<br>• *Enable the Playback Duration interrupt and repeat duration interrupt (optional)<br>• check the FIFO status. If the fifo is not empty, enable the SGMCTL.SOGCHx | | Start the sound playback<br>Generate the 1st repeat |
| | DMA transfer completion ISR: configure the DMA descriptor of moving the next part PCM data from memory to FIFO. | ..... |
| | ..... | *At the end of 1st PlayBack duration, SGM will raise the interrupt of PlayBack Duration. (optional) |
| | | DeadTime<br>... |

Table 39-42. Work flow for Alarm Sound in WAV mode (continued)

| CPU (main) | CPU(ISR) | SGM |
|---|---|---|
| | | Generate the 2nd repeat |
| | | ... |
| | | ... |
| | | Repeat number is reached.<br>*SGM will rise the repeat duration interrupt .(optional)<br>SGM clear the SGMCTL.SOGCHx. |
| **To Stop the current PlayBack or DeadTime:** | | |
| 4. Clear the SGMCTL.SOGCHx. | | |
| 5. Check the SGMST. Wait until the operation status is IDLE(STATCHx=3'b000) | | |
| 6. Disable the DMA request. | | |
| 7. Write 0/1 to FIFO read/write pointer of this channel. Clear the FIFO and Flush the DataPath. | | |
| 8. Check the SGMST. Wait until the FIFO of current channel is ready(FLSCHx=0). | | |
| 9. Configure the DMA descriptor of move PCM data from memory to FIFO for WAV another PlayBack-DeadT sequence | | |
| 10. enable the DMA request. | | |
| | | DMA controller will move the PCM data from memory to SGM FIFO |
| | | ... |
| 11. Configure the SGM (interrupt, clocking ...) | | |
| 12. check the FIFO status. If the fifo is not empty, set the SGMCTL.SOGCHx to enable the PlayBack-DeadT sequence | | |
| | ... | ... |

## 39.9.2 DDS Mode Use Cases

SGM can also generate the sound with a wavetable. The frequency, amplitude and the shape of the notes are configurable.

The typical use cases for this function are:

- Polyphonic Sound
- Polyphonic Alarm sound

### 39.9.2.1 Polyphonic sound

#### 39.9.2.1.1 Description

In this use case the application generates sequences of notes on each channel. When mixed, these generate a polyphonic sound. For every individual note, the duration is defined by the configurable ASR envelope.

#### 39.9.2.1.2 SGM functionality

The SGM needs to be configured with the following information:

- Wavetable
- ASR envelope
- Frequencies of every individual notes, which are defined by the timing configuration of the channel

To generate a sequence of notes using a single channel, enable the interrupt on reaching the configurable target volume level of the release phase. The frequency of each note should be configured by software.

All 4 channels can be configured simultaneously or in some sequence. When all 4 channels are mixed a polyphonic sound is generated.

The figures below shows an example of generating/mixing polyphonic sound. And Table 1-6 shows a work flow of generating a sequence of notes by one channel.

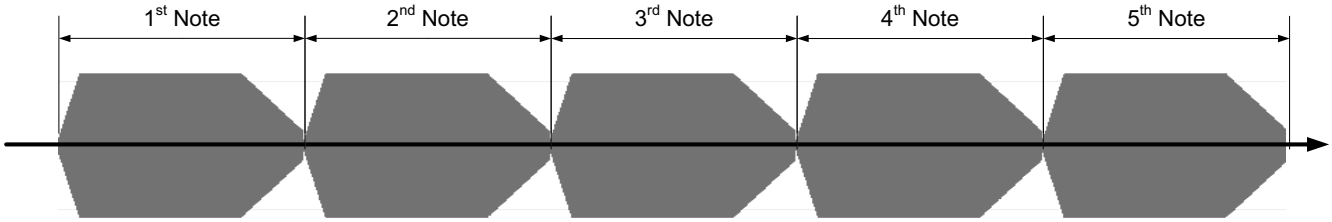



**Figure 39-61. Channel 0**

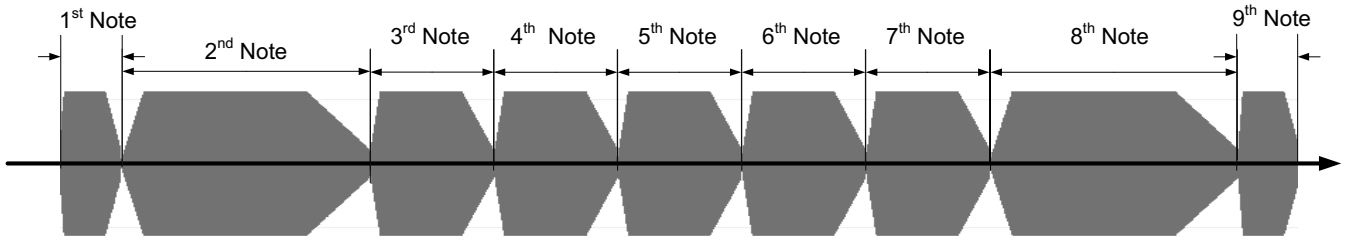



**Figure 39-62. Channel 1**

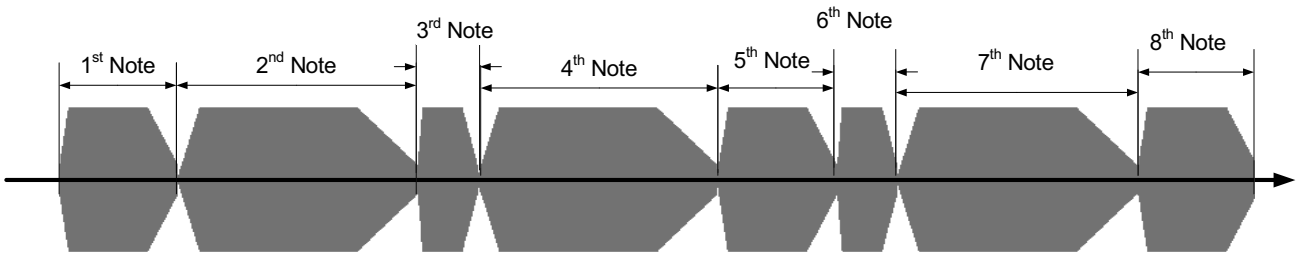



**Figure 39-63. Channel 2**

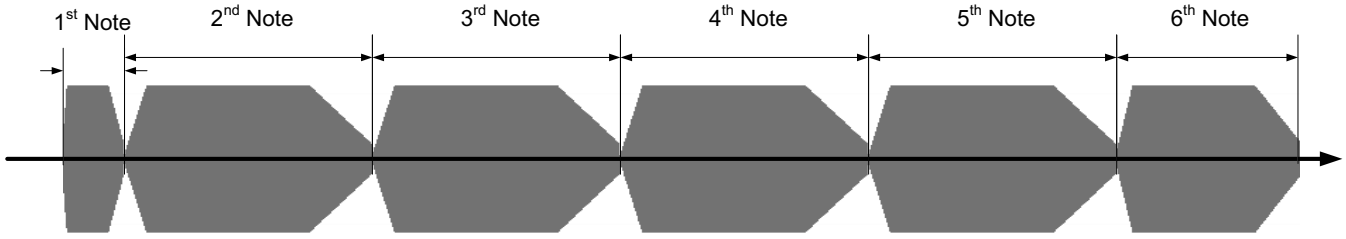



**Figure 39-64. Channel 3**

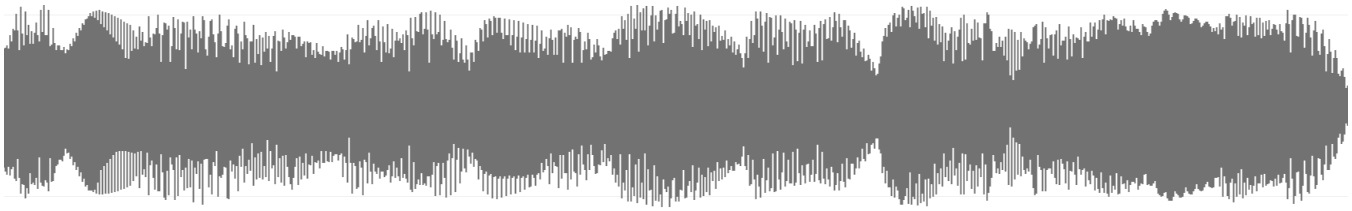


**Figure 39-65. Mixed polyphonic sound**

**Table 39-43. Work flow of generating a sequence of notes by one channel**

| CPU (main) | CPU(ISR) | SGM |
|---|---|---|
| 0. Clear the SGMCTL.MDIS to enable SGM | | |
| 1. Configure the DMA descriptor of move the wavetable from memory to FIFO | | |
| 2. Check the SGMST. If the FIFO of current channel is ready(FLSCHx=0), configure the FIFO watermark. Then enable the DMA request | | |
| | | DMA controller will move the PCM data from memory to SGM FIFO |

**Table 39-43. Work flow of generating a sequence of notes by one channel (continued)**

| | | |
|---|---|---|
| 3. Configure the SGM<br>• Clocking/ DDS mode<br>• ASR envelope<br>• Enable the interrupt when reaching the configurable volume level at Release/no-output Phase<br>• Configure the DDSF (frequency) of 1st note for this channel<br>• Check the SGMST. Wait until the operation status is IDLE(STATCHx=3'b000).<br>• check the FIFO status. If the fifo is Full, enable the SGMCTL.SOGCHx<br>• Configure the DDSF (frequency) and/or other configurations (envelope...) of 2nd note for this channel | | all DDS-configuration are buffered<br><br>Start the Sound generation by DDS. Generate the 1st NOTE<br><br>..... |
| | SGM ISR:<br>Configure the DDSF (frequency) and/or other configurations ( envelope...) of the 3nd note for this channel | When reaching the target volume at end of Release/no-output Phase of the 1st pulse (note), raise the interrupt and synchronously reload the buffered configurations of the 2nd note. |
| | | Generate the 2nd note |
| | | ..... |
| | SGM ISR:<br>Configure the DDSF (frequency) and/or other configurations ( envelope...) of the 4nd note for this channel | When reaching the target volume at Release/no-output Phase of the 2nd pulse (note), raise the interrupt and synchronously reload the buffered configurations of the 3rd note. |
| | | ..... |
| | ..... | ... |
| | | |
| **To Stop the current DDS Generation:** | | |
| 4. Clear the SGMCTL.SOGCHx. | | |
| 5. Check the SGMST. Wait until the operation status is IDLE(STATCHx=3'b000) | | |
| * 6. Disable the DMA request. (optional) | | |
| *7. Write 0/1 to FIFO read/write pointer of this channel. Clear the FIFO and Flush the DataPath. | | |
| *8. Check the SGMST. Wait until the FIFO of current channel is ready(FLSCHx=0). | | |
| *9. Configure the DMA descriptor of move another wavetable from memory to FIFO | | |
| *10. Set the watermark. Enable the DMA request. | | |
| | | * DMA controller will move the PCM data from memory to SGM FIFO (optional) |
| | | ... |

**Table 39-43. Work flow of generating a sequence of notes by one channel (continued)**

| | | |
|---|---|---|
| 11. Configure the SGM (interrupt, clocking ...) | | |
| 12. check the FIFO status. If the fifo is Full, set the SGMCTL.SOGCHx to enable another DDS generation of another note sequence. | | |
| | ... | ... |

**Note:** * If user want to use the same wavetable for generating another note sequence, step 6-10 can be skipped.

### 39.9.2.2 Polyphonic alarm sound

#### 39.9.2.2.1 Description

In this use case the application generates a repetitive polyphonic sound with a blank period in-between.

The duration of a single sound and the inter-note no-output time can be controlled by the ASR envelope. The duration of the repetition can be programmed in the number of the note pulses.

#### 39.9.2.2.2 SGM functionality

A typical/simple example could be:

- Every channel generates one note periodically. Four channels generate four different notes.
- CPU only needs to configure the SGM once.
- All four channels can be configured simultaneously or in some sequence.
- The number of the note pulses are programmable to control the entire duration.

The figures below show an simple example of polyphonic alarm sound.

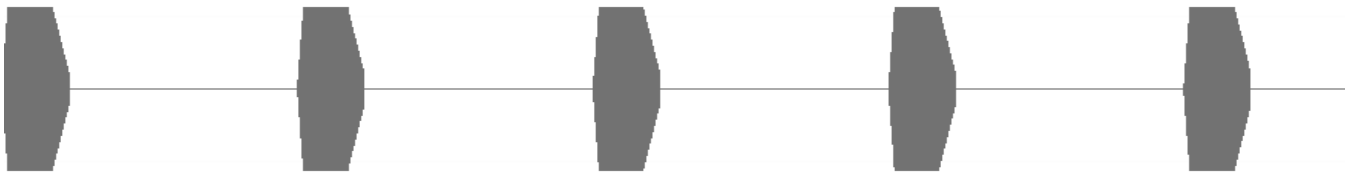


**Figure 39-66. Channel 0**

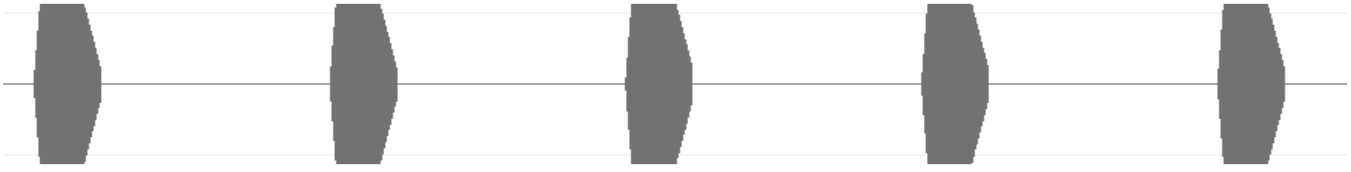


**Figure 39-67. Channel 1**

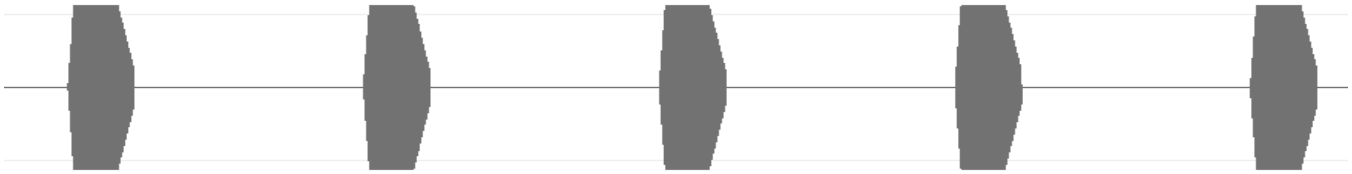


Figure 39-68. Channel 2

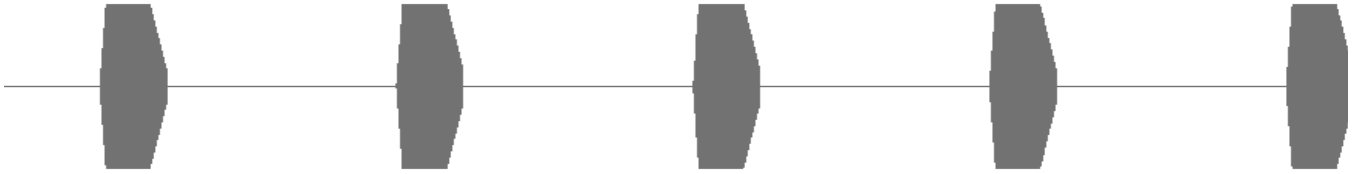


Figure 39-69. Channel 3

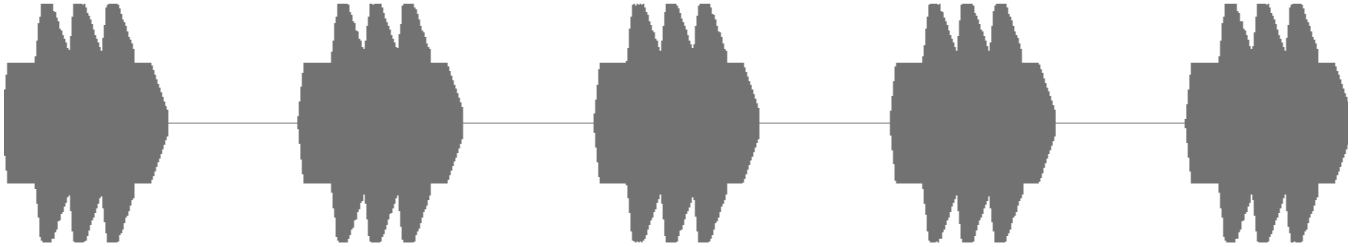


Figure 39-70. Mixed polyphonic alarm sound

This page is intentionally left blank.

# Chapter 40
# Static RAM (SRAM)

## 40.1 Introduction

This device includes a 64 KB SRAM module.

The main features of the SRAM module are:

- 64-bit RAM organization with ECC
- Available for data and program storage
- 64-bit ECC with single-bit correction, double-bit detection on a 32-bit boundary for data integrity
- Supports byte (8-bit), half word (16-bit), word (32-bit) and long word (64-bit) writes for optimal use of memory
- User transparent ECC encoding and decoding for byte, half word, and word accesses
- Separate internal power domains applied RAM block
- Standby modes to retain contents during low power mode
- The device can boot from the RAM for fast recovery from low power mode without the need to wait for the flash memory to be available
- Option available to set the SRAM access to one wait state.

### 40.1.1 Modes of operation

There are two main SRAM operating modes: normal mode and standby mode. These modes are briefly described in this section.

#### 40.1.1.1 Normal (functional) mode

Normal mode allows for reads and writes of the SRAM memory arrays.

#### 40.1.1.2 Standby mode

Standby mode preserves the contents of all or a the portion of the memory during low-power standby mode.

## 40.2 External signal description

There are no external signals associated with the SRAM.

## 40.3 Memory map and registers

There are no control or status registers directly associated with the SRAM module, although error-correcting code (ECC) registers are provided in the error correction status module (ECSM). See Chapter 19, “Error Correction Status Module (ECSM),” for more information.

## 40.4 Functional description

ECC checks are performed during the read portion of an SRAM ECC read/write (R/W) operation, and ECC calculations are performed during the write portion of a read/write (R/W) operation. Because the ECC bits can contain random data after the device is powered on, you must initialize the SRAM by executing 64-bit write instructions to the entire SRAM. For more information, refer to Section 40.8, "Initialization and application information."

## 40.5 SRAM ECC mechanism

The SRAM ECC detects the following conditions and produces the following results:

- Detects and corrects all 1-bit errors
- Detects and flags all 2-bit errors as non-correctable errors
- Detects 72-bit reads (64-bit data bus plus the 8-bit ECC) that return all zeros or all ones, asserts an error indicator on the bus cycle, and sets the error flag

The intent of this is to detect all odd-bit failures, all two-bit failures, some three-bit failures, and some multi-bit failures, with regard to IEC 61508-7 A.5.6.

**NOTE**

The SRAM does not detect all errors greater than 2 bits.

Internal SRAM write operations are performed on the following byte boundaries:

- 1 byte (0:7 bits)
- 2 bytes (0:15 bits)
- 4 bytes or 1 word (0:31 bits)
- 8 bytes or 2 words (0:63 bits)

If the entire 64 data bits are written to SRAM, no read operation is performed and the ECC is calculated across the 64-bit data bus. The 8-bit ECC is appended to the data segment and written to SRAM.

If the write operation is less than the entire 64-bit data width (1-, 2-, or 4-byte segment), the following occurs:

1. The ECC mechanism checks the entire 64-bit data bus for errors, detecting and either correcting or flagging errors.
2. The write data bytes (1-, 2-, or 4-byte segment) are merged with the corrected 64 bits on the data bus.
3. The ECC is then calculated on the resulting 64 bits formed in the previous step.
4. The 8-bit ECC result is appended to the 64 bits from the data bus, and the 72-bit value is then written to SRAM.

### 40.5.1 Access timing

The system bus is a two-stage pipelined bus, which makes the timing of any access dependent on the access during the previous clock. Additionally, the ECSM_MUDCR register optionally forces a minimum of one

wait state for every access; refer to Chapter 19, “Error Correction Status Module (ECSM) for more information. Table 40-1 lists the various combinations of read and write operations to SRAM and the number of wait states used for each operation. The table columns contain the following information:

Current operation — Lists the type of SRAM operation executing currently

Previous operation — Lists the valid types of SRAM operations that can precede the current SRAM operation (valid operation during the preceding clock)

Wait states — Lists the number of wait states (bus clocks) the operation requires which depends on the combination of the current and previous operation

**Table 40-1. Number of wait states required for SRAM operations**

| | Current operation | Previous operation | Number of wait states required[1] |
|---|---|---|---|
| Read operation | Read | Idle | 1 |
| | | Pipelined read | |
| | | Burst read | |
| | | 64-bit write | 2 |
| | | 8-, 16-, or 32-bit write | 0<br>(read from the same address) |
| | | | 1<br>(read from a different address) |
| | Pipelined read | Read | 0 |
| | Burst read | Idle | 1,0,0,0 |
| | | Pipelined read | |
| | | Burst read | |
| | | 64-bit write | 2,0,0,0 |
| | | 8-, 16-, or 32-bit write | 0,0,0,0<br>(read from the same address) |
| | | | 1,0,0,0<br>(read from a different address) |

**Table 40-1. Number of wait states required for SRAM operations (continued)**

| | Current operation | Previous operation | Number of wait states required[1] |
|---|---|---|---|
| Write operation | 8-, 16-, or 32-bit write | Idle | 1 |
| | | Read | |
| | | Pipelined 8-, 16-, or 32-bit write | 2 |
| | | 64-bit write | |
| | | 8-, 16-, or 32-bit write | 0<br>(write to the same address) |
| | Pipelined 8-, 16-, or 32-bit write | 8-, 16-, or 32-bit write | 0 |
| | 64-bit write | Idle | 0 |
| | | 64-bit write | |
| | | Read | |
| | 64-bit burst write | Idle | 0,0,0,0 |
| | | 64-bit write | |
| | | Read | |

1 The ECSM_MUDCR[SRAM_ADD_ONE_WS] bit forces a minimum access of 1 cycle when enabled.

### 40.5.2 Reset effects on SRAM accesses

An asynchronous functional reset will possibly corrupt RAM if it asserts during a read or write operation to SRAM. The completion of that access depends on the cycle at which the reset occurs. Data read from or written to SRAM before the reset event occurred is retained, and no other address locations are accessed or changed.

In case of no access ongoing when reset occurs, the RAM corruption does not happen.

Instead synchronous reset (SW reset) should be used in a controlled function (without RAM accesses) in case initialization procedure is needed without RAM initialization.

After any Destructive reset, the contents of RAM are not guaranteed.

## 40.6 DMA requests

There are no DMA requests associated with the system SRAM.

## 40.7 Interrupt Requests

There are no interrupt requests associated with the system SRAM, except for the ECC reporting through the ECSM.

## 40.8 Initialization and application information

To use the SRAM, the ECC must check all bits that require initialization after power on. Use a 64-bit cache-inhibited write to each SRAM location to initialize the SRAM array as part of the application initialization code. All writes must specify an even number of registers performed on 64-bit word-aligned boundaries. If the write is not the entire 64 bits (e.g., 8, 16, or 32 bits), a read / modify / write operation is generated that checks the ECC value upon the read. See Section 40.5, “SRAM ECC mechanism.”

**NOTE**

You *must* initialize SRAM, even if the application does not use ECC reporting.

### 40.8.1 Example code

To initialize SRAM correctly, use a store multiple word **(stmw)** instruction to implement 64-bit writes to all SRAM locations. The **stmw** instruction concatenates two 32-bit registers to implement a single 64-bit write. To ensure the writes are 64-bits, specify an even number of registers and write on 64-bit word-aligned boundaries.

The following example code illustrates the use of the **stmw** instruction to initialize the SRAM ECC bits.

**Example 40-1. Initializing SRAM ECC bits**

```
init_RAM:
lis      r11,0x4000        # base address of the SRAM, 64-bit word aligned
ori      r11,r11,0         # not needed for this address but could be for others
li       r12,640           # loop counter to get all of SRAM;
                           # 80k/4 bytes/32 GPRs = 640
mtctr    r12
init_ram_loop:
stmw     r0,0(r11)         # write all 32 GPRs to SRAM
addi     r11,r11,128       # inc the ram ptr; 32 GPRs * 4 bytes = 128
bdnz     init_ram_loop     # loop for 80k of SRAM
```

This page is intentionally left blank.

# Chapter 41
# Stepper Motor Controller (SMC)

## 41.1 Introduction

The SMC block is a PWM motor controller suitable for driving small stepper and air core motors used in instrumentation applications. The module can also be used for other motor control or PWM applications that match the frequency, resolution and output drive capabilities of the module. The SMC has 12 PWM channels associated with two pins each (24 pins in total).

### 41.1.1 Features

The SMC includes the following features:

- 10/11-bit PWM counter
- 11-bit resolution with selectable PWM dithering function
- Left, right, or center aligned PWM
- Short-circuit detection in each PWM channel with programmable time-out

### 41.1.2 Modes of operation

#### 41.1.2.1 Functional modes

##### 41.1.2.1.1 Dither function

Dither function can be selected or deselected by setting or clearing the MCCTL0[DITH] bit. This bit influences all PWM channels. For details, please refer to Section 41.4.1.3.5, “Dither Bit (MCCTL0[DITH])”.

#### 41.1.2.2 PWM channel configuration modes

The 12 PWM channels can operate in three functional modes. Those modes are, with some restrictions, selectable for each channel independently.

##### 41.1.2.2.1 Dual full H-bridge mode

This mode is suitable to drive a stepper motor or a $360^{o}$ air gauge instrument. For details, please refer to Section 41.4.1.1.1, “Dual Full H-Bridge Mode”. In this mode two adjacent PWM channels are combined, and two PWM channels drive four pins.

##### 41.1.2.2.2 Full H-bridge mode

This mode is suitable to drive any load requiring a PWM signal in a H-bridge configuration using two pins. For details please refer to Section 41.4.1.1.2, “Full H-Bridge Mode”.

#### 41.1.2.2.3 Half H-bridge mode

This mode is suitable to drive a $90^{o}$ instrument driven by one pin. For details, please refer to Section 41.4.1.1.3, “Half H-Bridge Mode”.

### 41.1.2.3 PWM alignment modes

Each PWM channel can operate independently in three different alignment modes. For details, please refer to Section 41.4.1.3.1, “PWM Alignment Modes”.

### 41.1.2.4 Low-power modes

The behavior of the SMC when it is disabled by the Mode Entry module is programmable. For details, please see Section 41.4.5, “Operation in SMC stop mode”.

### 41.1.3 Block diagram

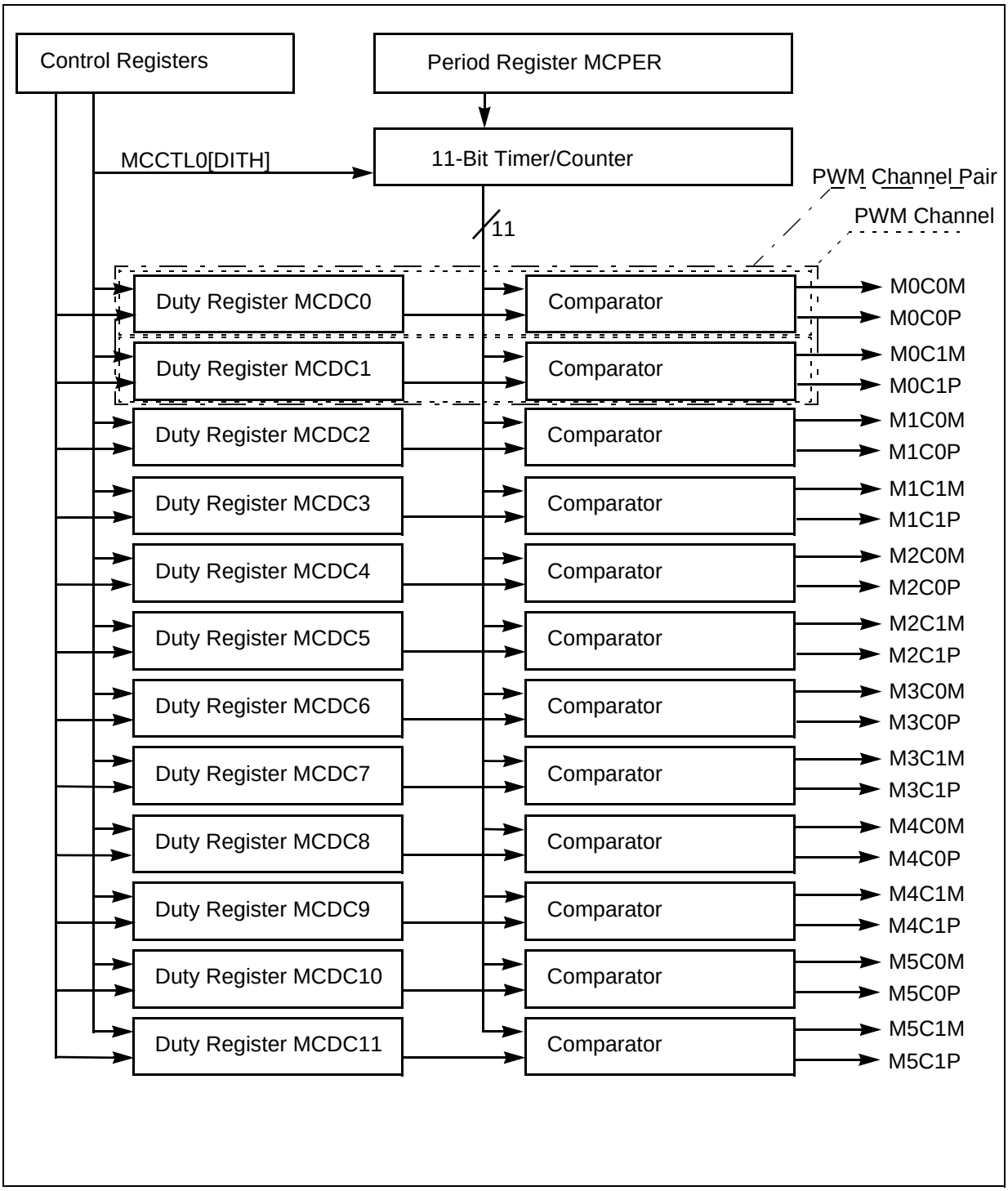



**Figure 41-1. SMC Block Diagram**

## 41.2 External signal description

The SMC is associated with 24 pins. Table 41-1 lists the relationship between the PWM channels, signal pins, PWM channel pairs (motor numbers), coils and nodes they are supposed to drive if all channels are set to dual full H-bridge configuration.

**Table 41-1. PWM Channel and Pin Assignment**

| Pin Name | PWM Channel PW | M Channel Pair[1] | Coil No | de |
|---|---|---|---|---|
| M0C0M | 0 | 0 | 0 | Minus |
| M0C0P | | | | Plus |
| M0C1M | 1 | | 1 | Minus |
| M0C1P | | | | Plus |
| M1C0M | 2 | 1 | 0 | Minus |
| M1C0P | | | | Plus |
| M1C1M | 3 | | 1 | Minus |
| M1C1P | | | | Plus |
| M2C0M | 4 | 2 | 0 | Minus |
| M2C0P | | | | Plus |
| M2C1M | 5 | | 1 | Minus |
| M2C1P | | | | Plus |
| M3C0M | 6 | 3 | 0 | Minus |
| M3C0P | | | | Plus |
| M3C1M | 7 | | 1 | Minus |
| M3C1P | | | | Plus |
| M4C0M | 8 | 4 | 0 | Minus |
| M4C0P | | | | Plus |
| M4C1M | 9 | | 1 | Minus |
| M4C1P | | | | Plus |
| M5C0M | 10 | 5 | 0 | Minus |
| M5C0P | | | | Plus |
| M5C1M | 11 | | 1 | Minus |
| M5C1P | | | | Plus |

1 A PWM Channel Pair always consists of PWM channel x and PWM channel x+1 (x = 2·n). The term “PWM Channel Pair” is equivalent to the term “Motor”. For example, Channel Pair 0 is equivalent to Motor 0.

### 41.2.1 M0C0M/M0C0P/M0C1M/M0C1P — PWM Output Pins for Motor 0

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 0. PWM output on M0C0M results in a positive current flow through coil 0 when M0C0P is driven to a logic high state. PWM output on M0C1M results in a positive current flow through coil 1 when M0C1P is driven to a logic high state (for details refer to Section 41.4.1, “Modes of Operation”).

### 41.2.2 M1C0M/M1C0P/M1C1M/M1C1P — PWM Output Pins for Motor 1

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 1. PWM output on M1C0M results in a positive current flow through coil 0 when M1C0P is driven to a logic high state. PWM output on M1C1M results in a positive current flow through coil 1 when M1C1P is driven to a logic high state (for details refer to Section 41.4.1, “Modes of Operation”).

### 41.2.3 M2C0M/M2C0P/M2C1M/M2C1P — PWM Output Pins for Motor 2

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 2. PWM output on M2C0M results in a positive current flow through coil 0 when M2C0P is driven to a logic high state. PWM output on M2C1M results in a positive current flow through coil 1 when M2C1P is driven to a logic high state (for details refer to Section 41.4.1, “Modes of Operation”).

### 41.2.4 M3C0M/M3C0P/M3C1M/M3C1P — PWM Output Pins for Motor 3

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 3. PWM output on M3C0M results in a positive current flow through coil 0 when M3C0P is driven to a logic high state. PWM output on M3C1M results in a positive current flow through coil 1 when M3C1P is driven to a logic high state (for details refer to Section 41.4.1, “Modes of Operation”).

### 41.2.5 M4C0M/M4C0P/M4C1M/M4C1P — PWM Output Pins for Motor 4

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 4. PWM output on M4C0M results in a positive current flow through coil 0 when M4C0P is driven to a logic high state. PWM output on M4C1M results in a positive current flow through coil 1 when M4C1P is driven to a logic high state (for details refer to Section 41.4.1, “Modes of Operation”).

### 41.2.6 M5C0M/M5C0P/M5C1M/M5C1P — PWM Output Pins for Motor 5

High current PWM output pins that can be used for motor drive. These pins interface to the coils of motor 5. PWM output on M5C0M results in a positive current flow through coil 0 when M5C0P is driven to a logic high state. PWM output on M5C1M results in a positive current flow through coil 1 when M5C1P is driven to a logic high state (for details refer to Section 41.4.1, “Modes of Operation”).

## 41.3 Memory map and register definition

This section provides a detailed description of all registers of the 10-bit 12-channel SMC module.

### 41.3.1 Module memory map

Table 41-2 shows the memory map of the 10-bit 12-channel SMC module.

Access type can be

- RW: Read and Write

- Data access type is 8,16 or 32 bit. It is recommended to access the various register using the access types shown in Table 41-2 for consistent write operations.

**Table 41-2. SMC — Memory Map**

| Address offset | Use | Recommended Access type | Location |
|---|---|---|---|
| 0x00 | Motor Controller Control Register 0 (MCCTL0) | RW, 8 bit | on page 1420 |
| 0x01 | Motor Controller Control Register 1 (MCCTL1) | RW, 8 bit | on page 1421 |
| 0x02 | Motor Controller Period Register (MCPER) | RW, 16 bit | on page 1422 |
| 0x04 | Reserved | — | |
| 0x05 | Reserved | — | |
| 0x06 | Reserved | — | |
| 0x07 | Reserved | — | |
| 0x08 | Reserved | — | |
| 0x09 | Reserved | — | |
| 0x0A | Reserved | — | |
| 0x0B | Reserved | — | |
| 0x0C | Reserved | — | |
| 0x0D | Reserved | — | |
| 0x0E | Reserved | — | |
| 0x0F | Reserved | — | |
| 0x10 | Motor Controller Channel Control Register 0 (MCCC0) | RW, 8 bit | on page 1422 |
| 0x11 | Motor Controller Channel Control Register 1 (MCCC1) | RW, 8 bit | on page 1422 |
| 0x12 | Motor Controller Channel Control Register 2 (MCCC2) | RW, 8 bit | on page 1422 |
| 0x13 | Motor Controller Channel Control Register 3 (MCCC3) | RW, 8 bit | on page 1422 |
| 0x14 | Motor Controller Channel Control Register 4 (MCCC4) | RW, 8 bit | on page 1422 |
| 0x15 | Motor Controller Channel Control Register 5 (MCCC5) | RW, 8 bit | on page 1422 |
| 0x16 | Motor Controller Channel Control Register 6 (MCCC6) | RW, 8 bit | on page 1422 |
| 0x17 | Motor Controller Channel Control Register 7 (MCCC7) | RW, 8 bit | on page 1422 |
| 0x18 | Motor Controller Channel Control Register 8 (MCCC8) | RW, 8 bit | on page 1422 |
| 0x19 | Motor Controller Channel Control Register 9 (MCCC9) | RW, 8 bit | on page 1422 |
| 0x1A | Motor Controller Channel Control Register 10 (MCCC10) | RW, 8 bit | on page 1422 |
| 0x1B | Motor Controller Channel Control Register 11 (MCCC11) | RW, 8 bit | on page 1422 |
| 0x1C | Reserved | - | |
| 0x1D | Reserved | - | |
| 0x1E | Reserved | - | |
| 0x1F | Reserved | - | |
| 0x20 | Motor Controller Duty Cycle Register 0 (MCDC0) | RW, 16 bit | on page 1423 |
| 0x22 | Motor Controller Duty Cycle Register 1 (MCDC1) | RW, 16 bit | on page 1423 |
| 0x24 | Motor Controller Duty Cycle Register 2 (MCDC2) | RW, 16 bit | on page 1423 |
| 0x26 | Motor Controller Duty Cycle Register 3 (MCDC3) | RW, 16 bit | on page 1423 |
| 0x28 | Motor Controller Duty Cycle Register 4 (MCDC4) | RW, 16 bit | on page 1423 |
| 0x2A | Motor Controller Duty Cycle Register 5 (MCDC5) | RW, 16 bit | on page 1423 |
| 0x2C | Motor Controller Duty Cycle Register 6 (MCDC6) | RW, 16 bit | on page 1423 |
| 0x2E | Motor Controller Duty Cycle Register 7 (MCDC7) | RW, 16 bit | on page 1423 |

Table 41-2. SMC — Memory Map (continued)

| Address offset | Use | Recommended Access type | Location |
|---|---|---|---|
| 0x30 | Motor Controller Duty Cycle Register 8 (MCDC8) | RW, 16 bit | on page 1423 |
| 0x32 | Motor Controller Duty Cycle Register 9 (MCDC9) | RW, 16 bit | on page 1423 |
| 0x34 | Motor Controller Duty Cycle Register 10 (MCDC10) | RW, 16 bit | on page 1423 |
| 0x36 | Motor Controller Duty Cycle Register 11 (MCDC11) | RW, 16 bit | on page 1423 |
| 0x38 | Reserved | - | |
| 0x39 | Reserved | - | |
| 0x3A | Reserved | - | |
| 0x3B | Reserved | - | |
| 0x3C | Reserved | - | |
| 0x3D | Reserved | - | |
| 0x3E | Reserved | - | |
| 0x3F | Reserved | - | |
| 0x40 | Short-circuit Detector Time-out Register (MCSDTO) | RW, 8 bit | on page 1425 |
| 0x41 | Reserved | - | |
| 0x42 | Reserved | - | |
| 0x43 | Reserved | - | |
| 0x44 | Short-circuit Detector Enable Register 0 (MCSDE0) | RW, 8 bit | on page 1425 |
| 0x45 | Short-circuit Detector Enable Register 1 (MCSDE1) | RW, 8 bit | on page 1426 |
| 0x46 | Short-circuit Detector Enable Register 2 (MCSDE2) | RW, 8 bit | on page 1426 |
| 0x47 | Reserved | - | |
| 0x48 | Short-circuit Detector Interrupt Enable Register 0 (MCSDIEN0) | RW, 8 bit | on page 1427 |
| 0x49 | Short-circuit Detector Interrupt Enable Register 1 (MCSDIEN1) | RW, 8 bit | on page 1427 |
| 0x4A | Short-circuit Detector Interrupt Enable Register 2 (MCSDIEN2) | RW, 8 bit | on page 1428 |
| 0x4B | Reserved | - | |
| 0x4C | Short-circuit Detector Interrupt Register 0 (MCSDI0) | R/MW, 8 bit | on page 1428 |
| 0x4D | Short-circuit Detector Interrupt Register 1 (MCSDI1) | R/MW, 8 bit | on page 1429 |
| 0x4E | Short-circuit Detector Interrupt Register 2 (MCSDI2) | R/MW, 8 bit | on page 1429 |
| 0x4F | Reserved | - | |

## 41.3.2 Register description

This section provides detailed descriptions of all registers in ascending address order. Table 41-3 provides a key for the register figures and register tables.

Table 41-3. Register Access Conventions

| Convention | Description |
|---|---|
| | Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable. |
| FIELDNAME | Identifies the field. Its presence in the read or write row indicates that it can be read or written. |
| **Register Field Types** | |

**Table 41-3. Register Access Conventions (continued)**

| Convention | Description |
|---|---|
| rwm | A read/write bit that may be modified by hardware in some fashion other than by a reset. |
| w1c | Write one to clear. A flag bit that can be read, is cleared by writing a one, writing 0 has no effect. |
| **Reset Value** | |
| 0 | Resets to zero. |
| 1 | Resets to one. |

### 41.3.2.1 Motor Controller Control Register 0 (MCCTL0)

This register controls the operating mode of the SMC module.

Offset Module Base + 0x0000

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | MCPRE | | 0 | 0 | DITH | 0 | MCTOIF |
| W | | | | | | | | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-2. Motor Controller Control Register 0 (MCCTL0)**

**Table 41-4. MCCTL0 Field Descriptions**

| Field De | scription |
|---|---|
| MCPRE | **Motor Controller Prescaler Select** — MCPRE determines the prescaler value that sets the motor controller timer counter clock frequency ($f_{TC}$). The clock source for the prescaler is the peripheral bus clock ($f_{BUS}$) as shown in Figure 41-32. Writes to MCPRE will not affect the timer counter clock frequency $f_{TC}$ until the start of the next PWM period.<br>00 $\mathbf{f_{TC}} = f_{Bus}$<br>01 $\mathbf{f_{TC}} = f_{Bus}/2$<br>10 $\mathbf{f_{TC}} = f_{Bus}/4$<br>11 $\mathbf{f_{TC}} = f_{Bus}/8$ |
| DITH | **Motor Control/Driver Dither Feature Enable** (refer to Section 41.4.1.3.5, “Dither Bit (MCCTL0[DITH])”)<br>0 Dither feature is disabled.<br>1 Dither feature is enabled. |
| MCTOIF | **Motor Controller Timer Counter Overflow Interrupt Flag** — This bit is set when a motor controller timer counter overflow occurs. The bit is cleared by writing a 1 to the bit.<br>0 A motor controller timer counter overflow has not occurred since the last reset or since the bit was cleared.<br>1 A motor controller timer counter overflow has occurred. |

### 41.3.2.2 Motor Controller Control Register 1 (MCCTL1)

This register controls the behavior of the analog section of the SMC as well as the interrupt enables.

Offset Module Base + 0x0001

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | RECIRC | 0 | 0 | 0 | 0 | 0 | 0 | MCTOIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-3. Motor Controller Control Register 1 (MCCTL1)**

**Table 41-5. MCCTL1 Field Descriptions**

| Field De | scription |
|---|---|
| RECIRC | **Recirculation in (Dual) Full H-Bridge Mode** (refer to Section 41.4.1.3.3, “Recirculation Bit (MCCTL1[RECIRC])”) — RECIRC only affects the outputs in (dual) full H-bridge modes. In half H-bridge mode, the PWM output is always active low. RECIRC = 1 will also invert the effect of the MCDCx[SIGN] bits (refer to Section 41.4.1.3.2, “Sign Bit (MCDCx[SIGN])”) in (dual) full H-bridge modes. RECIRC must be changed only while no PWM channel is operating in (dual) full H-bridge mode; otherwise, erroneous output pattern may occur.<br>0 Recirculation on the high side transistors. Active state for PWM output is logic low, the static channel will output logic high.<br>1 Recirculation on the low side transistors. Active state for PWM output is logic high, the static channel will output logic low. |
| MCTOIE | **Motor Controller Timer Counter Overflow Interrupt Enable**<br>0 Interrupt disabled.<br>1 Interrupt enabled. An interrupt will be generated when the motor controller timer counter overflow interrupt flag (MCCTL0[MCTOIF]) is set. |

### 41.3.2.3 Motor Controller Period Register (MCPER)

Offset Module Base + 0x0002, 0x0003

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

**Figure 41-4. Motor Controller Period Register (MCPER)**

**Table 41-6. MCPER Field Descriptions**

| Field De | scription |
|---|---|
| PER | **PWM Period** — PER defines the number of motor controller timer counter clocks a PWM period lasts. The motor controller timer counter is clocked with the frequency $f_{TC}$. If dither mode is enabled (MCCTL0[DITH] = 1, refer to Section 41.4.1.3.5, “Dither Bit (MCCTL0[DITH])”), PER[0] is ignored and reads as a 0. In this case PER = 2 * MCDCx[DUTY[10:1]]. |

Setting PER to 0 will shut off all PWM channels as if MCCCx[MCAM] is set to 0 in all channel control registers after the next period timer counter overflow. In this case, the motor controller releases all pins.

**NOTE**

Programming PER to 1 and setting the MCCTL0[DITH] bit will be managed as if PER is programmed to 0. All PWM channels will be shut off after the next period timer counter overflow.

### 41.3.2.4 Motor Controller Channel Control Register (MCCC0..11)

Each PWM channel has one associated control register to control output delay, PWM alignment, and output mode. The number of each register refers directly the PWM channel it controls. The relation between channels, pin names and register names is shown in Table 41-19.

Offset Module Base + 0x0010 . . . 0x001B

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | MCOM | | MCAM | | 0 | 0 | CD | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-5. Motor Controller Channel Control Register (MCCC0..11)**

**Table 41-7. MCCCx Field Descriptions**

| Field De | scription |
|---|---|
| MCOM | **Output Mode** — MCOM controls the PWM channel's output mode.<br>00 Half H-bridge mode, PWM on pin MnCxM, pin MnCxP is released<br>01 Half H-bridge mode, PWM on pin MnCxP, pin MnCxM is released<br>10 Full H-bridge mode<br>11 Dual full H-bridge mode |
| MCAM | **PWM Channel Alignment Mode** — MCAM controls the PWM channel's PWM alignment mode and operation.<br>MCAM and MCOM are double buffered. The values used for the generation of the output waveform will be copied to the working registers either at once (if all PWM channels are disabled or MCPER[PER] is set to 0) or if a timer counter overflow occurs. Reads of the register return the most recent written value, which are not necessarily the currently active values.<br>00 Channel disabled<br>01 Left aligned<br>10 Right aligned<br>11 Center aligned |
| CD | **PWM Channel Delay** — Each PWM channel can be individually delayed by a programmable number of PWM timer counter clocks. The delay will be n/$f_{TC}$.<br>00 Zero PWM clocks channel delay<br>01 One PWM clock channel delay<br>10 Two PWM clocks channel delay<br>11 Three PWM clocks channel delay |

**NOTE**

The SMC will release the pins after the next PWM timer counter overflow without accommodating any channel delay if a single channel has been disabled or if the period register has been cleared or all channels have been disabled. Program one or more inactive PWM frames (duty cycle = 0) before writing a configuration that disables a single channel or the entire SMC.

### 41.3.2.5 Motor Controller Duty Cycle Register (MCDC0..11)

Each duty cycle register sets the sign and duty functionality for the respective PWM channel. The number of each register refires directly the PWM channel it controls. The relation between channels, pin names and register names is shown in Table 41-19.

Offset Module Base + 0x0020 . . . 0x0037

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SIGN [4] | SIGN[3:0] | | | | DUTY | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-6. Motor Controller Duty Cycle Register (MCDC0..11)**

**Table 41-8. MCDCx Field Descriptions**

| Field De | scription |
|---|---|
| SIGN[4] | **Sign Bit** — The SIGN[4] bit is used to define which output will drive the PWM signal in (dual) full-H-bridge modes. The SIGN[4] bit has no effect in half-bridge modes. See Section 41.4.1.3.2, "Sign Bit (MCDCx[SIGN])" and Table 41-20 for detailed information about the impact of MCCTL1[RECIRC] and SIGN[4] bit on the PWM output. |
| SIGN[3:0] | **Sign Bit Extension** — Replicates the SIGN[4] bit towards the DUTY field to make the whole register a signed representation for the duty cycle length. |
| DUTY | **Duty Cycle Length** — DUTY defines the number of motor controller timer counter clocks the corresponding output is driven low (MCCTL1[RECIRC] = 0) or is driven high (MCCTL1[RECIRC] = 1). Setting all bits to 0 will give a static high output in case of MCCTL1[RECIRC] = 0; otherwise, a static low output. Values greater than or equal to the contents of the period register will generate a static low output in case of MCCTL1[RECIRC] = 0, or a static high output if MCCTL1[RECIRC] = 1. |

To prevent the output from inconsistent signals, the duty cycle registers are double buffered. The SMC module will use working registers to generate the output signals. The working registers are copied from the bus accessible registers at the following conditions:

- MCPER[PER] is set to 0 (all channels are disabled in this case)
- MCCCx[MCAM] of the respective channel is set to 0 (channel is disabled)
- A PWM timer counter overflow occurs while in half H-bridge or full H-bridge mode
- A PWM channel pair is configured to work in Dual Full H-Bridge mode and a PWM timer counter overflow occurs after the odd[1] duty cycle register of the channel pair has been written.

In this way, the output of the PWM will always be either the old PWM waveform or the new PWM waveform, not some variation in between.

Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active sign, duty cycle, and dither functionality due to the double buffering scheme.

1. Odd duty cycle register: MCDCx+1, x = 2·n

### 41.3.2.6 Short-circuit Detector Time-out Register (MCSDTO)

Offset Module Base + 0x0040

| | 0 | 1 | 2 | 3 | 5 | 7 |
|---|---|---|---|---|---|---|
| R<br>W | TOUT | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-7. Short-circuit Detector Time-out Register (MCSDTO)**

**Table 41-9. MCSDTO Field Descriptions**

| Field De | scription |
|---|---|
| TOUT | **Time-out** — The value TOUT is an unsigned 8-bit number. This value is used as load value for the short-circuit detection counters. This value is applied to all 24 short-circuit detection blocks. Due to synchronization and sampling, TOUT must always be larger than 2 (see also Section 41.4.6, “Short-circuit detection”). |

### 41.3.2.7 Short-circuit Detector Enable Register 0 (MCSDE0)

Offset Module Base + 0x0044

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SDEN | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-8. Short-circuit Detector Enable Register 0 (MCSDE0)**

**Table 41-10. MCSDE0 Field Description**

| Field De | scription |
|---|---|
| SDEN | **Short-Circuit Detector Enable** — Each short-circuit detector can be enabled or disabled according to the mapping described in Table 41-23. The short-circuit detector of a given pin is enabled if the related enable bit is set to 1. |

### 41.3.2.8 Short-circuit Detector Enable Register 1 (MCSDE1)

Offset Module Base + 0x0045

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SDEN | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-9. Short-circuit Detector Enable Register 1 (MCSDE1)**

**Table 41-11. MCSDE1 Field Description**

| Field De | scription |
|---|---|
| SDEN | **Short-Circuit Detector Enable** — Each short-circuit detector can be enabled or disabled according to the mapping described in Table 41-23. The short-circuit detector of a given pin is enabled if the related enable bit is set to 1. |

### 41.3.2.9 Short-circuit Detector Enable Register 2 (MCSDE2)

Offset Module Base + 0x0046

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SDEN | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-10. Short-circuit Detector Enable Register 2 (MCSDE2)**

**Table 41-12. MCSDE2 Field Description**

| Field De | scription |
|---|---|
| SDEN | **Short-Circuit Detector Enable** — Each short-circuit detector can be enabled or disabled according to the mapping described in Table 41-23. The short-circuit detector of a given pin is enabled if the related enable bit is set to 1. |

### 41.3.2.10 Short-circuit Detector Interrupt Enable Register 0 (MCSDIEN0)

Offset Module Base + 0x0048

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SDIE | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-11. Short-circuit Detector Interrupt Enable Register 0 (MCSDIEN0)**

**Table 41-13. MCSDIEN0 Field Descriptions**

| Field De | scription |
|---|---|
| SDIE | **Short-Circuit Detector Interrupt Enable** — The interrupt of each short-circuit detector can individually be enabled or disabled according to the mapping described in Table 41-23. The short-circuit detector interrupt of a given pin is enabled if the related interrupt enable bit is set to 1. |

### 41.3.2.11 Short-circuit Detector Interrupt Enable Register 1 (MCSDIEN1)

Offset Module Base + 0x0049

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SDIE | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-12. Short-circuit Detector Interrupt Enable Register 1 (MCSDIEN1)**

**Table 41-14. MCSDIEN1 Field Descriptions**

| Field De | scription |
|---|---|
| SDIE | **Short-Circuit Detector Interrupt Enable** — The interrupt of each short-circuit detector can individually be enabled or disabled according to the mapping described in Table 41-23. The short-circuit detector interrupt of a given pin is enabled if the related interrupt enable bit is set to 1. |

### 41.3.2.12 Short-circuit Detector Interrupt Enable Register 2 (MCSDIEN2)

Offset Module Base + 0x004A

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | SDIE | | | | | | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-13. Short-circuit Detector Interrupt Enable Register 2 (MCSDIEN2)**

**Table 41-15. MCSDIEN2 Field Descriptions**

| Field De | scription |
|---|---|
| SDIE | **Short-Circuit Detector Interrupt Enable** — The interrupt of each short-circuit detector can individually be enabled or disabled according to the mapping described in Table 41-23. The short-circuit detector interrupt of a given pin is enabled if the related interrupt enable bit is set to 1. |

### 41.3.2.13 Short-circuit Detector Interrupt Register 0 (MCSDI0)

Offset Module Base + 0x004C

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | SDIF | | | | | | | |
| W | rwm | rwm | rwm | rwm | rwm | rwm | rwm | rwm |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-14. Short-circuit Detector Interrupt Register 0 (MCSDI0)**

**Table 41-16. MCSDI0 Field Descriptions**

| Field De | scription |
|---|---|
| SDIF | **Short-circuit Detector Interrupt Flag** — In case of a detected short-circuit, the corresponding bit according to the mapping in Table 41-23 is set in the short-circuit detector interrupt register.<br>If this specific interrupt is also enabled in the interrupt enable register MCSDIEN0, than this event will rise an external interrupt. |

### 41.3.2.14 Short-circuit Detector Interrupt Register 1 (MCSDI1)

Offset Module Base + 0x004D

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | SDIF | | | | | | | |
| W | rwm | rwm | rwm | rwm | rwm | rwm | rwm | rwm |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-15. Short-circuit Detector Interrupt Register 1 (MCSDI1)**

**Table 41-17. MCSDI1 Field Descriptions**

| Field De | scription |
|---|---|
| SDIF | **Short-circuit Detector Interrupt Flag** — In case of a detected short-circuit, the corresponding bit according to the mapping in Table 41-23 is set in the short-circuit detector interrupt register.<br>If this specific interrupt is also enabled in the interrupt enable register MCSDIEN1, than this event will rise an external interrupt. |

### 41.3.2.15 Short-circuit Detector Interrupt Register 2 (MCSDI2)

Offset Module Base + 0x004E

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | SDIF | | | | | | | |
| W | rwm | rwm | rwm | rwm | rwm | rwm | rwm | rwm |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 41-16. Short-circuit Detector Interrupt Register 2 (MCSDI2)**

**Table 41-18. MCSDI2 Field Descriptions**

| Field Des | criptions |
|---|---|
| SDIF | **Short-circuit Detector Interrupt Flag** — In case of a detected short-circuit, the corresponding bit according to the mapping in Table 41-23 is set in the short-circuit detector interrupt register.<br>If this specific interrupt is also enabled in the interrupt enable register MCSDIEN2, than this event will rise an external interrupt. |

## 41.4 Functional Description

### 41.4.1 Modes of Operation

#### 41.4.1.1 PWM Output Modes

The SMC is configured between three output modes.

- Dual full H-bridge mode can be used to control either a stepper motor or a 360° air core instrument. In this case two PWM channels are combined.
- In full H-bridge mode, each PWM channel is updated independently.
- In half H-bridge mode, one pin of the PWM channel can generate a PWM signal to control a 90° air core instrument (or other load requiring a PWM signal) and the other pin is unused.

The mode of operation for PWM channel x is determined by the output mode bits MCCCx[MCOM]. After a reset occurs, each PWM channel will be disabled, the corresponding pins are released.

Each PWM channel consists of two pins. One output pin will generate a PWM signal. The other will operate as logic high or low output depending on the state of the recirculation bit MCCTL1[RECIRC] (refer to Section 41.4.1.3.3, “Recirculation Bit (MCCTL1[RECIRC])”), while in (dual) full H-bridge mode, or will be released, while in half H-bridge mode. The state of the sign bit MCDCx[SIGN[4]] in the duty cycle register determines the pin where the PWM signal is driven in full H-bridge mode. While in half H-bridge mode, the state of the released pin is determined by other modules associated with this pin.

Associated with each PWM channel pair n are two PWM channels, x and x + 1, where x = 2 * n and n (0,1,2... 5) is the PWM channel pair number. Duty cycle register x controls the sign of the PWM signal (which pin drives the PWM signal) and the duty cycle of the PWM signal for SMC channel x. The pins associated with PWM channel x are MnC0P and MnC0M. Similarly, duty cycle register x + 1 controls the sign of the PWM signal and the duty cycle of the PWM signal for channel x + 1. The pins associated with PWM channel x + 1 are MnC1P and MnC1M. This is summarized in Table 41-19.

**Table 41-19. Corresponding Registers and Pin Names for each PWM Channel Pair**

| PWM Channel Pair Number | PWM Channel Control Register | Duty Cycle Register | Channel Number | Pin Names |
|---|---|---|---|---|
| n | MCCCx | MCDCx | PWM Channel x, x = 2·n | MnC0M |
| | | | | MnC0P |
| | MCCCx+1 | MCDCx+1 | PWM Channel x+1, x = 2·n | MnC1M |
| | | | | MnC1P |
| 0 | MCCC0 | MCDC0 | PWM Channel 0 | M0C0M |
| | | | | M0C0P |
| | MCCC1 | MCDC1 | PWM Channel 1 | M0C1M |
| | | | | M0C1P |

**Table 41-19. Corresponding Registers and Pin Names for each PWM Channel Pair (continued)**

| PWM Channel Pair Number | PWM Channel Control Register | Duty Cycle Register | Channel Number | Pin Names |
|---|---|---|---|---|
| 1 | MCCC2 | MCDC2 | PWM Channel 2 | M1C0M |
| | | | | M1C0P |
| | MCCC3 | MCDC3 | PWM Channel 3 | M1C1M |
| | | | | M1C1P |
| 2 | MCCC4 | MCDC4 | PWM Channel 4 | M2C0M |
| | | | | M2C0P |
| | MCCC5 | MCDC5 | PWM Channel 5 | M2C1M |
| | | | | M2C1P |
| 3 | MCCC6 | MCDC6 | PWM Channel 6 | M3C0M |
| | | | | M3C0P |
| | MCCC7 | MCDC7 | PWM Channel 7 | M3C1M |
| | | | | M3C1P |
| 4 | MCCC8 | MCDC8 | PWM Channel 8 | M4C0M |
| | | | | M4C0P |
| | MCCC9 | MCDC9 | PWM Channel 9 | M4C1M |
| | | | | M4C1P |
| 5 | MCCC10 | MCDC10 | PWM Channel 10 | M5C0M |
| | | | | M5C0P |
| | MCCC11 | MCDC11 | PWM Channel 11 | M5C1M |
| | | | | M5C1P |

#### 41.4.1.1.1 Dual Full H-Bridge Mode

PWM channel pairs x and x + 1 operate in dual full H-bridge mode if both channels are enabled (MCCCx[MCAM]=0x1, 0x2, or 0x3) and the output mode bits MCCCx[MCOM] in both PWM channel control registers are set to 0x3.

A typical configuration in dual full H-bridge mode is shown in Figure 41-17. PWM channel x drives the PWM output signal on either MnC0P or MnC0M. If MnC0P drives the PWM signal, MnC0M will be output either high or low depending on the MCCTL1[RECIRC] bit. If MnC0M drives the PWM signal, MnC0P will be an output high or low. PWM channel x + 1 drives the PWM output signal on either MnC1P or MnC1M. If MnC1P drives the PWM signal, MnC1M will be an output high or low. If MnC1M drives the PWM signal, MnC1P will be an output high or low. This results in motor recirculation currents on the high side drivers (MCCTL1[RECIRC] = 0) while the PWM signal is at a logic high level, or motor recirculation currents on the low side drivers (MCCTL1[RECIRC] = 1) while the PWM signal is at a logic low level. The pin driving the PWM signal is determined by the sign bit MCDCx[SIGN[4]] for the corresponding channel and the state of the MCCTL1[RECIRC] bit. The value of the PWM duty cycle is determined by the value of the duty cycle bits MCDCx[DUTY] for the corresponding channel.

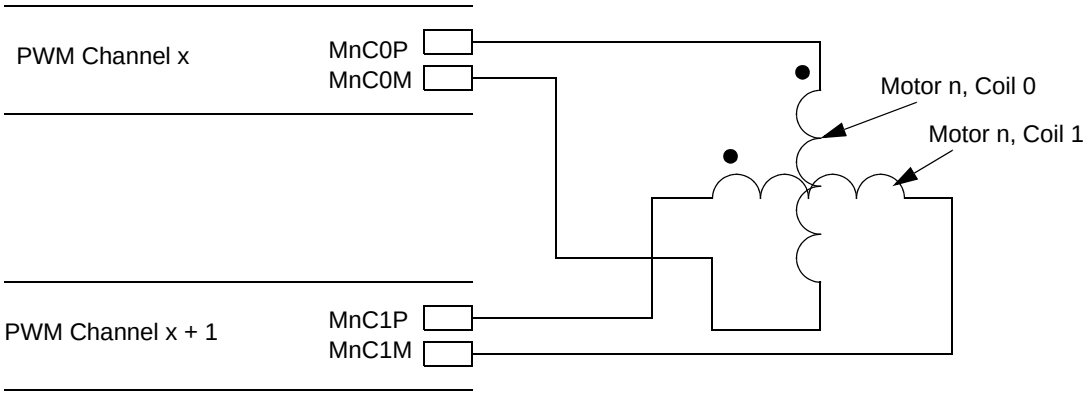



**Figure 41-17. Typical Dual Full H-Bridge Mode Configuration**

16-bit write accesses to the duty cycle registers are allowed, 8-bit write accesses can lead to unpredictable duty cycles.

The following sequence should be used to update the current magnitude and direction for coil 0 and coil 1 of the motor to achieve consistent PWM output:

1. Write to duty cycle register x
2. Write to duty cycle register x + 1

At the next timer counter overflow, the duty cycle registers will be copied to the working duty cycle registers. Sequential writes to the duty cycle register x will result in the previous data being overwritten.

#### 41.4.1.1.2 Full H-Bridge Mode

In full H-bridge mode (MCCCx[MCOM]=0x2), the PWM channels x and x + 1 operate independently. The duty cycle working registers are updated whenever a timer counter overflow occurs.

#### 41.4.1.1.3 Half H-Bridge Mode

In half H-bridge mode (MCCCx[MCOM] = 0x0 or 0x1), the PWM channels x and x + 1 operate independently. In this mode, each PWM channel can be configured such that one pin is released and the other pin is a PWM output. Figure 41-18 shows a typical configuration in half H-bridge mode.

The two pins associated with each channel are switchable between released mode and PWM output dependent upon the state of the output mode bits MCCCx[MCOM]. See register description in Section 41.3.2.4, “Motor Controller Channel Control Register (MCCC0..11)”. In half H-bridge mode, the state of the MCDCx[SIGN[4]] bit has no effect.

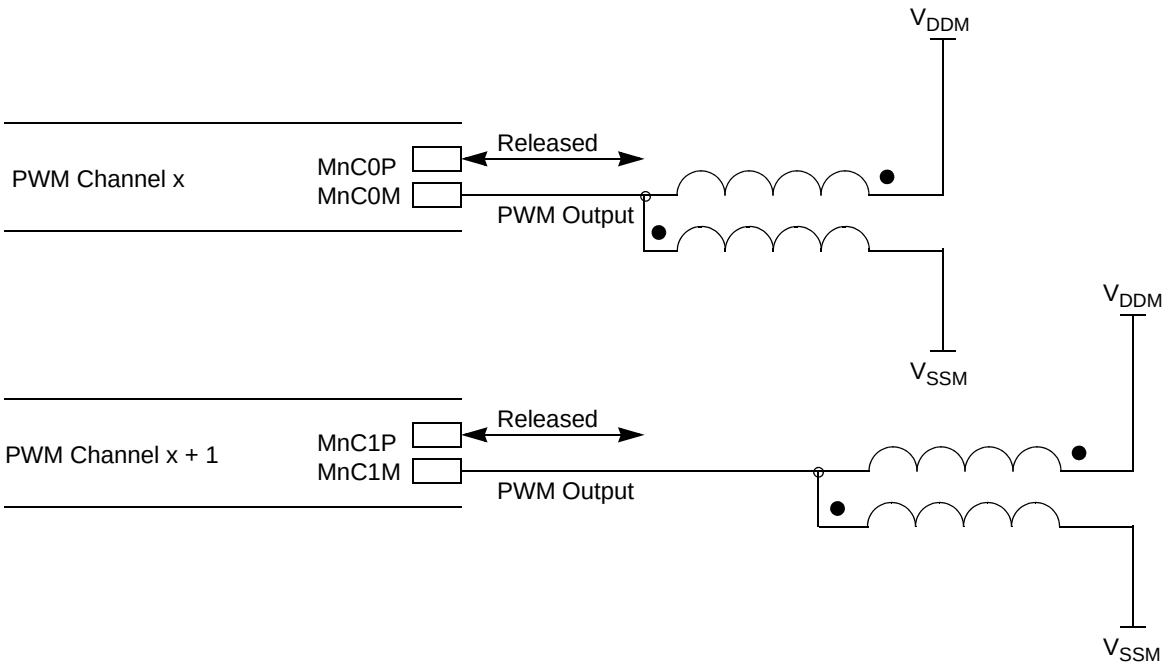



**Figure 41-18. Typical Quad Half H-Bridge Mode Configuration**

### 41.4.1.2 Relationship Between PWM Mode and PWM Channel Enable

The pair of SMC channels cannot be placed into dual full H-bridge mode unless both SMC channels have been enabled (MCCCx[MCAM] not equal to 0) and dual full H-bridge mode is selected for both PWM channels (MCCCx[MCOM] = 0x3). If only one channel is set to dual full H-bridge mode, this channel will operate in full H-bridge mode, the other as programmed.

### 41.4.1.3 Relationship Between Sign, Duty, Dither, RECIRC, Period, and PWM Mode Functions

#### 41.4.1.3.1 PWM Alignment Modes

Each PWM channel can be programmed individually to three different alignment modes. The alignment mode is determined by the MCCCx[MCAM] bits in the corresponding channel control register.

Left aligned (MCCCx[MCAM] = 0x1): The output will start active (low if MCCTL1[RECIRC] = 0 or high if MCCTL1[RECIRC] = 1) and will turn inactive (high if MCCTL1[RECIRC] = 0 or low if MCCTL1[RECIRC] = 1) after the number of counts specified by the corresponding duty cycle register.

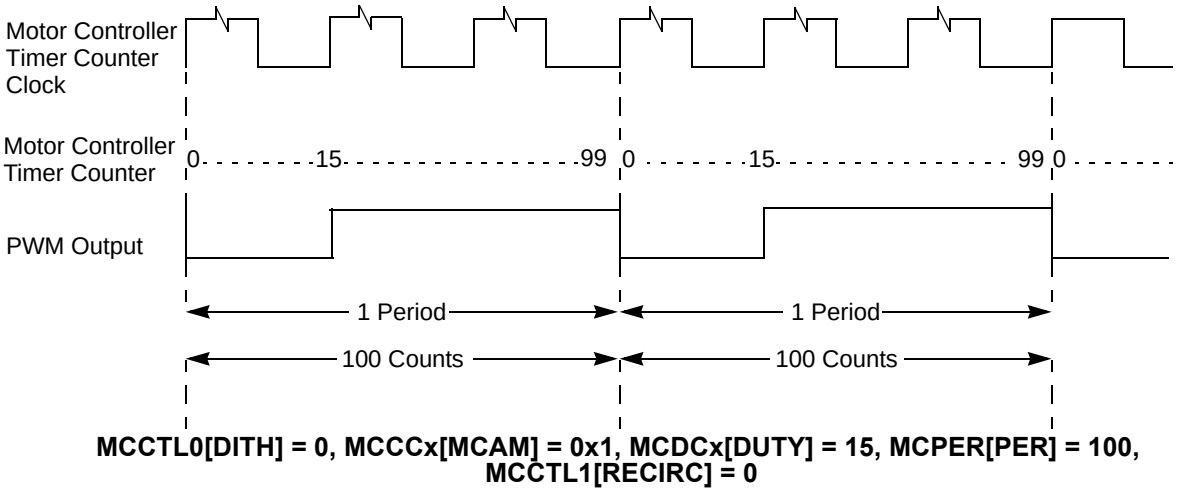



**Figure 41-19. Left Aligned**

Right aligned (MCCCx[MCAM] = 0x2): The output will start inactive (high if MCCTL1[RECIRC] = 0 and low if
MCCTL1[RECIRC] = 1) and will turn active after the number of counts specified by the difference of the contents of period register and the corresponding duty cycle register.

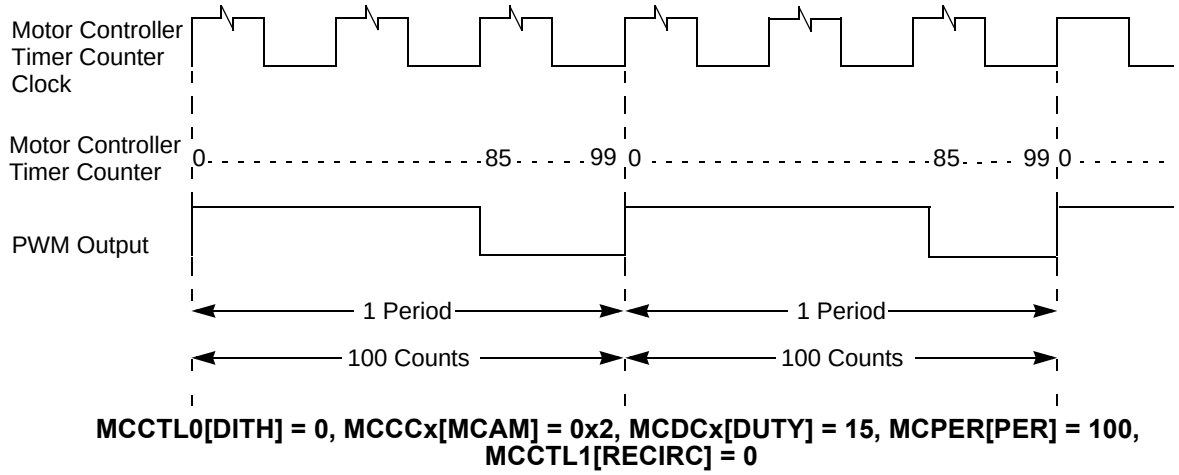



**Figure 41-20. Right Aligned**

Center aligned (MCCCx[MCAM] = 0x3): Even periods will be output left aligned, odd periods will be output right aligned. PWM operation starts with the even period after the channel has been enabled. PWM operation in center aligned mode might start with the odd period if the channel has not been disabled before changing the alignment mode to center aligned.

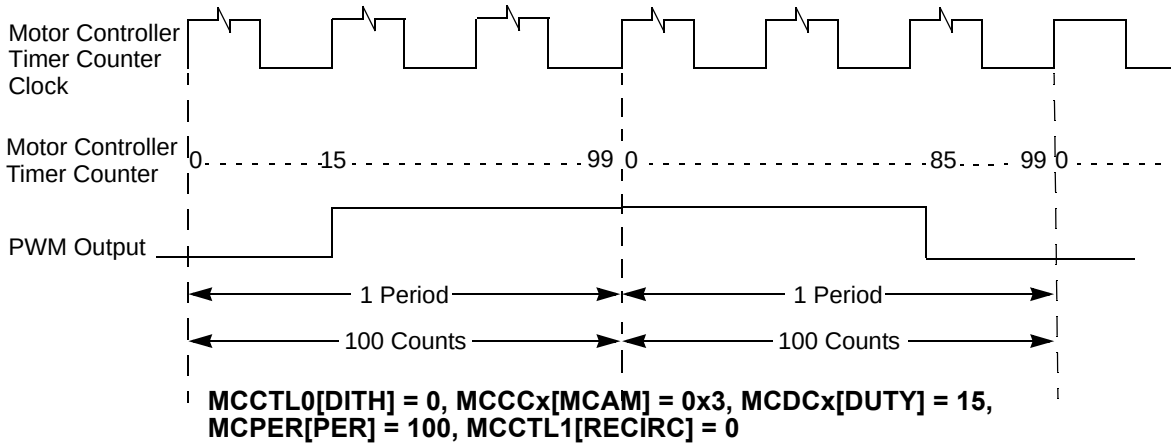



**Figure 41-21. Center Aligned**

#### 41.4.1.3.2 Sign Bit (MCDCx[SIGN])

Assuming MCCTL1[RECIRC] = 0 (the active state of the PWM signal is low), when the MCDCx[SIGN[4]] bit for the corresponding channel is cleared, MnC0P (if the PWM channel number is even, n = 0, 1, 2...5, see Table 41-19) or MnC1P (if the PWM channel number is odd, n = 0, 1, 2...5 see Table 41-19), outputs a logic high while in (dual) full H-bridge mode. In half H-bridge mode the state of the MCDCx[SIGN[4]] bit has no effect. The PWM output signal is generated on MnC0M (if the PWM channel number is even, n = 0, 1, 2...5, see Table 41-19) or MnC1M (if the PWM channel number is odd, n = 0, 1, 2...5).

Assuming MCCTL1[RECIRC] = 0 (the active state of the PWM signal is low), when the MCDCx[SIGN[4]] bit for the corresponding channel is set, MnC0M (if the PWM channel number is even, n = 0, 1, 2...5, see Table 41-19) or MnC1M (if the PWM channel number is odd, n = 0, 1, 2...5, see Table 41-19), outputs a logic high while in (dual) full H-bridge mode. In half H-bridge mode the state of the MCDCx[SIGN[4]] bit has no effect. The PWM output signal is generated on MnC0P (if the PWM channel number is even, n = 0, 1, 2...5, see Table 41-19) or MnC1P (if the PWM channel number is odd, n = 0, 1, 2...5).

Setting MCCTL1[RECIRC] = 1 will also invert the effect of the MCDCx[SIGN[4]] bit such that while MCDCx[SIGN[4]] = 0, MnC0P or MnC1P will generate the PWM signal and MnC0M or MnC1M will be a static low output. While MCDCx[SIGN[4]] = 1, MnC0M or MnC1M will generate the PWM signal and MnC0P or MnC1P will be a static low output. In this case the active state of the PWM signal will be high.

See Table 41-20 for detailed information about the impact of MCDCx[SIGN[4]] and MCCTL1[RECIRC] bit on the PWM output.

**Table 41-20. Impact of MCCTL1[RECIRC] and MCDCx[SIGN[4]] Bit on the PWM Output**

| Output Mode | MCCTL1[RECIRC] | MCDCx[SIGN[4]] | MnCyM | MnCyP |
|---|---|---|---|---|
| (Dual) Full H-Bridge | 0 | 0 | $\overline{\text{PWM}}$[1] | 1 |
| (Dual) Full H-Bridge | 0 | 1 | 1 | $\overline{\text{PWM}}$ |

**Table 41-20. Impact of MCCTL1[RECIRC] and MCDCx[SIGN[4]] Bit on the PWM Output (continued)**

| Output Mode | MCCTL1[RECIRC] | MCDCx[SIGN[4]] | MnCyM | MnCyP |
|---|---|---|---|---|
| (Dual) Full H-Bridge | 1 | 0 | 0 | PWM[2] |
| (Dual) Full H-Bridge | 1 | 1 | PWM | 0 |
| Half H-Bridge: PWM on MnCyM | Don't care | Don't care | $\overline{\text{PWM}}$ | —[3] |
| Half H-Bridge: PWM on MnCyP | Don't care | Don't care | — | $\overline{\text{PWM}}$ |

1 $\overline{\text{PWM}}$: The PWM signal is low active. e.g., the waveform starts with 0 in left aligned mode. Output M generates the PWM signal. Output P is static high.

2 PWM: The PWM signal is high active. e.g., the waveform starts with 1 in left aligned mode. output P generates the PWM signal. Output M is static low.

3 The state of the output transistors is not controlled by the SMC.

#### 41.4.1.3.3 Recirculation Bit (MCCTL1[RECIRC])

The MCCTL1[RECIRC] bit controls the flow of the recirculation current of the load. Setting MCCTL1[RECIRC] = 0 will cause recirculation current to flow through the high side transistors, and MCCTL1[RECIRC] = 1 will cause the recirculation current to flow through the low side transistors. The MCCTL1[RECIRC] bit is only active in (dual) full H-bridge modes.

Effectively, MCCTL1[RECIRC] = 0 will cause a static high output on the output terminal not driven by the PWM, MCCTL1[RECIRC] = 1 will cause a static low output on the output terminals not driven by the PWM. To achieve the same current direction, the MCDCx[SIGN[4]] bit behavior is inverted if MCCTL1[RECIRC] = 1. Figure 41-22, Figure 41-23, Figure 41-24, and Figure 41-25 illustrate the effect of the MCCTL1[RECIRC] bit in (dual) full H-bridge modes.

MCCTL1[RECIRC] bit must be changed only while no PWM channel is operated in (dual) full H-bridge mode.

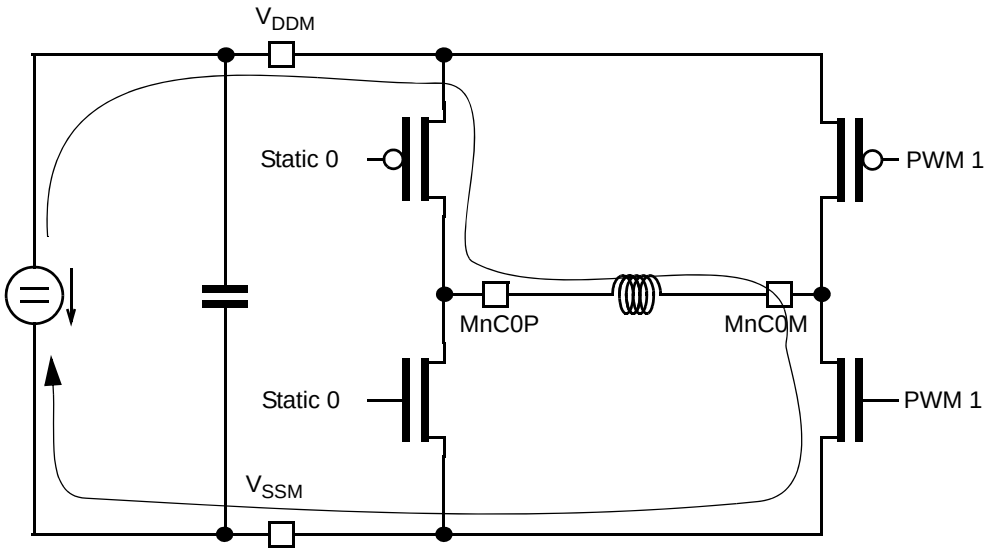



**Figure 41-22. PWM Active Phase, MCCTL1[RECIRC] = 0, MCDCx[SIGN[4]] = 0**

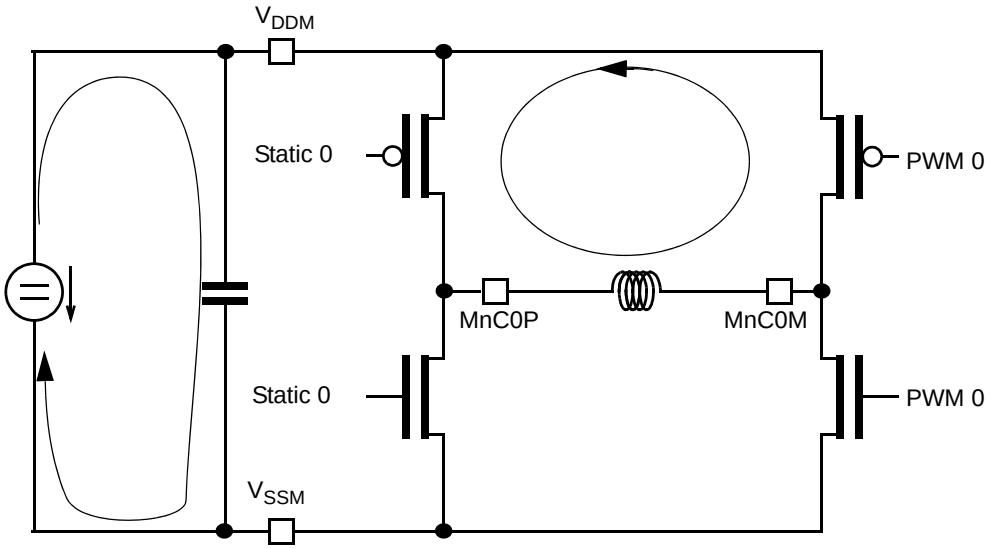



**Figure 41-23. PWM Passive Phase, MCCTL1[RECIRC] = 0, MCDCx[SIGN[4]] = 0**

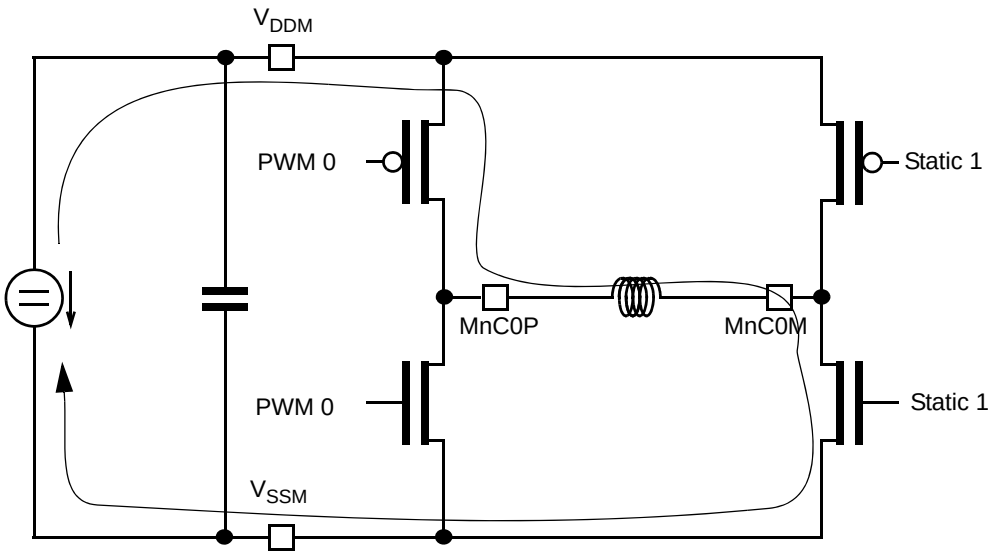



**Figure 41-24. PWM Active Phase, MCCTL1[RECIRC] = 1, MCDCx[SIGN[4]] = 0**

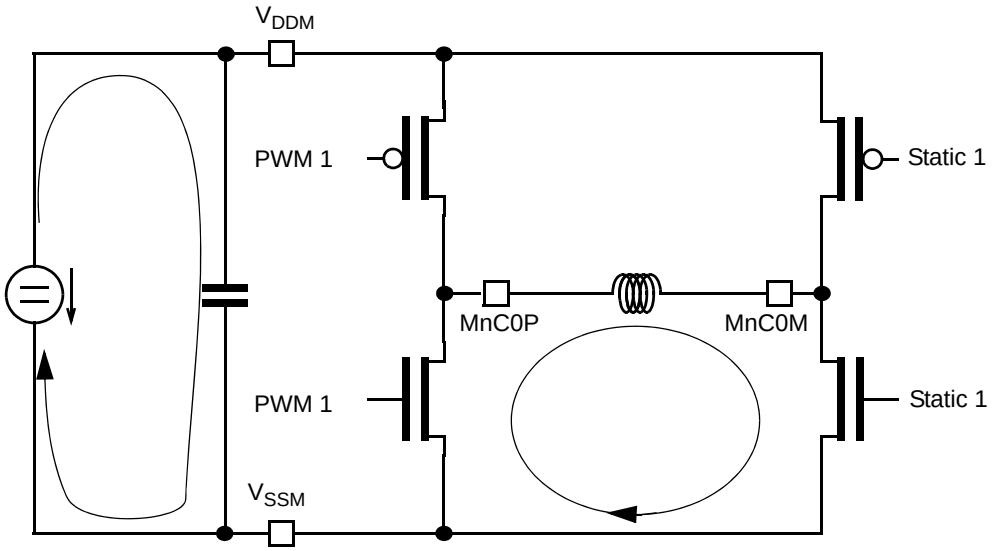



**Figure 41-25. PWM Passive Phase, MCCTL1[RECIRC] = 1, MCDCx[SIGN[4]] = 0**

#### 41.4.1.3.4 Relationship Between MCCTL1[RECIRC] Bit, MCDCx[SIGN[4]] Bit, MCCCx[MCOM] Bits, PWM State, and Output Transistors

Please refer to Figure 41-26 for the output transistor assignment.

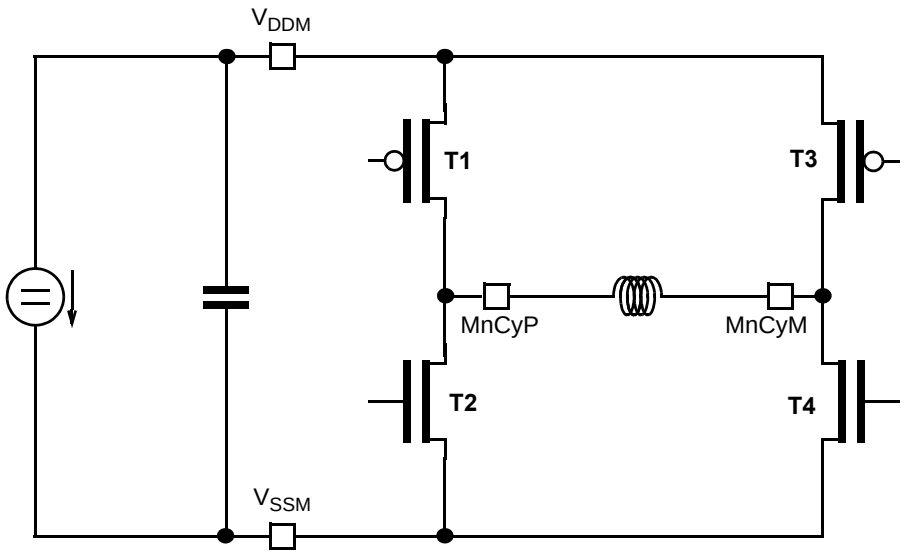



**Figure 41-26. Output Transistor Assignment**

Table 41-21 illustrates the state of the output transistors in different states of the SMC module. '—' means that the state of the output transistor is not controlled by the SMC.

**Table 41-21. State of Output Transistors in Various Modes**

| Mode | MCCCx [MCOM] | PWM Duty[1] | MCCTL1 [RECIRC] | MCDCx [SIGN[4]] | T1 | T2 | T3 | T4 | MnCyM | MnCyP |
|---|---|---|---|---|---|---|---|---|---|---|
| Off | Don't care | — | Don't care | Don't care | — | — | — | — | — | — |
| Half H-Bridge | 0x0 | Active | Don't care | Don't care | — | — | OFF | ON | 0 | — |
| Half H-Bridge | 0x0 | Passive | Don't care | Don't care | — | — | ON | OFF | 1 | — |
| Half H-Bridge | 0x1 | Active | Don't care | Don't care | OFF | ON | — | — | — | 0 |
| Half H-Bridge | 0x1 | Passive | Don't care | Don't care | ON | OFF | — | — | — | 1 |
| (Dual) Full | 0x2 or 0x3 | Active | 0 | 0 | ON | OFF | OFF | ON | 0 | 1 |
| (Dual) Full | 0x2 or 0x3 | Passive | 0 | 0 | ON | OFF | ON | OFF | 1 | 1 |
| (Dual) Full | 0x2 or 0x3 | Active | 0 | 1 | OFF | ON | ON | OFF | 1 | 0 |
| (Dual) Full | 0x2 or 0x3 | Passive | 0 | 1 | ON | OFF | ON | OFF | 1 | 1 |
| (Dual) Full | 0x2 or 0x3 | Active | 1 | 0 | ON | OFF | OFF | ON | 0 | 1 |
| (Dual) Full | 0x2 or 0x3 | Passive | 1 | 0 | OFF | ON | OFF | ON | 0 | 0 |
| (Dual) Full | 0x2 or 0x3 | Active | 1 | 1 | OFF | ON | ON | OFF | 1 | 0 |
| (Dual) Full | 0x2 or 0x3 | Passive | 1 | 1 | OFF | ON | OFF | ON | 0 | 0 |

1 When in (Dual) Full mode and RECIRC=0, the PWM is 0 when the duty cycle is active and 1 when it is passive. When RECIRC = 1, the opposite is true.

#### 41.4.1.3.5 Dither Bit (MCCTL0[DITH])

The purpose of the dither mode is to increase the minimum length of output pulses without decreasing the PWM resolution, in order to limit the pulse distortion introduced by the slew rate control of the outputs. If dither mode is selected the output pattern will repeat after two timer counter overflows. For the same output frequency, the shortest output pulse will have twice the length while dither feature is selected. To achieve the same output frame frequency, the prescaler of the SMC module has to be set to twice the division rate if dither mode is selected; e.g., with the same prescaler division rate the repeat rate of the output pattern is the same as well as the shortest output pulse with or without dither mode selected.

The MCCTL0[DITH] bit enables or disables the dither function.

MCCTL0[DITH] = 0: dither function is disabled.

When MCCTL0[DITH] is cleared and assuming left aligned operation and MCCTL1[RECIRC] = 0, the PWM output will start at a logic low level at the beginning of the PWM period (motor controller timer counter = 0x000). The PWM output remains low until the motor controller timer counter matches the 11-bit PWM duty cycle value MCDCx[DUTY]. When a match (output compare between motor controller timer counter and MCDCx[DUTY]) occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the contents of MCPER[PER] – 1). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level. This completes one PWM period. The PWM period repeats every MCPER[PER] counts of the motor controller timer counter. If MCDCx[DUTY] >= MCPER[PER], the output will be static low. If MCDCx[DUTY] = 0, the output will be continuously at a logic high level. The relationship between the motor controller timer counter clock, motor controller timer counter value, and PWM output while MCCTL0[DITH] = 0 is shown in Figure 41-27.

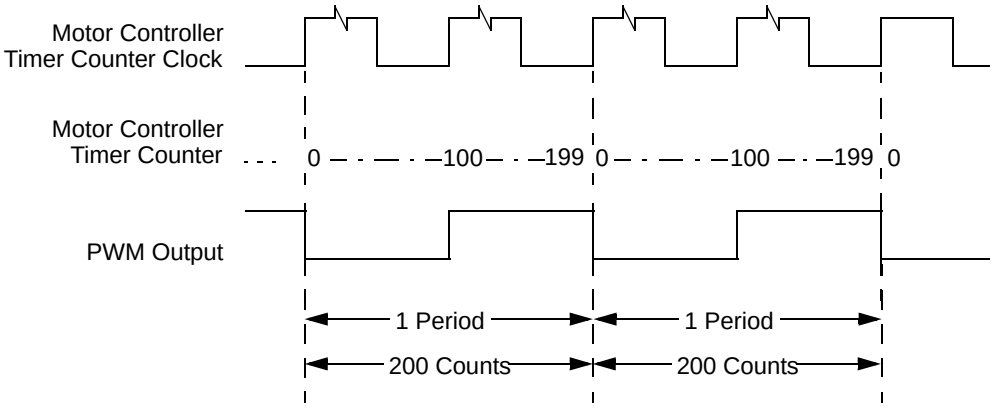



**Figure 41-27. PWM Output: MCCTL0[DITH] = 0, MCCCx[MCAM] = 0x1, MCDCx[DUTY] = 100, MCPER[PER] = 200, MCCTL1[RECIRC] = 0**

MCCTL0[DITH] = 1: dither function is enabled

Please note if MCCTL0[DITH] = 1, the bit MCPER[PER[0]] will be internally forced to 0 and read always as 0.

When MCCTL0[DITH] is set and assuming left aligned operation and MCCTL1[RECIRC] = 0, the PWM output will start at a logic low level at the beginning of the PWM period (when the motor controller timer counter = 0). The PWM output remains low until the motor controller timer counter matches the 10-bit

PWM duty cycle value MCDCx[DUTY]. When a match (output compare between motor controller timer counter and MCDCx[DUTY]) occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the value defined by MCPER[PER[10:1]] – 1). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level. This completes the first half of the PWM period. During the second half of the PWM period, the PWM output will remain at a logic low level until either the motor controller timer counter matches the 10-bit PWM duty cycle value MCDCx[DUTY] if MCDCx[DUTY[0]]= 0, or the motor controller timer counter matches the 10-bit PWM duty cycle value + 1 (the value of MCDCx[DUTY[10:1]] is incremented by 1 and is compared with the motor controller timer counter value) if MCDCx[DUTY[0]] = 1 for the corresponding channel. When a match occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the value defined by MCPER[PER[10:1]] – 1). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level.

This process will repeat every number of counts of the motor controller timer counter defined by the period register contents (MCPER[PER]). If the output is neither set to 0% nor to 100% there will be four edges on the PWM output per PWM period in this case. Therefore, the PWM output compare function will alternate between MCDCx[DUTY] and MCDCx[DUTY] + 1 every half PWM period if MCDCx[DUTY[0]] for the corresponding channel is set to 1. The relationship between the motor controller timer counter clock ($f_{TC}$), motor controller timer counter value, and left aligned PWM output if MCCTL0[DITH] = 1 is shown in Figure 41-28 and Figure 41-29. Figure 41-30 and Figure 41-31 show right aligned and center aligned PWM operation respectively, with dither feature enabled and MCDCx[DUTY[0]] = 1. Please note: In the following examples, the MCPER[PER] value, which is, if MCCTL0[DITH] = 1, always an even number.

**NOTE**

The MCCTL0[DITH] bit must be changed only if the SMC is disabled (all channels disabled or period register cleared) to avoid erroneous waveforms.

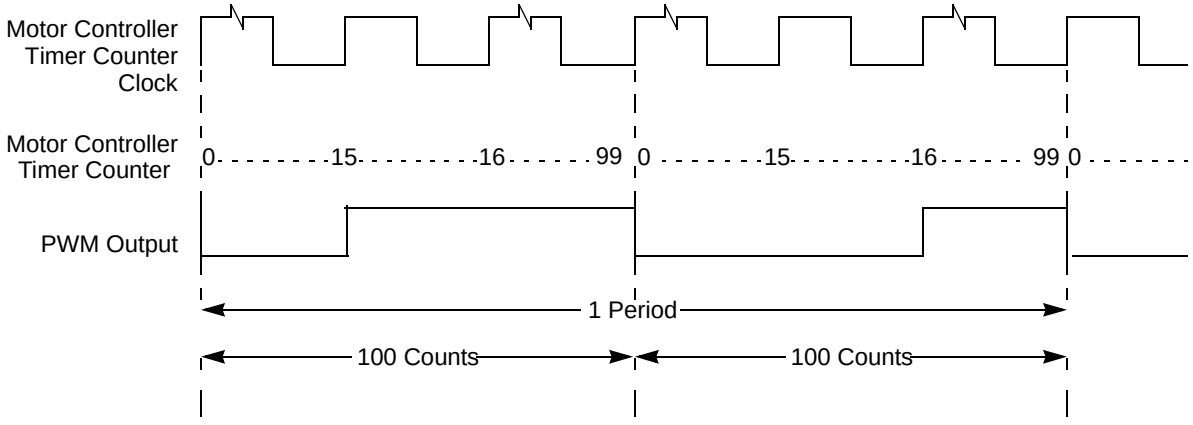



**Figure 41-28. PWM Output: MCCTL0[DITH] = 1, MCCCx[MCAM] = 0x1, MCDCx[DUTY] = 31, MCPER[PER] = 200, MCCTL1[RECIRC] = 0**

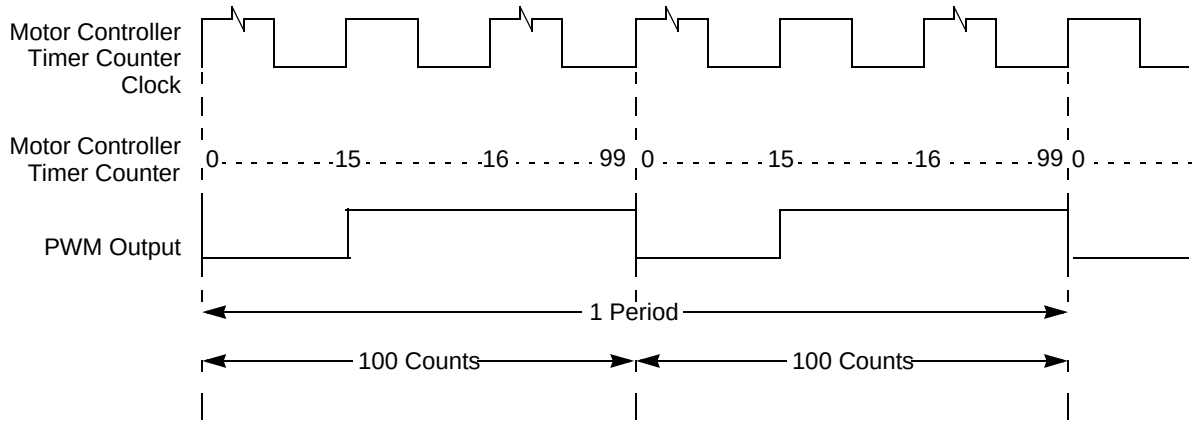



**Figure 41-29. PWM Output: MCCTL0[DITH] = 1, MCCCx[MCAM] = 0x1, MCDCx[DUTY] = 30, MCPER[PER] = 200, MCCTL1[RECIRC] = 0**

.

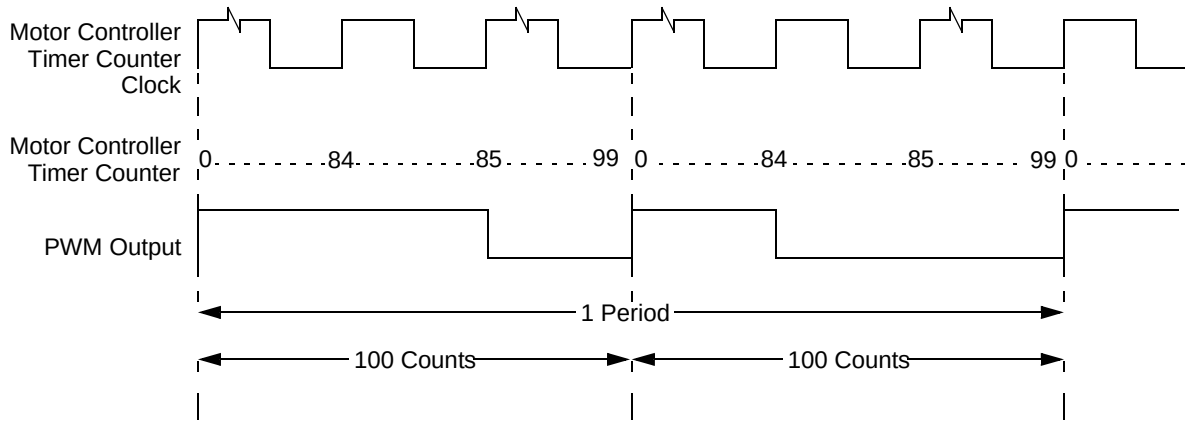



**Figure 41-30. PWM Output: MCCTL0[DITH] = 1, MCCCx[MCAM] = 0x2, MCDCx[DUTY] = 31, MCPER[PER] = 200, MCCTL1[RECIRC] = 0**

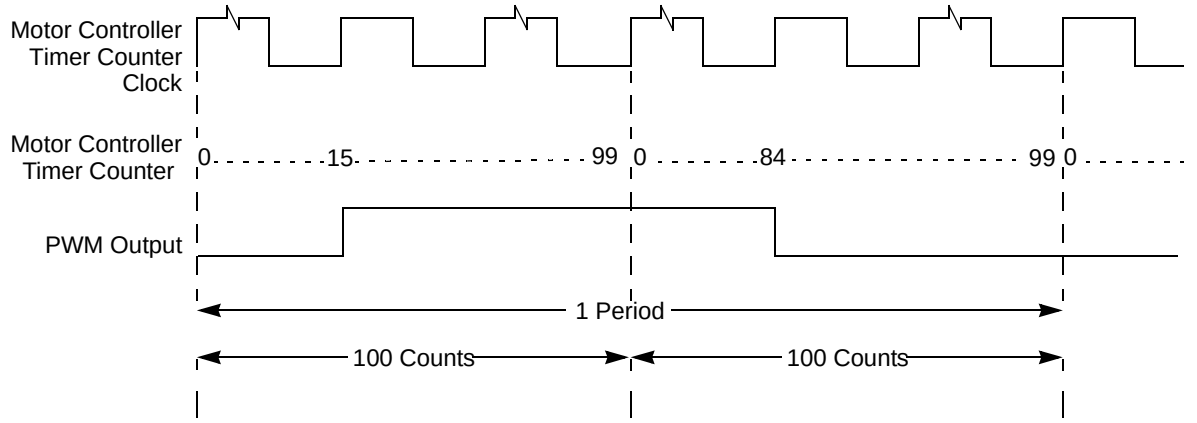



**Figure 41-31. PWM Output: MCCTL0[DITH] = 1, MCCCx[MCAM] = 0x3, MCDCx[DUTY] = 31, MCPER[PER] = 200, MCCTL1[RECIRC] = 0**

## 41.4.2 PWM Duty Cycle

The PWM duty cycle for the SMC channel x can be determined by dividing the decimal representation of the bits MCDCx[DUTY] by the decimal representation of the bits MCPER[PER] and multiplying the result by 100% as shown in Equation 41-1.

$$\text{Effective PWM Channel X \% Duty Cycle} = \frac{\text{DUTY}}{\text{MCPER}} \cdot 100\% \qquad \textit{Eqn. 41-1}$$

**NOTE**

x = PWM Channel Number = 0, 1, 2, 3 ... 11. This equation is only valid if MCDCx[DUTY] <= MCPER[PER] and MCPER[PER] is not equal to 0.

Whenever MCDCx[DUTY] >= MCPER[PER], a constant low level (MCCTL1[RECIRC] = 0) or high level (MCCTL1[RECIRC] = 1) will be output.

## 41.4.3 Motor Controller Counter Clock Source

Figure 41-32 shows how the PWM motor controller timer counter clock source is selected.

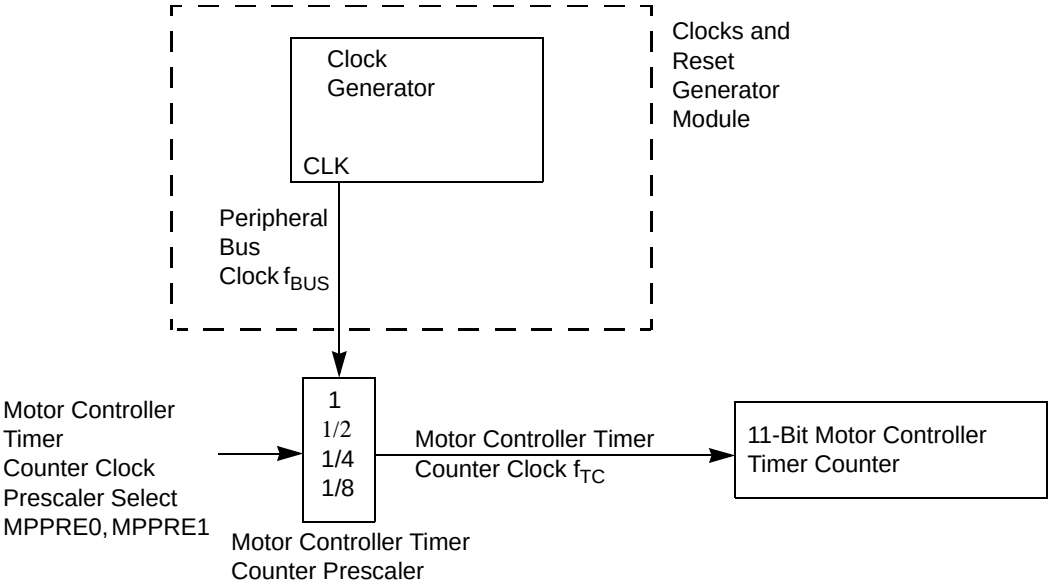



**Figure 41-32. Motor Controller Counter Clock Selection**

The peripheral bus clock is the source for the motor controller counter prescaler. The motor controller counter clock rate, $f_{TC}$, is set by selecting the appropriate prescaler value. The prescaler is selected with the MCCTL0[MCPRE] bits. The SMC channel frequency of operation can be calculated using Equation 41-2 if MCCTL0[DITH] = 0.

$$\text{Motor Channel Frequency (Hz)} = \frac{f_{TC}}{\text{MCPER} \cdot \text{M}} \qquad \textit{Eqn. 41-2}$$

The SMC channel frequency of operation can be calculated using Equation 41-3 if MCCTL0[DITH] = 1.

$$\text{Motor Channel Frequency (Hz)} = \frac{f_{TC}}{\text{MCPER} \cdot \text{M}/2} \qquad \textit{Eqn. 41-3}$$

**NOTE**

Both equations are only valid if MCPER[PER] is not equal to 0. M = 1 for left or right aligned mode, M = 2 for center aligned mode.

Table 41-22 shows examples of the SMC channel frequencies that can be generated based on different peripheral bus clock frequencies and the prescaler value.

**Table 41-22. SMC Channel Frequencies (Hz), MCPER[PER] = 256, MCCTL0[DITH] = 0, MCCCx[MCAM] = 0x2, 0x1**

| Prescaler | Peripheral Bus Clock Frequency | | | | |
|---|---|---|---|---|---|
| | 16 MHz | 10 MHz | 8 MHz | 5 MHz | 4 MHz |
| 1 | 62500 | 39063 | 31250 | 19531 | 15625 |
| 1/2 | 31250 | 19531 | 15625 | 9766 | 7813 |
| 1/4 | 15625 | 9766 | 7813 | 4883 | 3906 |
| 1/8 | 7813 | 4883 | 3906 | 2441 | 1953 |

**NOTE**

Due to the selectable slew rate control of the outputs, clipping may occur on short output pulses.

### 41.4.4 Output Switching Delay

In order to prevent large peak current draw from the motor power supply, selectable delays can be used to stagger the high logic level to low logic level transitions on the SMC outputs. The timing delay, $t_d$, is determined by the MCCCx[CD] bits in the corresponding channel control register and is selectable between 0, 1, 2, or 3 motor controller timer counter clock cycles.

**NOTE**

A PWM channel gets disabled at the next timer counter overflow without notice of the switching delay.

### 41.4.5 Operation in SMC stop mode

All module clocks are stopped and the associated port pins are set to their inactive state, which is defined by the state of the MCCTL1[RECIRC] bit. The SMC module registers stay the same as they were prior to entering stop mode. Therefore, after exiting from stop mode, the associated port pins will resume to the same functionality they had prior to entering stop mode.

### 41.4.6 Short-circuit detection

Each PWM pin is equipped with a short-circuit detection function. Hence, 24 instances (4 for each PWM module) of the short-circuit detector exist.

**Table 41-23. Cross-reference PWM Signal to Short-circuit Detector Register Bits**

| Short-Circuit Detector Index *sd* | PWM Channel | Pin Name | Related SD Enable Bit (see 41.3.2.7) | Related SD Int Enable Bit (see 41.3.2.10) | Related SD Int Bit (see 41.3.2.13) |
|---|---|---|---|---|---|
| 23 | 10 | M5C0M | MCSDE2[SDEN[7]] | MCSDIEN2[SDIE[7]] | MCSDI2[SDIF[7]] |
| 22 | 8 | M4C0M | MCSDE2[SDEN[6]] | MCSDIEN2[SDIE[6]] | MCSDI2[SDIF[6]] |
| 21 | 6 | M3C0M | MCSDE2[SDEN[5]] | MCSDIEN2[SDIE[5]] | MCSDI2[SDIF[5]] |
| 20 | 4 | M2C0M | MCSDE2[SDEN[4]] | MCSDIEN2[SDIE[4]] | MCSDI2[SDIF[4]] |
| 19 | 2 | M1C0M | MCSDE2[SDEN[3]] | MCSDIEN2[SDIE[3]] | MCSDI2[SDIF[3]] |
| 18 | 0 | M0C0M | MCSDE2[SDEN[2]] | MCSDIEN2[SDIE[2]] | MCSDI2[SDIF[2]] |
| 17 | 11 | M5C1M | MCSDE2[SDEN[1]] | MCSDIEN2[SDIE[1]] | MCSDI2[SDIF[1]] |
| 16 | 9 | M4C1M | MCSDE2[SDEN[0]] | MCSDIEN2[SDIE[0]] | MCSDI2[SDIF[0]] |
| 15 | 7 | M3C1M | MCSDE1[SDEN[7]] | MCSDIEN1[SDIE[7]] | MCSDI1[SDIF[7]] |
| 14 | 5 | M2C1M | MCSDE1[SDEN[6]] | MCSDIEN1[SDIE[6]] | MCSDI1[SDIF[6]] |
| 13 | 3 | M1C1M | MCSDE1[SDEN[5]] | MCSDIEN1[SDIE[5]] | MCSDI1[SDIF[5]] |
| 12 | 1 | M0C1M | MCSDE1[SDEN[4]] | MCSDIEN1[SDIE[4]] | MCSDI1[SDIF[4]] |
| 11 | 10 | M5C0P | MCSDE1[SDEN[3]] | MCSDIEN1[SDIE[3]] | MCSDI1[SDIF[3]] |
| 10 | 8 | M4C0P | MCSDE1[SDEN[2]] | MCSDIEN1[SDIE[2]] | MCSDI1[SDIF[2]] |
| 9 | 6 | M3C0P | MCSDE1[SDEN[1]] | MCSDIEN1[SDIE[1]] | MCSDI1[SDIF[1]] |
| 8 | 4 | M2C0P | MCSDE1[SDEN[0]] | MCSDIEN1[SDIE[0]] | MCSDI1[SDIF[0]] |
| 7 | 2 | M1C0P | MCSDE0[SDEN[7]] | MCSDIEN0[SDIE[7]] | MCSDI0[SDIF[7]] |
| 6 | 0 | M0C0P | MCSDE0[SDEN[6]] | MCSDIEN0[SDIE[6]] | MCSDI0[SDIF[6]] |
| 5 | 11 | M5C1P | MCSDE0[SDEN[5]] | MCSDIEN0[SDIE[5]] | MCSDI0[SDIF[5]] |
| 4 | 9 | M4C1P | MCSDE0[SDEN[4]] | MCSDIEN0[SDIE[4]] | MCSDI0[SDIF[4]] |
| 3 | 7 | M3C1P | MCSDE0[SDEN[3]] | MCSDIEN0[SDIE[3]] | MCSDI0[SDIF[3]] |
| 2 | 5 | M2C1P | MCSDE0[SDEN[2]] | MCSDIEN0[SDIE[2]] | MCSDI0[SDIF[2]] |
| 1 | 3 | M1C1P | MCSDE0[SDEN[1]] | MCSDIEN0[SDIE[1]] | MCSDI0[SDIF[1]] |
| 0 | 1 | M0C1P | MCSDE0[SDEN[0]] | MCSDIEN0[SDIE[0]] | MCSDI0[SDIF[0]] |

Each single short-circuit detector is a timer, measuring the time during which the signals PWM and FB are not equal (see Figure 41-33). If this time is greater than or equal to the time represented by MCSDTO[TOUT], then a short-circuit is assumed.

To enable the short-circuit detector on a pin, ensure that the pin's Input Buffer is enabled in the corresponding Pad Configuration Register in the SIU module.

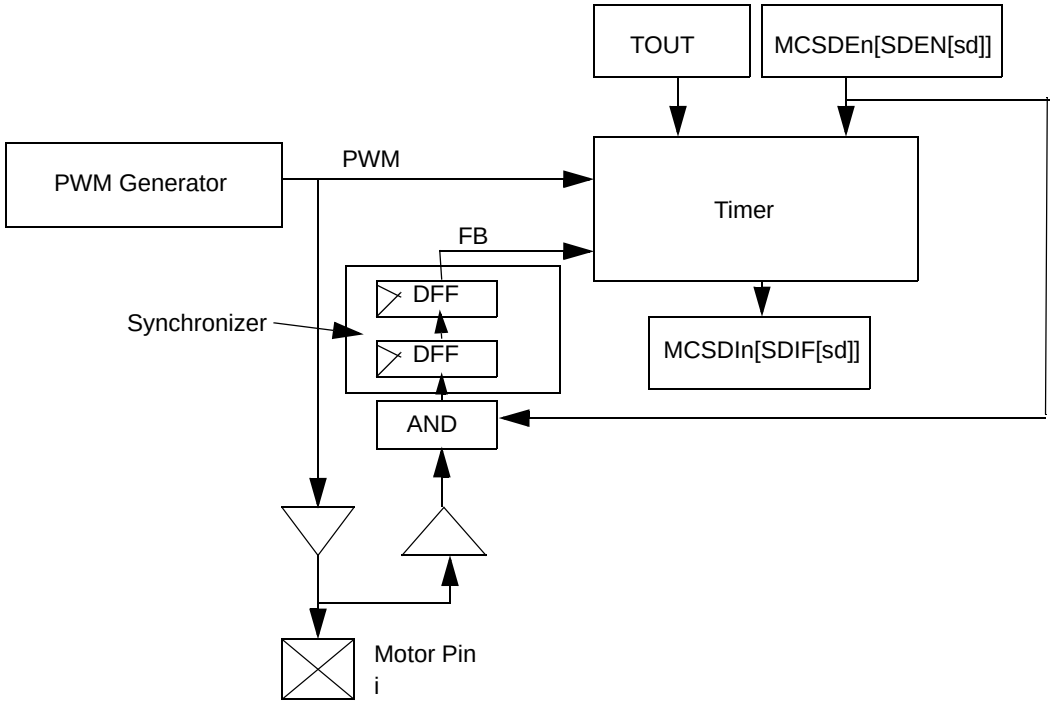
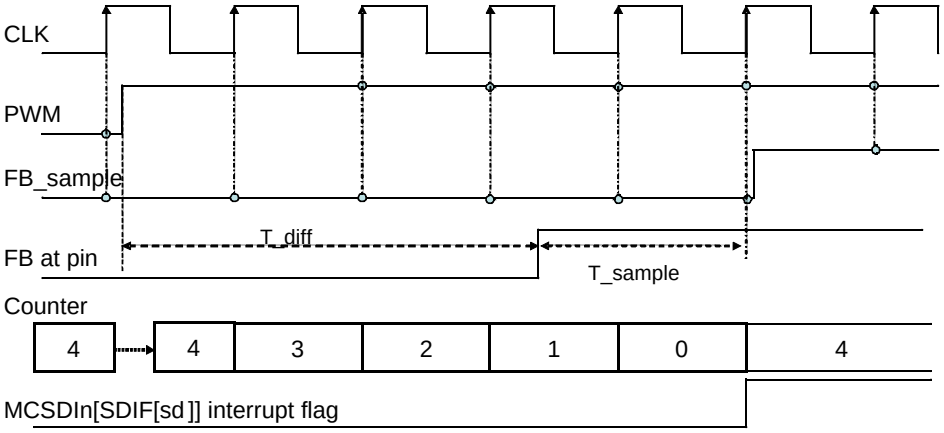



**Figure 41-33. Short-circuit Detector Overview**




| | |
|---|---|
| T_sample | :Sample and synchronization delay |
| T_diff | :Delay between FB at pin and the internal PWM signal |
| PWM | :Signal from motor controller |
| FB at pin | :This is the signal directly after the input driver of the pad s ignal |
| FB_Sample | :Pad signal after sampling and synchronization |

**Figure 41-34. Example for MCSDTO[TOUT] = 4**

- Hence, if PWM != FB, the timer starts counting if the short-circuit detector is enabled at least one clock cycle before (see MCSDE0, MCSDE1 or MCSDE2")
- If PWM == FB, the timer stops and is reset to the time-out value MCSDTO[TOUT] in order to be ready for the next transaction
- IF PWM != FB and the timer state is larger or equal MCSDTO[TOUT], than
  - — One of the interrupt flags MCSDI0[SDIF], MCSDE1[SDIF] or MCSDI2[SDIF] is set in order to flag a short-circuit.
  - — The interrupt flag is cleared by writing one, by reset or by disabling the short-circuit detector
  - — The timer is stopped and reloaded with the time-out value MCSDTO[TOUT] in order to be ready for the next transaction
- If a short-circuit detector is disabled (MCSDEn[SDEN[*sd*]] == 0), the related short-circuit detector counter is halted and preloaded with the register value of MCSDTO[TOUT]. The related bit in MCSDIn[SDIF[*sd*]] of this specific short-circuit detector is set to 0. This means that, if all short-circuit detectors are disabled, all bits of MCSDIn stay at 0. No interrupt from the detector can be generated independently of the interrupt mask in MCSDIENn.

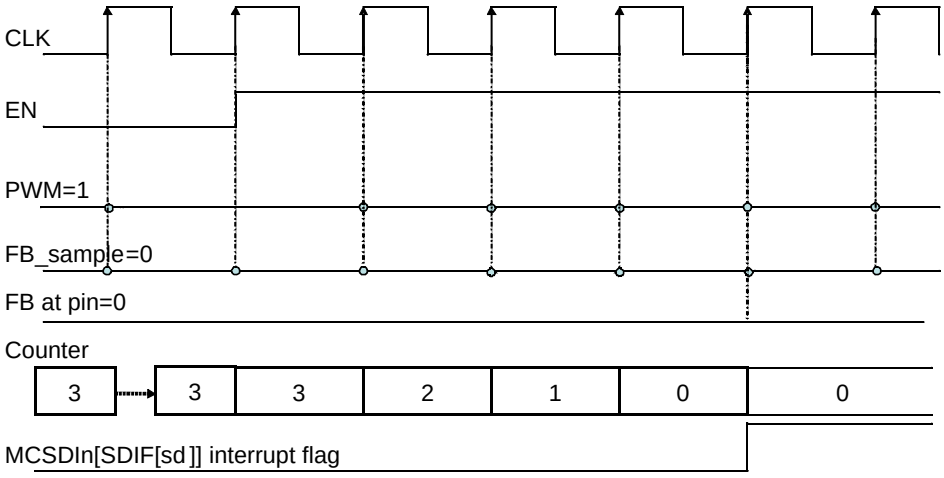
- In case of low power-modes, the state of the short-circuit detector is frozen. After exit of the low power mode, the short-circuit detector will resume operation from the previous state. If the short-circuit detector should restart with defined state (counter value = MCSDTO[TOUT], than the related detector shall be disabled and enabled again. This will reload the counter with the MCSDTO[TOUT] value and restart the short-circuit detector.

The maximum time span which the timer can cover depends on the clock frequency F of the main clock. The maximum delay D covered by the counter is D = MCSDTO[TOUT]* F. Due to sampling and synchronization of the feedback signal, the value of MCSDTO[TOUT] must always be larger than 2.

The two synchronizer stages imply also, that a a short-circuit with a duration of less than or equal to 2 clock cycles cannot be detected.

Two special cases shall be highlighted:

- Static short-circuit to ground and PWM signal = 1 see Figure 41-35: In this case, the enable of the short-circuit detector starts the sampling process and the interrupt bit is set MCSDTO[TOUT]+1 cycles after enabling the short-circuit detector
- Static short-circuit to VDD and PWM signal = 0 see Figure 41-36: In this case. the enable of the short-circuit detector starts the sampling process and the interrupt bit is set MCSDTO[TOUT]+3 cycles after enabling the short-circuit detector due to the synchronizer which has been cleared during disable of the short-circuit detector




| | |
|---|---|
| T_sample | :Sample and synchronization delay |
| T_diff | :Delay between FB at pin and the internal PWM signal |
| PWM | :Signal from motor controller |
| FB at pin | :This is the signal directly after the input driver of the pad s ignal |
| FB_Sample | :Pad signal after sampling and synchronization |

**Figure 41-35. Static Short-circuit, PWM signal always at 1 and FB always at 0, MCSDTO[TOUT]=3**

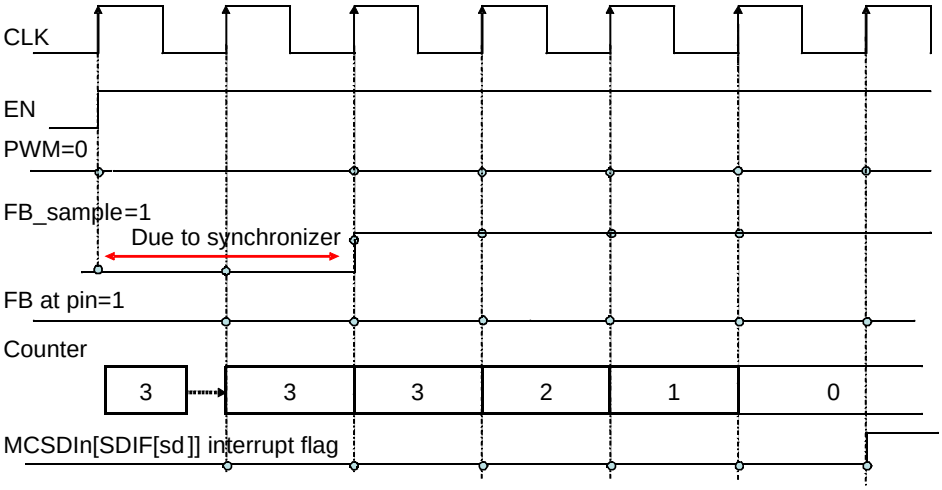



**Figure 41-36. Static Short-circuit, PWM signal always at 0 and FB always at 1, MCSDTO[TOUT]=3**

**NOTE**

The short-circuit detection block does not disable a port in case of a short-circuit. It is task of the microcontroller to manage the event of a short-circuit.

## 41.5 Reset

The SMC is reset by system reset. All associated ports are released, all registers of the SMC module will switch to their reset state as defined in Section 41.3.2, “Register description”.

## 41.6 Interrupts

The SMC has one interrupt output which is the bitwise OR function of 25 individual interrupt request sources:

- One time counter overflow interrupt: An interrupt will be requested when the MCCTL1[MCTOIE] bit in is set and the running PWM frame is finished. The interrupt is cleared by either setting the MCCTL1[MCTOIE] bit to 0 or to write a one to the MCCTL0[MCTOIF] bit
- 24 Interrupts for the short-circuit detection, one for each PWM pin: Whenever a short-circuit is detected on one PWM pin and the short-circuit detector enable bit MCSDEn[SDEN[*sd*]] is set, than the related interrupt flag MCSDIn[SDIF[*sd*]] is set according to the mapping shown in Table 41-23. The interrupt flag in MCSDIn[SDIF[*sd*]] will also rise an external interrupt if the

interrupt enable bit MCSDIENn[SDIE[*sd*]] is set. To clear the interrupt flag, either write a one into the related bit position MCSDIn[SDIF[*sd*]] or disable the related short-circuit detector by writing zero to MCSDEn[SDEN[*sd*]]. If the short-circuit detector is enabled and a static short-circuit exists, then the MCSDIn[SDIF[*sd*]] flag will be asserted directly after clearing it, because the short-circuit detector is still enabled and will detect the short-circuit again. To avoid this behavior, disable the short-circuit detector channel after detection of the short-circuit.

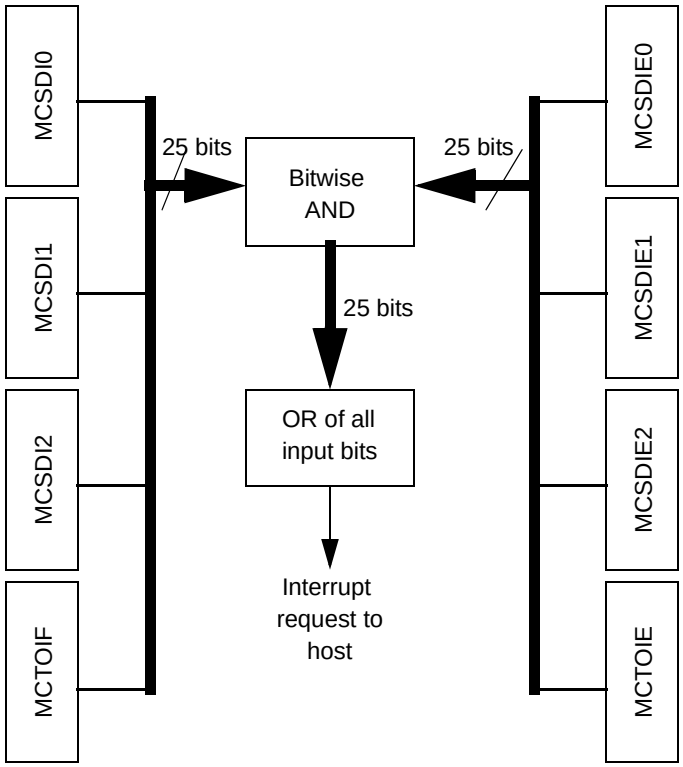



**Figure 41-37. SMC Interrupt Generation**

# Chapter 42
# Stepper Stall Detect (SSD)

## 42.1 Introduction

### 42.1.1 Overview

The SSD block connects to one stepper motor (SM) with two coils. It can be used to monitor the movement of the SM to detect that the attached gauge pointer has reached the stall position of the scale.

Basis of the movement detection is to drive one of the coils and to integrate the back EMF (electromotive force) induced in the other coil. This back EMF is present only if the SM is rotating. Therefore, if the integral of the back EMF exceeds a certain threshold, the SM is still rotating; otherwise it can be regarded as being stalled.

The SSD block shares the pins connected to the SM coils together with the motor controller block responsible for driving the SM in the main application (e.g. moving the gauge pointer to a certain position of the scale).

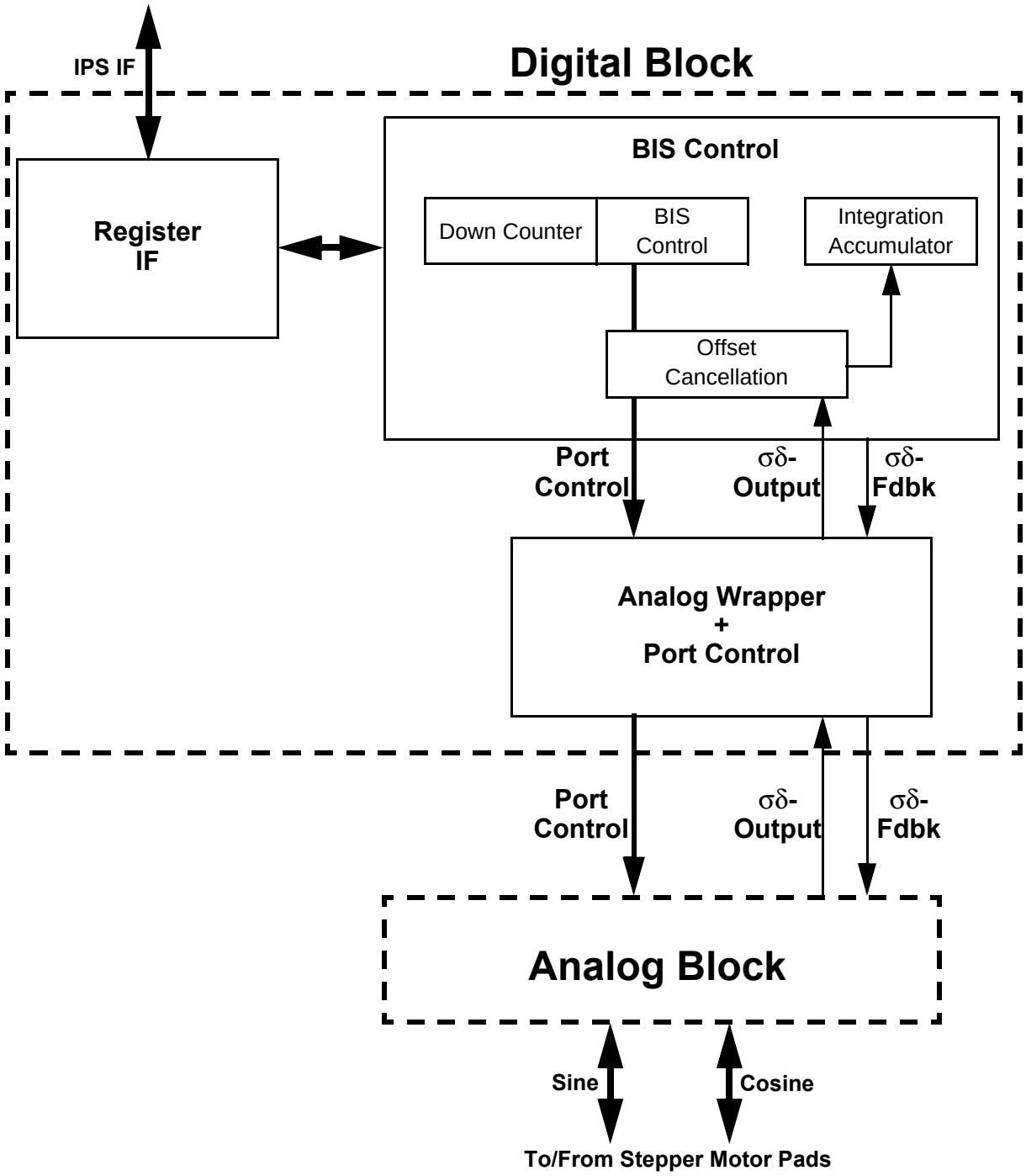



**Figure 42-1.SSD Overall Block Diagram**

The names of the sub blocks given in the diagram above relate to the descriptions given in Section 42.4.1, "Main Building Blocks of the SSD":

- 'Analog Block' relates to Section 42.4.1.1, "Analog Block".
- 'Analog Wrapper and Port Control' relates to Section 42.4.1.2, "Analog Wrapper + Port Control".

- ‘Register Interface’ relates to Section 42.4.1.3, “Register Interface”.
- ‘BIS (blanking-integration sequence) Control Logic’ relates to Section 42.4.1.4, “BIS Control”.

## 42.1.2 Features

The most important features of the SSD block are listed below:

- Programmable Full Step States according to 2 coil stepper motors (4 states).
- Programmable Integration Polarity
- Integration Accumulator of 16 bits with programmable clock divider.
- Programmable Down Counter (16 bit timer).
  - 64 MHz bus clock: Finest resolution in time down to 125 ns, maximum length of blanking or integration phase for this case is ~8.2 ms.
  - 64 MHz bus clock: Maximum length of blanking or integration phase is 1.05 s, resolution in time reduced to 16 µs for this case.
- Automatic sequence of blanking followed by integration (BIS) triggered by the user.
- Full flexibility over blanking and integration phase of the BIS:
  - Separate down counter initialization values and divider factors.
  - Separate interrupt flags and interrupt enable bits
  - Separate coil drive enable bits.
- Polarity switching to cancel DC offset errors programmable. The down counter value for the integration phase can be divided by 2, 4 or 8 to switch the polarity during the integration phase. Additionally the offset cancellation can be switched off completely.
- Seamless changeover into or from SSD (stepper stall detect) mode: The coil control signals outside of the BIS are fully programmable.

## 42.1.3 Modes of Operation

This section describes the different modes of operation of the SSD block.

### 42.1.3.1 Disabled Mode

In this mode the SSD block is disabled. None of the SM coils is driven nor one of the coils sensed. This is the case if the RTZE bit in the CONTROL register is cleared.

### 42.1.3.2 Normal Mode

In this mode the SSD block needs exclusive control over the SM coils. This must be ensured on at the device level. Setting the RTZE bit in the CONTROL register is not sufficient since the SSD does not provide a global port enable signal. Refer to Section 42.4, “Functional Description” for more details.

#### 42.1.3.3 Power Down Modes

When the SSD is disabled by the Mode Entry module, the analog block and the system clocks driving the SSD blocks are disabled.

## 42.2 External Signal Description

Each of the four analog I/Os of the SSD block is used either as the output of a half-bridge sourcing or sinking the current of the SM coil driven currently or - in case of acting as an input - the back EMF of the non-driven coil with respect to an internally-generated reference voltage is supplied to the σδ-modulator of the analog block. The signal properties are given in Table 42-1.

**Table 42-1. Signal Properties**

| Name | Port | Coil | Coil Node | I/O | Reset |
|---|---|---|---|---|---|
| COSP | COSP | Cosine | Plus | Analog I/O | Z |
| COSM | COSM | | Minus | Analog I/O | Z |
| $\overline{\text{SINP}}$ | SINP | Sine | Plus | Analog I/O | Z |
| SINM | SINM | | Minus | Analog I/O | Z |

## 42.3 Memory Map and Register Definition

This section provides a detailed description of the registers of the SSD block. Note that all registers are 16 bits in width. There is no access on byte level.

### 42.3.1 Memory Map

Table 42-2 lists the registers of the SSD block.

**Table 42-2. Block Memory Map**

| Offset | Register Name (Long) | Register Name (Short) | Access[1] | Location |
|---|---|---|---|---|
| 0x00 | SSD Control and Status Register | CONTROL | R/W | on page 1455 |
| 0x02 | SSD Interrupt Flag and Enable Register | IRQ | R/W | on page 1457 |
| 0x04 | SSD Integrator Accumulator Register | ITGACC | R | on page 1458 |
| 0x06 | SSD Down Counter Count register | DCNT | R | on page 1458 |
| 0x08 | SSD Blanking Counter Load Register | BLNCNTLD | R/W | on page 1459 |
| 0x0A | SSD Integration Counter Load Register | ITGCNTLD | R/W | on page 1459 |
| 0x0C | SSD Prescaler Register | PRESCALE | R/W | on page 1460 |
| 0x0E | RESERVED[2] | | n/a | — |

[1] Note that R/W registers may contain some read-only or write-only bits.

[2] Read access provides 0x0000. No write allowed.

## 42.3.2 Register Summary

Figure 42-2 and Table 42-3 below illustrate the different access modes of some register bits.

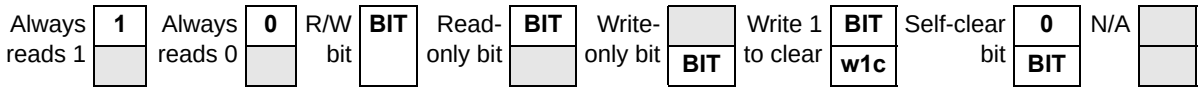



**Figure 42-2. Key to Register Fields**

Table 42-3 provides a key for register figures and tables.

**Table 42-3. Register Conventions**

| Convention | Description |
|---|---|
| | Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable. |
| FIELDNAME | Identifies the field. Its presence in the read or write row indicates that it can be read or written. |
| **Register Field Types** | |
| R | Read only. Writing this bit has no effect. Register content is updated in hardware. |
| W | Write only. If no read value is given any read access will provide undefined results. Refer to description of individual bits for additional effects of read. |
| R/W | Standard read/write bit. Only software can change the bit's value (other than a hardware reset). |
| w1c | Write one to clear. A status bit that can be read, and is cleared by writing a one. |
| **Reset Values** | |
| 0 | Resets to zero. |
| 1 | Resets to one. |
| — | Undefined at reset. |
| u | Unaffected by reset. |
| [*signal_name*] | Reset value is determined by polarity of indicated signal. |

## 42.3.3 Register Descriptions

This section describes the individual bits of all the SSD registers. Note that the details of the functional description linked to these bits is given in Section 42.4, "Functional Description".

### 42.3.3.1 SSD Control and Status Register (CONTROL)

Figure 42-3 below describes the fields of the main control (CONTROL) register:

Offset 0x00 Access: User read/write

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | STEP | | RCIR | ITGDIR | BLNDCL | ITGDCL | RTZE | 0 | BLNST | ITGST | 0 | 0 | 0 | SDCPU | D_RSVD |
| W | TRIG | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 42-3. SSD Control and Status Register (CONTROL)**

The function of the CONTROL register bits is shown in Table 42-4.

**Table 42-4. CONTROL Register Field Description**

| Field | Description |
|---|---|
| 15 | TRIG - Trigger Blanking -> Integration sequence (BIS).<br>1 Sequence of blanking -> integration is triggered.<br>0 No effect. |
| 14-13 | STEP - Full Step State. These bits determine which coil is driven for SM movement,. Refer to Table 42-11 for details of the step states.<br>00 Select0$^{o}$ angle (east pole) state for the electromagnetic field in the SM.<br>01 Select90$^{o}$ angle (north pole) state for the electromagnetic field in the SM.<br>10 Select180$^{o}$ angle (west pole) state for the electromagnetic field in the SM.<br>11 Select270$^{o}$ angle (south pole) state for the electromagnetic field in the SM. |
| 12 | RCIR - Blanking Polarity for coil recirculation. Refer to Section 42.4, “Functional Description,” for details of the recirculation mode.<br>1 Coil recirculation via low side transistors (VSSM, analog GND).<br>0 Coil recirculation via high side transistors (VDDM, analog supply voltage). |
| 11 | ITGDIR - Direction (polarity) of integration. Refer to Section 42.4.1.4.3, “DC Offset Cancellation” for details |
| 10 | BLNDCL - Drive Coil during Blanking.<br>1 During the BIS blanking phase the other coil is actually driven by the SSD block (genuine use case).<br>0 During the BIS blanking phase the other coils is not driven by the SSD. The SM will not move during blanking. |
| 9 | ITGDCL - Drive Coil during Integration and outside of any BIS.<br>1 During the BIS integration phase and outside of any BIS the other coil is actually driven by the SSD block (genuine use case). Outside of any BIS the same coil is driven.<br>0 During the BIS integration phase the other coils is not driven by the SSD. Outside of any BIS no coil is driven. The SM will not move during integration (not useful for SSD). |
| 8 | RTZE - Return to Zero Enable. This is in fact the enable bit of the SSD logic to take over control of the SM coils[1].<br>1 Control of the SM coils by the SSD block is enabled.<br>0 Control of the SM coils by the SSD block is disabled. |
| 6 | BLNST - Blanking Status. Refer to Section 42.1.3.2, “Normal Mode” for details.<br>1 The SSD block is currently in the blanking phase of an ongoing BIS.<br>0 The SSD block is not in the blanking phase of an ongoing BIS. |
| 5 | ITGST - Integration Status. Refer to Section 42.1.3.2, “Normal Mode” for details.<br>1 The SSD block is currently in the integration phase of an ongoing BIS.<br>0 The SSD block is not in the integration phase of an ongoing BIS. |

**Table 42-4. CONTROL Register Field Description (continued)**

| Field | Description |
|---|---|
| 1 | SDCPU - σδ-modulator Power Up. Setting this bit enables the analog block of the SSD and enables the clocking of the port control logic of the digital part.<br>1 Analog block of the SSD is enabled.<br>0 Analog block of the SSD is not enabled. |
| 0 | D_RSVD - Reserved bit.".<br>This bit is writable but should be kept as value 0. |

[1] The application must switch off any other blocks possibly interfering with port control of the SSD block.

### 42.3.3.2 Interrupt Enable and Flag Register (IRQ)

Figure 42-4 below describes the fields of the interrupt enable and flag (IRQ) register:

Offset 0x02 Access: User read/write

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLNIF | ITGIF | 0 | 0 | 0 | 0 | 0 | ACOVIF | BLNIE | ITGIE | 0 | 0 | 0 | 0 | 0 | ACOVIE |
| W | w1c | w1c | | | | | | w1c | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 42-4. SSD Interrupt Flag and Enable Register (IRQ)**

The function of the IRQ register bits is shown in Table 42-5.

**Table 42-5. IRQ Register Field Description**

| Field | Description |
|---|---|
| 15 | BLNIF - Blanking expired Interrupt Flag.<br>1 This flag is set when the BIS blanking phase has expired.<br>0 No such event. |
| 14 | ITGIF - Integration expired Interrupt Flag.<br>1 This flag is set when the BIS integration phase has expired.<br>0 No such event. |
| 8 | ACOVIF - Accumulator Overflow Interrupt Flag.<br>1 This flag is set when during the BIS integration phase the integration logic attempted either to increment the ITGACC register above 0x7FFF or to decrement it below 0x8000.<br>0 No such event. |
| 7 | BLNIE - Blanking expired Interrupt Enable.<br>1 A module interrupt will occur if the BLNIF bit is set.<br>0 The BLNIF flag will not trigger an interrupt on the ips_int output. |

**Table 42-5. IRQ Register Field Description (continued)**

| Field | Description |
|---|---|
| 6 | ITGIE - Integration expired Interrupt Enable.<br>1 A module interrupt will occur if the ITGIF bit is set.<br>0 The ITGIF flag will not trigger an interrupt on the ips_int output. |
| 0 | ACOVIE - Accumulator Interrupt Enable.<br>1 A module interrupt will occur if the ACOVIF bit is set.<br>0 The ACOVIF flag will not trigger an interrupt on the ips_int output. |

### 42.3.3.3 Integration Accumulator Register (ITGACC)

Figure 42-5 below describes the fields of the integration accumulator (ITGACC) register:

Offset 0x04 Access: User read/write

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

**Figure 42-5. SSD Integration Accumulator Register (ITGACC)**

The function of the ITGACC register bits is shown in Table 42-6.

**Table 42-6. ITGACC Register Field Description**

| Field | Description |
|---|---|
| 15-0 | ITGACC - Integration Accumulator readout value. This 2's complement register represents the accumulator register of the back EMF integrator of the SSD block.<br>Refer to the Functional Description of the integrator for further details. |

### 42.3.3.4 Down Counter Register (DCNT)

Figure 42-6 below describes the fields of the down counter (DCNT) register:

Offset 0x06 Access: User read/write

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

**Figure 42-6. SSD Down Counter Register (DCNT)**

The function of the DCNT register bits is shown in Table 42-7.

**Table 42-7. DCNT Register Field Description**

| Field | Description |
|---|---|
| 15-0 | DCNT - Down Counter value. This register represents the actual value of the down counter in unsigned format. Refer to the Functional Description of the integrator for further details. |

### 42.3.3.5 Blanking Counter Load Register (BLNCNTLD)

Figure 42-7 below describes the fields of the blanking counter load (BLNCNTLD) register:

Offset 0x08 Access: User read/write

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BLNCNTLD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 42-7. SSD Blanking Counter Load Register (BLNCNTLD)**

The function of the BLNCNTLD register bits is shown in Table 42-8.

**Table 42-8. BLNCNTLD Register Field Description**

| Field | Description |
|---|---|
| 15-0 | BLNCNTLD - Blanking Count Load value. This register is programmed with the number of down counter periods belonging to the blanking phase of the following BISs. Number format is unsigned. Refer to the Functional Description of the integrator for further details.<br>Programming all 0's into the BLNCNTLD register bits disables blanking completely. |

### 42.3.3.6 Integration Counter Load Register (ITGCNTLD)

Figure 42-8 below describes the fields of the integration counter load (ITGCNTLD) register:

Offset 0x0A Access: User read/write

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ITGCNTLD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 42-8. SSD Integration Counter Load Register (ITGCNTLD)**

The function of the ITGCNTLD register bits is shown in Table 42-9.

**Table 42-9. ITGCNTLD Register Field Description**

| Field | Description |
|---|---|
| 15-0 | ITGCNTLD - Integration Count Load value. This register is programmed with the number of down counter periods belonging to the integration phase of the following BISs. Number format is unsigned. Refer to the Functional Description of the integrator for further details.<br>Programming all 0's into the ITGCNTLD register bits disables integration completely. |

### 42.3.3.7 SSD Prescale and Divider Register (PRESCALE)

Figure 42-9 below describes the fields of the prescale and divider factor (PRESCALE) register:

Offset 0x0C — Access: User read/write

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | BLNDIV | | | 0 | ITGDIV | | | 0 | 0 | OFFCNC | | 0 | ACDIV | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 42-9. SSD Prescale and Divider Factor Register (PRESCALE)**

The function of the PRESCALE register bits is shown in Table 42-10 below:

**Table 42-10. PRESCALE Register Field Description**

| Field | Description |
|---|---|
| 14-12 | BLNDIV - Blanking Counter Clock Divider Select. The frequency for updating the down counter in the blanking phase of the next BISs is derived from the bus clock according to the formula<br><down counter clock> = <bus clock> / ($8 * 2^{BLNDIV}$)<br>According to this formula the divider factors are:<br>000 8<br>001 16<br>010 32<br>011 64<br>100 128<br>101 256<br>110 512<br>111 1024 |
| 10-8 | ITGDIV - Integration Counter Clock Divider Select. The frequency for updating the down counter in the integration phase of the next BISs is derived from the bus clock according to the formula<br><down counter clock> = <bus clock> / ($8 * 2^{ITGDIV}$)<br>According to this formula the divider factors are:<br>000 8<br>001 16<br>010 32<br>011 64<br>100 128<br>101 256<br>110 512<br>111 1024 |

**Table 42-10. PRESCALE Register Field Description (continued)**

| Field | Description |
|---|---|
| 5-4 | OFFCNC - Offset Cancellation polarity flip select. Refer to Section 42.4.1.4.3, "DC Offset Cancellation" for details of the offset cancellation mechanism. The OFFCNC bits set the preset value of the internal counter which determines the polarity flips during the integration phase.<br>The preset value is derived from the ITGCNTLD register value with the following divider factor:<br>00 0: Selected polarity remains unchanged for all the time of the integration phase.<br>01 2: 1st polarity switch (and possibly a succeeding one) occurs after [ITGCNTLD div 2] DCNT ticks.<br>10 4: 1st polarity switch and succeeding ones occur after [ITGCNTLD div 4] DCNT ticks.<br>11 8: 1st polarity switch and succeeding ones occur after [ITGCNTLD div 8] DCNT ticks.<br>If the ITGCNTLD register value cannot be divided by the required factor an additional polarity flip occurs with a duration corresponding to the bits shifted out. |
| 2-0 | ACDIV - Accumulator Sample Clock Divider Select. The accumulator sample clock is derived from the bus clock according to the formula<br><accumulator sample clock> = <bus clock> / ($8 * 2^{ACDIV}$)<br>According to this formula the divider factors are:<br>000 8<br>001 16<br>010 32<br>011 64<br>100 128<br>101 256<br>110 512<br>111 1024<br>The first ITGACC register update occurs when ($8 * 2^{ACDIV}$) bus clocks have expired after the ITGST bit has been set by the SSD block. |

## 42.4 Functional Description

For all the descriptions given here it is assumed that the SSD block has gained exclusive control of the SM coils and the analog block is enabled appropriately.

### 42.4.1 Main Building Blocks of the SSD

The functional description given in this chapter deals with the main functional blocks. It concentrates on the description of the implemented functionality. Refer to Figure 42-1 for details.

#### 42.4.1.1 Analog Block

An overview of the analog block of the SSD block is given in Figure 42-10 below. Additionally the most important sub blocks of the digital part which are connected to the analog blocks are shown in order to clarify the joint operation of the analog block and the digital part.

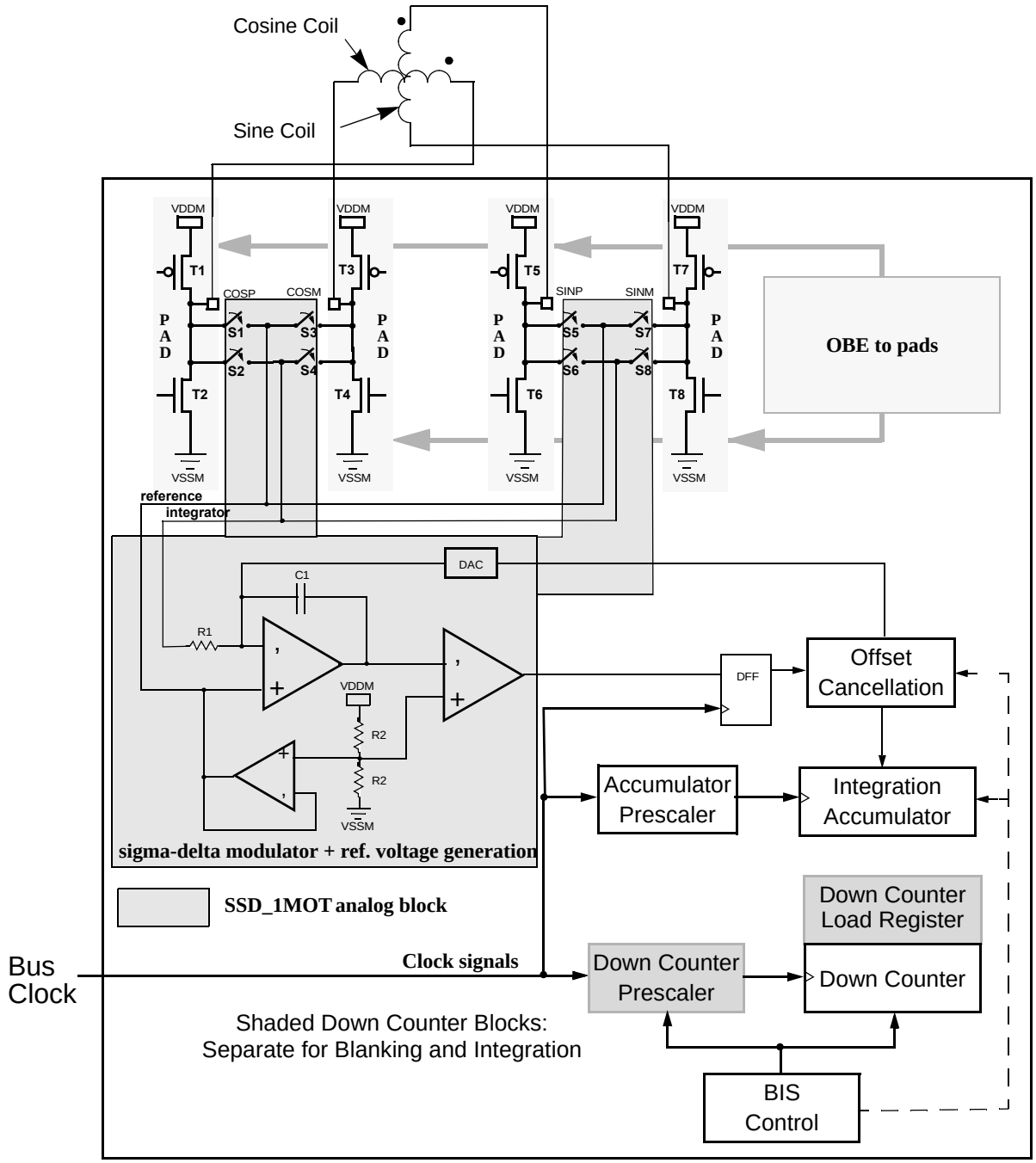



**Figure 42-10. SSD Block Diagram, Analog Block**

Main part of the analog block is the σδ-modulator. It is operational during the BIS integration phase only (step 5 in Section 42.4.2.2, “Details of the SSD Measurement”). The clock to update the feedback path is derived from the bus clock using the ACDIV setting. The 1-bit output value provided to the digital part is used to increment or decrement the ITGACC register.

For the correct movement of the SM the sine and cosine coil connections to VDDM and VDDS need to be set properly, depending from the angular position. This is achieved by enabling and disabling of the pad transistors T1 to T8. These pad transistors are not part of the SSD block, only the OBE (output buffer enable) signals for the pads are provided. For the switch characteristics refer to Section 42.4.1.2.1, “Transistor Condition States”.

Aside from the pad transistors the switches S1 to S8 determine which coil provides the back EMF to the integrator in which polarity w.r.t. the reference voltage. The switches are implemented in the analog block of the SSD (denoted by shading them in the same manner like the analog block).

The σδ-modulator is enabled by setting the SDCPU bit in the CONTROL register. To compensate for switching effects of the analog circuitry the user must take into account sufficient startup time, described in Section 42.5.1, “Analog Block Startup Time” prior to starting the BIS.

### 42.4.1.2 Analog Wrapper + Port Control

This sub block controls the outputs to the coils and the inputs to the analog block. Refer to Figure 42-10 for details.

The most relevant bits of the CONTROL register belonging to that functional block are:

- STEP bits: These 2 bit vector has two functions.

  One function is to determine which coil is driven and which coil is connected to the σδ-modulator. Additionally the direction of the current flow is selected with these bits. For clockwise direction of the SM movement the value must be decremented and for counter-clockwise movement it must be incremented when advancing from one STEP setting to the next.
- BLNDCL: This bit is the enable of the supply voltage to be routed to the coil driven in the appropriate STEP setting during the blanking phase of an ongoing BIS.
- ITGDCL: This bit is the enable of the supply voltage to be routed to the coil driven in the appropriate STEP setting during the integration phase of an ongoing BIS. Additionally it determines the coil drive setting outside of any BIS.
- ITGDIR: This bit is relevant only in the integration phase. Together with the STEP bits the polarity of the integration is determined by enabling or disabling the appropriate analog switches. Refer to Section 42.4.1.4.3, “DC Offset Cancellation”.

These control bits are translated into the appropriate switching scheme of the pad transistors and the σδ-modulator switches described below. Note that it must be ensured at the device level that the SSD block has exclusive control over the analog pads connected to the SM coils.

Additionally it is a precondition that the RTZE bit in the CONTROL register is set.

#### 42.4.1.2.1 Transistor Condition States

The pad transistors T1 to T8 are responsible for connecting the SM coils to the analog supply voltages VDDM and VDDS. Table 42-11 below shows the pad transistor condition states implemented in the analog block. In the table the columns have the following meaning:

- STEP denotes the setting of the corresponding bits in the CONTROL register.
- ITGST reflects the status indicator of the BIS integration phase.

- ‘Resulting DCOIL’ is the (BIS state dependent) resulting coil drive setting. See Table 42-12 for how this setting is derived from the value of the CONTROL register.
- RCIR denotes the setting of the corresponding bit in the CONTROL register.

**Table 42-11. Transistor Condition States[1]**

| STEP | ITGST | Resulting DCOIL | RCIR | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| xx | 1 | 0 | x | OFF | OFF | OFF | OFF | OFF | OFF | OFF | OFF | Integration with no drive |
| 00 | 0 | 0 | 0 | OFF | OFF | OFF | OFF | ON | OFF | ON | OFF | Blanking withno drive (RCIR bit determines supply voltage for recirculation) |
| 00 | 0 | 0 | 1 | OFF | OFF | OFF | OFF | OFF | ON | OFF | ON | |
| 01 | 0 | 0 | 0 | ON | OFF | ON | OFF | OFF | OFF | OFF | OFF | |
| 01 | 0 | 0 | 1 | OFF | ON | OFF | ON | OFF | OFF | OFF | OFF | |
| 10 | 0 | 0 | 0 | OFF | OFF | OFF | OFF | ON | OFF | ON | OFF | |
| 10 | 0 | 0 | 1 | OFF | OFF | OFF | OFF | OFF | ON | OFF | ON | |
| 11 | 0 | 0 | 0 | ON | OFF | ON | OFF | OFF | OFF | OFF | OFF | |
| 11 | 0 | 0 | 1 | OFF | ON | OFF | ON | OFF | OFF | OFF | OFF | |
| 00 | 0 | 1 | 0 | ON | OFF | OFF | ON | ON | OFF | ON | OFF | Cosine coil driven P -> M |
| 00 | 0 | 1 | 1 | ON | OFF | OFF | ON | OFF | ON | OFF | ON | |
| 00 | 1 | 1 | x | ON | OFF | OFF | ON | OFF | OFF | OFF | OFF | |
| 01 | 0 | 1 | 0 | ON | OFF | ON | OFF | ON | OFF | OFF | ON | Sine coil driven P -> M |
| 01 | 0 | 1 | 1 | OFF | ON | OFF | ON | ON | OFF | OFF | ON | |
| 01 | 1 | 1 | x | OFF | OFF | OFF | OFF | ON | OFF | OFF | ON | |
| 10 | 0 | 1 | 0 | OFF | ON | ON | OFF | ON | OFF | ON | OFF | Cosine coil driven M -> P |
| 10 | 0 | 1 | 1 | OFF | ON | ON | OFF | OFF | ON | OFF | ON | |
| 10 | 1 | 1 | x | OFF | ON | ON | OFF | OFF | OFF | OFF | OFF | |
| 11 | 0 | 1 | 0 | ON | OFF | ON | OFF | OFF | ON | ON | OFF | Sine coil driven M -> P |
| 11 | 0 | 1 | 1 | OFF | ON | OFF | ON | OFF | ON | ON | OFF | |
| 11 | 1 | 1 | x | OFF | OFF | OFF | OFF | OFF | ON | ON | OFF | |

1 ‘x’ means ‘Don’t care’

Table 42-12 below shows the logic dependency of the resulting coil drive from the internal state of the SSD block and from the setting of the different coil control bits in the CONTROL register:

Table 42-12. Generation of the ‘Resulting DCOIL’

| BLNST | BLNDCL | ITGST | ITGDCL | Resulting DCOIL | Remarks |
|---|---|---|---|---|---|
| 0 | x | 0 | 0 | 0 | No running BIS<br>ITGDCL determines result |
| | | | 1 | 1 | |
| 1 | 0 | 0 | x | 0 | Running BIS in blanking phase<br>BLNDCL determines result |
| | 1 | | | 1 | |
| 0 | x | 1 | 0 | 0 | Running BIS in integration phase<br>ITGDCL determines result |
| | | | 1 | 1 | |

#### 42.4.1.2.2 Switch Condition States

The analog switches S1 to S8 are used to select the appropriate source and polarity of the non-driven coil into the σδ-modulator for integration during the integration phase of an ongoing BIS. Outside of the integration phase none of the switches is enabled. Refer to Table 42-13 below for details.

Table 42-13. Switch Condition States

| ITGST | STEP (Register Bit) | Integration Polarity (Input to analog block) | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | xx | x | Open | Open | Open | Open | Open | Open | Open | Open |
| 1 | 00 | 0 | Open | Open | Open | Open | Close | Open | Open | Close |
| 1 | 00 | 1 | Open | Open | Open | Open | Open | Close | Close | Open |
| 1 | 01 | 0 | Open | Close | Close | Open | Open | Open | Open | Open |
| 1 | 01 | 1 | Close | Open | Open | Close | Open | Open | Open | Open |
| 1 | 10 | 0 | Open | Open | Open | Open | Open | Close | Close | Open |
| 1 | 10 | 1 | Open | Open | Open | Open | Close | Open | Open | Close |
| 1 | 11 | 0 | Close | Open | Open | Close | Open | Open | Open | Open |
| 1 | 11 | 1 | Open | Close | Close | Open | Open | Open | Open | Open |

**NOTE**

The value given in the column ‘Integration Polarity’ in Table 42-13 is linked to the ITGDIR bit in the CONTROL register in the way described in Section 42.4.1.4.3, “DC Offset Cancellation” below.

### 42.4.1.3 Register Interface

The register interface processes the IPS accesses from the device level. Access size is 32 bits, the SSD block supports 16- and 32-bit accesses.

Any write access on byte-level is ignored.

All reserved registers provide 0x0000 on read. Write access is not allowed.

#### 42.4.1.4 BIS Control

Once triggered the sequence control logic walks through a single individual BIS. In the normal application one BIS corresponds to a single step (90° movement of the SM). In detail the BIS is implemented in the SSD block in the following way:

- Each BIS starts with setting the TRIG bit. Ending a running BIS manually is only possibly by clearing the RTZE bit.
- If the BLNCNTLD register is set to a value other than 0x0000 the DCNT is loaded with (BLNCNTLD - 1) and is started using the BLNDIV bit setting for the clock divider. The BLNST is set.

  The blanking phase of the BIS is executed, the BLNDCL bit is used to determine whether one of the coils is driven during the blanking phase.

  If the appropriate number of down counter periods (equal to the BLNCNTLD register value) expires the BLNIF is set, the interrupt is triggered according to the BLNIE bit and the BLNST bit is cleared.
- If the ITGCNTLD register is set to a value other than 0x0000 the DCNT is loaded with (ITGCNTLD - 1) and is started using the ITGDIV bit setting for the clock divider. The ITGST is set and the ITGACC register is initialized with 0x0000.

  The integration phase of the BIS is executed, the ITGDCL bit is used to determine whether one of the coils is driven during the integration phase. The σδ-modulator of the analog block is functional and the ITGACC register is updated. During the integration phase the polarity is switched according to the OFFCNC bits.

  If the appropriate number of down counter periods (equal to the ITGCNTLD register value) expires the ITGIF is set, the interrupt is triggered according to the ITGIE bit and the ITGST bit is cleared.

The state of the ongoing BIS can be monitored by the following status bits:

- The BLNST bit is set during the blanking phase exclusively.
- The ITGST bit is set during the integration phase exclusively. When it is set, the BIS control enables the σδ-modulator in the analog block together with the integration circuitry. As long as the integration phase is active the ITGACC register content is modified depending from the output of the σδ-modulator.

In the normal use case the end of the BIS is the end of the integration phase. Independent from the time required by the software to detect and act upon the end of the BIS the ITGACC register is not changed after the end of the integration phase.

The sequence control logic itself makes use of the following sub blocks:

##### 42.4.1.4.1 Down Counter

The down counter is basically a timer with the divider factor (BLNDIV or ITGDIV) and length (BLNCNTLD or ITGCNTLD) determined by the current state of the BIS. Additionally to defining the

length of the integration phase the ITGCNTLD setting determines the DCNT value to switch the integration polarity for DC Offset Cancellation depending from the OFFCNC bit setting.

When the down counter reaches 0x0000 and the associated divider period has expired the appropriate flag is set and the corresponding interrupt is triggered depending from the interrupt enable bit. Note that polling the DCNT register for 0x0000 is misleading since the DCNT time out is reached at the **end** of the divider period belonging to the value of 0x0000. Refer to Section 42.6.3, “Watching Internal States of the SSD”.

Table 42-14 below shows details how the BLNDIV/ITGDIV and BLNCNTLD/ITGCNTLD settings determine the length and granularity of the blanking and integration phase depending from the bus clock frequency.

**Table 42-14. Blanking and Integration Phase Length Vs. Bus Clock[1]**

| Bus Clock | 40 MHz | | 64 MHz | | 80 MHz | |
|---|---|---|---|---|---|---|
| **BLNDIV/ITGDIV** | **0** | **7** | **0** | **7** | **0** | **7** |
| Timing Granularity | 0.2 µs | 25.6 µs | 0.125 µs | 16 µs | 0.1 µs | 12.8 µs |
| Max. length of BLN/ITG | 13.107 ms | 1.678 s | 8.192 ms | 1.049 s | 6.554 ms | 0.839 s |

1 Numbers rounded appropriately

#### 42.4.1.4.2 Integration Accumulator

This is the fundamental sub block of the SSD, it is responsible for collecting the result of the back EMF integration from the σδ-modulator located in the analog block. The only time when the value of the accumulator can change is during the integration phase of a BIS.

In terms of signal processing the ITGACC register is the counterpart of the σδ-modulator in the analog block, working as the σδ-demodulator: Depending from the ACDIV bits in the PRESCALE register the output of the analog block is sampled periodically and the content of the accumulator incremented or decremented. Therefore the ITGACC register in fact counts the ‘imbalance’ between 1 and 0 output samples from the analog block.

The value of the ITGACC register can change only during the integration phase of an ongoing BIS. Before the first update the content is initialized to 0x0000 and starting from that it is incremented or decremented according to the σδ-modulator output.

Number format is two’s complement, if an overflow (attempt to increment ITGACC register value of 0x7FFF) or an underflow (attempt to decrement ITGACC register value of 0x8000) the ACOVIF bit indicates the over-/underflow condition, the SSD interrupt is triggered if the ACOVIE bit is set and the ITGACC register values is not changed. For the rest of the integration phase of the current BIS the ITGACC register value does not change. Reaching the accumulator end values **without** an over-/underflow condition does not prevent the ITGACC register from incrementing 0x8000 (-32768) or decrementing 0x7FFF (+32767).

Table 42-15 below shows details how the ACDIV setting determines the σδ-demodulator sampling clock w.r.t. the bus clock. The recommended setting for the sampling is a resulting clock between 500 kHz and 2 MHz. Therefore the ACDIV values for sampling clock values in this recommended range are given:

**Table 42-15. ITGACC Update Clock Vs. Bus Clock - Recommended Settings[1]**

| Bus Clock | 40 MHz | | 64 MHz | | 80 MHz | |
|---|---|---|---|---|---|---|
| ACDIV | 3'b010 | 3'b011 | 3'b011 | 3'b100 | 3'b011 | 3'b100 |
| Divider factor | 32 | 64 | 64 | 128 | 64 | 128 |
| Sampling frequency | 1.25 MHz | 625 kHz | 1.00 MHz | 500 kHz | 1.25 MHz | 625 kHz |
| Update interval | 0.8 µs | 1.6 µs | 1.00 µs | 2.00 µs | 0.8 µs | 1.6 µs |

1 Numbers rounded appropriately

#### 42.4.1.4.3 DC Offset Cancellation

Due to deviations from the mid point of the analog supply voltages and other effects in the hardware of the analog blocks a DC offset may be introduced into the output of the σδ-modulator. As a consequence of such a DC offset the value obtained in the integration accumulator would depend from the 'direction' of the integration (e.g. accumulator increment for positive back EMF in clockwise movement). The DC offset cancellation implemented in the SSD block can eliminate (or at least reduce) the influence of such a DC offset:

When active the DC offset cancellation reverts two internal settings in the SSD block during the integration phase of the current BIS:

- The input into the analog block controlling the integration polarity which sets the switch condition state (third column in Table 42-13):

  Initial value (when the integration starts at step 5 in Section 42.4.2.2, "Details of the SSD Measurement") is the bit value given in the ITGDIR register in the CONTROL register.
- The output of the σδ-modulator being applied to the integration accumulator (ITGACC register):

  Initially it is applied without change to the integration accumulator.

As a result the switch conditions in the analog circuitry change the direction of the voltage representing the back EMF measured by the σδ-modulator. But the change direction of the ITGACC register is maintained because the interpretation of the σδ-modulator output is reverted, too.

The offset cancellation is implemented as an additional counter running during the integration phase with the same clock setting like the DCNT register. The preset value for this counter is derived from the ITGCNTLD register by shifting right by 0, 1, 2 or 3 bits, depending from the OFFCNC bit setting. Clearing all the OFFCNC bits obviously disables the offset cancellation completely. Note that increasing the number of flips improves the offset cancellation because the different polarities are distributed equally over the complete integration phase (if ITGCNTLD can be divided by the appropriate number). Refer to Figure 42-11 for more details.

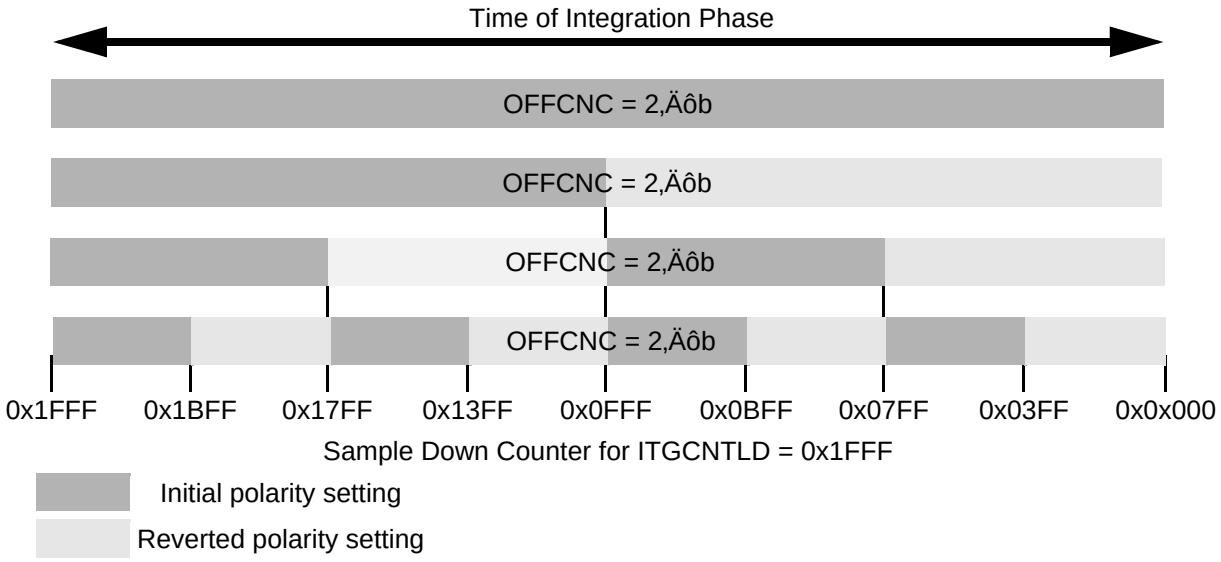



**Figure 42-11. Offset Cancellation Polarity Distribution**

If the shift process shifts out LS bits from the ITGCNTLD register (non-integer divide) the number shifted out creates an additional polarity flip which lasts the appropriate number of DCNT update periods.

## 42.4.2 Stepper Stall Detection Measurement

This part of the functional description deals with the main intended use case of the SSD block, this is the detection of the scale end boundaries of the gauge pointer moved by the SM which, in turn, is driven by the SSD block. For details of the related sub blocks refer to Section 42.4.1, “Main Building Blocks of the SSD”.

### 42.4.2.1 Overview of the SSD Measurement

The generic flow of SSD measurement is given in Figure 42-12 below, the numbers denoted at each step belong to the detailed explanations given in Section 42.4.2.2, “Details of the SSD Measurement”.

The two phases of the BIS are executed in sequence:

1. Blanking phase: Since the non-driven coil used for measurement was driven in the previous step switching transients are induced when changing from the driven into the non-driven state. Therefore both pins of the non-driven coil are connected to one of the analog supply voltages VDDM or VSSM (depending from the RCIR bit) to allow recirculation of these transient currents.
2. Integration phase: This is the actual measurement where the ITGACC register is changed according to the results of the σδ-modulator of the analog block.

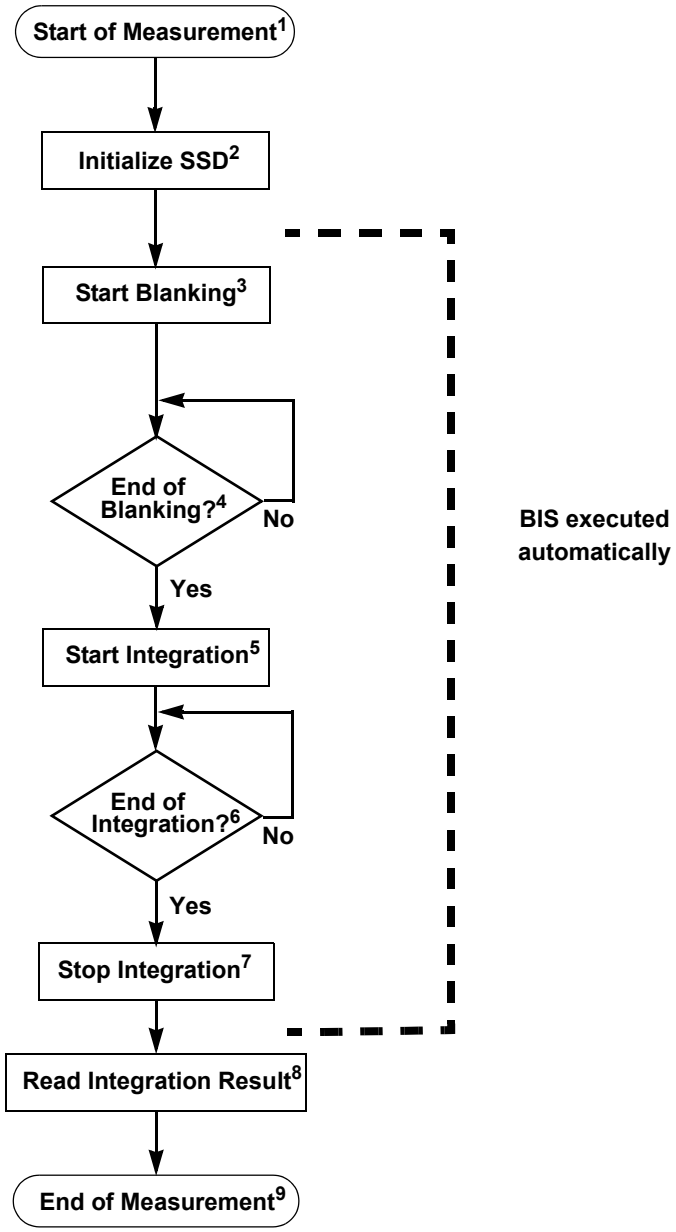



**Figure 42-12. Generic SSD Flow**

### 42.4.2.2 Details of the SSD Measurement

This section describes in detail the steps introduced in Figure 42-12. Note that it describes only the measurement process. The decision whether the stall position has been reached or another SM step is required must be made by the controlling CPU depending from the measurement result. All control bits are assumed to have their (inactive) reset values prior to entering the SSD measurement flow. The following paragraphs relate to one complete BIS including the (optional) blanking phase followed by the integration phase:

1. Start of Measurement

For proper usage of the SSD block it must have exclusive control over the coils belonging to the SM whose stall position must be detected. This must be ensured at the device level. Following this the SSD must be enabled by setting the RTZE bit in the CONTROL register, this bit must be left asserted for the complete SSD flow.

It is at this point in time that application-dependent control settings are set which remain constant over the complete SSD flow. These settings consist of the integrate direction of the ITGACC register, which is set by the ITGDIR bit and the direction to advance the STEP setting (increment or decrement influence clockwise or counter-clockwise movement of the SM). Additionally the PRESCALE and the IRQ register must be programmed (it is not recommended to change the content of the PRESCALE register during a running BIS).

2. Initialize SSD

   At this point the STEP bits are set according to the angular position of the SM for the current position. After programming the STEP bits the analog block can be enabled by setting the SDCPU bit.

3. Start Blanking

   This step starts with setting the TRIG bit together with the STEP bits initializing the complete BIS for the next step. Depending from the direction of the rotation the previous step setting is either decremented or incremented, wrapping from 2'b11 to 2'b00 or vice versa. If the BLNDCL bit is set this step marks the start of the SM movement, During blanking both pins of the non-driven coil are connected either to VDDM or VSSM for recirculation, depending from the RCIR bit. The bus clock is divided accordingly to the BLNDIV bits to decrement the DCNT. The BLNST bit is set to allow the user to monitor the status.

4. End of Blanking?

   The end of the blanking phase is automatically detected. If the DCNT reaches 0x0000 and the complete blanking time is expired the BLNIF flag is set and the interrupt triggered according to the BLNIE bit. The BLNST bit is cleared.

5. Start Integration

   After the end of the blanking phase the SSD block continues automatically with the integration phase:

   The ITGCNTLD register is used to initialize the DCNT and is decremented according to the ITGDIV bits setting. The driving coils is powered according to the ITGDCL bit. During the integration phase the polarity flip for offset cancellation is triggered according to the OFFCNC bits. The ITGST bit is set to allow the user to monitor the status.

6. End of Integration?

   The down counter is monitored in the same way like in step 4. The ITGIF flag is set and the interrupt is triggered according to the ITGIE bit. The ITGST bit is cleared.

7. Stop Integration

   On the expiration of the current BIS the integration is stopped, the σδ-modulator is disabled, the ITGACC register is frozen. Note that the current to the coil driven by the SSD block continues according to the ITGDCL and the STEP setting.

8. Read Integration Result

Now the result of the back EMF integration over the time set with the BLNCNTLD register value can be read from the ITGACC register.

9. End of Measurement

   Depending from the result of the measurement the SSD block now can be disabled by clearing the RTZE bit or another measurement can be started. Any additional measurement should start from step 2.

### 42.4.3 Additional Modes of Operation

There are several additional modes how the SSD block can operate with the SM coils. They are aside from the main use case.

#### 42.4.3.1 Blanking with No Drive

During blanking of one coil (refer to step 3 in Section 42.4.2, “Stepper Stall Detection Measurement”) the user can disable the drive of the other coil by not setting the BLNDCL bit. Since the SM has moved in the integration phase of the previous BIS this means that the SM movement is interrupted during blanking. This mode is not useful for SSD.

#### 42.4.3.2 Integration with No Drive

If the ITGDCL bit is switched off the driving coil of the SM is not powered during the integration phase, the SM will not move. Only the σδ-modulator output is integrated with the reference voltage setting belonging to that step. This mode makes no sense for stall detection, but it can be used for certain measurements of the analog sub blocks.

## 42.5 Initialization Information

### 42.5.1 Analog Block Startup Time

No specific initialization after a hard reset or after leaving one of the power down modes is necessary, but the user should allow a sufficient startup time of the analog block after hard reset or enabling of the SSD block to compensate switching transients. The startup time specified for the analog block is 20 µs.

### 42.5.2 Analog Block Polarity Switching Time

If the offset cancellation is used the polarity of the back EMF measurement in the analog block is reverted during the integration phase. Note that the time for the σδ-modulator to achieve a stable output depends strongly from the external circuitry, e.g. coil inductance, parasitic capacitance of the leads.

As an initial estimation for typical applications a time of maximum 10 µs should be taken into account by the user.

### 42.5.3 SSD Startup

The SSD block takes care of smooth transitions between the different steps to avoid current glitches when the coil driven by the block (sine or cosine) changes. Single-cycle glitches in the coil drive current at startup of the SSD flow can be avoided in the following way:

- Program the STEP bits reflecting the current position of the SM prior to the SSD flow with the SDCPU bit still cleared.
- After programming the STEP bits enable the analog block by setting the SDCPU bit.

## 42.6 Application Information

This is additional information intended for use by the customer.

### 42.6.1 Current Flow Examples

Figure 42-13 below shows the current flow for a complete sequence of the STEP bits for counter-clockwise movement including the recirculation time:

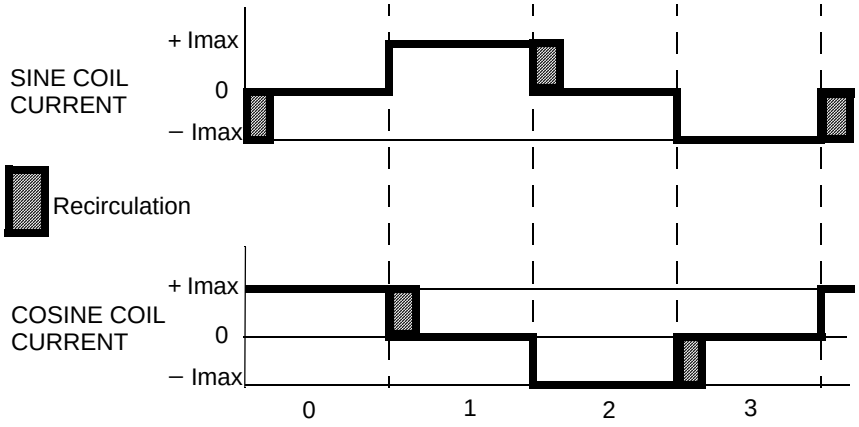



**Figure 42-13. Full Step Sequence for counter-clockwise movement**

Different Examples of the current flow in the SM coils for different settings of the control bits are given in Figure 42-14 to Figure 42-17 below:

Figure 42-14 below shows the recirculation at the beginning of STEP = 0. The cosine coil is driven in the direction P -> M and the sine coil is recirculated against VDDM.

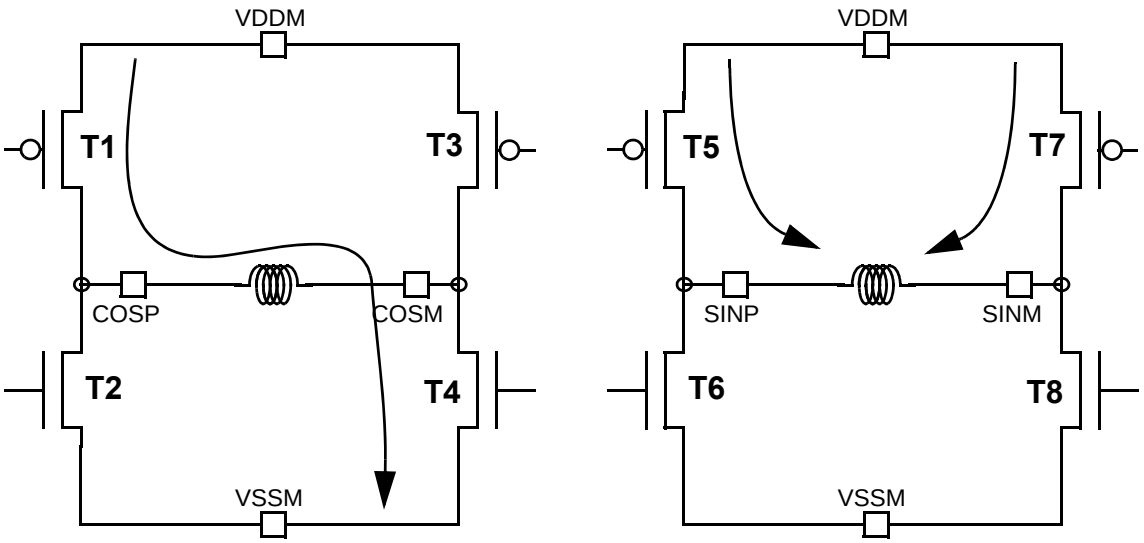



**Figure 42-14. Current flow for Blanking (STEP = 0, BLNDCL = 1, RCIR = 0)**

In Figure 42-15 below the recirculation at the beginning of the next BIS for STEP = 1 with changed setting of the RCIR bit is shown. The sine coil is driven again in P -> M direction and the cosine coil is recirculated against VSSM.

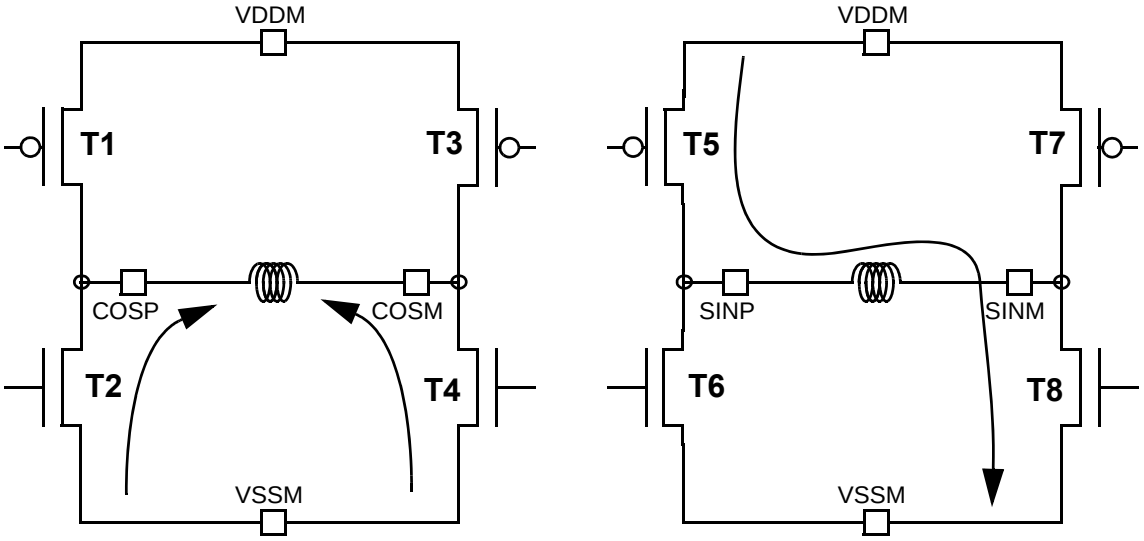



**Figure 42-15. Current flow for Blanking (STEP = 1, BLNDCL = 1, RCIR = 1)**

In Figure 42-16 below it is shown that for the next step (STEP = 2) the cosine coil is driven in reverse direction with respect to STEP = 0 (M -> P direction). Because it is the integration phase of the BIS, the sine coil is isolated from the analog supply voltages. Instead, it is connected to the σδ-modulator (not shown).

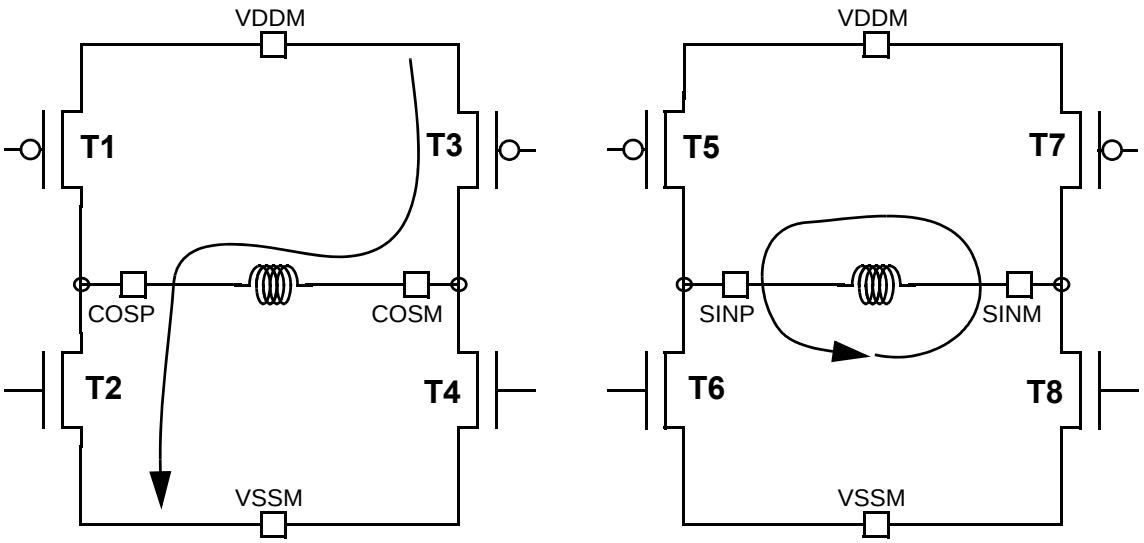



**Figure 42-16. Current flow for Integration (STEP = 2, ITGDCL = 1)**

Figure 42-17 below shows that the sine coil is driven for STEP = 3 in reverse direction with respect to STEP = 1 (M -> P direction). Again the other coil (cosine) is isolated from the analog supply voltages because it is the integration phase of the current BIS.

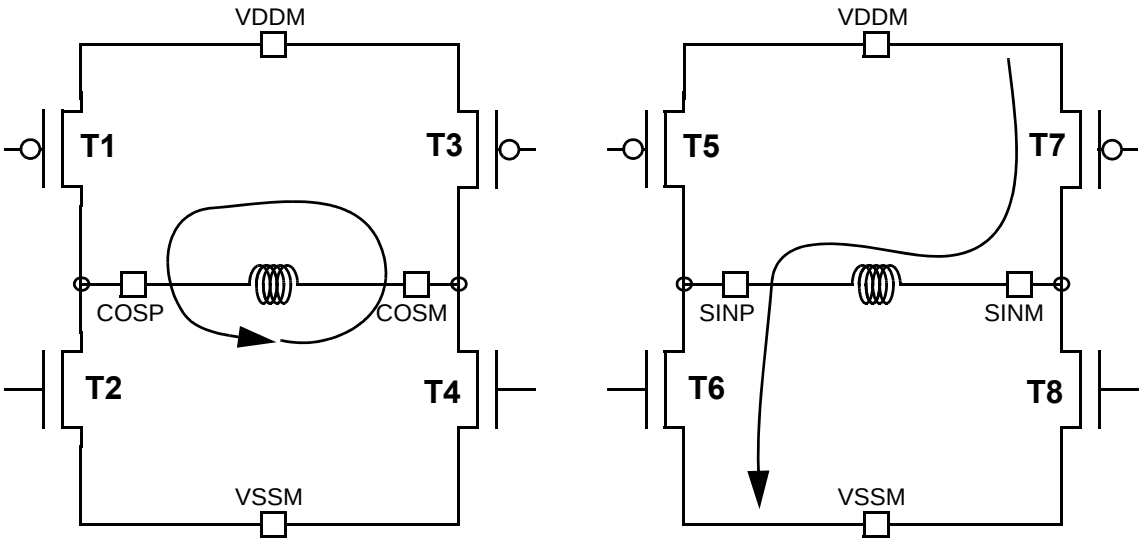



**Figure 42-17. Current flow for Integration (STEP = 3, ITGDCL = 1)**

## 42.6.2 Setting of the PRESCALE Register

### 42.6.2.1 Timing Resolution Considerations

Set the ACDIV bits to the lowest division factor possible, resulting in the highest possible clock frequency for the integration accumulator. This will give the most precise result.

Setting the BLNDIV or ITGDIV bits will influence the resolution of the down counter in the corresponding phase of the BIS (fine resolution required for blanking) as well as the available (absolute) time interval that can be covered by the length of the DCNT register (must be long enough to cover almost one SM step movement for integration). Due to the different prescaler settings for blanking and integration no compromise is necessary between fine resolution for blanking and long time for integration when using high bus frequencies.

It is recommended to select the setting with the best timing resolution for the blanking phase for the BLNDIV bit setting (lowest value). Most likely a different value must be chosen for the ITGDIV bit setting, nevertheless the lowest possible value should be chosen, too.

Note that in normal operation it should not occur that the ACOVIF bit in the IRQ register reads out to be set during the integration. If this happens the bit indicates that either an overflow or an underflow occurred in the ITGACC register. The result should be discarded because the setup of the SSD block was wrong for the current SSD attempt.

#### 42.6.2.2 Offset Cancellation Considerations

Note that the polarity switching for offset cancellation like depicted in Figure 42-11 is controlled by the DCNT register update which is updated depending from the ITGDIV settings. All the divider settings are powers of 2, so the distance in time between two DCNT register updates is always an integer multiple or divider of the ITGACC register update, depending which divider factor is greater than the other one.

If the offset cancellation is used the measurement polarity in the analog block is reverted at least once during the integration phase. As a consequence the σδ-modulator needs some time after each polarity flip to achieve a stable output. An estimation for that time is given in Section 42.5.2, “Analog Block Polarity Switching Time”.

If the ITGACC register is updated **before** the σδ-modulator output is stable at least one count is incorrect. Since the next polarity flip takes place in the opposite direction this incorrect count will be compensated for by the following polarity flip. Therefore it may be useful to add a small number of DCNT register updates to the integration phase to have an even number of polarity flips in the offset cancellation.

Another mean to improve accuracy is to adjust the DCNT register updates where the polarity switches occur with respect to the following ITGACC register update to allow enough settling time for the σδ-modulator output.

### 42.6.3 Watching Internal States of the SSD

For some applications it might be required by the application to know the current state of an ongoing BIS. It is recommended to do that by reading the BLNST and the ITGST bits of the CONTROL register. If necessary the DCNT register value may be read to know how far the DCNT register period has expired.

Do **not** poll the DCNT register value for 0x0000 to find out the end of the blanking or integration phase. Aside from generating more CPU load than required this introduces an inaccuracy. Not earlier than the DCNT register divider period belonging to the 0x0000 value has expired the complete blanking or integration phase has been processed.

It should be kept in mind that the recommended way of operation is to enable the appropriate interrupt flag instead of polling the SSD registers.

## 42.6.4 Stepper Motor Transition Considerations

### 42.6.4.1 SSD Phase-In and Phase-Out

Prior to starting the SSD flow the SM gauge is usually moved in the proximity of the expected stall position using another SM driver module. To change the driving source of the SM without visible interruption to the SSD block it is essential for the user to replicate the coil drive setting valid at the end of the SM drive into the SSD CONTROL register. Given that the transfer of the port control from the previous SM driver module to the SSD block can be done sufficiently fast the SM will not move but will start at a know position with the SSD flow.

The same applies to the end of the SSD flow when the SM control is transferred to another module. It is essential for the application in this use case that the coil drive setting at the end of the SSD flow is replicated in this other module.

Basically the seamless hand over of the SM coils to another block can be handled by appropriately programming the STEP bits and the ITGDCL bit in the CONTROL register prior to pass pad control to the SSD block. This ensures that one of the SM coils is driven and the SM retains its position when the SSD block gets pad control.

Vice versa the STEP and ITGDCL setting at the end of the last BIS where the gauge stall was detected allow the user to replicate this specific coil drive setting to the module taking over pad control from the SSD block.

For more details refer to specific application notes describing the usage of the SSD block.

### 42.6.4.2 Changing of SSD Internal States

Depending from the application it may be required to lock the SM into the current angular position to prevent occasional movement prior to the next step. This is especially useful after the integration phase of a BIS. To achieve this the internal logic of the SSD block does the following after the integration phase has expired:

- The coil drive setting of the integration phase is held active: The ITGDCL bit determines the coil drive; the coil and coil direction is determined by the STEP bits.
- The undriven coil is set (back to) recirculation like in the blanking phase.

This leaves both coils in a well-defined state, nevertheless the user should keep this time outside of the BIS as short as possible to avoid visible interruptions of the gauge movement.

When changing to the next step within the SSD flow it should be noted that the update of the STEP bits should coincide (done in the same write to the CONTROL register) like the trigger of the BIS by writing 1 into the TRIG bit. This ensures seamless changeover from one step to the next as well as immediate start of the BIS when the coil driven has changed.

### 42.6.5 Legacy Modes - Separate Blanking and Integration Phase

Despite the automatic BIS it is still possible to use the SSD block in a way similar to the old design. Separate blanking and integration phases can be obtained very easily by setting the down counter preload value for the undesired phase to 0x0000. When the corresponding BIS is executed this phase is simply skipped.

Note that in this case the BLNIF bit will be important for the user because its assertion marks the end of the blanking phase.

To ease the programming of separate BLNCNTLD and ITGCNTLD register values on-the-fly the two adjacent registers are placed into one single 32-bit access. Therefore the user who wants to implement programmed-control switching between blanking and integration needs only one single 32-bit register write (to switch the down counter preload values) prior to the execution of the blanking or integration phase.

# Chapter 43
# System Integration Unit Lite (SIUL)

## 43.1 Introduction

This chapter describes the System Integration Unit Lite (SIUL), which is used for the management of the pads and their configuration. It controls the multiplexing of the alternate functions used on all pads as well as being responsible for the management of the external interrupts to the device.

## 43.2 Overview

The System Integration Unit Lite (SIUL) controls the MCU pad configuration, ports, general-purpose input and output (GPIO) signals and external interrupts with trigger event configuration. Figure 43-1 is a block diagram of the SIUL and its interfaces to other system components.

The module provides the capability to configure, read, and write to the device's general-purpose I/O pads that can be configured as either inputs or outputs.

• When a pad is configured as an input, the state of the pad (logic high or low) is obtained by reading an associated data input (GPDI or PGPDI) register.

• When a pad is configured as an output, the value driven onto the pad is determined by writing to an associated data output (GPDO, PGPDO or MPGDO) register. Enabling the input buffers when a pad is configured as an output allows the actual state of the pad to be read.

• To enable monitoring of an output pad value, the pad can be configured as both output and input so the actual pad value can be read back and compared with the expected value.

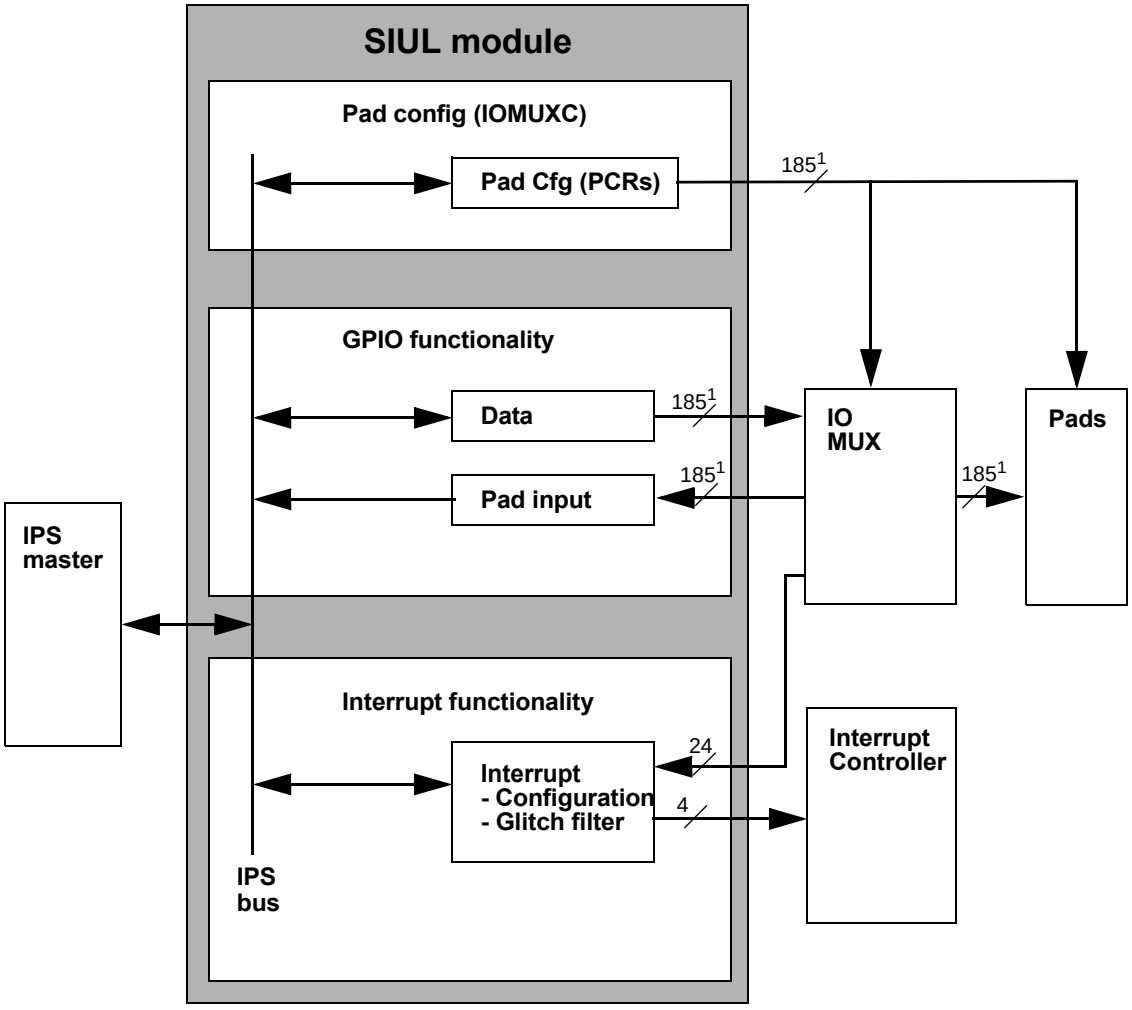



**Figure 43-1. System Integration Unit Lite block diagram**

[1] Up to 128 pins in the 176-pin package; up to 150 pins in the 208-pin package; up to 177 pins on the 416-pin package

## 43.3 Features

The System Integration Unit Lite supports these distinctive features:

- GPIO
  - GPIO function on up to 185 unique I/O pins
  - Dedicated input and output registers for each GPIO pin
- External interrupts
  - 3 system interrupt vectors for up to 24 interrupt sources
  - 16 programmable digital glitch filters
  - Independent interrupt mask

- Edge detection
- System configuration
  - Pad configuration control
- Special function control
  - DDR pad configuration
  - RSDS pad configuration
  - Nexus pad configuration

## 43.4 External signal description

Most device pads support multiple device functions. Pad configuration registers are provided to enable selection between GPIO and other signals. These other signals, also referred to as alternate functions, are typically peripheral functions.

GPIO pads are grouped in “ports”, with each port containing up to 16 pads. With appropriate configuration, all pins in a port can be read or written to in parallel with a single R/W access.

**NOTE**

In order to use GPIO port functionality, all pads in the port must be configured as GPIO rather than as alternate functions.

### 43.4.1 Detailed signal descriptions

#### 43.4.1.1 General-purpose I/O pins (GPIO[0:184])

The GPIO pins provide general-purpose input and output function. The GPIO pins are generally multiplexed with other I/O pin functions. Each GPIO input and output is separately controlled by an input (GPDI*n_n*) or output (GPDO*n_n*) register.

#### 43.4.1.2 External interrupt request input pins (EIRQ[0:23])[1]

The EIRQ[0:23] are connected to the SIU inputs. Rising or falling edge events are enabled by setting the corresponding bits in the SIU_IREER or the SIU_IFEER register.

Table 43-1 lists the external interrupt pins used by the SIUL.

1. EIRQ[0:15] in the 176-pin package; EIRQ[0:18] in the 208-pin package; EIRQ[0:23] in the 416-pin package

Table 43-1. SIUL external interrupt mapping

| External IRQ | Flag | PCR | Port | Package | | |
|---|---|---|---|---|---|---|
| | | | | 176 | 208 | 416 |
| IRQ_0 | EIF[0] | PCR[0] | PA[0] | x | x | x |
| | EIF[1] | PCR[1] | PA[1] | x | x | x |
| | EIF[2] | PCR[8] | PA[8] | x | x | x |
| | EIF[3] | PCR[9] | PA[9] | x | x | x |
| | EIF[4] | PCR[16] | PB[0] | x | x | x |
| | EIF[5] | PCR[18] | PB[2] | x | x | x |
| | EIF[6] | PCR[27] | PB[11] | x | x | x |
| | EIF[7] | PCR[29] | PB[13] | x | x | x |
| IRQ_1 | EIF[8] | PCR[71] | PF[1] | x | x | x |
| | EIF[9] | PCR[79] | PF[9] | x | x | x |
| | EIF[10] | PCR[86] | PG[0] | x | x | x |
| | EIF[11] | PCR[87] | PG[1] | x | x | x |
| | EIF[12] | PCR[98] | PG[12] | x | x | x |
| | EIF[13] | PCR[129] | PK[8] | | x | x |
| | EIF[14] | PCR[131] | PK[10] | x | x | x |
| | EIF[15] | PCR[132] | PK[11] | x | x | x |
| IRQ_2 | EIF[16] | PCR[134] | PL[1] | | x | x |
| | EIF[17] | PCR[136] | PL[3] | | x | x |
| | EIF[18] | PCR[151] | PM[4] | | x | x |
| | EIF[19] | PCR[158] | PM[11] | x | | |
| | EIF[20] | PCR[162] | PN[1] | | | x |
| | EIF[21] | PCR[164] | PN[3] | | | x |
| | EIF[22] | PCR[172] | PN[11] | | | x |
| | EIF[23] | PCR[180] | PP[3] | | | x |

#### 43.4.1.3 Special function output pins configuration (PCR[185:281])

PCR registers exist for the special function pads. These typically have much more limited functionality than the GPIO pads.

## 43.5 Memory Map and Register Description

This section provides a detailed description of all registers accessible in the SIUL module.

### 43.5.1 SIU memory map

Table 43-2 gives an overview on the SIUL registers implemented.

**Table 43-2. SIUL memory map**

| Address Na | me | Description | Size (bits) | Location |
|---|---|---|---|---|
| Base (0xC3F9_0000) | — | Reserved | — | |
| Base + 0x0004 | MIDR1 | MCU ID Register #1 | 32 | on page 1485 |
| Base + 0x0008 | MIDR2 | MCU ID Register #2 | 32 | on page 1485 |
| Base + (0x000C - 0x0013) | — | Reserved | — | |
| Base + 0x0014 | ISR | Interrupt Status Flag Register | 32 | on page 1486 |
| Base + 0x0018 | IRER | Interrupt Request Enable Register | 32 | on page 1487 |
| Base + (0x001C - 0x0027) | — | Reserved | — | |
| Base + 0x0028 | IREER | Interrupt Rising Edge Event Enable | 32 | on page 1487 |
| Base + 0x002C | IFEER | Interrupt Falling-Edge Event Enable | 32 | on page 1488 |
| Base + 0x0030 | IFER | IFER Interrupt Filter Enable Register | 32 | on page 1489 |
| Base + (0x0034 - 0x003F) | — | Reserved | — | |
| Base + 0x0040 - Base + 0x01B1 | PCR0–PCR184[1] | Pad Configuration Registers 0–184 | 16 | on page 1489 |
| Base + 0x01B2– Base + 0x0272 | PCR185–PCR281 | Pad Configuration Registers 185–281 (used for non-GPIO and special functions) | 16 | on page 1491 |
| Base + (0x0274–0x04FF) | — | Reserved | — | |
| Base + 0x0500 - Base + 0x052B | PSMI0_3– PSMI50_53 | Pad Selection for Multiplexed Inputs | 32 | on page 1493 |
| Base + (0x052C–0x05FF) | — | Reserved | — | |
| Base + 0x0600 - Base + 0x06B8 | GPDO0_3 - GPDO184_187[1] | GPIO Pad Data Output Register | 32 | on page 1497 |
| Base + (0x06BC–0x07FF) | — | Reserved | — | |
| Base + 0x0800 - Base + 0x08B8 | GPDI0_3 - GPDI184_187[1] | GPIO Pad Data Input Register | 32 | on page 1498 |
| Base + (0x08BC–0x0BFF) | — | Reserved | — | |
| Base + 0x0C00 - Base + 0x0C14 | PGPDO0 - PGPDO5 | Parallel GPIO Pad Data Out Register | 32 | on page 1499 |
| Base + (0x0C18–0x0C3F) | — | Reserved | — | |
| Base + 0x0C40 - Base + 0x0C54 | PGPDI0 - PGPDI5 | Parallel GPIO Pad Data In Register | 32 | on page 1500 |
| Base + (0x0C58–0x0C7F) | — | Reserved | — | |
| Base + 0x0C80 - Base + 0x0CAC | MPGPDO0 - MPGPDO11 | Masked Parallel GPIO Pad Data Out Register | 32 | on page 1501 |

Table 43-2. SIUL memory map (continued)

| Address Na | me | Description | Size (bits) | Location |
|---|---|---|---|---|
| Base + (0x0CB0–0x0FFF) | — | Reserved | — | |
| Base + 0x1000 - Base + 0x105C | IFMC0 - IFMC23[1] | Interrupt Filter Maximum Counter Register | 32 | on page 1502 |
| Base + (0x1060–0x107C) | — | Reserved | — | |
| Base + 0x1080 | IFCP | Interrupt Filter Clock Prescaler Register | 32 | on page 1503 |
| Base + (0x1084 - 0x3FFF) | — | Reserved | — | |

1 Not all port pins are available in all packages.

**NOTE**

A transfer error will be issued when trying to access reserved register space.

## 43.5.2 Register protection

The individual registers of System Integration Unit Lite are protected from accidental writes. The following registers are protected:

- IRER (*Interrupt Request Enable Register*)
- IREER (*Interrupt Rising-Edge Event Enable Register*)
- IFEER (*Interrupt Falling-Edge Event Enable Register*)
- IFER (*Interrupt Filter Enable Register*), entire PortA, PortB[0:3] and PortC[2:15]
- PSMI[n] (*Pad Selection for Multiplexed Inputs*)
- IFMC[n] (*Interrupt Filter Maximum Counter*)
- IFCP (*Interrupt Filter Clock Prescaler*)
- PCR[n] (*Pad Configuration Registers*)

Refer to Appendix A, “Registers Under Protection,” for details.

## 43.5.3 Register description

This section describes in address order all the SIUL registers. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order. The numbering convention of register is MSB=0, however the numbering of internal field is LSB=0, e.g. PARTNUM[5] = MIDR1[10].

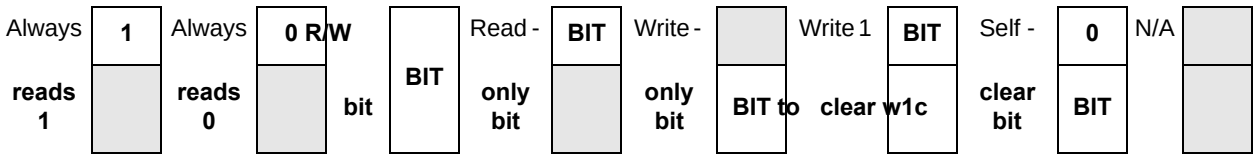



Figure 43-2. Key to Register Fields

### 43.5.3.1 MCU ID Register #1 (MIDR1)

This register holds identification information about the device.

Address: Base + 0x0004 Access: None

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PARTNUM[15:0] | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CSP | PKG[4:0] | | | | | | | MAJOR_MASK[3:0] | | | | MINOR_MASK[3:0] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 or 1 depending on package | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Figure 43-3. MCU ID Register #1 (MIDR1)**

**Table 43-3. MIDR1 Field Descriptions**

| Field D | escription |
|---|---|
| PARTNUM [15:0] | MCU Part Number<br>Device part number of the MCU.<br>0101_0110_0100_0101: Device with 2 MB flash memory<br>For the full part number this field needs to be combined with MIDR2.PARTNUM[0:7] |
| CSP | Always reads back 0 |
| PKG[4:0] | Package Settings<br>Can be read by software to determine the package type that is used for the particular device:<br>0b10001: 176-pin QFP<br>0b10101: 208-pin QFP<br>0b10100: 416-pin BGA |
| MAJOR_MASK[3:0] | Major Mask Revision<br>Counter starting at 0x0. Incremented each time when there is a resynthesis. |
| MINOR_MASK[3:0] | Minor Mask Revision<br>Counter starting at 0x0. Incremented each time a mask change is done. |

### 43.5.3.2 MCU ID Register #2 (MIDR2)

Field description same as SSCM MCU ID Register #2 (see Section 44.2.2, “Register description”).

Address: Base + 0x0008 Access: None

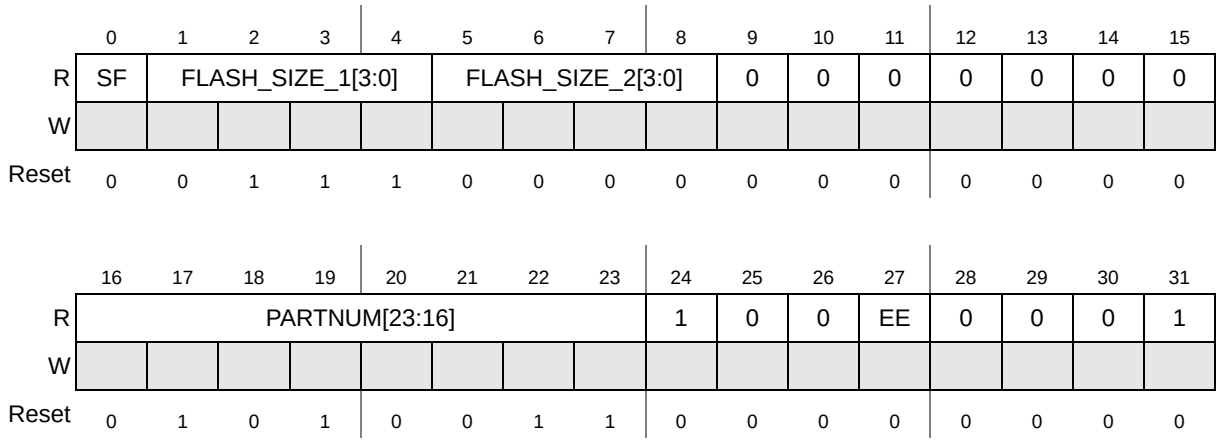



**Figure 43-4. MCU ID Register #2 (MIDR2)**

**Table 43-4. MIDR2 Field Descriptions**

| Field De | scription |
|---|---|
| SF | Manufacturer<br>0: Freescale Semiconductor<br>1: Reserved |
| FLASH_SIZE_1 | Coarse granularity for Flash memory size<br>Needs to be added to the memory size indicated by FLASH_SIZE_2 to calculate the actual memory size.<br>0b0111: 2 MB |
| FLASH_SIZE_2 | Fine granularity for Flash memory size<br>Needs to be added to the memory size indicated by FLASH_SIZE_1 to calculate the actual memory size.<br>0b0000: 0 x (FLASH_SIZE_1 / 8)<br>0b0010: 2 x (FLASH_SIZE_1 / 8)<br>0b0100: 4 x (FLASH_SIZE_1 / 8) |
| PARTNUM | ASCII character in MCU Part Number<br>0x53: Character ‘S’ (this device) |
| EE | Data Flash present<br>0: No Data Flash is present<br>1: Data Flash is present |

### 43.5.3.3 Interrupt Status Flag Register (ISR)

This register holds the external interrupt flags.

Address: Base + 0x0014 Access: User read/write (write 1 to clear)

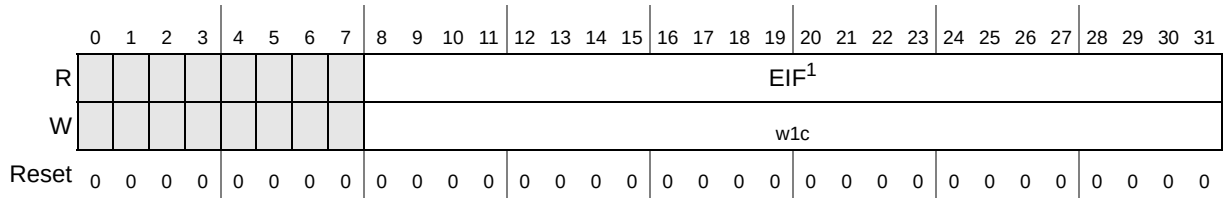



**Figure 43-5. Interrupt Status Flag Register (ISR)**

1 {EIF[19], EIF[15:14], EIF[12:0]} in the 176-pin package; EIF[18:0] in the 208-pin package; EIF[23:20, 18:0] in the 416-pin package

**Table 43-5. ISR Field Descriptions**

| Field Des | cription |
|---|---|
| EIF[x] | External Interrupt Status Flag x<br>This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER[x]), EIF[x] causes an interrupt request.<br>0: No interrupt event has occurred on the pad<br>1: An interrupt event as defined by IREER[x] and IFEER[x] has occurred |

### 43.5.3.4 Interrupt Request Enable Register (IRER)

This register is used to enable the external interrupt messaging to the interrupt controller.

Address: Base + 0x0018 Access: User read/write

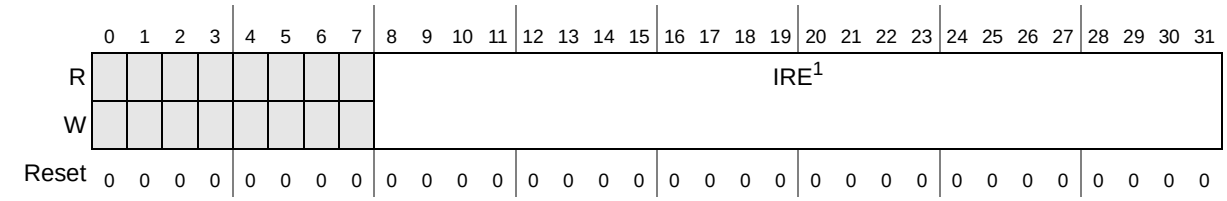



**Figure 43-6. Interrupt Request Enable Register (IRER)**

1 {IRE[19], IRE[15:14], IRE[12:0]} in the 176-pin package; IRE[18:0] in the 208-pin package; IRE[23:20, 18:0] in the 416-pin package

**Table 43-6. IRER Field Descriptions**

| Field Des | cription |
|---|---|
| IRE[x] | External Interrupt Request Enable x<br>1: A set EIF[x] bit causes an interrupt request<br>0: Interrupt requests from the corresponding EIF[x] bit are disabled |

### 43.5.3.5 Interrupt Rising-Edge Event Enable Register (IREER)

This register is used to enable rising-edge triggered events to be enabled on the corresponding external interrupt pads.

Address: Base + 0x0028 Access: User read/write

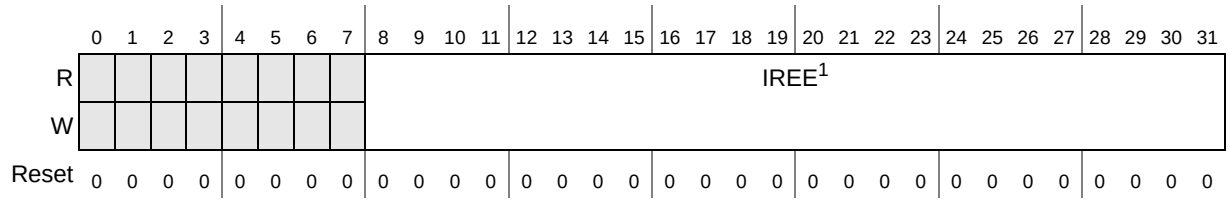



**Figure 43-7. Interrupt Rising-Edge Event Enable Register (IREER)**

1 {IREE[19], IREE[15:14], IREE[12:0]} in the 176-pin package; IREE[18:0] in the 208-pin package; IREE[23:20, 18:0] in the 416-pin package

**Table 43-7. IREER Field Descriptions**

| Field Des | cription |
|---|---|
| IREE[x] | Enable rising-edge events to cause the EIF[x] bit to be set.<br>1: Rising-edge event is enabled<br>0: Rising-edge event is disabled |

### 43.5.3.6 Interrupt Falling-Edge Event Enable Register (IFEER)

This register is used to enable falling-edge triggered events to be enabled on the corresponding external interrupt pads.

Address: Base + 0x002C Access: User read/write

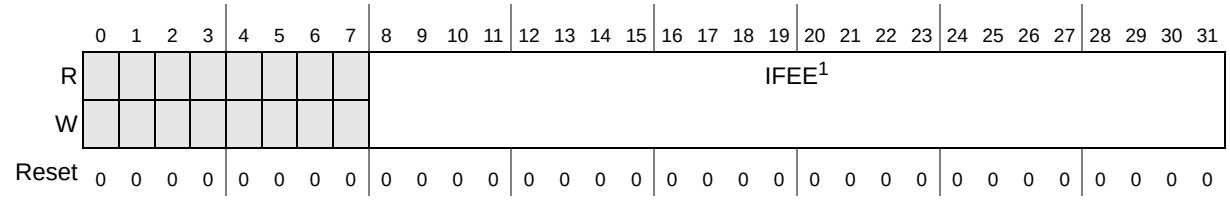



**Figure 43-8. Interrupt Falling-Edge Event Enable Register (IFEER)**

1 {IFEE[19], IFEE[15:14], IFEE[12:0]} in the 176-pin package; IFEE[18:0] in the 208-pin package; IFEE[23:20, 18:0] in the 416-pin package

**Table 43-8. IFEER Field Descriptions**

| Field Des | cription |
|---|---|
| IFEE[x] | Enable falling-edge events to cause the EIF[x] bit to be set.<br>1: Falling-edge event is enabled<br>0: Falling-edge event is disabled |

**NOTE**

If both the IREE and IFEE bit is cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

### 43.5.3.7 Interrupt Filter Enable Register (IFER)

This register is used to enable a digital filter counter on the corresponding external interrupt pads to filter out glitches on the inputs.

Address: Base + 0x0030 Access: User read/write

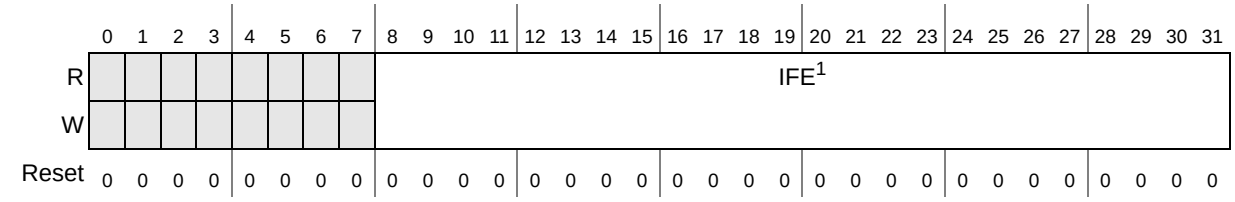



**Figure 43-9. Interrupt Filter Enable Register (IFER)**

1 IFE[15:0] in the 176-pin package; IFE[18:0] in the 208-pin package; IFE[23:0] in the 416-pin package

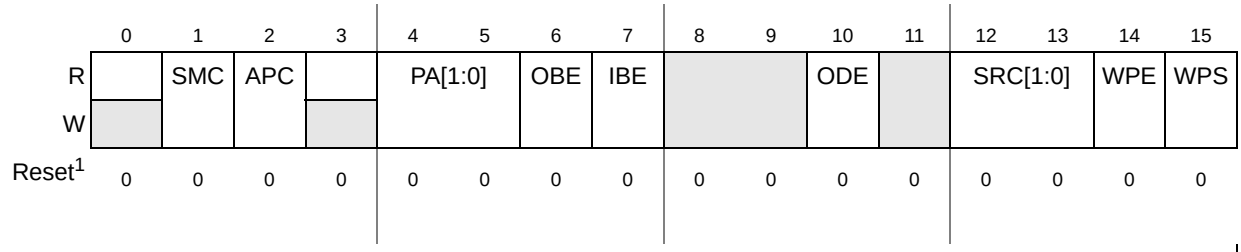
**Table 43-9. IFER Field Descriptions**

| Field Des | cription |
|---|---|
| IFE[x] | Enable digital glitch filter on the interrupt pad input.<br>1: Filter is enabled<br>0: Filter is disabled |

### 43.5.3.8 Pad Configuration Registers (PCR0–PCR184)

The Pad Configuration Registers allow configuration of the static electrical and functional characteristics associated with I/O pads. Each PCR controls the characteristics of a single pad.

Address: Base + 0x0040 (PCR0)(185 registers) Access: User read/write
Base + 0x0042 (PCR1)
...
Base + 0x01B0 (PCR184)




1 Reset value shown is for the most of the PCRs, however, some PCRs are initialized to different values dependent on the requirements of the device. See Chapter 3, “Signal Description,” for the reset configurations of each PCR on this device.

**Figure 43-10. Pad Configuration Registers (PCRx)**

**NOTE**

16/32-bit access supported

In addition to the bit map above, the following Table 43-11 describes the PCR register depending on the pad type. The bits in shaded fields are not implemented for the particular I/O type. The PA field selecting

the number of alternate functions may or may not be present depending on the number of alternate functions actually mapped on the pad.

**Figure 43-11. Pad Configuration Register (PCR) for the different pad types in MPC5645S**

| Pad type | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pad with slew rate control (F,M, and S) | | | SMC | APC | | PA[1:0] | | | OBE | IBE | | | ODE | | SRC[1:0] | | | WPE | WPS |
| Pad with reduced slew rate control (SMD) | | | SMC | APC | | PA[1:0] | | | OBE | IBE | | | ODE | | | SRC [1] | | WPE | WPS |
| Pad with GPIO and analog functionality (J) | | | SMC | APC | | PA[1:0] | | | OBE | IBE | | | ODE | | SRC[1:0] | | | WPE | WPS |

**Table 43-10. PCRx Field Descriptions**

| Field De | scription |
|---|---|
| SMC | Safe Mode Control<br>This bit supports the overriding of the automatic deactivation of the output buffer of the associated pad upon entering SAFE mode of the SoC.<br>1: In SoC SAFE mode, the output buffer remains functional.<br>0: In SoC SAFE mode, the output buffer of the pad is disabled. |
| APC | Analog Pad Control<br>This bit enables the usage of the pad as analog input.<br>1: Analog input path switch can be enabled by the ADC.<br>0: Analog input path from the pad is gated and can not be used. |
| PA[1:0] | Pad Output Assignment<br>This field is used to select the function to drive the output of a multiplexed pad. Note that for non-GPIO functions the output buffer is automatically enabled when the function is selected.<br>00: Alternative Mode 0: GPIO.<br>01: Alternative Mode 1: see Chapter 3, “Signal Description”<br>10: Alternative Mode 2: see Chapter 3, “Signal Description”<br>11: Alternative Mode 3: see Chapter 3, “Signal Description”<br>**Note:** Number of bit depending of the number of actual alternate function. Please refer to datasheet |
| OBE | Output Buffer Enable<br>This bit enables the output buffer of the pad when the pad is in GPIO mode.<br>1: Output Buffer of the pad is enabled when PA = 00.<br>0: Output Buffer of the pad is disabled when PA = 00. |
| IBE | Input Buffer Enable<br>This bit enables the input buffer of the pad.<br>1: Input Buffer of the pad is enabled.<br>0: Input Buffer of the pad is disabled. |

**Table 43-10. PCRx Field Descriptions (continued)**

| Field De | scription |
|---|---|
| ODE | Open Drain Output Enable<br>This bit controls output driver configuration for the pads connected to this signal. Either open drain or push/pull driver configurations can be selected. This feature applies to output pads only.<br>1: Open drain enable signal is asserted for the pad.<br>0: Open drain enable signal is negated for the pad. |
| SRC[1:0] | Slew Rate Control<br>This field controls the slew rate control output signals from the SIUL. The output signals are driven to the value of this field. The actual slew rates are defined by the implementation of the pad devices for a given SoC.<br>**Note:** For low-power modes, keeping these bits asserted may result in more leakage. It is recommended to not drive these bits during low-power modes. |
| WPE | Weak Pull Up/Down Enable<br>This bit controls whether the weak pull up/down devices are enabled/disabled for the pad connected to this signal.<br>1: Weak pull device enabled for the pad.<br>0: Weak pull device disabled for the pad. |
| WPS | Weak Pull Up/Down Select<br>This bit controls whether weak pull up or weak pull down devices are used for the pads connected to this signal when weak pull up/down devices are enabled.<br>1: Weak pull-up selected<br>0: Weak pull-down selected |

### 43.5.3.9 Pad Configuration Registers (PCR185–PCR281)

The non-GPIO and special functions pins have dedicated Pad Configuration Registers that allow configuration of the electrical characteristics associated with the pads. Each PCR controls the characteristics of a single pad.

**Figure 43-12. Pad Configuration Register (PCR) definition for PCR185–281**

| Pad type | 0 1 2 3 4 5 6 7 | | | | | | | | 8 9 | 10 | 11 | 12 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nexus | | SMC | | | | | OBE | IBE | | ODE | | SRC[1:0] | WPE | WPS |
| DDR | | SMC | | | | | | | | ODE | SRC[2:0] | | WPE | WPS |
| RSDS | | SMC | | | | | | | | ODE | | SRC[1:0] | WPE | WPS |

**Table 43-11. PCR185–281 Field Descriptions**

| Field De | scription |
|---|---|
| SMC | Safe Mode Control<br>This bit supports the overriding of the automatic deactivation of the output buffer of the associated pad upon entering SAFE mode of the SoC.<br>1: In SoC SAFE mode, the output buffer remains functional.<br>0: In SoC SAFE mode, the output buffer of the pad is disabled. |
| OBE | Output Buffer Enable<br>This bit enables the output buffer of the pad in case the pad is in GPIO mode.<br>1: Output Buffer of the pad is enabled when PA = 00.<br>0: Output Buffer of the pad is disabled when PA = 00. |
| IBE | Input Buffer Enable<br>This bit enables the input buffer of the pad.<br>1: Input Buffer of the pad is enabled.<br>0: Input Buffer of the pad is disabled. |
| ODE | Open Drain Output Enable<br>This bit controls output driver configuration for the pads connected to this signal. Either open drain or push/pull driver configurations can be selected. This feature applies to output pads only.<br>1: Open drain enable signal is asserted for the pad.<br>0: Open drain enable signal is negated for the pad. |
| SRC | Slew Rate Control<br>This field controls the slew rate control output signals from the SIUL. The output signals are driven to the value of this field. The actual slew rates are defined by the implementation of the pad devices for a given SoC.<br>**Note:** For low-power modes, keeping these bits asserted may result in more leakage. It is recommended to not drive these bits during low-power modes. |
| WPE | Weak Pull Up/Down Enable<br>This bit controls whether the weak pull up/down devices are enabled/disabled for the pad connected to this signal.<br>1: Weak pull device enabled for the pad.<br>0: Weak pull device disabled for the pad. |
| WPS | Weak Pull Up/Down Select<br>This bit controls whether weak pull up or weak pull down devices are used for the pads connected to this signal when weak pull up/down devices are enabled.<br>1: Weak pull-up selected<br>0: Weak pull-down selected |

**NOTE**

The SIUL_PCR_224 controls the ddr_dummy pad's IBE, OBE.

- In run/stop mode, default value for IBE, OBE for ddr_dumy is 1 in SIUL_PCR as well as at signal level
- In run/stop mode, IBE and OBE are configurable.
- In standby, IBE, OBE would be 0 for ddr_dummy.

It is important to configure the slew rate control correctly for the DDR pads. See Table 43-12 for appropriate configuration values.

**Table 43-12. PCR202-268 slew rate settings**

| SRC2 | SRC1 | SRC0 | Mode |
|---|---|---|---|
| 1 | 1 | 1 | 3.3 V SDR |
| 1 | 1 | 0 | 1.8 V DDR2 full strength |
| 1 | 0 | 1 | n/a |
| 1 | 0 | 0 | n/a |
| 0 | 1 | 1 | 2.5 V DDR1 |
| 0 | 1 | 0 | 1.8 V DDR2 half strength |
| 0 | 0 | 1 | 1.8 V low-power DDR full speed |
| 0 | 0 | 0 | 1.8 V low power DDR half speed |

### 43.5.3.10 Pad Selection for Multiplexed Inputs Registers (PSMI0_3–PSMI50_53)

The PSMI registers control selection of the source pad for input signals to on-chip modules. Each PSMI register applies to a single input signal. Configure the PSMI registers to select the pad on which the external input signal is connected.

Address: Base + 0x0500 - 0x0534 (54 registers) Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PADSEL0 | | | | 0 | 0 | 0 | 0 | PADSEL1 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | PADSEL2 | | | | 0 | 0 | 0 | 0 | PADSEL3 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 43-13. Pad Selection for Multiplexed Inputs Register (PSMI0_3)**

**Table 43-13. PSMI0_3 Field Descriptions**

| Field De | scription |
|---|---|
| PADSEL0 - 3,<br>PADSEL4 - 7,<br>...<br>PADSEL40 - 43 | Pad Selection Bits<br>Each PADSEL field selects the pad currently used for a certain input function. See Table 43-14. |

In order to multiplex different pads to the same peripheral input, the SIUL provides a register that controls the selection between the different sources.

**Table 43-14. Peripheral input pin selection**

| PSMI register | SIUL address offset | Peripheral input | Mapping of input pin to peripheral input [1] |
|---|---|---|---|
| PSMI[0] | 0x500 | CANRX_0 | 0:PCR[17]<br>1:PCR[135] |
| PSMI[1] | 0x501 | CANRX_1 | 0:PCR[26]<br>1:PCR[133] |
| PSMI[2] | 0x502 | CANRX_2 | 0:PCR[150]<br>1:PCR[157]<br>2:PCR[179] |
| PSMI[3] | 0x503 | PDI_PCLK | 0:PCR[119]<br>1:PCR[142]<br>2:PCR[156] |
| PSMI[4] | 0x504 | VIU[0]_PDI8 | 0:PCR[123]<br>1:PCR[163] |
| PSMI[5] | 0x505 | VIU[1]_PDI9 | 0:PCR[124]<br>1:PCR[164] |
| PSMI[6] | 0x506 | VIU[2]_PDI10 | 0:PCR[125]<br>1:PCR[171] |
| PSMI[7] | 0x507 | VIU[3]_PDI11 | 0:PCR[126]<br>1:PCR[172] |
| PSMI[8] | 0x508 | VIU[4]_PDI12 | 0:PCR[127]<br>1:PCR[179] |
| PSMI[9] | 0x509 | VIU[5]_PDI13 | 0:PCR[118]<br>1:PCR[137]<br>2:PCR[152] |
| PSMI[10] | 0x50A | VIU[6]_PDI14 | 0:PCR[120]<br>1:PCR[138]<br>2:PCR[153] |
| PSMI[11] | 0x50B | VIU[7]_PDI15 | 0:PCR[104]<br>1:PCR[122]<br>2:PCR[139] |
| PSMI[12] | 0x50C | VIU[8]_PDI16 | 0:PCR[75]<br>1:PCR[140]<br>2:PCR[154] |
| PSMI[13] | 0x50D | VIU[9]_PDI17 | 0:PCR[76]<br>1:PCR[141]<br>2:PCR[155] |
| PSMI[14] | 0x50E | PDI_DE | 0:PCR[98]<br>1:PCR[108]<br>2:PCR[180] |
| PSMI[15] | 0x50F | DSPI1_SS | 0:PCR[43]<br>1:PCR[98] |

**Table 43-14. Peripheral input pin selection (continued)**

| PSMI register | SIUL address offset | Peripheral input | Mapping of input pin to peripheral input [1] |
|---|---|---|---|
| PSMI[16] | 0x510 | EMIOS0[8] | 0:PCR[50]<br>1:PCR[106]<br>2:PCR[156] |
| PSMI[17] | 0x511 | EMIOS0[9] | 0:PCR[49]<br>1:PCR[55]<br>2:PCR[107] |
| PSMI[18] | 0x512 | EMIOS0[10] | 0:PCR[56]<br>1:PCR[116] |
| PSMI[19] | 0x513 | EMIOS0[11] | 0:PCR[57]<br>1:PCR[115]<br>2:PCR[181] |
| PSMI[20] | 0x514 | EMIOS0[12] | 0:PCR[58]<br>1:PCR[114] |
| PSMI[21] | 0x515 | EMIOS0[13] | 0:PCR[59]<br>1:PCR[113]<br>2:PCR[182] |
| PSMI[22] | 0x516 | EMIOS0[14] | 0:PCR[60]<br>1:PCR[112] |
| PSMI[23] | 0x517 | EMIOS0[15] | 0:PCR[61]<br>1:PCR[111]<br>2:PCR[183] |
| PSMI[24] | 0x518 | EMIOS0[16] | 0:PCR[51]<br>1:PCR[110]<br>2:PCR[157] |
| PSMI[25] | 0x519 | EMIOS0[17] | 0:PCR[1]<br>1:PCR[113]<br>2:PCR[173] |
| PSMI[26] | 0x51A | EMIOS0[18] | 0:PCR[0]<br>1:PCR[112]<br>2:PCR[174] |
| PSMI[27] | 0x51B | EMIOS0[19] | 0:PCR[9]<br>1:PCR[111]<br>2:PCR[175] |
| PSMI[28] | 0x51C | EMIOS0[20] | 0:PCR[8]<br>1:PCR[110]<br>2:PCR[176] |
| PSMI[29] | 0x51D | EMIOS0[21] | 0:PCR[86]<br>1:PCR[109]<br>2:PCR[177] |
| PSMI[30] | 0x51E | EMIOS0[22] | 0:PCR[87]<br>1:PCR[108]<br>2:PCR[178] |

**Table 43-14. Peripheral input pin selection (continued)**

| PSMI register | SIUL address offset | Peripheral input | Mapping of input pin to peripheral input [1] |
|---|---|---|---|
| PSMI[31] | 0x51F | EMIOS0[23] | 0:PCR[52]<br>1:PCR[109]<br>2:PCR[158] |
| PSMI[32] | 0x520 | EMIOS1[8] | 0:PCR[46]<br>1:PCR[104]<br>2:PCR[118] |
| PSMI[33] | 0x521 | EMIOS1[9] | 0:PCR[106]<br>1:PCR[120]<br>2:PCR[127] |
| PSMI[34] | 0x522 | EMIOS1[10] | 0:PCR[28]<br>1:PCR[123]<br>2:PCR[143] |
| PSMI[35] | 0x523 | EMIOS1[11] | 0:PCR[29]<br>1:PCR[124]<br>2:PCR[144] |
| PSMI[36] | 0x524 | EMIOS1[12] | 0:PCR[125]<br>1:PCR[131]<br>2:PCR[145] |
| PSMI[37] | 0x525 | EMIOS1[13] | 0:PCR[126]<br>1:PCR[132]<br>2:PCR[146] |
| PSMI[38] | 0x526 | EMIOS1[14] | 0:PCR[74]<br>1:PCR[107]<br>2:PCR[122] |
| PSMI[39] | 0x527 | EMIOS1[15] | 0:PCR[75]<br>1:PCR[77]<br>2:PCR[116] |
| PSMI[40] | 0x528 | EMIOS1[16] | 0:PCR[47]<br>1:PCR[76]<br>2:PCR[154] |
| PSMI[41] | 0x529 | EMIOS1[17] | 0:PCR[115]<br>1:PCR[119]<br>2:PCR[149] |
| PSMI[42] | 0x52A | EMIOS1[18] | 0:PCR[25]<br>1:PCR[121]<br>2:PCR[139] |
| PSMI[43] | 0x52B | EMIOS1[19] | 0:PCR[24]<br>1:PCR[70]<br>2:PCR[140] |
| PSMI[44] | 0x52C | EMIOS1[20] | 0:PCR[23]<br>1:PCR[71]<br>2:PCR[141] |

Table 43-14. Peripheral input pin selection (continued)

| PSMI register | SIUL address offset | Peripheral input | Mapping of input pin to peripheral input [1] |
|---|---|---|---|
| PSMI[45] | 0x52D | EMIOS1[21] | 0:PCR[73]<br>1:PCR[103]<br>2:PCR[142] |
| PSMI[46] | 0x52E | EMIOS1[22] | 0:PCR[114]<br>1:PCR[135]<br>2:PCR[152] |
| PSMI[47] | 0x52F | EMIOS1[23] | 0:PCR[48]<br>1:PCR[136]<br>2:PCR[153]<br>3:PCR[155] |
| PSMI[48] | 0x530 | SCL_1 | 0:PCR[1]<br>1:PCR[77]<br>2:PCR[132]<br>3:PCR[134] |
| PSMI[49] | 0x531 | SDA_1 | 0:PCR[0]<br>1:PCR[74]<br>2:PCR[131]<br>3:PCR[133] |
| PSMI[50] | 0x532 | RXD_0 | 0:PCR[19]<br>1:PCR[17] |
| PSMI[51] | 0x533 | RXD_1 | 0:PCR[28]<br>1:PCR[78] |
| PSMI[52] | 0x534 | RXD_2 | 0:PCR[128]<br>1:PCR[157]<br>2:PCR[163] |
| PSMI[53] | 0x535 | RXD_3 | 0:PCR[150]<br>1:PCR[171] |
| PSMI[54] | 0x536 | EVTI | 0:PCR[80]<br>1:EVTI |

1 Connecting a peripheral input to a pad requires assigning both the PSMI value for the peripheral input and the pad assignment in the SIU_PCR register for that signal.

### 43.5.3.11 GPIO Pad Data Output Registers (GPDO0_3 - GPDO184)

These registers can be used to set or clear a single GPIO pad with byte access.

The GPIO pad data output registers are a group of 185 one-byte registers used to set or clear the logic value on their associated pads. Each word contains four registers. The word beginning at Base + 0x0600 contains GPDO0 - GPDO3, the word beginning at Base + 0x0604 contains GPDO3 - GPDO07, and so on.

Address: Base + 0x0600 - 0x06B8 (47 registers) Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO [0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO [1] |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO [2] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDO [3] |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 43-14. Port GPIO Pad Data Output register 0 - 3 (GPDO0_3)**

**Table 43-15. GPDO0_3 Field Descriptions**

| Field De | scription |
|---|---|
| PDO[x] | Pad Data Out<br>This bit stores the data to be driven out on the external GPIO pad controlled by this register.<br>1: Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output<br>0: Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output |

### 43.5.3.12 GPIO Pad Data Input Registers (GPDI0_3–GPDI184)

These registers can be used to read the GPIO pad data with a byte access.

The GPIO pad data input registers are a group of 185 one-byte registers used to set or clear the logic value on their associated pads. Each word contains four registers. The word beginning at Base + 0x0600 contains GPDI0 - GPDI3, the word beginning at Base + 0x0604 contains GPDI3 - GPDI07, and so on.

Address: Base + 0x0800 - 0x08B8 (47 registers) Access: User read

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDI [0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDI [1] |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDI [2] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PDI [3] |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 43-15. Port GPIO Pad Data Input register 0 - 3 (GPDI0_3)**

**Table 43-16. GPDO0_3 Field Descriptions**

| Field De | scription |
|---|---|
| PDI[x] | Pad Data In<br>This bit stores the value of the external GPIO pad associated with this register.<br>1: The value of the data in signal for the corresponding GPIO pad is logic high<br>0: The value of the data in signal for the corresponding GPIO pad is logic low |

### 43.5.3.13 Parallel GPIO Pad Data Out Registers (PGPDO0–PGPDO5)

The input/output ports are constructed such that they contain 16 GPIO pins, for example PortA[0..15]. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration.

Writing a parallel PGPDO register directly sets the associated GPDO register bits. There is also a masked parallel port output register allowing the user to determine which pins within a port are written.

While very convenient and fast, this approach does have implications regarding current consumption for the device power segment containing the port GPIO pads. Toggling several GPIO pins simultaneously can significantly increase current consumption.

**WARNING**

Caution must be taken to avoid exceeding maximum current thresholds when toggling multiple GPIO pins simultaneously. Please refer to data sheet.

Address: Base + 0x0C00 - 0x0C14 (6 registers) Access: User read/write

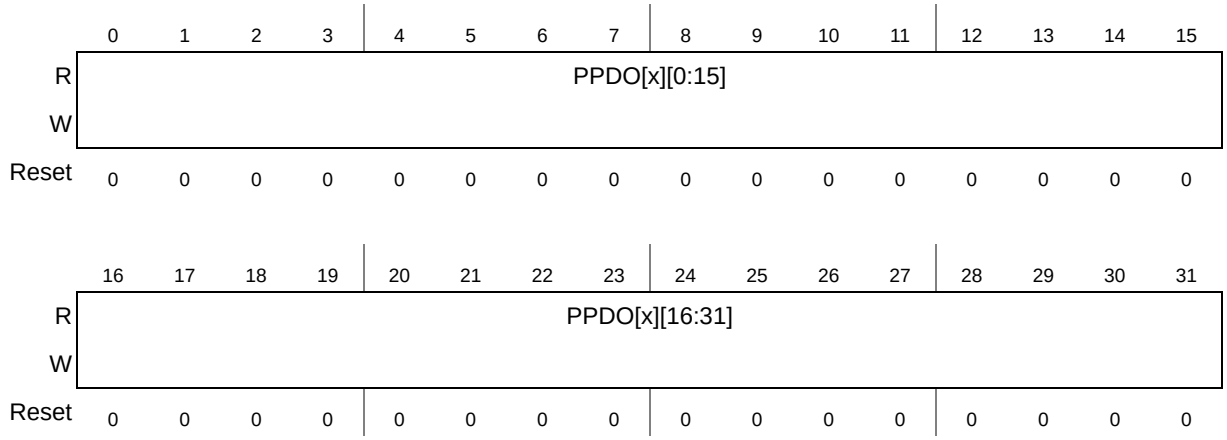



**Figure 43-16. Parallel GPIO Pad Data Out Register (PGPDO0)**

**Table 43-17. PGPDO0_5 Field Descriptions**

| Field De | scription |
|---|---|
| PPDO[x] | Parallel Pad Data Out<br>Write or read the data register that stores the value to be driven on the pad in output mode. Accesses to this register location are coherent with accesses to the bit-wise GPIO Pad Data Output Registers (GPDO0_3 - GPDO184).<br>The x and bit index define which PPDO register bit is equivalent to which PDO register bit according to the equation<br><br>PPDO[x][y] = PDO[(x*32)+y]<br><br>where x is the PGPDO register number (0..5) and y is the bit number within that register (0..31). |

**NOTE**

It is important to note the bit ordering of the ports in the parallel port registers. The most significant bit of the parallel port register corresponds to the least significant pin in the port.

### 43.5.3.14 Parallel GPIO Pad Data In Register (PGPDI0–PGPDI5)

The SIU_PGPDI registers are similar in operation to the PGPDO registers, described in the previous section (Section 43.5.3.13, “Parallel GPIO Pad Data Out Registers (PGPDO0–PGPDO5)”) but they are used to read port pins simultaneously.

**NOTE**

The port pins to be read need to be configured as inputs but even if a single pin within a port has IBE set, then you can still read that pin using the parallel port register. However, this does mean you need to be very careful.

Reads of PGPDI registers are equivalent to reading the corresponding GPDI registers but significantly faster since as many as two ports can be read simultaneously with a single 32-bit read operation.

Address: Base + 0x0C40 - 0x0C54 (6 registers) Access: User read

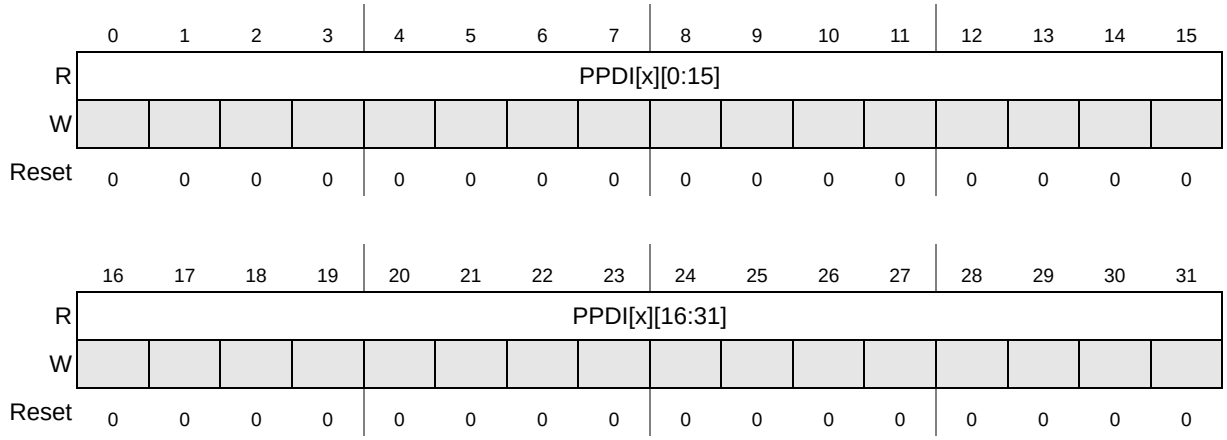



**Figure 43-17. Parallel GPIO Pad Data In Register (PGPDI0)**

**Table 43-18. PGPDI0_5 Field Descriptions**

| Field De | scription |
|---|---|
| PPDI[x] | Parallel Pad Data In<br>Read the current pad value.<br>Accesses to this register location are coherent with accesses to the bit-wise GPIO Pad Data Input Registers (GPDI0_3–GPDI184).<br>The x and bit index define which PPDI register bit is equivalent to which PDI register bit according to the equation:<br><br>PPDI[x][y] = PDI[(x*32)+y]<br><br>where x is the PGPDI register number (0..5) and y is the bit number within that register (0..31). |

**NOTE**

It is important to note the bit ordering of the ports in the parallel port registers. The most significant bit of the parallel port register corresponds to the least significant pin in the port.

### 43.5.3.15 Masked Parallel GPIO Pad Data Out Register (MPGPDO0–MPGPDO11)

The MPGPDO registers are similar in operation to the PGPDO ports described in Section 43.5.3.13, “Parallel GPIO Pad Data Out Registers (PGPDO0–PGPDO5),” but with two significant differences:

• The MPGPDO registers support masked port-wide changes to the data out on the pads of the respective port. Masking effectively allows selective bitwise writes to the full 16-bit port.

• Each 32-bit MPGPDO register is associated to only one port.

**NOTE**

The MPGPDO registers may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and will cause a transfer error response by the module. Read accesses return ‘0’.

Address: Base + 0x0C80 - 0x0CA8 (12 registers) Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | MASK[x][0:15] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | MPPDO[x][0:15] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 43-18. Masked Parallel GPIO Pad Data Out Register (MPGPDO0)**

**Table 43-19. MPGPDO0_3 field descriptions**

| Field De | scription |
|---|---|
| MASK[x][0:15] | Mask Field<br>Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location.<br>1: The associated bit value in the MPPDO[x] field is written<br>0: The associated bit value in the MPPDO[x] field is ignored |
| MPPDO[x][0:15] | Masked Parallel Pad Data Out<br>Write the data register that stores the value to be driven on the pad in output mode.<br>Accesses to this register location are coherent with accesses to the bit-wise GPIO Pad Data Output Registers (GPDO0_3 - GPDO184).<br>The x and bit index define which MPPDO register bit is equivalent to which PDO register bit according to the following equation:<br>MPPDO[x][y] = PDO[(x*16)+y] |

**NOTE**

It is important to note the bit ordering of the ports in the parallel port registers. The most significant bit of the parallel port register corresponds to the least significant pin in the port.

**WARNING**

Toggling several IOs at the same time can significantly increase the current in a pad group. Caution must be taken to avoid exceeding maximum current thresholds. Please refer to data sheet.

### 43.5.3.16 Interrupt Filter Maximum Counter Registers (IFMC0–IFMC23)

These registers are used to configure the filter counter associated with each digital glitch filter.

**NOTE**

For the pad transition to trigger an interrupt it must be steady for at least the filter period.

Address: Base + 0x1000 - 0x105C (24 registers) Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | MAXCNTx[3:0] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 43-19. Interrupt Filter Maximum Counter Registers (IFMC0 - IFMC15)**

**Table 43-20. IFMC field descriptions**

| Field De | scription |
|---|---|
| MAXCNTx [3:0] | Maximum Interrupt Filter Counter setting.<br>Filter Period = T(IRC) × (IFCP+1) × MAXCNTx + n × T(IRC) × (IFCP+1)<br>Where n is a synchronisation uncertainty between -1 and 3<br>MAXCNTx can be 3 to 15. MAXCNT < 3 causes the filter to be bypassed.<br>T(IRC): Basic Filter Clock Period: 62.5 ns (F = 16 MHz) |

### 43.5.3.17 Interrupt Filter Clock Prescaler Register (IFCPR)

This register is used to configure a clock prescaler which is used to select the clock for all digital filter counters in the SIUL.

Address: Base + 0x1080 Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | IFCP[3:0] | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 43-20. Interrupt Filter Clock Prescaler Register (IFCPR)**

Table 43-21. IFCPR Field Descriptions

| Field De | scription |
|---|---|
| IFCP [3:0] | Interrupt Filter Clock Prescaler setting<br>Prescaled Filter Clock Period = T(IRC) x (IFCP + 1)<br>T(IRC): Basic Filter Clock Period: 62.5 ns (F = 16 MHz)<br>IFCP can be 0 to 15 |

## 43.6 Functional description

### 43.6.1 General

This section provides a functional description of the System Integration Unit Lite.

### 43.6.2 Pad control

The SIUL controls the configuration and electrical characteristic of the device pads. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The pad configuration registers (PCR*n*, see Section 43.5.3.8, “Pad Configuration Registers (PCR0–PCR184)”) allow software control of the static electrical characteristics of external pins with a single write. These are used to configure the following pad features:

- Open drain output enable
- Slew rate control
- Pull control
- Pad assignment
- Control of analog path switches
- Safe mode behavior configuration

### 43.6.3 General purpose input and output pads (GPIO)

The SIUL allows each pad to be configured as either a General Purpose Input Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is normally determined by the peripheral that will use the pad.

The SIUL manages 185 GPIO pads organized as ports that can be accessed for data reads and writes as 32-bit, 16-bit or 8-bit.

As shown in Figure 43-21, all port accesses are identical with each read or write being performed only at a different location to access a different port width.

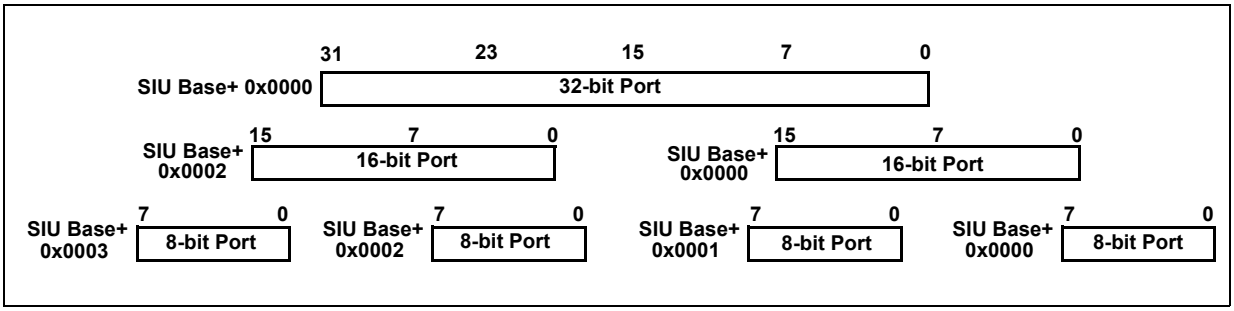



**Figure 43-21. Data Port example arrangement showing configuration for different port width accesses**

This implementation requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The SIUL has separate data input (GPDI*n_n*, see Section 43.5.3.12, “GPIO Pad Data Input Registers (GPDI0_3–GPDI184)”) and data output (GPDO*n_n*, see Section 43.5.3.11, “GPIO Pad Data Output Registers (GPDO0_3 - GPDO184)”) registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

Data output registers allow an output pad to be driven high or low (with the option of push-pull or open drain drive). Input registers are read-only and reflect the respective pad value.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non GPIO), this write will not be reflected by the pad value until reconfigured to GPIO.

The allocation of what input function is connected to the pin is defined by the PSMI registers (PCR*n*, see Section 43.5.3.8, “Pad Configuration Registers (PCR0–PCR184)”).

## 43.6.4 External interrupts

The SIUL supports 24 external interrupts, EIRQ0–EIRQ23. The mapping of these interrupts to external pins is summarized in Table 43-1 and described in full detail in Chapter 3, “Signal Description.”

The SIUL supports three interrupt vectors to the interrupt controller. Each vector interrupt has eight external interrupts combined together with the presence of flag generating an interrupt for that vector if enabled. All of the external interrupt pads within a single group have equal priority.

Refer to Figure 43-22 for an overview of the External Interrupt implementation.

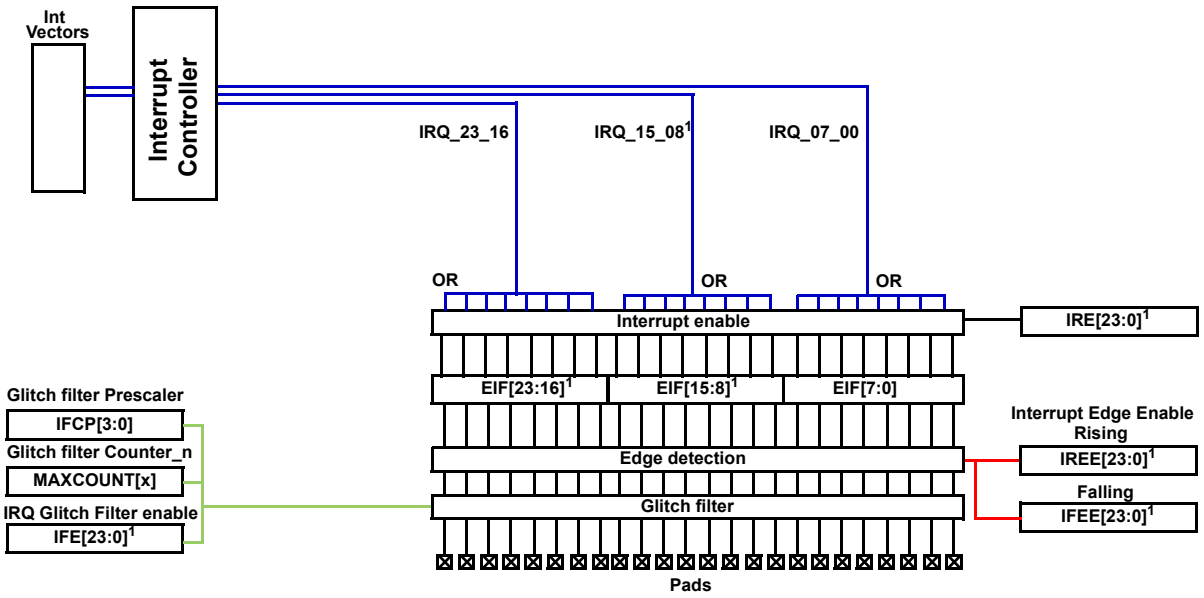



**Figure 43-22. External interrupt pad diagram**

[1] Bit count is [0:15] in the 176-pin package, [0:18] in the 208-pin package, and [0:23] in the 416-pin package.

### 43.6.4.1 External interrupt management

Each interrupt can be enabled or disabled independently. This can be performed using the Interrupt Request Enable Register (IRER - Section 43.5.3.4, “Interrupt Request Enable Register (IRER)”). A pad defined as an external interrupt can be configured to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled is reserved and should not be configured. External interrupts require that the associated input buffer for the pad is enabled (PCR[IBE]=1).

The active EIRQ edge is controlled through the configuration of the registers IREER and IFEER.

Each external interrupt supports an individual flag which is held in the Flag register (ISR - Section 43.5.3.3, “Interrupt Status Flag Register (ISR)”). This register is a c lear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

## 43.7 Pin muxing

For pin muxing, please refer to Chapter 3, “Signal Description,” of this document.

# Chapter 44
# System Status and Configuration Module (SSCM)

## 44.1 Introduction

### 44.1.1 Overview

The System Status and Configuration Module (SSCM), pictured in Figure 44-1, provides central SOC functionality. On devices with a separate Standby power domain, the System Status block is part of that domain.

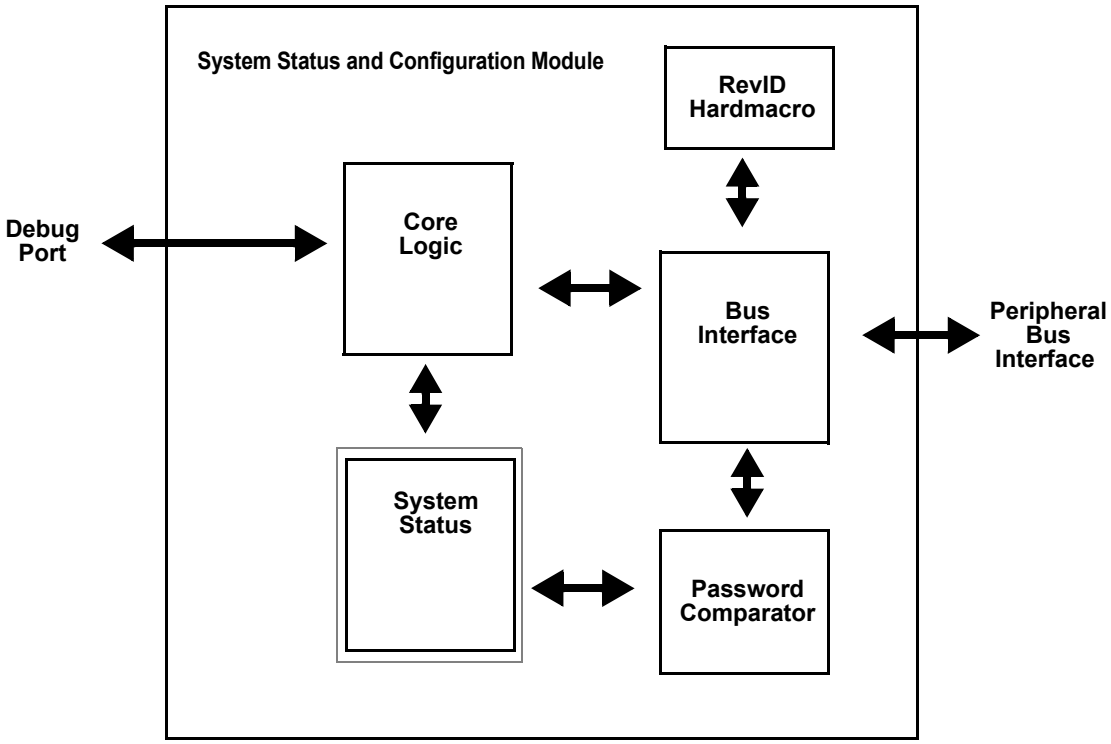



**Figure 44-1. System Status and Configuration Module Block Diagram**

### 44.1.2 Features

The SSCM includes these distinctive features:

- System Configuration and Status
  - Memory sizes/status
  - Device Mode and Security Status
  - Determine boot vector
  - Search Code Flash for bootable sector
- Device identification information (MCU ID Registers)

- Debug Status Port enable and selection
- Bus and peripheral abort enable/disable

### 44.1.3 Modes of operation

The SSCM operates identically in all system modes.

## 44.2 Memory map and register description

This section provides a detailed description of all memory-mapped registers in the SSCM.

### 44.2.1 Memory map

Table 44-1 shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

**Table 44-1. SSCM memory map**

| Address | Register | Location |
|---|---|---|
| Base + 0x0000 | System Status (STATUS) | on page 1508 |
| Base + 0x0002 | System Memory Configuration (MEMCONFIG) | on page 1509 |
| Base + 0x0004 | Reserved | |
| Base + 0x0006 | Error Configuration (ERROR) | on page 1510 |
| Base + 0x0008 | Debug Status Port (DEBUGPORT) | on page 1511 |
| Base + 0x000A | Reserved | |
| Base + 0x000C | Password Comparison Register High Word | on page 1512 |
| Base + 0x0010 | Password Comparison Register Low Word | on page 1512 |

All registers are accessible via 8-bit, 16-bit or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit READ/WRITE to address ‘Base + 0x0002’, but performing a 16-bit access to ‘Base + 0x0003’ is illegal.

### 44.2.2 Register description

The following memory-mapped registers are available in the SSCM. Those bits that are shaded out are reserved for future use. To optimize future compatibility, these bits should be masked out during any read/write operations to avoid conflict with future revisions.

#### 44.2.2.1 System Status Register (STATUS)

The System Status register is a read-only register that reflects the current state of the system.

Address: Base + 0x0000 Access: Read Only

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | NXEN | PUB | SEC | 0 | BMODE | | | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0/1 | 0 | 0 | 0 | 0 | 0 |

= Reserved

**Figure 44-2. Status (STATUS) Register**

**Table 44-2. STATUS Allowed Register Accesses**

| | 8-bit | 16-bit | 32-bit[1] |
|---|---|---|---|
| READ | Allowed | Allowed | Allowed |
| WRITE | Not Allowed | Not Allowed | Not Allowed |

1 All 32-bit accesses must be aligned to 32-bit addresses (i.e. 0x0, 0x4, 0x8 or 0xC).

**Table 44-3. STATUS field descriptions**

| Field | Description |
|---|---|
| NXEN | Nexus enabled. |
| PUB | Public Serial Access Status. This bit indicates whether serial boot mode with public password is allowed.<br>1 Serial boot mode with public password is allowed<br>0 Serial boot mode with private Flash password is allowed, provided the key hasn't been swallowed |
| SEC | Security Status. This bit reflects the current security state of the Flash.<br>1 The Flash is secured<br>0 The Flash is not secured |
| BMODE | Device Boot Mode.<br>000 reserved for FlexRay Boot Serial Boot Loader<br>001 legacy bootstrap via CAN<br>010 legacy bootstrap via UART<br>011 Single Chip<br>100 - 111 Reserved<br>This field is only updated during reset. |

### 44.2.2.2 System Memory Configuration Register

The System Memory Configuration register is a read-only register that reflects the memory configuration of the system.

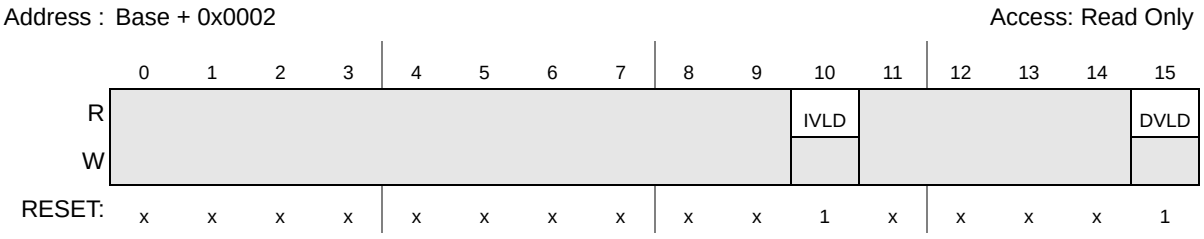



**Figure 44-3. System Memory Configuration (MEMCONFIG) Register**

**Table 44-4. MEMCONFIG Field Descriptions**

| Field | Description |
|---|---|
| 10<br>IVLD | Instruction Flash Valid. This bit identifies whether or not the on-chip Instruction Flash is accessible in the system memory map. The Flash may not be accessible due to security limitations, or because there is no Flash in the system.<br>1 Instruction Flash is accessible<br>0 Instruction Flash is not accessible |
| 15<br>DVLD | Data Flash Valid. This bit identifies whether or not the on-chip Data Flash is visible in the system memory map. The Flash may not be accessible due to security limitations, or because there is no Flash in the system.<br>1 Data Flash is visible<br>0 Data Flash is not visible |

**Table 44-5. MEMCONFIG Allowed Register Accesses**

| | 8-bit | 16-bit | 32-bit |
|---|---|---|---|
| READ | Allowed | Allowed | Allowed<br>(also reads STATUS register) |
| WRITE | Not Allowed | Not Allowed | Not Allowed |

### 44.2.2.3 Error Configuration

The Error Configuration register is a read-write register that controls the error handling of the system.

Address : Base + 0x0006 Access: Read/Write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PAE | RAE |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Reserved

**Figure 44-4. Error Configuration (ERROR) Register**

**Table 44-6. ERROR Field Descriptions**

| Field | Description |
|---|---|
| 14<br>PAE | Peripheral Bus Abort Enable. This bit enables bus aborts on any access to a peripheral slot that is not used on the device. This feature is intended to aid in debugging when developing application code.<br>1 Illegal accesses to non-existing peripherals produce a Prefetch or Data Abort exception<br>0 Illegal accesses to non-existing peripherals do not produce a Prefetch or Data Abort exception |
| 15<br>RAE | Register Bus Abort Enable. This bit enables bus aborts on illegal accesses to off-platform peripherals. Illegal accesses are defined as reads or writes to reserved addresses within the address space for a particular peripheral. This feature is intended to aid in debugging when developing application code.<br>1 Illegal accesses to peripherals produce a Prefetch or Data Abort exception<br>0 Illegal accesses to peripherals do not produce a Prefetch or Data Abort exception<br>Transfers to Peripheral Bus resources may be aborted even before they reach the Peripheral Bus (i.e. at the AIPS level). In this case, the **PER_ABORT** and **REG_ABORT** register bits will have no effect on the abort. |

**Table 44-7. ERROR Allowed Register Accesses**

| | 8-bit | 16-bit | 32-bit |
|---|---|---|---|
| READ | Allowed | Allowed | Allowed |
| WRITE | Allowed | Allowed | Not Allowed |

### 44.2.2.4 Debug Status Port Register

The Debug Status Port register is used to (optionally) provide debug data on a set of pins.

Address: Base + 0x0008 Access: Read/Write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DEBUG_MODE[2:0]] | | |
| W | | | | | | | | | | | | | | | | |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Reserved for future use

**Figure 44-5. Debug Status Port (DEBUGPORT) Register**

**Table 44-8. DEBUGPORT Field Descriptions**

| Field | Description |
|---|---|
| 13-15<br>DEBUG<br>_MODE<br>[0:2] | Debug Status Port Mode. This field selects the alternate debug functionality for the Debug Status Port<br>000 No alternate functionality selected<br>001 Mode 1 Selected<br>010 Mode 2 Selected<br>011 Mode 3 Selected<br>100 Mode 4 Selected<br>101 Mode 5 Selected<br>110 Mode 6 Selected<br>111 Mode 7 Selected<br>Table 44-9 describes the functionality of the Debug Status Port in each mode. |

**Table 44-9. Debug Status Port Modes**

| Pin[1] | Mode 1 | Mode 2 | Mode 3 | Mode 4 | Mode 5 | Mode 6 | Mode 7 |
|---|---|---|---|---|---|---|---|
| 0 | STATUS[0] | STATUS[8] | MEMCONFIG[0] | MEMCONFIG[8] | Reserved | Reserved | Reserved |
| 1 | STATUS[1] | STATUS[9] | MEMCONFIG[1] | MEMCONFIG[9] | Reserved | Reserved | Reserved |
| 2 | STATUS[2] | STATUS[10] | MEMCONFIG[2] | MEMCONFIG[10] | Reserved | Reserved | Reserved |
| 3 | STATUS[3] | STATUS[11] | MEMCONFIG[3] | MEMCONFIG[11] | Reserved | Reserved | Reserved |
| 4 | STATUS[4] | STATUS[12] | MEMCONFIG[4] | MEMCONFIG[12] | Reserved | Reserved | Reserved |
| 5 | STATUS[5] | STATUS[13] | MEMCONFIG[5] | MEMCONFIG[13] | Reserved | Reserved | Reserved |
| 6 | STATUS[6] | STATUS[14] | MEMCONFIG[6] | MEMCONFIG[14] | Reserved | Reserved | Reserved |
| 7 | STATUS[7] | STATUS[15] | MEMCONFIG[7] | MEMCONFIG[15] | Reserved | Reserved | Reserved |

1 All signals are active high, unless otherwise noted

**Table 44-10. DEBUGPORT Allowed Register Accesses**

| | 8-bit | 16-bit | 32-bit[1] |
|---|---|---|---|
| READ | Allowed | Allowed | Not Allowed |
| WRITE | Allowed | Allowed | Not Allowed |

1 All 32-bit accesses must be aligned to 32-bit addresses (i.e. 0x0, 0x4, 0x8 or 0xC).

### 44.2.2.5 Password Comparison Registers

These registers allow to unsecure the device, if the correct password is known.

Address: 0x000C Access: Read/Write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | PWD_HI[0:15] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | PWD_HI[16:31] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 44-6. Password Comparison Register High Word (PWCMPH)**

Address: 0x0010 Access: Read/Write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | PWD_LO[0:15] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | PWD_LO[16:31] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 44-7. Password Comparison Register Low Word (PWCMPL)**

**Table 44-11. Password Comparison Register Field Descriptions**

| Field | Description |
|---|---|
| 0-31<br>PWD_HI<br>[0:31] | Upper 32 bits of the password |
| 0-31<br>PWD_LO<br>[0:31] | Lower 32 bits of the password |

**Table 44-12. PWCMPH/L Allowed Register Accesses**

| | 8-bit | 16-bit | 32-bit[1] |
|---|---|---|---|
| READ | Allowed | Allowed | Allowed |
| WRITE | Not Allowed | Not Allowed | Allowed |

1 All 32-bit accesses must be aligned to 32-bit addresses (i.e. 0x0, 0x4, 0x8 or 0xC).

In order to unsecure the device, the password needs to be written as follows: first the upper word to the PWCMPH register, then the lower word to the PWCMPL register. The SSCM compares the password and if the password is correct, unlocks the device.

## 44.3 Functional Description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system. For convenience the description of the boot vector selection is contained in Chapter 6, "Boot Assist Module (BAM)." This allows all of the reset behavior to be described in a single place.

## 44.4 Initialization/Application Information

### 44.4.1 Reset

The reset state of each individual bit is shown within the Section 44.2.2, "Register description."

# Chapter 45
# System Timer Module (STM)

## 45.1 Introduction

### 45.1.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

### 45.1.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

### 45.1.3 Modes of operation

The STM supports two device modes of operation: normal and debug. When the STM is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the STM_CR register. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run.

## 45.2 External signal description

The STM does not have any external interface signals.

## 45.3 Memory map and register definition

The STM programming model has fourteen 32-bit registers. The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

### 45.3.1 Memory map

The STM memory map is shown in Table 45-1.

**Table 45-1. STM memory map**

| Address offset | Register | Location |
|---|---|---|
| 0x0000 | STM Control Register (STM_CR) | on page 1516 |
| 0x0004 | STM Counter Value (STM_CNT) | on page 1517 |
| 0x0008 | Reserved | |
| 0x000C | Reserved | |
| 0x0010 | STM Channel 0 Control Register (STM_CCR0) | on page 1518 |
| 0x0014 | STM Channel 0 Interrupt Register (STM_CIR0) | on page 1518 |
| 0x0018 | STM Channel 0 Compare Register (STM_CMP0) | on page 1519 |
| 0x001C | Reserved | |
| 0x0020 | STM Channel 1 Control Register (STM_CCR1) | on page 1518 |
| 0x0024 | STM Channel 1 Interrupt Register (STM_CIR1) | on page 1518 |
| 0x0028 | STM Channel 1 Compare Register (STM_CMP1) | on page 1519 |
| 0x002C | Reserved | |
| 0x0030 | STM Channel 2 Control Register (STM_CCR2) | on page 1518 |
| 0x0034 | STM Channel 2 Interrupt Register (STM_CIR2) | on page 1518 |
| 0x0038 | STM Channel 2 Compare Register (STM_CMP2) | on page 1519 |
| 0x003C | Reserved | |
| 0x0040 | STM Channel 3 Control Register (STM_CCR3) | on page 1518 |
| 0x0044 | STM Channel 3 Interrupt Register (STM_CIR3) | on page 1518 |
| 0x0048 | STM Channel 3 Compare Register (STM_CMP3) | on page 1519 |
| 0x004C - 0x3FFF | Reserved | |

## 45.3.2 Register descriptions

The following sections detail the individual registers within the STM programming model.

Figure 45-1 shows the conventions used in the register figures.

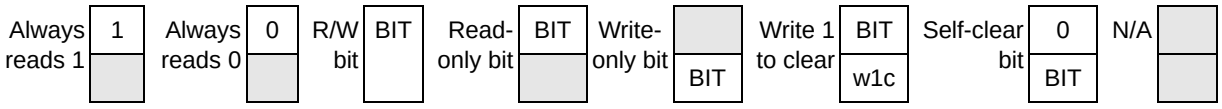



**Figure 45-1. Key to Register Fields**

### 45.3.2.1 STM Control Register (STM_CR)

The STM Control Register (STM_CR) includes the prescale value, freeze control and timer enable bits.

Offset: 0x000 Access: Read/Write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CPS | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | FRZ | TEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 45-2. STM Control Register (STM_CR)**

**Table 45-2. STM_CR field descriptions**

| Field | Description |
|---|---|
| 16:23<br>CPS | Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256).<br>0x00 Divide system clock by 1<br>0x01 Divide system clock by 2<br>...<br>0xFF Divide system clock by 256 |
| 30<br>FRZ | Freeze. Allows the timer counter to be stopped when the device enters debug mode.<br>0 STM counter continues to run in debug mode.<br>1 STM counter is stopped in debug mode. |
| 31<br>TEN | Timer Counter Enabled.<br>0 Counter is disabled.<br>1 Counter is enabled. |

### 45.3.2.2 STM Count Register (STM_CNT)

The STM Count Register (STM_CNT) holds the timer count value.

Offset: 0x004 Access: Read/Write

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

**Figure 45-3. STM Count Register (STM_CNT)**

**Table 45-3. STM_CNT field descriptions**

| Field | Description |
|---|---|
| 0:31<br>CNT | Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the system clock divided by the prescale value. |

### 45.3.2.3 STM Channel Control Register (STM_CCR*n*)

The STM Channel Control Register (STM_CCR*n*) has the enable bit for channel n of the timer.

Offset: 0x10+0x10*n — Access: Read/Write

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

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

**Figure 45-4. STM Channel Control Register (STM_CCR*n*)**

**Table 45-4. STM_CCR*n* field descriptions**

| Field | Description |
|---|---|
| 31<br>CEN | Channel Enable.<br>0 The channel is disabled.<br>1 The channel is enabled. |

### 45.3.2.4 STM Channel Interrupt Register (STM_CIR*n*)

The STM Channel Interrupt Register (STM_CIR*n*) has the interrupt flag for channel n of the timer.

Offset: 0x14+0x10*n — Access: Read/Write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CIF |
| W | | | | | | | | | | | | | | | | w1c |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 45-5. STM Channel Interrupt Register (STM_CIR*n*)**

**Table 45-5. STM_CIR*n* field descriptions**

| Field | Description |
|---|---|
| 31<br>CIF | Channel Interrupt Flag<br>0 No interrupt request.<br>1 Interrupt request due to a match on the channel. |

### 45.3.2.5 STM Channel Compare Register (STM_CMP*n*)

The STM channel compare register (STM_CMP*n*) holds the compare value for channel n.

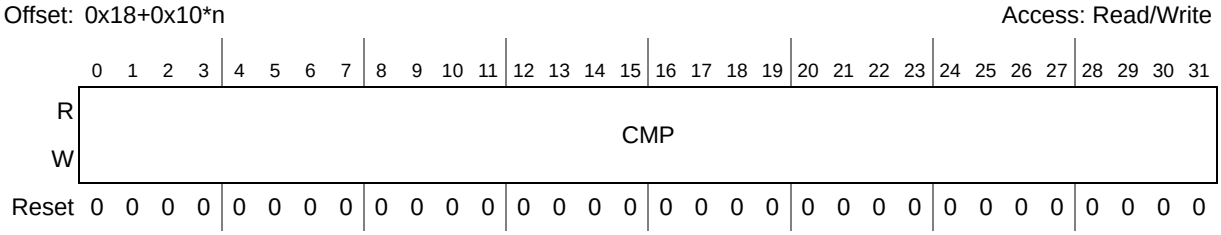



**Figure 45-6. STM Channel Compare Register (STM_CMP*n*)**

**Table 45-6. STM_CMP*n* field descriptions**

| Field | Description |
|---|---|
| 0:31<br>CMP | Compare value for channel n. If the STM_CCR*n*[CEN] bit is set and the STM_CMP*n* register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIR*n*[CIF] bit is set. |

## 45.4 Functional description

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel.

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the system clock frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode the counter continuously increments. When enabled in debug mode the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode, otherwise it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCR*n*), a channel interrupt register (STM_CIR*n*) and a channel compare register (STM_CMP*n*). The channel is enabled by setting the STM_CCR*n*[CEN] bit. When enabled, the channel will set the STM_CIR*n*[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing a 1 to the STM_CIR*n*[CIF] bit. A write of 0 to the STM_CIR*n*[CIF] bit has no effect.

**NOTE**

The STM counter does not advance when the system clock is stopped.

This page is intentionally left blank.

# Chapter 46
# Timing Controller (TCON)

## 46.1 Introduction

The Timing Controller module (TCON) provides an alternative interface for the DCU3 that provides RGB data and timing signals for “raw” TFT panels which have no embedded TCON. The TCON drives the panel via a RSDS (Reduced Swing Differential Signalling) interface (refer to RSDS™ “Intra-panel“ interface specification, revision 1.0, National Semiconductor). The TCON and associated RSDS I/O module enable direct drive of the row and column drivers of display panels enabling emulation of TCON ICs used in display panels.

Figure 46-1 shows a typical application diagram of the TCON.

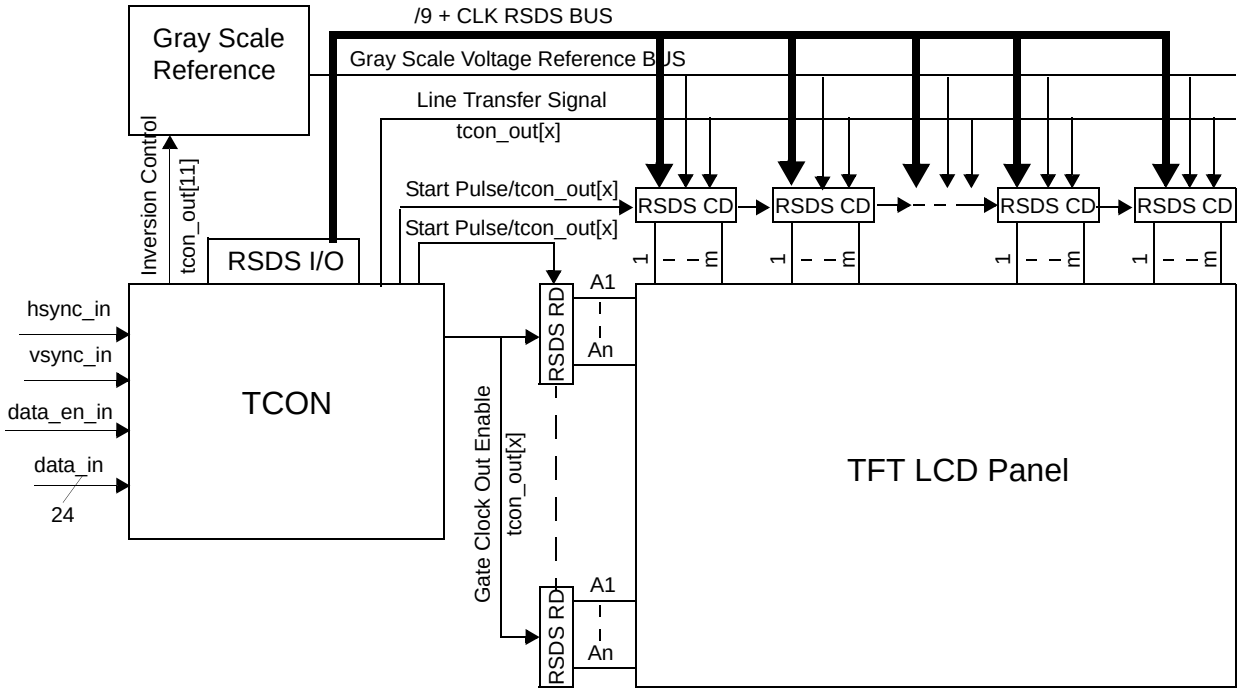



**Figure 46-1. Application diagram of the TCON**

### 46.1.1 Features

- Flexible timing generation unit supporting 12 timing signal channels
- Conforms to RSDS™ “Intra-panel“ interface specification, revision 1.0 (National Semiconductor)
- Flexible mapping of RGB data to RSDS channels
- Support bit mapping or 8-bit or 6-bit color depth
- Output data signal transition minimization through data inversion control
- Blanking of RGB data during inactive period (driven to all “0” or all “1”)
- Flexible RSDS output setup/hold time control in granulatiry of 1/2 system clock cycle

### 46.1.2 Modes of operation

The TCON has 3 operation modes:

- Bypass mode: the input signals are passed through, and the TCON shall be turned off
- TTL mode: the TCON is functional, driving parallel RGB output in non-RSDS mode, and the TCON timing signals
- RSDS mode: the TCON is functional, driving the RSDS interface and timing signals

## 46.2 External signal descriptions

The TCON has the following external signals:

**Table 46-1. TCON External Signals**

| Signal | Description | I/O | Reset |
|---|---|---|---|
| data_out[25:0][1] | pixel data and clock out, can be configured to be single-ended TTL output or RSDS differential output | O | 0 |
| tcon_out[11:0] | 12 tcon timing signals | O | 0 |
| rsds_mode[2] | RSDS/TTL format directive | O | 0 |

1 These signals output through the RSDS I/O interface

2 This signal goes to the RSDS I/O only

## 46.3 Memory map and register definition

### 46.3.1 Memory map

Figure 46-2 shows the register memory map of the TCON.

**Table 46-2. TCON memory map**

| Address Offset | Register | Access | Location |
|---|---|---|---|
| 0x0000 | (TCON_CTRL1) — TCON control1 register | R/W | on page 1526 |
| 0x0004 | (TCON_BMC) — Bit map control register | R/W | on page 1528 |
| 0x0008-0x0014 | (TCON_COMP0-TCON_COMP3)<br>- comparator 0/1/2/3 configure register | R/W | on page 1529 |
| 0x0018-0x0024 | (TCON_COMP0_MSK - TCON_CONP3_MSK)<br>- comparator 0/1/2/3 compare value mask register | R/W | on page 1530 |
| 0x0028-0x003C | (TCON_PULSE0 - TCON_PULSE5)<br>- pulse 0/1/2/3/4/5 configure register | R/W | on page 1531 |
| 0x0040-0x0054 | (TCON_PULSE0_MSK - TCON_PULSE5_MSK)<br>- pulse 0/1/2/3/4/5 compare value mask register | R/W | on page 1532 |
| 0x0058-0x008C | (TCON_SMX0 - TCON_SMX13)<br>- SMX0 - SMX13 function control register | R/W | on page 1532 |

Table 46-2. TCON memory map (continued)

| Address Offset | Register | Access | Location |
|---|---|---|---|
| 0x0090 | (TCON_OMUX_LOW) - TCON output mux control low | R/W | on page 1535 |
| 0x0094 | (TCON_OMUX_HIGH) - TCON output mux control high | R/W | on page 1536 |
| 0x0098-0x00CC | (TCON_LUT0 - TCON_LUT13) - TCON look up table 0 - 13 | R/W | on page 1538 |
| 0x00D0-0x0100 | (TCON_DATA0_DLY - TCON_DATA12_DLY) - RSDS I/O 0 - 12 control | R/W | on page 1539 |
| 0x0104 | (TCON_CTRL2) -- TCON control2 register | R/W | on page 1542 |

## 46.3.2 Register summary

Table 46-3. TCON register summary

| Name | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TCON_CTRL1 0x0000 | R | TCON_EN | 0 | TCON_BYPASS | data_inv_en | TCON11_INV | TCON10_INV | TCON9_INV | TCON8_INV | TCON7_INV | TCON6_INV | TCON5_INV | TCON4_INV | TCON3_INV | TCON2_INV | TCON1_INV | TCON0_INV |
| | W | | | | | | | | | | | | | | | | |
| | R | INIT_DELAY | | | V_REF_SEL | | | H_REF_SEL | | VLEN | | HSYNC_INV | VSYNC_INV | COLOR_DEPTH | RGB_PADDING_EN | RGB_PADDING | RSDS_MODE |
| | W | | | | | | | | | | | | | | | | |
| TCON_BMC 0x0004 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | CLK_POS | | | | | COLOR_ ORDER | | | BIT_ORDER | BIT_SWAP |
| | W | | | | | | | | | | | | | | | | |
| TCON_COMPx (0x0008-0x0014 ) | R | FUNC _SEL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | COMP_VALUE | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 46-3. TCON register summary (continued)**

| Name | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TCON_COMPx_MASK (0x0018-0x0024) | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | MASK | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| TCON_PULSEx (0x0028-0x003C) | R | FUNC_SEL | | 0 | 0 | SET | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | COMPARATOR_SEL | | 0 | 0 | RESET | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| TCON_PULSEx_MSK (0x0040-0x0054) | R | 0 | 0 | 0 | 0 | SET_MASK | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | RESET_MASK | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| TCON_SMXx (0x0058-0x008C) | R | INDEX3_SEL | | INDEX2_SEL | | INDEX1_SEL | | INDEX0_SEL | | | | | | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | Y_SEL | | | | | X_SEL | | | | |
| | W | | | | | | | | | | | | | | | | |
| TCON_OMUX_LOW 0x0090 | R | 0 | 0 | TCON5 | | | | TCON4 | | | | | TCON3 | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | TCON 3[4:0] | | TCON2 | | | | TCON1 | | | | | TCON0 | | | | |
| | W | | | | | | | | | | | | | | | | |
| TCON_OMUX_HIGH 0x0094 | R | 0 | 0 | TCON11 | | | | TCON10 | | | | | TCON9 | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | TCON 9[4:0] | | TCON8 | | | | TCON7 | | | | | TCON6 | | | | |
| | W | | | | | | | | | | | | | | | | |
| TCON_LUTx (0x0098-0x00CC) | R | LUT[31:16] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | LUT[15:0] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

Table 46-3. TCON register summary (continued)

| Name | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TCON_DATAx_DLY (0x00D0-0x0100) | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |
| | R | POL | 0 | 0 | OSUFINE2 | OSUFINE1 | OSUCRSE2 | OSUCRSE1 | OSDFINE | SKEWOPT | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| TCON_CTRL2 0x0104 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLK_OFFSET | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DIV_RATIO | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

## 46.3.3 Register descriptions

This section contains the detailed descriptions of TCON registers.

### 46.3.3.1 Control Register 1 (TCON_CTRL1)

Offset 0x000
Access: User read/write

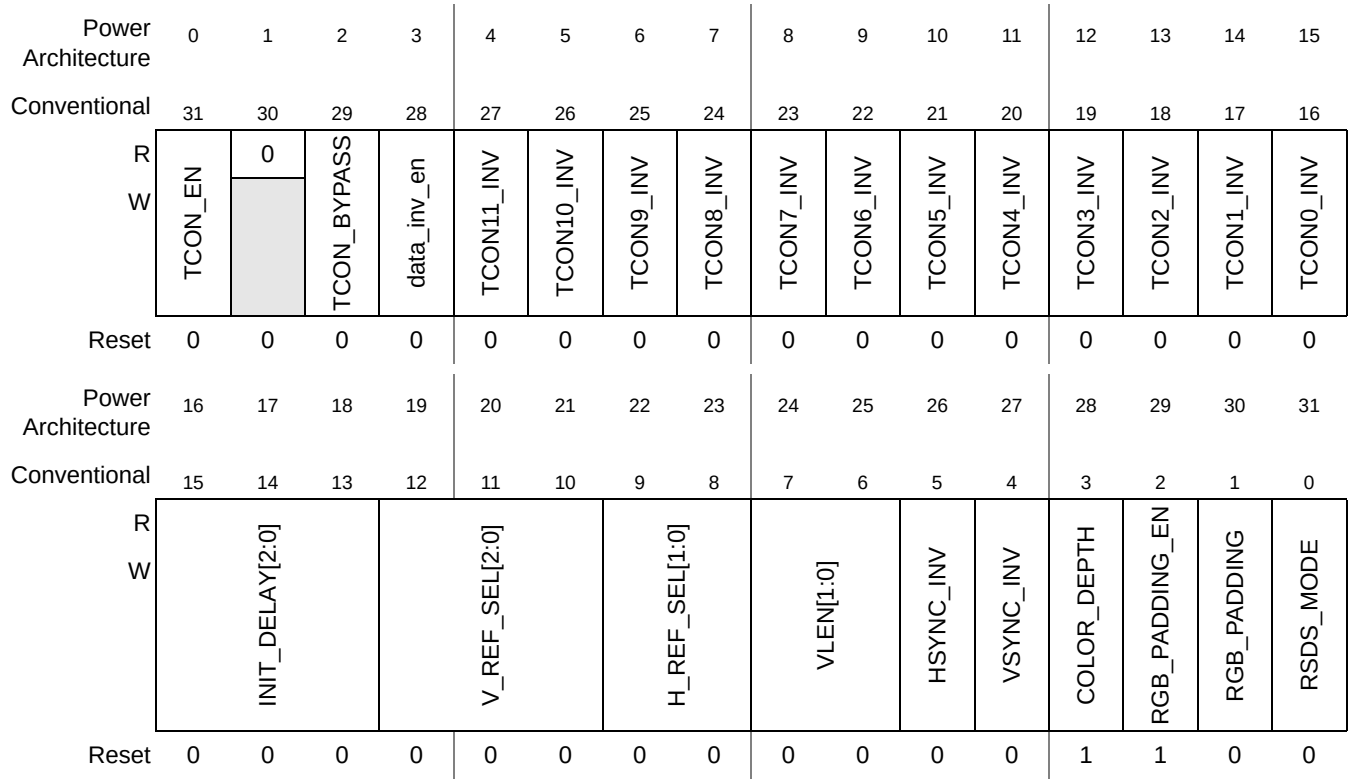



**Figure 46-2. TCON_CTRL1 register**

**Table 46-4. TCON_CTRL1 field descriptions**

| Field | Description |
|---|---|
| TCON_EN | 1: enable TCON.<br>0: Disable TCON. |
| TCON_BYPASS | 1: Bypass TCON, both data and timing signals will pass through the TCON unmodified. see 46.4.1.3/46-1543 for pin mapping in that mode.<br>0: Not bypass TCON. State of the TCON is decided by TCON_EN. |
| HSYNC_INV | 1: hsync_in signal is active low<br>0: hsync_in signal is active high |
| VSYNC_INV | 1: vsync_in signal is active low<br>0: vsync_in signal is active high |
| TCON11_INV - TCON0_INV | TCONx output inversion control<br>0: do not invert tconx<br>1: invert output tconx |
| INIT_DELAY | Initialization delay, set the number of frames before the TCON starts to output timing signals. |

**Table 46-4. TCON_CTRL1 field descriptions (continued)**

| Field | Description |
|---|---|
| V_REF_SEL | v_ref selector, see Section 46.4.2.3, “Toggle generator for use of v_ref signal.<br>0: select tcon_pulse0 output as v_ref<br>1: select tcon_pulse1 output as v_ref<br>2: select tcon_pulse2 output as v_ref<br>3: select tcon_pulse3 output as v_ref<br>4: select tcon_pulse4 output as v_ref<br>5: select tcon_pulse5 output as v_ref<br>6: select tcon_pulse0 output as v_ref, while reset vtgl_counter and vtgl[3] to 0 at rising edge of v_ref.<br>7: select tcon_pulse0 output as v_ref, while set vtgl_counter to 1 and reset vtgl[3] to 0 at rising edge of v_ref.<br>Note: pulse generated must be vertical based for the toggle generator. |
| H_REF_SEL | h_ref selector, see Section 46.4.2.3, “Toggle generator for use of h_ref signal.<br>0: select tcon_comp0 output as h_ref.<br>1: select tcon_comp1 output as h_ref.<br>2: select tcon_comp2 output as h_ref.<br>3: select tcon_comp3 output as h_ref. |
| VLEN | maximum counter value for vtgl_counter in the toggle generator. see Section 46.4.2.3, “Toggle generator for detailed functionality of this field. |
| DATA_INV_EN | 1: Enable output data inversion<br>The input pixel (n) is compared with pixel (n-1). If more than half of the RGB data bits toggle, the pixel (n) output will be inverted and the tcon_out[11] flags this data inversion to the gray scale reference. tcon_out[11] =1 -> data_out is inverted, tcou_out[11] =0 -> data_out is not inverted. Intention of this feature is for board level EMI reduction<br>0: Disable output data inversion |
| COLOR_DEPTH | Color depth of each color component<br>1: 8 bits<br>0: 6 bits, the 2 LSB’s are set to 2’b0 |
| RGB_PADDING_EN | RGB data padding enable<br>1: enable padding during blanking<br>0: disable padding during blanking |
| RGB_PADDING | RGB data driven during blanking.<br>1: all “1”<br>0: all “0“ |
| RSDS_MODE | 1: RSDS mode, output RSDS bit mapping<br>0: TTL mode, output TTL bit mapping |

### 46.3.3.2 Bit Mapping Control (TCON_BMC)

Offset 0x0004

Access: User read/write

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

| Power Architecture | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Conventional | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | 0 | 0 | 0 | 0 | 0 | 0 | CLK_POS | | | | | COLOR_ORDER | | | BIT_ORDER | BIT_SWAP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 46-3. TCON_BMC Register**

**Table 46-5. TCON_BMC Register Field Descriptions**

| Field | Description |
|---|---|
| CLK_POS | clock position selection, output pixel clock can be assigned to any data_out pins.<br>For RSDS mode, see Table 46-31 for detailed mapping relationship.<br>For TTL mode, see Table 46-32 for detailed mapping relationship. |
| COLOR_ORDER | Color component order configuration bits.<br>000: RGB<br>001: BRG<br>010: GBR<br>011: RBG<br>100: GRB<br>101: BGR<br>Other values:are reserved.for use |
| BIT_ORDER | 0: MSB 7 down to LSB 0 for every color component<br>1: LSB 0 upto MSB 7, for every color component. (inverted order) |
| BIT_SWAP | 0: No swap<br>1: Swap odd and even bits for every color component: bit 6 and 7, 4 and 5, 2 and 3, 0 and 1 are swapped. This is needed for RSDS channel order inversion |

### 46.3.3.3 TCON_COMP0 - TCON_COMP3

Offset 0x0008 - 0x0014

Access: User read/write

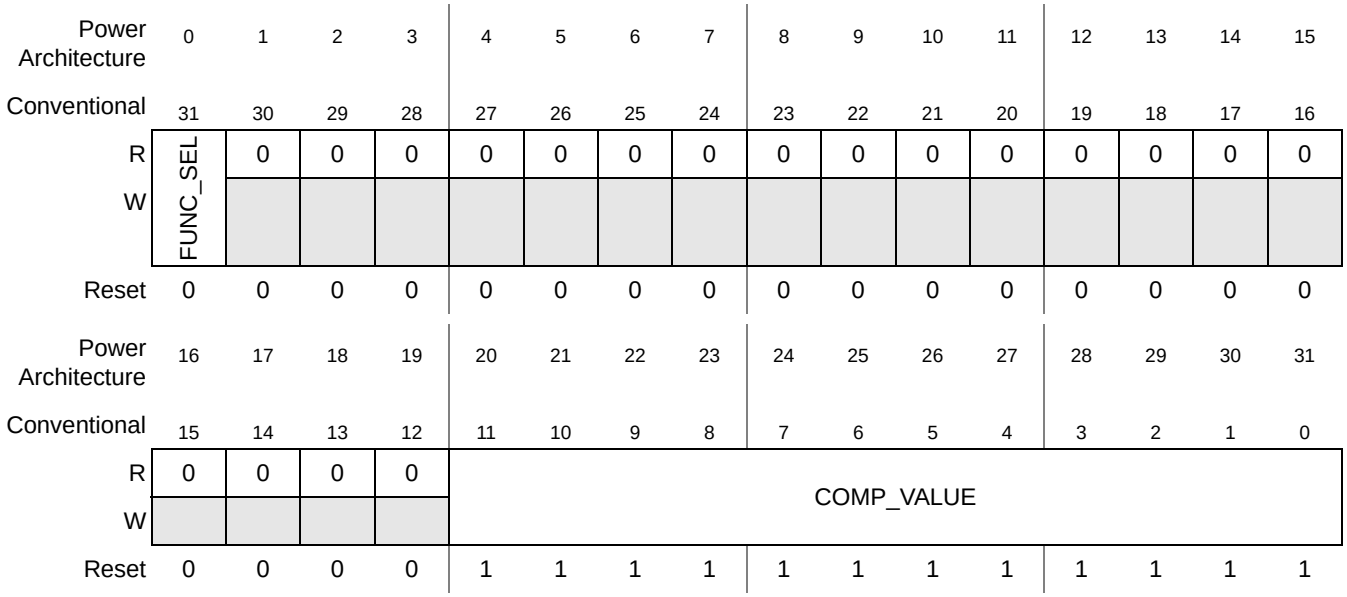



**Figure 46-4. TCON_COMPx Register**

**Table 46-6. TCON_COMPx Register Field Descriptions**

| Field | Description |
|---|---|
| FUNC_SEL | On which direction the comparation is based, vertical or horizontal<br>0: compare on horizontal direction<br>1: compare on vertical direction |
| COMP_VALUE | Comparison value.<br>FUNC_SEL=0 -> A one-pixel clock cycle high pulse is generated when h_count = COMP_VALUE + 2;<br>FUNC_SEL=1 -> A one-line wide high pulse is generated when v_count = COMP_VALUE.<br>See Section 46.4.2.1, “Comparator for detailed explanation |

### 46.3.3.4 TCON_COMP0_MSK - TCON_COMP3_MSK

Offset 0x0018 - 0x0024

Access: User read/write

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

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

**Figure 46-5. TCON_COMPx_MSK Register**

**Table 46-7. TCON_COMPx_MSK Register Field Descriptions**

| Field | Description |
|---|---|
| MASK | Comparison value mask:<br>MASK[x] = 1: include bit x in comparator matching<br>MASK[x] = 0 ignore bit x in comparator matching |

### 46.3.3.5 TCON_PULSE0 - TCON_PULSE5

Offset 0x0028 - 0x003C

Access: User read/write

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Power Architecture | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Conventional | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| R | FUNC_SEL | | 0 | 0 | SET | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Power Architecture | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Conventional | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | COMPARATOR_SEL | | 0 | 0 | RESET | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 46-6. TCON_PULSEx Register**

**Table 46-8. TCON_PULSEx Register Field Descriptions**

| Field | Description |
|---|---|
| FUNC_SEL | Select the type of pulse, horizontal/vertical/or mix<br>00: SET/RESET value both for horizontal compare<br>01: SET/RESET value both for vertical compare, signal transition point determined by COMPARATOR_SEL<br>10: SET/RESET value both for vertical compare, signal transitions at the beginning of the line which is matched.<br>11: SET value for horizontal compare, RESET value for vertical compare. Signal transition point for RESET determined by COMPARATOR_SEL. |
| COMPARATOR_SEL | When FUNC_SEL is set to 01, 1 of the 4 comparator outputs is selected to define the horizontal change point for both SET/RESET. When FUNC_SEL is set to 11, 1 of the 4 comparator outputs is selected to define the horizontal change point for RESET.<br>00: select comparator0<br>01: select comparator1<br>10: select comparator2<br>11: select comparator3<br>Note: when FUNC_SEL set to 01 or 11, user should program the corresponding comparator for horizontal compare to ensure the timing signal is generated as expected. |
| SET | set point compare value<br>Note: There will be a two pixel clock cycle delay between the internal pixel counter and the output pixel data in horizontal direction, user should take this two cycle delay into account when programming this field. See Section 46.4.2.2, “Pulse generator for detailed explanation. |
| RESET | reset point compare value<br>Note: There will be a two pixel clock cycle’s delay between the internal pixel count and the output pixel data in horizontal direction, user should take this two cycle delay into account when programming this field. See Section 46.4.2.2, “Pulse generator for detailed explanation. |

### 46.3.3.6 TCON_PULSE0_MSK - TCON_PULSE5_MSK

Offset 0x0040 - 0x0054

Access: User read/write

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

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

**Figure 46-7. TCON_PULSEx_MSK Register**

**Table 46-9. TCON_PULSEx_MSK Register Field Descriptions**

| Field | Description |
|---|---|
| SET_MASK | set point compare mask value<br>SET_MASK[x] = 1: include x postion in pulse generator set value comparison<br>SET_MASK[x] = 0, ignore x position in pulse generator set value comparison |
| RESET_MASK | reset point compare mask value<br>RESET_MASK[x] = 1, include x position in pulse generator reset value comparison<br>RESET_MASK[x] = 0, ignore x position in pulse generator reset value comparison. |

### 46.3.3.7 TCON_SMX0 - TCON_SMX13

Offset 0x0058 - 0x008C

Access: User read/write

| Power Architecture | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Conventional | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| R | INDEX3_SEL | | | | INDEX2_SEL | | | | INDEX1_SEL | | | | INDEX0_SEL | | | | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 46-8. TCON_SMXx Register**

**Table 46-10. TCON_SMXx field descriptions**

| Field | Description |
|---|---|
| INDEX3_SEL | SMXx LUT index3 selection. {index3,index2,index1,index0} will be used as address to LUT to generate timing signal.}<br>0: index3 = 0;<br>1: index3 = X;<br>2: index3 = Y;<br>3: index3 = X&Y;<br>4: index3 = X\|Y;<br>5: index3 = X^Y;<br>6: index3 = !(X&Y))<br>7: reserved for use |
| INDEX2_SEL | SMXx LUT index2 selection. Same functionality as INDEX3_SEL |

**Table 46-10. TCON_SMXx field descriptions (continued)**

| Field | Description |
|---|---|
| INDEX1_SEL | SMXx LUT index1 selection. Same functionality as INDEX3_SEL |
| INDEX0_SEL | SMXx LUT index0 selection. Same functionality as INDEX3_SEL |
| Y_SEL | SMXx logic input Y selection.<br>00: const0<br>01: const1<br>02: pulse0<br>03: pulse1<br>04: pulse2<br>05: pulse3<br>06: pulse4<br>07: pulse5<br>08: vtgl[0]<br>09: vtgl[1]<br>0A: vtgl[2]<br>0B: vtgl[3]<br>0C: SMX0<br>0D: SMX1<br>0E: SMX2<br>0F: SMX3<br>10: SMX4<br>11: SMX5<br>12: SMX6<br>13: SMX7<br>14: SMX8<br>15: SMX9<br>16: SMX10<br>17: SMX11<br>18: comp0<br>19: comp1<br>1A: comp2<br>1B: comp3<br>1C~1F: reserved |
| X_SEL | SMX logic input A selection, the same selection logic is implemented here as in Y_SEL. |

## 46.3.3.8 TCON_OMUX_LOW

Offset 0x0090

Access: User read/write

| Power Architecture | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Conventional | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| R | 0 | 0 | TCON5 | | | | | | TCON4 | | | | TCON3 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Power Architecture | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Conventional | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | TCON3 | TCON2 | | | | | | TCON1 | | | | | TCON0 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 46-9. TCON_OMUX_LOW Register**

**Table 46-11. TCON_OMUX_LOW field descriptions**

| Field | Description |
|---|---|
| TCON0 | output selection for tcon pin0<br>00: const0<br>01: const1<br>02: pulse0<br>03: pulse1<br>04: pulse2<br>05: pulse3<br>06: pulse4<br>07: pulse5<br>08: vtgl[0]<br>09: vtgl[1]<br>0A: vtgl[2]<br>0B: vtgl[3]<br>0C: SMX6<br>0D: SMX7<br>0E: SMX8<br>0F: SMX9<br>10: SMX10<br>11: SMX11<br>12: SMX12<br>13: SMX13<br>14 - 1F: reserved, default to const0 |
| TCON1 | output selection for tcon pin1, refer to field TCON0 for detailed functional description |

**Table 46-11. TCON_OMUX_LOW field descriptions (continued)**

| Field | Description |
|---|---|
| TCON2 | output selection for tcon pin2, refer to field TCON0 for detailed functional description |
| TCON3 | output selection for tcon pin3, refer to field TCON0 for detailed functional description |
| TCON4 | output selection for tcon pin4, refer to field TCON0 for detailed functional description |
| TCON5 | output selection for tcon pin5, refer to field TCON0 for detailed functional description |

## 46.3.3.9 TCON_OMUX_HIGH

Offset 0x0094

Access: User read/write

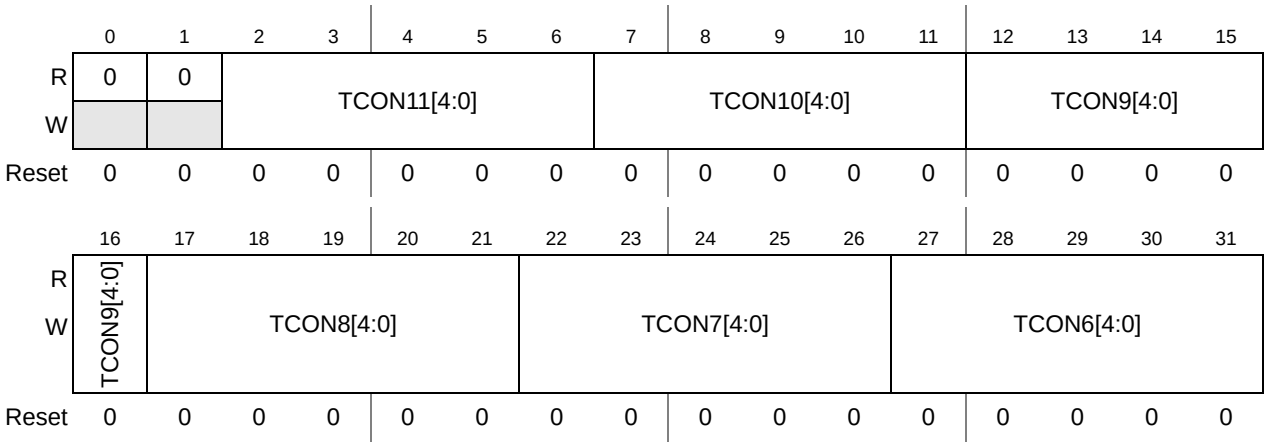



**Figure 46-10. TCON_OMUX_HIGH Register**

**Table 46-12. TCON_OMUX_HIGH field descriptions**

| Field | Description |
|---|---|
| TCON6[4:0] | output selection for tcon pin6<br>00: const0<br>01: const1<br>02: pulse0<br>03: pulse1<br>04: pulse2<br>05: pulse3<br>06: pulse4<br>07: pulse5<br>08: vtgl[0]<br>09: vtgl[1]<br>0A: vtgl[2]<br>0B: vtgl[3]<br>0C: SMX6<br>0D: SMX7<br>0E: SMX8<br>0F: SMX9<br>10: SMX10<br>11: SMX11<br>12: SMX12<br>13: SMX13<br>14 - 1F: reserved |
| TCON7[4:0] | output selection for tcon pin7, refer to field TCON6 for detailed functional description |
| TCON8[4:0] | output selection for tcon pin8, refer to field TCON6 for detailed functional description |
| TCON9[4:0] | output selection for tcon pin9, refer to field TCON6 for detailed functional description |
| TCON10[4:0] | output selection for tcon pin10, refer to field TCON6 for detailed functional description |
| TCON11[4:0] | output selection for tcon pin11, refer to field TCON6 for detailed functional description |

### 46.3.3.10 TCON_LUT0 - TCON_LUT13

Offset 0x0098 - 0x00CC

Access: User read/write

| Power Architecture | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Conventional | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| R<br>W | LUT[31:16] | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Power Architecture | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Conventional | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | LUT[15:0] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |

**Figure 46-11. TCON_LUTx Register**

**Table 46-13. TCON_LUTx Register Field Descriptions**

| Field | Description |
|---|---|
| LUT[31:0] | Look up table for SMXx.<br>SMXx(n) = TCON_LUTx[{index3,index2,index1,index0,SMXx(n-1)}], see Table 46-10 for construction of index3 to index0. |

## 46.3.3.11 TCON_DATA0_DLY - TCON_DATA12_DLY

Offset 0x00D0 - 0x0100

Access: User read/write

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

| | Power Architecture | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Conventional | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | | POL | 0 | 0 | OSUFINE2 | OSUFINE1 | OSUCRSE2 | OSUCRSE1 | OSDFINE | SKEWOPT | | | | | | | |
| W | | | | | | | | | | | | | | | | | |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 46-12. TCON_DATAx_DLY register**

**Table 46-14. TCON_DATAx_DLY field descriptions [1]**

| Field | Description |
|---|---|
| POL | Control signal for choosing polarity if same data is to be transmitted or opposite data polarity is to be transmitted |
| OSUFINE2[2] | Control signal for increasing the voltage value.(Fine Adjustment) |
| OSUFINE1[2] | Control signal for increasing the voltage value.(Fine Adjustment) |
| OSUCRSE2[3] | Control signal for increasing the voltage value to higher order.(Coarse Adjustment) |
| OSUCRSE1[3] | Control signal for increasing the voltage value.(Coarse Adjustment) |
| OSDFINE[2] | Control signal for decreasing the voltage value.(Fine Adjustment) |
| SKEWOPT[4] | 8 Programmable bits to control skew |

1 Bits 13,14,16-31 can be written and read back although these bits do not affect the functionality of the TCON module.

2 SeeTable 46-16 for details of how these bits affect the fine adjustment of the RSDS output voltage swing.

3 See Table 46-15 for details of how these bits affect the coarse adjustment of the RSDS output voltage swing.

4 See Table 46-17 for details of how these bits affect the absolute skew of the channel timing.

**Table 46-15. RSDS option bits to increase/decrease the output swing (coarse adjustment)**

| S. No | OSUCRSE2 | OSUCRSE1 | Voltage of the driver | Differential voltage across pad_p and pad_n |
|---|---|---|---|---|
| 1 | 0 | 0 | normal | default |
| 2 | 0 | 1 | increased | increase swing by 60mV |
| 3 | 1 | 0 | increased | increase swing by 100mV |
| 4 | 1 | 1 | increased | increase swing by 160mV |

**Table 46-16. RSDS option bits to increase/decrease the output swing (fine adjustment)**

| S. No | OSDFINE | OSUCRSE1 | OSUCRSE2 | Voltage of the driver | Differential voltage across pad_p and pad_n |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | normal | default |
| 2 | 0 | 0 | 1 | increased | increase swing by 25mV |
| 3 | 0 | 1 | 0 | increased | increase swing by 15mV |
| 4 | 0 | 1 | 1 | increased | increase swing by 40mV |
| 5 | 1 | 0 | 0 | decreased | decrease swing by 30mV |
| 6 | 1 | 0 | 1 | decreased | decrease swing by 5mV |
| 7 | 1 | 1 | 0 | decreased | decrease swing by 15mV |
| 8 | 1 | 1 | 1 | increased | increase swing by 10mV |

**Table 46-17. RSDS option bits to increase the skew**

| ipp_skew_opt7 | ipp_skew_opt6 | ipp_skew_opt5 | ipp_skew_opt4 | ipp_skew_opt3 | ipp_skew_opt2 | ipp_skew_opt1 | ipp_skew_opt0 | Output skew (psec) |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N.A. |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 52 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 117 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 171 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 211 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 255 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 299 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 346 |
| 1.2V | 0V | 0V | 0V | 0V | 0V | 0V | 0V | 391 |

**Table 46-17. RSDS option bits to increase the skew (continued)**

| ipp_skew_opt7 | ipp_skew_opt6 | ipp_skew_opt5 | ipp_skew_opt4 | ipp_skew_opt3 | ipp_skew_opt2 | ipp_skew_opt1 | ipp_skew_opt0 | Output skew (psec) |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N.A. |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 52 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 117 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 171 |
| 1.2V | 1.2V | 1.2V | 1.2V | 1.2V | 1.2V | 1.2V | 1.2V | 44 |

Notes:
(1) We have 256 different skew numbers for various combinations of these 8 programmable bits, in this table only few are listed. Please refer extended table for all the skew numbers.
(2) All these skew numbers are at typical corners
(3) Default value for the 8 skew option are all ‘1’.
(4) If all zero combination is applied, transmitter will not go to powerdown, instead we shall see a fixed dc transmitter output irrespective of input data. This state is N.A.

### 46.3.3.12 TCON_CTRL2

Offset 0x0104

Access: User read/write

| | Power Architecture | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Conventional | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| R | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLK_OFFSET[8:0] | | | | | | | | |
| W | | | | | | | | | | | | | | | | | |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Power Architecture | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| | Conventional | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | DIV_RATIO[7:0] | | | | | | | |
| W | | | | | | | | | | | | | | | | | |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**Figure 46-13. TCON_CTRL2 Register**

**Table 46-18. TCON_CTRL2 Register Field Descriptions**

| Field | Description |
|---|---|
| DIV_RATIO[7:0] | TCON pixel clock divide ration. When DIV_RATIO set to N, pix_clk will be ipg_clk divided by N+1. |
| CLK_OFFSET[8:0] | Output pixel clock offset value in half ipg_clk cycle unit. please refer to Section 46.4.5, “Clock/Data Skew Adjustment for use of this field. The valid range of thif field is to 0 to 2N-1 in TTL mode and 0 to N-1 in RSDS mode, where N=(DIV_RATIO+1). |

## 46.4 Functional description

The major functions of the TCON are to generate timing signals needed to drive the external row drivers and column drivers, and map the RGB data bits to the outputs according to the RSDS™ standard. It accepts RGB data, and Horizontal/Vertical synchronization signals from the Display Controller Unit module, maintains two internal counters (x_count, and y_count), and uses these to generate the timing signals. In order to support as many type of panels aspossible, the timing generation unit has been designed to be flexible and can be configured freely to generate complicated timing signals. See Section 46.4.2, “Timing signal generator for details about the timing signal generation unit.

The TCON has three operation modes including RSDS mode, TTL mode, and bypass mode. In bypass mode the input signals are passed through the TCON unchanged, and in the other two modes the TCON performs signal (the RGB data and pixel clock) bit mapping. For details of the operation modes see Section 46.4.1, “Modes of operation, and for the signal bit mapping see Section 46.4.4, “Bit Mapping Control (BMC).

The TCON also has a data inversion control feature which can be used to minimize transitions on the output data bus. For details see Section 46.4.3, “Data inversion control.

## 46.4.1 Modes of operation

### 46.4.1.1 RSDS mode

The RSDS mode is the typical operation mode of the TCON. In RSDS mode, the TCON maps the RGB data bits and the pixel clock according to the RSDS™ “Intra-panel“ interface specification, revision 1.0 (National Semiconductor). For details see Section 46.4.4.2, “Bit mapping in RSDS mode.

In RSDS mode, the TCON drives the RSDS interface, where the RGB data and the pixel clock (25 signals for 8-bit color depth, and 19 signals for 6-bit color depth) are transfered via 13 or 10 RSDS differential pairs, each differential pair carries two signals. So data is driven on both rising and falling edge of the pixel clock. The TCON timing signals are transfered via single ended TTL interface.

In RSDS mode, the TCON places the RSDS I/O in RSDS output mode.

To ease board design, the pixel clock can be assigned to any one of the differential pairs. See Section 46.4.4.4, “Clock mapping in RSDS mode.

To enable the RSDS mode, set TCON_CTRL1[RSDS_MODE]=’1’, TCON_CTRL1[TCON_BYPASS]=’0’, and TCON_CTRL1[TCON_EN]=’1’. For the TCON_CTRL1 register see 46.3.3.1/46-1526.

### 46.4.1.2 TTL mode

In TTL mode the TCON drives the RGB data, pixel clock, and the TCON timing signals all via TTL interface. In this mode simpler signal bit mapping is performed, see Section 46.4.4.1, “Bit mapping in TTL mode.

It is also possible to assign the pixel clock to any one of the 25 output pins in TTL mode, see Section 46.4.4.5, “Clock mapping in TTL mode.

To enable TTL mode, set TCON_CTRL1[RSDS_MODE]=’0’, TCON_CTRL1[TCON_BYPASS]=’0’, and TCON_CTRL1[TCON_EN]=’1’.

### 46.4.1.3 Bypass mode

For applications where the TCON is not in use, a bypass mode is provided in which the input signals for the display controller are passed through the TCON unchanged. See Table 46-19 about how the input signals are assigned to the TCON output.

**Table 46-19. input/output signal mapping in bypass mode**

| Input | Output |
|---|---|
| data_in[23:0] | data_out[25:2] |
| pix_clk | data_out[1] |
| dcu_tag | tcon_out[0] |
| hsync_in | tcon_out[1] |

**Table 46-19. input/output signal mapping in bypass mode (continued)**

| Input | Output |
|---|---|
| vsync_in | tcon_out[2] |
| data_en_in | tcon_out[3] |

To put the TCON in bypass mode, set TCON_CTRL1[TCON_EN]='0'. and TCON_CTRL1[TCON_BYPASS]='1'.

## 46.4.2 Timing signal generator

The Timing Signal Generator (TSG) is the TCON timing signal generation unit. It accepts the HSYNC/VSYNC signals from the input parallel display interface, and maintains two counters (*hcount*, *vcount*) based on that. All the timing signals are generated based on these two counters.

For every line the *hcount* counts the current pixel number, from 1 to the number of pixels per line (the full line, including inactive period). It restarts from 1 with the HSYNC signal rising edge.

Similarly the *vcount* counts from 1 with the VSYNC signal rising edge (first line), increments by "1" at the end of each line, till the last line in a frame (including inactive lines).

The timing signal generator is composed of 4 comparators (Section 46.4.2.1, "Comparator), 6 pulse generators (Section 46.4.2.2, "Pulse generator), 1 toggle generator (Section 46.4.2.3, "Toggle generator), and a signal mixer or synthesizer (Section 46.4.2.4, "Signal Mixer (SMX)). See the following sections for detailed descriptions.

### 46.4.2.1 Comparator

There are 4 comparators in the TCON, each can be configured to compare *hcount* or *vcount* with the value specified in a TCON_COMPx Register (46.3.3.3/46-1529). If desired the user can mask some bits from comparing by setting the corresponding bits in both TCON_COMPx_MSK (46.3.3.4/46-1530) and TCON_COMPx register to "0". An one cycle (pixel clock cycle, or line cycle) pulse will be generated when *the* compare result is match, and the comparison logic is shown below:

**compare_out = (TCON_COMPx.COMP_VALUE == (TCON_COMPx_MSK.MASK & hcount ))** ***Eqn. 46-1***

or

**compare_out = (TCON_COMPx.COMP_VALUE == (TCON_COMPx_MSK.MASK & vcount ))** ***Eqn. 46-2***

TCON_COMPx.FUNC_SEL determines whether Equation 46-1 or Equation 46-2 is used (i.e. comparison is done on horizontal or vertical direction).

### 46.4.2.2 Pulse generator

There are 6 pulse generators in the TCON which can be used to generate pulses which have a *set* point and a *reset* point, and the pulse length is discretional. The *set* and *reset* point is determined by TCON_PULSEx.[SET] and TCON_PULSEx.[RESET] (46.3.3.5/46-1531), to which the *hcount* or *vcount*

value will be compared. If desired the user can mask some bits from comparing by setting the corresponding bits in both TCON_PULSEx_MSK and TCON_PULSEx register to 0. The equations used to find the *set/clear* point are similar to Equation 46-1 or Equation 46-2.

TCON_PULSEx.[FUNC_SEL] controls the type of the pulse, a.k.a. whether the *set/reset* point comparison is performed on horizotnal direction (compare with *hcount*) or vertical direction (compare with *vount*). And when vertical comparison is selected (ie. FUNC_SEL = 01 or 11), 1 of the 4 comparator outputs (must be a horizontal pulse) can be selected to further determine the signal transition point on horizontal direction. Or when FUNC_SEL = 10, the signal transition will happen immediately when the vertical compare matches (ie. at the beginning of the line). When needed TCON_PULSEx.[COMPARATOR_SEL] selects 1 of the 4 comparator outputs as the horizontal reference.

#### 46.4.2.3 Toggle generator

There's 1 toggle generator in the TCON which can be used to generate signals which toggles line to line, or frame to frame, or signal which toggles line to line but polarity changes from frame to frame. The toggle generator accepts the 4 comparator outputs (TCON_COMP[0:3]) and the 6 pulses (TCON_PULSE[0-5]), and generates 4 toggle signals (*vtgl[0-3]*) based on that.

The toggle generator uses a counter (vtgl_counter) and a T flip-flop to generate the 4 toggle signals. The vtgl_counter takes 1 of the 4 comparator outputs as the clock (*h_ref*), and 1 of the 6 pulses as the enable signal (*v_ref*). The *h_ref/v_ref* selection is controlled via TCON_CTRL1.H_REF_SEL (46.3.3.1/46-1526) and TCON_CTRL1.V_REF_SEL. TCON_CTRL1.VLEN sets the maximun counter value. The vtgl_counter increments at the rising edge of *h_ref* when *v_ref* is high, and it falls back to "0" when the counter value exceeds VLEN. And then the *vtgl[0-2]* are generated by decoding the counter value (Equation 46-3 to Equation 46-5).

**vtgl[0] = (vtgl_counter == 0)** ***Eqn. 46-3***

**vtgl[1] = (vtgl_counter == 1)** ***Eqn. 46-4***

**vtgl[2] = (vtgl_counter == 2)** ***Eqn. 46-5***

The H_REF_SEL and V_REF_SEL shall be programmed so that the *h_ref* is a horizontal pulse (one pixel clock cycle pulse per line) and the *v_ref* is a vertical pulse (pulse lasts for 1 line or several lines). Figure 46-20 shows the relationship between vlen, vtgl_counter sequence and vtgl[2:0].

**Table 46-20. vtgl[0-2] sequence**

| INPUT | OUTPUT | vtgl_counter sequence | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| vlen | vtgl | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 46-20. vtgl[0-2] sequence (continued)**

| INPUT | OUTPUT | vtgl_counter sequence | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| | 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

A T flip-flop is used to generate vtgl[3]. T flip-flop is a type of flip-flop whose output toggles (inverts) at the rising edge of the clock input when the enable signal input is high. The T flip-flop in the toggle generator is clocked by *h_ref*, and the enable signal (*en*) is the bit-AND of *vtgl[0]* and *v_ref*.

By programming the VLEN, and control the pulse length of v_ref, the vtgl[3] can be used to generate toggle signals whose polarity changes from frame to frame. See Table 46-21 for some programming examples.

**Table 46-21. vtgl[3] programming examples**

| vtgl[3] | v_ref pulse length | vlen |
|---|---|---|
| toggles every line, polarity inverts frame to frame, steady during vertical blanking (2 frame sequence) | odd (eg. number of active lines + 1) | 0 |
| toggles every other line, polarity inverts frame to frame, steady during vertical blanking (2 frame sequence) | odd*2 (eg. number of active lines) | 1 |
| toggles every 3rd line, polarity inverts frame to frame, steady during vertical blanking (2 frame sequence) | odd*3 (eg. number of active lines + 3) | 2 |
| toggles every other line, walking pattern (4 frame sequence) | odd (eg. number of active lines + 1) | 1 |

.To make it easier to understand, Table 46-22 shows an simpler example where the number of active lines is 6.

**Table 46-22. ctgl[3] signal example A**

| VLEN | Frame | Active Line Number | | | | | | | | | v_ref pulse length |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | | | 7 |
| | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | | | 7 |

Table 46-22. ctgl[3] signal example A (continued)

| VLEN | Frame | Active Line Number | | | | | | | | | v_ref pulse length |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | | | | 6 |
| | 1 | 1 | 0 | 0 | 1 | 1 | 0 | | | | 6 |
| 2 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 9 |
| | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 9 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | | | 7 |
| | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | | | 7 |
| | 2 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | | | 7 |
| | 3 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | | | 7 |

### 46.4.2.4 Signal Mixer (SMX)

To generate more complicated timing signals, there are 14 Signal Mixer/Synthesizer (SMX) modules in the TCON which can be used to perform logic operations between every two signals. Each SMX module takes 2 signal inputs (X and Y), and generates 1 SMXx output.

The X and Y input of each SMX module can be selected from any two of the 28 signals (4 comparator outputs, 6 pulses, 4 toggle generator outputs, constant 0 or 1, and 12 feedback signals SMX0~11). The X/Y signal selection is configurabled via TCON_SMXx.X_SEL and TCON_SMXx.Y_SEL (46.3.3.7/46-1532). 12 of the SMXx output (SMX0~11) can be feedback to the SMXx input again in order to generate even more complicated signals (combining information of more than 2 signals),

The SMX module is composed of several logic operation units (AND, OR, XOR or NAND), and a look-up-table (LUT).

The LUT takes 4 inputs, including the current LUT output value and a 4-bit index (index[3:0]), the LUT output is the next cycle SMX output. Function performed by the LUT is determined by the values programmed into TCON_LUTx (46.3.3.10/46-1538). By default the value of TCON_LUTx is 32’hffff_005c and the default truth table of the LUT is shown in Table  where SMXx(n-1) is the current SMXx state and SMXx(n) is the next cycle SMXx value. With this setting the LUT emulates the function of a Set/Reset/Toggle/Data flip-flop, where index3 to index0 are the Set/Reset/Toggle enable/Data inputs, “Set” has the highest priority. The user can also reprogram the TCON_LUTx registers so that the LUT performs different operations (e.g. change the priority order of the Set/Reset/Toggle enable/Data signals).

The index[3:0] input of the LUT is generated by the logic operation units. Each bit of it is a logic operation of X, Y, 0 or 1. The logic operations performed are selected by TCON_SMXx.INDEX0-3_SEL (46.3.3.7/46-1532).

Table 46-23. Default TCON LUT Truth Table

| index3 | index2 | index1 | index0 | SMXx(n-1) | SMXx(n) |
|---|---|---|---|---|---|
| 1 | x[1] | x | x | x | 1 |
| 0 | 1 | x | x | x | 0 |
| 0 | 0 | 1 | x | 0 | 1 |
| 0 | 0 | 1 | x | 1 | 0 |
| 0 | 0 | 0 | 1 | x | 1 |
| 0 | 0 | 0 | 0 | x | 0 |

[1] Don't care

### 46.4.2.5 Output Crossbar Mux

The output crossbar mux is used to select 12 from the generated 20 timing signals as the 12 TCON timing signal outputs. The 20 timing signals include: 6 pulses, 4 toggle generator outputs (vtgl[0:3]), constant "0" and "1", and 8 of the 14 SMXs output (SMX6~13). TCON_OMUX_HIGH (46.3.3.8/46-1535) and TCON_OMUX_LOW (46.3.3.9/46-1536) determines which 12 timing signals are selected as the output.

## 46.4.3 Data inversion control

The data inversion control is adopted to minimize data transition on the data bus, in order to reduce EMI on board level. The idea is to invert the data output when more than half of the data bits changed compare to the previous data output, and use a signal (tcon_out[11]) to indicate to the external gray scale reference circuit that the data is inverted.

The user can enable the data inverse control by setting TCON_CTRL1.data_inv_en to "1" (46.3.3.1/46-1526), in that case, tcon_out[11] will indicate the data inversion so it can't be used as timing signal output. But it can if the data inversion control is disabled.

## 46.4.4 Bit Mapping Control (BMC)

The Bit Mapping Control (BMC) module is adopted to remap the color component order, color bit order and output clock position for both RSDS and TTL mode. Figure 46-14 shows the diagram of the bit mapping control module.

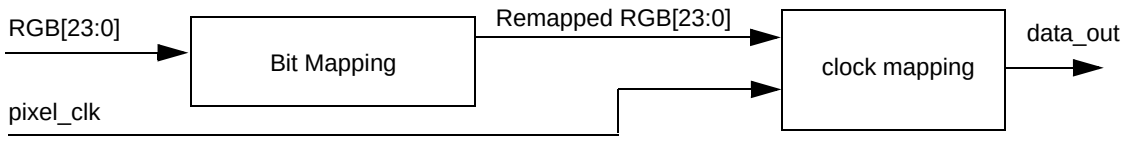



Figure 46-14. Bit mapping control

### 46.4.4.1 Bit mapping in TTL mode

See Table 46-24 for default bit mapping in 8-bit color mode.

**Table 46-24. TTL mode bit mapping, 8-bit color**

| data signal | Rising of pix_clk |
|---|---|
| Remapped RGB[23:16] | {R7,R6,R5,R4,R3,R2,R1,R0} |
| Remapped RGB[15:8] | {G7,G6,G5,G4,G3,G2,G1,G0} |
| Remapped RGB[7:0] | {B7,B6,B5,B4,B3,B2,B1,B0} |

See Table 46-25 for default bit mapping in 6-bit color mode.

**Table 46-25. TTL mode bit mapping, 6-bit color**

| data signal | Rising of pix_clk |
|---|---|
| Remapped RGB[23:16] | {R7,R6,R5,R4,R3,R2,2'b0} |
| Remapped RGB[15:8] | {G7,G6,G5,G4,G3,G2,2'b0} |
| Remapped RGB[7:0] | {B7,B6,B5,B4,B3,B2,2'b0} |

Beside the default bit mapping, it's possible to swap the RGB color component order (TCON_BMC.COLOR_ORDER) and the LSB/MSG bit order in each color component (TCON_BMC.BIT_ORDER), see Section 46.3.3.2, "Bit Mapping Control (TCON_BMC).

### 46.4.4.2 Bit mapping in RSDS mode

See Table 46-26 for default bit mapping in 8-bit color mode.

**Table 46-26. RSDS mode bit mapping, 8-bit color**

| data signal | Rising of pix_clk | Falling of pix_clk |
|---|---|---|
| Remapped RGB[23] | R6 | R7 |
| Remapped RGB[22] | R4 | R5 |
| Remapped RGB[21] | R2 | R3 |
| Remapped RGB[20] | R0 | R1 |
| Remapped RGB[19:16] | 0 | 0 |
| Remapped RGB[15] | G6 | G7 |
| Remapped RGB[14] | G4 | G5 |
| Remapped RGB[13] | G2 | G3 |
| Remapped RGB[12] | G0 | G1 |
| Remapped RGB[11:8] | 0 | 0 |

**Table 46-26. RSDS mode bit mapping, 8-bit color (continued)**

| data signal | Rising of pix_clk | Falling of pix_clk |
|---|---|---|
| Remapped RGB[7] | B6 | B7 |
| Remapped RGB[6] | B4 | B5 |
| Remapped RGB[5] | B2 | B3 |
| Remapped RGB[4] | B0 | B1 |
| Remapped RGB[3:0] | 0 | 0 |

See Table 46-27 for default bit mapping in 6-bit color mode.

**Table 46-27. RSDS mode bit mapping, 6-bit color**

| data signal | Rising of pix_clk | Falling of pix_clk |
|---|---|---|
| Remapped RGB[23] | R6 | R7 |
| Remapped RGB[22] | R4 | R5 |
| Remapped RGB[21] | R2 | R3 |
| Remapped RGB[20:16] | 0 | 0 |
| Remapped RGB[15] | G6 | G7 |
| Remapped RGB[14] | G4 | G5 |
| Remapped RGB[13] | G2 | G3 |
| Remapped RGB[12:8] | 0 | 0 |
| Remapped RGB[7] | B6 | B7 |
| Remapped RGB[6] | B4 | B5 |
| Remapped RGB[5] | B2 | B3 |
| Remapped RGB[4:0] | 0 | 0 |

Beside the default bit mapping, it's possible to swap the RGB color component order (TCON_BMC.COLOR_ORDER), the LSB/MSG bit order in each color component (TCON_BMC.BIT_ORDER), and the 2 bits transfered during rising and falling pixel clock cycle (TCON_BMC.BIT_SWAP). see Section 46.3.3.2, "Bit Mapping Control (TCON_BMC).

### 46.4.4.3 Bit mapping examples

```
a.  RSDS mode,  Below settings will remap the bit order as in Table 46-28
TCON_CTRL1.OP_MODE = 1;   //RSDS mode
TCON_CTRL1.COLOR_DEPTH = 1;  //8bit per color
TCON_BMC.COLOR_ORDER = 3'b000;  //color order RGB
TCON_BMC.BMC.BIT_ORDER = 1;   //0 up to 7MSB
TCON_BMC.BMC.BIT_SWAP = 0;    //no odd/even bit order swap
```

**Table 46-28. Bit mapping example A**

| data signal | Rising of pix_clk | Falling of pix_clk |
|---|---|---|
| Remapped RGB[23] | R0 | R1 |
| Remapped RGB[22] | R2 | R3 |
| Remapped RGB[21] | R4 | R5 |
| Remapped RGB[20] | R6 | R7 |
| Remapped RGB [19:16] | 0 | 0 |
| Remapped RGB[15] | G0 | G1 |
| Remapped RGB[14] | G2 | G3 |
| Remapped RGB[13] | G4 | G5 |
| Remapped RGB[12] | G6 | G7 |
| Remapped RGB [11:8] | 0 | 0 |
| Remapped RGB[7] | B0 | B1 |
| Remapped RGB[6] | B2 | B3 |
| Remapped RGB[5] | B4 | B5 |
| Remapped RGB[4] | B6 | B7 |
| Remapped RGB [3:0] | 0 | 0 |

b. RSDS mode, Below settings will remap the bit order as in Table 46-29

```
TCON_CTRL1.DATA_MODE = 1;   //RSDS mode
TCON_CTRL1.COLOR_DEPTH = 0;  //6 bit per color
TCON_BMC.COLOR_ORDER = 3'b101;  //color order BGR
TCON_BMC.BMC.BIT_ORDER = 0;   //MSB 7  down to 0
TCON_BMC.BMC.BIT_SWAP = 1;     //swap odd and even bit
```

**Table 46-29. Bit mapping example B**

| data signal | Rising of pix_clk | Falling of pix_clk |
|---|---|---|
| Remapped RGB[23] | B7 | B6 |
| Remapped RGB[22] | B5 | B4 |
| Remapped RGB[21] | B3 | B2 |
| Remapped RGB [20:16] | 0 | 0 |
| Remapped RGB[15] | G7 | G6 |
| Remapped RGB[14] | G5 | G4 |
| Remapped RGB[13] | G3 | G2 |
| Remapped RGB [12:8] | 0 | 0 |

**Table 46-29. Bit mapping example B (continued)**

| data signal | Rising of pix_clk | Falling of pix_clk |
|---|---|---|
| Remapped RGB[7] | R7 | R6 |
| Remapped RGB[6] | R5 | R4 |
| Remapped RGB[5] | R3 | R2 |
| Remapped RGB [4:0] | 0 | 0 |

```
c.   TTL mode, below settings will remap the bit order as in Table 46-30
TCON_CTRL1.DATA_MODE = 0;   //TTL mode
TCON_BMC.COLOR_ORDER = 3'b000;  //color order RGB
TCON_BMC.BIT_ORDER = 1;   //0  up to MSB7
TCON_CTRL1.COLOR_DEPTH = 1;  //6bit per color;
```

**Table 46-30. Bit mapping example C**

| RSDS 8-bits | Rising of pix_clk |
|---|---|
| Remapped RGB[23:16] | {R2,R3,R4,R5,R6,R7,2'b0} |
| Remapped RGB[15:8] | {G2,G3,G4,G5,G6,G7,2'b0} |
| Remapped RGB[7:0] | {B2,B3,B4,B5,B6,B7,2'b0} |

### 46.4.4.4 Clock mapping in RSDS mode

To aid board design, it has been made possible to assign the pixel clock output to any of the 13 RSDS differential pairs, configured via TCON_BMC.CLK_POS (46.3.3.2/46-1528). The flexible clock mapping in RSDS mode is shown in Table 46-31.

**Table 46-31. Clock mapping in RSDS mode**

| data_out | CLK_POS[4:0] | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 0/1 | CLK[1] | ~4/4 | | | | | | | | | | | |
| 2/3 | ~4/4[2] | CLK | ~5/5 | | | | | | | | | | |
| 4/5 | ~5/5 | | CLK | ~6/6 | | | | | | | | | |
| 6/7 | ~6/6 | | | CLK | ~7/7 | | | | | | | | |
| 8/9 | ~7/7 | | | | CLK | ~12/12 | | | | | | | |
| 10/11 | ~12/12 | | | | | CLK | ~13/13 | | | | | | |
| 12/13 | ~13/13 | | | | | | CLK | ~14/14 | | | | | |
| 14/15 | ~14/14 | | | | | | | CLK | ~15/15 | | | | |
| 16/17 | ~15/15 | | | | | | | | CLK | ~20/20 | | | |

**Table 46-31. Clock mapping in RSDS mode (continued)**

| data_out | CLK_POS[4:0] | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 18/19 | ~20/20 | | | | | | | | | CLK | ~21/21 | | |
| 20/21 | ~21/21 | | | | | | | | | | CLK | ~22/22 | |
| 22/23 | ~22/22 | | | | | | | | | | | CLK | ~23/23 |
| 24/25 | ~23/23 | | | | | | | | | | | | CLK |

1 CLK stands for differential clock signal pair: clk_out_b/clk_out

2 ~n/n stands for differential data signal pair: ~remapped rgb[n]/remapped rgb[n]

### 46.4.4.5 Clock mapping in TTL mode

Similar to RSDS mode, it's also possible to assign the pixel clock output to any bit of the data_out[1:25] in TTL mode, configured via TCON_BMC.CLK_POS (46.3.3.2/46-1528). See Table 46-32 for details.

**Table 46-32. Clock mapping in TTL mode**

| data_out | CLK_POS[4:0] | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 0 | const0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | C[1] | 0 | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | 0[2] | C | 1 | | | | | | | | | | | | | | | | | | | | | | |
| 3 | 1 | | C | 2 | | | | | | | | | | | | | | | | | | | | | |
| 4 | 2 | | | C | 3 | | | | | | | | | | | | | | | | | | | | |
| 5 | 3 | | | | C | 4 | | | | | | | | | | | | | | | | | | | |
| 6 | 4 | | | | | C | 5 | | | | | | | | | | | | | | | | | | |
| 7 | 5 | | | | | | C | 6 | | | | | | | | | | | | | | | | | |
| 8 | 6 | | | | | | | C | 7 | | | | | | | | | | | | | | | | |
| 9 | 7 | | | | | | | | C | 8 | | | | | | | | | | | | | | | |
| 10 | 8 | | | | | | | | | C | 9 | | | | | | | | | | | | | | |

**Table 46-32. Clock mapping in TTL mode (continued)**

| data_out | CLK_POS[4:0] | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 11 | 9 | | | | | | | | | | C | 10 | | | | | | | | | | | | | |
| 12 | 10 | | | | | | | | | | | C | 11 | | | | | | | | | | | | |
| 13 | 11 | | | | | | | | | | | | C | 12 | | | | | | | | | | | |
| 14 | 12 | | | | | | | | | | | | | C | 13 | | | | | | | | | | |
| 15 | 13 | | | | | | | | | | | | | | C | 14 | | | | | | | | | |
| 16 | 14 | | | | | | | | | | | | | | | C | 15 | | | | | | | | |
| 17 | 15 | | | | | | | | | | | | | | | | C | 16 | | | | | | | |
| 18 | 16 | | | | | | | | | | | | | | | | | C | 17 | | | | | | |
| 19 | 17 | | | | | | | | | | | | | | | | | | C | 18 | | | | | |
| 20 | 18 | | | | | | | | | | | | | | | | | | | C | 19 | | | | |
| 21 | 19 | | | | | | | | | | | | | | | | | | | | C | 20 | | | |
| 22 | 20 | | | | | | | | | | | | | | | | | | | | | C | 21 | | |
| 23 | 21 | | | | | | | | | | | | | | | | | | | | | | C | 22 | |
| 24 | 22 | | | | | | | | | | | | | | | | | | | | | | | C | 23 |
| 25 | 23 | | | | | | | | | | | | | | | | | | | | | | | | C |

1 CLOCK

2 n stand for remapped rgb[n]

### 46.4.5 Clock/Data Skew Adjustment

In order to fulfill different setup/hold timing requirements on the output interface, it has been made possible to adjust the signal skew between the pixel clock output and the data output (including RGB data and timing signals). The output pixel clock can be flexibly shifted compared to the data signals. It can be shifted over the whole pixel clock period, in granularity of half pixel clock source cycle (the clock from which the pixel clock is derived/divided).

TCON_CTRL2.DIV_RATIO (46.3.3.12/46-1542) configures the pixel clock divide ratio, and TCON_CTRL2.CLK_OFFSET configures the offset between the internal pixel clock and the output pixel clock..

## 46.5 RSDS interface description

### 46.5.1 Introduction

RSDS (Reduced swing differential signaling) is an intra panel interface bus standard. It is a Sub-LVDS system, as its application is with a subsystem, the signal swing is further reduced from LVDS to further lower power. As compared to LVDS, the RSDS scheme uses a 2:1 serialization scheme, resulting in a less complex and low power architecture. This interface operates at a maximum frequency of 80 MHz. Apart from RSDS_Tx pad there is another cell called RSDS_ref which is to provide current and voltage biases to the transmitter cell.

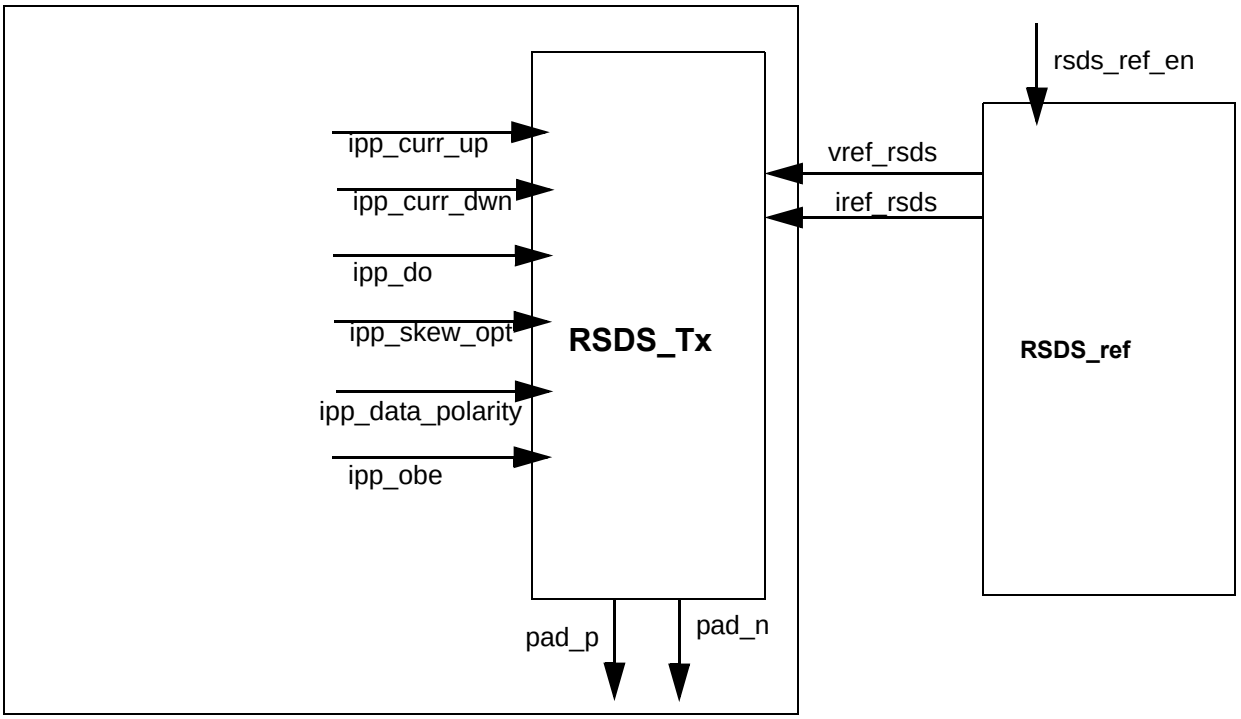



**Figure 46-15. RSDS interface**

## 46.5.2 Features

Main features of RSDS transmitter are as follows

- Common Mode voltage between 0.5 V to 1.5 V
- Supports data rates upto 120 Mbps
- Output Impedence of 100 Ω differential
- Programmable voltage swing
- Programmable option for controlling data polarity
- Programmable skew control

## 46.5.3 Data path signals

### 46.5.3.1 pad_p

This is the “positive” I/O signal connected bond pad for the RSDS output pad. The voltage supported by the pad is defined by the VREF_RSDS and VSS associated with the cell.

### 46.5.3.2 pad_n

This is the “negative” I/O signal connected bond pad for the RSDS output pad. The voltage supported by the pad is defined by the VREF_RSDS and VSS associated with the cell.

## 46.5.4 Cell description

### 46.5.4.1 RSDS_ref - RSDS Reference cell

This pad cell provides the current reference bias that is mirrored to set the output current for the RSDS driver. This pad also provides a voltage reference that is used in the RSDS driver common-mode feedback circuit.

### 46.5.4.2 RSDS_Tx - RSDS transmitter cell

This pad cell drives the differential pads, pad_p and pad_n with a specified offset voltage (common mode voltage) and differential voltage swing. This is done in current mode, i.e. current is sourced at pad_p and sinked at pad_n (and vice-versa) depending on whether the data to be transmitted is High (or Low). Below is the illustrative diagram for the same. Internally, it has H-bridge kind of output stage with one current source and one current sink. There are four switches which steer the current into the load connected between pad_p and pad_n. This happens by controlling the switches in such a way that at one time, a set of two diagonal switches is on when incoming data is high, and the other set of two diagonal switches is on when data is Low. Thus, same amount of current flows through load in opposite direction depending on the data value.

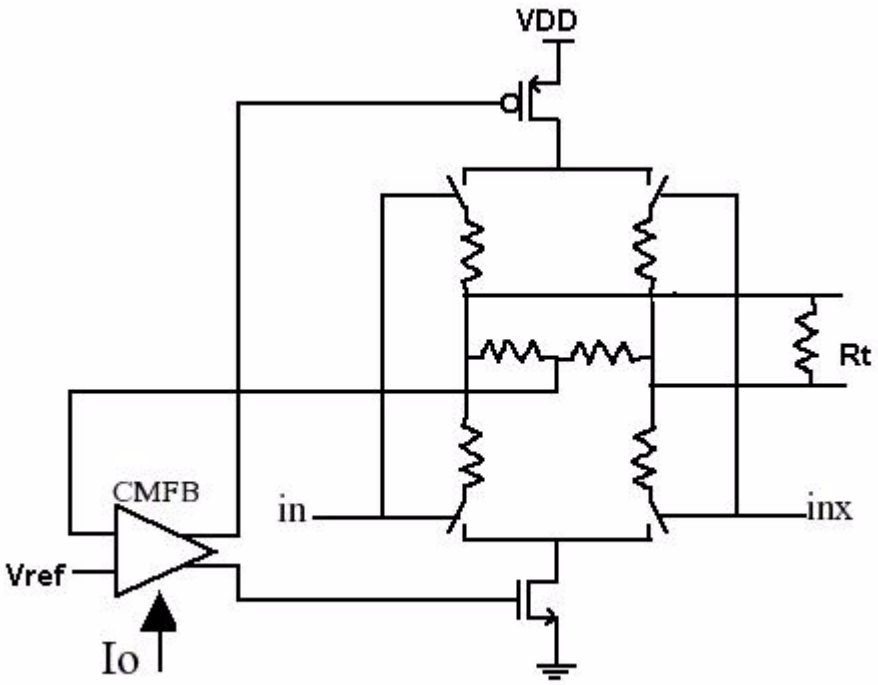



**Figure 46-16. Illustrative diagram of RSDS driver**

## 46.5.5 Functionality and modes of operation:

**Table 46-33. Modes of operation**

| modes | description | conditions |
|---|---|---|
| normal (asynch) | data transmission (asynchronous) | rsds_ref_en = ipp_obe =1.2, |
| normal (asynch positive edge) | data transmission (asynchronous), with positive data polarity. | rsds_ref_en = ipp_obe =1.2, ipp_data_polarity =0V |
| normal (asynch, negative edge) | data transmission (asynchronous), with negative data polarity. | rsds_ref_en = ipp_obe =1.2, ipp_data_polarity = 1.2V |
| Power down (ref) | rsds_ref is in power down. | rsds_ref_en = 0 |
| Power down (driver) | rsds_tx is in power down/disable. | ipp_obe = 0 |

**Table 46-34. Logic table for outputs**

| ipg_powseq_dr_off | rsds_ref_en | ipp_obe | ipp_do | pad_p | pad_n |
|---|---|---|---|---|---|
| 1 | X | X | X | Z | Z |
| 0 | 0 | X | X | X | X |
| 0 | 1 | 0 | X | Z | Z |

Table 46-34. Logic table for outputs (continued)

| ipg_powseq_dr_off | rsds_ref_en | ipp_obe | ipp_do | pad_p | pad_n |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |

## 46.5.6 General

Some pad timing diagrams are included below.

## 46.5.7 Timing Diagrams

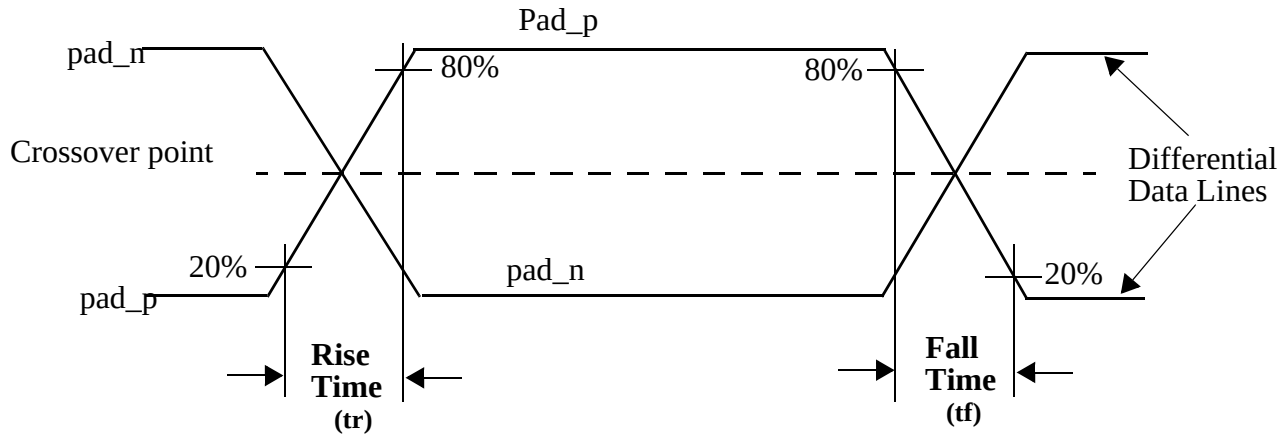



Figure 46-17. Rise/Fall Transition

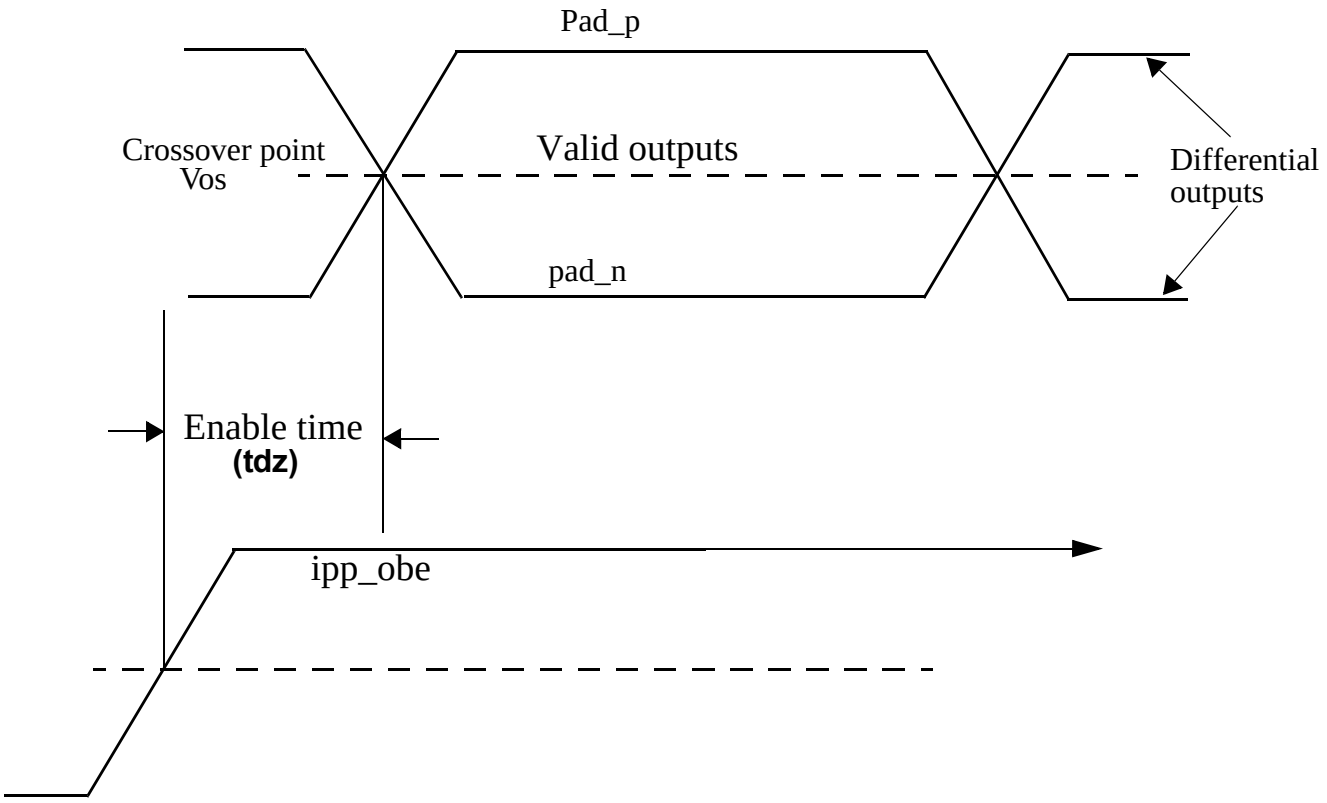



**Figure 46-18. Enable time**

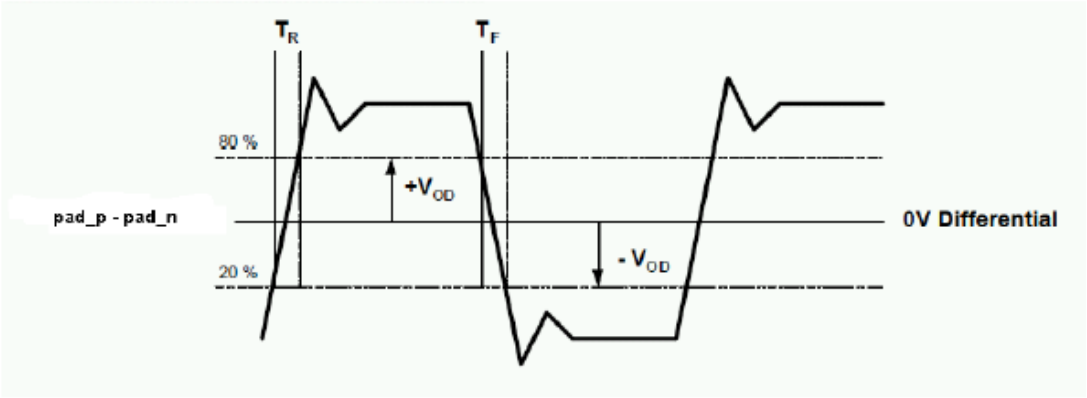



**Figure 46-19. Rise & Fall Transition time for Differential output signal (pad_p - pad_n)**

## 46.6 Initialization/application information

### 46.6.1 TCON Initialization

The procedure to bring up the TCON out of reset state and start data and timing signal generation:

1. Program TCON_COMPx and  TCON_COMPx_MSK registers to configure the comparator.
2. Program TCON_PULSEx and TCON_PULSEx_MSK registers to configure the pulse generator.
3. Program TCON_SMXx and TCON_LUTx registers if necessary.
4. Program TCON_BMC (46.3.3.2/46-1528) register to configure the bit mapping control.
5. Program the TCON_CTRL1 (46.3.3.1/46-1526) register to configure the operation mode of TCON and enable TCON (It's recommended to do this in two steps, and set TCON_EN in the second).

# Chapter 47
# Video Input Unit (VIU2)

## 47.1 Introduction

The VIU2 is an enhanced version of the original Video-In (VIU) block, with new features added including Down-scaling, Brightness and Contrast adjustment, and YUV 4:2:2 output.

Figure 47-1 shows a block diagram of the VIU2.

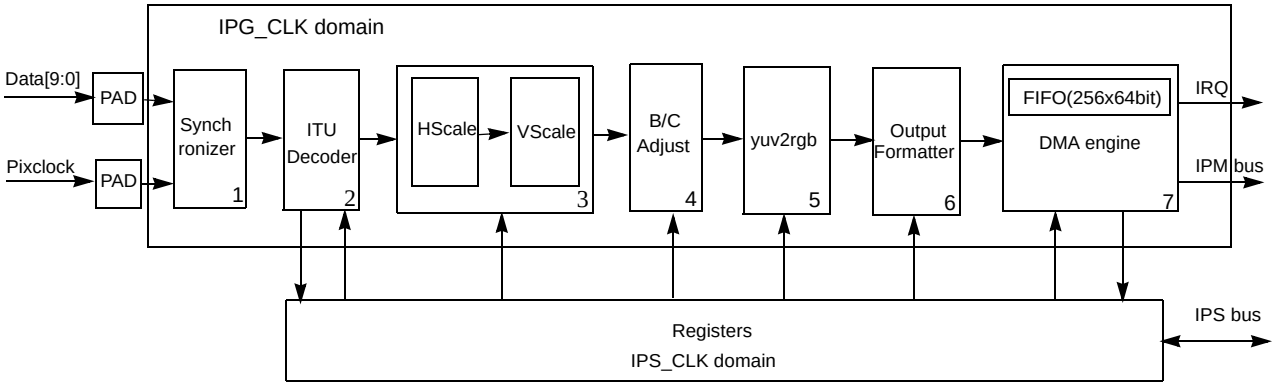



**Figure 47-1. VIU2 block diagram**

### 47.1.1 Features

- Support from QVGA to XVGA 8-bit/10-bit ITU656 video input[1]
- Up to 1/8 video down-scaling
- Support different scaling ratio on horizontal and vertical direction
- Brightness/contrast adjust
- YUV to RGB 888/565 conversion
- Simple de-interlace function (weaving) for interlaced/or pseudo interlaced video input
- Internal DMA engine for transfering data from FIFO to system memory

## 47.2 Memory map and register definition

### 47.2.1 Memory map

**Table 47-1. VIU2 memory map**

| Address offset | Register | Access | Location |
|---|---|---|---|
| 0x00 | Status And Configuration Register (SCR) | RW | on page 1565 |
| 0x04 | Luminance Coefficients For Red, Green And Blue Matrix (LUMA_COMP) | RW | on page 1567 |
| 0x08 | Chroma Coefficients For Red Matrix (CHROMA_RED) | RW | on page 1568 |
| 0x0c | Chroma Coefficients For Green Matrix (CHROMA_GREEN) | RW | on page 1569 |
| 0x10 | Chroma Coefficients For Blue Matrix (CHROMA_BLUE) | RW | on page 1569 |
| 0x14 | Base Address Of Every Field/Frame Of Picture In Memory (DMA_ADDR) | RW | on page 1570 |
| 0x18 | Horizontal DMA Increment (DMA_INC) | RW | on page 1570 |
| 0x1c | Input Video Pixel and Line Count (INVSZ) | RW | on page 1571 |
| 0x20 | High IPM Request Priority Alarm (HPALRM) | RW | on page 1572 |
| 0x24 | Programable Alpha Value (ALPHA) | RW | on page 1572 |
| 0x28 | Down Scaling Factor At Horizonal Direction (HFACTOR) | RW | on page 1573 |
| 0x2c | Down Scaling Factor At Vertical Direction (VFACTOR) | RW | on page 1573 |
| 0x30 | Down Scaling Destination Pixel and Line Count (VID_SIZE) | RW | on page 1574 |
| 0x34 | B/C Adjust Look-up-table Current Address (LUT_ADDR) | RW | on page 1574 |
| 0x38 | B/C Adjust Look-up-table Data Entry (LUT_DATA) | RW | on page 1575 |

### 47.2.2 Register Summary

Table 47-2 shows the VIU2 register summary table.

1. When down scaling and/or B/C adjust is enabled, the two LSB's of the 10-bit input are ignored.

**Table 47-2. VIU2 Register Summary**

| Name | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x00 SCR | R | MODE32BIT | ROUND_ON | DITHER_ON | FIELD_NO | DMA_ACT | SCALER_EN | RGB_EN | BC_EN | MODE444 | | ERROR_IRQ | DMA_END_IRQ | VSTART_IRQ | HSYNC_IRQ | VSYNC_IRQ | FIELD_IRQ |
| | W | | | | | | | | | | | w1c | w1c | w1c | w1c | w1c | w1c |
| | R | | ECC_EN | ERROR_EN | DMA_END_EN | VSTART_EN | HSYNC_EN | VSYNC_EN | FIELD_EN | ERROR_CODE | | | | FORMAT_CTRL | | | SOFT_RESET |
| | W | | | | | | | | | | | | | | | | |
| Ox04 LUMA_COMP | R | Y_RED[9:0] | | | | | | | | | | | Y_GREEN[9:5] | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | Y_GREEN[4:0] | | | | | | Y_BLUE[9:0] | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| Ox08 CHROMA_RED | R | | | | | | CR_RED[10:0] | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | CB_RED[11:0] | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| Ox0C CHROMA_GREEN | R | | | | | | CR_GREEN[10:0] | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | CB_GREEN[11:0] | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| Ox10 CHROMA_BLUE | R | | | | | | CR_BLUE[10:0] | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | CB_BLUE[11:0] | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x14 DMA_ADDR | R | ADDR[31:16] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | ADDR[15:3] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x18 DMA_INC | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | INC | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

**Table 47-2. VIU2 Register Summary**

| Name | | 31 15 | 30 14 | 29 13 | 28 12 | 27 11 | 26 10 | 25 9 | 24 8 | 23 7 | 22 6 | 21 5 | 20 4 | 19 3 | 18 2 | 17 1 | 16 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x1C INVSZ | R | LINEC | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | PIXELC | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x20 HPALRM | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | ALARM | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x24 ALPHA | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | | | | | ALPHA[7:0] | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x28 HFACTOR | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | | FACTOR | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x2C VFACTOR | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | | FACTOR | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x30 VID_SIZE | R | LINEC | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | PIXELC | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x34 LUT_ADDR | R | | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | | | | | | | ADDR | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| 0x38 LUT_DATA | R | DATA[31:16] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |
| | R | DATA[15:0] | | | | | | | | | | | | | | | |
| | W | | | | | | | | | | | | | | | | |

## 47.2.3 Register descriptions

### 47.2.3.1 SCR

Offset 0x00 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| R | MODE32BIT | ROUND_ON | DITHER_ON | FIELD_NO | DMA_ACT | SCALER_EN | RGB_EN | BC_EN |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|
| R | MODE444 | | ERROR_IRQ | DMA_END_IRQ | VSTART_IRQ | HSYNC_IRQ | VSYNC_IRQ | FIELD_IRQ |
| W | | | w1c | w1c | w1c | w1c | w1c | w1c |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|
| R | | ECC_EN | ERROR_EN | DMA_END_EN | VSTART_EN | HSYNC_EN | VSYNC_EN | FIELD_EN |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|
| R | ERROR_CODE | | | | FORMAT_CTRL | | | SOFT_RESET |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 47-2. SCR Register**

**Table 47-3. SCR Fields**

| Field | Description |
|---|---|
| SOFT_RESET | Writing 1 to this bit generates an internal reset to all components except registers in VIU2 block. This bit should be set by software when an error interrupt is detected, and it needs to be cleared by software to release the software reset. |
| FORMAT_CTRL | Output pixel data format control. See below or refer to Figure 47-20 for detailed definition. Here 32-bit means MODE32BIT is set and 16-bit means MODE32BIT is cleared. RGB mode means RGB_EN is set and YUV mode means RGB_EN is cleared. C means U or V. Dummy means “don’t care” data.<br>16-bit RGB mode:<br>3'b000: {R[7:3], G[7:2], B[7:3]};<br>3'b001: {G[4:2], B[7:2], R[7:3], G[7:5]};<br><br>32-bit RGB mode:<br>3'b000: {alpha, R, G, B};<br>3'b001: {alpha, B, G, R};<br>3'b010: {R, G, B, alpha};<br>3'b011: {B, G, R, alpha};<br><br>16-bit YUV mode:<br>3'b000: {C,Y};<br>3'b001: {Y,C};<br><br>32-bit YUV mode:<br>3'b000: {dummy, Y, U, V};<br>3'b001: {dummy, Y, V, U};<br>3'b010: {dummy, U, V, Y};<br>3'b011: {dummy, V, U, Y};<br>3'b100: {Y, U, V, dummy};<br>3'b101: {Y, V, U, dummy};<br>3'b110: {U, V, Y, dummy};<br>3'b111: {V, U, Y, dummy}; |
| ERROR_CODE | Error code. Signals errors that triggered error IRQ.<br>0000 : No error<br>0001 : DMA arm command given during vertical active, DMA_ACT does not accept the value on IPS bus.<br>0010 : DMA arm command given during vertical blanking when DMA_ACT is set.<br>0100 : Line too long<br>0101 : Too many lines in a field/frame<br>0110 : Line too short<br>0111 : Not enough lines in a field/frame<br>1000 : FIFO overflow<br>1001 : FIFO underflow<br>1010 : One bit ECC error<br>1011 : Two or more bits ECC error<br>others: Reserved |
| FIELD_EN | Interrupt enable bit for FIELD_IRQ. |
| VSYNC_EN | Interrupt enable bit for VSYNC_IRQ. |
| HSYNC_EN | Interrupt enable bit for HSYNC_IRQ. |
| VSTART_EN | Interrupt enable bit for VSTART_IRQ. |
| DMA_END_EN | Interrupt enable bit for DMA_END_IRQ. |
| ERROR_EN | Interrupt enable bit for ERROR_IRQ. |

**Table 47-3. SCR Fields**

| Field | Description |
|---|---|
| ECC_EN | Enable bit for ECC error reporting. VIU will also trigger interrupt upon detecting one or more errors.. |
| FIELD_IRQ | FIELD_IRQ occurs when the field number indicator is reversed. Write '1' to clear. |
| VSYNC_IRQ | Interrupt status bit. Write ‘1’ to clear VSYNC_IRQ. |
| HSYNC_IRQ | Interrupt status bit. Write ‘1’ to clear HSYNC_IRQ. |
| VSTART_IRQ | VSTART_IRQ occurs when a frame is coming from ITU bus. Write '1' to clear. |
| DMA_END_IRQ | DMA_END_IRQ occurs when a frame is transferred over by DMA master. Write '1' to clear. |
| ERROR_IRQ | Interrupt status bit. Write ‘1’ to clear ERROR_IRQ. |
| MODE444 | YUV 4:4:4 mode enable bit. When it’s set ITU decoder sends out YUV 4:4:4 format data, otherwise YUV 4:2:2 is sent by default. It shall be cleared when down scaling is enabled because the down-scaler works on YUV 4:2:2 format. |
| BC_EN | Bright/Contrast adjust enable. |
| RGB_EN | YUV to RGB conversion enable. |
| SCALER_EN | Down scaling enable. |
| DMA_ACT | DMA transfer of current field/frame is busy (write by software, cleared at end of transfer). When DMA_ACT is cleared, input video data is ignored and not put into FIFO. |
| FIELD_NO | Field number, extracted from ITU-656 stream. |
| DITHER_ON | Dithering is on. Used when video data is stored in buffer as RGB565 format and ROUND_ON is not set. |
| ROUND_ON | Round is on. Used when video data is stored in buffer as RGB565 format. |
| MODE32BIT | Select 32-bit or 16-bit output.<br>0 : 16-bit RGB or YUV 4:2:2 output<br>1: 32-bit RGB or YUV 4:4:4 output. DITHER_ON and ROUND_ON are ignored if output is 32-bit RGB. |

## 47.2.3.2 LUMA_COMP

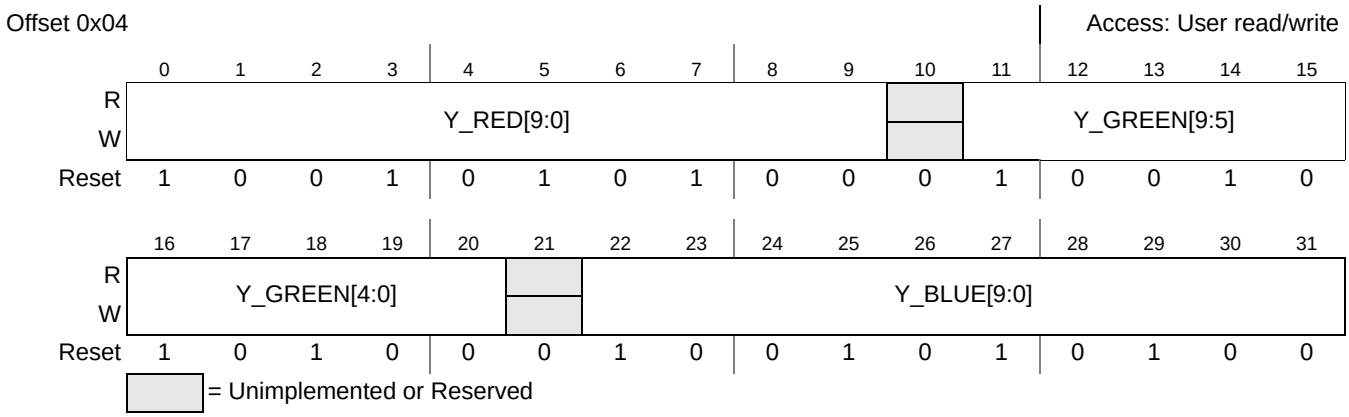



**Figure 47-3. LUMA_COMP Register**

**Table 47-4. LUMA_COMP Fields**

| Field | Description |
|---|---|
| Y_RED[9:0] | Luminance coefficient for red matrix. |
| Y_GREEN[9:0] | Luminance coefficient for green matrix. |
| Y_BLUE[9:0] | Luminance coefficient for blue matrix. |

The RGB pixel value is computed using following formulae:

$$\text{Red} = \frac{(Y-16)\cdot yred}{512} + \frac{(Cr-128)Crred}{512} + \frac{(Cb-128)Cbred}{512} \quad \textit{Eqn. 47-1}$$

$$\text{Green} = \frac{(Y-16)\cdot Ygreen}{512} + \frac{(Cr-128)Crgreen}{512} + \frac{(Cb-128)Cbgreen}{512} \quad \textit{Eqn. 47-2}$$

$$\text{Blue} = \frac{(Y-16)\cdot yblue}{512} + \frac{(Cr-128)Crblue}{512} + \frac{(Cb-128)Cbblue}{512} \quad \textit{Eqn. 47-3}$$

The multiplications with Y_red, Y_green, and Y_blue are unsigned multiplications. The multiplications with Cr_red, Cb_red, Cr_green, Cb_green, Cr_blue, and Cb_blue are signed multiplications. The addition is saturated to prevent overflow.

### 47.2.3.3 CHROMA_RED

Offset 0x08 — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | CR_RED[10:0] | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | CB_RED[11:0] | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 47-4. CHROMA_RED Register**

**Table 47-5. CHROMA_RED Fields**

| Field | Description |
|---|---|
| CR_RED[10:0] | Cr coefficient for red matrix. |
| CB_RED[11:0] | Cb coefficient for red matrix. |

### 47.2.3.4 CHROMA_GREEN

Offset 0x0C — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | CR_GREEN[10:0] | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | CB_GREEN[11:0] | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 47-5. CHROMA_GREEN Register**

**Table 47-6. GREEN_CHROMA_COEFFICIENTS Fields**

| Field | Description |
|---|---|
| CR_GREEN[10:0] | Cr coefficient for green matrix. |
| CB_GREEN[11:0] | Cb coefficient for green matrix. |

### 47.2.3.5 CHROMA_BLUE

Offset 0x10 — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | CR_BLUE[10:0] | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | CB_BLUE[11:0] | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

= Unimplemented or Reserved

**Figure 47-6. CHROMA_BLUE Register**

**Table 47-7. CHROMA_BLUE Fields**

| Field | Description |
|---|---|
| CR_BLUE[10:0] | Cr coefficient for blue matrix. |
| CB_BLUE[11:0] | Cb coefficient for blue matrix. |

### 47.2.3.6 DMA_ADDR

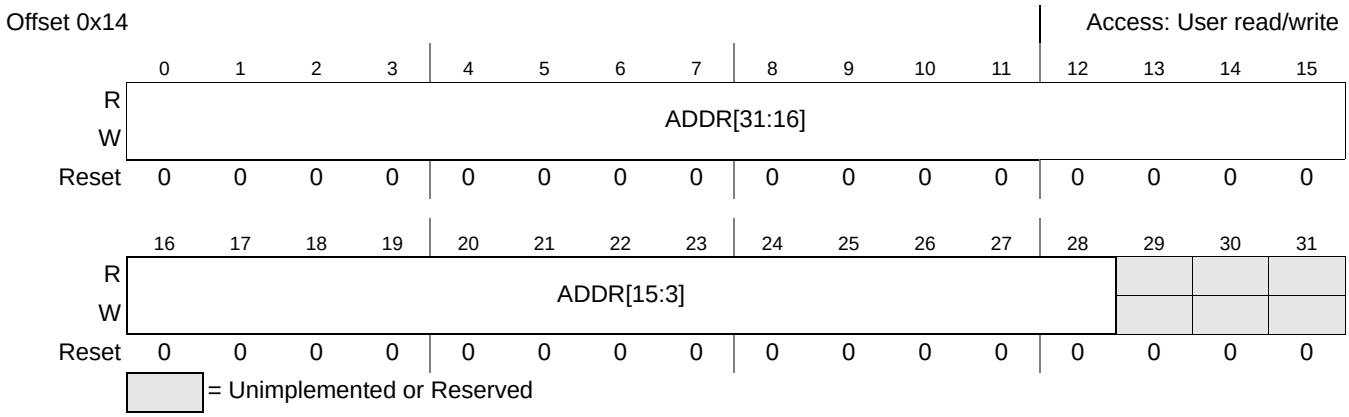



**Figure 47-7. DMA_ADDR Register**

**Table 47-8. DMA_ADDR Fields**

| Field | Description |
|---|---|
| DMA_ADDRESS[31:3] | Base address of every field of picture in memory used by DMA.<br>Rewrite only after receiving DMA done interrupt and before arming DMA. The lowest 3 bits of DMA_ADDRESS cannot be set. It is always 3'b0.<br>See section 47.4.2/47-1584 for more details. |

### 47.2.3.7 DMA_INC

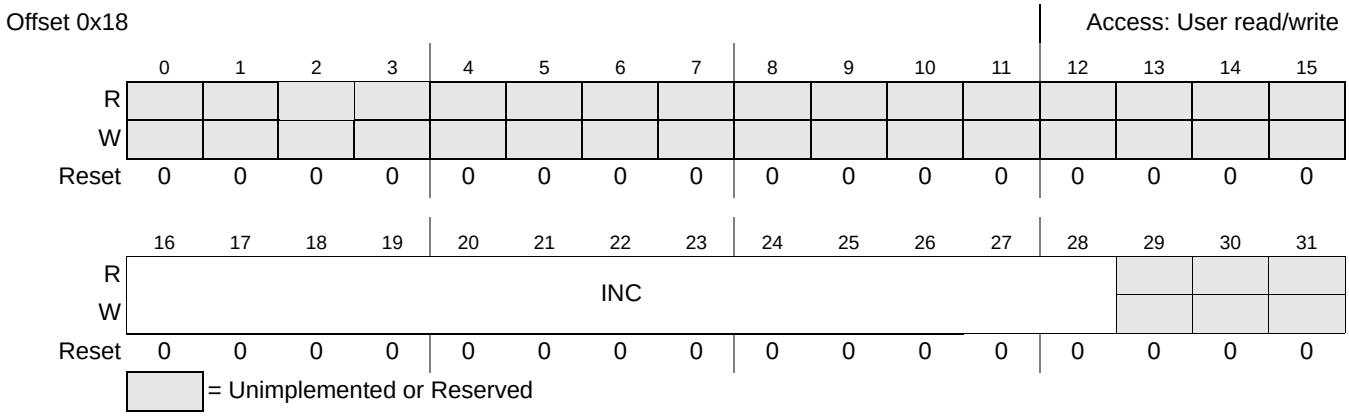



**Figure 47-8. DMA_INCREMENT Register**

**Table 47-9. DMA_INC Fields**

| Field | Description |
|---|---|
| INC | Value of this field should be zero or memory size that one active line occupies in memory. It will be added to the memory mapped rounded address at the end of every line. See section 47.3.9/47-1581.<br>Memory size of one active line depends on line pixel number. It is<br>• PIXEL_COUNT[15:2] + \|PIXEL_COUNT[1:0] when MODE32BIT=0;<br>• PIXEL_COUNT[15:1] + PIXEL_COUNT[0] when MODE32BIT=1;<br>It shall only be configured when DMA is inactive, during vertical blanking.<br>See section 47.4.2/47-1584 for more details. |

### 47.2.3.8 INVSZ

Offset 0x1C — Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LINEC | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PIXELC | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 47-9. INVSZ Register**

**Table 47-10. INVSZ Fields**

| Field | Description |
|---|---|
| LINEC | Expected number of active lines in each input video field/frame.<br>It shall only be configured when DMA is non-active, during vertical blanking. See section 47.4.2/47-1584 for more details.<br>If more lines are found during data receive part, a too many lines error interrupt is generated when ERROR_IRQ is set. Redundant lines are discarded.<br>If less lines are found during data receive part, a not enough lines error interrupt is generated when ERROR_IRQ is set. |
| PIXELC | Expected number of active pixels in each input video line, it shall be integer multiply of 4.<br>It shall only be configured when DMA is non-active, during vertical blanking. See section 47.4.2/47-1584 for more details.<br>If more pixels are found during data receive part, a line too long error interrupt is generated when ERROR_IRQ isset. Redundant pixels are discarded.<br>If less pixels are found during data receive part, a line too short error interrupt is generated when ERROR_IRQ is set. |

### 47.2.3.9 HPALRM

Offset 0x20 — Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ALARM | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 47-10. HPALRM Register**

**Table 47-11. HPALRM Fields**

| Field | Description |
|---|---|
| ALARM | High priority alarm threshold. When FIFO_FILL is higher than this value, high priority bus request will be asserted. |

### 47.2.3.10 ALPHA

offset 0x24 — read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | ALPHA[7:0] | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

= Unimplemented or Reserved

**Figure 47-11. ALPHA Register**

**Table 47-12. ALPHA Fields**

| Field | Description |
|---|---|
| ALPHA[7:0] | Alpha value used for picture blending. This register is configured during vertical blanking and used from the next video field. |

### 47.2.3.11 HFACTOR

Offset 0x28 — Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | FACTOR | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 47-12. FACTOR_H Register**

**Table 47-13. FACTOR_H Fields**

| Field | Description |
|---|---|
| FACTOR | Down scaling factor at horizonal direction.<br>FACTOR[10:8] is used as integer part of the factor.<br>FACTOR[7:0] is used as fractional part of the factor. |

### 47.2.3.12 VFACTOR

Offset 0x2C — Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | FACTOR_V | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 47-13. VFACTOR Register**

**Table 47-14. VFACTOR Fields**

| Field | Description |
|---|---|
| FACTOR | Down scaling factor at vertical direction.<br>FACTOR[10:8] is used as integer part of the factor.<br>FACTOR[7:0] is used as fractional part of the factor. |

### 47.2.3.13 VID_SIZE

offset 0x30 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LINEC | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PIXELC | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 47-14. VID_SIZE Register**

**Table 47-15. VID_SIZE Fields**

| Field | Description |
|---|---|
| LINEC | Expected number of lines in each output video frame after down scaling. |
| PIXELC | Expected number of pixels in each output video line after down scaling. It shall be multiply of 2 in 32-bit output mode, and multiply of 4 in 16-bit output mode. |

### 47.2.3.14 LUT_ADDR

Offset 0x34 Access: User read/write

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

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

= Unimplemented or Reserved

**Figure 47-15. LUT_ADDR Register**

**Table 47-16. LUT_ADDR Fields**

| Field | Description |
|---|---|
| ADDR | Current address pointer of the B/C adjust look-up-table. Value of this register increments (by 4) automatically at the end of each LUT_DATA write operation. This function allows fast update to the whole look-up-table, to the table of one color component, or even to any random field of the table. Note: LUT_ADDR reflects correct address only when clock of B/C adjust block is valid. |

### 47.2.3.15 LUT_DATA

Offset 0x38 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA[31:16] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DATA[15:0] | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 47-16. LUT_DATA Register**

**Table 47-17. LUT_DATA Fields**

| Field | Description |
|---|---|
| DATA | B/C adjust look-up-table data entry. Data in this register is actually written/read to/from the address pointed by the current LUT_ADDR value in the table.<br>Note: DATA reflects correct data value only when clock of B/C adjust block is valid. |

## 47.3 Functional Description

The VIU2 accepts ITU-R BT.656 compatible video stream on its parallel interface, decodes it and optionally performs processes like down-scaling, brightness and contrast adjust, YUV to RGB conversion, and de-interlace (weaving), and then stores the result video stream to the system memory which can then be displayed by a display controller, or post processed.

Functions of the VIU2 are designed in a way that they can be flexibly enabled or disable by software. But there are a few limitations as listed below:

- The down-scaler works on YUV 4:2:2 format, so to enable the down-scaler ITU decoder shall be configured in YUV 4:2:2 mode.
- To use the down-scaler, progressive video input shall be used for display quality's sake, and the de-interlace shall be disabled.

### 47.3.1 ITU656

The ITU-R BT.656-4 recommendation describes the means of interconnecting digital television equipment operating on the 525-line or 625-line standards and combines with the 4:2:2 encoding parameters as defined in the ITU-R BT.601 recommendation.

The data stream structure on ITU-R BT.656-4 interface is shown in Figure 47-17. There are two timing reference signals, one at the beginning of each video data block (start of active video, SAV) and one at the end of each video data block (end of active video, EAV).

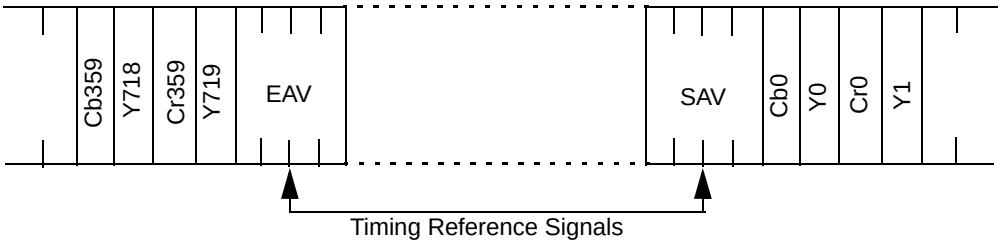



**Figure 47-17. Interface Data Stream of ITU-R BT.656-4**

Each timing reference signal consists of a four-word sequence in the following format: FF 00 00 XY. Values are expressed in hexadecimal notation. Value FF and 00 are reserved to be used in the timing reference signals.

The first three words are a fixed preamble. The fourth word contains information defining field 2 identification, the state of field blanking, and the state of line blanking. The assignment of bits within the timing reference signal is shown in Table 47-18.

**Table 47-18. Video Timing Reference Codes**

| Data Bit Number | First Word (FF) | Second Word (00) | Third Word (00) | Fourth Word (XY) |
|---|---|---|---|---|
| 9(MSB) | 1 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | F(0: field 1, 1: field 2) |
| 7 | 1 | 0 | 0 | V(0: elsewhere, 1: field blanking) |
| 6 | 1 | 0 | 0 | H(0: in SAV, 1: in EAV) |
| 5 | 1 | 0 | 0 | P3 |
| 4 | 1 | 0 | 0 | P2 |
| 3 | 1 | 0 | 0 | P1 |
| 2 | 1 | 0 | 0 | P0 |
| 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |

**NOTE**

When 8-bit interface is used, the values of bits D1 and D0 are not defined. These signals should be left disconnected.

In above table, bits P0, P1, P2, P3 have states dependent on the states of the bits F, V and H. At the receiver side this arrangement permits one-bit errors to be corrected and two-bit errors to be detected.

Note that when used in progressive video mode (instead of interlaced mode), where frames are transfered versus odd and even fields, the F bit can be either ignored or considered as indication of frames.

Refer to the ITU-R BT.656-4 recommendation for more details.

### 47.3.2 Input Synchronizer

The Synchronizer block (1) captures ITU656 protocol signals from the interface, and synchronizes them to the IPG_CLK domain.

### 47.3.3 ITU Decoder

The ITU Decoder block (2) detects the ITU656 timing reference signal (consists of a four-word sequence in the following format: FF-00-00-XY) and extracts HSYNC, VSYNC, field number signals and video data from the ITU data stream. The format of active pixel data from the ITU stream is YUV 4:2:2. The decoder block can directly send out this YUV 4:2:2 data, or interpolate it to YUV 4:4:4 format and send

out. This is determined by the MODE444 field in the SCR register (47.2.3.1/47-1565). This bit shall not be set if the down-scaler is enabled, because the down-scaler works on YUV 4:2:2 data format.

### 47.3.4 Down Scaling

The Down-scaler block (3) performs down-scaling to the incoming video stream. It's able to scale down the input video stream by a fractional scaling ratio up to 1/8. The down-scaler block can be enabled/disabled by software via the SCALER_EN bit in the SCR register. To use the down-scaler video data extracted from ITU decoder shall be in YUV 4:2:2 format.

The down-scaler uses a bi-linear filter with simple linear interpolation between the two nearest neighbors, on both horizontal and vertical direction. Scaling is firstly done on the horizonal direction (called XScale), and then the vertical direction (called YScale). Different scaling factors are supported on horizonal direction and vertical direction.

Assuming the scaling factors are (factor_h, factor_v), in theory for every pixel (x, y) of the scaled picture the coordinates of the corresponding pixel in the incoming picture (x1, y1) can be calculated by multiplying the coordinates with the scaling factors, say (x1 = x * factor_h, y1 = y * factor_v). Now because the scaling factors can be fractions, the coordinates (x1, y1) are not integer anymore, they are fractional values. The scalers use the integer part of (x1, y1) to find the two neighboring pixels from the incoming picture as input to the filters, and the fractional part to derive the weighting factors for interpolation.

The scaling factors, (factor_h, factor_v), are both 11 bit with the lower 8 bit as the factional part and the highest 3 bit the integer part. It's capable of scaling the input picture by up to 8, in steps of 1/256. It's by 8 if the factor is programmed as all zeros.

Instead of using multipliers to calculate (x1, y1), two 20-bit phase accumulator is used to step through the source pixels/or lines, where the lower 8 bit is the fractional part. For every output pixel/or line the phase adder is added to the accumulator. With the 12 bit integer part of the accumulator, up to 4096 source pixels/or lines can be supported.

So for each target pixel, the current accumulator position is used to determine how the pixel is going to be produced. As an example, accumulator value 0x123.3a means the step position is between pixel[0x123] and pixel[0x124], and the output pixel will be calculated by (pixel[0x124] * (0x100 - 0x3a) + pixel[0x123] * 0x3a) >> 8. The same concept is used for horizontal and vertical scaling.

Because ot the vertical scaling, a line buffer is used for each color component to store the data from horizontal scaling, it stores one line of data. Size of the line buffer determines the maximum video output size after scaling

The scaling factors are programmed via the HFACTOR (47.2.3.11/47-1573) and VFACTOR (47.2.3.12/47-1573) registers. And video size after scaling is programmed via the VID_SIZE register (47.2.3.13/47-1574).

Three scalers are instantiated in the down-scaler block, one for each component.

**NOTE**

The scaled pixel count per line shall be integer multiply of 2 in 32-bit output mode, or 4 in 16-bit output mode. The user shall divide the input pixel count by the horizontal scaling factor and truncate the result to multiply of 2 or 4.

**NOTE**

To achieve good display quality, progressive video input shall be used if down-scaling is needed.

### 47.3.5 Brightness and Contrast Adjust

The B/C Adjust block (4) performs brightness and contrast adjustment to the input video stream via three internal look-up-tables, one table per color component. Each table contains 256 8-bit entries, it maps every incoming pixel to the value of one of it's entries according to the original value of the pixel. The component is then replaced by the value in the table. If the incoming stream contains 10 bits per component, then only the 8 most significant bits are mapped to the look-up table. This feature allows the user to adjust brightness and/or contrast of the incoming picture according to three arbitrary adjustment curves, one adjustment curve per color component.

To use this feature, the B/C adjust look-up-table shall be programmed in software in the following format.

**Table 47-19.**

| Color Component | BC LUT Offset | Local Address [1:0] | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 10 | 11 |
| Y | 0x000 | $BC0_Y$ | $BC1_Y$ | $BC2_Y$ | $BC3_Y$ |
| | ..... | | | | |
| | 0x0FC | $BC252_Y$ | $BC253_Y$ | $BC254_Y$ | $BC255_Y$ |
| U | 0x100 | $BC0_U$ | $BC1_U$ | $BC2_U$ | $BC3_U$ |
| | ..... | | | | |
| | 0x1FC | $BC252_U$ | $BC253_U$ | $BC254_U$ | $BC255_U$ |
| V | 0x200 | $BC0_V$ | $BC1_V$ | $BC2_V$ | $BC3_V$ |
| | ..... | | | | |
| | 0x2FC | $BC252_V$ | $BC253_V$ | $BC254_V$ | $BC255_V$ |

**Figure 47-18. B/C Adjust Look-up-table Format**

Two registers are provided to program the look-up-table, they are LUT_ADDR (47.2.3.14/47-1574) and LUT_DATA (47.2.3.15/47-1575). LUT_ADDR is the current address pointer, which always points to the current table offset to be programmed. It can be set by software, and it increases (by 4) automatically when each word is written to the table, and it falls back to 0x000 if the current value is 0x2FC and the last word is written to the table. LUT_DATA is the table data entry, data written to this register will be stored into the table, to the address pointed to by LUT_ADDR. This register shall only be written by 4-byte word. With combination of these two registers user can program the B/C look-up-table conveniently, either

program the whole table, or reprogram the table for one color component only, or even change one single arbitrary word in the table.

The B/C adjust can be enabled/disabled by software via the BC_EN bit in the STATUS_CONFIG register (47.2.3.1/47-1565).

### 47.3.6 YUV to RGB Conversion

The YUV2RGB block (5) is used to convert YUV (4:2:2 or 4:4:4) to RGB (888 or 565). The coefficients of the YUV to RGB conversion matrix are programmed via four registers (47.2.3.2/47-1567, 47.2.3.3/47-1568, 47.2.3.4/47-1569, 47.2.3.5/47-1569). When the input is YUV 4:2:2, it's interpolated to YUV 4:4:4 first.

### 47.3.7 Round and Dither

In RGB565 output mode, when pixel data is converted from RGB888 to RGB565 the image is anamorphic more or less, because of losing color information conveyed by the least significant two or three bits of the original color components, which are dropped. VIU2 block provides two simple algorithms, round and dither, to compensate this color information loss.

#### 47.3.7.1 Round

In round mode, VIU2 will round in 1 to LSB if the decimal fraction is bigger than 0.5 and ignore the smaller fraction when ROUND_ON is set in the STATUS_CONFIG register (47.2.3.1/47-1565).

#### 47.3.7.2 Dither

Dither is a little more complex but better than round for recovering image. It's a statistical compensation algorithm. It doesn't render all pixels with the same grey or color level, but some with the lower one, and some with a color level of 1 LSB more. The selection of adding one LSB or not depends on the position of the pixel on the screen.

Figure 47-19 shows the implementation of dither in the VIU2 block.

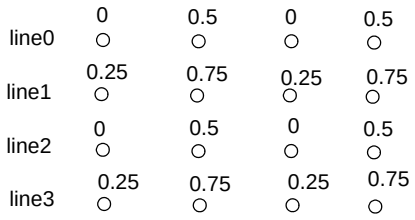



**Figure 47-19. Dither Implementation**

The number above the pixel position in the diagram is the compensation value for this pixel. When pixels have a value of 0.25, they are rendered 0 in 75% of the pixels and 1 in 25% of pixels. This averages to 0.25. Similarly, pixel values of 0.5, 0.75, and 1.0 are rendered 50%, 75% and 100% of the pixels as 1. For human eyes, this rendering result of dither makes the holistc image smoother and closer to the original one.

## 47.3.8 Output Formatter

The Output Formatter block (6) accepts YUV or RGB data from the YUV2RGB block, arranges them in expected format, and then sends it to the DMA engine. The FORMAT_CTRL field in STATUS_CONFIG register controls how the VIU2 stores data to system memory, see Figure 47-20 for details.

**Figure 47-20. VIU2 Output Data Stream Format**

| Pixel Format | FORMAT_CTRL[1] | Local Address [2:0] | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| RGB 565 | 000 | R0[7:3], G0[7:2], B0[7:3] | | R1[7:3], G1[7:2], B1[7:3] | | R2[7:3], G2[7:2], B2[7:3] | | R3[7:3], G3[7:2], B3[7:3] | |
| | 001[2] | G0[4:2], B0[7:3], R0[7:3], G0[7:5] | | G1[4:2], B1[7:3], R1[7:3], G1[7:5] | | G2[4:2], B2[7:3], R2[7:3], G2[7:5] | | G3[4:2], B3[7:3], R3[7:3], G3[7:5] | |
| RGB 8888 | 000 | A0 | R0 | G0 | B0 | A1 | R1 | G1 | B1 |
| | 001 | A0 | B0 | G0 | R0 | A1 | B1 | G1 | R1 |
| | 010 | R0 | G0 | B0 | A0 | R1 | G1 | B1 | A1 |
| | 011 | B0 | G0 | R0 | A0 | B1 | G1 | R1 | A1 |
| YUV 4:2:2 | 000 | U0 | Y0 | V0 | Y1 | U2 | Y2 | V2 | Y3 |
| | 001 | Y0 | U0 | Y1 | V0 | Y2 | U2 | Y3 | V2 |
| YUV 4:4:4 | 000 | dummy | Y0 | U0 | V0 | dummy | Y1 | U1 | V1 |
| | 001 | dummy | Y0 | V0 | U0 | dummy | Y1 | V1 | U1 |
| | 010 | dummy | U0 | V0 | Y0 | dummy | U1 | V1 | Y1 |
| | 011 | dummy | V0 | U0 | Y0 | dummy | V1 | U1 | Y1 |
| | 100 | Y0 | U0 | V0 | dummy | Y1 | U1 | V1 | dummy |
| | 101 | Y0 | V0 | U0 | dummy | Y1 | V1 | U1 | dummy |
| | 110 | U0 | V0 | Y0 | dummy | U1 | V1 | Y1 | dummy |
| | 111 | V0 | U0 | Y0 | dummy | V1 | U1 | Y1 | dummy |

1 All values not shown in the table shall be considered as reserved and shall not be used.

2 This format is basically intended for data communication with any little-endian peripheral in the system, assuming big-endian system memory is used.

## 47.3.9 DMA and De-interlace

The DMA engine block (7) functions as the FIFO controller and DMA engine. It stores the data from the output formatter block into a FIFO and then writes them to system memory via the IPM interface.

VIU2 block has an embeded DMA. When video data leaves the output formatter and is placed into a 256 × 64 bit FIFO, it waits for transfering to memory by this internal DMA.

After doing some necessary register configuration, such as coefficients, INVSZ and DMA_ADDR, user can activate DMA by setting the DMA_ACT of the status and configuration register. But note that DMA_ACT can be configured only during vertical blanking since VIU2 block won't transfer a fragment

of video field to memory. If it’s configured during field active time, DMA transfer can not be started and an error interrupt will be asserted.

The VIU2 block asserts a transfer request when data in FIFO is enough for one transfer. Normally one transfer conveys 32 bytes from FIFO to memory. While at the end of a line, all remaining data of this line is transfered, though it may not be 32 bytes.

VIU2 also provides a simple way to de-interlace for interlaced/or pseudo interlaced video image. It is implemented by setting DMA_ADDR and DMA_INC registers.

Figure 47-21 shows the implementation of de-interlace. Value of DMA_INC is added to the rounded address at the end of every active line. So, when DMA_INC is zero, pixel data is stored in memory line by line, meaning de-interlace is off, as shown in figure (1) and (2) ; otherwise when DMA_INC equals to one-line memory mapped pixel size, and DMA_ADDR(2) = DMA_ADDR(1) + DMA_INC, odd field and even field will be merged into one frame in memory, as shown in figure (3).

Here DMA_ADDR(1) means base address of field 1, and DMA_ADDR(2) means base address of field 2. Memory mapped line size means memory size that occupied by one active line pixels. It depends on pixel number of one line and video data format (RGB888 or RGB565). See register description in section 47.2.3.7/47-1570.

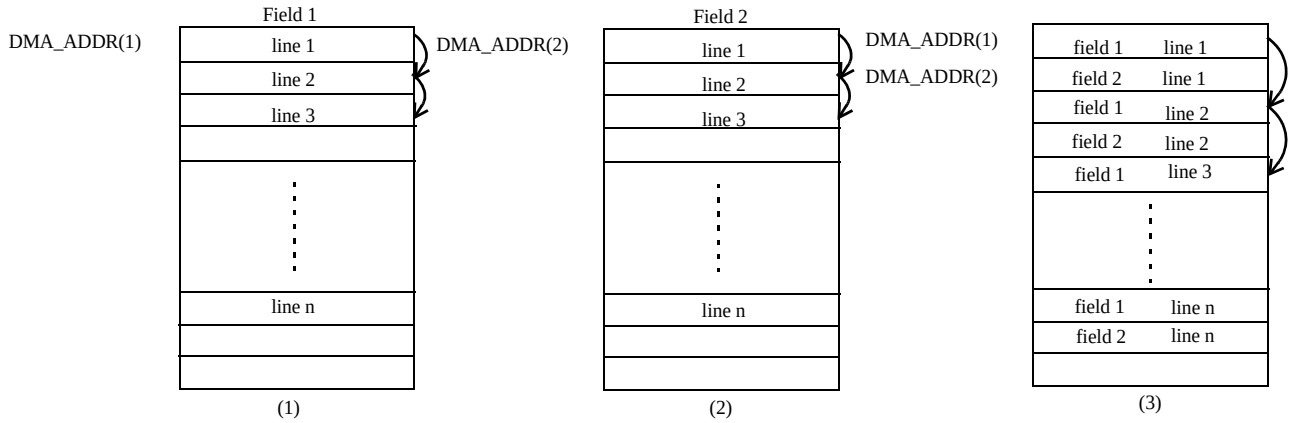



**Figure 47-21. Implementation of De-interlace**

## 47.3.10 Error Case

Normally, user should provide a standard and totally ITU-compatible video stream to VIU2 block. However, it’s difficult to avoid unexpected errors all the time. VIU2 can manage error cases like ECC error, line too long, line too short, too many lines, or not enough line in a field, even FIFO overflow.

- ECC error: ITU stream provides 4-bit error correcting code P[3:0] in its SAV and EAV. It is decoded in VIU2 to use correct field number, horizonal sync and vertical sync bits. It can correct one bit error and find two bits error. When ECC error is found, an interrupt is asserted.
- Line too long error: When pixels of active line is longer than PIXELC, a line too long error interrupt is asserted and redundant pixels are discarded.
- Too many lines error: When active lines of a field are bigger than LINEC, a too many lines error interrupt is asserted and redundant lines is discarded.

- Line too short error: When pixels of active line is less than PIXELC, a line too short error interrupt is asserted.
- Not enough line error: When active lines of a field is less than LINEC, a not enough line error interrupt is asserted.
- FIFO overflow error: If system bus is blocked for a long time, video data is stored in FIFO and makes FIFO overflow. When FIFO is overflow, an interrupt is asserted, and the coming data is discarded until FIFO works normally again. Current field is jumbled. However, VIU2 recovers to work at the next field if bus is unblocked at that time.
- FIFO underflow: When FIFO is read when it's empty, a FIFO underflow error interrupt is asserted. Normally, this error shouldn't occur.

VIU2 can manage above error cases to a certain extent. However, when it doesn't recover to work, user should write the SOFT_RESET bit of the status and configuration register to reset VIU2 block.

## 47.4 Initialization/Application Information

### 47.4.1 Initialization Information

When VIU2 block comes out of reset, software should implement the following steps to start this block.

1. Program STATUS_CONFIG register (47.2.3.1/47-1565) to set the VIU2 to desired operation mode
   - To enable YUV 4:2:2 to 4:4:4 interpolation in ITU decoder, set the MODE444 bit[1]
   - To enable down-scaler, set the SCALER_EN bit
   - To enable B/C adjust, set the BC_EN bit
   - To enable RGB565 output mode, set RGB_EN bit and clear MODE32BIT bit. Optionally set DITHER_ON or ROUND_ON bit to enable dither or round[2]
   - To enable ARGB8888 output mode, set RGB_EN and MODE32BIT bit
   - To enable YUV output mode, clear RGB_EN bit
2. Set the FORMAT_CTRL field so that VIU2 outputs data in correct format. Configure the input video size via the INVSZ register (47.2.3.8/47-1571).
3. If want to use RGB output, configure YUV to RGB conversion coefficients (47.2.3.2/47-1567, 47.2.3.3/47-1568, 47.2.3.4/47-1569, 47.2.3.5/47-1569) or use the default values after reset.
4. If want to use the down scaling function, program the down scaling factors (47.2.3.11/47-1573, 47.2.3.12/47-1573), and destination video size after scaling (47.2.3.13/47-1574).
5. If want to use the B/C adjust function, program the B/C adjust look-up-table via the LUT_DATA (47.2.3.15/47-1575) and LUT_ADDR (47.2.3.14/47-1574) registers.
6. Configure the HPALRM (47.2.3.9/47-1572) and ALPHA (47.2.3.10/47-1572) registers if necessary.
7. Set the VSYNC_EN and/or FIELD_EN bits in STATUS_CONFIG register to enable vsync or field interrupt. Meanwhile, disable error interrupt.

1. MODE444 bit shall not be set when the down-scaler is enabled.
2. If DITHER_ON or ROUND_ON are both set, only round will be enabled.

8. When software receives vsync interrupt and/or field interrupt, read FIELD_NO bit of STATUS_CONFIG register[1].
9. According to FIELD_NO bit, program the DMA_ADDR register (47.2.3.6/47-1570). This is the field start address in system memory, or frame start address in progressive video input mode.
10. If want to use the de-interlace (weaving) function, program the line start address offset value in the DMA_INC (47.2.3.7/47-1570) register[2].
11. Clear error status first if necessary.
12. Write DMA_ACT bit of STATUS_CONFIG register to start FIFO and DMA transfer. This operation actually starts the VIU2 to operate.

## 47.4.2 Application Information

Normally, the user shall not change the register values of the function enable bits, input/output video size, color conversion coefficients, DMA_INC, MODE32BIT, scaling factors and B/C adjust loop-up-table contents after VIU2 is started up. When video input source is changed, user should reset VIU2 and re-configure related registers.

### 47.4.2.1 Register Configuration Timing Window

As mentioned above, dynamic configuration on registers is not recommended because it may cause error if it's not configured in a certain timing window, especially for INVSZ and DMA_INC.

Register configuration timing window is shown in below diagram. All registers, except SOFT_RESET bit, are recommended to be configured during vertical blanking, after DMA transfer is done and field identification bit is changed (field interrupt is asserted).

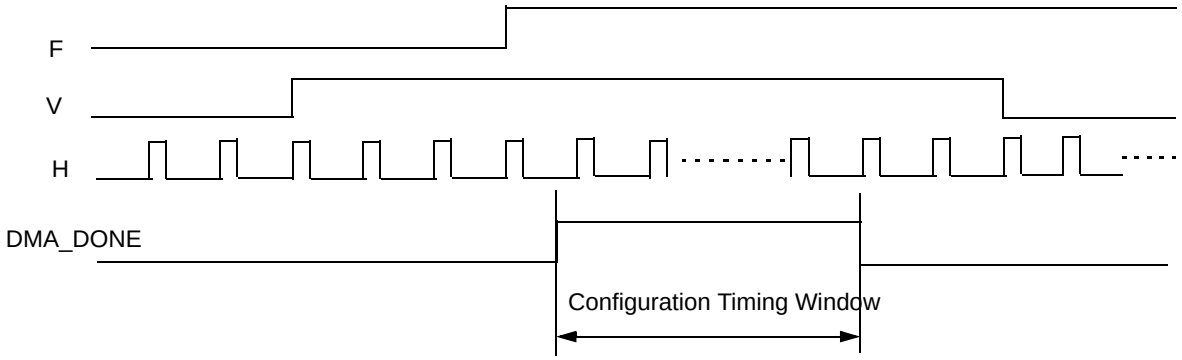



**Figure 47-22. Register Configuration Timing Window**

1. Reading the FIELD_NO bit is optional, especially in progressive video input mode.
2. Progressive video input shall be used when down scaling is enabled for better display quality.

# Chapter 48
# Voltage Regulators and Power Supplies

## 48.1 Introduction

This device includes three on-chip voltage regulators for power management and distribution, allowing low-power operation and optimization of power consumption.

On this device, the general purpose inputs and outputs (GPIO) are arranged in several banks with separate external GPIO power supply pins, allowing groups of GPIO to operate at different supply voltages.

## 48.2 Power-up sequencing

The preferred power-up sequence for MPC5645S is as follows:

1. Generic IO supplies or noise-free supplies, consisting of:
   - VDDA
   - VDDE_A
   - VDDE_B
   - VDDM
   - VDD_DR
   - VDD33_DR
   - VDDPLL
2. All 3.3V supplies (VDDE_B & VDD33_DR) should be ramped up first, and then the rest of the I/O supplies should be ramped up (VDDA, VDDE_A, VDDM, VDD_DR).
3. VDDR, the regulator input supply, should be the last supply to ramp up; all supplies can be ramped up together as long as VDDR is included. So all 5V supplies should be ramped up after the 3.3V supplies, and if all the supplies are of the same level, they can be ramped up together as well.
4. LV supply (VDD12). If Vreg is in bypass mode and the core supply (1.2V) is supplied externally, then this should be the last supply given.

**NOTE**

For DDR, the 3.3 V supply (VDD33_DR) should come before VDD_DR.

**NOTE**

Vreg bypass mode is for factory testing only.

This sequence ensures that when VREG releases its LVDs, the IO and other HV segments are powered properly. This is important because MPC5645S doesn't monitor LVDs on IO HV supplies.

## 48.3 Voltage regulators

The internal voltage regulators are used to provide a 1.2 V digital supply to the internal logic of the device. The main/input supply is 3.3 V to 5.0 V ± 10% and the digital/regulated output supply is 1.2 V ± 10%. The voltage regulator used in this device comprises three regulators.

- High power or main regulator (HPREG) requiring an external NPN ballast transistor.
- Low power regulator (LPREG)
- Ultra low power regulator (ULPREG)

The HPREG and LPREG regulators are switched off in STANDBY mode to save power consumption by the regulator itself. During STOP and HALT modes only, the HPREG regulator may be switched off. The ULPREG regulator is always kept on.

The internal voltage regulators require an external capacitance to be connected to the 1.2 V supply pins in order to provide a stable low voltage digital supply to the device. Capacitances should be placed on the board as near as possible to the associated pins.

The regulator has two digital domains, one for the main regulator (HPREG) and the low power regulator (LPREG) called the “High Power domain”, and one for the ultra low power regulator (ULPREG) called the “Standby domain”. For each domain there is a low voltage detector (LVD) for the 1.2 V output voltage (ULVDD and MLVDD[1]). Additionally, there are two low voltage detectors for the main/input supply with different thresholds, one at 3.3 V (ULVDM), the other at 5 V (ULVDM5).

1. MLVDD stands for Main Regulator LV supply (main regulator means HPREG).

### 48.3.1 Block diagram

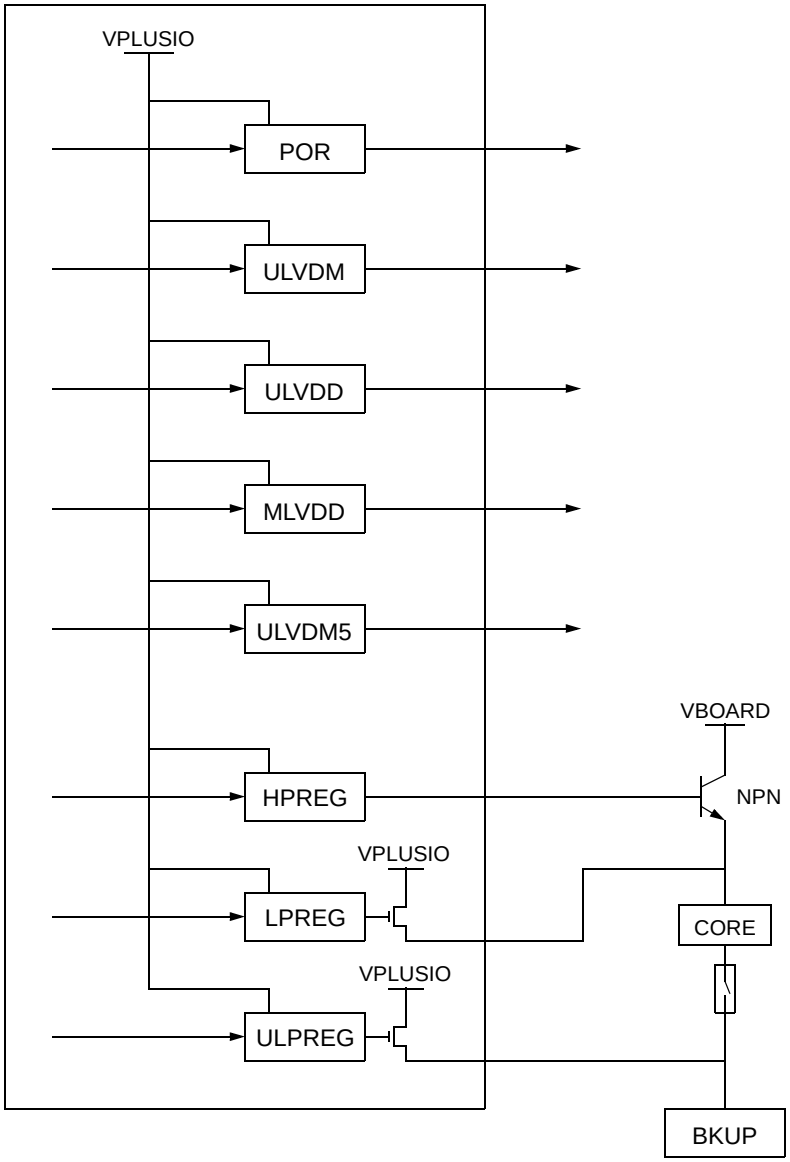



**Figure 48-1. Voltage Regulator Diagram**

### 48.3.2 External signals

Table 48-1 provides an overview of the voltage regulator external signals.

Table 48-1. Voltage Regulator External Signals

| Name T | ype | Voltage | Description |
|---|---|---|---|
| VDDR | Supply | 3.0 V - 5.5 V | Power Supply for the voltage regulators |
| VSSR | Ground | - | Ground supply for digital core and voltage regulators |
| VRC | Output | - | Regulator drive for external NPN transistor base |

### 48.3.3 Detailed signal descriptions

#### 48.3.3.1 VDDR

VDDR is the 3.3 V to 5 V supply for the voltage regulators and LVD blocks. See the *MPC5645S Microcontroller Data Sheet* for details of recommended decoupling capacitance on this pin.

#### 48.3.3.2 VRC

VRC is the 1.2 V regulator output that drives the base of the external NPN ballast transistor.

## 48.4 Memory map and register definition

Table 48-2. Voltage Regulator Memory Map

| Address Re | gister | Location |
|---|---|---|
| VREG_BASE + 0x0000 | VREG_CTL - Voltage Regulator Control Register | on page 1588 |

### 48.4.1 Voltage Regulator Control Register (VREG_CTL)

Address Offset: 00h — Access: User read/write

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

| | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5V_LVD_MASK |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Figure 48-2. Voltage Regulator Control Register (VREG_CTL)

Table 48-3. VREG_CTL field descriptions

| Field De | scription |
|---|---|
| 5V_LVD_MASK | **5 V LVD MASK**: Mask bit for 5 V LVD from regulator<br>This is a read/write bit and must be unmasked by writing a ‘0’ by software to generate LVD functional reset request to MC_RGM for 5 V trip.<br>1 5 V LVD is masked<br>0 5 V LVD is not masked. |

## 48.5 Functional description

### 48.5.1 High Power or Main Regulator (HPREG)

The HPREG converts the 3.3 V to 5 V input supply voltage to the 1.2 V digital supply. The nominal target output is 1.2 V. The actual output will be in range of 1.08–1.32 V in the full current load range 0–400 mA after trimming.

The HPREG regulator requires an external NPN ballast transistor. Recommended transistors are NJD2873 or BCP68 from ON Semiconductor.

Stabilization of the HPREG is achieved using an external capacitance. The minimum recommended value is 600 nF with low ESR. Limit the series inductance per pad to less than 15 nH.

The regulator is enabled by the MVRON bits in the mode configuration registers in the MC_ME module. These bits allow the regulator to be optionally disabled in certain low power modes.

### 48.5.2 Low Power Regulator (LPREG)

The LPREG generates power for the chip in STOP mode, providing the output supply of 1.2 V. The control part of the regulator can be used to disable the low power regulator. This action is managed by MC_ME.

### 48.5.3 Ultra Low Power Regulator (ULPREG)

The ULPREG generates power for the standby domain as well as a part of the main domain. The control circuit of ULPREG can be used to disable the ultra low power regulator by SW. This action is managed by MC_ME.

### 48.5.4 Low Voltage Detectors (LVD) and Power On Reset (POR)

Four LVDs are available (see Figure 48-1).

- ULVDM for the 3.3 V to 5 V input supply with threshold at the 3.3 V level
- ULVDM5 for the 3.3 V to 5 V input supply with threshold at the 5 V level
- Two LVD_DIGs, ULVDD and MLVDD, for the 1.2 V output voltage

ULVDM and ULVDM5 sense the 3.3 V to 5 V power supply for the core digital logic, shared with the GPIO ring supply, and indicate when the 3.3 V to 5 V supply is stable. The threshold levels of ULVDM5 are factory-trimmed.

Two LVD_DIGs are provided in the design. One LVD_DIG is placed in the high power domain and senses the HPREG/LPREG output, indicating that the 1.2 V output is stable. The other LVD_DIG is placed in the standby domain and senses the standby 1.2 V supply level, indicating that the 1.2 V output is stable. The reference voltage used for all LVDs is generated by the low power reference generator and is factory-trimmed for LVD_DIG. Therefore, during the pre-trimming period, LVD_DIG exhibits higher thresholds whereas, after trimming, the thresholds come within the desired range. Power-down pins are provided for LVDs. When LVDs are powered down, their outputs are pulled high.

POR is required to initialize the chip during the voltage supply rise time. POR works only on the rising edge of the main supply voltage. To ensure that it functions during the following rising edge of the supply voltage, it is reset by the output of the ULVDM block when the main supply goes below the lower voltage threshold of ULVDM.

POR is asserted on power-up when the Vdd supply is above the minimum value of $V_{PORUP}$ (refer to the device data sheet for this value). It will be released only after the Vdd supply goes above $V_{PORH}$ (refer to the device data sheet for this value). Vdd above $V_{PORH}$ ensures that the power management module, including the internal LVD modules, are fully functional.

### 48.5.5 VREG digital interface

The voltage regulator digital interface provides the temporization delay at initial power-up and at exit from low-power modes. A signal, indicating that Ultra Low Power domain is powered, is used at power-up to release reset to temporization counter. At exit from low-power modes, the power-down for high power regulator request signal is monitored by the digital interface and used to release reset to the temporization counter. In both cases, on completion of the delay counter, an end-of-count signal is released; this is gated with an other signal indicating that the main domain voltage is fine, in order to release the VREGOK signal. This is used by MC_RGM to release the reset to the device. It manages other specific requirements, including the transition between high power or low power mode to ultra low power mode, avoiding a voltage drop below the permissible threshold limit of 1.08V.

The VREG digital interface also contains a control register to mask the 5 V LVD status from the voltage regulator at power-up.

## 48.6 GPIO power supply configuration

The GPIO pins on this device are organized in four separate banks. Each bank has associated VDD/VSS power supply pairs. The four banks of GPIO can be powered independently, allowing these banks to be run at different voltages. VDDA must always be at the same voltage as VDDE_A.

Table 48-4 describes the GPIO banks, their main functions, supply pins and associated port pins. For full details of GPIO functionality, refer to Chapter 3, “Signal Description.”

**Table 48-4. GPIO power bank supplies and functionality**

| GPIO bank | Available functions[1] | Supply pins | Port pins |
|---|---|---|---|
| Stepper Motor Bank | Stepper Motors [1:0], eMIOS<br>Stepper Motors [3:2], eMIOS<br>Stepper Motors [5:4] | VDDM, VSSM | PortD[7:0]<br>PortD[15:8]<br>PortE[7:0] |
| Analog Bank | ADC, 32 kHz SXOSC | VDDE_A, VSSE_A | PortC[15:0]<br>PortL[3:0] |
| DRAM Bank | DRAM Memory control, address and data pins | VDD_DR, VSS | DDR_ODT<br>DDR_CKE<br>DDR_CLK<br>DDR_CLKB<br>DDR_CS<br>DDR_RAS<br>DDR_CAS<br>DDR_WEB<br>DDR_A[0:15]<br>DDR_BA[0:2]<br>DDR_DQ[0:31]<br>DDR_DQS[0:3]<br>DDR_DM[0:3] |
| Digital Bank | DCU3, DCULite, DSPI, QuadSPI, FlexCAN, LINFlex, $I^2C$, eMIOS0, eMIOS1, PDI, VIU2, ADC-MUX, SGM, Nexus | VDDE_B, VSS | PortA[15:0]<br>PortB[15:0]<br>PortF[15:0]<br>PortG[12:0]<br>PortH[0:5]<br>PortJ[15:0]<br>PortK[11:0]<br>PortL[13:4]<br>PortM[13:0]<br>PortN[15:0]<br>PortP[7:0]<br>RESET |

1 Not all functions are available simultaneously. Refer to Chapter 3, “Signal Description.”

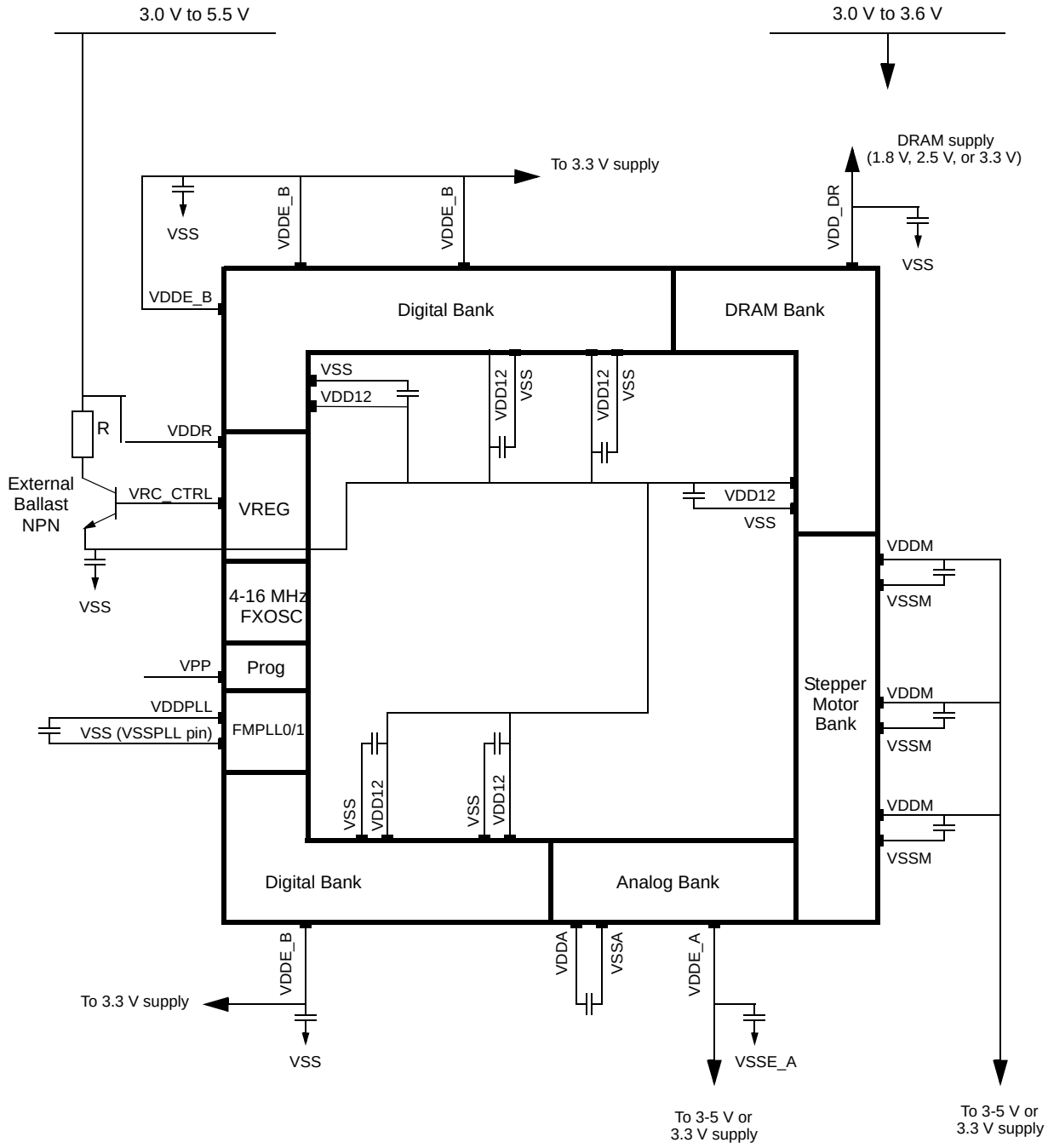



**Figure 48-3. Power supply configuration**

## 48.7 Power domain organization

To satisfy stringent requirements for current consumption in the different operational modes, this device is partitioned into different power domains. Organization into these power domains primarily means having separate power supplies that are separated from each other by the use of power switches. These

different separated power supplies are hence able to switch off power to certain regions of the device, to avoid even leakage current consumption in logic supplied by the corresponding power supply.

This device employs three primary power domains: power domains PD0 and PD1, and a RAM-only power domain PD2.See Chapter 2, “Memory Map for details of RAM locations in each domain.

Power domain organization and connections to the internal regulator are depicted in Figure 48-4.

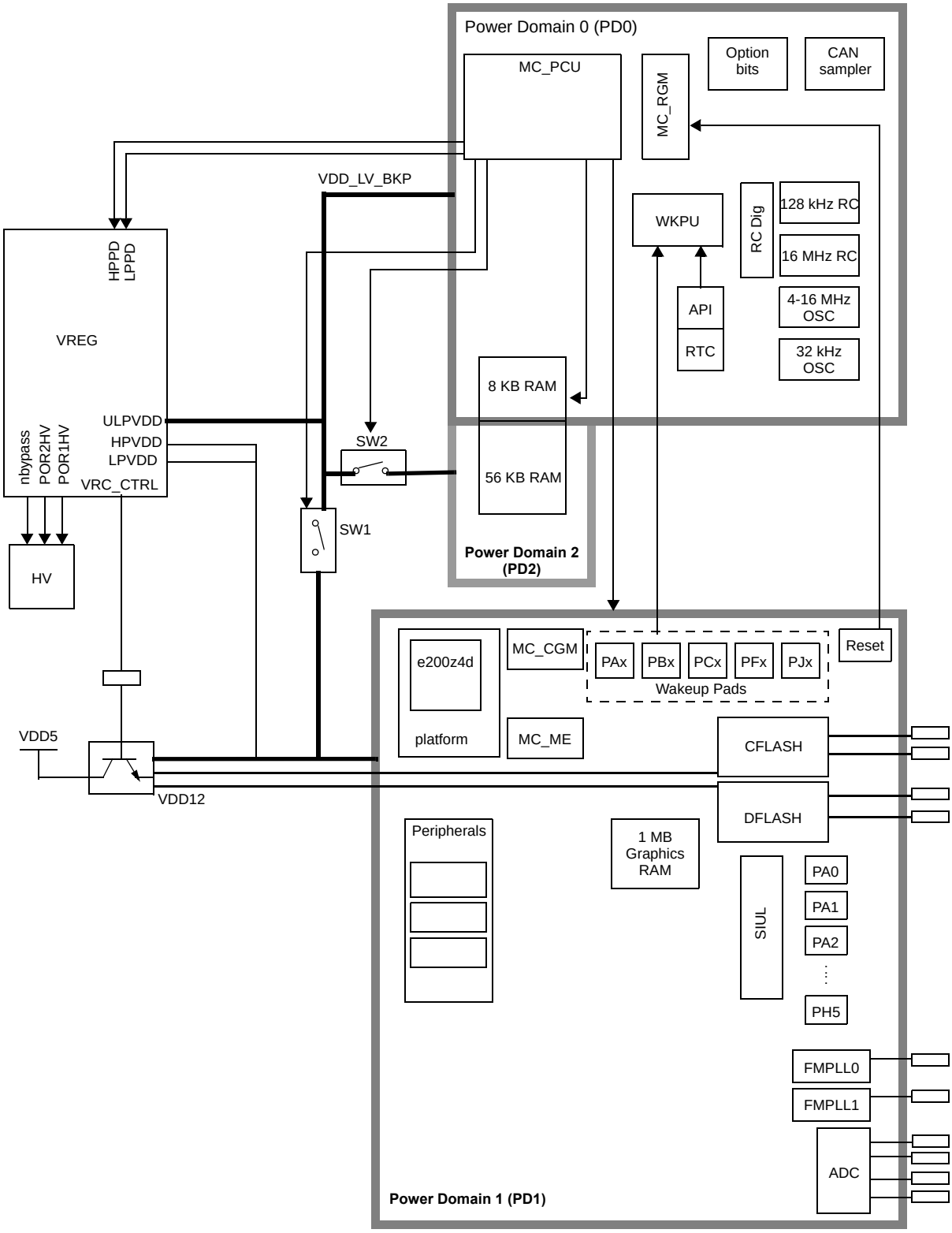



**Figure 48-4. Power domain organization**

# Chapter 49
# Wakeup Unit (WKPU)

## 49.1 Overview

The WKPU supports 24[1] external sources that can generate both interrupts and wakeup events, of which one can cause non-maskable interrupt requests. Figure 49-1 is a block diagram of the Wakeup Unit and its interfaces to other system components.

The WKPU operates separately from the external SIUL and its interrupt capability is independent from the SIUL external interrupt feature.

The wakeup lines are mapped to the interrupt vectors as shown in Table 49-1.

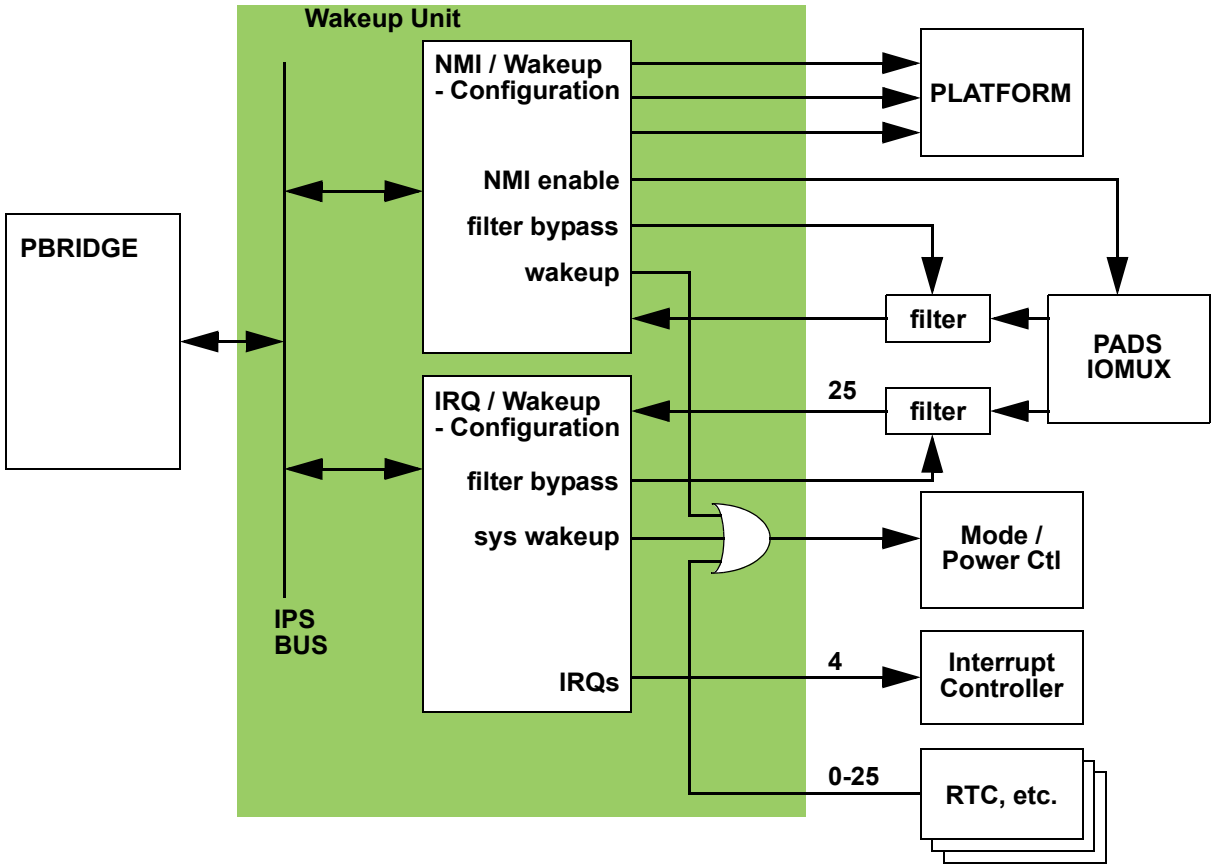



**Figure 49-1. Wakeup Unit block diagram**

1. Up to 23 external sources in the 416-pin package; up to 19 external sources in the 208-pin package; up to 15 external sources in the 176-pin package

**Table 49-1. Wakeup vector mapping**

| Wakeup vector | Wakeup number | Function | | | | Package | | |
|---|---|---|---|---|---|---|---|---|
| | | Port | #1 | #2 | #3 | 176 | 208 | 416 |
| 0 | WKUP0 | PB1 | CANRX_0 | — | — | x | x | x |
| | WKUP1 | PB3 | LIN_RXD_0 | — | — | x | x | x |
| | WKUP2 | PB4 | SCK_1 | MA0 | — | x | x | x |
| | WKUP3 | PB9 | SCK_0 | eMIOS1[20] | — | x | x | x |
| | WKUP4 | PB10 | CANRX_1 | I2S_D0 | — | x | x | x |
| | WKUP5 | PB12 | RXD_1 | eMIOS1[19] | CS2_0 | x | x | x |
| | WKUP6 | PC0 | AN[0] | — | — | x | x | x |
| | WKUP7 | PC10 | AN[10] | — | — | x | x | x |
| 1 | WKUP8 | PF0 | eMIOS1[19] | EVTO | DCULITE_B2 | x | x | x |
| | WKUP9 | PF2 | NMI | — | — | x | x | x |
| | WKUP10 | PF3 | eMIOS1[21] | MSEO | DCULITE_B4 | x | x | x |
| | WKUP11 | PF8 | SDA_0 | CS2_1 | RXD_1 | x | x | x |
| | WKUP12 | PJ4 | VIU[0] | eMIOS0[21] | eMIOS0[23] | x | x | x |
| | WKUP13 | PJ6 | VIU[2] | eMIOS0[19] | eMIOS0[15] | x | x | x |
| | WKUP14 | PK7 | RXD_2 | DCULITE_R2 | TCON8 | — | x | x |
| | WKUP15 | PL0 | AN[19] | CANRX_1 | SDA_1 | — | x | x |
| 2 | WKUP16 | PL2 | AN[17] | CANRX_0 | eMIOS1[22] | — | x | x |
| | WKUP17 | PL9 | SCK_2 | PDI_PCLK | eMIOS1[21] | — | x | x |
| | WKUP18 | PM3 | CANRX_2 | RXD_3 | TCON[4] | — | x | x |
| | WKUP19 | PM10 | RXD_2 | CANRX_2 | eMIOS0[16] | x | — | — |
| | WKUP20 | PN0 | DCULITE_HSYNC | — | TCON4 | — | — | x |
| | WKUP21 | PN2 | DCULITE_R0 | RXD_2 | VIU[0] | — | — | x |
| | WKUP22 | PN10 | DCULITE_G0 | RXD_3 | VIU[2] | — | — | x |
| | WKUP23 | PP2 | DCULITE_B0 | CANRX_2 | VIU[4] | — | — | x |
| 3 | WKUP24 | — | API | — | — | — | — | — |
| | WKUP25 | — | RTC | — | — | — | — | — |

## 49.2 Features

The Wakeup Unit supports these distinctive features:

- Non-maskable interrupt support with
  - 1 NMI source with bypassable glitch filter

- independent interrupt destination: non-maskable interrupt, critical interrupt, or machine check request
- edge detection

- External wakeup/interrupt support with
  - up to 4 system interrupt vectors for 26 interrupt sources
  - analog glitch filter per each wakeup line
  - independent interrupt mask
  - edge detection
  - configurable system wakeup triggering from all interrupt sources
  - configurable pullup
- On-chip wakeup support
  - up to 26 wakeup sources
  - wakeup status mapped to same register as external wakeup/interrupt status

## 49.3 External signal description

The WKPU has 24 external signal inputs. These can be used as system wakeup sources in certain power down modes. They can also be used as additional external interrupt sources in normal run mode. When used as external interrupt sources they operate independently from the external interrupt sources available on the SIUL although they may share a single pin.

These 24 external signal inputs include one signal input that can be used as a non-maskable interrupt source in normal run mode or a system wakeup sources in certain power down modes.

**NOTE**

Be aware that the Wake-up pins are enabled in ALL modes. Therefore, the Wake-up pins should be correctly terminated to ensure minimal current consumption. Any unused Wake-up signal input should be terminated by using an external pull-up or pull-down, or by internal pull-up enabled at WIPUER. Also, care must be taken on packages where the Wake-up signal inputs are not bonded. For these packages, the user must ensure the internal pull-ups are enabled for those signals not bonded.

## 49.4 Memory map and register description

This section provides a detailed description of all registers accessible in the WKPU module.

### 49.4.1 Memory map

Table 49-2 gives an overview on the WKPU registers implemented.

Table 49-2. WKPU memory map

| Address offset | Register | Abbreviation | Location |
|---|---|---|---|
| 0x0000 | NMI Status Flag Register | NSR | on page 1599 |
| 0x0004 - 0x0007 | Reserved | | |
| 0x0008 | NMI Configuration Register | NCR | on page 1599 |
| 0x000C - 0x0013 | Reserved | | |
| 0x0014 | Wakeup/Interrupt Status Flag Register | WISR | on page 1601 |
| 0x0018 | Interrupt Request Enable Register | IRER | on page 1601 |
| 0x001C | Wakeup Request Enable Register | WRER | on page 1602 |
| 0x0020 - 0x0027 | Reserved | | |
| 0x0028 | Wakeup/Interrupt Rising-Edge Event Enable Register | WIREER | on page 1602 |
| 0x002C | Wakeup/Interrupt Falling-Edge Event Enable Register | WIFEER | on page 1603 |
| 0x0030 | Wakeup/Interrupt Filter Enable Register | WIFER | on page 1603 |
| 0x0034 | Wakeup/Interrupt Pullup Enable Register | WIPUER | on page 1604 |
| 0x0038 - 0x03FFF | Reserved | | |

**NOTE**

Reserved registers will read as 0, writes will have no effect. If supported and enabled by the SoC, a transfer error will be issued when trying to access completely reserved register space.

## 49.4.2 Register description

This section describes in address order all the Wakeup Unit registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order. The numbering convention of register is MSB=0, however the numering of internal field is LSB=0, e.g. EIF[5] = WISR[26].

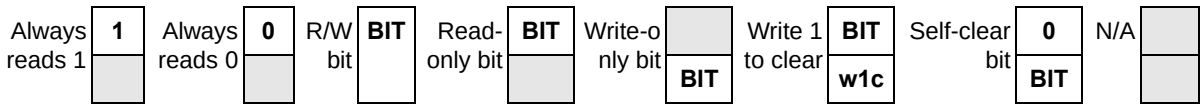



Figure 49-2. Key to Register Fields

### 49.4.2.1 NMI Status Flag Register (NSR)

This register holds the non-maskable interrupt status flags.

Address:0x0000 Access: User read/write (write 1 to clear)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NIF | NOVF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | w1c | w1c | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 49-3. NMI Status Flag Register (NSR)**

**Table 49-3. NSR Field Descriptions**

| Field D | escription |
|---|---|
| 0<br>NIF | NMI Status Flag<br>This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE or NFEE set), NIF causes an interrupt request.<br>1 An event as defined by NREE and NFEE has occurred<br>0 No event has occurred on the pad |
| 1<br>NOVF | NMI Overrun Status Flag<br>This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF value whenever an NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. If enabled (NREE or NFEE set), NOVF causes an interrupt request.<br>1 An overrun has occurred on NMI input<br>0 No overrun has occurred on NMI input |

### 49.4.2.2 NMI Configuration Register (NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

Address:
0x0008

Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NLOCK | NDSS | | NWRE | 0 | NREE | NFEE | NFE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Figure 49-4. NMI Configuration Register (NCR)**

**Table 49-4. NCR Field Descriptions**

| Field D | escription |
|---|---|
| 0<br>NLOCK | NMI Configuration Lock Register<br>Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset. Writing a 0 has no effect. |
| 1-2<br>NDSS | NMI Destination Source Select<br>00 Non-maskable interrupt<br>01 Critical interrupt<br>10 Machine check request<br>11 Reserved - no NMI, critical interrupt, or machine check request generated |
| 3<br>NWRE[x] | NMI Wakeup Request Enable<br>1 A set NIF bit or set NOVF bit causes a system wakeup request<br>0 System wakeup requests from the corresponding NIF bit are disabled |
| 5<br>NREE | NMI Rising-edge Events Enable<br>1 Rising-edge event is enabled<br>0 Rising-edge event is disabled |
| 6<br>NFEE | NMI Falling-edge Events Enable<br>1 Falling-edge event is enabled<br>0 Falling-edge event is disabled |
| 7<br>NFE | NMI Filter Enable<br>Enable analog glitch filter on the NMI pad input.<br>1 Filter is enabled<br>0 Filter is disabled |

## NOTE

Writing a ‘0’ to both NREE and NFEE disables the NMI functionality completely (i.e. no system wakeup or interrupt will be generated on any pad activity)!

### 49.4.2.3 Wakeup/Interrupt Status Flag Register (WISR)

This register holds the wakeup/interrupt flags.

**NOTE**

Not all bits are available in all packages.

Address: 0x0014 Access: User read/write (write 1 to clear)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | EIF25 | EIF24 | EIF23 | EIF22 | EIF21 | EIF20 | EIF19 | EIF18 | EIF17 | EIF16 | EIF15 | EIF14 | EIF13 | EIF12 | EIF11 | EIF10 | EIF9 | EIF8 | EIF7 | EIF6 | EIF5 | EIF4 | EIF3 | EIF2 | EIF1 | EIF0 |
| W | | | | | | | w1c | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 49-5. Wakeup/Interrupt Status Flag Register (WISR)**

**Table 49-5. WISR Field Descriptions**

| Field D | escription |
|---|---|
| EIF*x* | External Wakeup/Interrupt Status Flag *x*.<br>This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER[x]), EIF[x] causes an interrupt request.<br>1 An event as defined by WIREER and WIFEER has occurred<br>0 No event has occurred on the pad |

**NOTE**

Status bits associated with on-chip wakeup sources are located to the left of the external wakeup/interrupt status bits and are read only. The wakeup for these sources must be configured and cleared at the on-chip wakeup source. Also, the configuration registers for the external interrupts/wakeups do not have corresponding bits.

### 49.4.2.4 Interrupt Request Enable Register (IRER)

This register is used to enable the interrupt messaging from the wakeup/interrupt pads to the interrupt controller.

**NOTE**

Not all bits are available in all packages.

Address: 0x0018 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | EIRE25 | EIRE24 | EIRE23 | EIRE22 | EIRE21 | EIRE20 | EIRE19 | EIRE18 | EIRE17 | EIRE16 | EIRE15 | EIRE14 | EIRE13 | EIRE12 | EIRE11 | EIRE10 | EIRE9 | EIRE8 | EIRE7 | EIRE6 | EIRE5 | EIRE4 | EIRE3 | EIRE2 | EIRE1 | EIRE0 |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 49-6. Interrupt Request Enable Register (IRER)**

**Table 49-6. IRER Field Descriptions**

| Field D | escription |
|---|---|
| EIRE[x] | External Interrupt Request Enable x.<br>1 A set EIF[x] bit causes an interrupt request<br>0 Interrupt requests from the corresponding EIF[x] bit are disabled |

### 49.4.2.5 Wakeup Request Enable Register (WRER)

This register is used to enable the system wakeup messaging from the wakeup/interrupt pads to the mode entry and power control modules.

**NOTE**

Not all bits are available in all packages.

Address: 0x001C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | WRE25 | WRE24 | WRE23 | WRE22 | WRE21 | WRE20 | WRE19 | WRE18 | WRE17 | WRE16 | WRE15 | WRE14 | WRE13 | WRE12 | WRE11 | WRE10 | WRE9 | WRE8 | WRE7 | WRE6 | WRE5 | WRE4 | WRE3 | WRE2 | WRE1 | WRE0 |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 49-7. Wakeup Request Enable Register (WRER)**

**Table 49-7. WRER Field Descriptions**

| Field D | escription |
|---|---|
| WRE[x] | External Wakeup Request Enable x.<br>1 A set EIF[x] bit causes a system wakeup request<br>0 System wakeup requests from the corresponding EIF[x] bit are disabled |

### 49.4.2.6 Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)

This register is used to enable rising-edge triggered events on the corresponding wakeup/interrupt pads.

**NOTE**

Not all bits are available in all packages.

The RTC_API can only be configured on the rising edge.

Address: 0x0028 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | IREE25 | IREE24 | IREE23 | IREE22 | IREE21 | IREE20 | IREE19 | IREE18 | IREE17 | IREE16 | IREE15 | IREE14 | IREE13 | IREE12 | IREE11 | IREE10 | IREE9 | IREE8 | IREE7 | IREE6 | IREE5 | IREE4 | IREE3 | IREE2 | IREE1 | IREE0 |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 49-8. Wakeup/Interrupt Rising-Edge Event Enable Register (WIREER)**

**Table 49-8. WIREER Field Descriptions**

| Field D | escription |
|---|---|
| IREE[x] | External Interrupt Rising-edge Events Enable x.<br>1 Rising-edge event is enabled<br>0 Rising-edge event is disabled |

### 49.4.2.7 Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)

This register is used to enable falling-edge triggered events on the corresponding wakeup/interrupt pads.

**NOTE**

Not all bits are available in all packages.

Address: 0x002C Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | IFEE25 | IFEE24 | IFEE23 | IFEE22 | IFEE21 | IFEE20 | IFEE19 | IFEE18 | IFEE17 | IFEE16 | IFEE15 | IFEE14 | IFEE13 | IFEE12 | IFEE11 | IFEE10 | IFEE9 | IFEE8 | IFEE7 | IFEE6 | IFEE5 | IFEE4 | IFEE3 | IFEE2 | IFEE1 | IFEE0 |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 49-9. Wakeup/Interrupt Falling-Edge Event Enable Register (WIFEER)**

**Table 49-9. WIFEER Field Descriptions**

| Field D | escription |
|---|---|
| IFEEx | External Interrupt Falling-edge Events Enable x.<br>1 Falling-edge event is enabled<br>0 Falling-edge event is disabled |

### 49.4.2.8 Wakeup/Interrupt Filter Enable Register (WIFER)

This register is used to enable an analog filter on the corresponding interrupt pads to filter out glitches on the inputs.

**NOTE**

Not all bits are available in all packages.

There is no analog filter for the RTC_API.

Address: 0x0030 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | IFE25 | IFE24 | IFE23 | IFE22 | IFE21 | IFE20 | IFE19 | IFE18 | IFE17 | IFE16 | IFE15 | IFE14 | IFE13 | IFE12 | IFE11 | IFE10 | IFE9 | IFE8 | IFE7 | IFE6 | IFE5 | IFE4 | IFE3 | IFE2 | IFE1 | IFE0 |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 49-10. Wakeup/Interrupt Filter Enable Register (WIFER)**

**Table 49-10. WIFER field descriptions**

| Field D | escription |
|---|---|
| IFE[x] | External Interrupt Filter Enable x.<br>Enable analog glitch filter on the external interrupt pad input.<br>1 Filter is enabled<br>0 Filter is disabled |

### 49.4.2.9 Wakeup/Interrupt Pullup Enable Register (WIPUER)

This register is used to enable a pullup on the corresponding interrupt pads to pull an unconnected wakeup/interrupt input to a value of ‘1’.

**NOTE**

Not all bits are available in all packages.

Address: 0x0034 Access: User read/write

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | IPUE25 | IPUE24 | IPUE23 | IPUE22 | IPUE21 | IPUE20 | IPUE19 | IPUE18 | IPUE17 | IPUE16 | IPUE15 | IPUE14 | IPUE13 | IPUE12 | IPUE11 | IPUE10 | IPUE9 | IPUE8 | IPUE7 | IPUE6 | IPUE5 | IPUE4 | IPUE3 | IPUE2 | IPUE1 | IPUE0 |
| W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 49-11. Wakeup/Interrupt Pullup Enable Register (WIPUER)**

**Table 49-11. WIPUER field descriptions**

| Field D | escription |
|---|---|
| IPUE[x] | External Interrupt Pullup Enable x.<br>1 Pullup is enabled<br>0 Pullup is disabled |

## 49.5 Functional description

### 49.5.1 WKPU behavior

The WKPU provides external internal interrupts and low-power wakeups for the chip. It is possible to use the connected pins to raise an interrupt, a wakeup, or both. Table 49-12 describes the behavior of the chip for various operating modes and wakeup configurations.

**Table 49-12. WKPU behavior**

| Operating mode | Wakeup | Interrupt | Standby restart[1] | Program flow after low power mode exit |
|---|---|---|---|---|
| HALT | N/A | Enabled | N/A | Jump to ISR |

Table 49-12. WKPU behavior (continued)

| Operating mode | Wakeup | Interrupt | Standby restart[1] | Program flow after low power mode exit |
|---|---|---|---|---|
| STOP - System Clock Enabled | Enabled | Disabled | N/A | Continue from previous instruction |
| | Enabled | Enabled | N/A | Continue from previous instruction then jump to ISR |
| STOP - System Clock Disabled | Enabled | Disabled | N/A | Continue from previous instruction |
| | Enabled | Enabled | N/A | Continue from previous instruction then jump to ISR |
| STANDBY | Enabled | Disabled | Flash memory | Jump to reset vector in flash memory |
| | | | RAM | Jump to start of RAM (0x4000_0000) |
| | Enabled | Enabled | Flash memory | Jump to reset vector in flash memory, then jump to ISR[2] |
| | | | RAM | Jump to start of RAM (0x4000_0000) then jump to ISR[2] |

1 Configured using STANDBY Reset Sequence Register (RGM_STDBY) in the MC_RGM module

2 ISR is pending until interrupts have been configured at INTC.

## 49.5.2 Non-maskable interrupts

The Wakeup Unit supports one non-maskable interrupt which is allocated to pin 45 of the 176-pin package, to pin 53 of the 208-pin package, and to pin AA7 of the BGA package.

The Wakeup Unit supports the generation of three types of interrupts from the NMI. The Wakeup Unit supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

Each NMI passes through a bypassable analog glitch filter.

**NOTE**

Glitch filter control and pad configuration should be done while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

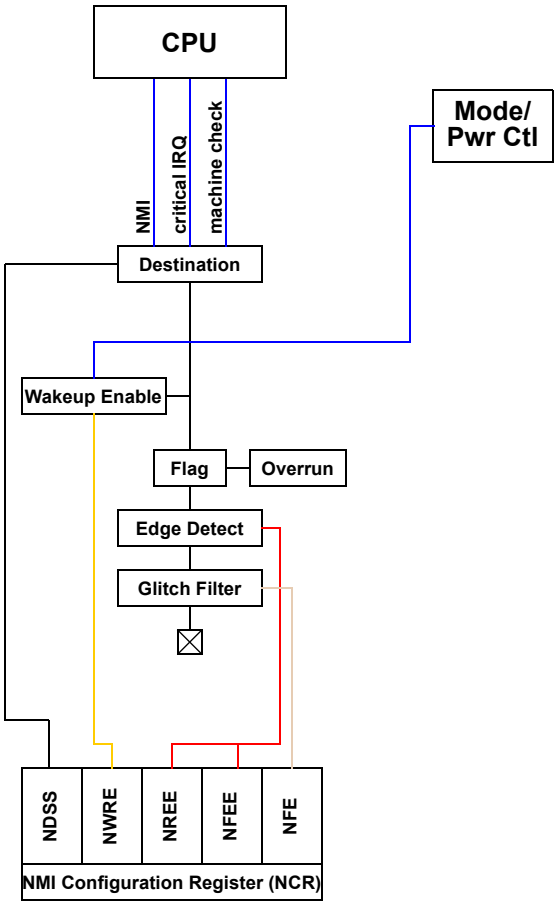



**Figure 49-12. NMI pad diagram**

### 49.5.2.1 NMI management

The NMI can be enabled or disabled using the single NCR register laid out to contain all configuration bits for an NMI in a single byte (see Figure 49-4). The pad defined as an NMI can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled results in no interrupt being detected and should not be configured.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits.

**NOTE**

After reset, NREE and NFEE are set to ‘0’, therefore the NMI functionality is disabled after reset and must be enabled explicitly by software.

Once the pad’s NMI functionality has been enabled, the pad cannot be reconfigured in the IOMUX to override or disable the NMI.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See Table 49-4 for details.

An NMI supports a status flag and an overrun flag which are located in the NSR register (see Table 49-3). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI event is detected. The overrun flag is set whenever an NMI event is detected and the status flag is set (i.e. has not yet been cleared).

**NOTE**

The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR register. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

### 49.5.3 External wakeups and interrupts

The Wakeup Unit supports four interrupt vectors to the INTC of the chip and a single wakeup source for the chip. Each interrupt vector can support up to the number of external interrupt sources from the device pads with the total across all vectors being equal to the number of external interrupt sources. Each external interrupt source is assigned to one interrupt vector. The interrupt vector assignment is sequential so that one interrupt vector is for external interrupt sources 0 through N-1, the next is for N through N+M-1, and so forth.

See Figure 49-13 for an overview of the external interrupt and wakeup implementation including the control and status registers.

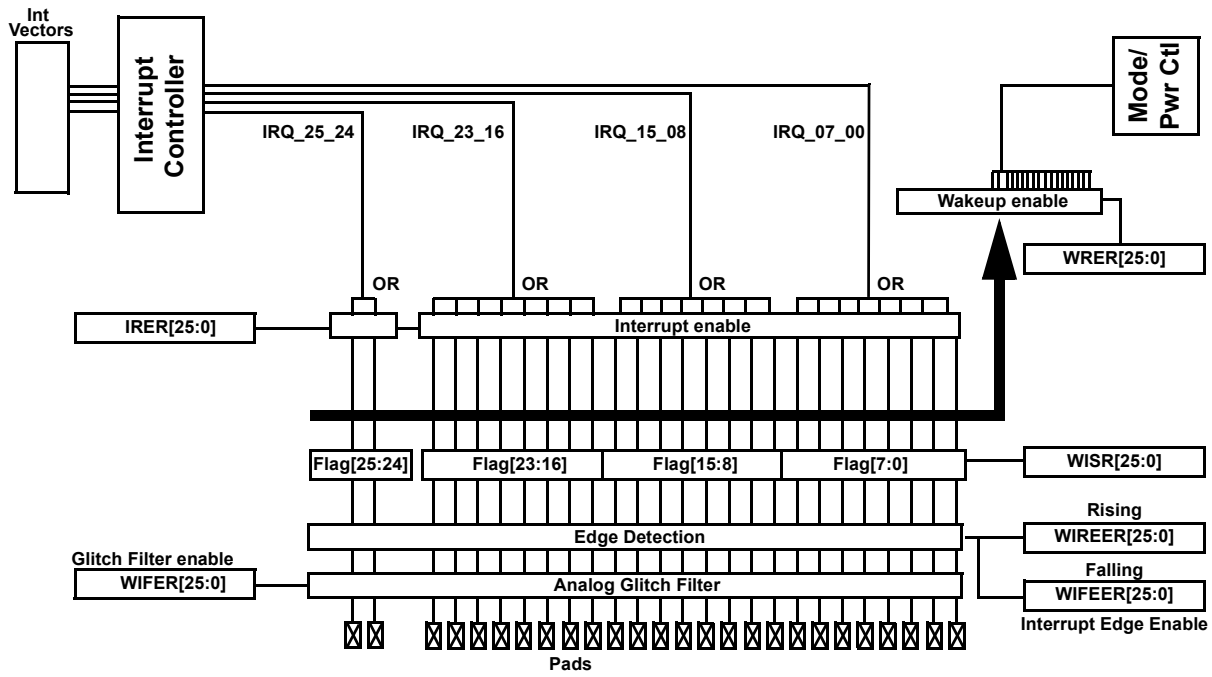



**Figure 49-13. External interrupt pad diagram**

All of the external interrupt pads within a single group have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

**NOTE**

Glitch filter control and pad configuration should be done while the external interrupt line is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

#### 49.5.3.1 External interrupt management

Each external interrupt can be enabled or disabled independently. This can be performed using the Interrupt Request Enable Register (IRER). A pad defined as an external interrupt can be configured by the user to recognize external interrupts with an active rising edge, an active falling edge or both edges being active.

**NOTE**

Writing a ‘0’ to both IREE[x] and IFEE[x] disables the external interrupt functionality for that pad completely (i.e. no system wakeup or interrupt will be generated on any activity on that pad)!

The active IRQ edge is controlled by the users through the configuration of the registers WIREER and WIFEER.

Each external interrupt supports an individual flag which is held in the flag register (WISR). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

#### 49.5.3.2 On-chip wakeup management

Each wakeup can be enabled or disabled independently using the Wakeup Request Enable Register (WRER). The active wakeup edge is controlled by the users through the configuration of the WIREER and WIFEER registers.

Each internal and external wakeup sets an individual flag which is held in the flag register (WISR). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

# Chapter 50
# Device Performance Optimization

## 50.1 Introduction

The MPC5645S contains several features that can influence the overall level of performance provided by the device.

Some of these features may be initialized upon negation of reset either by a software program called the Boot Assist Module (BAM), by a hardware state machine or by appropriate default register settings. Although the device exits the reset state into a functional state it does not necessarily have the default optimum performance settings for any given application.

This chapter provides guidance for users to fully optimize their application to achieve the highest possible performance from the MPC5645S. It provides a description of the areas that should be focused on when optimizing an application for performance by describing the features and recommending settings to be applied. It focuses on hardware configurations although certain aspects of the application software such as compiler settings and optimizations will be discussed.

## 50.2 Features

The MPC5645S has the following hardware features that can be configured to impact the overall performance of the device:

- Branch Prediction
  - Branch Target Buffer
  - Branch Prediction Control
- Frequency-modulated PLL
- Platform Flash Controller
  - Flash access wait state and address pipelining control
  - Flash instruction prefetching
  - Flash data prefetching
- Crossbar switch
- System Cache
  - Instruction Cache
- Memory Management Unit
- DRAM Controller Priority Manager

Further application level features can impact the application performance:

- Hardware Single Precision Floating point
- Signal Processing Extension (SPE-APU)
- Variable Length Encoding (VLE)
- Compiler optimizations

Further factors that impact the overall application performance are the use of the intelligent peripherals:

- Use of DMA rather than CPU to transfer data efficiently
- Use of DMA service requests rather than CPU interrupts to avoid software polling
- Off-loading tasks from the CPU to the eDMA
- Careful allocation of cache usage for code ranges.

Different items in this list will have different performance impacts in a real system. Features like the crossbar switch, system cache, the FMPLL and the flash access times tend to provide the most significant performance impacts in terms of hardware settings.

The subsequent sections in this chapter describe how to configure and use these features.

## 50.3 Configuring hardware features

### 50.3.1 Branch target buffer (BTB)

#### 50.3.1.1 Description

To resolve branch instructions and improve the accuracy of branch predictions the e200z4d core implements a dynamic branch prediction mechanism using a branch target buffer (BTB), a fully associative address cache of branch target addresses. Its purpose is to accelerate the execution of software loops with some potential change of flow within the loop body. In addition, the BTB on the e200z4d has a subroutine call stack that speeds up indirect branches.

#### 50.3.1.2 Recommended configuration

By default, the BTB is disabled following reset onthe e200z4d. It is controlled by the Branch Unit Control and Status Register (BUCSR). The BTB's contents should be flushed and invalidated by writing BUCSR[BBFI] = 1, and it may be enabled by subsequently writing BUCSR[BPEN] = 1.

Additional control is available by configuring BUCSR[BPRED] and BUCSR[BALLOC] to control whether forward or backward branches (or both) are candidates for entry into the BTB, and thus for branch prediction. By default the BUCSR[BPRED] and BUCSR[BALLOC] fields are set to 0b00, which enables forward and backward branch prediction. It is recommended to not disable branch prediction although for extremely fine tuning of a given application the optimum setting of BUCSR[BPRED] and BUCSR[BALLOC] should be assessed.

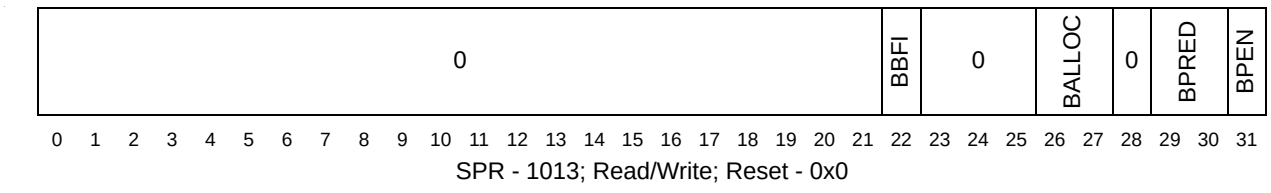



**Figure 50-1. Branch Unit Control and Status Register (BUCSR)**

Table 50-1. BUCSR field descriptions

| Field | Description |
|---|---|
| BBFI | Branch target buffer flash invalidate<br>When set, BBFI flash clears the valid bit of all BTB entries; clearing occurs regardless of the value of the enable bit (BPEN).<br>**Note**: BBFI is always read as 0. |
| BALLOC | Branch Target Buffer Allocation Control<br>00: Branch Target Buffer allocation for all branches is enabled.<br>01: Branch Target Buffer allocation is disabled for backward branches.<br>10: Branch Target Buffer allocation is disabled for forward branches.<br>11: Branch Target Buffer allocation is disabled for both branch directions.<br>This field controls BTB allocation for branch acceleration when BPEN = 1. Note that BTB hits are not affected by the settings of this field. Note that for branches with AA = '1', the MSB of the displacement field is still used to indicate forward/backward, even though the branch is absolute. |
| BPRED | Branch Prediction Control (Static)<br>00: Branch predicted taken on BTB miss for all branches.<br>01: Branch predicted taken on BTB miss only for forward branches.<br>10: Branch predicted taken on BTB miss only for backward branches.<br>11: Branch predicted not taken on BTB miss for both branch directions.<br>This field controls operation of static prediction mechanism on a BTB miss. Unless disabled, fetching of the predicted target location will be performed for branch acceleration. BPRED operates independently of BPEN, and with a BPEN setting of 0, will be used to perform static prediction of all unresolved branches.<br>Note that BTB hits are not affected by the settings of this field. Note that for certain applications, setting BPRED to a non-default value may result in improved performance. |
| BPEN | Branch target buffer (BTB) enable<br>0: BTB prediction disabled. No hits are generated from the BTB and no new entries are allocated. Entries are not automatically invalidated when BPEN is cleared; BBFI controls entry invalidation.<br>1: BTB prediction enabled (enables BTB to predict branches). |

## 50.3.2 Frequency-modulated PLL0

### 50.3.2.1 Description

The frequency-modulated phase-locked loop (FMPLL0) allows the user to generate high speed system clocks from a crystal oscillator or external clock generator. Further, the FMPLL0 supports programmable frequency modulation of the system clock. This module is typically configured early in the initialization code to ensure satisfactory performance levels are achieved.

### 50.3.2.2 Recommended configuration

After reset the MPC5645S device uses the Fast Internal RC Oscillator (FIRC) as its system clock (approximately 16 MHz). Typically, the source of the system clock is changed to the FMPLL0 to provide acceptable performance. 8.5, "Frequency-modulated phase-locked loop (FMPLL), provides details on how the FMPLL0 should be initialized in an application. The maximum frequency of operation for this device is specified in the device data sheet.

System performance cannot be linearly extrapolated with system frequency, as is often the expectation. It is due to the insertion of additional flash wait states as system frequency increases that system performance

does not scale linearly. Take care to ensure that the correct internal and/or external flash configuration is chosen for the selected system frequency. The specific flash access times to be applied are detailed in Section 21.4.5, “PFLASH2P.

## 50.3.3 Flash bus interface unit

### 50.3.3.1 Description

The Platform Flash Controller (PFC) interfaces the system bus to the flash memory array controller. The PFC contains prefetch buffers and a prefetch controller which, if enabled, speculatively prefetches sequential lines of data from the flash array into the buffer. Prefetch buffer hits allow zero-wait state responses.

The Platform Flash Configuration Registers (PFCRx) control access to the internal flash array. Its settings define the number of cycles required to access the array, access times, and how the prefetch buffering scheme operates.

Following negation of reset the instruction and data prefetching is disabled, and the number of cycles required to access the internal flash array is set to its maximum value of fifteen additional wait states.

### 50.3.3.2 Recommended configuration

As the operating frequency of the device is set by configuring the FMPLL0 (see Section 8.5, “Frequency-modulated phase-locked loop (FMPLL)), the number of cycles required to access the internal array should be configured accordingly. Note that the flash PFCRx registers cannot be altered by code executing from the flash array. Code for configuring the flash should be executed from a separate memory array i.e copied to and executed from system RAM.

Section 21.4.5, “PFLASH2P, documents the register fields used to configure flash wait state settings. The “*Platform flash controller electrical characteristics*” section of the device data sheet contains the specific values for the flash wait state settings for a given operating frequency. This also provides recommendations for the prefetch buffer settings. Note that the PFCRx settings may vary between revisions of the MPC5645S.

## 50.3.4 Crossbar switch

### 50.3.4.1 Description

The multi-port crossbar switch (XBAR) supports simultaneous connections between master ports and slave ports. The XBAR allows for concurrent transactions to occur from any master port to any slave port. If a slave port is simultaneously requested by more than one master port, arbitration logic selects the higher priority master and grants it ownership of the slave port. All other masters requesting that slave port are stalled until the higher priority master completes its transactions. By default, requesting masters are granted access based on a fixed priority. A round-robin priority mode also is available.

The main goal of the XBAR is to increase overall system performance by allowing multiple masters to communicate concurrently with multiple slaves. In order to maximize data throughput it is essential to

keep arbitration delays to a minimum. The configuration of the crossbar can have implications for the performance of a system and particular care should be taken when assigning master priorities in a fixed priority application. Further, by correctly parking saves on relevant masters the initial access times to the slaves can be minimized by negating any initial arbitration penalties.

#### 50.3.4.2 Recommended configuration

The specific settings for a given situation are application dependent and thus should be assessed by the user. The primary flash port is fixed to the e200z4d instruction bus master to maximise execution speed for that core however the assignment of the second flash port will depend on which other masters are accessing the flash the most. Best performance may be obtained by prioritising the eDMA or the data bus of the e200z4d. Similarly assignment of the system and graphics RAM ports depend on how the cores, DCU and DCULite, and eDMA use the RAMs.

More details of the XBAR register configuration can be found in Section 9.3, “XBAR registers.

### 50.3.5 Cache

#### 50.3.5.1 Description

The MPC5645S e200z4d provides an 8 KB Instruction, 2-way or 4-way set-associative, Harvard cache design with a 32-byte line size. The cache is disabled by default after reset.

The cache improves system performance by providing low-latency instructions to the instruction pipelines, which decouples processor performance from system memory performance. There are several stages to enabling the cache. Not only does the cache itself have to be invalidated then enabled, but memory regions upon which it can operate must be configured in the MMU to permit cache access.

#### 50.3.5.2 Recommended configuration

The exact usage of cache is application dependent but some general guidelines for using cache to improve performance in a typical application are listed below:

- Enable instruction cache for all internal and external memories that code is being executed from.
- Consider locking critical performance routines in cache.

The process of enabling the instruction cache involves first invalidating the cache (by setting L1CSR1[ICINV]) then when invalidation is completed (L1CSR1[ICINV, ICABT] = 0) enabling the cache (by setting L1CSR1[ICE]).

The L1CSR1 special purpose register is detailed below.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | ICECE | ICEI | 0 | ICEDT | | 0 | ICUL | ICLO | ICLFC | ICLOA | ICEA | | 0 | | ICABT | ICINV | ICE |

SPR - 1011; Read/Write; Reset - 0x0

**Figure 50-2. L1 Cache Control and Status Register 1 (L1CSR1)**

Table 50-2. L1CSR1 field descriptions

| Field | Description |
|---|---|
| ICECE | Instruction Cache Error Checking Enable |
| ICEI | Instruction Cache Error Injection Enable |
| ICEDT | Instruction Cache Error Detection Type |
| ICUL | Instruction Cache Unable to Lock |
| ICLO | Instruction Cache Lock Overflow |
| ICLFC | Instruction Cache Lock Bits Flash Clear |
| ICLOA | Instruction Cache Lock Overflow Allocate |
| ICEA | Instruction Cache Error Action |
| ICABT | Instruction Cache Operation Aborted<br>Indicates a Cache Invalidate or a Cache Lock Bits Flash Clear operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location. |
| ICINV | Instruction Cache Invalidate<br>0: No cache invalidate<br>1: Cache invalidation operation<br>When written to a ‘1’, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to ‘0’. Writing a ‘1’ while an invalidation operation is in progress will result in an undefined operation. Writing a ‘0’ to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 36 cycles to complete. Invalidation occurs regardless of the enable (ICE) value.<br>During cache invalidations, the parity check bits are written with a value dependent on the ICEDT selection. ICEDT should be written with the desired value for subsequent cache operation when ICINV is set to ‘1’ for proper operation of the cache. |
| ICE | Instruction Cache Enable<br>0: Cache is disabled<br>1: Cache is enabled<br>When disabled, cache lookups are not performed for instruction accesses.<br>Other L1CSR0 cache control operations are still available. |

Note that configuration of the cache has to be performed in conjunction with configuration of the Memory Management unit. Refer to section 50.3.6, “Memory management unit (MMU).

## 50.3.6 Memory management unit (MMU)

### 50.3.6.1 Description

The Memory Management Unit is a 32-bit Power Architecture compliant implementation which provides functionality that includes address translation and application of access attributes to memory ranges defined in Translation Lookaside Buffer entries. Although the MMU does not directly impact performance, it is within the MMU that memory regions are configured to permit the use of system cache to improve performance and Variable Length Encoding (VLE) to enhance code density. Thus it is essential that the MMU is correctly configured to ensure optimal application performance is achieved.

#### 50.3.6.1.1 Recommended configuration

The core uses MMU Assist Registers (MASx) which are special purpose registers to facilitate reading, writing and searching the Translation Lookaside Buffer (TLB) entries. These MAS registers are software managed by **tlbre**, **tlbwe**, **tlbsx**, **tlbsync**, and **tlbivax** instructions.

There are several MMU Assist Register registers (MAS0–3) that require configuring. Specifically, the MAS2 register contains the fields to control whether a specified memory region described by the valid TLB Entry is cache inhibited or whether VLE encoding is valid.

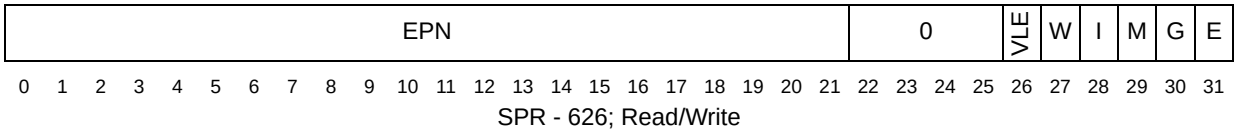



**Figure 50-3. MMU Assist Register 2 (MAS2)**

**Table 50-3. MAS2 field descriptions**

| Field | Description |
|---|---|
| EPN | Effective page number [0:21] |
| VLE | VLE<br>0: This page is a standard BookE page<br>1: This page is a VLE page |
| W | Write-through required |
| I | Cache inhibited<br>0: This page is considered cacheable<br>1: This page is considered cache-inhibited |
| M | Memory coherence required |
| G | Memory coherence required |
| E | Endianness |

## 50.3.7 DRAMC Priority Manager

### 50.3.7.1 Description

The DRAMC Priority Manager provides a way of automatically adjusting the priorities of masters requesting access to the DRAM such that access is shared according to the needs of the application. The priority of each requesting master is dynamically adjusted so that its priority increases if it is not being serviced and decreases once it has been serviced. The adjustment of the priority is controlled by look up tables (LUTs) for each of the masters. In this way it is possible for a master to very quickly gain a higher priority or delay until absolutely necessary before a change is made.

#### 50.3.7.1.1 Recommended configuration

Assign rapidly escalating LUTs to masters with limited internal buffers, for example the GFX2D and the core. In these cases the masters will stall if they do have rapid access to the DRAM. Masters with internal

buffers will tend to call on the DRAM for a burst access more periodically and can be delayed for a time since their internal buffers will allow them to continue processing therefore their priority can be escalated more slowly.

The DCU is a special case since it has an override option which allows fastest access when data is requested.

## 50.4 Application software

### 50.4.1 Compiler optimizations

The most significant opportunity for influencing the performance of a given application is by compiler and linker optimizations. Optimizing is a trade off between code size and performance. Typically higher performance of the application comes at the expense of larger code size. Compilers use a host of features, such as loop unrolling, function inlining and application profile feedback to make the desired trade-offs between enhanced performance and minimized code size.

The data in Figure 50-4 shows the effects of compiler optimization on a simple application. In this case, the Dhrystone benchmark was run under three conditions:

- Optimized for small code size
- Optimized for high performance
- A trade-off between code size and performance

Although this is an extreme example, it highlights how significant the role of the compiler and linker is in determining the overall performance of an application.

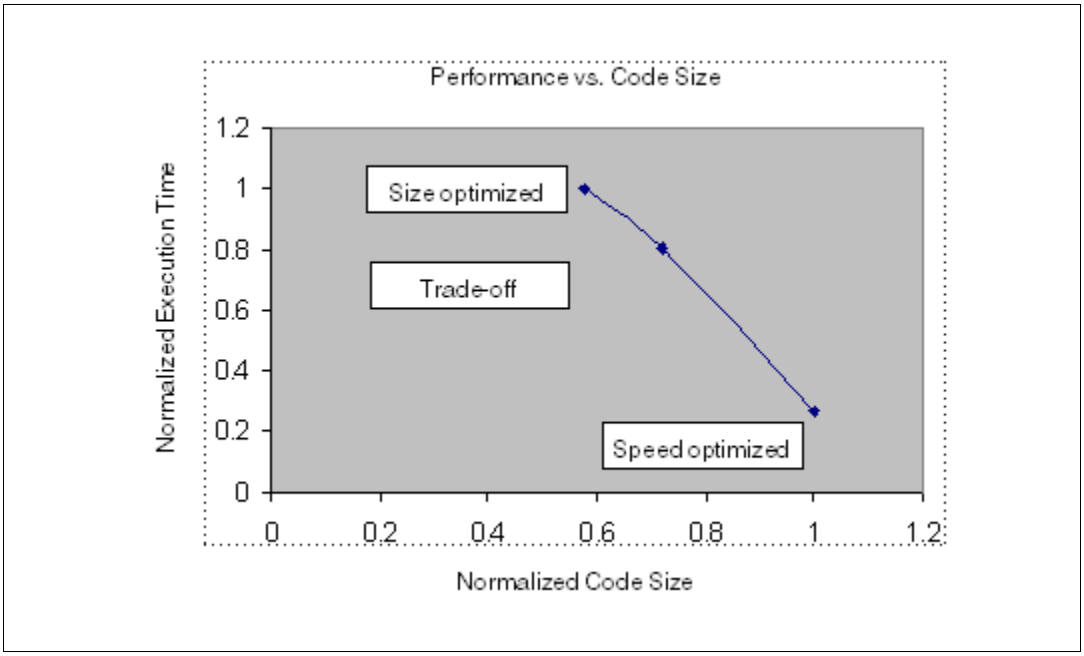



**Figure 50-4. Influence of compiler settings on application performance and code size**

**NOTE**

Data measured using Dhrystone version 2.1 run on a Power Architecture based powertrain device using a standard commercial compiler.

The compiler optimizations do not necessarily have to be applied to the entire application. Analysis of an application can identify time critical functions that may subsequently be targeted for performance optimization, without incurring the impact of optimizing the entire application.

There are several other aspects of the compiler and linker that should be considered. In particular, the use of Small Data Areas (SDAs, sometimes referred to as Special Data Areas) can make a significant performance improvement. Refer to compiler documentation for usage guidelines on Small Data Areas.

## 50.4.2 Signal processing extension

To further optimize time critical functions, the Signal Processing Extension Auxiliary Processing Unit (SPE-APU) may be used. The SPE-APU provides a set of Single Instruction Multiple Data (SIMD) instructions. These SIMD instructions typically involve performing the same operation on multiple data elements stored within a single 64-bit register. Through the implementation of SIMD instructions, including vector multiply and accumulate (MAC) instructions, the SPE APU provides Digital Signal Processing (DSP) functionality. This can be used to accelerate signal processing routines, such as Finite Impulse Response (FIR), Infinite Impulse Response (IIR) and Discrete Fourier Transforms (DFT). A more general benefit of the SPE instruction set is the ability to load/store 64-bits of data in single instruction. Thus highly load/store intensive functions make good candidates for SPE optimization.

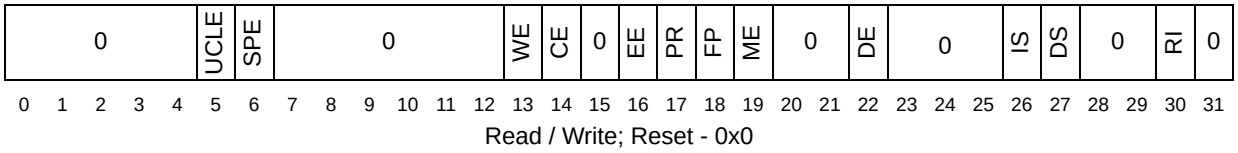



**Figure 50-5. Machine State Register (MSR)**

**Table 50-4. MSR field descriptions**

| Field | Description |
|---|---|
| UCLE | User Cache Lock Enable |
| SPE | SPE Available<br>0: Execution of SPE APU vector instructions is disabled; SPE Unavailable exception taken instead, and SPE bit is set in ESR.<br>1: Execution of SPE APU vector instructions is enabled. |
| WE | Wait State (Power management) enable |
| CE | Critical Interrupt Enable |
| EE | External Interrupt Enable |
| PR | Problem State |
| FP | Floating-Point Available |
| ME | Machine Check Enable |
| DE | Debug Interrupt Enable |

Table 50-4. MSR field descriptions (continued)

| Field | Description |
|---|---|
| IS | Instruction Address Space |
| DS | Data Address Space |
| RI | Recoverable Interrupt |

### 50.4.3 Hardware single precision floating point

The SPE-APU also supports 32-bit IEEE®-754 single-precision floating-point formats, and supports vector and scalar single-precision floating-point operations. Most compiler vendors include libraries that can emulate floating point functionality. However, by specifying the correct compiler options, the single precision floating point instructions may be used.

To enable use of hardware floating point the MSR[SPE] field must be set. Refer to Section 50.4.2, “Signal processing extension for register details.

### 50.4.4 Variable length encoding

In addition to the base Power Architecture instruction set support, the e200z4d core also implements the VLE (variable-length encoding) APU, providing improved code density. The VLE-APU can be viewed as a supplement to the existing Power Architecture instruction set that can be conditionally applied to a portion of, or an entire application for which improved code density is desired.

Using it is straightforward:

1. Select the appropriate compiler target and option to generate VLE code.
2. Configure the Memory Management Unit (MMU) to specify VLE attributes for the relevant MMU pages. Refer to the register description in Section 50.3.6, “Memory management unit (MMU).

VLE-enabled cores run both Power Architecture and VLE instruction encodings on a page by page basis, with pages defined by the MMU. The reduction is code size is typically between 25% and 30%.

## 50.5 Peripherals and general application guidelines

Optimizing the device configuration and compiler setup is only one part of optimizing an entire application. Correct use of the peripherals can also dramatically improve overall system performance. In particular, use of the interrupt controller,and the enhanced Direct Memory Access (eDMA), can off-load significant work from the e200z4d.

For example, eDMA may be used to shift data to avoid unnecessary CPU loading. Most peripheral modules can generate eDMA requests to trigger data transfers. An example of a typical application is to use the eDMA to move CAN messages from one FlexCAN module to another.

Section 50.6, “Performance optimization checklist provides several system level examples of how to optimize an application.

## 50.6 Performance optimization checklist

**Table 50-5. Performance optimization checklist—Part 1. Hardware configuration**

| Description | Register(s) | Details |
|---|---|---|
| Branch Target Buffer | Flush with BUCSR[BBFI]<br>Enable with BUCSR[BPEN] | Flush and enable to improve accuracy of branch predictions. |
| Branch Prediction | BUCSR[BPRED]/HID0[BPRED]<br>BUCSR[BALLOC] | Consider fine tuning of BTB operation for specific applications. |
| System Frequency | FMPLL_CR<br>FMPLL_MR | Select desired frequency and frequency modulation taking into account the performance impact of additional wait states. |
| Flash Wait States | FPCR0[APC, WWSC, RWSC] | Refer to Flash chapter Section 21.3.2.8, “Platform Flash Configuration Registers (PFCRP0 and PFCRP1), for FPCR settings for FMPLL frequency ranges. |
| Flash Prefetching | FPCR[DPFEN, IPFEN, PFLIM, BFEN] | Enable prefetching for instructions. Prefetching for data should be assessed for the specific application. |
| Flash Prefetch Algorithm | FPCR[BCFG] | Allocate buffers to data and/or instructions. Fine tune for specific applications. |
| Crossbar Switch | SGPCRn and MPRn | Configure ports according to most likely master to access a slave. In single core designs assign the e200z4d data port to the second flash port. In multi-core designs prioritise the flash and RAM ports to the heaviest users |
| Cache | Invalidate Icache with L1CSR1[ICINV]<br>Enable Icache with L1CSR1[ICE] | Invalidate and the enable the cache for instructions. |
| Memory Management Unit | TLB_MAS2[VLE, I] | Configure relevant pages for cache and VLE by setting MMU TLB attributes. |
| DRAMC Priority Manager | MLUTn and associated configuration registers | Configure the priority manager such that core cache misses or GFX2D requests escalate quickly |

**Table 50-6. Performance optimization checklist—Part 2. Software configuration**

| Description | Registers | Details |
|---|---|---|
| Compiler optimization | — | Use the features of the compiler to select the optimum trade off between code size and performance improvements. |
| Hardware Single Precision Floating Point | Enable with MSR[SPE] | Set compiler switches to specify using hardware single precision floating point as opposed to software emulation. |
| Signal Processing Extensions | Enable with MSR[SPE] | Take advantage of the SPE-APU to encode time critical functions using SPE assembly code. |
| Variable Length Encoding | Enabled with TLB_MAS2[VLE] | Set compiler switches and configure the MMU to take advantage of the VLE-APU. |

**Table 50-7. Performance optimization checklist—Part 3. Peripherals and general application guidelines**

| Peripherals and general application guidelines |
|---|
| Use eDMA rather than the core to move data where possible. Most peripherals can generate eDMA requests to shift data.<br>• Use DMA to fill and empty communication devices buffers.<br>• Use DMA to copy graphics from flash into RAM for further processing or reuse<br>• Use DMA to initialize DCU and DCULite CLUT |
| Shift loading from the CPU to the e200z0h whenever possible.<br>• The e200z0h can handle interrupts and scheduling for comunications peripherals. |
| Avoid software polling and allow peripherals to trigger interrupts or request eDMA servicing.<br>• Use hardware instead of software vectored interrupts to reduce latency.<br>• Trigger eDMA requests rather than interrupting the CPU to move data/results. |

# Appendix A
# Registers Under Protection

For this device the Register Protection module is operable on the registers of Table A-1.

**NOTE**

Registers protected in the DCU3 and DCULite are described in their respective chapters.

**Table A-1. Registers under protection**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| **Code flash memory, 4 registers to protect** | | | | | |
| Code Flash | MCR | 32 | C3F88000 | 000 | bits[0:31] |
| Code Flash | BIU0 | 32 | C3F88000 | 01C | bits[0:31] |
| Code Flash | BIU1 | 32 | C3F88000 | 020 | bits[0:31] |
| Code Flash | BIU2 | 32 | C3F88000 | 024 | bits[0:31] |
| **PRAM2P, 1 register to protect** | | | | | |
| PRAM2P | CR | | | | |
| **VIU2, 1 register to protect** | | | | | |
| VIU2 | STATUS_CONFIG | | | | |
| **GXG, 16 registers to protect** | | | | | |
| GXG | GXGCNFG0 | | | | |
| GXG | GXGCNFG1 | | | | |
| GXG | GXGCNFG2 | | | | |
| GXG | GXGCNFG3 | | | | |
| GXG | GXGBASE0 | | | | |
| GXG | GXGBASE1 | | | | |
| GXG | GXGBASE2 | | | | |
| GXG | GXGBASE3 | | | | |
| GXG | GXGFRST0 | | | | |
| GXG | GXGFRST1 | | | | |
| GXG | GXGFRST2 | | | | |
| GXG | GXGFRST3 | | | | |
| GXG | GXGLAST0 | | | | |
| GXG | GXGLAST1 | | | | |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| GXG | GXGLAST2 | | | | |
| GXG | GXGLAST3 | | | | |
| **DRAMC** | | | | | |
| DRAMC | All registers | | | | |
| **TCON, 53 registers to protect** | | | | | |
| TCON | TCON_CTRL | | | | |
| TCON | TCON_BMC | | | | |
| TCON | TCON_COMP0 | | | | |
| TCON | TCON_COMP1 | | | | |
| TCON | TCON_COMP2 | | | | |
| TCON | TCON_COMP3 | | | | |
| TCON | TCON_COMP0_MSK0 | | | | |
| TCON | TCON_COMP0_MSK1 | | | | |
| TCON | TCON_COMP0_MSK2 | | | | |
| TCON | TCON_COMP0_MSK3 | | | | |
| TCON | TCON_PULSE0 | | | | |
| TCON | TCON_PULSE1 | | | | |
| TCON | TCON_PULSE2 | | | | |
| TCON | TCON_PULSE3 | | | | |
| TCON | TCON_PULSE4 | | | | |
| TCON | TCON_PULSE5 | | | | |
| TCON | TCON_PULSE0_MSK | | | | |
| TCON | TCON_PULSE1_MSK | | | | |
| TCON | TCON_PULSE2_MSK | | | | |
| TCON | TCON_PULSE3_MSK | | | | |
| TCON | TCON_PULSE4_MSK | | | | |
| TCON | TCON_PULSE5_MSK | | | | |
| TCON | TCON_SMX0 | | | | |
| TCON | TCON_SMX1 | | | | |
| TCON | TCON_SMX2 | | | | |
| TCON | TCON_SMX3 | | | | |
| TCON | TCON_SMX4 | | | | |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| TCON | TCON_SMX5 | | | | |
| TCON | TCON_SMX6 | | | | |
| TCON | TCON_SMX7 | | | | |
| TCON | TCON_SMX8 | | | | |
| TCON | TCON_SMX9 | | | | |
| TCON | TCON_SMX10 | | | | |
| TCON | TCON_SMX11 | | | | |
| TCON | TCON_SMX12 | | | | |
| TCON | TCON_SMX13 | | | | |
| TCON | TCON_OMUX_LOW | | | | |
| TCON | TCON_OMUX_HIGH | | | | |
| TCON | TCON_LUT0 | | | | |
| TCON | TCON_LUT1 | | | | |
| TCON | TCON_LUT2 | | | | |
| TCON | TCON_LUT3 | | | | |
| TCON | TCON_LUT4 | | | | |
| TCON | TCON_LUT5 | | | | |
| TCON | TCON_LUT6 | | | | |
| TCON | TCON_LUT7 | | | | |
| TCON | TCON_LUT8 | | | | |
| TCON | TCON_LUT9 | | | | |
| TCON | TCON_LUT10 | | | | |
| TCON | TCON_LUT11 | | | | |
| TCON | TCON_LUT12 | | | | |
| TCON | TCON_LUT13 | | | | |
| TCON | TCON_CTRL2 | | | | |
| **SIUL, 64 registers to protect** | | | | | |
| SIUL | IRER | 32 | C3F90000 | 018 | bits[0:31] |
| SIUL | IREER | 32 | C3F90000 | 028 | bits[0:31] |
| SIUL | IFEER | 32 | C3F90000 | 02C | bits[0:31] |
| SIUL | IFER | 32 | C3F90000 | 030 | bits[0:31] |
| SIUL | PCR0 | 16 | C3F90000 | 040 | bits[0:15] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| SIUL | PCR1 | 16 | C3F90000 | 042 | bits[0:15] |
| SIUL | PCR2 | 16 | C3F90000 | 044 | bits[0:15] |
| SIUL | PCR3 | 16 | C3F90000 | 046 | bits[0:15] |
| SIUL | PCR4 | 16 | C3F90000 | 048 | bits[0:15] |
| SIUL | PCR5 | 16 | C3F90000 | 04A | bits[0:15] |
| SIUL | PCR6 | 16 | C3F90000 | 04C | bits[0:15] |
| SIUL | PCR7 | 16 | C3F90000 | 04E | bits[0:15] |
| SIUL | PCR8 | 16 | C3F90000 | 050 | bits[0:15] |
| SIUL | PCR9 | 16 | C3F90000 | 052 | bits[0:15] |
| SIUL | PCR10 | 16 | C3F90000 | 054 | bits[0:15] |
| SIUL | PCR11 | 16 | C3F90000 | 056 | bits[0:15] |
| SIUL | PCR12 | 16 | C3F90000 | 058 | bits[0:15] |
| SIUL | PCR13 | 16 | C3F90000 | 05A | bits[0:15] |
| SIUL | PCR14 | 16 | C3F90000 | 05C | bits[0:15] |
| SIUL | PCR15 | 16 | C3F90000 | 05E | bits[0:15] |
| SIUL | PCR16 | 16 | C3F90000 | 060 | bits[0:15] |
| SIUL | PCR17 | 16 | C3F90000 | 062 | bits[0:15] |
| SIUL | PCR18 | 16 | C3F90000 | 064 | bits[0:15] |
| SIUL | PCR19 | 16 | C3F90000 | 066 | bits[0:15] |
| SIUL | PCR34 | 16 | C3F90000 | 084 | bits[0:15] |
| SIUL | PCR35 | 16 | C3F90000 | 086 | bits[0:15] |
| SIUL | PCR36 | 16 | C3F90000 | 088 | bits[0:15] |
| SIUL | PCR37 | 16 | C3F90000 | 08A | bits[0:15] |
| SIUL | PCR38 | 16 | C3F90000 | 08C | bits[0:15] |
| SIUL | PCR39 | 16 | C3F90000 | 08E | bits[0:15] |
| SIUL | PCR40 | 16 | C3F90000 | 090 | bits[0:15] |
| SIUL | PCR41 | 16 | C3F90000 | 092 | bits[0:15] |
| SIUL | PCR42 | 16 | C3F90000 | 094 | bits[0:15] |
| SIUL | PCR43 | 16 | C3F90000 | 096 | bits[0:15] |
| SIUL | PCR44 | 16 | C3F90000 | 098 | bits[0:15] |
| SIUL | PCR45 | 16 | C3F90000 | 09A | bits[0:15] |
| SIUL | PCR46 | 16 | C3F90000 | 09C | bits[0:15] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| SIUL | PCR47 | 16 | C3F90000 | 09E | bits[0:15] |
| SIUL | PCR48 | 16 | C3F90000 | 0A0 | bits[0:15] |
| SIUL | PCR49 | 16 | C3F90000 | 0A2 | bits[0:15] |
| SIUL | PCR50 | 16 | C3F90000 | 0A4 | bits[0:15] |
| SIUL | PCR51 | 16 | C3F90000 | 0A6 | bits[0:15] |
| SIUL | PCR52 | 16 | C3F90000 | 0A8 | bits[0:15] |
| SIUL | PCR53 | 16 | C3F90000 | 0AA | bits[0:15] |
| SIUL | PCR54 | 16 | C3F90000 | 0AC | bits[0:15] |
| SIUL | PCR55 | 16 | C3F90000 | 0AE | bits[0:15] |
| SIUL | PCR56 | 16 | C3F90000 | 0B0 | bits[0:15] |
| SIUL | PCR57 | 16 | C3F90000 | 0B2 | bits[0:15] |
| SIUL | PCR58 | 16 | C3F90000 | 0B4 | bits[0:15] |
| SIUL | PCR59 | 16 | C3F90000 | 0B6 | bits[0:15] |
| SIUL | PCR60 | 16 | C3F90000 | 0B8 | bits[0:15] |
| SIUL | PCR61 | 16 | C3F90000 | 0BA | bits[0:15] |
| SIUL | PCR62 | 16 | C3F90000 | 0BC | bits[0:15] |
| SIUL | PCR63 | 16 | C3F90000 | 0BE | bits[0:15] |
| SIUL | PCR64 | 16 | C3F90000 | 0C0 | bits[0:15] |
| SIUL | PCR65 | 16 | C3F90000 | 0C2 | bits[0:15] |
| SIUL | PCR66 | 16 | C3F90000 | 0C4 | bits[0:15] |
| SIUL | PCR67 | 16 | C3F90000 | 0C6 | bits[0:15] |
| SIUL | PCR68 | 16 | C3F90000 | 0C8 | bits[0:15] |
| SIUL | PCR69 | 16 | C3F90000 | 0CA | bits[0:15] |
| SIUL | PCR70 | 16 | C3F90000 | 0CC | bits[0:15] |
| SIUL | PCR71 | 16 | C3F90000 | 0CE | bits[0:15] |
| SIUL | PCR72 | 16 | C3F90000 | 0D0 | bits[0:15] |
| SIUL | PCR73 | 16 | C3F90000 | 0D2 | bits[0:15] |
| SIUL | PCR74 | 16 | C3F90000 | 0D4 | bits[0:15] |
| SIUL | PCR75 | 16 | C3F90000 | 0D6 | bits[0:15] |
| SIUL | PCR76 | 16 | C3F90000 | 0D8 | bits[0:15] |
| SIUL | PCR77 | 16 | C3F90000 | 0DA | bits[0:15] |
| SIUL | PCR78 | 16 | C3F90000 | 0DC | bits[0:15] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| SIUL | PCR79 | 16 | C3F90000 | 0DE | bits[0:15] |
| SIUL | PCR80 | 16 | C3F90000 | 0E0 | bits[0:15] |
| SIUL | PCR81 | 16 | C3F90000 | 0E2 | bits[0:15] |
| SIUL | PCR82 | 16 | C3F90000 | 0E4 | bits[0:15] |
| SIUL | PCR83 | 16 | C3F90000 | 0E6 | bits[0:15] |
| SIUL | PCR84 | 16 | C3F90000 | 0E8 | bits[0:15] |
| SIUL | PCR85 | 16 | C3F90000 | 0EA | bits[0:15] |
| SIUL | PCR86 | 16 | C3F90000 | 0EC | bits[0:15] |
| SIUL | PCR87 | 16 | C3F90000 | 0EE | bits[0:15] |
| SIUL | PCR88 | 16 | C3F90000 | 0F0 | bits[0:15] |
| SIUL | PCR89 | 16 | C3F90000 | 0F2 | bits[0:15] |
| SIUL | PCR90 | 16 | C3F90000 | 0F4 | bits[0:15] |
| SIUL | PCR91 | 16 | C3F90000 | 0F6 | bits[0:15] |
| SIUL | PCR92 | 16 | C3F90000 | 0F8 | bits[0:15] |
| SIUL | PCR93 | 16 | C3F90000 | 0FA | bits[0:15] |
| SIUL | PCR94 | 16 | C3F90000 | 0FC | bits[0:15] |
| SIUL | PCR95 | 16 | C3F90000 | 0FE | bits[0:15] |
| SIUL | PCR96 | 16 | C3F90000 | 100 | bits[0:15] |
| SIUL | PCR97 | 16 | C3F90000 | 102 | bits[0:15] |
| SIUL | PCR98 | 16 | C3F90000 | 104 | bits[0:15] |
| SIUL | PCR99 | 16 | C3F90000 | 106 | bits[0:15] |
| SIUL | PCR100 | 16 | C3F90000 | 108 | bits[0:15] |
| SIUL | PCR101 | 16 | C3F90000 | 10A | bits[0:15] |
| SIUL | PCR102 | 16 | C3F90000 | 10C | bits[0:15] |
| SIUL | PCR103 | 16 | C3F90000 | 10E | bits[0:15] |
| SIUL | PCR104 | 16 | C3F90000 | 110 | bits[0:15] |
| SIUL | PCR105 | 16 | C3F90000 | 112 | bits[0:15] |
| SIUL | PCR106 | 16 | C3F90000 | 114 | bits[0:15] |
| SIUL | PCR107 | 16 | C3F90000 | 116 | bits[0:15] |
| SIUL | PCR108 | 16 | C3F90000 | 118 | bits[0:15] |
| SIUL | PCR109 | 16 | C3F90000 | 11A | bits[0:15] |
| SIUL | PCR110 | 16 | C3F90000 | 11C | bits[0:15] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| SIUL | PCR111 | 16 | C3F90000 | 11E | bits[0:15] |
| SIUL | PCR112 | 16 | C3F90000 | 120 | bits[0:15] |
| SIUL | PCR113 | 16 | C3F90000 | 122 | bits[0:15] |
| SIUL | PCR114 | 16 | C3F90000 | 124 | bits[0:15] |
| SIUL | PCR115 | 16 | C3F90000 | 126 | bits[0:15] |
| SIUL | PCR116 | 16 | C3F90000 | 128 | bits[0:15] |
| SIUL | PCR117 | 16 | C3F90000 | 12A | bits[0:15] |
| SIUL | PCR118 | 16 | C3F90000 | 12C | bits[0:15] |
| SIUL | PCR119 | 16 | C3F90000 | 12E | bits[0:15] |
| SIUL | PCR120 | 16 | C3F90000 | 130 | bits[0:15] |
| SIUL | PCR121 | 16 | C3F90000 | 132 | bits[0:15] |
| SIUL | PCR122 | 16 | C3F90000 | 134 | bits[0:15] |
| SIUL | PCR123 | 16 | C3F90000 | 136 | bits[0:15] |
| SIUL | PCR124 | 16 | C3F90000 | 138 | bits[0:15] |
| SIUL | PCR125 | 16 | C3F90000 | 13A | bits[0:15] |
| SIUL | PCR126 | 16 | C3F90000 | 13C | bits[0:15] |
| SIUL | PCR127 | 16 | C3F90000 | 13E | bits[0:15] |
| SIUL | PCR128 | 16 | C3F90000 | 140 | bits[0:15] |
| SIUL | PCR129 | 16 | C3F90000 | 142 | bits[0:15] |
| SIUL | PCR130 | 16 | C3F90000 | 144 | bits[0:15] |
| SIUL | PCR131 | 16 | C3F90000 | 146 | bits[0:15] |
| SIUL | PCR132 | 16 | C3F90000 | 148 | bits[0:15] |
| SIUL | PCR133 | 16 | C3F90000 | 14A | bits[0:15] |
| SIUL | PCR134 | 16 | C3F90000 | 14C | bits[0:15] |
| SIUL | PCR135 | 16 | C3F90000 | 14E | bits[0:15] |
| SIUL | PCR136 | 16 | C3F90000 | 150 | bits[0:15] |
| SIUL | PCR137 | 16 | C3F90000 | 152 | bits[0:15] |
| SIUL | PCR138 | 16 | C3F90000 | 154 | bits[0:15] |
| SIUL | PCR139 | 16 | C3F90000 | 156 | bits[0:15] |
| SIUL | PCR140 | 16 | C3F90000 | 158 | bits[0:15] |
| SIUL | PCR141 | 16 | C3F90000 | 15A | bits[0:15] |
| SIUL | PCR142 | 16 | C3F90000 | 15C | bits[0:15] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| SIUL | PCR143 | 16 | C3F90000 | 15E | bits[0:15] |
| SIUL | PCR144 | 16 | C3F90000 | 160 | bits[0:15] |
| SIUL | PCR145 | 16 | C3F90000 | 162 | bits[0:15] |
| SIUL | PCR146 | 16 | C3F90000 | 164 | bits[0:15] |
| SIUL | PCR147 | 16 | C3F90000 | 166 | bits[0:15] |
| SIUL | PCR148 | 16 | C3F90000 | 168 | bits[0:15] |
| SIUL | PCR149 | 16 | C3F90000 | 16A | bits[0:15] |
| SIUL | PCR150 | 16 | C3F90000 | 16C | bits[0:15] |
| SIUL | PCR151 | 16 | C3F90000 | 16E | bits[0:15] |
| SIUL | PCR152 | 16 | C3F90000 | 170 | bits[0:15] |
| SIUL | PCR153 | 16 | C3F90000 | 172 | bits[0:15] |
| SIUL | PCR154 | 16 | C3F90000 | 174 | bits[0:15] |
| SIUL | PCR155 | 16 | C3F90000 | 176 | bits[0:15] |
| SIUL | PCR156 | 16 | C3F90000 | 178 | bits[0:15] |
| SIUL | PCR157 | 16 | C3F90000 | 17A | bits[0:15] |
| SIUL | PCR158 | 16 | C3F90000 | 17C | bits[0:15] |
| SIUL | PCR159 | 16 | C3F90000 | 17E | bits[0:15] |
| SIUL | PCR160 | 16 | C3F90000 | 180 | bits[0:15] |
| SIUL | PCR161 | 16 | C3F90000 | 182 | bits[0:15] |
| SIUL | PCR162 | 16 | C3F90000 | 184 | bits[0:15] |
| SIUL | PCR163 | 16 | C3F90000 | 186 | bits[0:15] |
| SIUL | PCR164 | 16 | C3F90000 | 188 | bits[0:15] |
| SIUL | PCR165 | 16 | C3F90000 | 18A | bits[0:15] |
| SIUL | PCR166 | 16 | C3F90000 | 18C | bits[0:15] |
| SIUL | PCR167 | 16 | C3F90000 | 18E | bits[0:15] |
| SIUL | PCR168 | 16 | C3F90000 | 190 | bits[0:15] |
| SIUL | PCR169 | 16 | C3F90000 | 192 | bits[0:15] |
| SIUL | PCR170 | 16 | C3F90000 | 194 | bits[0:15] |
| SIUL | PCR171 | 16 | C3F90000 | 196 | bits[0:15] |
| SIUL | PCR172 | 16 | C3F90000 | 198 | bits[0:15] |
| SIUL | PCR173 | 16 | C3F90000 | 19A | bits[0:15] |
| SIUL | PCR174 | 16 | C3F90000 | 19C | bits[0:15] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| SIUL | PCR175 | 16 | C3F90000 | 19E | bits[0:15] |
| SIUL | PCR176 | 16 | C3F90000 | 1A0 | bits[0:15] |
| SIUL | PCR177 | 16 | C3F90000 | 1A2 | bits[0:15] |
| SIUL | PCR178 | 16 | C3F90000 | 1A4 | bits[0:15] |
| SIUL | PCR179 | 16 | C3F90000 | 1A6 | bits[0:15] |
| SIUL | PCR180 | 16 | C3F90000 | 1A8 | bits[0:15] |
| SIUL | PCR181 | 16 | C3F90000 | 1AA | bits[0:15] |
| SIUL | PCR182 | 16 | C3F90000 | 1AC | bits[0:15] |
| SIUL | PCR183 | 16 | C3F90000 | 1AE | bits[0:15] |
| SIUL | PCR184 | 16 | C3F90000 | 1B0 | bits[0:15] |
| SIUL | PCR185 | 16 | C3F90000 | 1B2 | bits[0:15] |
| SIUL | PCR186 | 16 | C3F90000 | 1B4 | bits[0:15] |
| SIUL | PCR187 | 16 | C3F90000 | 1B6 | bits[0:15] |
| SIUL | PCR188 | 16 | C3F90000 | 1B8 | bits[0:15] |
| SIUL | PCR189 | 16 | C3F90000 | 1BA | bits[0:15] |
| SIUL | PCR190 | 16 | C3F90000 | 1BC | bits[0:15] |
| SIUL | PCR191 | 16 | C3F90000 | 1BE | bits[0:15] |
| SIUL | PCR192 | 16 | C3F90000 | 1C0 | bits[0:15] |
| SIUL | PCR193 | 16 | C3F90000 | 1C2 | bits[0:15] |
| SIUL | PCR194 | 16 | C3F90000 | 1C4 | bits[0:15] |
| SIUL | PCR195 | 16 | C3F90000 | 1C6 | bits[0:15] |
| SIUL | PCR196 | 16 | C3F90000 | 1C8 | bits[0:15] |
| SIUL | PCR197 | 16 | C3F90000 | 1CA | bits[0:15] |
| SIUL | PCR198 | 16 | C3F90000 | 1CC | bits[0:15] |
| SIUL | PCR199 | 16 | C3F90000 | 1CE | bits[0:15] |
| SIUL | PCR200 | 16 | C3F90000 | 1D0 | bits[0:15] |
| SIUL | PCR201 | 16 | C3F90000 | 1D2 | bits[0:15] |
| SIUL | PCR202 | 16 | C3F90000 | 1D4 | bits[0:15] |
| SIUL | PCR203 | 16 | C3F90000 | 1D6 | bits[0:15] |
| SIUL | PCR204 | 16 | C3F90000 | 1D8 | bits[0:15] |
| SIUL | PCR205 | 16 | C3F90000 | 1DA | bits[0:15] |
| SIUL | PCR206 | 16 | C3F90000 | 1DC | bits[0:15] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| SIUL | PCR207 | 16 | C3F90000 | 1DE | bits[0:15] |
| SIUL | PCR208 | 16 | C3F90000 | 1E0 | bits[0:15] |
| SIUL | PCR209 | 16 | C3F90000 | 1E2 | bits[0:15] |
| SIUL | PCR210 | 16 | C3F90000 | 1E4 | bits[0:15] |
| SIUL | PCR211 | 16 | C3F90000 | 1E6 | bits[0:15] |
| SIUL | PCR212 | 16 | C3F90000 | 1E8 | bits[0:15] |
| SIUL | PCR213 | 16 | C3F90000 | 1EA | bits[0:15] |
| SIUL | PCR214 | 16 | C3F90000 | 1EC | bits[0:15] |
| SIUL | PCR215 | 16 | C3F90000 | 1EE | bits[0:15] |
| SIUL | PCR216 | 16 | C3F90000 | 1F0 | bits[0:15] |
| SIUL | PCR217 | 16 | C3F90000 | 1F2 | bits[0:15] |
| SIUL | PCR218 | 16 | C3F90000 | 1F4 | bits[0:15] |
| SIUL | PCR219 | 16 | C3F90000 | 1F6 | bits[0:15] |
| SIUL | PCR220 | 16 | C3F90000 | 1F8 | bits[0:15] |
| SIUL | PCR221 | 16 | C3F90000 | 1FA | bits[0:15] |
| SIUL | PCR222 | 16 | C3F90000 | 1FC | bits[0:15] |
| SIUL | PCR223 | 16 | C3F90000 | 1FE | bits[0:15] |
| SIUL | PCR224 | 16 | C3F90000 | 200 | bits[0:15] |
| SIUL | PCR225 | 16 | C3F90000 | 202 | bits[0:15] |
| SIUL | PCR226 | 16 | C3F90000 | 204 | bits[0:15] |
| SIUL | PCR227 | 16 | C3F90000 | 206 | bits[0:15] |
| SIUL | PCR228 | 16 | C3F90000 | 208 | bits[0:15] |
| SIUL | PCR229 | 16 | C3F90000 | 20A | bits[0:15] |
| SIUL | PCR230 | 16 | C3F90000 | 20C | bits[0:15] |
| SIUL | PCR231 | 16 | C3F90000 | 20E | bits[0:15] |
| SIUL | PCR232 | 16 | C3F90000 | 210 | bits[0:15] |
| SIUL | PCR233 | 16 | C3F90000 | 212 | bits[0:15] |
| SIUL | PCR234 | 16 | C3F90000 | 214 | bits[0:15] |
| SIUL | PCR235 | 16 | C3F90000 | 216 | bits[0:15] |
| SIUL | PCR236 | 16 | C3F90000 | 218 | bits[0:15] |
| SIUL | PCR237 | 16 | C3F90000 | 21A | bits[0:15] |
| SIUL | PCR238 | 16 | C3F90000 | 21C | bits[0:15] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| SIUL | PCR239 | 16 | C3F90000 | 21E | bits[0:15] |
| SIUL | PCR240 | 16 | C3F90000 | 220 | bits[0:15] |
| SIUL | PCR241 | 16 | C3F90000 | 222 | bits[0:15] |
| SIUL | PCR242 | 16 | C3F90000 | 224 | bits[0:15] |
| SIUL | PCR243 | 16 | C3F90000 | 226 | bits[0:15] |
| SIUL | PCR244 | 16 | C3F90000 | 228 | bits[0:15] |
| SIUL | PCR245 | 16 | C3F90000 | 22A | bits[0:15] |
| SIUL | PCR246 | 16 | C3F90000 | 22C | bits[0:15] |
| SIUL | PCR247 | 16 | C3F90000 | 22E | bits[0:15] |
| SIUL | PCR248 | 16 | C3F90000 | 230 | bits[0:15] |
| SIUL | PCR249 | 16 | C3F90000 | 232 | bits[0:15] |
| SIUL | PCR250 | 16 | C3F90000 | 234 | bits[0:15] |
| SIUL | PCR251 | 16 | C3F90000 | 236 | bits[0:15] |
| SIUL | PCR252 | 16 | C3F90000 | 238 | bits[0:15] |
| SIUL | PCR253 | 16 | C3F90000 | 23A | bits[0:15] |
| SIUL | PCR254 | 16 | C3F90000 | 23C | bits[0:15] |
| SIUL | PCR255 | 16 | C3F90000 | 23E | bits[0:15] |
| SIUL | PCR256 | 16 | C3F90000 | 240 | bits[0:15] |
| SIUL | PCR257 | 16 | C3F90000 | 242 | bits[0:15] |
| SIUL | PCR258 | 16 | C3F90000 | 244 | bits[0:15] |
| SIUL | PCR259 | 16 | C3F90000 | 246 | bits[0:15] |
| SIUL | PCR260 | 16 | C3F90000 | 248 | bits[0:15] |
| SIUL | PCR261 | 16 | C3F90000 | 24A | bits[0:15] |
| SIUL | PCR262 | 16 | C3F90000 | 24C | bits[0:15] |
| SIUL | PCR263 | 16 | C3F90000 | 24E | bits[0:15] |
| SIUL | PCR264 | 16 | C3F90000 | 250 | bits[0:15] |
| SIUL | PCR265 | 16 | C3F90000 | 252 | bits[0:15] |
| SIUL | PCR266 | 16 | C3F90000 | 254 | bits[0:15] |
| SIUL | PCR267 | 16 | C3F90000 | 256 | bits[0:15] |
| SIUL | PCR268 | 16 | C3F90000 | 258 | bits[0:15] |
| SIUL | PCR269 | 16 | C3F90000 | 25A | bits[0:15] |
| SIUL | PCR270 | 16 | C3F90000 | 25C | bits[0:15] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| SIUL | PCR271 | 16 | C3F90000 | 25E | bits[0:15] |
| SIUL | PCR272 | 16 | C3F90000 | 260 | bits[0:15] |
| SIUL | PCR273 | 16 | C3F90000 | 262 | bits[0:15] |
| SIUL | PCR274 | 16 | C3F90000 | 264 | bits[0:15] |
| SIUL | PCR275 | 16 | C3F90000 | 266 | bits[0:15] |
| SIUL | PCR276 | 16 | C3F90000 | 268 | bits[0:15] |
| SIUL | PCR277 | 16 | C3F90000 | 26A | bits[0:15] |
| SIUL | PCR278 | 16 | C3F90000 | 26C | bits[0:15] |
| SIUL | PCR279 | 16 | C3F90000 | 26E | bits[0:15] |
| SIUL | PCR280 | 16 | C3F90000 | 270 | bits[0:15] |
| SIUL | PCR281 | 16 | C3F90000 | 272 | bits[0:15] |
| SIUL | PSMI0 | 8 | C3F90000 | 500 | bits[0:31] |
| SIUL | PSMI4 | 8 | C3F90000 | 504 | bits[0:31] |
| SIUL | PSMI8 | 8 | C3F90000 | 508 | bits[0:31] |
| SIUL | PSMI12 | 8 | C3F90000 | 50C | bits[0:31] |
| SIUL | PSMI16 | 8 | C3F90000 | 510 | bits[0:31] |
| SIUL | PSMI20 | 8 | C3F90000 | 514 | bits[0:31] |
| SIUL | PSMI24 | 8 | C3F90000 | 518 | bits[0:31] |
| SIUL | PSMI28 | 8 | C3F90000 | 51C | bits[0:31] |
| SIUL | PSMI32 | 8 | C3F90000 | 520 | bits[0:31] |
| SIUL | PSMI36 | 8 | C3F90000 | 524 | bits[0:31] |
| SIUL | PSMI40 | 8 | C3F90000 | 528 | bits[0:31] |
| SIUL | PSMI44 | 8 | C3F90000 | 52C | bits[0:31] |
| SIUL | PSMI48 | 8 | C3F90000 | 530 | bits[0:31] |
| SIUL | PSMI52 | 8 | C3F90000 | 534 | bits[0:31] |
| SIUL | IFMC0 | 32 | C3F90000 | 1000 | bits[0:31] |
| SIUL | IFMC1 | 32 | C3F90000 | 1004 | bits[0:31] |
| SIUL | IFMC2 | 32 | C3F90000 | 1008 | bits[0:31] |
| SIUL | IFMC3 | 32 | C3F90000 | 100C | bits[0:31] |
| SIUL | IFMC4 | 32 | C3F90000 | 1010 | bits[0:31] |
| SIUL | IFMC5 | 32 | C3F90000 | 1014 | bits[0:31] |
| SIUL | IFMC6 | 32 | C3F90000 | 1018 | bits[0:31] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| SIUL | IFMC7 | 32 | C3F90000 | 101C | bits[0:31] |
| SIUL | IFMC8 | 32 | C3F90000 | 1020 | bits[0:31] |
| SIUL | IFMC9 | 32 | C3F90000 | 1024 | bits[0:31] |
| SIUL | IFMC10 | 32 | C3F90000 | 1028 | bits[0:31] |
| SIUL | IFMC11 | 32 | C3F90000 | 102C | bits[0:31] |
| SIUL | IFMC12 | 32 | C3F90000 | 1030 | bits[0:31] |
| SIUL | IFMC13 | 32 | C3F90000 | 1034 | bits[0:31] |
| SIUL | IFMC14 | 32 | C3F90000 | 1038 | bits[0:31] |
| SIUL | IFMC15 | 32 | C3F90000 | 103C | bits[0:31] |
| SIUL | IFMC16 | 32 | C3F90000 | 1040 | bits[0:31] |
| SIUL | IFMC17 | 32 | C3F90000 | 1044 | bits[0:31] |
| SIUL | IFMC18 | 32 | C3F90000 | 1048 | bits[0:31] |
| SIUL | IFMC19 | 32 | C3F90000 | 104C | bits[0:31] |
| SIUL | IFMC20 | 32 | C3F90000 | 1050 | bits[0:31] |
| SIUL | IFMC21 | 32 | C3F90000 | 1054 | bits[0:31] |
| SIUL | IFMC22 | 32 | C3F90000 | 1058 | bits[0:31] |
| SIUL | IFMC23 | 32 | C3F90000 | 105C | bits[0:31] |
| SIUL | IFCPR | 32 | C3F90000 | 1080 | bits[0:31] |
| **MC_ME, 41 registers to protect** | | | | | |
| MC_ME | ME_ME | 32 | C3FDC000 | 008 | bits[0:31] |
| MC_ME | ME_IM | 32 | C3FDC000 | 010 | bits[0:31] |
| MC_ME | ME_TEST_MC | 32 | C3FDC000 | 024 | bits[0:31] |
| MC_ME | ME_SAFE_MC | 32 | C3FDC000 | 028 | bits[0:31] |
| MC_ME | ME_DRUN_MC | 32 | C3FDC000 | 02C | bits[0:31] |
| MC_ME | ME_RUN0_MC | 32 | C3FDC000 | 030 | bits[0:31] |
| MC_ME | ME_RUN1_MC | 32 | C3FDC000 | 034 | bits[0:31] |
| MC_ME | ME_RUN2_MC | 32 | C3FDC000 | 038 | bits[0:31] |
| MC_ME | ME_RUN3_MC | 32 | C3FDC000 | 03C | bits[0:31] |
| MC_ME | ME_HALT_MC | 32 | C3FDC000 | 040 | bits[0:31] |
| MC_ME | ME_STOP_MC | 32 | C3FDC000 | 048 | bits[0:31] |
| MC_ME | ME_STANDBY_MC | 32 | C3FDC000 | 054 | bits[0:31] |
| MC_ME | ME_RUN_PC0 | 32 | C3FDC000 | 080 | bits[0:31] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| MC_ME | ME_RUN_PC1 | 32 | C3FDC000 | 084 | bits[0:31] |
| MC_ME | ME_RUN_PC2 | 32 | C3FDC000 | 088 | bits[0:31] |
| MC_ME | ME_RUN_PC3 | 32 | C3FDC000 | 08C | bits[0:31] |
| MC_ME | ME_RUN_PC4 | 32 | C3FDC000 | 090 | bits[0:31] |
| MC_ME | ME_RUN_PC5 | 32 | C3FDC000 | 094 | bits[0:31] |
| MC_ME | ME_RUN_PC6 | 32 | C3FDC000 | 098 | bits[0:31] |
| MC_ME | ME_RUN_PC7 | 32 | C3FDC000 | 09C | bits[0:31] |
| MC_ME | ME_LP_PC0 | 32 | C3FDC000 | 0A0 | bits[0:31] |
| MC_ME | ME_LP_PC1 | 32 | C3FDC000 | 0A4 | bits[0:31] |
| MC_ME | ME_LP_PC2 | 32 | C3FDC000 | 0A8 | bits[0:31] |
| MC_ME | ME_LP_PC3 | 32 | C3FDC000 | 0AC | bits[0:31] |
| MC_ME | ME_LP_PC4 | 32 | C3FDC000 | 0B0 | bits[0:31] |
| MC_ME | ME_LP_PC5 | 32 | C3FDC000 | 0B4 | bits[0:31] |
| MC_ME | ME_LP_PC6 | 32 | C3FDC000 | 0B8 | bits[0:31] |
| MC_ME | ME_LP_PC7 | 32 | C3FDC000 | 0BC | bits[0:31] |
| MC_ME | ME_PCTL[4..7] | 32 | C3FDC000 | 0C4 | bits[0:31] |
| MC_ME | ME_PCTL[16..19] | 32 | C3FDC000 | 0D0 | bits[0:31] |
| MC_ME | ME_PCTL[20..23] | 32 | C3FDC000 | 0D4 | bits[0:31] |
| MC_ME | ME_PCTL[32..35] | 32 | C3FDC000 | 0E0 | bits[0:31] |
| MC_ME | ME_PCTL[44..47] | 32 | C3FDC000 | 0EC | bits[0:31] |
| MC_ME | ME_PCTL[48..51] | 32 | C3FDC000 | 0F0 | bits[0:31] |
| MC_ME | ME_PCTL[56..59] | 32 | C3FDC000 | 0F8 | bits[0:31] |
| MC_ME | ME_PCTL[60..63] | 32 | C3FDC000 | 0FC | bits[0:31] |
| MC_ME | ME_PCTL[68..71] | 32 | C3FDC000 | 104 | bits[0:31] |
| MC_ME | ME_PCTL[72..75] | 32 | C3FDC000 | 108 | bits[0:31] |
| MC_ME | ME_PCTL[88.91] | 32 | C3FDC000 | 118 | bits[0:31] |
| MC_ME | ME_PCTL[92..95] | 32 | C3FDC000 | 11C | bits[0:31] |
| MC_ME | ME_PCTL[104..107] | 32 | C3FDC000 | 128 | bits[0:31] |
| **MC_CGM, 3 registers to protect** | | | | | |
| MC_CGM | CGM_OC_EN | 8 | C3FE0000 | 373 | bits[0:7] |
| MC_CGM | CGM_OCDS_SC | 8 | C3FE0000 | 374 | bits[0:7] |
| MC_CGM | CGM_SC_DC[0..3] | 32 | C3FE0000 | 37C | bits[0:31] |

**Table A-1. Registers under protection (continued)**

| Module | Register | Register size | Module Base | Register Offset | Protected bitfields |
|---|---|---|---|---|---|
| **CMU 0, 1 register to protect** | | | | | |
| CMU 0 | CMU_CSR | 32 | C3FE0100 | 000 | bits[24:31] |
| **MC_RGM, 9 registers to protect** | | | | | |
| MC_RGM | RGM_FES | 16 | C3FE4000 | 000 | bits[0:15] |
| MC_RGM | RGM_DES | 16 | C3FE4000 | 002 | bits[0:15] |
| MC_RGM | RGM_FERD | 16 | C3FE4000 | 004 | bits[0:15] |
| MC_RGM | RGM_DERD | 16 | C3FE4000 | 006 | bits[0:15] |
| MC_RGM | RGM_FEAR | 16 | C3FE4000 | 010 | bits[0:15] |
| MC_RGM | RGM_DEAR | 16 | C3FE4000 | 012 | bits[0:15] |
| MC_RGM | RGM_FESS | 16 | C3FE4000 | 018 | bits[0:15] |
| MC_RGM | RGM_STDBY | 16 | C3FE4000 | 01A | bits[0:15] |
| MC_RGM | RGM_FBRE | 16 | C3FE4000 | 01C | bits[0:15] |
| **MC_PCU, 1 register to protect** | | | | | |
| MC_PCU | PCONF2 | 32 | C3FE8000 | 008 | bits[0:31] |

This page is intentionally left blank.

# Appendix B
# Revision History

This appendix describes corrections to the *MPC5645S Microcontroller Reference Manual*. For convenience, the corrections are grouped by revision.

## B.1 Changes between revisions 7 and 8

**Table A-1. Changes between revisions 7 and 8**

| Chapter | Description |
|---|---|
| Signal Description | Removed “DCULITE_TAG” tag from Figure 3-1., “176-pin LQFP pinout”.<br>Added a note in Section 3.3.1, “Pad configuration during reset phases“: TCK pin should have an external pull-down resistor of 4.7K. |
| Display Control Unit (DCU3) | Updated Figure 11-85., “Safety Mode block diagram”. Added a note also in this diagram. |
| Quad Serial Peripheral Interface (QuadSPI) | Added “Supports both 24 and 32 bit address mode” feature in Section 35.1.1, “Features“<br>Updated Section 35.5.3.7.1, “Extended Address Mode“.<br>Updated Section 35.8, “Serial Flash Devices“.<br>Updated the entry of QSPI_FR to QSPI_SFMFR in Table 35-27., “Assignment of interrupt request lines”. |
| Sound Generator Module (SGM) | Updated the field description of PLAYBT field in Table 39-15., “PBTCCH3 register Description”. |

## B.2 Changes between revisions 6 and 7

**Table A-2. Changes between revisions 6 and 7**

| Chapter | Description |
|---|---|
| Signal Description | In Figure 3-1., “176-pin LQFP pinout“:<br>• DCULITE_TAG added for PK10.<br>• Updated MCK0 to MCKO<br>In Figure 3-2., “208-pin LQFP pinout“:<br>• “DCULITE_TAG” added and “DCU_TAG” removed for PK10.<br>• “DCU_TAG” added and "DCULITE_TAG” removed for PK11.<br>• “SDA_1” added for PL0.<br>• “SCL_1” added for PL1.<br>• “SDA_B” changed to “SDA_1” for PF4.<br>In Table 5., “Port pin summary“:<br>• Updated “Function” entry for PB[10], PL[0], and PL[1], updated “Peripheral” entry for PL[0] and PL[1].<br>• For PB[7], PM[0], PM[8], and PM[12], changed PWMOA to PWMO.<br>• For PB[8], PB[10], and PM[7], changed PWMO to PWMOA. |
| Analog-to-Digital Converter (ADC) | Added programming guidelines in Section 5.3.8.2, “Power-down Exit Delay Register (PDEDR)“.<br>Added a note on injected conversion chain in Section 5.4.1.5, “Abort conversion“. |
| Clock Description | Updated Figure 8-1., “MPC5645S system clock generation“ - Removed “÷1 to ÷32” block for Osc A and removed “FXOSC_clk_divided” signal. |

**Table A-2. Changes between revisions 6 and 7**

| Chapter | Description |
|---|---|
| Display Control Unit (DCU3) | Updated the Section 11.5.2, “Managing the DCU3 FIFOs and DMA activity“ for four-pixel blending.<br>Removed “DCU3_” from all entries in Table 11-78., “Mapping of RGB data onto PDI pins”.<br>Updated the description of IPM_ERROR bit field in Table 11-24., “INT_STATUS Field Descriptions”. |
| Display Control Unit Lite (DCULite) | Removed “DCULite_” from all entries in Table 12-75., “Mapping of RGB data onto PDI pins”.<br>Updated the description of IPM_ERROR bit field in Table 12-24., “INT_STATUS field descriptions”. |
| DRAM Controller (DRMAC) | In Table 13-2., “DRAMC_SCR field descriptions“:<br>• Added a note for WDLY bitfield |
| DRAMC Priority Manager | Updated the bit description of DCUOVR bit in Table 14-3., “CFG1 and CFG2 field descriptions”. |
| Error Correction Status Module (ECSM) | Updated the field description of SRAM_ADD_ONE_WS field of MUDCR register, in Table 19-2., “Miscellaneous User-Defined Control Register (MUDCR) field descriptions” |
| FlexCAN | Added a note in Section 20.5.10, “Interrupts“, regarding the care to be taken when using bit manipulation instructions (BSET). |
| Quad Serial Peripheral Interface (QuadSPI) | Added a new field bit SCLKCFG in Figure 35-3., “Module Configuration Register (QSPI_MCR)“.<br>Updated Figure 35-1., “QuadSPI block diagram“ and Figure 35-19., “Dual-die Support“.<br>Updated Section 35.8, “Serial Flash Devices“. |
| Reset Generation Module (MC_RGM) | Updated the description of F_CORE field of RGM_FES register in Table 37-3., “Functional Event Status Register (RGM_FES) Field Descriptions”. |
| System Integration Unit Lite (SIUL) | Updated entries for PSMI[48] - PSMI[54] in Table 43-14., “Peripheral input pin selection”.<br>In Table 43-20., “IFMC field descriptions”, updated the equation of Filter period in MAXCNTx [3:0] field description. |
| Video Input Unit (VIU2)) | Updated the descriptions of FIELD_IRQ, VSTART_IRQ, and DMA_END_IRQ in Table 47-3., “SCR Fields“.<br>Added a note below Table 47-18., “Video Timing Reference Codes“. |
| Wakeup Unit (WKPU) | Updated “Function” entries for Wakeup numbers WKUP4 and WKPU15 in Table 49-1., “Wakeup vector mapping”. |

## B.3 Changes between revisions 5 and 6

**Table A-3. Changes between revisions 5 and 6**

| Chapter | Description |
|---|---|
| Introduction | Updated the feature with “The INTC supports 171 interrupt requests(163 peripheral + 8 s/w configurable)”. |
| Signal Description | — Added “Location of TCON[0:3] pins” table.<br>— Removed all occurrences of MSEO2 from the chapter.<br>— Removed TCON[3] from PK11 in “Port pin summary” table. |

**Table A-3. Changes between revisions 5 and 6 (continued)**

| Chapter | Description |
|---|---|
| Analog-to-Digital Converter (ADC) | — Added a note in “ADC sampling and conversion timing” section.<br>— Added a note in “Decode Signals Delay Register (DSDR)” section.<br>— Added “Channel 51 Data Register (CDR51)” in “ADC Digital Registers” table.<br>— Added a note in “Abort conversion” section. |
| Boot Assist Module (BAM) | — Added information in “Flash memory password swapping” section.<br>— Removed all occurrences of NVSCC1 from “Flash memory password swapping” section. |
| Clock Description | In all of the IDF column of the 16 MHz part of the “Examples of Typical PLL Settings” table, “0” is changed to “1”.<br>Updated “MPC5645S system clock generation” figure. |
| Deserial Serial Peripheral Interface (DSPI) | Updated “DSPI Block Diagram”, removed three Chip Select signals. |
| Display Control Unit (DCU3) | Added “Can drive a maximum of 2048 x 2048 hardware capable resolution and a recommended maximum of 800 x 480 resolution” feature. |
| Display Control Unit Lite (DCULite) | Added “Can drive a maximum of 2048 x 2048 hardware capable resolution and a recommended maximum of 800 x 480 resolution” feature. |
| DRAM Controller (DRMAC) | — Changed the decription of EODT bit in “DRAMC_SCR field descriptions” table.<br>— Updated the decription of RTP parameter in “Timing parameters” table |
| Enhanced Modular IO Subsystem (eMIOS) | Updated “eMIOS1 channel configurations” table. |
| FlexCAN | Updated the description of MDIS bit in “MCR field description” table. |
| Flash Memory | Removed Nonvolatile System Censoring Information 1 register (NVSCC1) and all its occurences. |
| Inter-Integrated Circuit Bus Controller Module (I2C) | Changed register access to “Supervisor/Test” for all the $I^2C$ registers. |
| Interrupt Controller (INTC) | Updated the footnote in “INTC block diagram” figure. |
| Mode Entry Module (MC_ME) | — Added the following note in “Software Guideline: ME_DRUN_CSRCON[2] should not be configured, to keep PLL ON in DRUN on standby exit.”<br>— Added the following note in “Peripheral Clocks Disable“ section - “If any clock source needs to be cut-off, then the peripherals which are running on this source, need to be switched to some other clock source or should be switched off”.<br>— Added a note in ““After modifying any of the ME_RUN_PCx, ME_LP_PCx, ME_PCTLx registers, request a mode change and wait for the mode change to be completed before entering debug mode in order to have consistent behaviour on peripheral clock control process and clock status reporting in the ME_PSx registers.”<br>— Updated “MC_ME Transition Diagram |
| Nexus Development Interface (NDI) | Added the following note in “Port Configuration Register (PCR)” section - “For any write to be possible in npc PCR register, TCK frequency should be less than the IP clock frequency”. |

**Table A-3. Changes between revisions 5 and 6 (continued)**

| Chapter | Description |
|---|---|
| Power Control Unit (MC_PCU) | Updated “STANDBY Mode Transition” section |
| Real-Time Clock (RTC/API) | Updated APIVAL note in “RTCC field description” table. |
| Reset Generation Module (MC_RGM) | — Removed “checkstop reset” from “MC_RGM block diagram” figure.<br>— Added description of F_FMPLL1” bit in “Figure Functional Event Status Register (RGM_FES)” figure, in place of F_FLASH description.<br>— Added description of of BE_SOFT in “Functional Bidirectional Reset Enable Register (RGM_FBRE)" Field description” table. |
| System Integration Unit Lite (SIUL) | Added a note below “PCR185–281 Field Descriptions” table. |
| Voltage Regulators and Power Supplies | Added description of MLVDD in “Voltage Regulators” section. |
| Wakeup Unit (WKPU) | Updated “Wakeup Vector Mapping” table |
| Device Performance Optimization | Removed CTU information from “Performance optimization checklist—Part 3. Peripherals and general application guidelines” table. |

## B.4 Changes between revisions 4 and 5

**Table A-4. Changes between revisions 4 and 5**

| Chapter | Description |
|---|---|
| Introduction | Changed STOP value of 16 MHz IRC from “On” to “OP” in “Operating mode summary” table.<br>Removed one functionality of BAM in “BAM” section. |
| Memory Map | Changed “Gauge Driver 0” to “Stepper Motor Controller” in “MPC5645S system memory map” table. |
| Signal Description | Changed “VREG_BYPASS should be pulled down externally“ to “VREG_BYPASS should be connected to ground during normal operation” in “System pin descriptions” table.<br>Changed “SMD” peripheral to “SMC” peripheral throughout “Port pin summary” table.<br>Added “DCULite_G7” as Function and “DCULite” as Peripheral in Port K of “Port pin summary” table. |
| Display Control Unit (DCU3) | Changed “rising” to “falling” and vice-versa for “INV_PXCK” pin in “SYN_POL Field Descriptions” table. |
| eDMA Channel Mux (DMACHMUX) | Incorporated Errata ERR003556 as follows:<br>• Added a new section “Low power considerations”. |
| Flash Memory | Added description about the block that contains non-volatile configuration values, in “Module Memory Map” section.<br>Added missing rows in “Flash Memory Map” table.<br>Updated all RESET bits as “1”, in “PFlash Data Access Control Register (PFSACC)” figure.<br>Updated “PFDACC field descriptions” table.<br>Added five registers and their description in “Functional Description” section. |

**Table A-4. Changes between revisions 4 and 5 (continued)**

| Chapter | Description |
|---|---|
| Graphics Accelerator Gasket (GXG) | Added new register “0x40 \| GXGSTRIDE - GFX2D Stride Setting” in “GXG Memory Map” table.<br>Added new register :GFX2D Stride Setting (GXGSTRIDE)” in “Registers Description” section.<br>Updated “Frame buffer color depth converter” section. |
| Interrupt Controller (INTC) | Changed “SMD” to “SMC” in “Interrupt vectors” table. |
| Mode Entry Module (MC_ME) | Updated the description of “S_SYSCLK” in “Global Status Register (ME_GS) Field Descriptions” table.<br>Added “I_ICONF_CU” bit in “Interrupt Status Register (ME_IS)” figure and “Interrupt Status Regiter (ME_IS) Field Descriptions” table.<br>Added “M_ICONF_CU” bit in “Interrupt Mask Register (ME_IM)” figure and “Interrupt Mask Register (ME_IM) Field Descriptions” table.<br>Updated “FXOSCON” bit as writable in “STOP Mode Configuration Register (ME_STOP_MC)” figure.<br>Updated “MC_ME Resource Control Overview” table.<br>Updated “Protection of Mode Configuration Registers” section. |
| Peripheral Bridge (PBRIDGE) | Updated the description of PBRIDGE in “Introduction” section. |
| Real-Time Clock (RTC/API) | Updated “RTC Status Register (RTCS)” figure. |
| Reset Generation Module (MC_RGM) | Added register address in “Destructive Event Alternate Request Register (RGM_DEAR)” figure.<br>Updated missing borders in “Functional Event Short Sequence Register (RGM_FESS)”. |
| Run-Length Encoding Decoder (RLE_DEC) | Changed “ RXEIE” to “RXEIF” and “ TXEIE” to “TXEIF” in “Interrupt Request Status Register (RLE_DEC_ISR)” figure. |
| Sound Generator Module (SGM) | Added missing “0” in “Clock Configuration Register for Channel 3 (CLKCH3)” figure.<br>Removed borders from reset values in “DDS Configuration Register for Channel 3 (DDSCH3)” figure.<br>Added missing “0” in “Envelope Timing Configuration Register of sustain timing for Channel 3 (ECRSCH3)”, “Inter-Note No-Output phase timing for Channel 3 (NTCH3)”, “Target note Pulse Number for Channel 3 (TPCCH3)”, “PlayBack Timing Configuration Register for channel 3(PTCCH3)”, “Dead Time Configuration Register for channel 3(DTCCH3)”, “Repeat Number Configuration Register for channel 3(RNCCH3)”, “Mixer Configuration Register (MIXCR)”, “Clock Configuration Register for PWM (CLKPWM)”, “FIFO Write Pointer (FIFOWP)”, “SGM Interrupt Control Register for DMA and FIFO (SGMICDF)”, “SGM Interrupt Control Register (SGMIC)”, “SGM Interrupt Status Register for FIFO and DMA(SGMISFD)”, “SGM Interrupt Status Register (SGMIS)”, “ I2S Enable Register (I2SEN)”, “I2S control register (I2SCTL)”, “I2S Output Data Format Control Register (I2SDFC)”, “I2S Interrupt Control Register (I2SINTC)”. |
| Static RAM (SRAM) | Changed “Reset operation” section to “Reset effects on SRAM accesses” sections. Updated its content also. |

## B.5 Changes between revisions 3 and 4

**Table A-5. Changes between revisions 3 and 4**

| Chapter | Description |
|---|---|
| Throughout | Editorial changes and improvements. |
| Introduction | Changed “324_TEPBGA” to “416_TEPBGA in the Rainbow “MPC5645S device comparison” table, and updated the footnote.<br>Changed 324 TEPBGA package to 416, and dimensions (both) from 23 mm to 27 mm.<br>Changed “324” BGA to “416” in the “Nexus Development Interface“ section.<br>Changed “DCU-Lite” to “DCULite” throughout the section. |
| Memory Map | Updated the “External, Memory Mapped Serial Flash for QuadSPI“ section of the “MPC5645S system memory map” table.<br>In the “Functional Description” section, “Access Errors” subsection, added text to item #7 specifying that this set of actions results in a transfer error. |
| Safety | Editorial changes.<br>In the “Functional Description” section, “Access Errors” subsection, added text to item #7 specifying that this set of actions results in a transfer error.<br>In the “SWT Time-Out register description, changed the default counter value to 1920 (0x780) for 15ms at 128 KHz. Updated the register “Reset” field to match. |
| Signal Description | Editorial changes.<br>Changed “324” TEPBGA to “416” throughout section.<br>Added $V_{DD33_DR}$ to the “Voltage supply pin descriptions“ table.<br>Revised the “Pad Types” section.<br>Changed column heading “324” TEPBGA to “416”, and updated the corresponding pin numbers for all pins in the “Voltage supply pin descriptions“, “System pin descriptions“, “Nexus pins“, and “DRAM interface pin summary“ tables.<br>For port pin “DDR_DQS[2]” in the “DRAM interface pin summary“ table, changed PCR[230] to PCR[231].<br>Added PM[2] to the “RSDA special function muxing“ section.Changed “324” TEPBGA to “416” in the “Package pinouts” section and replaced 324 ballmap with new 416 ballmap.<br>Changed column heading “324” TEPBGA to “416”, and updated the corresponding pin numbers for all pins in the “Port pin summary“ table.<br>Updated functions and peripherals for pins PB[0], PB[1], PB[10], PB[11], PK[10], PK[11], PL[0], and PL[1].<br>Added text regarding slew rate adjustment to the text in the “Functional ports” section.<br>Added “Pad type descriptions” table.<br>Updated PDI functions of pins PJ1, 2, 8, 9, 10, 11, 4, 5, 6, and 7 in the 176 LQFP pinout.<br>In the 208 LQFP pinout:<br>• Corrected error: Changed PDI_HSYNC_VIU8 to PDI_VSYNC_VIU8.<br>• Updated PDI functions of pins PJ1, 2, 8, 9, 10, 11, 4, 5, 6, and 7 in the 176 LQFP pinout.<br>• Updated labeling on pins PL0 and PL1.<br>In the “Port pin summary” table, updated VIU functions of pins PJ2, 1, 7, 6, 5, 8, 4, 13, 11, 10, and 9.<br>Removed “DCULITE_TAG” from pin “PK10“ in the “176-pin LQFP pinout“ figure. |
| CAN Sampler | Updated the “Active_CK” description in the “CR field descriptions“ table. |

**Table A-5. Changes between revisions 3 and 4 (continued)**

| Chapter | Description |
|---|---|
| Clock Description | Changed “PLL0 clk / 2” at 125 MHz to “PLL0 clk” at 250 MHz, at the output of the “Primary FMPLL0” block.<br>Deleted clock signal from “Primary FMPLL0“ block to “DRAM Controller” and “Clock Selector“ blocks.<br>In the NOTE accompanying the “MPC5645S peripheral sets“ table, updated the last sentence regarding the DRAM controller.<br>Made numerous modifications to the “MPC5645S system clock generation figure.<br>In the “Auxiliary clocks” section, changed the number of auxiliary clocks from six to five; removed DRAM controller from the list. |
| Clock Generation Module | Added the “AUTO_DIV2” field to the “CGM_VIU_MUX” register.<br>Changed the “DIV3” field in the “CGM_SC_DC” register to non-writable and changed the reset value to 4’b0001.<br>Changed “STOP0” to “STOP” in the “System clock disable” section.<br>Added note that “peripheral set 4 clock” is “fixed at /2” in the “System clock dividers“ section.<br>Removed the “Aux Clock 5” entries from the “MC_CGM register description” table.<br>Removed the AC5 register entries from the “MC_CGM memory map” table and updated the reserved space accordingly.<br>In the “System Clock Divider Configuration Registers (CGM_SC_DC0...3)“ section:<br>• Changed the DIV3 reset value to 0b0000.<br>• Changed “peripheral set 4 clock” to “peripheral set 3 clock” in the DIV2 entry of the Field Descriptions table.<br>• Changed the “Divider 3 Division Value” description in the Field Descriptions table.<br>Removed the Auxiliary Clock 5 “Select Control” and “Divider Configuration“ register descriptions.<br>Removed the CGM_SC_DC3 Register from the “MCG_CGM system clock generation overview“ figure.<br>In the “MCG_CGM system clock generation overview“ figure:<br>• Removed the CGM_SC_DC3 Register.<br>• For the “peripheral set 4 clock”, changed “clock divider” to be “÷2“.<br>In the “Functional description” section:<br>• Changed “peripheral set 4 clock” to “fixed at ÷2”.in the “System clock dividers“ subsection.<br>• Removed “Auxiliary Clock 5 Select Control Register” from the “Auxiliary clock generation“ section.<br>• Removed the “MC_CGM Auxiliary Clock 5 Generation Overview“ figure.<br>• Removed the list of derived clocks generated by the MC_CGM.<br>• Removed the “Auxiliary Clock 5 Divider Configuration Register” from the “Dividers Functional Description” subsection.<br>• Added the “DRAM Controller Clock” subsection.<br>Removed “FXOSC registers” from and updated the address column in the “MC_CGM memory map“ table.<br>Updated the “Memory map and register definition“ section. |
| Clock Monitor Unit | Editorial updates and improvements.<br>Changed “CKSEL” to “CLKSEL” in the “Clock Monitor Unit diagram”. |
| Configurable Enhanced Modular I/O Subsystem | Changed name of chapter from “Configurable eMIOS” to “eMIOS”.<br>Added Counter Bus [A] to the “eMIOS200 block diagram“ figure.<br>Added a “Location” column to the “Unified Channel Memory Map” table. |

**Table A-5. Changes between revisions 3 and 4 (continued)**

| Chapter | Description |
|---|---|
| Crossbar Switch | Added “Graphics RAM” to the GFX2D block in the “Device-specific block diagram” figure.<br>Added an “XBAR Port ID“ column to the “XBAR Master ID Numbers” table.<br>Removed ASGPCR (Alternate SGPCR) text throughout the section.<br>Removed all Alternate registers from the “XBAR registers“ table.<br>Removed AMPRn and ASGPCRn from the “XBAR Register Summary“ table.<br>Removed the “Alternate Master Priority Register“ section.<br>Removed the “Alternate Slave General-Purpose Control Register“ section. |
| Deserial Serial Peripheral Interface | Added new paragraph specifying proper filling of the TXFIFO for both master and slave cases.<br>Added cross-references to the INTC and DMACHMUX chapters in the “Interrupt/DMA requests“ section. |
| Device Optimization | Initial version - added this chapter.<br>Removed text-embedded comments in the “Frequency-modulated PLL0“ and “Flash bus interface unit“ sections. |
| Display Control Unit | Updated the BPP field of the “CtrlDescL0_4 field descriptions” table.<br>Made changes to the Functional Description’s “Layer configuration and blending“ section, in the “Graphics and data format“ subsection:<br>• Added text with further examples of the RGB565, RGB4444, and RGB1555 formats.<br>• Changed instance of “DCU” to “DCU3”.<br>• Added text stating limitation of blending only one layer when interpolating chroma between two pixels during conversion to RGB format.<br>In the “DCU_MODE field descriptions” table, updated the DCU_MODE description for Test mode.<br>Removed the “Debugging of DCU3“ subsection as this description is now covered by the DCU_MODE description. |
| Display Control Unit Lite | Updated the BPP field of the “CtrlDescL0_4 field descriptions” table.<br>Made changes to the Functional Description’s “Layer configuration and blending“ section, in the “Graphics and data format“ subsection:<br>• Added text with further examples of the RGB565, RGB4444, and RGB1555 formats.<br>• Changed instance of “DCU” to “DCU3”.<br>• Added text stating limitation of blending only one layer when interpolating chroma between two pixels during conversion to RGB format.<br>In the “DCU_MODE field descriptions” table, updated the DCU_MODE description for Test mode.<br>- Removed the “Debugging of DCULite“ subsection as this description is now covered by the DCU_MODE description. |
| DRAM Controller | Editorial changes and improvements.<br>Changed register names from “DDR” to “DRAMC” throughout the chapter.<br>Numerous corresponding changes to register bit field names.<br>In the “DRAMC_SCR field descriptions” table:<br>• Updated the “FIFO_OVP / FIFO_UVP / FIFO_OVEN / FIFO_UVEN” field.<br>• Updated the “REFRESH” field. |
| Enhanced Direct Memory Access | Changed the number of programmable channels from “n” to “16” in the “Introduction” section. |
| eDMA Channel Mux | Changed “Reserved” sources 29-38 to LINFlex Tx/Rx FIFO in the “DMACHMUX request assignments“ table.<br>Changed “DRAMC priority manager DMA req” from “DMA MUX Source #49” to #39 in the “DMACHMUX request assignments“ table. |

**Table A-5. Changes between revisions 3 and 4 (continued)**

| Chapter | Description |
|---|---|
| Error Correction Status Module | Changed “PRAM” to “SRAM” in the “PRAM_ADD_ONE_WS bit“ section.<br>Folded the newly-renamed “SRAM_ADD_ONE_WS” section into the “Miscellaneous User-Defined Control“ register, and changed register fields as appropriate. |
| Flash Memory | Editorial changes.<br>Replaced the “Master ID” table within the “MnPFE” field description with a link to the “XBAR Master ID” table in the XBAR chapter.<br>Replaced the list of Master ID numbers in the “Flash memory setting recommendations” section with a link to the “XBAR Master ID” table in the XBAR chapter.<br>Added cross-reference to “XBAR Master ID Numbers“ table in the “Platform flash configuration registers” section.<br>Removed reference to Tdrcv in the “Power Down mode” section.<br>Added a NOTE to the “Factory Margin Read” section that specifies the conditions under which a reset must be performed when margin reads are to be executed. |
| Frequency-Modulated Phase-Locked Loop | Added “Examples of typical PLL settings“ table. |
| Graphics Accelerator Gasket | Editorial changes.<br>Replaced the “Master ID” table within the “MnPFE” field description with a link to the “XBAR Master ID” table in the XBAR chapter.<br>Replaced the list of Master ID numbers in the “Flash memory setting recommendations” section with a link to the “XBAR Master ID” table in the XBAR chapter. |
| JTAGC | Updated the PIN field reset value in the “Device Identification Register” figure. |
| LINFlexD Controller | New chapter added.<br>In the “LIN mode features” section, specified FIRC as the clock source in the “LIN automatic resynchronization” bullet.<br>Added the missing fields to the “LIN timeout control status register (LINTCSR)“ figure. |
| Memory Protection Unit | Updated the AHB slave ports list in the “Overview” section.<br>In the “MPU Region Descriptor n, Word 2“ section, added a cross-reference to the “Master ID Numbers” table in the XBAR chapter. |
| Mode Entry Module | Changed throughout chapter: HALT0 to HALT, STOP0 to STOP, and STANDBY0 to STANDBY.<br>Changed reset value to 0b00 for bits [12:13] in all Mode Configuration Registers and removed ambiguity by making these bits reserved.<br>Added text regarding boot from backup RAM to the “STANDBY Mode“ subsection of the “Functional Description” section. |
| Nexus Development Interface | Updated MDO to be MDO[11:0] in the “NDI“ and “NDI implementation“ block diagrams.<br>Changed “Four” to “Up to twelve” MDO pins in the “Operating mode” section.<br>Changed “324” TEPBGA to “416” in the “External signal description” section.<br>Updated MDO to be MDO[11:0] in the “Nexus Signal Reset State” table.<br>Changed the “MCKO_GT” bit in the “Port Configuration Register” to Reserved.<br>Changed “4 MDO pins” to “all MDO pins” in the OPC entry of the “DC1 Field Descriptions” table.<br>Changed “HALT0”, “STOP0”, and “STANDBY0” to “HALT”, “STOP”, and “STANDBY” throughout the chapter.<br>Added more detailed text describing TDO pin behavior with respect to entering/exiting low power mode. |

**Table A-5. Changes between revisions 3 and 4 (continued)**

| Chapter | Description |
|---|---|
| Oscillators | Editorial changes and improvements.<br>Removed “featuring also full swing Pierce (FSP) without internal bias resistor” from External square wave mode in the “Modes of operation“ section.<br>Updated the “NOTE” in the “EXTAL and XTAL - Input and output pins” section.<br>Removed the “Full-swing Pierce oscillator connections” figure.<br>Removed “FSP mode selected” from the “External clock connections” figure.<br>Removed “LCP mode” condition in the “Gain control” section.<br>Changed “Low Power RC Oscillator” section to “Slow Internal RC Oscillator”, and “LPRC“ to “SIRC“ throughout the section.<br>Added paragraph regarding the two’s complement trimming method in the “Slow Internal RC Oscillator“ section.<br>Changed the clock to “FIRC“ throughout the “FIRC digital interface” “Functional Description (16 MHz)“ section. |
| Power Control Unit | Changed “HALT0”, “STOP0”, and “STANDBY0” to “HALT”, “STOP”, and “STANDBY” throughout the chapter. |

**Table A-5. Changes between revisions 3 and 4 (continued)**

| Chapter | Description |
|---|---|
| Quad Serial Peripheral Interface | Extensive content updates throughout the chapter.<br>Editorial updates and improvements.<br>Updated the text in the “IOFA[3:0] - Data IO Flash A“ section.<br>Updated the text in the “IOFA[3:0] - Data IO Flash B“ section.<br>Updated text to indicate IOFX[3:2] is “Driven high all the time” for Single and Dual Instructions in the “Serial Flash Access Scheme“ timing diagram.<br>Updated write access conditions for the “Module Configuration Register”.<br>Added “EXT_ADD” bit to the “Module Configuration Register”.<br>In the “Features” section:<br>• Added support of 24- and 32-bit addresses.<br>• Changed support size of flash devices from 16MB to 64 MB.<br>• Updated the power-saving feature text.<br>Added Latency Configuration Regster (QSPI_LCR) information to chapter.<br>Changed RX buffer data registers depth from 14 to 31 throughout the chapter.<br>Changed Flash A and B address ranges, and sizes from 16MB to 128MB, and parallel flash mode from 32 MB to 256 MB in the “Serial flash address assignment“ table.<br>Added text regarding 24-bit and 32-bit addressing modes to the NOTE in the “Serial flash assignment“ section.<br>In the QSPI_MCR:<br>• Corrected the access field.<br>• Changed the upper 16 bits to be writable.<br>• Changed the “ECT_ADD” field to “EXT_ADD”.<br>• Removed the “unknown function” from the “EXT_ADD” field.<br>Added “in Disabled mode” to the “CAUTION” note in the “QSPI_MCR” section.<br>Added EXT_ADD to “QSPI_MCR field descriptions” table.<br>Added MCR.EXT_ADD text to the “QSPI_SFAR” section.<br>Updated the “SFADR address assignment“ table.<br>Updated cross-reference to AHB commands in the “QSPI_ACR” section.<br>Made extensive updates to the “QSPI_RBDR0–QSPI_RBDR31” section.<br>Changed “QSPI_AMBA_BASE” to “0x7000_0000”, and updated other addresses similarly, throughout the “AHB Bus register memory map descriptions” section.<br>Removed the sentence regarding 24 address bits and 16 MB per flash from the “Memory Mapped Serial Flash Data - Parallel Flash Mode“ section.<br>Changed ARDB buffer upper bit from 15 to 31.<br>Changed RX buffer depth from 15 to 32.<br>Updated the entire “Supported Instruction Codes in Winbond Devices“ section.<br>In the “Features” section, changed the supported flash size to be up to 128 MB.<br>In the “Memory map” section, added a cross-reference to the Latency Configuration Register.<br>Updated the VMID field in the “QSPI_MCR Field Descriptions” table.<br>Changed “Numonvx” to “Numonyx“ throughout the chapter.<br>Corrected the end addresses in the “Memory Mapped Individual Flash Mode - Flash B Address Scheme“ table.<br>Corrected bit numbering error in the “ICO field on Winbond Devices - Parallel Flash Mode“ table. |
| Register Protection | Changed “ME_HALT0_MC”, “ME_STOP0_MC”, and “ME_STANDBY0_MC” to “ME_HALT_MC”, “ME_STOP_MC”, and “ME_STANDBY_MC”, respectively, in the “Registers under protection“ table. |

**Table A-5. Changes between revisions 3 and 4 (continued)**

| Chapter | Description |
|---|---|
| Reset Generation Module | Added note for the special case when RGM_FERD[D_CMU0_OLR] = 1 & RGM_FEAR[AR_CMU0_OLR] = 1 in the “Functional Event Alternate Request Register” description.<br>Added text regarding short-sequence reset when the Flash is powered down to the NOTE in the “Functional Event Short Sequence Register“ description.<br>Changed the “F_FLASH” bit to “F_FMPLL1” in the “MC_RGM Memory Map“ table.<br>Changed the “AR_CHKSTOP“ bit to reserved/unimplemented in the “MC_RGM Memory Map“ table.<br>Changed the “F_FLASH” bit to “F_FMPLL1” in the “Functional Event Status Register (RGM_FES)“ figure.<br>Changed the “AR_CHKSTOP“ bit to reserved/unimplemented in the “Functional Event Alternate Request Register (RGM_FEAR)“ figure.<br>Removed the “AR_CHKSTOP“ description from the “Functional Event Alternate Request Register (RGM_FEAR) Field Descriptions“ table.<br>Added a NOTE regarding short reset sequences to the “Functional Event Short Sequence Register (RGM_FESS)“ section. |
| Run-Length Encoding Decoder | Added text describing the two independent interfaces and module operation to the “Introduction” section.<br>Changed “AMBA Bus” to “Crossbar switch” in the “RLE_DEC block diagram“ block diagram.<br>Swapped the text of the “Introduction” and “Overview“ sections.<br>Updated the text of the new “Introduction” section.<br>Changed “AHB” to “Crossbar switch (AHB)“ in the “AMBA Bus Register Memory Map“ section.<br>Updated the addresses in the “RLE_DEC AMBA Bus Memory Map“ table.<br>In the “Module Configuration Register”, changed bit 29 to be writable bit “TXFFEN“.<br>Added “TXFFEN” bit description to the “RLE_DEC_MCR Field Descriptions“ table.<br>Corrected error: changed “RX” to “Tx” in the “TXIEF” entry of the “RLE_DEC_ISR field descriptions“ table.<br>Changed title of the “AHB bus register memory map descriptions“ section to “Crossbar switch memory map descriptions“.<br>Updated the “Rx FIFO address range”, “Tx FIFO address range“, “Memory mapped Rx FIFO“, and “Memory mapped Tx FIFO“ sections.<br>Updated the steps in the “RLE decoding process” section.<br>Removed “The Rx FIFO receives data from AHB interface” from the “Normal mode” section.<br>Removed the diagram from the “Power-saving features“ section.<br>Updated the ending addresses in the “RLE_DEC AMBA Bus Memory Map“ table.<br>In the “RLE_DEC_MCR Field Descriptions“ table, swapped the definitions for 0 and 1 values of the “TXFFEN“ field. |
| Sound Generator Module | Editorial changes and improvements.<br>Removed text specifying that IPS clock must be 1/2 the system clock frequency in the “Device-specific configuration“ section. |
| Static RAM | Editorial changes and improvements.<br>Added “Option available to set the SRAM access to one wait state” bullet in the “Introduction” section.<br>In the “Access timing“ subsection, added text that the ECSM_MUDCR register optionally forces a minimum of one wait state for every access and referred the user to the Error Correction Status Module chapter for more information.<br>Added footnote regarding the SRAM_ADD_ONE_WS bit to the “Number of wait states required for SRAM operations” table. |

**Table A-5. Changes between revisions 3 and 4 (continued)**

| Chapter | Description |
|---|---|
| Stepper Motor Controller | Removed the “Motor Controller Halt Mode Enable (MCHME)” bit from the “Motor Controller Control Register 0 (MCCTL0)” and changed this bit to reserved. |
| System Integration Unit Lite | Editorial changes and improvements.<br>Changed “324” package to “416” package throughout the chapter.<br>In the “Pad Configuration Register (PCR) for the different pad types in MPC5645S“ description:<br>• Changed the SRC[1:0] bits graphics from grey to white;<br>• Updated the “Pad Type” information.<br>Added note regarding output buffer for non-GPIO functions in the PA[1:0] field of the “PCRx Field Descriptions“.<br>In the “Peripheral Input pin selection” table, changed the “Peripheral input” for PSMI[9] to VIU[5]_PDI13.<br>In the “MCU ID Register #1 (MIDR1)“ section, changed the “MAJOR_MASK[3:0]“ reset value from 0b0000 to 0b0001.<br>In the “MCU ID Register #2 (MIDR2)“ section, changed the reset value of Reserved bits 24 and 31 from 1 to 0.<br>Updated the pins listed in the footnotes of the following register sections:<br>• Interrupt Status Flag Register (ISR)<br>• Interrupt Request Enable Register (IRER)<br>• Interrupt Rising-Edge Event Enable Register (IREER)<br>• Interrupt Falling-Edge Event Enable Register (IFEER)<br>• Interrupt Filter Enable Register (IFER). |
| Voltage Regulators and Power Supplies | In the “Power domain organization“ section, referred the user to and added a cross-reference to the Memory Map chapter.<br>Updated the 3’b011 setting definition for the FORMAT_CTRL field in the “SCR Fields” table.<br>Updated the ECC_EN definition in the “SCR Fields” table.<br>Corrected bit field in “CHROMA BLUE” register: Changed “CB_BLUED” to “CB_BLUE“. |
| Wakeup Unit | Changed “324” package to “416” package throughout the chapter.<br>Removed 324-pkg-related footnote from the “Wakeup vector mapping“ table.<br>Removed the STANDBY mode exception from the configurable pullup in the “Features“ section.<br>Updated the NOTE in the “External signal description” section. |

## A.6 Changes between revisions 2 and 3

**Table A-6. Changes between revisions 2 and 3**

| Chapter | Description |
|---|---|
| Throughout | Editorial changes and improvements. |
| Introduction | In the device-comparison table, changed the GPIO count for the 176-pin package (was 127, is 128).<br>In the feature list:<br>• Changed “4 KB, 2-way instruction cache” to “4 KB, 2/4-way instruction cache”.<br>• Changed “188 peripheral interrupt sources” to “181 peripheral interrupt sources”.<br>• Changed “Primary FMPLL” to “Primary FMPLL (FMPLL0)”.<br>• Changed “Auxiliary FMPLL” to “Auxiliary FMPLL (FMPLL1)”.<br>• In the crossbar switch description, deleted the reference to AMBA.<br>• In the SGM description, changed “WAV file” to “PCM wave”.<br>• Revised the RLE decoder description.<br>• In the SIUL description, deleted “resets”.<br>• In the CMU section, changed “PLL” to “primary FMPLL (FMPLL0)”.<br>In the feature details:<br>• Revised the “Low-power operation” section.<br>• Revised the DCU3 section.<br>• In the “On-chip graphics SRAM” section, added an entry for independent data buffers.<br>• In the MPU section, changed “Protection offered for 3 concurrent read ports” to “Protection offered for 4 concurrent read ports”.<br>Added the “How to use the MPC5645S documents” section.<br>Added the “Using the MPC5645S” section.<br>Added the “Other features” section. |
| Memory Map | In the entry at start address 0x40002000, changed “Domain RD1” to “Domain PD2”. |

**Table A-6. Changes between revisions 2 and 3 (continued)**

| Chapter | Description |
|---|---|
| Signal Description | In the 176-pin pinout:<br>• Added content to indicate which functions are available only in this package.<br>• For pin 1, changed DCU_VSYNC_TCON2 to DCU_VSYNC.<br>• For pin 2, changed DCU_HSYNC_TCON1 to DCU_HSYNC.<br>• For pin 3, changed DCU_DE_TCON3 to DCU_DE.<br>• For pin 16, changed DCU_TAG_TCON3 to DCU_TAG.<br>• For pin 56, changed PDI0(VIU0) to PDI0_VIU0.<br>• For pin 57, changed PD11(VIU1) to PDI1_VIU1.<br>• For pin 58, changed PDI12(VIU2) to PDI2_VIU2.<br>• For pin 59, changed PDI3(VIU3) to PDI3_VIU3.<br>• For pin 116, changed PD113 to PDI13.<br>• For pin 117, changed PD114 to PDI14.<br>• For pin 148, deleted TCON0.<br>• For pin 155, deleted DCULITE_G7.<br>• For pin 157, changed eMIOSB19Q to eMIOS1[19].<br>In the 208-pin pinout:<br>• For pin 68, changed eMIOSA23 to eMIOS0[23].<br>• For pin 69, changed eMIOSA16 to eMIOS0[16].<br>• For pin 70, changed eMIOSA15 to eMIOS0[15].<br>• For pin 71, changed eMIOSA14 to eMIOS0[14].<br>• For pin 165, changed VRSDS to VREF_RSDS.<br>In the 324-pin pinout:<br>• Added content to indicate which functions are available only in this package.<br>• For pins C19 and D19, changed VRSDS to VREF_RSDS.<br>• Renamed pin J10 (was VSS_DR, is VSS).<br>In the “Voltage supply pin descriptions” table:<br>• Revised the entries for $V_{DDA}$ and $V_{DDE_A}$.<br>• Deleted the entry for VSS_DR.<br>• Added pin J10 to the VSS group in the 324-pin package.<br>Moved the VREF_RSDS specification from the “Voltage supply pin descriptions” table to the “System pin descriptions” table.<br>Revised the “Nexus pins” table.<br>In the “Port pin summary” table:<br>• Added a footnote to the “Peripheral” column title.<br>• For port PJ[4], changed the reset configuration (was “Input, Pull Up”, is “None, None”).<br>• For port PJ[6], changed the reset configuration (was “Input, Pull Up”, is “None, None”).<br>• For port PG[8], deleted TCON[2].<br>• For port PG[9], deleted TCON[1].<br>• For port PG[10], deleted TCON[3].<br>• For port PJ[12], deleted TCON0.<br>• For port PK[0], deleted DCULITE_G7.<br>• For port PM[5], deleted the TCON references. |
| Safety | In the SWT section, deleted text referring ambiguously to device-specific reset values.<br>In the SWT_TO section, changed “Default counter value (SWT_TO_RST) is 1280 (0x00000500 hexadecimal) which correspond to around 10 ms with a 128 kHz clock” to “The default counter value (SWT_TO_RST) is 1280 (0x0000_0500 hexadecimal) which corresponds to 10 ms with a 128 kHz clock”.<br>Revised the SWT > “Functional Description” section. |

**Table A-6. Changes between revisions 2 and 3 (continued)**

| Chapter | Description |
|---|---|
| Boot Assist Module | Added the “Flash memory password swapping” section and cross-references to this section as appropriate.<br>Changed “and ABD fields” to “field” in the “BAM software flow” section.<br>Corrected numbering of steps in the “Download and execute the new code“ section.<br>Changed text describing comparison done by the BAM code in the “Download 64-bit password and password check“ section.<br>Changed text describing mode for SEC = 1 to “static” in the “Download 64-bit password and password check“ section. |
| Clock Architecture | Renamed the chapter to “Clock Description”.<br>Consolidated the following content into this chapter:<br>• Clock Generation Module<br>• Oscillators (includes the correct information about the Pierce oscillator as FXOSC)<br>• Clock Monitor Unit<br>Revised the “MPC5645S system clock generation” figure.<br>Added a note to the “System clock generation” section. |
| Clock Generation Module | Migrated this chapter’s content to the new “Clock Description” chapter.<br>In the “MC_CGM block diagram” figure, changed “core” to “CPU”.<br>In the CGM_AC4_SC section, added a field description table.<br>In the CGM_AC5_DC[DIV0] field description, changed “the DCULite clock remains disabled” to “the DRAM clock remains disabled”.<br>Changed SELCTL description of case 0110 to “reserved” in “Output Clock Division Select Register “ section.<br>Changed case 6 to “reserved“ in “Output Clock Division Selection“ section. |
| Clock Monitor Unit | Migrated this chapter’s content to the new “Clock Description” chapter. |
| Configurable Enhanced Modular I/O Subsystem | Extensive modifications throughout the chapter. |
| Crossbar Switch | Revised the “Information specific to this device” section.<br>Deleted the footnote from the Overview section.<br>Deleted the “XBAR block diagram” figure.<br>Revised the reset value of the Master Priority Register (was undefined, is 0x0654_3210).<br>Changed section title “XBAR master IDs” to “XBAR master Port Numbers”.<br>Added text to “XBAR master Port Numbers” section.<br>Changed “IDs“ to “port numbers” in table title in “XBAR master Port Numbers” section.<br>Changed “ID“ to “Port Number” in table heading in “XBAR master Port Numbers” section.<br>Changed section title of “XBAR master Port Numbers” to “XBAR Master ID Numbers“ and corrected minor problems within the section. |
| Deserial Serial Peripheral Interface | Moved content from the Introduction section to a new “Device-specific information” section.<br>Revised the Features section. |

**Table A-6. Changes between revisions 2 and 3 (continued)**

| Chapter | Description |
|---|---|
| Display Control Unit | In the “DCU3 block diagram” figure, renamed the input signals to the “Parallel data interface” block.<br>In the Features section:<br>• Added “Optional support for panels without built in TCON features using the on-chip TCON module”.<br>• Changed “ARGB1555, ARGB4444, and ARGB8888 direct colors” to “ARGB1555, ARGB4444, and BGRA8888 direct colors”.<br>Revised the “External signal description” section.<br>In the CtrlDescL0_4[BPP] field description, revised the entry for 4’b0110 (was “32 bpp (ARGB8888)”, is “32 bpp (BGRA8888)”.<br>Revised the DCU_MODE[DDR_MODE] field description.<br>Revised the DCU_MODE[PDI_SYNC] field description.<br>Revised the field names in the “PDI Status Mask Register” section.<br>Revised the “TFT LCD panel configuration” section.<br>Added the “RLE decoding scheme” section.<br>Revised the “Graphics and data format” section.<br>In the “Alpha and chroma key blending” section, changed ARGB8888 to BGRA8888.<br>In the “Run Length Encoding (RLE) mode” section, changed ARGB8888 to BGRA8888.<br>Revised the “Managing the DCU3 FIFOs and DMA activity” section.<br>Revised the “Safety mode” section.<br>Revised the “PDI interface description” section.<br>Revised the “Glossary” table. |
| Display Control Unit Lite | In the “DCULite block diagram” figure, renamed the input signals to the “Parallel data interface” block.<br>In the Features section, changed “ARGB1555, ARGB4444, and ARGB8888 direct colors” to “ARGB1555, ARGB4444, and BGRA8888 direct colors”.<br>Renamed the signals in the “External signals” figure.<br>Renamed the signals in the “Detailed signal descriptions” table.<br>In the CtrlDescL0_4[BPP] field description, revised the entry for 4’b0110 (was “32 bpp (ARGB8888)”, is “32 bpp (BGRA8888)”.<br>Revised the DCU_MODE[DDR_MODE] field description.<br>Revised the DCU_MODE[PDI_SYNC] field description.<br>Revised the field names in the “PDI Status Mask Register” section.<br>Renamed the signals in the “TFT LCD panel configuration” section.<br>Revised the “Graphics and data format” section.<br>In the “Alpha and chroma key blending” section, changed ARGB8888 to BGRA8888.<br>In the “Run Length Encoding (RLE) mode” section, changed ARGB8888 to BGRA8888.<br>Revised the “Managing the DCU3 FIFOs and DMA activity” section.<br>Revised the “DCULite mode selection and background color” section.<br>Revised the “Special DDR mode” section.<br>Added the “RLE decoding scheme” section.<br>Revised the “Safety mode” section.<br>Revised the “PDI interface description” section.<br>Added the “Glossary” section. |
| DRAM Controller | Revised the “DRAMC block diagram” figure.<br>In the Features section, changed “Five incoming master buses” to “Seven incoming masters shared on five master busses”.<br>Changed “256“ to “128” in “DQS Config Offset Time field descriptions” section.<br>Added bullet regarding Electrical characteristics of DRAM interface signals in “Connecting the DRAM” section. Made SIU chapter reference into a cross-reference. |

**Table A-6. Changes between revisions 2 and 3 (continued)**

| Chapter | Description |
|---|---|
| DRAMC Priority Manager | In the Introduction section, changed references to 5 busses to 7 busses.<br>In the “Description of operation — block diagram” section, changed “PowerPC” to “e200z4d” and “AXE” to “GFX2D”.<br>Corrected reset values in “Priority manager memory map“ table. |
| e200z4d Core | Added footnote 2 in “e200z4d Supervisor Mode Programmer’s Model“ figure. |
| Enhanced Direct Memory Access | In the “TCDn Word 7 (TCDn.{biter, control/status}) field descriptions” table, in the int_half description, replaced the text “The halfway complete interrupt... less than two” with the note “If biter = 1, do not use int_half; use int_major instead.”<br>Removed pseudocode examples within the “Features” section. |
| eDMA Channel Mux | Revised the Introduction section.<br>In the “DMACHMUX request assignments” table, marked source numbers 47–50 as reserved. |
| Error Correction Status Module | Revised the Introduction section.<br>In the “Memory map and register description” section, revised content to reflect the fact that addresses 0x00–0x20 are reserved.<br>In the MUDCR section, replaced the “AXBS_lite force_round_robin bit” description with the “PRAM_ADD_ONE_WS” bit description.<br>In the EEGR section, added information about the RWBIT field.<br>Changed text in the “Introduction” section.<br>Removed AXBS_lite priority functions description in section "Features".<br>Removed introductory text regarding the programming model in “Memory map and register description” section.<br>Removed introductory text regarding control logic in the "Memory Map" section.<br>Changed description of table from “memory map” to “register map” in the “Memory Map” section.<br>Changed name of section from “Spp_ips_reg_protection“ to “Supervisor mode access protection”, plus the same change to reference within this section.<br>Removed the reference to the spp_ips_reg_protection module block diagram in the “Supervisor mode access protection“ section.<br>Removed “Spp_Ips_Reg_Protection Block Diagram”, Figure title and reference. |
| Flash Memory | Replaced the entire chapter. |
| FlexCAN | In the Overview section, replaced “Device User Guide” with “the device-specific information section”.<br>In the “FlexCAN module features” section, changed “either 1056 bytes” to “up to 1024 bytes” and “either 256 bytes” to “up to 256 bytes”.<br>Added the “Device-specific information” section.<br>Revised the “Interrupts” section.<br>In the “FlexCAN initialization sequence” section, deleted “and in MCR for Wake-Up interrupt”. |
| Frequency-Modulated Phase-Locked Loop | Migrated this chapter’s content to the new “Clock Description” chapter.<br>In the Overview section, changed “to 16 MHz input clock” to “120 MHz input clock”.<br>Revised the Features section. |
| Graphics Accelerator Gasket | Revised the “Alpha buffer write suppressor” section.<br>Revised the “Serial flash exclusive access” section.<br>In the “Frame buffer color depth converter” section, changed “with 16-byte aligned input addresses” to “with 16-byte aligned input addresses and a GFX2D stride of 4096”. |

**Table A-6. Changes between revisions 2 and 3 (continued)**

| Chapter | Description |
|---|---|
| Graphics Static RAM | Changed “AIPS” to “PBRIDGE”.<br>Revised the Features section.<br>Revised the “Functional description” section. |
| Interrupt Controller | In the Introduction section, changed “supports 187 interrupt requests” to “supports 171 interrupt requests”.<br>In the Features section, changed “Supports 179 peripheral” to “Supports 163 peripheral”.<br>Removed the “Interrupt sources available” table.<br>In the “Interrupt vectors” table:<br>• Changed “EMIOS_” to “EMIOS0_” and “EMIOS1_” as appropriate.<br>• Changed “core” to “CPU”.<br>• Labeled IRQ 44 as reserved.<br>• Changed the IRQ for offset 0x0310 (was 50, is 49).<br>• Labeled IRQs 63, 67, 87, and 107 as reserved.<br>• Changed the resource for IRQ 183 (was TBD, is SGM_IRQ).<br>• Revised the entries for IRQs 228–233. |
| LIN Controller | Added the “Device-specific information” section.<br>Added the “External signal description” section.<br>In the “Fractional baud rate generation” section, changed the second note (is “In LIN mode, LFDIV must be greater than or equal to 1.0. In UART mode, LFDIV must be greater than or equal to 1.5.”). |
| Memory Protection Unit | In the MPU_EDRn[EATTR] field description, changed “non-core” to “non-CPU”.<br>Added the “Opcode pre-fetch cycles and the execute permission” section.<br>In the MPU_CESR[NSP] field description, added “The MPC5645S has 4 slaves connected to the MPU.”.<br>In the MPU_CESR[NRGD] field description, added “(the number of descriptors on the MPC5645S)” to the last encoding. |
| Mode Entry Module | Changed “core” to “CPU”.<br>In the “MC_ME register description” table, deleted the “Access>Test” column.<br>Changed the ME_STANDBY0_MC[FXOSCON] field (was read-only, is read/write). |
| Nexus Development Interface | Changed “core” to “CPU”.<br>Added the “Relationship between TCK and system clock frequency” section and added cross-references to it from the PCR field descriptions. |
| Oscillators | Migrated this chapter’s content to the new “Clock Description” chapter.<br>In the FXOSC section, deleted the “Wait mode operation” and “Stop mode operation” sections. |
| Peripheral Bridge | Deleted the “Modes of operation” section. |
| Pierce Oscillator | Migrated this chapter’s content to the new “Clock Description” chapter. (On this device, the Pierce oscillator is the FXOSC).<br>In the Introduction section, changed $V_{DDPLL}$ to $V_{DDE_B}$.<br>In the Introduction section, changed “(3V or 5V)” to “(3.3 V)”.<br>In the Features section, changed “Operates from internally generated power supply (3V or 5V)” to “Operates from externally provided power $V_{DDE_B}$ (3.3 V)”.<br>Revised the “XOSC block diagram” figure. |
| Power Control Unit | In the “MC_PCU block diagram” figure, changed “core” to “CPU”.<br>In the “MC_PCU register description” table, deleted the “Access > Test” column. |

**Table A-6. Changes between revisions 2 and 3 (continued)**

| Chapter | Description |
|---|---|
| Quad Serial Peripheral Interface | In the “QuadSPI block diagram” figure, changed “core clock domain” to “system clock domain”.<br>In the Features section, changed “flash devices of up to 64 MB in size” to “flash devices of up to 16 MB in size”.<br>In the QSPI_MCR section, added a caution about writable reserved bits.<br>Revised the “RX Buffer Data Registers 0–14” section.<br>In the “AHB RX data buffer” section, changed “depends from the number of RX Buffer entries implemented and from the number of valid buffer entries” to “depends on the number of valid buffer entries”. |
| Real-Time Clock | Revised the clock names in the RTCC[CLKSEL] field description. |
| Reset Generation Module | Changed “core” to “CPU”.<br>In the “Modes of operation” section, replaced “code or data flash fatal error with “FMPLL1 fail fatal error”.<br>Revised the RGM_FERD and RGM_FEAR definitions.<br>In the RGM_FES register, changed bit 12 (was F_CHKSTOP, is reserved).<br>In the RGM_FERD register, changed bit 12 (was D_CHKSTOP, is reserved).<br>In the RGM_FERD register, changed the D_CORE field from read/write to read-only.<br>In the RGM_FEAR register, changed the AR_FMPLL1 field from read-only to read/write.<br>In the RGM_FESS register, changed bit 12 (was SS_CHKSTOP, is reserved).<br>In the RGM_FBRE register, changed bit 12 (was BE_CHKSTOP, is reserved).<br>In the RGM_FBRE register, changed bit 7 (was BE_FLASH, is reserved).<br>Revised the “MC_RGM reset implications” table. |
| Run-Length Encoding Decoder | Replaced the entire chapter. |
| RSDS Interface | Deleted this chapter (its contents have been migrated to the “Timing Controller” chapter). |
| Sound Generator Module | Replaced the entire chapter. |
| Static RAM | Changed “sleep” to “standby” throughout.<br>Changed “preserves the contents of the portion” to “preserves the contents of all or a portion”. |

**Table A-6. Changes between revisions 2 and 3 (continued)**

| Chapter | Description |
|---|---|
| System Integration Unit Lite | In the Features section, changed “4 system interrupt vectors” to “3 system interrupt vectors”.<br>Renamed the “SIUL signal properties” table (is “SIUL external interrupt mapping”), revised its content, and moved it to the “External interrupt request input pins (EIRQ[0:23])” section.<br>In the “Register protection” section, added PCR[n] to the list.<br>Revised the PSMI0_3–PSMI50_53 section.<br>Revised the IFMC[MAXCNTx] field description.<br>Revised the IFCPR[IFPC] field description.<br>Revised the “Parallel GPIO Pad Data Out Registers” section.<br>Revised the MPGPDO0 section.<br>In the “External interrupts” section, changed “Each vector interrupt has from five to eight” to “Each vector interrupt has eight”.<br>Replaced text in the “Overview” section.<br>Replaced all text in the “External Signal Description” section.<br>Changed “slew rate settings” table title to indicate PCR202-268.<br>Replaced text in the “Pad Selection for Multiplexed Inputs Registers“ section.<br>Replaced text in the "Parallel GPIO Pad Data Out Registers" section.<br>Replaced text in the “Parallel GPIO Pad Data In Register” section.<br>Added note regarding bit-ordering at the end of the “Parallel GPIO Pad Data In Register“ section.<br>Replaced text in the “Masked Parallel GPIO Pad Data Out Register“ section.<br>Added text to the “Interrupt Filter Maximum Counter Registers“ section.<br>Added text to the “General purpose input and output pads” section.<br>Changed number of interrupt vectors from “four” to “three” in the “External Interrupts“ section.<br>Modified the signal lines in the “External interrupt pad diagram” figure. |
| System status and configuration module | Added text to the “Functional Description” section. |
| Timing Controller | Replaced the entire chapter, merging the contents of the previous TCON and RSDS chapters. |

**Table A-6. Changes between revisions 2 and 3 (continued)**

| Chapter | Description |
|---|---|
| Voltage Regulators and Power Supplies | In the “Power-up sequencing” section, revised the second step in the sequence.<br>In the “GPIO power supply configuration”, revised the first paragraph.<br>In the “Power supply configuration” figure:<br>• Changed “FMPLL” to “FMPLL0/1”.<br>• Changed VSS_DR to VSS.<br>Revised the VREG_CTL field descriptions.<br>Revised the “GPIO power bank supplies and functionality” table.<br>In the “Power domain organization” section, changed “and a RAM power domain RD1” to “and a RAM-only power domain PD2”.<br>In the HPREG section, revised the description of how the HPREG is disabled.<br>In the “Low Voltage Detectors (LVD) and Power On Reset (POR)” section, changed “for CORE” to “for the core digital logic”. |
| Wakeup Unit | In the Overview section, changed “generate either interrupts or wakeup events” to “generate both interrupts and wakeup events”.<br>In the Features section, changed “configurable pullup” to “configurable pullup (except in STANDBY mode)”.<br>In the “External signal description” section:<br>• Changed “24 signal inputs” to “24 external signal inputs”.<br>• Changed part of the note text (was “Any unused Wake-up signal input should be terminated by using an external pull-up or pull-down, or by internal pull-up enabled at WIPUER”, is “Any unused Wake-up signal input should be terminated by using an external pull-up or pull-down, or by internal pull-up enabled at WIPUER (not available in STANDBY mode)”).<br>Added the “WKPU behavior” section.<br>Revised the “External wakeups and interrupts” section.<br>In the “External interrupt management” section, changed “using a single rolled-up register” to “using the Interrupt Request Enable Register (IRER)”.<br>Deleted the “On-chip wakeups” section.<br>Replaced the “On-chip wakeup management” section. |

## A.7 Changes between revisions 1 and 2

**Table A-7. Changes between revisions 1 and 2**

| Chapter | Description |
|---|---|
| Throughout | Editorial changes and improvements. |
| Introduction | In the device-comparison table:<br>• Revised the “Timed I/O” entry.<br>• Changed “Nexus 3” to “Nexus Class 3”.<br>In the device block diagram:<br>• Modified the connection between the core, crossbar switch, and DRAM controller.<br>• Changed the port splitter from a blue rectangle to a thick solid line.<br>• Modified the connection between GFX2D and PBRIDGE.<br>In the feature summary:<br>• Revised the information for the TCON and RSDS interface.<br>• Changed “OpenVG” to “OpenVG 1.1”.<br>• In the ADC entry, changed “20 internal channels” to “Up to 20 internal channels”.<br>• In the QuadSPI entry, changed “quad modes of operation” to “quad IO serial flash memory”.<br>• In the DCU3 and DCULite entry, changed “WVGA” to “XGA”.<br>In the feature details:<br>• In the QuadSPI section, changed “Maximum frequency 80 MHz” to “Maximum serial clock frequency 80 MHz”.<br>• In the FlexCAN section, changed “The FlexCan modules offer” to “Each FlexCAN module offers”.<br>• Revised the DCU3 section.<br>• In the DCULite section, changed “WVGA” to “XGA”. |
| Memory Map | Revised the “Emulation mapping” entries.<br>Changed “AIPS” to “PBRIDGE”.<br>Revised the “External, memory mapped serial flash for QuadSPI” entries.<br>Added an entry for DRAM (upper 512 MB) and revised the surrounding entries.<br>Revised the reserved-space entries below PBRIDGE(1). |

**Table A-7. Changes between revisions 1 and 2 (continued)**

| Chapter | Description |
|---|---|
| Signal Description | Changed the naming convention for the eMIOS pins/signals.<br>In the 176-pin pinout:<br>• Changed “MDD” to “MDO”.<br>• For pin 1, changed DCU_VSYNC_TCON0 to DCU_VSYNC_TCON2.<br>• For pin 3, changed DCU_DE_TCON2 to DCU_DE_TCON3.<br>• For pin 148, changed TCON3 to TCON0.<br>In the 208-pin pinout:<br>• For pin 1, changed DCU_VSYNC_TCON0 to DCU_VSYNC_TCON2.<br>• For pin 3, changed DCU_DE_TCON2 to DCU_DE_TCON3.<br>• For pin 82, changed DCU_TAG_TCON3 to DCU_TAG_TCON0.<br>• For pin 105, changed “VDDE” to “VDDE_B”.<br>• Renamed pin 145 (was VRSDS, is VREF_RSDS”)<br>• For pin 148, changed “RSRS3+P” to “RSDS3P”.<br>• For pin 160, changed “SCL_D” to “SCL_3”.<br>• For pin 161, changed “SDA_D” to “SDA_3”.<br>• For pin 178, changed “DCULITE_G8” to “DCULITE_G6” and DCU_TAG_TCON3 to DCU_TAG_TCON0.<br>In the 324-pin pinout:<br>• Renamed pin A1 (was VSS_DR, is VSS).<br>• Renamed pin B2 (was VDDE_DDR, is VDD_DR).<br>• Renamed pins D11 and J9 (were VDD33_DR, are VDDE_B).<br>Added the “Pad configuration during reset phases” section.<br>In the “Voltage supply pin descriptions” table:<br>• Renamed $V_{RSDS}$ to $V_{REF_RSDS}$ and added a footnote about the pin’s required termination.<br>• Added pin B2 to VDD_DR in the 324-pin package.<br>• Moved pins D11 and J9 to $V_{DDE_B}$.<br>Revised the “System pin descriptions” table.<br>Replaced the entire “DRAM interface pin summary” table.<br>In the “Nexus pins” table, added a PCR column.<br>Added the “VIU muxing” section.<br>Added the “SGM muxing” section.<br>Added the “RSDS special function muxing” section.<br>In the “Port pin summary” table, revised content for the following entries:<br>• PB[7]<br>• PB[8]<br>• PB[10]<br>• PB[11]<br>• PC[10]<br>• PF[6]<br>• PG[8]<br>• PG[10]<br>• PJ[3]<br>• PJ[4]<br>• PJ[12]<br>• PJ[15]<br>• PK[1]<br>• PK[6]<br>• PK[7]<br>• PK[8]<br>• PK[9]<br>• PL[0]–PL[3]<br>• PM[5] |

**Table A-7. Changes between revisions 1 and 2 (continued)**

| Chapter | Description |
|---|---|
| Analog-to-Digital Converter | Revised the “Device-specific features” section.<br>Revised the “ADC implementation” figure.<br>In the Introduction section, changed “A conversion can be triggered by software or hardware (PIT3)” to “An injected conversion can be triggered by software or hardware (PIT3). A normal conversion can be triggered by software or hardware (PIT2).”<br>In the “ADC digital registers” table, added information for CDR42, CDR49, and CDR50.<br>n the CTR[1..2] figure, changed the reset value of INPSAMP (was 0b101, is 0b11).<br>Revised the CDR section, figure, and table headings (were CDR[0..95], are CDR*n*). |
| Boot Assist Module | In the “Features” section, changed “whole memory map” to “startup code”.<br>In the RCHW section, added the VLE bit.<br>Revised the “Single Chip boot mode” section.<br>In the “Download and execute the new code” section, deleted the footnote from step 2.<br>Revised the “Download start address, VLE bit and code size” section. |
| CAN Sampler | In the “Main features” section, changed “CAN0RX or CAN1RX” to “CAN0RX, CAN1RX, or CAN2RX”.<br>Revised the CR[CAN_RX_SEL] field description. |
| Clock Architecture | Revised the “System clock generation” section.<br>Added the “Auxiliary clocks” section. |
| Clock Generation Module | In the Overview section, changed “The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC_CGM memory space” to “The MC_CGM memory space also includes the control registers of the clock sources themselves, for example PLLs, IRCs, and oscillators”.<br>In the “MC_CGM block diagram” figure, changed PA[0] to PF[15].<br>In the “External signal description” section, changed “PA[0] pin” to “PF[15] pin”.<br>Removed the trailing zeros from several register names.<br>Added the CGM_VIU_MUX register.<br>In the CGM_SC_DC0...3 section, added the text “The divided clock is the reference for the associated peripheral set.”.<br>In the CGM_AC0_SC section, replaced the text “See Figure 8-19 for details” with an actual field-description table.<br>Revised the “MC_CGM system clock generation overview” figure.<br>Changed the addresses In the following figures:<br>• CGM_AC3_SC (was 0xC3FE_0390, is 0xC3FE_0398)<br>• CGM_AC3_DC (was 0xC3FE_0394, is 0xC3FE_039C)<br>• CGM_AC4_SC (was 0xC3FE_0390, is 0xC3FE_03A0)<br>• CGM_AC4_DC (was 0xC3FE_0394, is 0XC3FE_03A4)<br>• CGM_AC5_SC (was 0xC3FE_0390, is 0xC3FE_03A8)<br>• CGM_AC5_DC (was 0xC3FE_0394, is 0XC3FE_03AC)<br>Revised the “System clock dividers” section.<br>Added the “Auxiliary clock generation” section. |

**Table A-7. Changes between revisions 1 and 2 (continued)**

| Chapter | Description |
|---|---|
| Configurable Enhanced Modular I/O Subsystem | Revised the “Unsupported features” section.<br>Revised the “eMIOS clocking configuration” figure.<br>Revised the “eMIOS1 channel configurations” table.<br>In the “Channel mode selection” table, added entries for “Modulus Counter Buffered (Up/down counter)” and “Quadrature Decode”.<br>Added a footnote to the “eMIOS200 block diagram” figure.<br>Revised the “Features” section.<br>Changed the EMIOSMCR[MDIS] reset value (was 0/1, is 0) and deleted the associated footnote.<br>Revised the EMIOSGFLAG section.<br>Revised the EMIOSOUDIS section.<br>Revised the EMIOSOUCDIS section.<br>Revised the EMIOSA[n], EMIOSB[n], EMIOSCNT[n], and EMIOSALTA[n] register figures.<br>In the “UC BSL bits” table, deleted the entries for channels 0 to 7.<br>In the “UC MODE bits” table, added entries for Quadrature Decode mode. |
| Crossbar Switch | Revised the “Device-specific block diagram” figure. |
| Display Control Unit | In the “DCU3 block diagram”, changed “CLUT/Tile RAM (6KB)” to “CLUT/Tile RAM (8 KB)”.<br>Revised the CtrlDescL0_2 section.<br>Revised the reset values for the INT_MASK, Mask_PARR_ERR_STATUS, THRESHOLD_INP_BUF_1, and THRESHOLD_INP_BUF_2 registers.<br>In the “DISP_SIZE Register” figure, changed the field name for bits 25–31 (was DELTA_Y, is DELTA_X).<br>Revised the LYR_INTPOL_EN field description and reset values.<br>Revised the DCU_MODE[DCU_MODE] field description.<br>Changed the reset value of the COMP_IMSIZE register (was 0x00000409, is 0x00000000).<br>In the CtrlDescL0_3[ADDR] field description, changed “32-bit aligned” to “64-bit aligned”.<br>Revised the “Graphics and data format” section.<br>In the “Operation of scheme” section, changed IVOR2 to IVOR1.<br>Revised the “DCU3 mode selection and background color” section.<br>Deleted the “Dithering block diagram” figure.<br>Revised the “Special DDR mode” section.<br>Changed “PDI YCbCr Mode” to “PDI YCbCr mode and DCU YCbCr color format” and revised that section.<br>Added the “Switch between DCU mode and PDI mode (top-level description)” section.<br>Revised the “DCU3 initialization” section. |

**Table A-7. Changes between revisions 1 and 2 (continued)**

| Chapter | Description |
|---|---|
| Display Control Unit Lite | In the “Overview” section, changed “blending the values of up to four separate graphics” to “blending the values of up to two separate graphics”.<br>In the “Features” sction, changed “1, 2, 4 and 8 bits per pixel indexed colors” to “1, 2, 4 and 8 bits per pixel indexed colors with alpha channel”.<br>In the “DCULite memory map”, changed the last entry (was “CLUT/tile address space”, is “CLUT address space (2 KB repeated)”).<br>Revised the CtrlDescL0_2 section.<br>Revised the CtrlDescL0_4[LUOFFS] field description.<br>Revised the DCU_MODE field descriptions.<br>Revised the “PARR_ERR Status Register (PARR_ERR_STATUS)” section.<br>Revised the LYR_INTPOL_EN[EN] field description.<br>Revised the “DCULite mode selection and background color” section.<br>Revised the “Graphics and data format” section.<br>Deleted the “Dithering block diagram” figure.<br>Revised the “Tile mode” section.<br>Revised the “Special DDR mode” section.<br>Renamed the “CLUT/Tile RAM” section (is “CLUT RAM”) and revised it.<br>In the “Synchronizing to panel frame rate” section, changed “CLUT or tile memory” to “CLUT memory”.<br>Revised the “Error detection” section.<br>Renamed the “Debugging of Display Control Unit” section (is “Debugging of DCULite”) and revised it.<br>In the “DCULite initialization” section, changed “Load the initial tile or palette colors into the CLUT/Tile memory” to “Load the palette colors into the CLUT memory”.<br>Fixed several invalid cross-references in the “DCULite register map” table.<br>In the “DCULite register map” table, changed CHROMA_RED to RED, CHROMA_GREEN to GREEN, and CHROMA_BLUE to BLUE to match the names used in the register description.<br>In the Features section, changed “UXGA (1600x1200)” to “XGA (1024×768)”.<br>Changed the reset value of the COMP_IMSIZE register (was 0x00000409, is 0x00000000).<br>In the CtrlDescL0_3[ADDR] field description, changed “32-bit aligned” to “64-bit aligned”.<br>In the CtrlDescL0_7 section, changed “For the other 16 layers” to “For the other 3 layers”.<br>Revised the “Blending priority of layers” section.<br>In the “Control Descriptors” section, changed “all 16 control descriptors” to “all 4 control descriptors”.<br>In the “Hardware cursor” section, changed “In addition to the 16 layers” to “In addition to the 4 layers”.<br>In the “Operation of scheme” section, changed IVOR2 to IVOR1.<br>Revised the “YCbCr mode” section.<br>Revised the “Switch between DCU mode and PDI mode (top-level description)” section. |
| DMA Channel Mux | Changed the module name (was “DMA Channel Mux”, is “eDMA Channel Mux”).<br>Revised the “DMACHMUX request assignments” table. |
| DRAM Controller | Revised the “Timing parameters” table.<br>Revised the “Write to precharge timing diagram” and “Write to read timing diagram” figures.<br>Removed the footnote in the “DQS Delay Status” section. |
| e200z4d Core | Revised the footnote to the “Exceptions and conditions” table. |
| Error Correction Status Module | Changed “sleep modes” to “low-power modes”. |

**Table A-7. Changes between revisions 1 and 2 (continued)**

| Chapter | Description |
|---|---|
| Flash Memory | Changed “sleep mode” to “Low Power mode” and “disable modes” to “Power Down mode”.<br>Removed extraneous content.<br>Replaced several registers as follows:<br>• BIU0 with PFCR0<br>• BIU1 with PFCR1<br>• BIU2 with PFAPR<br>• BIU3 with PFSACC<br>Deleted the BIU4 register. |
| FlexCAN | Renamed several registers as follows:<br>• IFLAG1 to IFRL<br>• IFLAG2 to IFRH<br>• IMASK1 to IMRL<br>• IMASK2 to IMRH |
| Frequency-Modulated Phase-Locked Loop | Revised the CR section.<br>In the “Normal mode” section, revised the equation for phi to use new terms ($R_{LOOP}$, $R_{INP}$, and $R_{OUT}$).<br>In the “Normal Mode with frequency modulation” section, changed SSCG_EN to FM_EN. |
| Internal Static RAM | Changed the module name (was “Internal Static RAM”, is “Graphics Static RAM”).<br>In the Features section:<br>• Changed “Up to 8MBytes” to “1 MB”.<br>• Changed “Status register available for the system to monitor operation” to “Status register available for the system to monitor fill operation”.<br>Revised the PRAM2P_CR field descriptions.<br>Revised the “Master assignments” table. |
| Interrupt Controller | In the “Interrupt sources available” table, revised the ECSM entry (was 3, is 2).<br>Revised the “Interrupt vector table” table. |
| Memory Protection Unit | Revised the MPU_CESR reset value. |
| Mode Entry Module | Revised the “Clock sources (main voltage regulator independent) switch-on” section. |
| Oscillators | Replaced the FXOSC content. |
| Periodic Interrupt Timer | Changed the module abbreviation (was PIT_RTI, is PIT).<br>Revised the Introduction section. |
| Quad Serial Peripheral Interface | Renamed registers as follows:<br>• QSPI_SR is now QSPI_SFMSR<br>• QSPI_FR is now QSPI_SFMFR<br>• QSPI_RSER is now QSPI_SFMRSER<br>Revised the QSPI_MCR section.<br>In the “RX Buffer Data Registers 0 - 14 (QSPI_RBDR0 - QSPI_RBDR14)” section, changed Example 1 to state “to QSPI_RBDR4” instead of “to QSPI_RBDR15”.<br>Added an empty column to the “ICO field on Winbond Devices - Parallel Flash Mode” table. |
| Real-Time Clock | In the “Device specific information”, changed “RCOSC128K” to “SIRC”.<br>Revised the RTCC[APIVAL] field description.<br>In the “API functional description” section, changed “reaches the offset count” to “reaches (offset count + 1)”. |
| Reset Generation Module | Replaced the entire chapter. |

**Table A-7. Changes between revisions 1 and 2 (continued)**

| Chapter | Description |
|---|---|
| RSDS Interface | Revised the “RSDS option bits to increase/decrease the output swing (coarse adjustment)” table.<br>Revised the “RSDS option bits to increase/decrease the output swing (fine adjustment)” table.<br>Revised the “RSDS option bits to increase the skew” table. |
| Run-Length Encoding Decoder | Revised the Features section.<br>- Revised the RLE_DEC_MCR section.<br>- Renamed fields in the following registers:<br>• RLE_DEC_ICR<br>• RLE_DEC_CISR<br>• RLE_DEC_DICR<br>• RLE_DEC_SR<br>• RLE_DEC_SPCR<br>• RLE_DEC_EPCR<br>Revised the “Interrupt and DMA request signals” section.<br>In the RLE_DEC_MCR figure, changed the reset value of bit 17 (was 1, is 0).<br>Revised the RLE_DEC_ISR section.<br>Revised the RLE_DEC_RIER section.<br>Revised the “Power-saving features” section.<br>Revised the “Functional description” section. |
| Safety | Replaced the entire SWT content. |
| Sound Generator Module | Extensive revisions throughout the chapter. |
| Static RAM | Added this chapter. |
| Stepper Motor Controller | Revised the “Impact of MCCTL1[RECIRC] and MCDCx[SIGN[4]] Bit on the PWM Output” table. |
| System Integration Unit Lite | In the Features section, added “Special function control” information.<br>Revised the “SIUL signal properties” table.<br>In the “Detailed signal descriptions” section, added the “Special function output pins configuration (PCR[185:281])” subsection.<br>Revised the “SIUL memory map” table.<br>Revised the MIDR1 reset value.<br>Revised the MIDR2 reset value and field descriptions.<br>Revised the “Pad Selection for Multiplexed Inputs Registers” section.<br>Revised the footnote in the ISR figure.<br>In the “GPIO Pad Data Output Registers” and “GPIO Pad Data Input Registers” headings, changed “185” to “184”.<br>In the footnote to the “External interrupt pad diagram” figure, changed “and in the 324-pin package” to “and [0:23] in the 324-pin package”.<br>Revised the PCR figure.<br>Added the “Pad Configuration Registers (PCR185–PCR281)” section.<br>Revised the “External interrupt management” section. |
| System Status and Configuration Module | Revised the STATUS register description to reflect that bit 12 is reserved. |
| Timing Controller | Revised the register map and figures to show that fields formerly indicated by empty gray areas are read-only with a read value of 0.<br>Changed the TCON_CTRL1[RSDS_MODE] reset value (was 1, is 0).<br>Revised the “TCON_DATA0_DLY - TCON_DATA12_DLY” section. |

**Table A-7. Changes between revisions 1 and 2 (continued)**

| Chapter | Description |
|---|---|
| Video Input Unit | Revised the “Brightness and Contrast Adjust” section.<br>Revised the “DMA and De-interlace” section. |
| Voltage Regulators and Power Supplies | Added the “Power-up sequencing” section.<br>Revised the “Voltage regulators” section.<br>In the VDDR section, replaced the capacitance values with a reference to see the values in the device data sheet.<br>Revised the “Low Voltage Detectors (LVD) and Power On Reset (POR)” section to indicate that the output is factory-trimmed.<br>Revised the “GPIO power bank supplies and functionality” table. |
| Wakeup Unit | Added a footnote to the “Wakeup vector mapping” table.<br>In the “External signal description” section:<br>• Changed the number of signal inputs (was 19, is 24) and added a note that wakeup pins are enabled in all modes.<br>• Changed “used as external interrupt sources” to “used as additional external interrupt sources”.<br>Revised several register figures to show individual one-bit fields instead of single multi-bit fields. |
| Appendix A: Registers Under Protection | Removed the entry for data flash memory (does not exist on this device).<br>Replaced the MC_RGM information.<br>Added entries for PCR48-281.<br>Added entries for PSMI20, PSMI24, PSMI28, PSMI32, PSMI36, PSMI40, PSMI44, PSMI48, and PSMI52.<br>Added entries for IFMC14–23. |

***How to Reach Us:***

**Home Page:**
freescale.com

**Web Support:**
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address:freescale.com/SalesTermsandConditions.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. All other product or service names are the property of their respective owners.

© 2009 - 2015 Freescale Semiconductor, Inc.

Document Number: MPC5645SRM
Rev. 8
Mar 2015