MC9S08JM60
MC9S08JM32

Data Sheet

HCS08
Microcontrollers

MC9S08JM60
Rev. 5
10/2014

freescale.com

freescale™
semiconductor

MC9S08JM60 Series Features

8-Bit HCS08 Central Processor Unit (CPU)

- 48-MHz HCS08 CPU (central processor unit)
- 24-MHz internal bus frequency
- HC08 instruction set with added BGND instruction
- Background debugging system
- Breakpoint capability to allow single breakpoint setting during in-circuit debugging (plus two more breakpoints in on-chip debug module)
- In-circuit emulator (ICE) debug module containing two comparators and nine trigger modes. Eight deep FIFO for storing change-of-flow addresses and event-only data. Debug module supports both tag and force breakpoints.
- Support for up to 32 interrupt/reset sources

Memory Options

- Up to 60 KB of on-chip in-circuit programmable flash memory with block protection and security options
- Up to 4 KB of on-chip RAM
- 256 bytes of USB RAM

Clock Source Options

- Clock source options include crystal, resonator, external clock
- MCG (multi-purpose clock generator) — PLL and FLL; internal reference clock with trim adjustment

System Protection

- Optional computer operating properly (COP) reset with option to run from independent 1-kHz internal clock source or the bus clock
- Low-voltage detection with reset or interrupt
- Illegal opcode detection with reset
- Illegal address detection with reset

Power-Saving Modes

- Wait plus two stops

Peripherals

- **USB —** USB 2.0 full-speed (12 Mbps) device controller with dedicated on-chip USB transceiver, 3.3-V regulator and USBDP pull-up resister; supports control, interrupt, isochronous, and bulk transfers; supports endpoint 0 and up to 6 additional endpoints; endpoints 5 and 6 can be combined to provide double buffering capability
- **ADC** — 12-channel, 12-bit analog-to-digital converter with automatic compare function; internal temperature sensor
- **ACMP** — Analog comparator with option to compare to internal reference; operation in stop3 mode
- **SCI** — Two serial communications interface modules with optional 13-bit break LIN extensions
- **SPI** — Two 8- or 16-bit selectable serial peripheral interface modules with a receive data buffer hardware match function
- **IIC** — Inter-integrated circuit bus module to operate at up to 100 kbps with maximum bus loading; multi-master operation; programmable slave address; interrupt-driven byte-by-byte data transfer; 10-bit addressing and broadcast modes support
- **Timers** — One 2-channel and one 6-channel 16-bit timer/pulse-width modulator (TPM) modules: Selectable input capture, output compare, and edge-aligned PWM capability on each channel. Each timer module may be configured for buffered, centered PWM (CPWM) on all channels
- **KBI** — 8-pin keyboard interrupt module
- **RTC** — Real-time counter with binary- or decimal-based prescaler

Input/Output

- Up to 51 general-purpose input/output pins
- Software selectable pullups on ports when used as inputs
- Software selectable slew rate control on ports when used as outputs
- Software selectable drive strength on ports when used as outputs
- Master reset pin and power-on reset (POR)
- Internal pullup on $\overline{\text{RESET}}$, IRQ, and BKGD/MS pins to reduce customer system cost

Package Options

- 64-pin quad flat package (QFP)
- 64-pin low-profile quad flat package (LQFP)
- 48-pin quad flat no-lead (QFN)
- 44-pin low-profile quad flat package (LQFP)

MC9S08JM60 Series Data Sheet

Covers MC9S08JM60
MC9S08JM32

MC9S08JM60
Rev. 5
10/2014

freescale™
semiconductor

Revision History

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://freescale.com/

The following revision history table summarizes changes contained in this document.

Revision Number	Revision Date	Description of Changes
1	11/27/2007	Initial release
2	3/4/2008	Changed the location of R_S to connect to EXTAL in Figure 2-4. Changed port rise and fall time in Table A-13. Added DC injection current and RAM retention voltage in Table A-6. Deleted note on 625 ns of item 17 in Table A-12. Moved Bandgap Voltage Reference item from Table A-8 to Table A-6. Added one paragraph on how to improve accuracy to Section 10.1.1.5, "Temperature Sensor."
3	1/21/2009	Changed the V_{TEMP25} from 1.396 mV to 1.396 V in Table A-10. Complete the EMC data in Section A.15, "EMC Performance." Revised the Typo in Table 11-4.
4	2/21/2010	Changed the location of R_S to connect to XTAL in Figure 2-4.
5	10/2014	Updated t_{LPO} in Table A-13.

List of Chapters

Contents

Section Number **Title** **Page**

Chapter 1
Device Overview

Chapter 2
Pins and Connections

Chapter 3
Modes of Operation

Chapter 4
Memory

Chapter 5
Resets, Interrupts, and System Configuration

Chapter 6
Parallel Input/Output

Chapter 7
Central Processor Unit (S08CPUV2)

Chapter 8
5 V Analog Comparator (S08ACMPV2)

Chapter 9
Keyboard Interrupt (S08KBIV2)

Chapter 10
Analog-to-Digital Converter (S08ADC12V1)

Chapter 11
Inter-Integrated Circuit (S08IICV2)

Chapter 12
Multi-Purpose Clock Generator (S08MCGV1)

Chapter 13
Real-Time Counter (S08RTCV1)

Chapter 14
Serial Communications Interface (S08SCIV4)

Chapter 15
16-Bit Serial Peripheral Interface (S08SPI16V1)

Chapter 16
Timer/Pulse-Width Modulator (S08TPMV3)

Chapter 17
Universal Serial Bus Device Controller (S08USBV1)

Chapter 18
Development Support

Appendix A
Electrical Characteristics

Appendix B
Ordering Information and Mechanical Drawings

Chapter 1
Device Overview

1.1 Introduction

MC9S08JM60 series MCUs are members of the low-cost, high-performance HCS08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced HCS08 core and are available with a variety of modules, memory sizes, memory types, and package types.

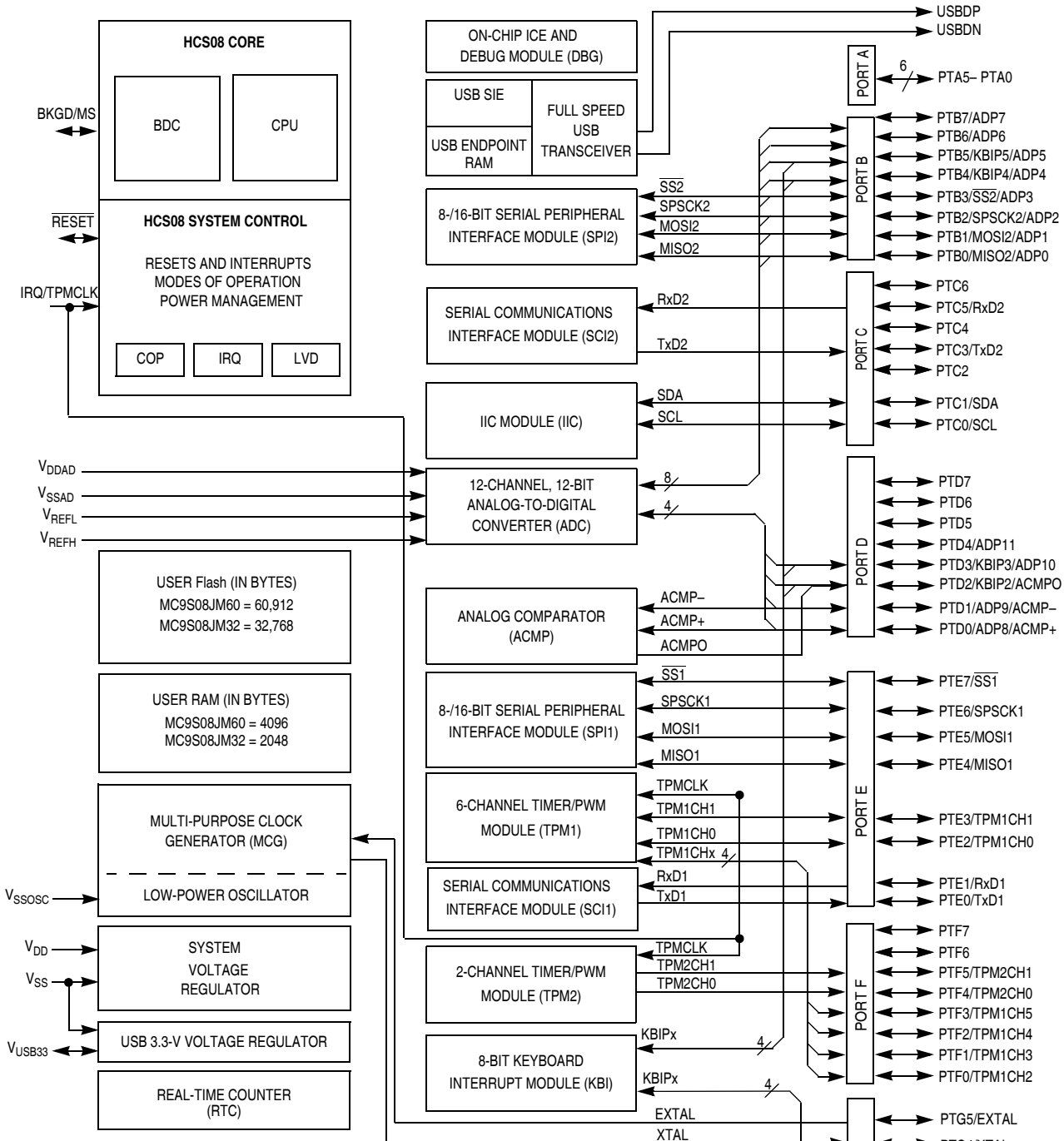
Table 1-1 summarizes the peripheral availability per package type for the devices available in the MC9S08JM60 series.

Table 1-1. Devices in the MC9S08JM60 Series

Feature	Device					
	MC9S08JM60			MC9S08JM32		
Package	64-pin	48-pin	44-pin	64-pin	48-pin	44-pin
Flash	60,912			32,768		
RAM	4096			2048		
USB RAM	256			256		
ACMP	yes			yes		
ADC	12-ch	8-ch	8-ch	12-ch	8-ch	8-ch
IIC	yes			yes		
IRQ	yes			yes		
KBI	8	7	7	8	7	7
SCI1	yes			yes		
SCI2	yes			yes		
SPI1	yes			yes		
SPI2	yes			yes		
TPM1	6-ch	4-ch	4-ch	6-ch	4-ch	4-ch
TPM2	2-ch			2-ch		
USB	yes			yes		
I/O pins	51	37	33	51	37	33
Package types	64 QFP 64 LQFP	48 QFN	44 LQFP	64 QFP 64 LQFP	48 QFN	44 LQFP

1.2 MCU Block Diagram

The block diagram in Figure 1-1 shows the structure of the MC9S08JM60 series MCU.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 1-1. MC9S08JM60 Series Block Diagram

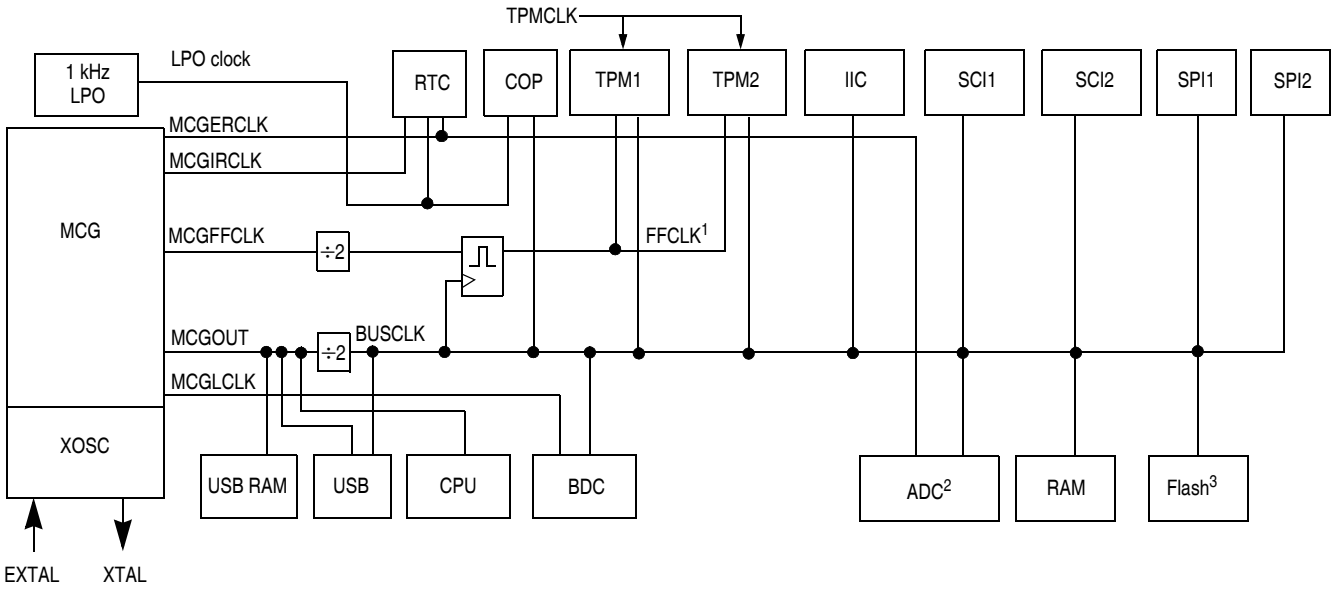
Table 1-2 lists the functional versions of the on-chip modules.

Table 1-2. Versions of On-Chip Modules

Module		Version
Analog Comparator	(ACMP)	2
Analog-to-Digital Converter	(ADC)	1
Central Processing Unit	(CPU)	2
IIC Module	(IIC)	2
Keyboard Interrupt	(KBI)	2
Multi-Purpose Clock Generator	(MCG)	1
Real-Time Counter	(RTC)	1
Serial Communications Interface	(SCI)	4
16-bit Serial Peripheral Interface	(SPI16)	1
Timer Pulse-Width Modulator	(TPM)	3
Universal Serial Bus	(USB)	1
Debug Module	(DBG)	2

1.3 System Clock Distribution

Figure 1-2 shows a simplified clock connection diagram. Some modules in the MCU have selectable clock inputs as shown. The clock inputs to the modules indicate the clock(s) that are used to drive the module function. All memory mapped registers associated with the modules are clocked with BUSCLK.



1. The FFCLK is internally synchronized to the bus clock and must not exceed one half of the bus clock frequency.

2. ADC has min. and max. frequency requirements. See Chapter 10, "Analog-to-Digital Converter (S08ADC12V1)," and Appendix A, "Electrical Characteristics," for details.

3. Flash has the frequency requirements for program and erase operation. See the Appendix A, "Electrical Characteristics," for details.

Figure 1-2. System Clock Distribution Diagram

The MCG supplies the following clock sources:

- MCGOUT — This clock source is used as the CPU, USB RAM and USB module clock, and is divided by two to generate the peripheral bus clock (BUSCLK). Control bits in the MCG control registers determine which of the three clock sources is connected:
  - Internal reference clock
  - External reference clock
  - Frequency-locked loop (FLL) or Phase-locked loop (PLL) output

  See Chapter 12, "Multi-Purpose Clock Generator (S08MCGV1)," for details on configuring the MCGOUT clock.
- MCGLCLK — This clock source is derived from the digitally controlled oscillator (DCO) of the MCG. Development tools can select this internal self-clocked source to speed up BDC communications in systems where the bus clock is slow.
- MCGIRCLK — This is the internal reference clock and can be selected as the real-time counter clock source. Chapter 12, "Multi-Purpose Clock Generator (S08MCGV1)," explains the MCGIRCLK in more detail. See Chapter 13, "Real-Time Counter (S08RTCV1)," for more information regarding the use of MCGIRCLK.
- MCGERCLK — This is the external reference clock and can be selected as the clock source of real-time counter and ADC module. Section 12.4.6, "External Reference Clock," explains the MCGERCLK in more detail. See Chapter 13, "Real-Time Counter (S08RTCV1)," and Chapter 10,

"Analog-to-Digital Converter (S08ADC12V1)," for more information regarding the use of MCGERCLK with these modules.

- MCGFFCLK — This clock source is divided by 2 to generate FFCLK after being synchronized to the BUSCLK. It can be selected as clock source for the TPM modules. The frequency of the MCGFFCLK is determined by the settings of the MCG. See the Section 12.4.7, "Fixed Frequency Clock," for details.
- LPO clock— This clock is generated from an internal Low Power Oscillator that is completely independent of the MCG module. The LPO clock can be selected as the clock source to the RTC or COP modules. See Chapter 13, "Real-Time Counter (S08RTCV1)," and Section 5.4, "Computer Operating Properly (COP) Watchdog," for details on using the LPO clock with these modules.
- TPMCLK — TPMCLK is the optional external clock source for the TPM modules. The TPMCLK must be limited to 1/4th the frequency of the BUSCLK for synchronization. See Chapter 16, "Timer/Pulse-Width Modulator (S08TPMV3)," for more details.

Chapter 2
Pins and Connections

2.1 Introduction

This chapter describes signals that connect to package pins. It includes pinout diagrams, a table of signal properties, and detailed discussion of signals.

2.2 Device Pin Assignment

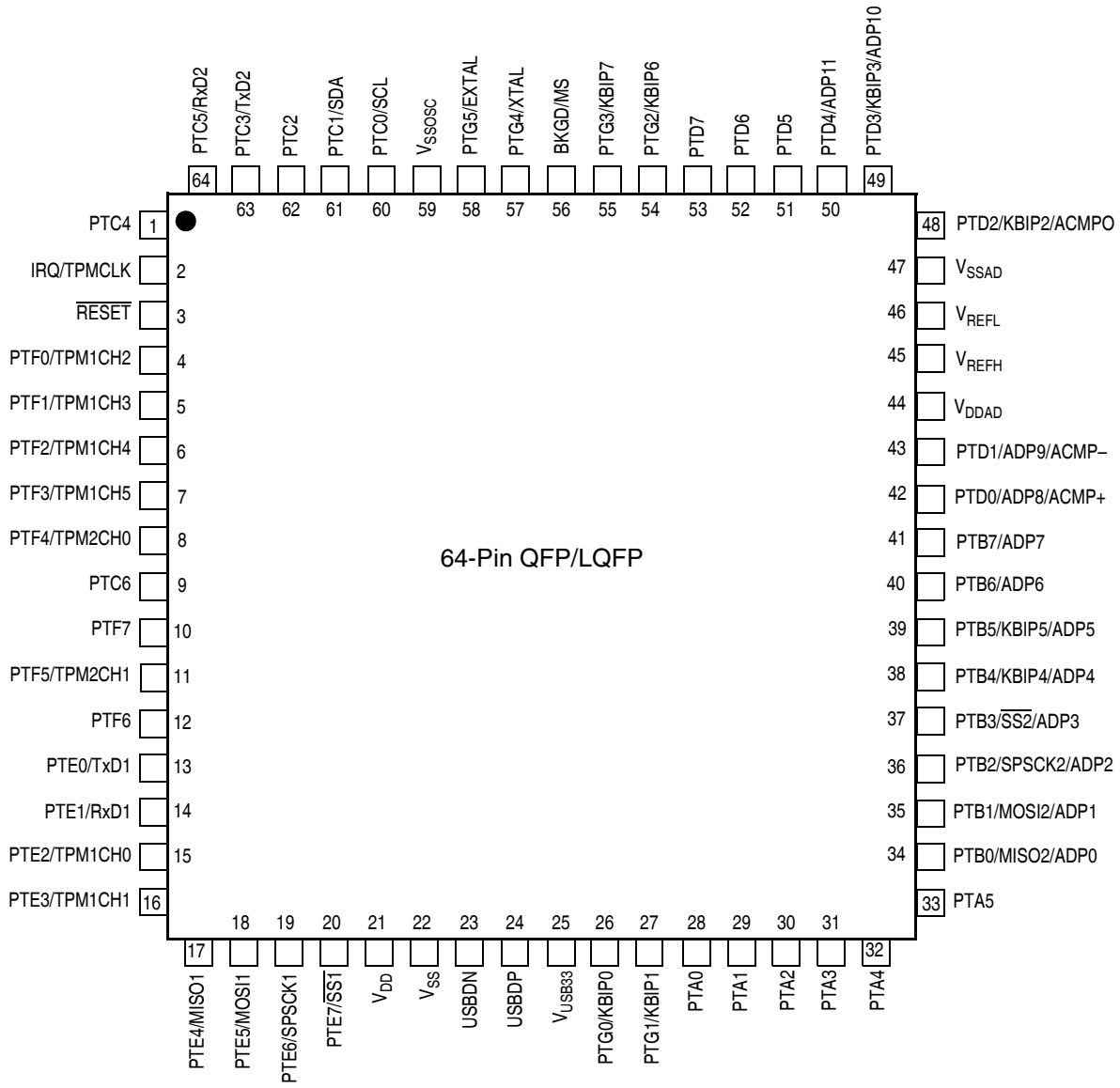


Figure 2-1. MC9S08JM60 in 64-Pin QFP/LQFP Package

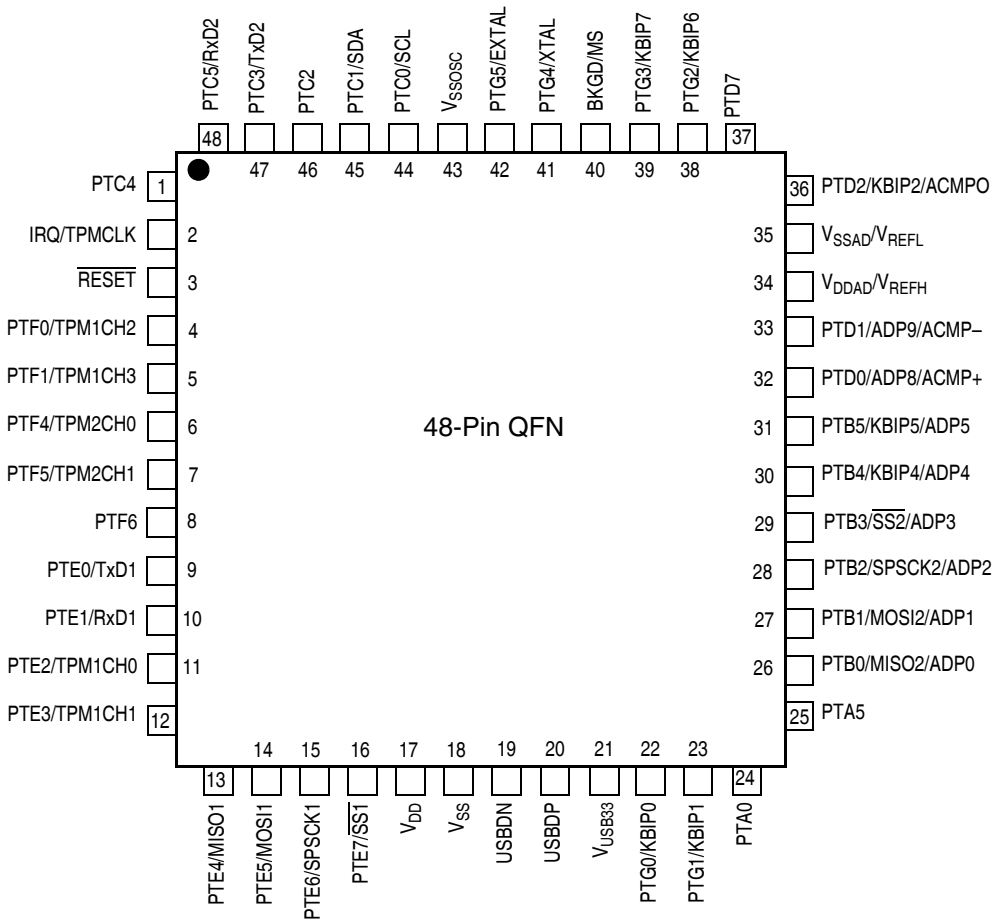


Figure 2-2. MC9S08JM60 Series in 48-Pin QFN Package

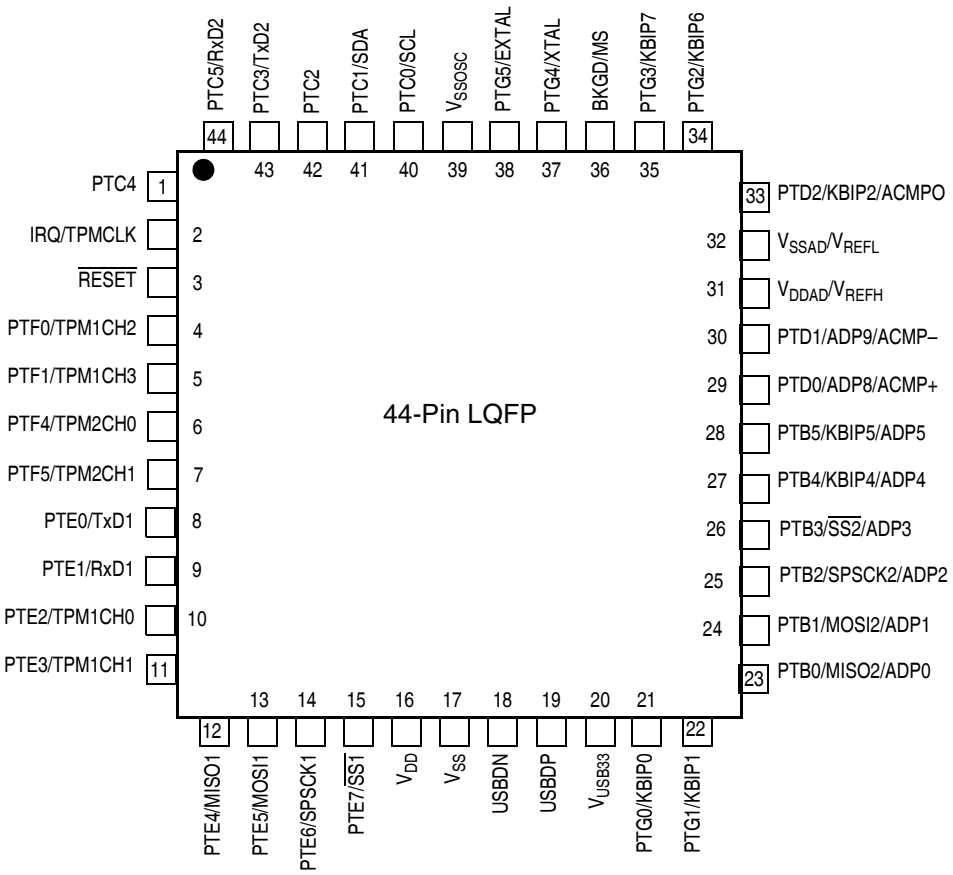
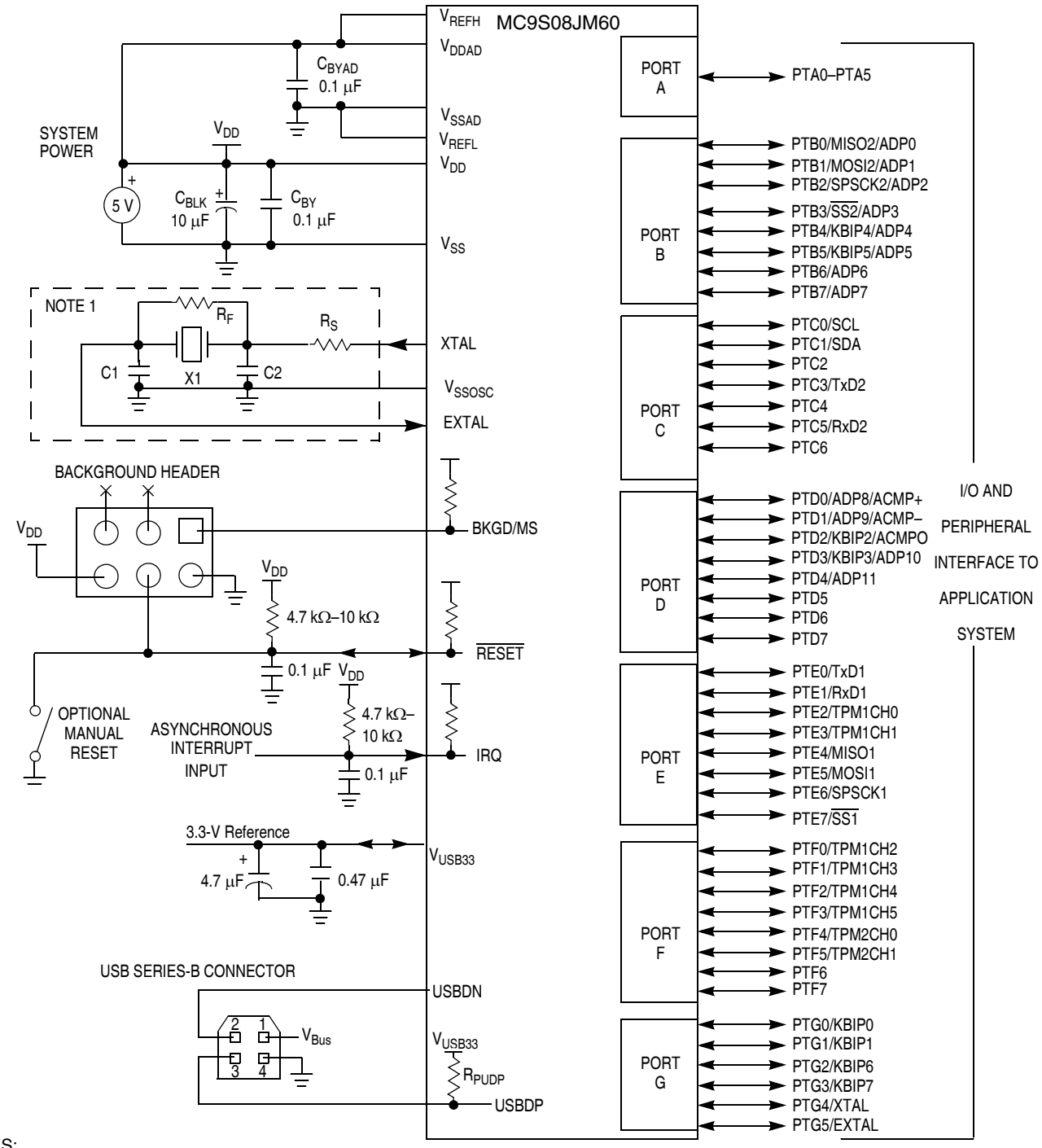


Figure 2-3. MC9S08JM60 Series in 44-Pin LQFP Package

2.3 Recommended System Connections

Figure 2-4 shows pin connections that are common to almost all MC9S08JM60 series application systems.



NOTES:
1. External crystal circuity is not required if using the MCG internal clock option. For USB operation, an external crystal is required.
2. XTAL and EXTAL are the same pins as PTG4 and PTG5, respectively.
3. RC filters on $\overline{\text{RESET}}$ and IRQ are recommended for EMC-sensitive applications.
4. R_{PUDP} is shown for full-speed USB only. The diagram shows a configuration where the on-chip regulator and R_{PUDP} are enabled. The voltage regulator output is used for R_{PUDP}. R_{PUDP} can optionally be disabled if using an external pullup resistor on USBDP
5. V_{BUS} is a 5.0-V supply from upstream port that can be used for USB operation
6. USBDP and USBDN are powered by the 3.3-V regulator or external 3.3-V supply on V_{USB33}.

Figure 2-4. Basic System Connections

2.3.1 Power (V_{DD}, V_{SS}, V_{SSOSC}, V_{DDAD}, V_{SSAD}, V_{USB33})

V_{DD} and V_{SS} are the primary power supply pins for the MCU. This voltage source supplies power to all I/O buffer circuitry and to an internal voltage regulator. The internal voltage regulator provides regulated lower-voltage source to the CPU and other internal circuitry of the MCU.

Typically, application systems have two separate capacitors across the power pins. In this case, there must be a bulk electrolytic capacitor, such as a 10-µF tantalum capacitor, to provide bulk charge storage for the overall system and a 0.1-µF ceramic bypass capacitor located as near to the paired V_{DD} and V_{SS} power pins as practical to suppress high-frequency noise. The MC9S08JM60 series have a V_{SSOSC} pin. This pin must be connected to the system ground plane or to the primary V_{SS} pin through a low-impedance connection.

V_{DDAD} and V_{SSAD} are the analog power supply pins for the MCU. This voltage source supplies power to the ADC module. A 0.1-µF ceramic bypass capacitor must be located as near to the analog power pins as practical to suppress high-frequency noise.

V_{USB33} is connected to the internal USB 3.3-V regulator. V_{USB33} maintains an output voltage of 3.3 V and can only source enough current for internal USB transceiver and USB pullup resistor. Two separate capacitors (4.7-µF bulk electrolytic stability capacitor and 0.47-µF ceramic bypass capacitors) must be connected across this pin to ground to decrease the output ripple of this voltage regulator when it is enabled.

2.3.2 Oscillator (XTAL, EXTAL)

Immediately after reset, the MCU uses an internally generated clock provided by the multi-purpose clock generator (MCG) module. For more information on the MCG, see Chapter 12, “Multi-Purpose Clock Generator (S08MCGV1).”

The oscillator (XOSC) in this MCU is a Pierce oscillator that can accommodate a crystal or ceramic resonator. Rather than a crystal or ceramic resonator, an external oscillator can be connected to the EXTAL input pin.

R_S (when used) and R_F must be low-inductance resistors such as carbon composition resistors. Wire-wound resistors, and some metal film resistors, have too much inductance. C1 and C2 normally must be high-quality ceramic capacitors that are specifically designed for high-frequency applications.

R_F is used to provide a bias path to keep the EXTAL input in its linear range during crystal startup; its value is not generally critical. Typical systems use 1 MΩ to 10 MΩ. Higher values are sensitive to humidity and lower values reduce gain and (in extreme cases) could prevent startup.

C1 and C2 are typically in the 5-pF to 25-pF range and are chosen to match the requirements of a specific crystal or resonator. Be sure to take into account printed circuit board (PCB) capacitance and MCU pin capacitance when selecting C1 and C2. The crystal manufacturer typically specifies a load capacitance which is the series combination of C1 and C2 (which are usually the same size). As a first-order approximation, use 10 pF as an estimate of combined pin and PCB capacitance for each oscillator pin (EXTAL and XTAL).

2.3.3 $\overline{\text{RESET}}$ Pin

$\overline{\text{RESET}}$ is a dedicated pin with a pullup device built in. It has input hysteresis, a high current output driver, and no output slew rate control. Internal power-on reset and low-voltage reset circuitry typically make external reset circuitry unnecessary. This pin is normally connected to the standard 6-pin background debug connector, so a development system can directly reset the MCU system. If desired, a manual external reset can be added by supplying a simple switch to ground (pull reset pin low to force a reset).

Whenever any reset is initiated (whether from an external source or from an internal source, the $\overline{\text{RESET}}$ pin is driven low for approximately 66 bus cycles and released. The reset circuity decodes the cause of reset and records it by setting a corresponding bit in the system control reset status register (SRS).

In EMC-sensitive applications, an external RC filter is recommended on the reset pin. See Figure 2-4 for an example.

2.3.4 Background/Mode Select (BKGD/MS)

When in reset, the BKGD/MS pin functions as a mode select pin. Immediately after reset rises the pin functions as the background pin and can be used for background debug communication. While functioning as a background/mode select pin, the pin includes an internal pullup device, input hysteresis, a standard output driver, and no output slew rate control.

If nothing is connected to this pin, the MCU will enter normal operating mode at the rising edge of reset. If a debug system is connected to the 6-pin standard background debug header, it can hold BKGD/MS low during the rising edge of reset which forces the MCU to active background mode.

The BKGD pin is used primarily for background debug controller (BDC) communications using a custom protocol that uses 16 clock cycles of the target MCU's BDC clock per bit time. The target MCU's BDC clock could be as fast as the bus clock rate, so there must never be any significant capacitance connected to the BKGD/MS pin that could interfere with background serial communications.

Although the BKGD pin is a pseudo open-drain pin, the background debug communication protocol provides brief, actively driven, high speedup pulses to ensure fast rise times. Small capacitances from cables and the absolute value of the internal pullup device play almost no role in determining rise and fall times on the BKGD pin.

2.3.5 ADC Reference Pins (V_{REFH}, V_{REFL})

The V_{REFH} and V_{REFL} pins are the voltage reference high and voltage reference low inputs respectively for the ADC module.

2.3.6 External Interrupt Pin (IRQ)

The IRQ pin is the input source for the IRQ interrupt and is also the input for the BIH and BIL instructions. If the IRQ function is not enabled, this pin can be used for TPMCLK.

In EMC-sensitive applications, an external RC filter is recommended on the IRQ pin. See Figure 2-4 for an example.

2.3.7 USB Data Pins (USBDP, USBDN)

The USBDP (D+) and USBDN (D–) pins are the analog input/output lines to/from full-speed internal USB transciever. An optional internal pullup resistor for the USBDP pin, R_{PUDP}, is available.

2.3.8 General-Purpose I/O and Peripheral Ports

The MC9S08JM60 series of MCUs support up to 51 general-purpose I/O pins, which are shared with on-chip peripheral functions (timers, serial I/O, ADC, keyboard interrupts, etc.).

When a port pin is configured as a general-purpose output or a peripheral uses the port pin as an output, software can select one of two drive strengths and enable or disable slew rate control. When a port pin is configured as a general-purpose input or a peripheral uses the port pin as an input, software can enable a pullup device.

For information about controlling these pins as general-purpose I/O pins, see the Chapter 6, “Parallel Input/Output.” For information about how and when on-chip peripheral systems use these pins, see the appropriate module chapter.

Immediately after reset, all pins are configured as high-impedance general-purpose inputs with internal pullup devices disabled.

Table 2-1. Pin Availability by Package Pin-Count

Pin Number			Lowest <--Priority--> Highest		
64	48	44	Port Pin	Alt1	Alt2
1	1	1	PTC4		
2	2	2		IRQ	TPMCLK
3	3	3			$\overline{\text{RESET}}$
4	4	4	PTF0	TPM1CH2	
5	5	5	PTF1	TPM1CH3	
6	—	—	PTF2	TPM1CH4	
7	—	—	PTF3	TPM1CH5	
8	6	6	PTF4	TPM2CH0	
9	—	—	PTC6		
10	—	—	PTF7		
11	7	7	PTF5	TPM2CH1	
12	8	—	PTF6		
13	9	8	PTE0	TxD1	
14	10	9	PTE1	RxD1	
15	11	10	PTE2	TPM1CH0	
16	12	11	PTE3	TPM1CH1	
17	13	12	PTE4	MISO1	
18	14	13	PTE5	MOSI1	
19	15	14	PTE6	SPSCK1	
20	16	15	PTE7	$\overline{\text{SS1}}$	
21	17	16			V_{DD}
22	18	17			V_{SS}
23	19	18			USBDN
24	20	19			USBDP
25	21	20			V_{USB33}
26	22	21	PTG0	KBIP0	
27	23	22	PTG1	KBIP1	
28	24		PTA0		
29			PTA1		
30			PTA2		
31			PTA3		
32			PTA4		

Pin Number			Lowest <--Priority--> Highest		
64	48	44	Port Pin	Alt1	Alt2
33	25		PTA5		
34	26	23	PTB0	MISO2	ADP0
35	27	24	PTB1	MOSI2	ADP1
36	28	25	PTB2	SPSCK2	ADP2
37	29	26	PTB3	$\overline{\text{SS2}}$	ADP3
38	30	27	PTB4	KBIP4	ADP4
39	31	28	PTB5	KBIP5	ADP5
40	—	—	PTB6	ADP6	
41	—	—	PTB7	ADP7	
42	32	29	PTD0	ADP8	ACMP+
43	33	30	PTD1	ADP9	ACMP–
44	34	31			V_{DDAD}
45					V_{REFH}
46	35	32			V_{REFL}
47					V_{SSAD}
48	36	33	PTD2	KBIP2	ACMPO
49	—	—	PTD3	KBIP3	ADP10
50	—	—	PTD4	ADP11	
51	—	—	PTD5		
52	—	—	PTD6		
53	37	—	PTD7		
54	38	34	PTG2	KBIP6	
55	39	35	PTG3	KBIP7	
56	40	36		BKGD	MS
57	41	37	PTG4	XTAL	
58	42	38	PTG5	EXTAL	
59	43	39			V_{SSOSC}
60	44	40	PTC0	SCL	
61	45	41	PTC1	SDA	
62	46	42	PTC2		
63	47	43	PTC3	TxD2	
64	48	44	PTC5	RxD2	

NOTE

When an alternative function is first enabled, it is possible to get a spurious edge to the module, user software must clear out any associated flags before interrupts are enabled. Table 2-1 illustrates the priority if multiple modules are enabled. The highest priority module will have control over the pin. Selecting a higher priority pin function with a lower priority function already enabled can cause spurious edges to the lower priority module. It is recommended that all modules that share a pin be disabled before enabling another module.

Chapter 3
Modes of Operation

3.1 Introduction

The operating modes of the MC9S08JM60 series are described in this chapter. Entry into each mode, exit from each mode, and functionality while in each mode are described.

3.2 Features

- Active background mode for code development
- Wait mode:
  - CPU halts operation to conserve power
  - System clocks running
  - Full voltage regulation is maintained
- Stop modes: CPU and bus clocks stopped
  - Stop2: Partial power down of internal circuits; RAM and USB RAM contents retained
  - Stop3: All internal circuits powered for fast recovery; RAM, USB RAM, and register contents are retained

3.3 Run Mode

Run is the normal operating mode for the MC9S08JM60 series. This mode is selected upon the MCU exiting reset if the BKGD/MS pin is high. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFE:0xFFFF after reset.

3.4 Active Background Mode

The active background mode functions are managed through the background debug controller (BDC) in the HCS08 core. The BDC, together with the on-chip in-circuit emulator (ICE) debug module (DBG), provides the means for analyzing MCU operation during software development.

Active background mode is entered in any of five ways:

- When the BKGD/MS pin is low at the rising edge of reset
- When a BACKGROUND command is received through the BKGD pin
- When a BGND instruction is executed
- When encountering a BDC breakpoint
- When encountering a DBG breakpoint

After entering active background mode, the CPU is held in a suspended state waiting for serial background commands rather than executing instructions from the user application program.

Background commands are of two types:

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD pin while the MCU is in run mode; non-intrusive commands can also be executed when the MCU is in the active background mode. Non-intrusive commands include:
  - Memory access commands
  - Memory-access-with-status commands
  - BDC register access commands
  - The BACKGROUND command
- Active background commands, which can only be executed while the MCU is in active background mode. Active background commands include commands to:
  - Read or write CPU registers
  - Trace one user program instruction at a time
  - Leave active background mode to return to the user application program (GO)

The active background mode is used to program a bootloader or user application program into the flash program memory before the MCU is operated in run mode for the first time. When the MC9S08JM60 series is shipped from the Freescale factory, the flash program memory is erased by default unless specifically noted, so there is no program that could be executed in run mode until the flash memory is initially programmed. The active background mode can also be used to erase and reprogram the flash memory after it has been previously programmed.

For additional information about the active background mode, refer to Chapter 18, “Development Support.”

3.5 Wait Mode

Wait mode is entered by executing a WAIT instruction. Upon execution of the WAIT instruction, the CPU enters a low-power state in which it is not clocked. The I bit in the condition code register (CCR) is cleared when the CPU enters wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits wait mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

While the MCU is in wait mode, there are some restrictions on which background debug commands can be used.

- Only the BACKGROUND command and memory-access-with-status commands are available while the MCU is in wait mode.
- The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in stop or wait mode.
- The BACKGROUND command can be used to wake the MCU from wait mode and enter active background mode.

3.6 Stop Modes

One of two stop modes is entered upon execution of a STOP instruction when STOPE in SOPT1 is set. In any stop mode, the bus and CPU clocks are halted. The MCG module can be configured to leave the reference clocks running. See Chapter 12, "Multi-Purpose Clock Generator (S08MCGV1)," for more information.

Table 3-1 shows all of the control bits that affect stop mode selection and the mode selected under various conditions. The selected mode is entered following the execution of a STOP instruction.

Table 3-1. Stop Mode Selection

STOPE	ENBDM [1]	LVDE	LVDSE	PPDC	Stop Mode
0	x	x		x	Stop modes disabled; illegal opcode reset if STOP instruction executed
1	1	x		x	Stop3 with BDM enabled [2]
1	0	Both bits must be 1		x	Stop3 with voltage regulator active
1	0	Either bit a 0		0	Stop3
1	0	Either bit a 0		1	Stop2

1 ENBDM is located in the BDCSCR which is only accessible through BDC commands, see Section 18.4.1.1, "BDC Status and Control Register (BDCSCR)."

2 When in stop3 mode with BDM enabled, The S_{IDD} will be near R_{IDD} levels because internal clocks are enabled.

3.6.1 Stop3 Mode

Stop3 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1. The states of all of the internal registers and logic, RAM contents, and I/O pin states are maintained.

Stop3 can be exited by asserting $\overline{\text{RESET}}$, or by an interrupt from one of the following sources: the real-time clock (RTC) interrupt, the USB resume interrupt, LVD, ADC, IRQ, KBI, SCI, or the ACMP.

If stop3 is exited by means of the $\overline{\text{RESET}}$ pin, then the MCU is reset and operation will resume after taking the reset vector. Exit by means of one of the internal interrupt sources results in the MCU taking the appropriate interrupt vector.

3.6.1.1 LVD Enabled in Stop Mode

The LVD system is capable of generating either an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in stop (LVDE and LVDSE bits in SPMSC1 both set) at the time the CPU executes a STOP instruction, then the voltage regulator remains active during stop mode. If the user attempts to enter stop2 with the LVD enabled for stop, the MCU will enter stop3 instead.

For the ADC to operate, the LVD must be left enabled when entering stop3. For the ACMP to operate when ACGBS in ACMPSC is set, the LVD must be left enabled when entering stop3.

For the XOSC to operate with an external reference when RANGE in MCGC2 is set, the LVD must be left enabled when entering stop3.

3.6.1.2 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if ENBDM in BDCSCR is set. This register is described in Chapter 18, "Development Support." If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation. If the user attempts to enter stop2 with ENBDM set, the MCU will enter stop3 instead.

Most background commands are not available in stop mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in stop or wait mode. The BACKGROUND command can be used to wake the MCU from stop and enter active background mode if the ENBDM bit is set. After entering background debug mode, all background commands are available.

3.6.2 Stop2 Mode

Stop2 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1. Most of the internal circuitry of the MCU is powered off in stop2, with the exception of the RAM. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their states during stop2.

Exit from stop2 is performed by asserting either wake-up pin: $\overline{\text{RESET}}$ or IRQ/TPMCLK.

NOTE

IRQ/TPMCLK always functions as an active-low wakeup input when the MCU is in stop2, regardless of how the pin is configured before entering stop2. It must be configured as an input before executing a STOP instruction to avoid an immediate exit from stop2. This pin must be driven or pulled high externally while in stop2 mode.

In addition, the RTC interrupt can wake the MCU from stop2, if enabled.

Upon wake-up from stop2 mode, the MCU starts up as from a power-on reset (POR):

- All module control and status registers are reset
- The LVD reset function is enabled and the MCU remains in the reset state if V_{DD} is below the LVD trip point (low trip point selected due to POR)
- The CPU takes the reset vector

In addition to the above, upon waking up from stop2, the PPDF bit in SPMSC2 is set. This flag is used to direct user code to go to a stop2 recovery routine. PPDF remains set and the I/O pin states remain latched until a 1 is written to PPDACK in SPMSC2.

To maintain I/O states for pins that were configured as general-purpose I/O before entering stop2, the user must restore the contents of the I/O port registers, which have been saved in RAM, to the port registers before writing to the PPDACK bit. If the port registers are not restored from RAM before writing to PPDACK, then the pins will switch to their reset states when PPDACK is written.

For pins that were configured as peripheral I/O, the user must reconfigure the peripheral module that interfaces to the pin before writing to the PPDACK bit. If the peripheral module is not enabled before

writing to PPDACK, the pins will be controlled by their associated port control registers when the I/O latches are opened.

3.6.3 On-Chip Peripheral Modules in Stop Modes

When the MCU enters any stop mode, system clocks to the internal peripheral modules are stopped. Even in the exception case (ENBDM = 1), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. Refer to Section 3.6.2, "Stop2 Mode," and Section 3.6.1, "Stop3 Mode," for specific information on system behavior in stop modes.

Table 3-2. Stop Mode Behavior

Peripheral	Mode	
	Stop2	Stop3
CPU	Off	Standby
RAM	Standby	Standby
Flash	Off	Standby
Parallel Port Registers	Off	Standby
ADC	Off	Optionally On[1]
ACMP	Off	Optionally On[2]
MCG	Off	Optionally On[3]
IIC	Off	Standby
RTC	Optionally on[4]	Optionally on[4]
SCI	Off	Standby
SPI	Off	Standby
TPM	Off	Standby
System Voltage Regulator	Off	Standby
XOSC	Off	Optionally On[5]
I/O Pins	States Held	States Held
USB (SIE and Transceiver)	Off	Optionally On[6]
USB 3.3-V Regulator	Off	Standby
USB RAM	Standby	Standby

1 Requires the asynchronous ADC clock and LVD to be enabled, else in standby.

2 If ACGBS in ACMPSC is set, LVD must be enabled, else in standby.

3 IRCLKEN and IREFSTEN set in MCGC1, else in standby.

4 RTCPS[3:0] in RTCSC does not equal 0 before entering stop, else off.

5 ERCLKEN and EREFSTEN set in MCGC2, else in standby. For high frequency range (RANGE in MCGC2 set) requires the LVD to also be enabled in stop3.

6 USBEN in CTL is set and USBPHYEN in USBCTL0 is set, else off.

Chapter 4
Memory

4.1 MC9S08JM60 Series Memory Map

Figure 4-1 shows the memory map for the MC9S08JM60 series. On-chip memory in the MC9S08JM60 series of MCUs consists of RAM, flash program memory for nonvolatile data storage, plus I/O and control/status registers. The registers are divided into three groups:

- Direct-page registers (0x0000 through 0x00AF)
- High-page registers (0x1800 through 0x185F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

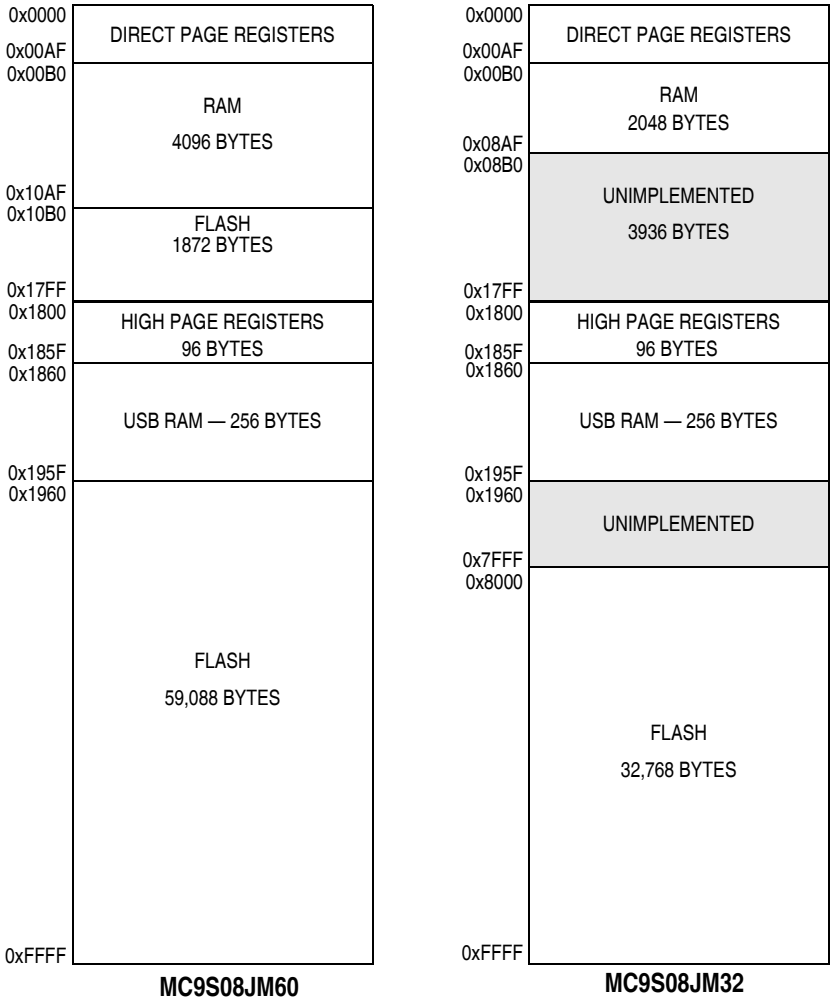


Figure 4-1. MC9S08JM60 Series Memory Map

4.1.1 Reset and Interrupt Vector Assignments

Figure 4-1 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the Freescale-provided equate file for the MC9S08JM60 series. For more details about resets, interrupts, interrupt priority, and local interrupt mask controls, refer to Chapter 5, "Resets, Interrupts, and System Configuration."

Table 4-1. Reset and Interrupt Vectors

Address (High/Low)	Vector	Vector Name
0xFFC0:0xFFC1 to 0xFFC2:FFC3	Unused Vector Space	
0xFFC4:FFC5	RTC	Vrtc
0xFFC6:FFC7	IIC	Viic
0xFFC8:FFC9	ACMP	Vacmp

Table 4-1. Reset and Interrupt Vectors (continued)

Address (High/Low)	Vector	Vector Name
0xFFCA:FFCB	ADC Conversion	Vadc
0xFFCC:FFCD	KBI	Vkeyboard
0xFFCE:FFCF	SCI2 Transmit	Vsci2tx
0xFFD0:FFD1	SCI2 Receive	Vsci2rx
0xFFD2:FFD3	SCI2 Error	Vsci2err
0xFFD4:FFD5	SCI1 Transmit	Vsci1tx
0xFFD6:FFD7	SCI1 Receive	Vsci1rx
0xFFD8:FFD9	SCI1 Error	Vsci1err
0xFFDA:FFDB	TPM2 Overflow	Vtpm2ovf
0xFFDC:FFDD	TPM2 Channel 1	Vtpm2ch1
0xFFDE:FFDF	TPM2 Channel 0	Vtpm2ch0
0xFFE0:FFE1	TPM1 Overflow	Vtpm1ovf
0xFFE2:FFE3	TPM1 Channel 5	Vtpm1ch5
0xFFE4:FFE5	TPM1 Channel 4	Vtpm1ch4
0xFFE6:FFE7	TPM1 Channel 3	Vtpm1ch3
0xFFE8:FFE9	TPM1 Channel 2	Vtpm1ch2
0xFFEA:FFEB	TPM1 Channel 1	Vtpm1ch1
0xFFEC:FFED	TPM1 Channel 0	Vtpm1ch0
0xFFEE:FFEF	Reserved	—
0xFFF0:FFF1	USB Status	Vusb
0xFFF2:FFF3	SPI2	Vspi2
0xFFF4:FFF5	SPI1	Vspi1
0xFFF6:FFF7	MCG Loss of Lock	Vlol
0xFFF8:FFF9	Low Voltage Detect	Vlvd
0xFFFA:FFFB	IRQ	Virq
0xFFFC:FFFD	SWI	Vswi
0xFFFE:FFFF	Reset	Vreset

4.2 Register Addresses and Bit Assignments

The registers in the MC9S08JM60 series are divided into these three groups:

- Direct-page registers are located in the first 176 locations in the memory map, so they are accessible with efficient direct addressing mode instructions.
- High-page registers are used much less often, so they are located above 0x1800 in the memory map. This leaves more room in the direct page for more frequently used registers and variables.
- The nonvolatile register area consists of a block of 16 locations in flash memory at 0xFFB0–0xFFBF.

  Nonvolatile register locations include:

  — Three values which are loaded into working registers at reset

— An 8-byte backdoor comparison key which optionally allows a user to gain controlled access to secure memory

Because the nonvolatile register locations are flash memory, they must be erased and programmed like other flash memory locations.

Direct-page registers can be accessed with efficient direct addressing mode instructions. Bit manipulation instructions can be used to access any bit in any direct-page register. Table 4-2 is a summary of all user-accessible direct-page registers and control bits.

The direct page registers in Table 4-2 can use the more efficient direct addressing mode which only requires the lower byte of the address. Because of this, the lower byte of the address in column one is shown in bold text. In Table 4-3 and Table 4-4, the whole address in column one is shown in bold. In Table 4-2, Table 4-3, and Table 4-4, the register names in column two are shown in bold to set them apart from the bit names to the right. Cells that are not associated with named bits are shaded. A shaded cell with a 0 indicates this unused bit always reads as a 0. Shaded cells with dashes indicate unused or reserved bit locations that could read as 1s or 0s.

Table 4-2. Direct-Page Register Summary (Sheet 1 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000	**PTAD**	—	—	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	**PTADD**	—	—	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
0x0002	**PTBD**	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x0003	**PTBDD**	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x0004	**PTCD**	—	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
0x0005	**PTCDD**	—	PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
0x0006	**PTDD**	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
0x0007	**PTDDD**	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
0x0008	**PTED**	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
0x0009	**PTEDD**	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
0x000A	**PTFD**	PTFD7	PTFD6	PTFD5	PTFD4	PTFD3	PTFD2	PTFD1	PTFD0
0x000B	**PTFDD**	PTFDD7	PTFDD6	PTFDD5	PTFDD4	PTFDD3	PTFDD2	PTFDD1	PTFDD0
0x000C	**PTGD**	—	—	PTGD5	PTGD4	PTGD3	PTGD2	PTGD1	PTGD0
0x000D	**PTGDD**	—	—	PTGDD5	PTGDD4	PTGDD3	PTGDD2	PTGDD1	PTGDD0
0x000E	**ACMPSC**	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD	
0x000F	Reserved	—	—	—	—	—	—	—	—
0x0010	**ADCSC1**	COCO	AIEN	ADCO	ADCH				
0x0011	**ADCSC2**	ADACT	ADTRG	ACFE	ACFGT	0	0	R	R
0x0012	**ADCRH**	0	0	0	0	ADR11	ADR10	ADR9	ADR8
0x0013	**ADCRL**	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
0x0014	**ADCCVH**	0	0	0	0	ADCV11	ADCV10	ADCV9	ADCV8
0x0015	**ADCCVL**	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
0x0016	**ADCCFG**	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x0017	**APCTL1**	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x0018	**APCTL2**	—	—	—	—	ADPC11	ADPC10	ADPC9	ADPC8
0x0019–0x001A	Reserved	—	—	—	—	—	—	—	—
0x001B	**IRQSC**	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x001C	**KBISC**	0	0	0	0	KBF	KBACK	KBIE	KBMOD
0x001D	**KBIPE**	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x001E	**KBIES**	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
0x001F	Reserved	—	—	—	—	—	—	—	—
0x0020	**TPM1SC**	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0021	**TPM1CNTH**	Bit 15	14	13	12	11	10	9	Bit 8
0x0022	**TPM1CNTL**	Bit 7	6	5	4	3	2	1	Bit 0
0x0023	**TPM1MODH**	Bit 15	14	13	12	11	10	9	Bit 8
0x0024	**TPM1MODL**	Bit 7	6	5	4	3	2	1	Bit 0
0x0025	**TPM1C0SC**	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0026	**TPM1C0VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x0027	**TPM1C0VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x0028	**TPM1C1SC**	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0029	**TPM1C1VH**	Bit 15	14	13	12	11	10	9	Bit 8

Table 4-2. Direct-Page Register Summary (Sheet 2 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x002A	TPM1C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x002B	TPM1C2SC	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x002C	TPM1C2VH	Bit 15	14	13	12	11	10	9	Bit 8
0x002D	TPM1C2VL	Bit 7	6	5	4	3	2	1	Bit 0
0x002E	TPM1C3SC	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	0	0
0x002F	TPM1C3VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0030	TPM1C3VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0031	TPM1C4SC	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	0	0
0x0032	TPM1C4VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0033	TPM1C4VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0034	TPM1C5SC	CH5F	CH5IE	MS5B	MS5A	ELS5B	ELS5A	0	0
0x0035	TPM1C5VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0036	TPM1C5VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0037	Reserved	—	—	—	—	—	—	—	—
0x0038	SCI1BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0039	SCI1BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x003A	SCI1C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x003B	SCI1C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x003C	SCI1S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x003D	SCI1S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x003E	SCI1C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x003F	SCI1D	Bit 7	6	5	4	3	2	1	Bit 0
0x0040	SCI2BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0041	SCI2BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x0042	SCI2C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x0043	SCI2C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x0044	SCI2S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x0045	SCI2S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x0046	SCI2C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x0047	SCI2D	Bit 7	6	5	4	3	2	1	Bit 0
0x0048	MCGC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
0x0049	MCGC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
0x004A	MCGTRM	TRIM							
0x004B	MCGSC	LOLS	LOCK	PLLST	IREFST	CLKST		OSCINIT	FTRIM
0x004C	MCGC3	LOLIE	PLLS	CME	0	VDIV			
0x004D	MCGT	0	0	0	0	0	0	0	0
0x004E–0x004F	Reserved	—	—	—	—	—	—	—	—
0x0050	SPI1C1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0051	SPI1C2	SPMIE	SPIMODE	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x0052	SPI1BR	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
0x0053	SPI1S	SPRF	SPMF	SPTEF	MODF	0	0	0	0

Table 4-2. Direct-Page Register Summary (Sheet 3 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0054	**SPI1DH**	Bit 15	14	13	12	11	10	9	Bit 8
0x0055	**SPI1DL**	Bit 7	6	5	4	3	2	1	Bit 0
0x0056	**SPI1MH**	Bit 15	14	13	12	11	10	9	Bit 8
0x0057	**SPI1ML**	Bit 7	6	5	4	3	2	1	Bit 0
0x0058	**IICA**	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
0x0059	**IICF**	MULT		ICR					
0x005A	**IICC1**	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
0x005B	**IICS**	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
0x005C	**IICD**	DATA							
0x005D	**IICC2**	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
0x005E–0x005F	Reserved	—	—	—	—	—	—	—	—
0x0060	**TPM2SC**	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0061	**TPM2CNTH**	Bit 15	14	13	12	11	10	9	Bit 8
0x0062	**TPM2CNTL**	Bit 7	6	5	4	3	2	1	Bit 0
0x0063	**TPM2MODH**	Bit 15	14	13	12	11	10	9	Bit 8
0x0064	**TPM2MODL**	Bit 7	6	5	4	3	2	1	Bit 0
0x0065	**TPM2C0SC**	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0066	**TPM2C0VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x0067	**TPM2C0VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x0068	**TPM2C1SC**	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0069	**TPM2C1VH**	Bit 15	14	13	12	11	10	9	Bit 8
0x006A	**TPM2C1VL**	Bit 7	6	5	4	3	2	1	Bit 0
0x006B	Reserved	—	—	—	—	—	—	—	—
0x006C	**RTCSC**	RTIF	RTCLKS		RTIE	RTCPS			
0x006D	**RTCCNT**	RTCCNT							
0x006E	**RTCMOD**	RTCMOD							
0x006F	Reserved	—	—	—	—	—	—	—	—
0x0070	**SPI2C1**	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0071	**SPI2C2**	SPMIE	SPIMODE	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x0072	**SPI2BR**	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
0x0073	**SPI2S**	SPRF	SPMF	SPTEF	MODF	0	0	0	0
0x0074	**SPI2DH**	Bit 15	14	13	12	11	10	9	Bit 8
0x0075	**SPI2DL**	Bit 7	6	5	4	3	2	1	Bit 0
0x0076	**SPI2MH**	Bit 15	14	13	12	11	10	9	Bit 8
0x0077	**SPI2ML**	Bit 7	6	5	4	3	2	1	Bit 0
0x0078–0x007F	Reserved	—	—	—	—	—	—	—	—
0x0080	**USBCTL0**	USBRESET	USBPU	USBRESMEN	LPRESF	—	USBVREN	—	USBPHYEN
0x0081–0x0087	Reserved	—	—	—	—	—	—	—	—
0x0088	**PERID**	0	0	ID5	ID4	ID3	ID2	ID1	ID0

Table 4-2. Direct-Page Register Summary (Sheet 4 of 4)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0089	**IDCOMP**	1	1	NID5	NID4	NID3	NID2	NID1	NID0
0x008A	**REV**	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
0x008B–0x008F	Reserved	—	—	—	—	—	—	—	—
0x0090	**INTSTAT**	STALLF	—	RESUMEF	SLEEPF	TOKDNEF	SOFTOKF	ERRORF	USBRSTF
0x0091	**INTENB**	STALL	—	RESUME	SLEEP	TOKDNE	SOFTOK	ERROR	USBRST
0x0092	**ERRSTAT**	BTSERRF	—	BUFERRF	BTOERRF	DFN8F	CRC16F	CRC5F	PIDERRF
0x0093	**ERRENB**	BTSERR	—	BUFERR	BTOERR	DFN8	CRC16	CRC5	PIDERR
0x0094	**STAT**	ENDP				IN	ODD	0	0
0x0095	**CTL**	—	—	TSUSPEND	—	—	CRESUME	ODDRST	USBEN
0x0096	**ADDR**	—	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
0x0097	**FRMNUML**	FRM7	FRM6	FRM5	FRM4	FRM3	FRM2	FRM1	FRM0
0x0098	**FRMNUMH**	0	0	0	0	0	FRM10	FRM9	FRM8
0x0099–0x009C	Reserved	—	—	—	—	—	—	—	—
0x009D	**EPCTL0**	—	—	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x009E	**EPCTL1**	—	—	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x009F	**EPCTL2**	—	—	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x00A0	**EPCTL3**	—	—	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x00A1	**EPCTL4**	—	—	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x00A2	**EPCTL5**	—	—	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x00A3	**EPCTL6**	—	—	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x00A4–0x00AF	Reserved	—	—	—	—	—	—	—	—

High-page registers, shown in Table 4-3, are accessed much less often than other I/O and control registers so they have been located outside the direct addressable memory space, starting at 0x1800.

Table 4-3. High-Page Register Summary (Sheet 1 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	**SRS**	POR	PIN	COP	ILOP	0	LOC	LVD	—
0x1801	**SBDFR**	0	0	0	0	0	0	0	BDFR
0x1802	**SOPT1**	COPT		STOPE	—	0	0	—	—
0x1803	**SOPT2**	COPCLKS	COPW	0	0	0	SPI1FE	SPI2FE	ACIC
0x1804–0x1805	Reserved	—	—	—	—	—	—	—	—
0x1806	**SDIDH**	—	—	—	—	ID11	ID10	ID9	ID8
0x1807	**SDIDL**	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	Reserved	—	—	—	—	—	—	—	—
0x1809	**SPMSC1**	LVWF	LVWACK	LVWIE	LVDRE	LVDSE	LVDE	0[1]	BGBE
0x180A	**SPMSC2**	—	—	LVDV	LVWV	PPDF	PPDACK	—	PPDC
0x180B–0x180F	Reserved	—	—	—	—	—	—	—	—

Table 4-3. High-Page Register Summary (Sheet 2 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1810	**DBGCAH**	Bit 15	14	13	12	11	10	9	Bit 8
0x1811	**DBGCAL**	Bit 7	6	5	4	3	2	1	Bit 0
0x1812	**DBGCBH**	Bit 15	14	13	12	11	10	9	Bit 8
0x1813	**DBGCBL**	Bit 7	6	5	4	3	2	1	Bit 0
0x1814	**DBGFH**	Bit 15	14	13	12	11	10	9	Bit 8
0x1815	**DBGFL**	Bit 7	6	5	4	3	2	1	Bit 0
0x1816	**DBGC**	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
0x1817	**DBGT**	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
0x1818	**DBGS**	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
0x1819–0x181F	Reserved	—	—	—	—	—	—	—	—
0x1820	**FCDIV**	DIVLD	PRDIV8	DIV5	DIV4	DIV3	DIV2	DIV1	DIV0
0x1821	**FOPT**	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
0x1822	Reserved	—	—	—	—	—	—	—	—
0x1823	**FCNFG**	0	0	KEYACC	0	0	0	0	0
0x1824	**FPROT**	FPS7	FPS6	FPS5	FPS4	FPS3	FPS2	FPS1	FPDIS
0x1825	**FSTAT**	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x1826	**FCMD**	FCMD7	FCMD6	FCMD5	FCMD4	FCMD3	FCMD2	FCMD1	FCMD0
0x1827–0x183F	Reserved	—	—	—	—	—	—	—	—
0x1840	**PTAPE**	—	—	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x1841	**PTASE**	—	—	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x1842	**PTADS**	—	—	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x1843	Reserved	—	—	—	—	—	—	—	—
0x1844	**PTBPE**	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x1845	**PTBSE**	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x1846	**PTBDS**	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x1847	Reserved	—	—	—	—	—	—	—	—
0x1848	**PTCPE**	—	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x1849	**PTCSE**	—	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x184A	**PTCDS**	—	PTCDS6	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x184B	Reserved	—	—	—	—	—	—	—	—
0x184C	**PTDPE**	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
0x184D	**PTDSE**	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
0x184E	**PTDDS**	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
0x184F	Reserved	—	—	—	—	—	—	—	—
0x1850	**PTEPE**	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
0x1851	**PTESE**	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
0x1852	**PTEDS**	PTEDS7	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
0x1853	Reserved	—	—	—	—	—	—	—	—
0x1854	**PTFPE**	PTFPE7	PTFPE6	PTFPE5	PTFPE4	PTFPE3	PTFPE2	PTFPE1	PTFPE0
0x1855	**PTFSE**	PTFSE7	PTFSE6	PTFSE5	PTFSE4	PTFSE3	PTFSE2	PTFSE1	PTFSE0
0x1856	**PTFDS**	PTFDS7	PTFDS6	PTFDS5	PTFDS4	PTFDS3	PTFDS2	PTFDS1	PTFDS0

Table 4-3. High-Page Register Summary (Sheet 3 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1857	Reserved	—	—	—	—	—	—	—	—
0x1858	**PTGPE**	—	—	PTGPE5	PTGPE4	PTGPE3	PTGPE2	PTGPE1	PTGPE0
0x1859	**PTGSE**	—	—	PTGSE5	PTGSE4	PTGSE3	PTGSE2	PTGSE1	PTGSE0
0x185A	**PTGDS**	—	—	PTGDS5	PTGDS4	PTGDS3	PTGDS2	PTGDS1	PTGDS0
0x185B–0x185F	Reserved	—	—	—	—	—	—	—	—

[1] This reserved bit must always be written to 0.

Nonvolatile flash registers, shown in Table 4-4, are located in the flash memory. These registers include an 8-byte backdoor key which optionally can be used to gain access to secure memory resources. During reset events, the contents of NVPROT and NVOPT in the nonvolatile register area of the flash memory are transferred into corresponding FPROT and FOPT working registers in the high-page registers to control security and block protection options.

Table 4-4. Nonvolatile Register Summary

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0xFFAE	Reserved for storage of FTRIM	0	0	0	0	0	0	0	FTRIM
0xFFAF	Res. for storage of MCGTRIM	TRIM							
0xFFB0–0xFFB7	NVBACKKEY	8-Byte Comparison Key							
0xFFB8–0xFFBC	Reserved	—	—	—	—	—	—	—	—
0xFFBD	**NVPROT**	FPS7	FPS6	FPS5	FPS4	FPS3	FPS2	FPS1	FPDIS
0xFFBE	Reserved	—	—	—	—	—	—	—	—
0xFFBF	**NVOPT**	KEYEN	FNORED	0	0	0	0	SEC01	SEC00

Provided the key enable (KEYEN) bit is 1, the 8-byte comparison key can be used to temporarily disengage memory security. This key mechanism can be accessed only through user code running in secure memory. (A security key cannot be entered directly through background debug commands.) This security key can be disabled completely by programming the KEYEN bit to 0. If the security key is disabled, the only way to disengage security is by mass erasing the flash if needed (normally through the background debug interface) and verifying that flash is blank. To avoid returning to secure mode after the next reset, program the security bits (SEC01:SEC00) to the unsecured state (1:0).

4.3 RAM (System RAM)

The MC9S08JM60 series includes static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

The RAM retains data when the MCU is in low-power wait, stop2, or stop3 mode. At power-on, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention.

For compatibility with older M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the MC9S08JM60 series, it is usually best to re-initialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale-provided equate file).

```
LDHX    #RamLast+1    ;point one past RAM
TXS                   ;SP<-(H:X-1)
```

When security is enabled, the RAM is considered a secure memory resource and is not accessible through BDM or through code executing from non-secure memory. See Section 4.6, “Security,” for a detailed description of the security feature.

4.4 USB RAM

USB RAM is discussed in detail in Chapter 17, “Universal Serial Bus Device Controller (S08USBV1).”

4.5 Flash

The flash memory is used for program storage. In-circuit programming allows the operating program to be loaded into the flash memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because no special voltages are needed for flash erase and programming operations, in-application programming is also possible through other software-controlled communication paths. For a more detailed discussion of in-circuit and in-application programming, refer to the *HCS08 Family Reference Manual, Volume I,* Freescale Semiconductor document order number HCS08RMv1.

4.5.1 Features

Features of the flash memory include:

- Flash size
  - MC9S08JM60 — 60,912 bytes (119 pages of 512 bytes each)
  - MC9S08JM32 — 32,768 bytes (64 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for flash and RAM
- Auto power-down for low-frequency read accesses

4.5.2 Program and Erase Times

Before any program or erase command can be accepted, the flash clock divider register (FCDIV) must be written to set the internal clock for the flash module to a frequency (f_{FCLK}) between 150 kHz and 200 kHz (see Section 4.7.1, "Flash Clock Divider Register (FCDIV)."). This register can be written only once, so normally this write is done during reset initialization. FCDIV cannot be written if the access error flag, FACCERR in FSTAT, is set. The user must ensure that FACCERR is not set before writing to the FCDIV register. One period of the resulting clock ($1/f_{FCLK}$) is used by the command processor to time program and erase pulses. An integer number of these timing pulses are used by the command processor to complete a program or erase command.

Table 4-5 shows program and erase times. The bus clock frequency and FCDIV determine the frequency of FCLK (f_{FCLK}). The time for one cycle of FCLK is $t_{FCLK} = 1/f_{FCLK}$. The times are shown as a number of cycles of FCLK and as an absolute time for the case where $t_{FCLK} = 5$ µs. Program and erase times shown include overhead for the command state machine and enabling and disabling of program and erase voltages.

Table 4-5. Program and Erase Times

Parameter	Cycles of FCLK	Time if FCLK = 200 kHz
Byte program	9	45 µs
Byte program (burst)	4	20 µs[1]
Page erase	4000	20 ms
Mass erase	20,000	100 ms

[1] Excluding start/end overhead

4.5.3 Program and Erase Command Execution

The steps for executing any of the commands are listed below. The FCDIV register must be initialized and any error flags cleared before beginning command execution. The command execution steps are:

1. Write a data value to an address in the flash array. The address and data information from this write is latched into the flash interface. This write is a required first step in any command sequence. For erase and blank check commands, the value of the data is not important. For page erase commands, the address may be any address in the 512-byte page of flash to be erased. For mass erase and blank check commands, the address can be any address in the flash memory. Whole pages of 512 bytes are the smallest block of flash that may be erased. In the 60K version, there are two instances where the size of a block that is accessible to the user is less than 512 bytes: the first page following RAM, and the first page following the high page registers. These pages are overlapped by the RAM and high page registers respectively.

NOTE

Do not program any byte in the flash more than once after a successful erase operation. Reprogramming bits to a byte which is already programmed is not allowed without first erasing the page in which the byte resides or mass erasing the entire flash memory. Programming without first erasing may disturb data stored in the flash.

2. Write the command code for the desired command to FCMD. The five valid commands are blank check (0x05), byte program (0x20), burst program (0x25), page erase (0x40), and mass erase (0x41). The command code is latched into the command buffer.
3. Write a 1 to the FCBEF bit in FSTAT to clear FCBEF and launch the command (including its address and data information).

A partial command sequence can be aborted manually by writing a 0 to FCBEF any time after the write to the memory array and before writing the 1 that clears FCBEF and launches the complete command. Aborting a command in this way sets the FACCERR access error flag which must be cleared before starting a new command.

A strictly monitored procedure must be obeyed or the command will not be accepted. This minimizes the possibility of any unintended changes to the flash memory contents. The command complete flag (FCCF) indicates when a command is complete. The command sequence must be completed by clearing FCBEF to launch the command. Figure 4-2 is a flowchart for executing all of the commands except for burst programming. The FCDIV register must be initialized before using any flash commands. This only must be done once following a reset.

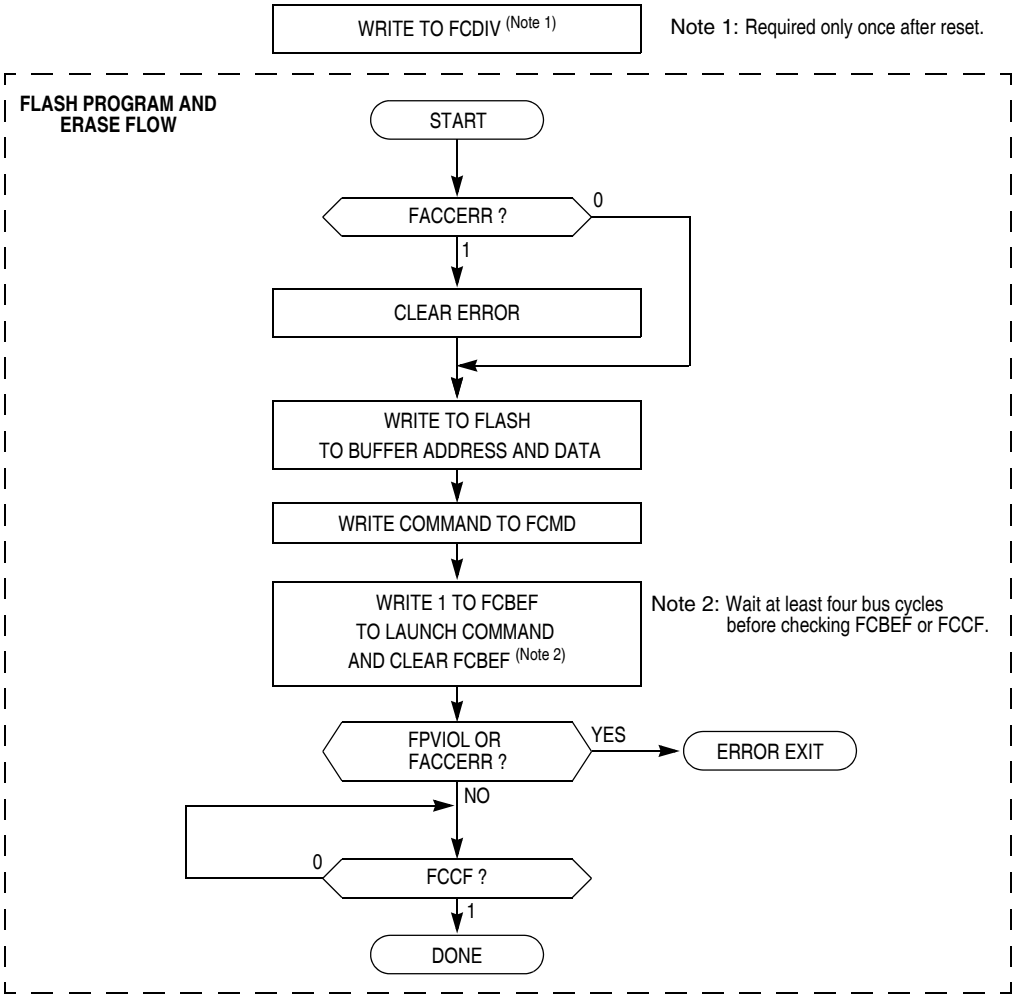


Figure 4-2. Flash Program and Erase Flowchart

4.5.4 Burst Program Execution

The burst program command is used to program sequential bytes of data in less time than would be required using the standard program command. This is possible because the high voltage to the flash array does not need to be disabled between program operations. Ordinarily, when a program or erase command is issued, an internal charge pump associated with the flash memory must be enabled to supply high voltage to the array. Upon completion of the command, the charge pump is turned off. When a burst program command is issued, the charge pump is enabled and then remains enabled after completion of the burst program operation if these two conditions are met:

- The next burst program command has been queued before the current program operation has completed.
- The next sequential address selects a byte on the same physical row as the current byte being programmed. A row of flash memory consists of 64 bytes. A byte within a row is selected by addresses A5 through A0. A new row begins when addresses A5 through A0 are all zero.

The first byte of a series of sequential bytes being programmed in burst mode will take the same amount of time to program as a byte programmed in standard mode. Subsequent bytes will program in the burst program time provided that the conditions above are met. In the case the next sequential address is the beginning of a new row, the program time for that byte will be the standard time instead of the burst time. This is because the high voltage to the array must be disabled and then enabled again. If a new burst command has not been queued before the current command completes, then the charge pump will be disabled and high voltage removed from the array.

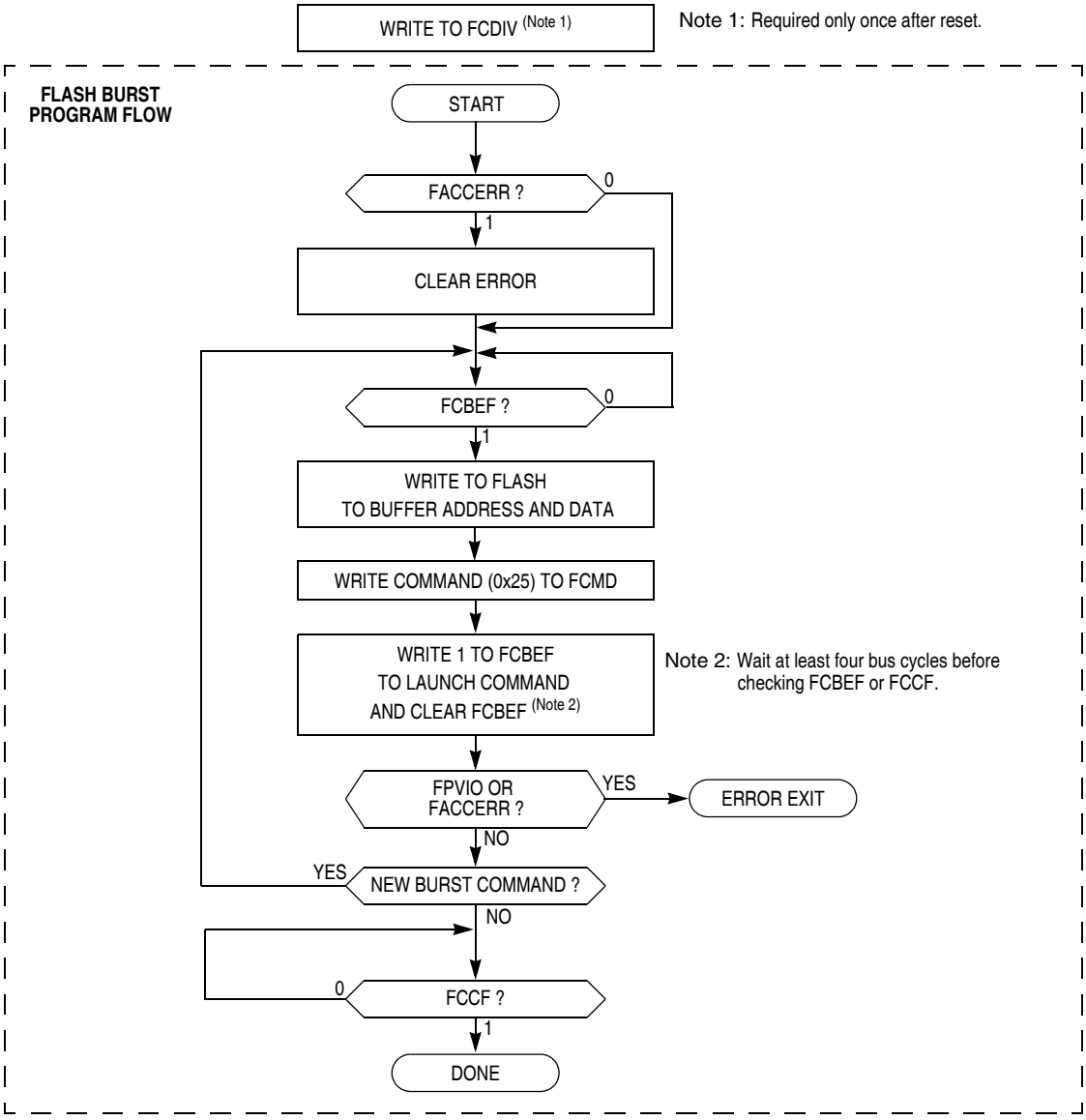


Figure 4-3. Flash Burst Program Flowchart

4.5.5 Access Errors

An access error occurs whenever the command execution protocol is violated.

Any of the following specific actions will cause the access error flag (FACCERR) in FSTAT to be set. FACCERR must be cleared by writing a 1 to FACCERR in FSTAT before any command can be processed.

- Writing to a flash address before the internal flash clock frequency has been set by writing to the FCDIV register
- Writing to a flash address while FCBEF is not set (A new command cannot be started until the command buffer is empty.)
- Writing a second time to a flash address before launching the previous command (There is only one write to flash for every command.)

- Writing a second time to FCMD before launching the previous command (There is only one write to FCMD for every command.)
- Writing to any flash control register other than FCMD after writing to a flash address
- Writing any command code other than the five allowed codes (0x05, 0x20, 0x25, 0x40, or 0x41) to FCMD
- Writing any flash control register other than the write to FSTAT (to clear FCBEF and launch the command) after writing the command to FCMD.
- The MCU enters stop mode while a program or erase command is in progress (The command is aborted.)
- Writing the byte program, burst program, or page erase command code (0x20, 0x25, or 0x40) with a background debug command while the MCU is secured (The background debug controller can only do blank check and mass erase commands when the MCU is secure.)
- Writing 0 to FCBEF to cancel a partial command

4.5.6 Flash Block Protection

The block protection feature prevents the protected region of flash from program or erase changes. Block protection is controlled through the flash protection register (FPROT). When enabled, block protection begins at any 512 byte boundary below the last address of flash, 0xFFFF. (see Section 4.7.4, “Flash Protection Register (FPROT and NVPROT).”)

After exit from reset, FPROT is loaded with the contents of the NVPROT location which is in the nonvolatile register block of the flash memory. FPROT cannot be changed directly from application software so a runaway program cannot alter the block protection settings. Since NVPROT is within the last 512 bytes of flash, if any amount of memory is protected, NVPROT is itself protected and cannot be altered (intentionally or unintentionally) by the application software. FPROT can be written through background debug commands which allows a way to erase and reprogram a protected flash memory.

The block protection mechanism is illustrated below. The FPS bits are used as the upper bits of the last address of unprotected memory. This address is formed by concatenating FPS7:FPS1 with logic 1 bits as shown. For example, in order to protect the last 8192 bytes of memory (addresses 0xE000 through 0xFFFF), the FPS bits must be set to 1101 111 which results in the value 0xDFFF as the last address of unprotected memory. In addition to programming the FPS bits to the appropriate value, FPDIS (bit 0 of NVPROT) must be programmed to logic 0 to enable block protection. Therefore the value 0xDE must be programmed into NVPROT to protect addresses 0xE000 through 0xFFFF.

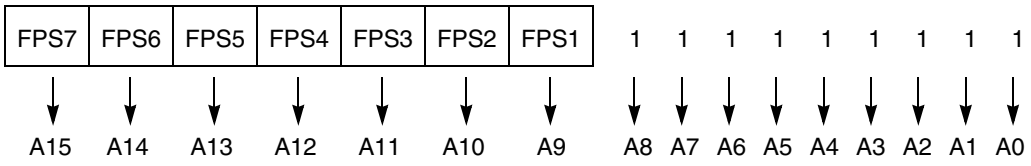


Figure 4-4. Block Protection Mechanism

One use for block protection is to block protect an area of flash memory for a bootloader program. This bootloader program then can be used to erase the rest of the flash memory and reprogram it. Because the

bootloader is protected, it remains intact even if MCU power is lost in the middle of an erase and reprogram operation.

4.5.7 Vector Redirection

Whenever any block protection is enabled, the reset and interrupt vectors will be protected. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. Vector redirection is enabled by programming the FNORED bit in the NVOPT register located at address 0xFFBF to zero. For redirection to occur, at least some portion but not all of the flash memory must be block protected by programming the NVPROT register located at address 0xFFBD. All of the interrupt vectors (memory locations 0xFFC0–0xFFFD) are redirected, though the reset vector (0xFFFE:FFFF) is not.

For example, if 512 bytes of flash are protected, the protected address region is from 0xFE00 through 0xFFFF. The interrupt vectors (0xFFC0–0xFFFD) are redirected to the locations 0xFDC0–0xFDFD. Now, if a TPM1 overflow interrupt is taken for instance, the values in the locations 0xFDE0:FDE1 are used for the vector instead of the values in the locations 0xFFE0:FFE1. This allows the user to reprogram the unprotected portion of the flash with new program code including new interrupt vector values while leaving the protected area, which includes the default vector locations, unchanged.

4.6 Security

The MC9S08JM60 Series includes circuitry to prevent unauthorized access to the contents of flash and RAM memory. When security is engaged, flash and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

Security is engaged or disengaged based on the state of two nonvolatile register bits (SEC01:SEC00) in the FOPT register. During reset, the contents of the nonvolatile location NVOPT are copied from flash into the working FOPT register in high-page register space. A user engages security by programming the NVOPT location which can be done at the same time the flash memory is programmed. The 1:0 state disengages security and the other three combinations engage security. Notice the erased state (1:1) makes the MCU secure. During development, whenever the flash is erased, it is good practice to immediately program the SEC00 bit to 0 in NVOPT so SEC01:SEC00 = 1:0. This would allow the MCU to remain unsecured after a subsequent reset.

The on-chip debug module cannot be enabled while the MCU is secure. The separate background debug controller can still be used for background memory access commands, but the MCU cannot enter active background mode except by holding BKGD/MS low at the rising edge of reset.

A user can choose to allow or disallow a security unlocking mechanism through an 8-byte backdoor security key. If the nonvolatile KEYEN bit in NVOPT/FOPT is 0, the backdoor key is disabled and there

is no way to disengage security without completely erasing all flash locations. If KEYEN is 1, a secure user program can temporarily disengage security by:

1. Writing 1 to KEYACC in the FCNFG register. This makes the flash module interpret writes to the backdoor comparison key locations (NVBACKKEY through NVBACKKEY+7) as values to be compared against the key rather than as the first step in a flash program or erase command.
2. Writing the user-entered key values to the NVBACKKEY through NVBACKKEY+7 locations. These writes must be done in order starting with the value for NVBACKKEY and ending with NVBACKKEY+7. STHX must not be used for these writes because these writes cannot be done on adjacent bus cycles. User software normally would get the key codes from outside the MCU system through a communication interface such as a serial I/O.
3. Writing 0 to KEYACC in the FCNFG register. If the 8-byte key that was just written matches the key stored in the flash locations, SEC01:SEC00 are automatically changed to 1:0 and security will be disengaged until the next reset.

The security key can be written only from secure memory (either RAM or flash), so it cannot be entered through background commands without the cooperation of a secure user program.

The backdoor comparison key (NVBACKKEY through NVBACKKEY+7) is located in flash memory locations in the nonvolatile register space so users can program these locations exactly as they would program any other flash memory location. The nonvolatile registers are in the same 512-byte block of flash as the reset and interrupt vectors, so block protecting that space also block protects the backdoor comparison key. Block protects cannot be changed from user application programs, so if the vector space is block protected, the backdoor security key mechanism cannot permanently change the block protect, security settings, or the backdoor key.

Security can always be disengaged through the background debug interface by taking these steps:

1. Disable any block protections by writing FPROT. FPROT can be written only with background debug commands, not from application software.
2. Mass erase flash if necessary.
3. Blank check flash. Provided flash is completely erased, security is disengaged until the next reset.

   To avoid returning to secure mode after the next reset, program NVOPT so SEC01:SEC00 = 1:0.

4.7 Flash Registers and Control Bits

The flash module has nine 8-bit registers in the high-page register space, three locations in the nonvolatile register space in flash memory which are copied into three corresponding high-page control registers at reset. There is also an 8-byte comparison key in flash memory. Refer to Table 4-3 and Table 4-4 for the absolute address assignments for all flash registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

4.7.1 Flash Clock Divider Register (FCDIV)

Bit 7 of this register is a read-only status flag. Bits 6 through 0 may be read at any time but can be written only one time. Before any erase or programming operations are possible, write to this register to set the frequency of the clock for the nonvolatile memory system within acceptable limits.

	7	6	5	4	3	2	1	0
R	DIVLD	PRDIV8	DIV5	DIV4	DIV3	DIV2	DIV1	DIV0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 4-5. Flash Clock Divider Register (FCDIV)

Table 4-6. FCDIV Register Field Descriptions

Field	Description
7 DIVLD	**Divisor Loaded Status Flag** — When set, this read-only status flag indicates that the FCDIV register has been written since reset. Reset clears this bit and the first write to this register causes this bit to become set regardless of the data written. 0 FCDIV has not been written since reset; erase and program operations disabled for flash. 1 FCDIV has been written since reset; erase and program operations enabled for flash.
6 PRDIV8	**Prescale (Divide) Flash Clock by 8** 0 Clock input to the flash clock divider is the bus rate clock. 1 Clock input to the flash clock divider is the bus rate clock divided by 8.
5:0 DIV[5:0]	**Divisor for Flash Clock Divider** — The flash clock divider divides the bus rate clock (or the bus rate clock divided by 8 if PRDIV8 = 1) by the value in the 6-bit DIV5:DIV0 field plus one. The resulting frequency of the internal flash clock must fall within the range of 200 kHz to 150 kHz for proper flash operations. Program/Erase timing pulses are one cycle of this internal flash clock which corresponds to a range of 5 μs to 6.7 μs. The automated programming logic uses an integer number of these pulses to complete an erase or program operation. See Equation 4-1, Equation 4-2, and Table 4-6.

$$\text{if PRDIV8 = 0 — } f_{FCLK} = f_{Bus} \div ([DIV5:DIV0] + 1) \qquad \textit{Eqn. 4-1}$$

$$\text{if PRDIV8 = 1 — } f_{FCLK} = f_{Bus} \div (8 \times ([DIV5:DIV0] + 1)) \qquad \textit{Eqn. 4-2}$$

Table 4-7 shows the appropriate values for PRDIV8 and DIV5:DIV0 for selected bus frequencies.

Table 4-7. Flash Clock Divider Settings

f_{Bus}	PRDIV8 (Binary)	DIV5:DIV0 (Decimal)	f_{FCLK}	Program/Erase Timing Pulse (5 μs Min, 6.7 μs Max)
24 MHz	1	14	200 kHz	5 μs
20 MHz	1	12	192.3 kHz	5.2 μs
10 MHz	0	49	200 kHz	5 μs
8 MHz	0	39	200 kHz	5 μs
4 MHz	0	19	200 kHz	5 μs
2 MHz	0	9	200 kHz	5 μs
1 MHz	0	4	200 kHz	5 μs
200 kHz	0	0	200 kHz	5 μs
150 kHz	0	0	150 kHz	6.7 μs

4.7.2 Flash Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from flash into FOPT. Bits 5 through 2 are not used and always read 0. This register may be read at any time, but writes have no meaning or effect. To change the value in this register, erase and reprogram the NVOPT location in flash memory as usual and then issue a new MCU reset.

	7	6	5	4	3	2	1	0
R	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
W								
Reset	This register is loaded from nonvolatile location NVOPT during reset.							

= Unimplemented or Reserved

Figure 4-6. Flash Options Register (FOPT)

Table 4-8. FOPT Register Field Descriptions

Field	Description
7 KEYEN	**Backdoor Key Mechanism Enable** — When this bit is 0, the backdoor key mechanism cannot be used to disengage security. The backdoor key mechanism is accessible only from user (secured) firmware. BDM commands cannot be used to write key comparison values that would unlock the backdoor key. For more detailed information about the backdoor key mechanism, refer to Section 4.6, "Security." 0 No backdoor key access allowed. 1 If user firmware writes an 8-byte value that matches the nonvolatile backdoor key (NVBACKKEY through NVBACKKEY+7 in that order), security is temporarily disengaged until the next MCU reset.
6 FNORED	**Vector Redirection Disable** — When this bit is 1, then vector redirection is disabled. 0 Vector redirection enabled. 1 Vector redirection disabled.
1:0 SEC0[1:0]	**Security State Code** — This 2-bit field determines the security state of the MCU as shown in Table 4-9. When the MCU is secure, the contents of RAM and flash memory cannot be accessed by instructions from any unsecured source including the background debug interface. For more detailed information about security, refer to Section 4.6, "Security."

Table 4-9. Security States

SEC01:SEC00	Description
0:0	secure
0:1	secure
1:0	unsecured
1:1	secure

SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of flash.

4.7.3 Flash Configuration Register (FCNFG)

Bits 7 through 5 may be read or written at any time. Bits 4 through 0 always read 0 and cannot be written.

	7	6	5	4	3	2	1	0
R	0	0	KEYACC	0	0	0	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 4-7. Flash Configuration Register (FCNFG)

Table 4-10. FCNFG Register Field Descriptions

Field	Description
5 KEYACC	**Enable Writing of Access Key** — This bit enables writing of the backdoor comparison key. For more detailed information about the backdoor key mechanism, refer to Section 4.6, “Security.” 0 Writes to 0xFFB0–0xFFB7 are interpreted as the start of a flash programming or erase command. 1 Writes to NVBACKKEY (0xFFB0–0xFFB7) are interpreted as comparison key writes.

4.7.4 Flash Protection Register (FPROT and NVPROT)

During reset, the contents of the nonvolatile location NVPROT is copied from flash into FPROT. This register may be read at any time, but user program writes have no meaning or effect. Background debug commands can write to FPROT.

	7	6	5	4	3	2	1	0
R	FPS7	FPS6	FPS5	FPS4	FPS3	FPS2	FPS1	FPDIS
W	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
Reset	This register is loaded from nonvolatile location NVPROT during reset.							

1 Background commands can be used to change the contents of these bits in FPROT.

Figure 4-8. Flash Protection Register (FPROT)

Table 4-11. FPROT Register Field Descriptions

Field	Description
7:1 FPS[7:1]	**Flash Protect Select Bits** — When FPDIS = 0, this 7-bit field determines the ending address of unprotected flash locations at the high address end of the flash. Protected flash locations cannot be erased or programmed.
0 FPDIS	**Flash Protection Disable** 0 Flash block specified by FPS[7:1] is block protected (program and erase not allowed). 1 No flash block is protected.

4.7.5 Flash Status Register (FSTAT)

Bits 3, 1, and 0 always read 0 and writes have no meaning or effect. The remaining five bits are status bits that can be read at any time. Writes to these bits have special meanings that are discussed in the bit descriptions.

	7	6	5	4	3	2	1	0
R	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
W								
Reset	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 4-9. Flash Status Register (FSTAT)

Table 4-12. FSTAT Register Field Descriptions

Field	Description
7 FCBEF	**Flash Command Buffer Empty Flag** — The FCBEF bit is used to launch commands. It also indicates that the command buffer is empty so that a new command sequence can be executed when performing burst programming. The FCBEF bit is cleared by writing a one to it or when a burst program command is transferred to the array for programming. Only burst program commands can be buffered. 0 Command buffer is full (not ready for additional commands). 1 A new burst program command may be written to the command buffer.
6 FCCF	**Flash Command Complete Flag** — FCCF is set automatically when the command buffer is empty and no command is being processed. FCCF is cleared automatically when a new command is started (by writing 1 to FCBEF to register a command). Writing to FCCF has no meaning or effect. 0 Command in progress 1 All commands complete
5 FPVIOL	**Protection Violation Flag** — FPVIOL is set automatically when FCBEF is cleared to register a command that attempts to erase or program a location in a protected block (the erroneous command is ignored). FPVIOL is cleared by writing a 1 to FPVIOL. 0 No protection violation. 1 An attempt was made to erase or program a protected location.

Table 4-12. FSTAT Register Field Descriptions (continued)

Field	Description
4 FACCERR	**Access Error Flag** — FACCERR is set automatically when the proper command sequence is not obeyed exactly (the erroneous command is ignored), if a program or erase operation is attempted before the FCDIV register has been initialized, or if the MCU enters stop while a command was in progress. For a more detailed discussion of the exact actions that are considered access errors, see Section 4.5.5, "Access Errors." FACCERR is cleared by writing a 1 to FACCERR. Writing a 0 to FACCERR has no meaning or effect. 0 No access error. 1 An access error has occurred.
2 FBLANK	**Flash Verified as All Blank (erased) Flag** — FBLANK is set automatically at the conclusion of a blank check command if the entire flash array was verified to be erased. FBLANK is cleared by clearing FCBEF to write a new valid command. Writing to FBLANK has no meaning or effect. 0 After a blank check command is completed and FCCF = 1, FBLANK = 0 indicates the flash array is not completely erased. 1 After a blank check command is completed and FCCF = 1, FBLANK = 1 indicates the flash array is completely erased (all 0xFF).

4.7.6 Flash Command Register (FCMD)

Only five command codes are recognized in normal user modes as shown in Table 4-14. Refer to Section 4.5.3, "Program and Erase Command Execution," for a detailed discussion of flash programming and erase operations.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	FCMD7	FCMD6	FCMD5	FCMD4	FCMD3	FCMD2	FCMD1	FCMD0
Reset	0	0	0	0	0	0	0	0

Figure 4-10. Flash Command Register (FCMD)

Table 4-13. FCMD Register Field Descriptions

Field	Description
FCMD[7:0]	**Flash Command Bits** — See Table 4-14

Table 4-14. Flash Commands

Command	FCMD	Equate File Label
Blank check	0x05	mBlank
Byte program	0x20	mByteProg
Byte program — burst mode	0x25	mBurstProg
Page erase (512 bytes/page)	0x40	mPageErase
Mass erase (all flash)	0x41	mMassErase

All other command codes are illegal and generate an access error.

It is not necessary to perform a blank check command after a mass erase operation. Only blank check is required as part of the security unlocking mechanism.

Chapter 5
Resets, Interrupts, and System Configuration

5.1 Introduction

This chapter discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the MC9S08JM60 series. Some interrupt sources from peripheral modules are discussed in greater detail within other chapters of this data manual. This chapter gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog, are not part of on-chip peripheral systems with their own sections but are part of the system control logic.

5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vectors for each module (reduces polling overhead) (see Table 5-1)

5.3 MCU Reset

Resetting the MCU provides a way to start processing from a known set of initial conditions. During reset, most control and status registers are forced to initial values and the program counter is loaded from the reset vector (0xFFFE:0xFFFF). On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pullup devices disabled. The I bit in the condition code register (CCR) is set to block maskable interrupts so the user program has a chance to initialize the stack pointer (SP) and system control settings. SP is forced to 0x00FF at reset.

The MC9S08JM60 series has seven sources for reset:

- Power-on reset (POR)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Background debug forced reset
- External reset pin ($\overline{\text{RESET}}$)
- Clock generator loss of lock and loss of clock reset (LOC)

Each of these sources, with the exception of the background debug forced reset, has an associated bit in the system reset status (SRS) register.

5.4 Computer Operating Properly (COP) Watchdog

The COP watchdog is intended to force a system reset when the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), application software must reset the COP counter periodically. If the application program gets lost and fails to reset the COP counter before it times out, a system reset is generated to force the system back to a known starting point.

After any reset, the COP watchdog is enabled (see Section 5.7.4, "System Options Register 1 (SOPT1)," for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPT bits in SOPT1.

The COP counter is reset by writing 0x55 and 0xAA (in this order) to the address of SRS during the selected timeout period. Writes do not affect the data in the read-only SRS. As soon as the write sequence is done, the COP timeout period is restarted. If the program fails to do this during the time-out period, the MCU will reset. Also, if any value other than 0x55 or 0xAA is written to SRS, the MCU is immediately reset.

The COPCLKS bit in SOPT2 (see Section 5.7.5, "System Options Register 2 (SOPT2)," for additional information) selects the clock source used for the COP timer. The clock source options are either the bus clock or an internal 1 kHz LPO clock source. With each clock source, there are three associated time-outs controlled by the COPT bits in SOPT1. Table 5-6 summaries the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1 kHz LPO clock source and the longest time-out (2^{10} cycles).

When the bus clock source is selected, windowed COP operation is available by setting COPW in the SOPT2 register. In this mode, writes to the SRS register to clear the COP timer must occur in the last 25% of the selected timeout period. A premature write immediately resets the MCU. When the 1 kHz LPO clock source is selected, windowed COP operation is not available.

The COP counter is initialized by the first writes to the SOPT1 and SOPT2 registers and after any system reset. Subsequent writes to SOPT1 and SOPT2 have no effect on COP operation. Even if the application will use the reset default settings of COPT, COPCLKS, and COPW bits, the user must write to the write-once SOPT1 and SOPT2 registers during reset initialization to lock in the settings. This will prevent accidental changes if the application program gets lost.

The write to SRS that services (clears) the COP counter must not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

If the bus clock source is selected, the COP counter does not increment while the MCU is in background debug mode or while the system is in stop mode. The COP counter resumes when the MCU exits background debug mode or stop mode.

If the 1 kHz LPO clock source is selected, the COP counter is re-initialized to zero upon entry to either background debug mode or stop mode and begins from zero upon exit from background debug mode or stop mode.

5.5 Interrupts

Interrupts provide a way to save the current CPU status and registers, execute an interrupt service routine (ISR), and then restore the CPU status so processing resumes where it left off before the interrupt. Other than the software interrupt (SWI), which is a program instruction, interrupts are caused by hardware events such as an edge on the IRQ pin or a timer-overflow event. The debug module can also generate an SWI under certain circumstances.

If an event occurs in an enabled interrupt source, an associated read-only status flag will become set. The CPU will not respond until and unless the local interrupt enable is a logic 1 to enable the interrupt. The I bit in the CCR is 0 to allow interrupts. The global interrupt mask (I bit) in the CCR is initially set after reset which masks (prevents) all maskable interrupt sources. The user program initializes the stack pointer and performs other system setup before clearing the I bit to allow the CPU to respond to interrupts.

When the CPU receives a qualified interrupt request, it completes the current instruction before responding to the interrupt. The interrupt sequence obeys the same cycle-by-cycle sequence as the SWI instruction and consists of:

- Saving the CPU registers on the stack
- Setting the I bit in the CCR to mask further interrupts
- Fetching the interrupt vector for the highest-priority interrupt that is currently pending
- Filling the instruction queue with the first three bytes of program information starting from the address fetched from the interrupt vector locations

While the CPU is responding to the interrupt, the I bit is automatically set to avoid the possibility of another interrupt interrupting the ISR itself (this is called nesting of interrupts). Normally, the I bit is restored to 0 when the CCR is restored from the value stacked on entry to the ISR. In rare cases, the I bit may be cleared inside an ISR (after clearing the status flag that generated the interrupt) so that other interrupts can be serviced without waiting for the first service routine to finish. This practice is not recommended for anyone other than the most experienced programmers because it can lead to subtle program errors that are difficult to debug.

The interrupt service routine ends with a return-from-interrupt (RTI) instruction which restores the CCR, A, X, and PC registers to their pre-interrupt values by reading the previously saved information off the stack.

NOTE

For compatibility with the M68HC08, the H register is not automatically saved and restored. It is good programming practice to push H onto the stack at the start of the interrupt service routine (ISR) and restore it immediately before the RTI that is used to return from the ISR.

When two or more interrupts are pending when the I bit is cleared, the highest priority source is serviced first (see Table 5-1).

5.5.1 Interrupt Stack Frame

Figure 5-1 shows the contents and organization of a stack frame. Before the interrupt, the stack pointer (SP) points at the next available byte location on the stack. The current values of CPU registers are stored on the stack starting with the low-order byte of the program counter (PCL) and ending with the CCR. After stacking, the SP points at the next available location on the stack which is the address that is one less than the address where the CCR was saved. The PC value that is stacked is the address of the instruction in the main program that would have executed next if the interrupt had not occurred.

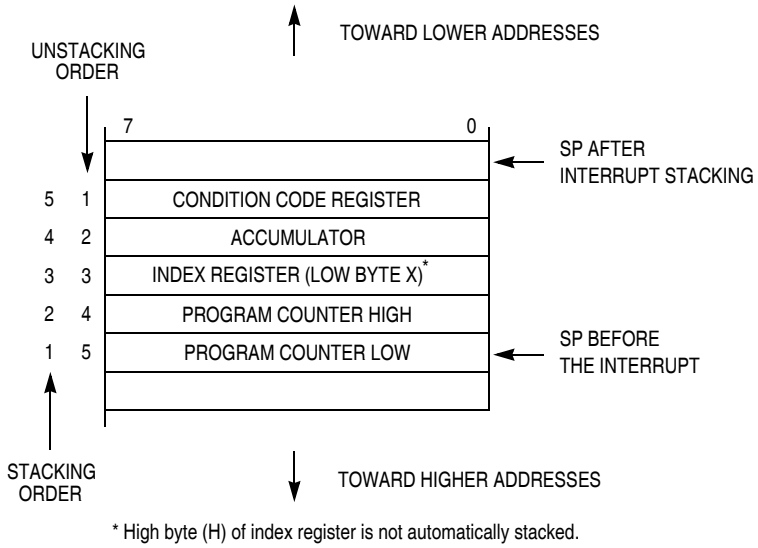


Figure 5-1. Interrupt Stack Frame

When an RTI instruction is executed, these values are recovered from the stack in reverse order. As part of the RTI sequence, the CPU fills the instruction pipeline by reading three bytes of program information, starting from the PC address recovered from the stack.

The status flag causing the interrupt must be acknowledged (cleared) before returning from the ISR. Typically, the flag must be cleared at the beginning of the ISR so that if another interrupt is generated by this same source, it will be registered so it can be serviced after completion of the current ISR.

5.5.2 External Interrupt Request (IRQ) Pin

External interrupts are managed by the IRQSC status and control register. When the IRQ function is enabled, synchronous logic monitors the pin for edge-only or edge-and-level events. When the MCU is in stop mode and system clocks are shut down, a separate asynchronous path is used so the IRQ (if enabled) can wake the MCU.

5.5.2.1 Pin Configuration Options

The IRQ pin enable (IRQPE) control bit in IRQSC must be 1 in order for the IRQ pin to act as the interrupt request (IRQ) input. As an IRQ input, the user can choose the polarity of edges or levels detected (IRQEDG), whether the pin detects edges-only or edges and levels (IRQMOD), and whether an event causes an interrupt or only sets the IRQF flag which can be polled by software.

The IRQ pin, when enabled, defaults to use an internal pull device (IRQPDD = 0), the device is a pullup or pull-down depending on the polarity chosen. If the user desires to use an external pullup or pull-down, the IRQPDD can be written to a 1 to turn off the internal device.

BIH and BIL instructions may be used to detect the level on the IRQ pin when the pin is configured to act as the IRQ input.

NOTE

This pin does not contain a clamp diode to V_{DD} and must not be driven above V_{DD}. The voltage measured on the internally pulled up IRQ pin may be as low as $V_{DD} - 0.7$ V. The internal gates connected to this pin are pulled all the way to V_{DD}.

5.5.2.2 Edge and Level Sensitivity

The IRQMOD control bit re-configure the detection logic so it detects edge events and pin levels. In this edge detection mode, the IRQF status flag becomes set when an edge is detected (when the IRQ pin changes from the deasserted to the asserted level), but the flag is continuously set (and cannot be cleared) as long as the IRQ pin remains at the asserted level.

5.5.3 Interrupt Vectors, Sources, and Local Masks

Table 5-1 provides a summary of all interrupt sources. Higher-priority sources are located toward the bottom of the table. The high-order byte of the address for the interrupt service routine is located at the first address in the vector address column, and the low-order byte of the address for the interrupt service routine is located at the next higher address.

When an interrupt condition occurs, an associated flag bit becomes set. If the associated local interrupt enable is 1, an interrupt request is sent to the CPU. Within the CPU, if the global interrupt mask (I bit in the CCR) is 0, the CPU will finish the current instruction, stack the PCL, PCH, X, A, and CCR CPU registers, set the I bit, and then fetch the interrupt vector for the highest priority pending interrupt. Processing then continues in the interrupt service routine.

Table 5-1. Vector Summary (from Lowest to Highest Priority)

Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
31 to 30	0xFFC0:FFC1 0xFFC2:FFC3	Unused vector space (available for user program)				
29	0xFFC4:FFC5	Vrtc	System control	RTIF	RTIE	RTC real-time interrupt
28	0xFFC6:FFC7	Viic	IIC	IICIF	IICIE	IIC
27	0xFFC8:FFC9	Vacmp	ACMP	ACF	ACIE	ACMP
26	0xFFCA:FFCB	Vadc	ADC	COCO	AIEN	ADC
25	0xFFCC:FFCD	Vkeyboard	KBI	KBF	KBIE	Keyboard pins
24	0xFFCE:FFCF	Vsci2tx	SCI2	TDRE TC	TIE TCIE	SCI2 transmit

Table 5-1. Vector Summary (from Lowest to Highest Priority) (continued)

Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
23	0xFFD0:FFD1	Vsci2rx	SCI2	IDLE RDRF	ILIE RIE	SCI2 receive
22	0xFFD2:FFD3	Vsci2err	SCI2	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI2 error
21	0xFFD4:FFD5	Vsci1tx	SCI1	TDRE TC	TIE TCIE	SCI1 transmit
20	0xFFD6:FFD7	Vsci1rx	SCI1	IDLE RDRF	ILIE RIE	SCI1 receive
19	0xFFD8:FFD9	Vsci1err	SCI1	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI1 error
18	0xFFDA:FFDB	Vtpm2ovf	TPM2	TOF	TOIE	TPM2 overflow
17	0xFFDC:FFDD	Vtpm2ch1	TPM2	CH1F	CH1IE	TPM2 channel 1
16	0xFFDE:FFDF	Vtpm2ch0	TPM2	CH0F	CH0IE	TPM2 channel 0
15	0xFFE0:FFE1	Vtpm1ovf	TPM1	TOF	TOIE	TPM1 overflow
14	0xFFE2:FFE3	Vtpm1ch5	TPM1	CH5F	CH5IE	TPM1 channel 5
13	0xFFE4:FFE5	Vtpm1ch4	TPM1	CH4F	CH4IE	TPM1 channel 4
12	0xFFE6:FFE7	Vtpm1ch3	TPM1	CH3F	CH3IE	TPM1 channel 3
11	0xFFE8:FFE9	Vtpm1ch2	TPM1	CH2F	CH2IE	TPM1 channel 2
10	0xFFEA:FFEB	Vtpm1ch1	TPM1	CH1F	CH1IE	TPM1 channel 1
9	0xFFEC:FFED	Vtpm1ch0	TPM1	CH0F	CH0IE	TPM1 channel 0
8	0xFFEE:FFEF	Reserved	—	—	—	—
7	0xFFF0:FFF1	Vusb	USB	STALLF RESUMEF SLEEPF TOKDNEF SOFTOKF ERRORF USBRSTF	STALL RESUME SLEEP TOKDNE SOFTOK ERROR USBRST	USB Status
6	0xFFF2:FFF3	Vspi2	SPI2	SPRF MODF SPTEF SPMF	SPIE SPIE SPTIE SPMIE	SPI2
5	0xFFF4:FFF5	Vspi1	SPI1	SPRF MODF SPTEF SPMF	SPIE SPIE SPTIE SPMIE	SPI1
4	0xFFF6:FFF7	Vlol	MCG	LOLS	LOLIE	MCG loss of lock
3	0xFFF8:FFF9	Vlvd	System control	LVWF	LVWIE	Low-voltage detect
2	0xFFFA:FFFB	Virq	IRQ	IRQF	IRQIE	IRQ pin

Table 5-1. Vector Summary (from Lowest to Highest Priority) (continued)

Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
1	0xFFFC:FFFD	Vswi	Core	SWI Instruction	—	Software interrupt
0	0xFFFE:FFFF	Vreset	System control	COP LVD $\overline{\text{RESET}}$ pin Illegal opcode LOC POR BDFR	COPE LVDRE — ILOP CME POR BDFR	Watchdog timer Low-voltage detect External pin Illegal opcode Loss of clock Power-on-reset BDM-forced reset

5.6 Low-Voltage Detect (LVD) System

The MC9S08JM60 series includes a system to protect against low-voltage conditions in order to protect memory contents and control MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and a LVD circuit with trip voltages for warning and detection. The LVD circuit is enabled when LVDE in SPMSC1 is set to 1. The LVD is disabled upon entering any of the stop modes unless LVDSE is set in SPMSC1. If LVDSE and LVDE are both set, then the MCU cannot enter stop2 (it will enter stop3 instead), and the current consumption in stop3 with the LVD enabled will be higher.

5.6.1 Power-On Reset Operation

When power is initially applied to the MCU, or when the supply voltage drops below the power-on reset rearm voltage level, V_{POR}, the POR circuit will cause a reset condition. As the supply voltage rises, the LVD circuit will hold the MCU in reset until the supply has risen above the low voltage detection low threshold, V_{LVDL}. Both the POR bit and the LVD bit in SRS are set following a POR.

5.6.2 Low-Voltage Detection (LVD) Reset Operation

The LVD can be configured to generate a reset upon detection of a low voltage condition by setting LVDRE to 1. The low voltage detection threshold is determined by the LVDV bit. After an LVD reset has occurred, the LVD system will hold the MCU in reset until the supply voltage has risen above the low voltage detection threshold. The LVD bit in the SRS register is set following either an LVD reset or POR.

5.6.3 Low-Voltage Warning (LVW) Interrupt Operation

The LVD system has a low voltage warning flag to indicate to the user that the supply voltage is approaching the low voltage condition. When a low voltage warning condition is detected and is configured for interrupt operation (LVWIE set to 1), LVWF in SPMSC1 will be set and an LVW interrupt request will occur.

5.7 Reset, Interrupt, and System Control Registers and Control Bits

One 8-bit register in the direct page register space and eight 8-bit registers in the high-page register space are related to reset and interrupt systems.

Refer to the direct-page register summary in Chapter 4, "Memory," of this data sheet for the absolute address assignments for all registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

Some control bits in the SOPT1 and SPMSC2 registers are related to modes of operation. Although brief descriptions of these bits are provided here, the related functions are discussed in greater detail in Chapter 3, "Modes of Operation."

5.7.1 Interrupt Pin Request Status and Control Register (IRQSC)

This direct page register includes status and control bits, which are used to configure the IRQ function, report status, and acknowledge IRQ events.

	7	6	5	4	3	2	1	0
R	0	IRQPDD	IRQEDG	IRQPE	IRQF	0	IRQIE	IRQMOD
W						IRQACK		
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 5-2. Interrupt Request Status and Control Register (IRQSC)

Table 5-2. IRQSC Register Field Descriptions

Field	Description
6 IRQPDD	**Interrupt Request (IRQ) Pull Device Disable** — This read/write control bit is used to disable the internal pullup device when the IRQ pin is enabled (IRQPE = 1) allowing for an external device to be used. 0 IRQ pull device enabled if IRQPE = 1. 1 IRQ pull device disabled if IRQPE = 1.
5 IRQEDG	**Interrupt Request (IRQ) Edge Select** — This read/write control bit is used to select the polarity of edges or levels on the IRQ pin that cause IRQF to be set. The IRQMOD control bit determines whether the IRQ pin is sensitive to both edges and levels or only edges. When the IRQ pin is enabled as the IRQ input and is configured to detect rising edges, the optional pullup resistor is re-configured as an optional pulldown resistor. 0 IRQ is falling edge or falling edge/low-level sensitive. 1 IRQ is rising edge or rising edge/high-level sensitive.
4 IRQPE	**IRQ Pin Enable** — This read/write control bit enables the IRQ pin function. When this bit is set the IRQ pin can be used as an interrupt request. 0 IRQ pin function is disabled. 1 IRQ pin function is enabled.
3 IRQF	**IRQ Flag** — This read-only status bit indicates when an interrupt request event has occurred. 0 No IRQ request. 1 IRQ event detected.

Table 5-2. IRQSC Register Field Descriptions (continued)

Field	Description
2 IRQACK	**IRQ Acknowledge** — This write-only bit is used to acknowledge interrupt request events (write 1 to clear IRQF). Writing 0 has no meaning or effect. Reads always return 0. If edge-and-level detection is selected (IRQMOD = 1), IRQF cannot be cleared while the IRQ pin remains at its asserted level.
1 IRQIE	**IRQ Interrupt Enable** — This read/write control bit determines whether IRQ events generate an interrupt request. 0 Interrupt request when IRQF set is disabled (use polling). 1 Interrupt requested whenever IRQF = 1.
0 IRQMOD	**IRQ Detection Mode** — This read/write control bit selects either edge-only detection or edge-and-level detection. See Section 5.5.2.2, "Edge and Level Sensitivity," for more details. 0 IRQ event on falling/rising edges only. 1 IRQ event on falling/rising edges and low/high levels.

5.7.2 System Reset Status Register (SRS)

This register includes seven read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by writing 1 to BDFR in the SBDFR register, none of the status bits in SRS will be set. Writing any value except 0x55 and 0xAA in sequence to this register address causes the MCU reset with the source of COP. The reset state of these bits depends on what caused the MCU to reset.

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	0	LOC	LVD	—
W	Writing any value to SRS address clears COP watchdog timer.							
POR	1	0	0	0	0	0	1	0
LVR:	U	0	0	0	0	0	1	0
Any other reset:	0	(1)	(1)	(1)	0	(1)	0	0

U = Unaffected by reset

1 Any of these reset sources that are active at the time of reset will cause the corresponding bit(s) to be set; bits corresponding to sources that are not active at the time of reset will be cleared.

Figure 5-3. System Reset Status (SRS)

Table 5-3. SRS Register Field Descriptions

Field	Description
7 POR	**Power-On Reset** — Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVR) status bit is also set to indicate that the reset occurred while the internal supply was below the LVR threshold. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	**External Reset Pin** — Reset was caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 Reset came from external reset pin.

Table 5-3. SRS Register Field Descriptions (continued)

Field	Description
5 COP	**Computer Operating Properly (COP) Watchdog** — Reset was caused by the COP watchdog timer timing out. This reset source may be blocked by COPE = 0. 0 Reset not caused by COP timeout. 1 Reset caused by COP timeout.
4 ILOP	**Illegal Opcode** — Reset was caused by an attempt to execute an unimplemented or illegal opcode. The STOP instruction is considered illegal if stop is disabled by STOPE = 0 in the SOPT register. The BGND instruction is considered illegal if active background mode is disabled by ENBDM = 0 in the BDCSC register. 0 Reset not caused by an illegal opcode. 1 Reset caused by an illegal opcode.
2 LOC	**Loss-of-Clock Reset** — Reset was caused by a loss of external clock. 0 Reset not caused by a loss of external clock. 1 Reset caused by a loss of external clock.
1 LVD	**Low Voltage Detect** — If the LVD is enable with the LVDRE or LVDSE bit is set, and the supply drops below the LVD trip voltage, an LVD reset will occur. This bit is also set by POR. 0 Reset not caused by LVD trip or POR. 1 Reset caused by LVD trip or POR.

5.7.3 System Background Debug Force Reset Register (SBDFR)

This register contains a single write-only control bit. A serial background command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return 0x00.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								BDFR[1]
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

1 BDFR is writable only through serial background debug commands, not from user programs.

Figure 5-4. System Background Debug Force Reset Register (SBDFR)

Table 5-4. SBDFR Register Field Descriptions

Field	Description
0 BDFR	**Background Debug Force Reset** — A serial background command such as WRITE_BYTE may be used to allow an external debug host to force a target system reset. Writing logic 1 to this bit forces an MCU reset. This bit cannot be written from a user program.

5.7.4 System Options Register 1 (SOPT1)

This register may be read at any time. Bits 3 and 2 are unimplemented and always read 0. This is a write-once register so only the first write after reset is honored. Any subsequent attempt to write to SOPT (intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. SOPT

must be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

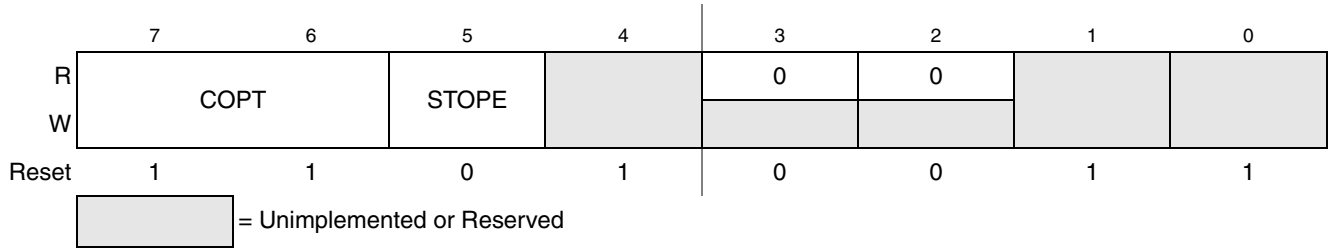


Figure 5-5. System Options Register (SOPT1)

Table 5-5. SOPT1 Register Field Descriptions

Field	Description
7:6 COPT[1:0]	**COP Watchdog Timeout** — These write-once bits select the timeout period of the COP. COPT along with COPCLKS in SOPT2 defines the COP timeout period. See Table 5-6.
5 STOPE	**Stop Mode Enable** — This write-once bit defaults to 0 after reset, which disables stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced. 0 Stop mode disabled. 1 Stop mode enabled.

Table 5-6. COP Configuration Options

Control Bits		Clock Source	COP Window[1] Opens (COPW = 1)	COP Overflow Count
COPCLKS	COPT[1:0]			
N/A	0:0	N/A	N/A	COP is disabled
0	0:1	1 kHz LPO clock	N/A	2^5 cycles (32 ms[2])
0	1:0	1 kHz LPO clock	N/A	2^8 cycles (256 ms[1])
0	1:1	1 kHz LPO clock	N/A	2^{10} cycles (1.024 s[1])
1	0:1	BUSCLK	6144 cycles	2^{13} cycles
1	1:0	BUSCLK	49,152 cycles	2^{16} cycles
1	1:1	BUSCLK	196,608 cycles	2^{18} cycles

1 Windowed COP operation requires the user to clear the COP timer in the last 25% of the selected timeout period. This column displays the minimum number of clock counts required before the COP timer can be reset when in windowed COP mode (COPW = 1).

2 Values shown in milliseconds based on t_{LPO} = 1 ms. See t_{LPO} in the appendix Section A.12.1, "Control Timing," for the tolerance of this value.

5.7.5 System Options Register 2 (SOPT2)

	7	6	5	4	3	2	1	0
R	COPCLKS[1]	COPW[1]	0	0	0	SPI1FE	SPI2FE	ACIC
W								
Reset	0	0	0	0	0	1	1	0

= Unimplemented or Reserved

[1] This bit can be written only one time after reset. Additional writes are ignored.

Figure 5-6. System Options Register 2 (SOPT2)

Table 5-7. SOPT2 Register Field Descriptions

Field	Description
7 COPCLKS	**COP Watchdog Clock Select** — This write-once bit selects the clock source of the COP watchdog. 0 Internal 1 kHz LPO clock is source to COP. 1 Bus clock is source to COP.
6 COPW	**COP Window** — This write-once bit selects the COP operation mode. When set, the 0x55-0xAA write sequence to the SRS register must occur in the last 25% of the selected period. Any write to the SRS register during the first 75% of the selected period will reset the MCU. 0 Normal COP operation. 1 Window COP operation.
2 SPI1FE	**SPI1 Ports Input Filter Enable** 0 Disable input filter on SPI1 port pins to allow for higher maximum SPI baud rate. 1 Enable input filter on SPI1 port pins to eliminate noise and restrict maximum SPI baud rate.
1 SPI2FE	**SPI2 Ports Input Filter Enable** 0 Disable input filter on SPI2 port pins to allow for higher maximum SPI baud rate. 1 Enable input filter on SPI2 port pins to eliminate noise and restrict maximum SPI baud rate.
0 ACIC	**Analog Comparator to Input Capture Enable**— This bit connects the output of ACMP to TPM input channel 0. 0 ACMP output not connected to TPM input channel 0. 1 ACMP output connected to TPM input channel 0.

5.7.6 System Device Identification Register (SDIDH, SDIDL)

This read-only register is included so host development systems can identify the HCS08 derivative and revision number. This allows the development software to recognize where specific memory blocks, registers, and control bits are located in a target MCU.

	7	6	5	4	3	2	1	0
R					ID11	ID10	ID9	ID8
W								
Reset	—	—	—	—	0	0	0	0

= Unimplemented or Reserved

Figure 5-7. System Device Identification Register — High (SDIDH)

Table 5-8. SDIDH Register Field Descriptions

Field	Description
7:4 Reserved	**Bits 7:4 are reserved. Reading these bits will result in an indeterminate value; writes have no effect.**
3:0 ID[11:8]	**Part Identification Number** — Each derivative in the HCS08 Family has a unique identification number. The MC9S08JM60 Series is hard coded to the value 0x016. See also ID bits in Table 5-9.

	7	6	5	4	3	2	1	0
R	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
W								
Reset	0	0	0	1	0	1	1	0

= Unimplemented or Reserved

Figure 5-8. System Device Identification Register — Low (SDIDL)

Table 5-9. SDIDL Register Field Descriptions

Field	Description
7:0 ID[7:0]	**Part Identification Number** — Each derivative in the HCS08 Family has a unique identification number. The MC9S08JM60 Series is hard coded to the value 0x016. See also ID bits in Table 5-8.

5.7.7 System Power Management Status and Control 1 Register (SPMSC1)

This high page register contains status and control bits to support the low-voltage detect function, and to enable the bandgap voltage reference for use by the ADC module. This register must be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

	7	6	5	4	3	2	1	0
R	LVWF[1]	0	LVWIE	LVDRE[2]	LVDSE	LVDE[2]	0	BGBE
W		LVWACK						
Reset:	0	0	0	1	1	1	0	0

= Unimplemented or Reserved

1 LVWF will be set in the case when V_{Supply} transitions below the trip point or after reset and V_{Supply} is already below V_{LVW}.

2 This bit can be written only one time after reset. Additional writes are ignored.

Figure 5-9. System Power Management Status and Control 1 Register (SPMSC1)

Table 5-10. SPMSC1 Register Field Descriptions

Field	Description
7 LVWF	**Low-Voltage Warning Flag** — The LVWF bit indicates the low-voltage warning status. 0 low-voltage warning is not present. 1 low-voltage warning is present or was present.
6 LVWACK	**Low-Voltage Warning Acknowledge** — If LVWF = 1, a low-voltage condition has occurred. To acknowledge this low-voltage warning, write 1 to LVWACK, which will automatically clear LVWF to 0 if the low-voltage warning is no longer present.
5 LVWIE	**Low-Voltage Warning Interrupt Enable** — This bit enables hardware interrupt requests for LVWF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVWF = 1.
4 LVDRE	**Low-Voltage Detect Reset Enable** — This write-once bit enables LVD events to generate a hardware reset (provided LVDE = 1). 0 LVD events do not generate hardware resets. 1 Force an MCU reset when an enabled low-voltage detect event occurs.
3 LVDSE	**Low-Voltage Detect Stop Enable** — Provided LVDE = 1, this read/write bit determines whether the low-voltage detect function operates when the MCU is in stop mode. 0 Low-voltage detect disabled during stop mode. 1 Low-voltage detect enabled during stop mode.
2 LVDE	**Low-Voltage Detect Enable** — This write-once bit enables low-voltage detect logic and qualifies the operation of other bits in this register. 0 LVD logic disabled. 1 LVD logic enabled.
0 BGBE	**Bandgap Buffer Enable** — This bit enables an internal buffer for the bandgap voltage reference for use by ACMP or ADC module on one of its internal channels. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

5.7.8 System Power Management Status and Control 2 Register (SPMSC2)

This register is used to report the status of the low voltage warning function, and to configure the stop mode behavior of the MCU.

	7	6	5	4	3	2	1	0
R	0	0	LVDV	LVWV	PPDF	0	0	PPDC[1]
W						PPDACK		
Power-on Reset:	0	0	0	0	0	0	0	0
LVD Reset:	0	0	u	u	0	0	0	0
Any other Reset:	0	0	u	u	0	0	0	0

= Unimplemented or Reserved u = Unaffected by reset

[1] This bit can be written only one time after reset. Additional writes are ignored.

Figure 5-10. System Power Management Status and Control 2 Register (SPMSC2)

Table 5-11. SPMSC2 Register Field Descriptions

Field	Description
5 LVDV	**Low-Voltage Detect Voltage Select** — This bit selects the low voltage detect (LVD) trip point setting.It also selects the warning voltage range. See Table 5-12.
4 LVWV	**Low-Voltage Warning Voltage Select** — This bit selects the low voltage warning (LVW) trip point voltage. See Table 5-12.
3 PPDF	**Partial Power Down Flag** — This read-only status bit indicates that the MCU has recovered from stop2 mode. 0 MCU has not recovered from stop2 mode. 1 MCU recovered from stop2 mode.
2 PPDACK	**Partial Power Down Acknowledge** — Writing a 1 to PPDACK clears the PPDF bit
0 PPDC	**Partial Power Down Control** — This write-once bit controls whether stop2 or stop3 mode is selected. 0 Stop3 mode enabled. 1 Stop2, partial power down, mode enabled.

Table 5-12. LVD and LVW trip point typical values[1]

LVDV:LVWV	LVW Trip Point	LVD Trip Point
0:0	V_{LVW0} = 2.74 V	V_{LVD0} = 2.56 V
0:1	V_{LVW1} = 2.92 V	
1:0	V_{LVW2} = 4.3 V	V_{LVD1} = 4.0 V
1:1	V_{LVW3} = 4.6 V	

[1] See Appendix A, “Electrical Characteristics,” for minimum and maximum values.

Chapter 6
Parallel Input/Output

6.1 Introduction

This chapter explains software controls related to parallel input/output (I/O). The MC9S08JM60 has seven I/O ports which include a total of 51 general-purpose I/O pins. See Chapter 2, "Pins and Connections," for more information about the logic and hardware aspects of these pins.

Not all pins are available on all devices. See Table 2-1 to determine which functions are available for a specific device.

Many of the I/O pins are shared with on-chip peripheral functions, as shown in Table 2-1. The peripheral modules have priority over the I/Os, so when a peripheral is enabled, the I/O functions are disabled.

After reset, the shared peripheral functions are disabled so that the pins are controlled by the parallel I/O. All of the parallel I/O are configured as inputs (PTxDDn = 0). The pin control functions for each pin are configured as follows: slew rate control enabled (PTxSEn = 1), low drive strength selected (PTxDSn = 0), and internal pullups disabled (PTxPEn = 0).

NOTE

Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the user's reset initialization routine in the application program must either enable on-chip pullup devices or change the direction of unconnected pins to outputs so the pins do not float.

6.2 Port Data and Data Direction

Reading and writing of parallel I/O is done through the port data registers. The direction, input or output, is controlled through the port data direction registers. The parallel I/O port function for an individual pin is illustrated in the block diagram below.

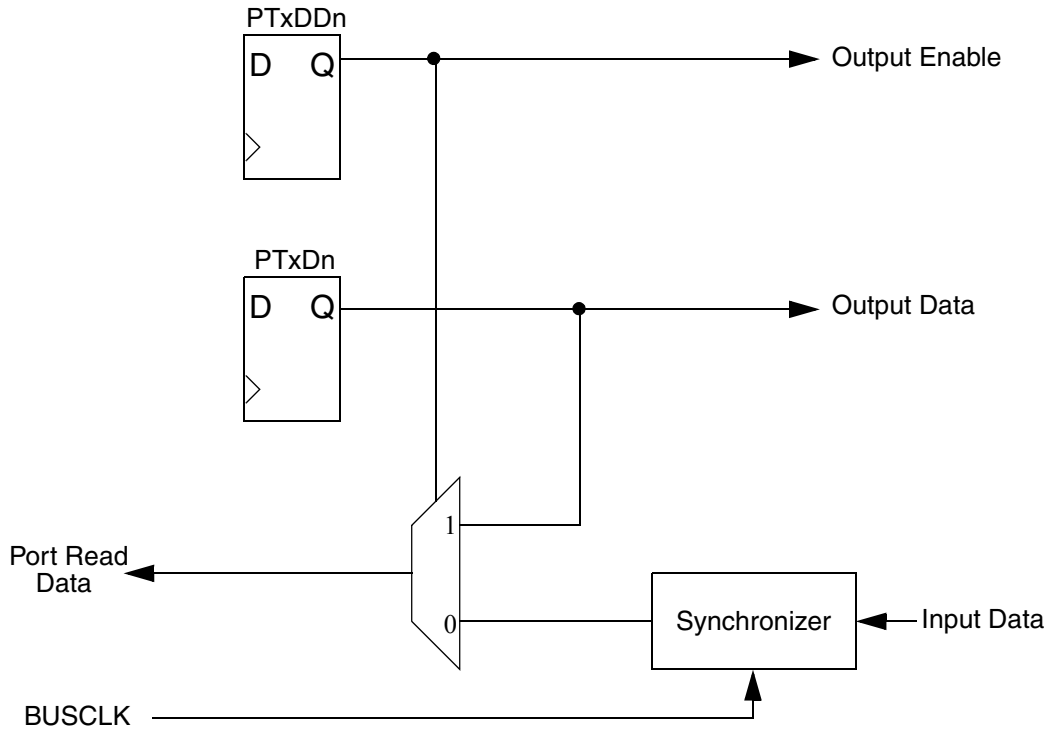


Figure 6-1. Parallel I/O Block Diagram

The data direction control bits determine whether the pin output driver is enabled, and they control what is read for port data register reads. Each port pin has a data direction register bit. When PTxDDn = 0, the corresponding pin is an input and reads of PTxD return the pin value. When PTxDDn = 1, the corresponding pin is an output and reads of PTxD return the last value written to the port data register. When a peripheral module or system function is in control of a port pin, the data direction register bit still controls what is returned for reads of the port data register, even though the peripheral system has overriding control of the actual pin direction.

When a shared analog function is enabled for a pin, all digital pin functions are disabled. A read of the port data register returns a value of 0 for any bits which have shared analog functions enabled. In general, whenever a pin is shared with both an alternate digital function and an analog function, the analog function has priority such that if both the digital and analog functions are enabled, the analog function controls the pin.

It is a good programming practice to write to the port data register before changing the direction of a port pin to become an output. This ensures that the pin will not be driven momentarily with an old data value that happened to be in the port data register.

6.3 Pin Control

The pin control registers are located in the high page register block of the memory. These registers are used to control pullups, slew rate, and drive strength for the I/O pins. The pin control registers operate independently of the parallel I/O registers.

6.3.1 Internal Pullup Enable

An internal pullup device can be enabled for each port pin by setting the corresponding bit in one of the pullup enable registers (PTxPEn). The pullup device is disabled if the pin is configured as an output by the parallel I/O control logic or any shared peripheral function regardless of the state of the corresponding pullup enable register bit. The pullup device is also disabled if the pin is controlled by an analog function.

6.3.2 Output Slew Rate Control Enable

Slew rate control can be enabled for each port pin by setting the corresponding bit in one of the slew rate control registers (PTxSEn). When enabled, slew control limits the rate at which an output can transition in order to reduce EMC emissions. Slew rate control has no effect on pins which are configured as inputs.

6.3.3 Output Drive Strength Select

An output pin can be selected to have high output drive strength by setting the corresponding bit in one of the drive strength select registers (PTxDSn). When high drive is selected a pin is capable of sourcing and sinking greater current. Even though every I/O pin can be selected as high drive, the user must ensure that the total current source and sink limits for the chip are not exceeded. Drive strength selection is intended to affect the DC behavior of I/O pins. However, the AC behavior is also affected. High drive allows a pin to drive a greater load with the same switching speed as a low drive enabled pin into a smaller load. Because of this the EMC emissions may be affected by enabling pins as high drive.

6.4 Pin Behavior in Stop Modes

Depending on the stop mode, I/O functions differently as the result of executing a STOP instruction. An explanation of I/O behavior for the various stop modes follows:

- Stop2 mode is a partial power-down mode, whereby I/O latches are maintained in their state as before the STOP instruction was executed. CPU register status and the state of I/O registers must be saved in RAM before the STOP instruction is executed to place the MCU in stop2 mode. Upon recovery from stop2 mode, before accessing any I/O, the user must examine the state of the PPDF bit in the SPMSC2 register. If the PPDF bit is 0, I/O must be initialized as if a power on reset had occurred. If the PPDF bit is 1, I/O data previously stored in RAM, before the STOP instruction was executed, peripherals may require being initialized and restored to their pre-stop condition. The user must then write a 1 to the PPDACK bit in the SPMSC2 register. Access to I/O is now permitted again in the user's application program.
- In stop3 mode, all I/O is maintained because internal logic circuity stays powered up. Upon recovery, normal I/O function is available to the user.

6.5 Parallel I/O and Pin Control Registers

This section provides information about the registers associated with the parallel I/O ports and pin control functions. These parallel I/O registers are located in page zero of the memory map and the pin control registers are located in the high page register section of memory.

Refer to tables in Chapter 4, "Memory," for the absolute address assignments for all parallel I/O and pin control registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

6.5.1 Port A I/O Registers (PTAD and PTADD)

Port A parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W			PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
Reset	0	0	0	0	0	0	0	0

Figure 6-2. Port A Data Register (PTAD)

Table 6-1. PTAD Register Field Descriptions

Field	Description
5:0 PTAD[5:0]	**Port A Data Register Bits** — For port A pins that are inputs, reads return the logic level on the pin. For port A pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port A pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTAD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W			PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
Reset	0	0	0	0	0	0	0	0

Figure 6-3. Data Direction for Port A Register (PTADD)

Table 6-2. PTADD Register Field Descriptions

Field	Description
5:0 PTADD[5:0]	**Data Direction for Port A Bits** — These read/write bits control the direction of port A pins and what is read for PTAD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port A bit n and PTAD reads return the contents of PTADn.

6.5.2 Port A Pin Control Registers (PTAPE, PTASE, PTADS)

In addition to the I/O control, port A pins are controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W			PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-4. Internal Pullup Enable for Port A (PTAPE)

Table 6-3. PTADD Register Field Descriptions

Field	Description
[5:0] PTAPE[5:0]	**Internal Pullup Enable for Port A Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTA pin. For port A pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port A bit n. 1 Internal pullup device enabled for port A bit n.

	7	6	5	4	3	2	1	0
R W			PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
Reset	0	0	1	1	1	1	1	1

Figure 6-5. Output Slew Rate Control Enable for Port A (PTASE)

Table 6-4. PTASE Register Field Descriptions

Field	Description
5:0 PTASE[5:0]	**Output Slew Rate Control Enable for Port A Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port A bit n. 1 Output slew rate control enabled for port A bit n.

	7	6	5	4	3	2	1	0
R W			PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
Reset	0	0	0	0	0	0	0	0

Figure 6-6. Output Drive Strength Selection for Port A (PTASE)

Table 6-5. PTASE Register Field Descriptions

Field	Description
5:0 PTADS[5:0]	**Output Drive Strength Selection for Port A Bits** — Each of these control bits selects between low and high output drive for the associated PTA pin. 0 Low output drive enabled for port A bit n. 1 High output drive enabled for port A bit n.

6.5.3 Port B I/O Registers (PTBD and PTBDD)

Port B parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
Reset	0	0	0	0	0	0	0	0

Figure 6-7. Port B Data Register (PTBD)

Table 6-6. PTBD Register Field Descriptions

Field	Description
7:0 PTBD[7:0]	**Port B Data Register Bits** — For port B pins that are inputs, reads return the logic level on the pin. For port B pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port B pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTBD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
Reset	0	0	0	0	0	0	0	0

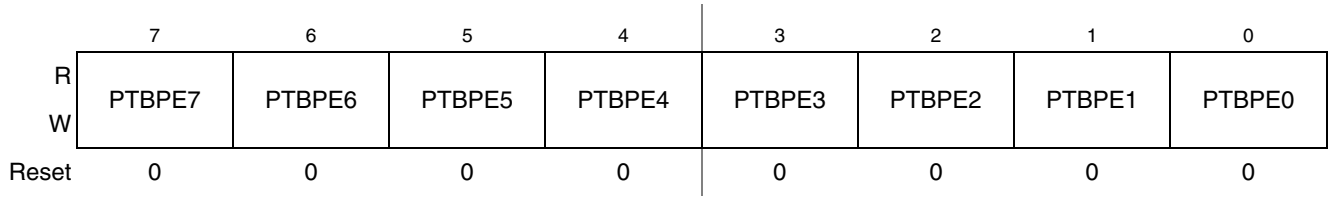
Figure 6-8. Data Direction for Port B (PTBDD)

Table 6-7. PTBDD Register Field Descriptions

Field	Description
7:0 PTBDD[7:0]	**Data Direction for Port B Bits** — These read/write bits control the direction of port B pins and what is read for PTBD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port B bit n and PTBD reads return the contents of PTBDn.

6.5.4 Port B Pin Control Registers (PTBPE, PTBSE, PTBDS)

In addition to the I/O control, port B pins are controlled by the registers listed below.

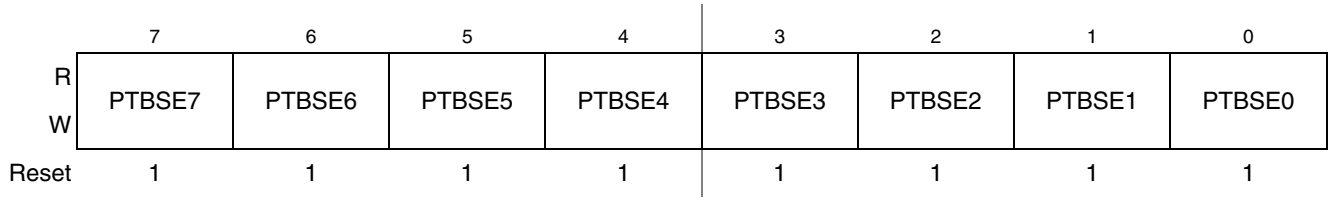


	7	6	5	4	3	2	1	0
R W	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-9. Internal Pullup Enable for Port B (PTBPE)

Table 6-8. PTBPE Register Field Descriptions

Field	Description
7:0 PTBPE[7:0]	**Internal Pullup Enable for Port B Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port B bit n. 1 Internal pullup device enabled for port B bit n.

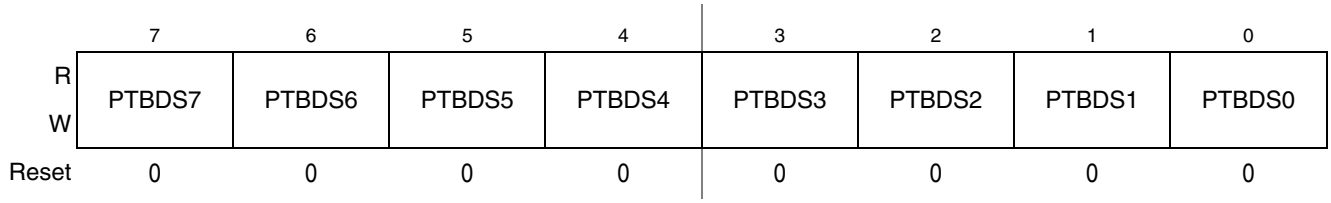


	7	6	5	4	3	2	1	0
R W	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
Reset	1	1	1	1	1	1	1	1

Figure 6-10. Output Slew Rate Control Enable (PTBSE)

Table 6-9. PTBSE Register Field Descriptions

Field	Description
7:0 PTBSE[7:0]	**Output Slew Rate Control Enable for Port B Bits**— Each of these control bits determine whether output slew rate control is enabled for the associated PTB pin. For port B pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port B bit n. 1 Output slew rate control enabled for port B bit n.



	7	6	5	4	3	2	1	0
R W	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
Reset	0	0	0	0	0	0	0	0

Figure 6-11. Output Drive Strength Selection for Port B (PTBDS)

Table 6-10. PTBDS Register Field Descriptions

Field	Description
7:0 PTBDS[7:0]	**Output Drive Strength Selection for Port B Bits** — Each of these control bits selects between low and high output drive for the associated PTB pin. 0 Low output drive enabled for port B bit n. 1 High output drive enabled for port B bit n.

6.5.5 Port C I/O Registers (PTCD and PTCDD)

Port C parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W		PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
Reset	0	0	0	0	0	0	0	0

Figure 6-12. Port C Data Register (PTCD)

Table 6-11. PTCD Register Field Descriptions

Field	Description
6:0 PTCD[6:0]	**Port C Data Register Bits** — For port C pins that are inputs, reads return the logic level on the pin. For port C pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port C pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTCD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W		PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
Reset	0	0	0	0	0	0	0	0

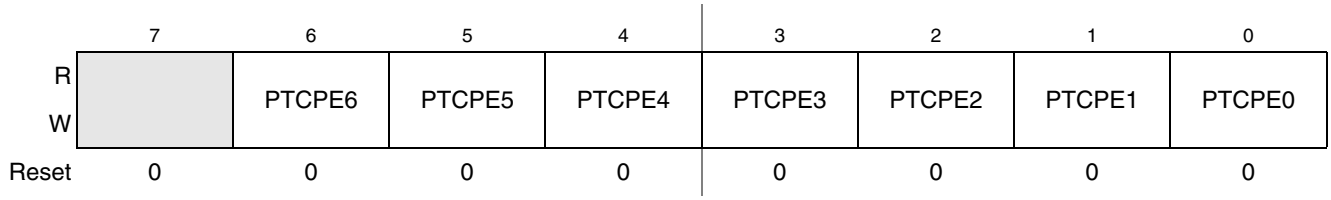
Figure 6-13. Data Direction for Port C (PTCDD)

Table 6-12. PTCDD Register Field Descriptions

Field	Description
6:0 PTCDD[6:0]	**Data Direction for Port C Bits** — These read/write bits control the direction of port C pins and what is read for PTCD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port C bit n and PTCD reads return the contents of PTCDn.

6.5.6 Port C Pin Control Registers (PTCPE, PTCSE, PTCDS)

In addition to the I/O control, port C pins are controlled by the registers listed below.

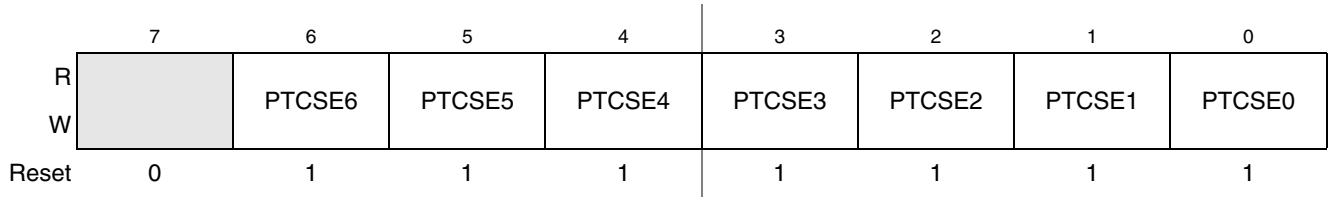


	7	6	5	4	3	2	1	0
R W		PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-14. Internal Pullup Enable for Port C (PTCPE)

Table 6-13. PTCPE Register Field Descriptions

Field	Description
6:0 PTCPE[6:0]	**Internal Pullup Enable for Port C Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTC pin. For port C pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port C bit n. 1 Internal pullup device enabled for port C bit n.

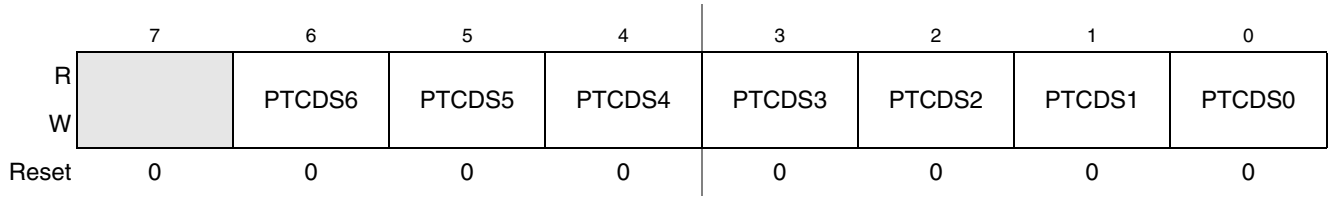


	7	6	5	4	3	2	1	0
R W		PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
Reset	0	1	1	1	1	1	1	1

Figure 6-15. Output Slew Rate Control Enable for Port C (PTCSE)

Table 6-14. PTCSE Register Field Descriptions

Field	Description
6:0 PTCSE[6:0]	**Output Slew Rate Control Enable for Port C Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTC pin. For port C pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port C bit n. 1 Output slew rate control enabled for port C bit n.



	7	6	5	4	3	2	1	0
R W		PTCDS6	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
Reset	0	0	0	0	0	0	0	0

Figure 6-16. Output Drive Strength Selection for Port C (PTCDS)

Table 6-15. PTCDS Register Field Descriptions

Field	Description
6:0 PTCDS[6:0]	**Output Drive Strength Selection for Port C Bits** — Each of these control bits selects between low and high output drive for the associated PTC pin. 0 Low output drive enabled for port C bit n. 1 High output drive enabled for port C bit n.

6.5.7 Port D I/O Registers (PTDD and PTDDD)

Port D parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0
Reset	0	0	0	0	0	0	0	0

Figure 6-17. Port D Data Register (PTDD)

Table 6-16. PTDD Register Field Descriptions

Field	Description
7:0 PTDD[7:0]	**Port D Data Register Bits** — For port D pins that are inputs, reads return the logic level on the pin. For port D pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port D pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTDD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0
Reset	0	0	0	0	0	0	0	0

Figure 6-18. Data Direction for Port D (PTDDD)

Table 6-17. PTDDD Register Field Descriptions

Field	Description
7:0 PTDDD[7:0]	**Data Direction for Port D Bits** — These read/write bits control the direction of port D pins and what is read for PTDD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port D bit n and PTDD reads return the contents of PTDDn.

6.5.8 Port D Pin Control Registers (PTDPE, PTDSE, PTDDS)

In addition to the I/O control, port D pins are controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-19. Internal Pullup Enable for Port D (PTDPE)

Table 6-18. PTDPE Register Field Descriptions

Field	Description
7:0 PTDPE[7:0]	**Internal Pullup Enable for Port D Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTD pin. For port D pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port D bit n. 1 Internal pullup device enabled for port D bit n.

	7	6	5	4	3	2	1	0
R W	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
Reset	1	1	1	1	1	1	1	1

Figure 6-20. Output Slew Rate Control Enable for Port D (PTDSE)

Table 6-19. PTDSE Register Field Descriptions

Field	Description
7:0 PTDSE[7:0]	**Output Slew Rate Control Enable for Port D Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTD pin. For port D pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port D bit n. 1 Output slew rate control enabled for port D bit n.

	7	6	5	4	3	2	1	0
R W	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
Reset	0	0	0	0	0	0	0	0

Figure 6-21. Output Drive Strength Selection for Port D (PTDDS)

Table 6-20. PTDDS Register Field Descriptions

Field	Description
7:0 PTDDS[7:0]	**Output Drive Strength Selection for Port D Bits** — Each of these control bits selects between low and high output drive for the associated PTD pin. 0 Low output drive enabled for port D bit n. 1 High output drive enabled for port D bit n.

6.5.9 Port E I/O Registers (PTED and PTEDD)

Port E parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
Reset	0	0	0	0	0	0	0	0

Figure 6-22. Port E Data Register (PTED)

Table 6-21. PTED Register Field Descriptions

Field	Description
7:0 PTED[7:0]	**Port E Data Register Bits** — For port E pins that are inputs, reads return the logic level on the pin. For port E pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port E pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTED to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
Reset	0	0	0	0	0	0	0	0

Figure 6-23. Data Direction for Port E (PTEDD)

Table 6-22. PTEDD Register Field Descriptions

Field	Description
7:0 PTEDD[7:0]	**Data Direction for Port E Bits** — These read/write bits control the direction of port E pins and what is read for PTED reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port E bit n and PTED reads return the contents of PTEDn.

6.5.10 Port E Pin Control Registers (PTEPE, PTESE, PTEDS)

In addition to the I/O control, port E pins are controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-24. Internal Pullup Enable for Port E (PTEPE)

Table 6-23. PTEPE Register Field Descriptions

Field	Description
7:0 PTEPE[7:0]	**Internal Pullup Enable for Port E Bits**— Each of these control bits determines if the internal pullup device is enabled for the associated PTE pin. For port E pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port E bit n. 1 Internal pullup device enabled for port E bit n.

	7	6	5	4	3	2	1	0
R W	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
Reset	1	1	1	1	1	1	1	1

Figure 6-25. Output Slew Rate Control Enable for Port E (PTESE)

Table 6-24. PTESE Register Field Descriptions

Field	Description
7:0 PTESE[7:0]	**Output Slew Rate Control Enable for Port E Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTE pin. For port E pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port E bit n. 1 Output slew rate control enabled for port E bit n.

	7	6	5	4	3	2	1	0
R W	PTEDS7	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
Reset	0	0	0	0	0	0	0	0

Figure 6-26. Output Drive Strength Selection for Port E (PTEDS)

Table 6-25. PTEDS Register Field Descriptions

Field	Description
7:0 PTEDS[7:0]	**Output Drive Strength Selection for Port E Bits** — Each of these control bits selects between low and high output drive for the associated PTE pin. 0 Low output drive enabled for port E bit n. 1 High output drive enabled for port E bit n.

6.5.11 Port F I/O Registers (PTFD and PTFDD)

Port F parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTFD7	PTFD6	PTFD5	PTFD4	PTFD3	PTFD2	PTFD1	PTFD0
Reset	0	0	0	0	0	0	0	0

Figure 6-27. Port F Data Register (PTFD)

Table 6-26. PTFD Register Field Descriptions

Field	Description
7:0 PTFD[7:0]	**Port F Data Register Bits**— For port F pins that are inputs, reads return the logic level on the pin. For port F pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port F pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTFD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W	PTFDD7	PTFDD6	PTFDD5	PTFDD4	PTFDD3	PTFDD2	PTFDD1	PTFDD0
Reset	0	0	0	0	0	0	0	0

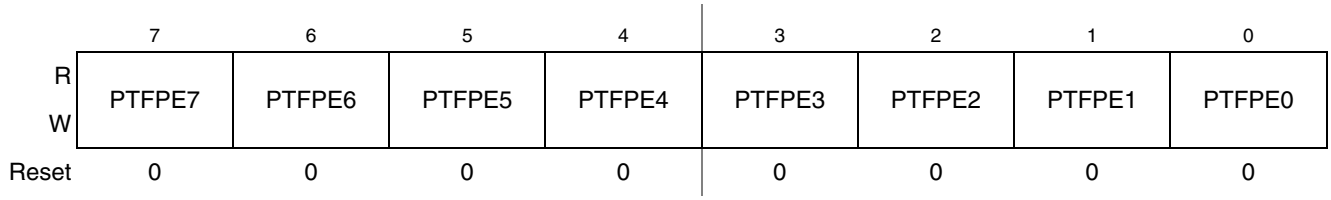
Figure 6-28. Data Direction for Port F (PTFDD)

Table 6-27. PTFDD Register Field Descriptions

Field	Description
7:0 PTFDD[7:0]	**Data Direction for Port F Bits** — These read/write bits control the direction of port F pins and what is read for PTFD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port F bit n and PTFD reads return the contents of PTFDn.

6.5.12 Port F Pin Control Registers (PTFPE, PTFSE, PTFDS)

In addition to the I/O control, port F pins are controlled by the registers listed below.

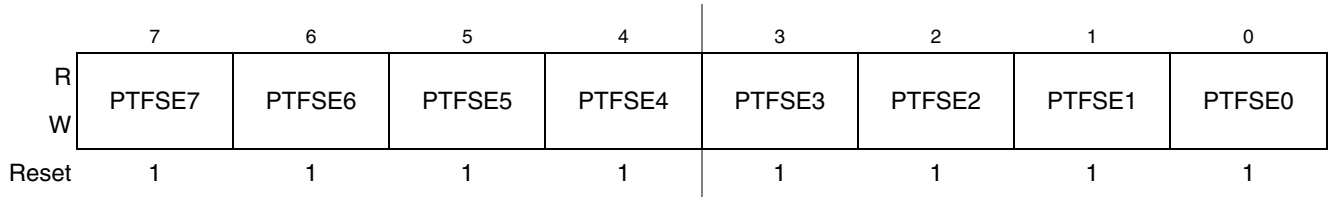


	7	6	5	4	3	2	1	0
R W	PTFPE7	PTFPE6	PTFPE5	PTFPE4	PTFPE3	PTFPE2	PTFPE1	PTFPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-29. Internal Pullup Enable for Port F (PTFPE)

Table 6-28. PTFPE Register Field Descriptions

Field	Description
7:0 PTFPE[7:0]	**Internal Pullup Enable for Port F Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTF pin. For port F pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port F bit n. 1 Internal pullup device enabled for port F bit n.

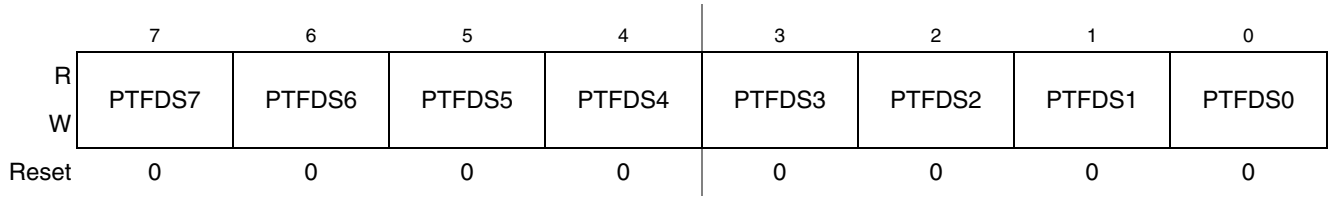


	7	6	5	4	3	2	1	0
R W	PTFSE7	PTFSE6	PTFSE5	PTFSE4	PTFSE3	PTFSE2	PTFSE1	PTFSE0
Reset	1	1	1	1	1	1	1	1

Figure 6-30. Output Slew Rate Control Enable for Port F (PTFSE)

Table 6-29. PTFSE Register Field Descriptions

Field	Description
7:0 PTFSE[7:0]	**Output Slew Rate Control Enable for Port F Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTF pin. For port F pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port F bit n. 1 Output slew rate control enabled for port F bit n.



	7	6	5	4	3	2	1	0
R W	PTFDS7	PTFDS6	PTFDS5	PTFDS4	PTFDS3	PTFDS2	PTFDS1	PTFDS0
Reset	0	0	0	0	0	0	0	0

Figure 6-31. Output Drive Strength Selection for Port F (PTFDS)

Table 6-30. PTFDS Register Field Descriptions

Field	Description
7:0 PTFDS[7:0]	**Output Drive Strength Selection for Port F Bits** — Each of these control bits selects between low and high output drive for the associated PTF pin. 0 Low output drive enabled for port F bit n. 1 High output drive enabled for port F bit n.

6.5.13 Port G I/O Registers (PTGD and PTGDD)

Port G parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W			PTGD5	PTGD4	PTGD3	PTGD2	PTGD1	PTGD0
Reset	0	0	0	0	0	0	0	0

Figure 6-32. Port G Data Register (PTGD)

Table 6-31. PTGD Register Field Descriptions

Field	Description
5:0 PTGD[5:0]	**Port G Data Register Bits** — For port G pins that are inputs, reads return the logic level on the pin. For port G pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port G pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTGD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W			PTGDD5	PTGDD4	PTGDD3	PTGDD2	PTGDD1	PTGDD0
Reset	0	0	0	0	0	0	0	0

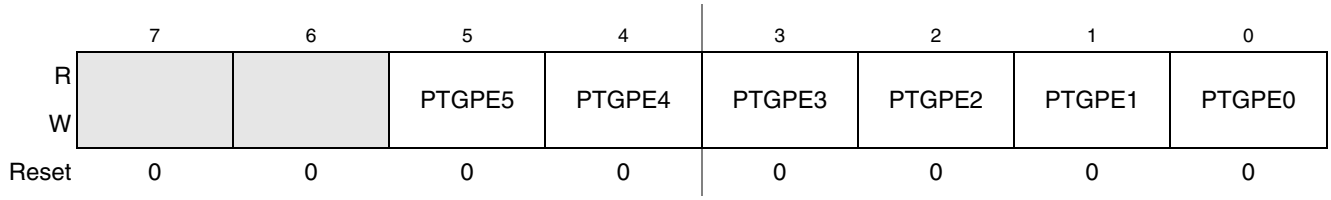
Figure 6-33. Data Direction for Port G (PTGDD)

Table 6-32. PTGDD Register Field Descriptions

Field	Description
5:0 PTGDD[5:0]	**Data Direction for Port G Bits** — These read/write bits control the direction of port G pins and what is read for PTGD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port G bit n and PTGD reads return the contents of PTGDn.

6.5.14 Port G Pin Control Registers (PTGPE, PTGSE, PTGDS)

In addition to the I/O control, port G pins are controlled by the registers listed below.

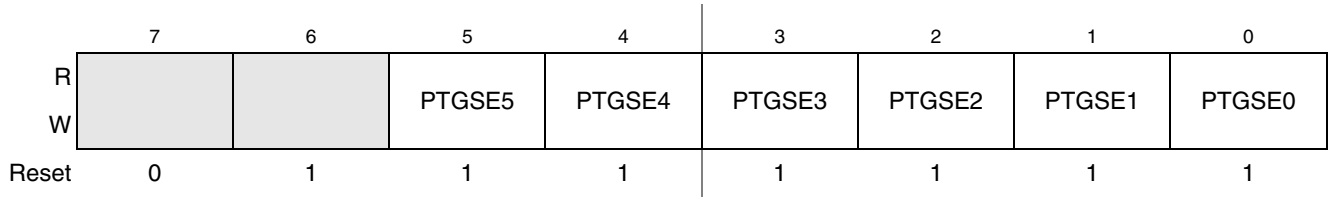


	7	6	5	4	3	2	1	0
R W			PTGPE5	PTGPE4	PTGPE3	PTGPE2	PTGPE1	PTGPE0
Reset	0	0	0	0	0	0	0	0

Figure 6-34. Internal Pullup Enable for Port G Bits (PTGPE)

Table 6-33. PTGPE Register Field Descriptions

Field	Description
5:0 PTGPEn	**Internal Pullup Enable for Port G Bits** — Each of these control bits determines if the internal pullup device is enabled for the associated PTG pin. For port G pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port G bit n. 1 Internal pullup device enabled for port G bit n.

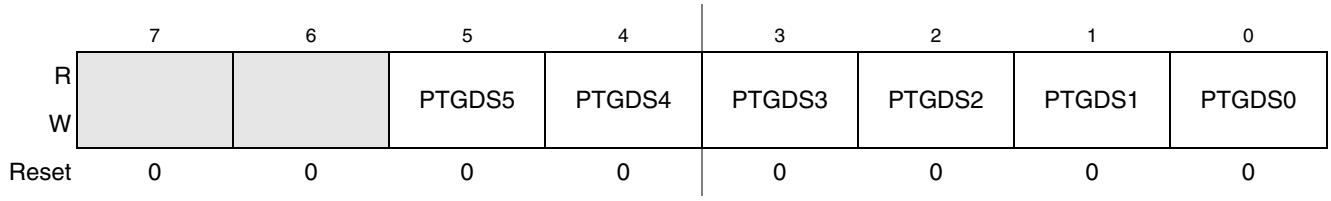


	7	6	5	4	3	2	1	0
R W			PTGSE5	PTGSE4	PTGSE3	PTGSE2	PTGSE1	PTGSE0
Reset	0	1	1	1	1	1	1	1

Figure 6-35. Output Slew Rate Control Enable for Port G Bits (PTGSE)

Table 6-34. PTGSE Register Field Descriptions

Field	Description
5:0 PTGSEn	**Output Slew Rate Control Enable for Port G Bits**— Each of these control bits determine whether output slew rate control is enabled for the associated PTG pin. For port G pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port G bit n. 1 Output slew rate control enabled for port G bit n.



	7	6	5	4	3	2	1	0
R W			PTGDS5	PTGDS4	PTGDS3	PTGDS2	PTGDS1	PTGDS0
Reset	0	0	0	0	0	0	0	0

Figure 6-36. Output Drive Strength Selection for Port G (PTGDS)

Table 6-35. PTGDS Register Field Descriptions

Field	Description
5:0 PTGDSn	**Output Drive Strength Selection for Port G Bits** — Each of these control bits selects between low and high output drive for the associated PTG pin. 0 Low output drive enabled for port G bit n. 1 High output drive enabled for port G bit n.

Chapter 7
Central Processor Unit (S08CPUV2)

7.1 Introduction

This section provides summary information about the registers, addressing modes, and instruction set of the CPU of the HCS08 Family. For a more detailed discussion, refer to the *HCS08 Family Reference Manual, volume 1,* Freescale Semiconductor document order number HCS08RMV1/D.

The HCS08 CPU is fully source- and object-code-compatible with the M68HC08 CPU. Several instructions and enhanced addressing modes were added to improve C compiler efficiency and to support a new background debug system which replaces the monitor mode of earlier M68HC08 microcontrollers (MCU).

7.1.1 Features

Features of the HCS08 CPU include:

- Object code fully upward-compatible with M68HC05 and M68HC08 Families
- All registers and memory are mapped to a single 64-Kbyte address space
- 16-bit stack pointer (any size stack anywhere in 64-Kbyte address space)
- 16-bit index register (H:X) with powerful indexed addressing modes
- 8-bit accumulator (A)
- Many instructions treat X as a second general-purpose 8-bit register
- Seven addressing modes:
  - Inherent — Operands in internal registers
  - Relative — 8-bit signed offset to branch destination
  - Immediate — Operand in next object code byte(s)
  - Direct — Operand in memory at 0x0000–0x00FF
  - Extended — Operand anywhere in 64-Kbyte address space
  - Indexed relative to H:X — Five submodes including auto increment
  - Indexed relative to SP — Improves C efficiency dramatically
- Memory-to-memory data move instructions with four address mode combinations
- Overflow, half-carry, negative, zero, and carry condition codes support conditional branching on the results of signed, unsigned, and binary-coded decimal (BCD) operations
- Efficient bit manipulation instructions
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- STOP and WAIT instructions to invoke low-power operating modes

7.2 Programmer's Model and CPU Registers

Figure 7-1 shows the five CPU registers. CPU registers are not part of the memory map.

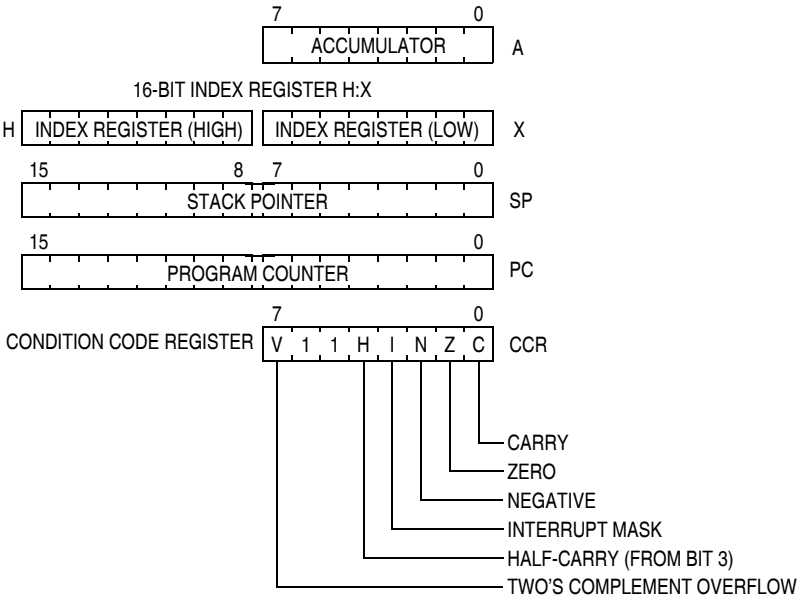


Figure 7-1. CPU Registers

7.2.1 Accumulator (A)

The A accumulator is a general-purpose 8-bit register. One operand input to the arithmetic logic unit (ALU) is connected to the accumulator and the ALU results are often stored into the A accumulator after arithmetic and logical operations. The accumulator can be loaded from memory using various addressing modes to specify the address where the loaded data comes from, or the contents of A can be stored to memory using various addressing modes to specify the address where data from A will be stored.

Reset has no effect on the contents of the A accumulator.

7.2.2 Index Register (H:X)

This 16-bit register is actually two separate 8-bit registers (H and X), which often work together as a 16-bit address pointer where H holds the upper byte of an address and X holds the lower byte of the address. All indexed addressing mode instructions use the full 16-bit value in H:X as an index reference pointer; however, for compatibility with the earlier M68HC05 Family, some instructions operate only on the low-order 8-bit half (X).

Many instructions treat X as a second general-purpose 8-bit register that can be used to hold 8-bit data values. X can be cleared, incremented, decremented, complemented, negated, shifted, or rotated. Transfer instructions allow data to be transferred from A or transferred to A where arithmetic and logical operations can then be performed.

For compatibility with the earlier M68HC05 Family, H is forced to 0x00 during reset. Reset has no effect on the contents of X.

7.2.3 Stack Pointer (SP)

This 16-bit address pointer register points at the next available location on the automatic last-in-first-out (LIFO) stack. The stack may be located anywhere in the 64-Kbyte address space that has RAM and can be any size up to the amount of available RAM. The stack is used to automatically save the return address for subroutine calls, the return address and CPU registers during interrupts, and for local variables. The AIS (add immediate to stack pointer) instruction adds an 8-bit signed immediate value to SP. This is most often used to allocate or deallocate space for local variables on the stack.

SP is forced to 0x00FF at reset for compatibility with the earlier M68HC05 Family. HCS08 programs normally change the value in SP to the address of the last location (highest address) in on-chip RAM during reset initialization to free up direct page RAM (from the end of the on-chip registers to 0x00FF).

The RSP (reset stack pointer) instruction was included for compatibility with the M68HC05 Family and is seldom used in new HCS08 programs because it only affects the low-order half of the stack pointer.

7.2.4 Program Counter (PC)

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched.

During normal program execution, the program counter automatically increments to the next sequential memory location every time an instruction or operand is fetched. Jump, branch, interrupt, and return operations load the program counter with an address other than that of the next sequential location. This is called a change-of-flow.

During reset, the program counter is loaded with the reset vector that is located at 0xFFFE and 0xFFFF. The vector stored there is the address of the first instruction that will be executed after exiting the reset state.

7.2.5 Condition Code Register (CCR)

The 8-bit condition code register contains the interrupt mask (I) and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to 1. The following paragraphs describe the functions of the condition code bits in general terms. For a more detailed explanation of how each instruction sets the CCR bits, refer to the *HCS08 Family Reference Manual, volume 1,* Freescale Semiconductor document order number HCS08RMv1.

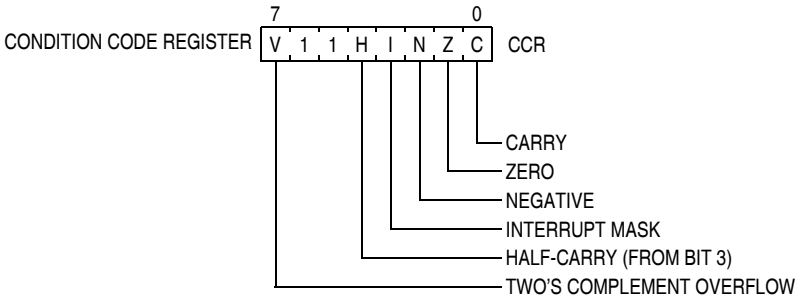


Figure 7-2. Condition Code Register

Table 7-1. CCR Register Field Descriptions

Field	Description
7 V	**Two's Complement Overflow Flag** — The CPU sets the overflow flag when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow flag. 0 No overflow 1 Overflow
4 H	**Half-Carry Flag** — The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an add-without-carry (ADD) or add-with-carry (ADC) operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C condition code bits to automatically add a correction value to the result from a previous ADD or ADC on BCD operands to correct the result to a valid BCD value. 0 No carry between bits 3 and 4 1 Carry between bits 3 and 4
3 I	**Interrupt Mask Bit** — When the interrupt mask is set, all maskable CPU interrupts are disabled. CPU interrupts are enabled when the interrupt mask is cleared. When a CPU interrupt occurs, the interrupt mask is set automatically after the CPU registers are saved on the stack, but before the first instruction of the interrupt service routine is executed. Interrupts are not recognized at the instruction boundary after any instruction that clears I (CLI or TAP). This ensures that the next instruction after a CLI or TAP will always be executed without the possibility of an intervening interrupt, provided I was set. 0 Interrupts enabled 1 Interrupts disabled
2 N	**Negative Flag** — The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result. Simply loading or storing an 8-bit or 16-bit value causes N to be set if the most significant bit of the loaded or stored value was 1. 0 Non-negative result 1 Negative result
1 Z	**Zero Flag** — The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of 0x00 or 0x0000. Simply loading or storing an 8-bit or 16-bit value causes Z to be set if the loaded or stored value was all 0s. 0 Non-zero result 1 Zero result
0 C	**Carry/Borrow Flag** — The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions — such as bit test and branch, shift, and rotate — also clear or set the carry/borrow flag. 0 No carry out of bit 7 1 Carry out of bit 7

7.3 Addressing Modes

Addressing modes define the way the CPU accesses operands and data. In the HCS08, all memory, status and control registers, and input/output (I/O) ports share a single 64-Kbyte linear address space so a 16-bit binary address can uniquely identify any memory location. This arrangement means that the same instructions that access variables in RAM can also be used to access I/O and control registers or nonvolatile program space.

Some instructions use more than one addressing mode. For instance, move instructions use one addressing mode to specify the source operand and a second addressing mode to specify the destination address. Instructions such as BRCLR, BRSET, CBEQ, and DBNZ use one addressing mode to specify the location of an operand for a test and then use relative addressing mode to specify the branch destination address when the tested condition is true. For BRCLR, BRSET, CBEQ, and DBNZ, the addressing mode listed in the instruction set tables is the addressing mode needed to access the operand to be tested, and relative addressing mode is implied for the branch destination.

7.3.1 Inherent Addressing Mode (INH)

In this addressing mode, operands needed to complete the instruction (if any) are located within CPU registers so the CPU does not need to access memory to get any operands.

7.3.2 Relative Addressing Mode (REL)

Relative addressing mode is used to specify the destination location for branch instructions. A signed 8-bit offset value is located in the memory location immediately following the opcode. During execution, if the branch condition is true, the signed offset is sign-extended to a 16-bit value and is added to the current contents of the program counter, which causes program execution to continue at the branch destination address.

7.3.3 Immediate Addressing Mode (IMM)

In immediate addressing mode, the operand needed to complete the instruction is included in the object code immediately following the instruction opcode in memory. In the case of a 16-bit immediate operand, the high-order byte is located in the next memory location after the opcode, and the low-order byte is located in the next memory location after that.

7.3.4 Direct Addressing Mode (DIR)

In direct addressing mode, the instruction includes the low-order eight bits of an address in the direct page (0x0000–0x00FF). During execution a 16-bit address is formed by concatenating an implied 0x00 for the high-order half of the address and the direct address from the instruction to get the 16-bit address where the desired operand is located. This is faster and more memory efficient than specifying a complete 16-bit address for the operand.

7.3.5 Extended Addressing Mode (EXT)

In extended addressing mode, the full 16-bit address of the operand is located in the next two bytes of program memory after the opcode (high byte first).

7.3.6 Indexed Addressing Mode

Indexed addressing mode has seven variations including five that use the 16-bit H:X index register pair and two that use the stack pointer as the base reference.

7.3.6.1 Indexed, No Offset (IX)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction.

7.3.6.2 Indexed, No Offset with Post Increment (IX+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction. The index register pair is then incremented (H:X = H:X + 0x0001) after the operand has been fetched. This addressing mode is only used for MOV and CBEQ instructions.

7.3.6.3 Indexed, 8-Bit Offset (IX1)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.3.6.4 Indexed, 8-Bit Offset with Post Increment (IX1+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction. The index register pair is then incremented (H:X = H:X + 0x0001) after the operand has been fetched. This addressing mode is used only for the CBEQ instruction.

7.3.6.5 Indexed, 16-Bit Offset (IX2)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.3.6.6 SP-Relative, 8-Bit Offset (SP1)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.3.6.7 SP-Relative, 16-Bit Offset (SP2)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.4 Special Operations

The CPU performs a few special operations that are similar to instructions but do not have opcodes like other CPU instructions. In addition, a few instructions such as STOP and WAIT directly affect other MCU circuitry. This section provides additional information about these operations.

7.4.1 Reset Sequence

Reset can be caused by a power-on-reset (POR) event, internal conditions such as the COP (computer operating properly) watchdog, or by assertion of an external active-low reset pin. When a reset event occurs, the CPU immediately stops whatever it is doing (the MCU does not wait for an instruction boundary before responding to a reset event). For a more detailed discussion about how the MCU recognizes resets and determines the source, refer to the Resets, Interrupts, and System Configuration chapter.

The reset event is considered concluded when the sequence to determine whether the reset came from an internal source is done and when the reset pin is no longer asserted. At the conclusion of a reset event, the CPU performs a 6-cycle sequence to fetch the reset vector from 0xFFFE and 0xFFFF and to fill the instruction queue in preparation for execution of the first program instruction.

7.4.2 Interrupt Sequence

When an interrupt is requested, the CPU completes the current instruction before responding to the interrupt. At this point, the program counter is pointing at the start of the next instruction, which is where the CPU should return after servicing the interrupt. The CPU responds to an interrupt by performing the same sequence of operations as for a software interrupt (SWI) instruction, except the address used for the vector fetch is determined by the highest priority interrupt that is pending when the interrupt sequence started.

The CPU sequence for an interrupt is:

1. Store the contents of PCL, PCH, X, A, and CCR on the stack, in that order.
2. Set the I bit in the CCR.
3. Fetch the high-order half of the interrupt vector.
4. Fetch the low-order half of the interrupt vector.
5. Delay for one free bus cycle.
6. Fetch three bytes of program information starting at the address indicated by the interrupt vector to fill the instruction queue in preparation for execution of the first instruction in the interrupt service routine.

After the CCR contents are pushed onto the stack, the I bit in the CCR is set to prevent other interrupts while in the interrupt service routine. Although it is possible to clear the I bit with an instruction in the

interrupt service routine, this would allow nesting of interrupts (which is not recommended because it leads to programs that are difficult to debug and maintain).

For compatibility with the earlier M68HC05 MCUs, the high-order half of the H:X index register pair (H) is not saved on the stack as part of the interrupt sequence. The user must use a PSHH instruction at the beginning of the service routine to save H and then use a PULH instruction just before the RTI that ends the interrupt service routine. It is not necessary to save H if you are certain that the interrupt service routine does not use any instructions or auto-increment addressing modes that might change the value of H.

The software interrupt (SWI) instruction is like a hardware interrupt except that it is not masked by the global I bit in the CCR and it is associated with an instruction opcode within the program so it is not asynchronous to program execution.

7.4.3 Wait Mode Operation

The WAIT instruction enables interrupts by clearing the I bit in the CCR. It then halts the clocks to the CPU to reduce overall power consumption while the CPU is waiting for the interrupt or reset event that will wake the CPU from wait mode. When an interrupt or reset event occurs, the CPU clocks will resume and the interrupt or reset event will be processed normally.

If a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in wait mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in wait mode.

7.4.4 Stop Mode Operation

Usually, all system clocks, including the crystal oscillator (when used), are halted during stop mode to minimize power consumption. In such systems, external circuitry is needed to control the time spent in stop mode and to issue a signal to wake up the target MCU when it is time to resume processing. Unlike the earlier M68HC05 and M68HC08 MCUs, the HCS08 can be configured to keep a minimum set of clocks running in stop mode. This optionally allows an internal periodic signal to wake the target MCU from stop mode.

When a host debug system is connected to the background debug pin (BKGD) and the ENBDM control bit has been set by a serial command through the background interface (or because the MCU was reset into active background mode), the oscillator is forced to remain active when the MCU enters stop mode. In this case, if a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in stop mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in stop mode.

Recovery from stop mode depends on the particular HCS08 and whether the oscillator was stopped in stop mode. Refer to the Modes of Operation chapter for more details.

7.4.5 BGND Instruction

The BGND instruction is new to the HCS08 compared to the M68HC08. BGND would not be used in normal user programs because it forces the CPU to stop processing user instructions and enter the active background mode. The only way to resume execution of the user program is through reset or by a host debug system issuing a GO, TRACE1, or TAGGO serial command through the background debug interface.

Software-based breakpoints can be set by replacing an opcode at the desired breakpoint address with the BGND opcode. When the program reaches this breakpoint address, the CPU is forced to active background mode rather than continuing the user program.

7.5 HCS08 Instruction Set Summary

Table 7-2 provides a summary of the HCS08 instruction set in all possible addressing modes. The table shows operand construction, execution time in internal bus clock cycles, and cycle-by-cycle details for each addressing mode variation of each instruction.

Table 7-2. . Instruction Set Summary (Sheet 1 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						VH	I N Z C
ADC #*opr8i* ADC *opr8a* ADC *opr16a* ADC *oprx16*,X ADC *oprx8*,X ADC ,X ADC *oprx16*,SP ADC *oprx8*,SP	Add with Carry A ← (A) + (M) + (C)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 ii B9 dd C9 hh ll D9 ee ff E9 ff F9 9E D9 ee ff 9E E9 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↕↕	– ↕ ↕ ↕

ADD #*opr8i*
ADD *opr8a*
ADD *opr16a*
ADD *oprx16*,X
ADD *oprx8*,X
ADD ,X
ADD *oprx16*,SP
ADD *oprx8*,SP
	Add without Carry
	A ← (A) + (M)
		IMM
		DIR
		EXT
		IX2
		IX1
		IX
		SP2
		SP1
			AB ii
			BB dd
			CB hh ll
			DB ee ff
			EB ff
			FB
			9E DB ee ff
			9E EB ff
				2
				3
				4
				4
				3
				3
				5
				4
					pp
					rpp
					prpp
					prpp
					rpp
					rfp
					pprpp
					prpp
						↕↕	– ↕ ↕ ↕

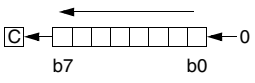
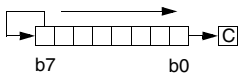
AIS #*opr8i*	Add Immediate Value (Signed) to Stack Pointer SP ← (SP) + (M)	IMM	A7 ii	2	pp	– –	– – – –
AIX #*opr8i*	Add Immediate Value (Signed) to Index Register (H:X) H:X ← (H:X) + (M)	IMM	AF ii	2	pp	– –	– – – –
AND #*opr8i* AND *opr8a* AND *opr16a* AND *oprx16*,X AND *oprx8*,X AND ,X AND *oprx16*,SP AND *oprx8*,SP	Logical AND A ← (A) & (M)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 ii B4 dd C4 hh ll D4 ee ff E4 ff F4 9E D4 ee ff 9E E4 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 –	– ↕ ↕ –
ASL *opr8a* ASLA ASLX ASL *oprx8*,X ASL ,X ASL *oprx8*,SP	Arithmetic Shift Left  (Same as LSL)	DIR INH INH IX1 IX SP1	38 dd 48 58 68 ff 78 9E 68 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ –	– ↕ ↕ ↕
ASR *opr8a* ASRA ASRX ASR *oprx8*,X ASR ,X ASR *oprx8*,SP	Arithmetic Shift Right  	DIR INH INH IX1 IX SP1	37 dd 47 57 67 ff 77 9E 67 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ –	– ↕ ↕ ↕
BCC *rel*	Branch if Carry Bit Clear (if C = 0)	REL	24 rr	3	ppp	– –	– – – –

Table 7-2. . Instruction Set Summary (Sheet 2 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						VH	I N Z C
BCLR *n,opr8a*	Clear Bit n in Memory (Mn ← 0)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 dd 13 dd 15 dd 17 dd 19 dd 1B dd 1D dd 1F dd	5 5 5 5 5 5 5 5	rfwpp rfwpp rfwpp rfwpp rfwpp rfwpp rfwpp rfwpp	– –	– – – –
BCS *rel*	Branch if Carry Bit Set (if C = 1) (Same as BLO)	REL	25 rr	3	ppp	– –	– – – –
BEQ *rel*	Branch if Equal (if Z = 1)	REL	27 rr	3	ppp	– –	– – – –
BGE *rel*	Branch if Greater Than or Equal To (if N ⊕ V = 0) (Signed)	REL	90 rr	3	ppp	– –	– – – –
BGND	Enter active background if ENBDM=1 Waits for and processes BDM commands until GO, TRACE1, or TAGGO	INH	82	5+	fp...ppp	– –	– – – –
BGT *rel*	Branch if Greater Than (if Z | (N ⊕ V) = 0) (Signed)	REL	92 rr	3	ppp	– –	– – – –
BHCC *rel*	Branch if Half Carry Bit Clear (if H = 0)	REL	28 rr	3	ppp	– –	– – – –
BHCS *rel*	Branch if Half Carry Bit Set (if H = 1)	REL	29 rr	3	ppp	– –	– – – –
BHI *rel*	Branch if Higher (if C | Z = 0)	REL	22 rr	3	ppp	– –	– – – –
BHS *rel*	Branch if Higher or Same (if C = 0) (Same as BCC)	REL	24 rr	3	ppp	– –	– – – –
BIH *rel*	Branch if IRQ Pin High (if IRQ pin = 1)	REL	2F rr	3	ppp	– –	– – – –
BIL *rel*	Branch if IRQ Pin Low (if IRQ pin = 0)	REL	2E rr	3	ppp	– –	– – – –
BIT #*opr8i* BIT *opr8a* BIT *opr16a* BIT *oprx16*,X BIT *oprx8*,X BIT ,X BIT *oprx16*,SP BIT *oprx8*,SP	Bit Test (A) & (M) (CCR Updated but Operands Not Changed)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 ii B5 dd C5 hh ll D5 ee ff E5 ff F5 9E D5 ee ff 9E E5 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 –	– ↕ ↕ –
BLE *rel*	Branch if Less Than or Equal To (if Z | (N ⊕ V) = 1) (Signed)	REL	93 rr	3	ppp	– –	– – – –
BLO *rel*	Branch if Lower (if C = 1) (Same as BCS)	REL	25 rr	3	ppp	– –	– – – –
BLS *rel*	Branch if Lower or Same (if C | Z = 1)	REL	23 rr	3	ppp	– –	– – – –
BLT *rel*	Branch if Less Than (if N ⊕ V = 1) (Signed)	REL	91 rr	3	ppp	– –	– – – –
BMC *rel*	Branch if Interrupt Mask Clear (if I = 0)	REL	2C rr	3	ppp	– –	– – – –
BMI *rel*	Branch if Minus (if N = 1)	REL	2B rr	3	ppp	– –	– – – –
BMS *rel*	Branch if Interrupt Mask Set (if I = 1)	REL	2D rr	3	ppp	– –	– – – –
BNE *rel*	Branch if Not Equal (if Z = 0)	REL	26 rr	3	ppp	– –	– – – –
BPL *rel*	Branch if Plus (if N = 0)	REL	2A rr	3	ppp	– –	– – – –

Table 7-2. . Instruction Set Summary (Sheet 3 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						VH	I N Z C
BRA *rel*	Branch Always (if I = 1)	REL	20 rr	3	ppp	– –	– – – –
BRCLR *n,oprx8,rel*	Branch if Bit *n* in Memory Clear (if (Mn) = 0)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 dd rr 03 dd rr 05 dd rr 07 dd rr 09 dd rr 0B dd rr 0D dd rr 0F dd rr	5 5 5 5 5 5 5 5	rpppp rpppp rpppp rpppp rpppp rpppp rpppp rpppp	– –	– – – ↕
BRN *rel*	Branch Never (if I = 0)	REL	21 rr	3	ppp	– –	– – – –
BRSET *n,oprx8,rel*	Branch if Bit *n* in Memory Set (if (Mn) = 1)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 dd rr 02 dd rr 04 dd rr 06 dd rr 08 dd rr 0A dd rr 0C dd rr 0E dd rr	5 5 5 5 5 5 5 5	rpppp rpppp rpppp rpppp rpppp rpppp rpppp rpppp	– –	– – – ↕
BSET *n,oprx8*	Set Bit *n* in Memory (Mn ← 1)	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 dd 12 dd 14 dd 16 dd 18 dd 1A dd 1C dd 1E dd	5 5 5 5 5 5 5 5	rfwpp rfwpp rfwpp rfwpp rfwpp rfwpp rfwpp rfwpp	– –	– – – –
BSR *rel*	Branch to Subroutine PC ← (PC) + $0002 push (PCL); SP ← (SP) – $0001 push (PCH); SP ← (SP) – $0001 PC ← (PC) + *rel*	REL	AD rr	5	ssppp	– –	– – – –
CBEQ *oprx8,rel* CBEQA #*opr8i,rel* CBEQX #*opr8i,rel* CBEQ *oprx8*,X+,*rel* CBEQ ,X+,*rel* CBEQ *oprx8*,SP,*rel*	Compare and... Branch if (A) = (M) Branch if (A) = (M) Branch if (X) = (M) Branch if (A) = (M) Branch if (A) = (M) Branch if (A) = (M)	DIR IMM IMM IX1+ IX+ SP1	31 dd rr 41 ii rr 51 ii rr 61 ff rr 71 rr 9E 61 ff rr	5 4 4 5 5 6	rpppp pppp pppp rpppp rfppp prpppp	– –	– – – –
CLC	Clear Carry Bit (C ← 0)	INH	98	1	p	– –	– – – 0
CLI	Clear Interrupt Mask Bit (I ← 0)	INH	9A	1	p	– –	0 – – –
CLR *oprx8* CLRA CLRX CLRH CLR *oprx8*,X CLR ,X CLR *oprx8*,SP	Clear M ← $00 A ← $00 X ← $00 H ← $00 M ← $00 M ← $00 M ← $00	DIR INH INH INH IX1 IX SP1	3F dd 4F 5F 8C 6F ff 7F 9E 6F ff	5 1 1 1 5 4 6	rfwpp p p p rfwpp rfwp prfwpp	0 –	– 0 1 –

Table 7-2. . Instruction Set Summary (Sheet 4 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						VH	I N Z C

CMP #*opr8i*
CMP *opr8a*
CMP *opr16a*
CMP *oprx16*,X
CMP *oprx8*,X
CMP ,X
CMP *oprx16*,SP
CMP *oprx8*,SP
	Compare Accumulator with Memory
	A – M
	(CCR Updated But Operands Not Changed)
		IMM
		DIR
		EXT
		IX2
		IX1
		IX
		SP2
		SP1
			A1 ii
			B1 dd
			C1 hh ll
			D1 ee ff
			E1 ff
			F1
			9E D1 ee ff
			9E E1 ff
				2
				3
				4
				4
				3
				3
				5
				4
					pp
					rpp
					prpp
					prpp
					rpp
					rfp
					pprpp
					prpp
						↕ –	– ↕ ↕ ↕

COM *opr8a* COMA COMX COM *oprx8*,X COM ,X COM *oprx8*,SP	Complement M ← ($\overline{M}$)= $FF – (M) (One's Complement) A ← ($\overline{A}$) = $FF – (A) X ← ($\overline{X}$) = $FF – (X) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M)	DIR INH INH IX1 IX SP1	33 dd 43 53 63 ff 73 9E 63 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	0 –	– ↕ ↕ 1
CPHX *opr16a* CPHX #*opr16i* CPHX *opr8a* CPHX *oprx8*,SP	Compare Index Register (H:X) with Memory (H:X) – (M:M + $0001) (CCR Updated But Operands Not Changed)	EXT IMM DIR SP1	3E hh ll 65 jj kk 75 dd 9E F3 ff	6 3 5 6	prrfpp ppp rrfpp prrfpp	↕ –	– ↕ ↕ ↕
CPX #*opr8i* CPX *opr8a* CPX *opr16a* CPX *oprx16*,X CPX *oprx8*,X CPX ,X CPX *oprx16*,SP CPX *oprx8*,SP	Compare X (Index Register Low) with Memory X – M (CCR Updated But Operands Not Changed)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A3 ii B3 dd C3 hh ll D3 ee ff E3 ff F3 9E D3 ee ff 9E E3 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↕ –	– ↕ ↕ ↕
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	INH	72	1	p	U –	– ↕ ↕ ↕
DBNZ *opr8a*,*rel* DBNZA *rel* DBNZX *rel* DBNZ *oprx8*,X,*rel* DBNZ ,X,*rel* DBNZ *oprx8*,SP,*rel*	Decrement A, X, or M and Branch if Not Zero (if (result) ≠ 0) DBNZX Affects X Not H	DIR INH INH IX1 IX SP1	3B dd rr 4B rr 5B rr 6B ff rr 7B rr 9E 6B ff rr	7 4 4 7 6 8	rfwpppp fppp fppp rfwpppp rfwppp prfwpppp	– –	– – – –
DEC *opr8a* DECA DECX DEC *oprx8*,X DEC ,X DEC *oprx8*,SP	Decrement M ← (M) – $01 A ← (A) – $01 X ← (X) – $01 M ← (M) – $01 M ← (M) – $01 M ← (M) – $01	DIR INH INH IX1 IX SP1	3A dd 4A 5A 6A ff 7A 9E 6A ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ –	– ↕ ↕ –
DIV	Divide A ← (H:A)÷(X); H ← Remainder	INH	52	6	ffffp	– –	– – ↕ ↕
EOR #*opr8i* EOR *opr8a* EOR *opr16a* EOR *oprx16*,X EOR *oprx8*,X EOR ,X EOR *oprx16*,SP EOR *oprx8*,SP	Exclusive OR Memory with Accumulator A ← (A ⊕ M)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A8 ii B8 dd C8 hh ll D8 ee ff E8 ff F8 9E D8 ee ff 9E E8 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 –	– ↕ ↕ –

Table 7-2. . Instruction Set Summary (Sheet 5 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						VH	I N Z C
INC *opr8a* INCA INCX INC *oprx8*,X INC ,X INC *oprx8*,SP	Increment M ← (M) + $01 A ← (A) + $01 X ← (X) + $01 M ← (M) + $01 M ← (M) + $01 M ← (M) + $01	DIR INH INH IX1 IX SP1	3C dd 4C 5C 6C ff 7C 9E 6C ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ –	– ↕ ↕ –
JMP *opr8a* JMP *opr16a* JMP *oprx16*,X JMP *oprx8*,X JMP ,X	Jump PC ← Jump Address	DIR EXT IX2 IX1 IX	BC dd CC hh ll DC ee ff EC ff FC	3 4 4 3 3	ppp pppp pppp ppp ppp	– –	– – – –
JSR *opr8a* JSR *opr16a* JSR *oprx16*,X JSR *oprx8*,X JSR ,X	Jump to Subroutine PC ← (PC) + *n* (*n* = 1, 2, or 3) Push (PCL); SP ← (SP) – $0001 Push (PCH); SP ← (SP) – $0001 PC ← Unconditional Address	DIR EXT IX2 IX1 IX	BD dd CD hh ll DD ee ff ED ff FD	5 6 6 5 5	ssppp pssppp pssppp ssppp ssppp	– –	– – – –

LDA #*opr8i*
LDA *opr8a*
LDA *opr16a*
LDA *oprx16*,X
LDA *oprx8*,X
LDA ,X
LDA *oprx16*,SP
LDA *oprx8*,SP
	Load Accumulator from Memory
	A ← (M)
		IMM
		DIR
		EXT
		IX2
		IX1
		IX
		SP2
		SP1
			A6 ii
			B6 dd
			C6 hh ll
			D6 ee ff
			E6 ff
			F6
			9E D6 ee ff
			9E E6 ff
				2
				3
				4
				4
				3
				3
				5
				4
					pp
					rpp
					prpp
					prpp
					rpp
					rfp
					pprpp
					prpp
						0 –	– ↕ ↕ –

LDHX #*opr16i*
LDHX *opr8a*
LDHX *opr16a*
LDHX ,X
LDHX *oprx16*,X
LDHX *oprx8*,X
LDHX *oprx8*,SP
	Load Index Register (H:X)
	H:X ← (M:M + $0001)
		IMM
		DIR
		EXT
		IX
		IX2
		IX1
		SP1
			45 jj kk
			55 dd
			32 hh ll
			9E AE
			9E BE ee ff
			9E CE ff
			9E FE ff
				3
				4
				5
				5
				6
				5
				5
					ppp
					rrpp
					prrpp
					prrfp
					pprrpp
					prrpp
					prrpp
						0 –	– ↕ ↕ –

LDX #*opr8i*
LDX *opr8a*
LDX *opr16a*
LDX *oprx16*,X
LDX *oprx8*,X
LDX ,X
LDX *oprx16*,SP
LDX *oprx8*,SP
	Load X (Index Register Low) from Memory
	X ← (M)
		IMM
		DIR
		EXT
		IX2
		IX1
		IX
		SP2
		SP1
			AE ii
			BE dd
			CE hh ll
			DE ee ff
			EE ff
			FE
			9E DE ee ff
			9E EE ff
				2
				3
				4
				4
				3
				3
				5
				4
					pp
					rpp
					prpp
					prpp
					rpp
					rfp
					pprpp
					prpp
						0 –	– ↕ ↕ –

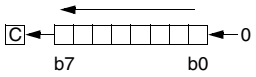
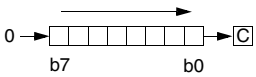
LSL *opr8a* LSLA LSLX LSL *oprx8*,X LSL ,X LSL *oprx8*,SP	Logical Shift Left  (Same as ASL)	DIR INH INH IX1 IX SP1	38 dd 48 58 68 ff 78 9E 68 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ –	– ↕ ↕ ↕
LSR *opr8a* LSRA LSR*X* LSR *oprx8*,X LSR ,X LSR *oprx8*,SP	Logical Shift Right  	DIR INH INH IX1 IX SP1	34 dd 44 54 64 ff 74 9E 64 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ –	– 0 ↕ ↕

Table 7-2. . Instruction Set Summary (Sheet 6 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						VH	I N Z C

MOV *oprx8*,*oprx8*
MOV *oprx8*,X+
MOV #*oprx8*,*oprx8*
MOV ,X+,*oprx8*
	Move
	$(M)_{destination} \leftarrow (M)_{source}$
	In IX+/DIR and DIR/IX+ Modes,
	H:X ← (H:X) + $0001
		DIR/DIR
		DIR/IX+
		IMM/DIR
		IX+/DIR
			4E dd dd
			5E dd
			6E ii dd
			7E dd
				5
				5
				4
				5
					rpwpp
					rfwpp
					pwpp
					rfwpp
						0 –	– ↕ ↕ –

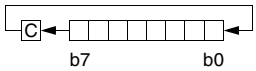
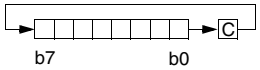
MUL	Unsigned multiply X:A ← (X) × (A)	INH	42	5	ffffp	– 0	– – – 0
NEG *oprx8* NEGA NEGX NEG *oprx8*,X NEG ,X NEG *oprx8*,SP	Negate M ← – (M) = $00 – (M) (Two's Complement) A ← – (A) = $00 – (A) X ← – (X) = $00 – (X) M ← – (M) = $00 – (M) M ← – (M) = $00 – (M) M ← – (M) = $00 – (M)	DIR INH INH IX1 IX SP1	30 dd 40 50 60 ff 70 9E 60 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ –	– ↕ ↕ ↕
NOP	No Operation — Uses 1 Bus Cycle	INH	9D	1	p	– –	– – – –
NSA	Nibble Swap Accumulator A ← (A[3:0]:A[7:4])	INH	62	1	p	– –	– – – –
ORA #*oprx8* ORA *oprx8* ORA *oprx16* ORA *oprx16*,X ORA *oprx8*,X ORA ,X ORA *oprx16*,SP ORA *oprx8*,SP	Inclusive OR Accumulator and Memory A ← (A) | (M)	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA ii BA dd CA hh ll DA ee ff EA ff FA 9E DA ee ff 9E EA ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 –	– ↕ ↕ –
PSHA	Push Accumulator onto Stack Push (A); SP ← (SP) – $0001	INH	87	2	sp	– –	– – – –
PSHH	Push H (Index Register High) onto Stack Push (H); SP ← (SP) – $0001	INH	8B	2	sp	– –	– – – –
PSHX	Push X (Index Register Low) onto Stack Push (X); SP ← (SP) – $0001	INH	89	2	sp	– –	– – – –
PULA	Pull Accumulator from Stack SP ← (SP + $0001); Pull (A)	INH	86	3	ufp	– –	– – – –
PULH	Pull H (Index Register High) from Stack SP ← (SP + $0001); Pull (H)	INH	8A	3	ufp	– –	– – – –
PULX	Pull X (Index Register Low) from Stack SP ← (SP + $0001); Pull (X)	INH	88	3	ufp	– –	– – – –
ROL *oprx8* ROLA ROLX ROL *oprx8*,X ROL ,X ROL *oprx8*,SP	Rotate Left through Carry 	DIR INH INH IX1 IX SP1	39 dd 49 59 69 ff 79 9E 69 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ –	– ↕ ↕ ↕
ROR *oprx8* RORA RORX ROR *oprx8*,X ROR ,X ROR *oprx8*,SP	Rotate Right through Carry 	DIR INH INH IX1 IX SP1	36 dd 46 56 66 ff 76 9E 66 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ –	– ↕ ↕ ↕

Table 7-2. . Instruction Set Summary (Sheet 7 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						VH	I N Z C
RSP	Reset Stack Pointer (Low Byte) SPL ← $FF (High Byte Not Affected)	INH	9C	1	p	– –	– – – –
RTI	Return from Interrupt SP ← (SP) + $0001; Pull (CCR) SP ← (SP) + $0001; Pull (A) SP ← (SP) + $0001; Pull (X) SP ← (SP) + $0001; Pull (PCH) SP ← (SP) + $0001; Pull (PCL)	INH	80	9	uuuuufppp	↕ ↕	↕ ↕ ↕ ↕
RTS	Return from Subroutine SP ← SP + $0001; Pull (PCH) SP ← SP + $0001; Pull (PCL)	INH	81	5	ufppp	– –	– – – –
SBC #*opr8i* SBC *opr8a* SBC *opr16a* SBC *oprx16*,X SBC *oprx8*,X SBC ,X SBC *oprx16*,SP SBC *oprx8*,SP	Subtract with Carry A ← (A) – (M) – (C)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A2 ii B2 dd C2 hh ll D2 ee ff E2 ff F2 9E D2 ee ff 9E E2 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↕ –	– ↕ ↕ ↕
SEC	Set Carry Bit (C ← 1)	INH	99	1	p	– –	– – – 1
SEI	Set Interrupt Mask Bit (I ← 1)	INH	9B	1	p	– –	1 – – –
STA *opr8a* STA *opr16a* STA *oprx16*,X STA *oprx8*,X STA ,X STA *oprx16*,SP STA *oprx8*,SP	Store Accumulator in Memory M ← (A)	DIR EXT IX2 IX1 IX SP2 SP1	B7 dd C7 hh ll D7 ee ff E7 ff F7 9E D7 ee ff 9E E7 ff	3 4 4 3 2 5 4	wpp pwpp pwpp wpp wp ppwpp pwpp	0 –	– ↕ ↕ –
STHX *opr8a* STHX *opr16a* STHX *oprx8*,SP	Store H:X (Index Reg.) (M:M + $0001) ← (H:X)	DIR EXT SP1	35 dd 96 hh ll 9E FF ff	4 5 5	wwpp pwwpp pwwpp	0 –	– ↕ ↕ –
STOP	Enable Interrupts: Stop Processing Refer to MCU Documentation I bit ← 0; Stop Processing	INH	8E	2	fp...	– –	0 – – –
STX *opr8a* STX *opr16a* STX *oprx16*,X STX *oprx8*,X STX ,X STX *oprx16*,SP STX *oprx8*,SP	Store X (Low 8 Bits of Index Register) in Memory M ← (X)	DIR EXT IX2 IX1 IX SP2 SP1	BF dd CF hh ll DF ee ff EF ff FF 9E DF ee ff 9E EF ff	3 4 4 3 2 5 4	wpp pwpp pwpp wpp wp ppwpp pwpp	0 –	– ↕ ↕ –

Table 7-2. . Instruction Set Summary (Sheet 8 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						VH	I N Z C

SUB #*opr8i*
SUB *opr8a*
SUB *opr16a*
SUB *oprx16*,X
SUB *oprx8*,X
SUB ,X
SUB *oprx16*,SP
SUB *oprx8*,SP
	Subtract
	A ← (A) – (M)
		IMM
		DIR
		EXT
		IX2
		IX1
		IX
		SP2
		SP1
			A0 ii
			B0 dd
			C0 hh ll
			D0 ee ff
			E0 ff
			F0
			9E D0 ee ff
			9E E0 ff
				2
				3
				4
				4
				3
				3
				5
				4
					pp
					rpp
					prpp
					prpp
					rpp
					rfp
					pprpp
					prpp
						↕ –	– ↕ ↕ ↕

SWI	Software Interrupt PC ← (PC) + $0001 Push (PCL); SP ← (SP) – $0001 Push (PCH); SP ← (SP) – $0001 Push (X); SP ← (SP) – $0001 Push (A); SP ← (SP) – $0001 Push (CCR); SP ← (SP) – $0001 I ← 1; PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	INH	83	11	ssssvvfppp	– –	1 – – –
TAP	Transfer Accumulator to CCR CCR ← (A)	INH	84	1	p	↕ ↕	↕ ↕ ↕ ↕
TAX	Transfer Accumulator to X (Index Register Low) X ← (A)	INH	97	1	p	– –	– – – –
TPA	Transfer CCR to Accumulator A ← (CCR)	INH	85	1	p	– –	– – – –
TST *opr8a* TSTA TSTX TST *oprx8*,X TST ,X TST *oprx8*,SP	Test for Negative or Zero (M) – $00 (A) – $00 (X) – $00 (M) – $00 (M) – $00 (M) – $00	DIR INH INH IX1 IX SP1	3D dd 4D 5D 6D ff 7D 9E 6D ff	4 1 1 4 3 5	rfpp p p rfpp rfp prfpp	0 –	– ↕ ↕ –
TSX	Transfer SP to Index Reg. H:X ← (SP) + $0001	INH	95	2	fp	– –	– – – –
TXA	Transfer X (Index Reg. Low) to Accumulator A ← (X)	INH	9F	1	p	– –	– – – –

Table 7-2. . Instruction Set Summary (Sheet 9 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						VH	I N Z C
TXS	Transfer Index Reg. to SP SP ← (H:X) – $0001	INH	94	2	fp	– –	– – – –
WAIT	Enable Interrupts; Wait for Interrupt I bit ← 0; Halt CPU	INH	8F	2+	fp...	– –	0 – – –

Source Form: Everything in the source forms columns, *except expressions in italic characters*, is literal information which must appear in the assembly source file exactly as shown. The initial 3- to 5-letter mnemonic and the characters (# , ( ) and +) are always a literal characters.

n	Any label or expression that evaluates to a single integer in the range 0-7.
opr8i	Any label or expression that evaluates to an 8-bit immediate value.
opr16i	Any label or expression that evaluates to a 16-bit immediate value.
opr8a	Any label or expression that evaluates to an 8-bit direct-page address ($00xx).
opr16a	Any label or expression that evaluates to a 16-bit address.
oprx8	Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing.
oprx16	Any label or expression that evaluates to a 16-bit value, used for indexed addressing.
rel	Any label or expression that refers to an address that is within –128 to +127 locations from the start of the next instruction.

Operation Symbols:

A	Accumulator
CCR	Condition code register
H	Index register high byte
M	Memory location
n	Any bit
opr	Operand (one or two bytes)
PC	Program counter
PCH	Program counter high byte
PCL	Program counter low byte
rel	Relative program counter offset byte
SP	Stack pointer
SPL	Stack pointer low byte
X	Index register low byte
&	Logical AND
|	Logical OR
⊕	Logical EXCLUSIVE OR
( )	Contents of
+	Add
–	Subtract, Negation (two's complement)
×	Multiply
÷	Divide
#	Immediate value
←	Loaded with
:	Concatenated with

Addressing Modes:

DIR	Direct addressing mode
EXT	Extended addressing mode
IMM	Immediate addressing mode
INH	Inherent addressing mode
IX	Indexed, no offset addressing mode
IX1	Indexed, 8-bit offset addressing mode
IX2	Indexed, 16-bit offset addressing mode
IX+	Indexed, no offset, post increment addressing mode
IX1+	Indexed, 8-bit offset, post increment addressing mode
REL	Relative addressing mode
SP1	Stack pointer, 8-bit offset addressing mode
SP2	Stack pointer 16-bit offset addressing mode

Cycle-by-Cycle Codes:

f	Free cycle. This indicates a cycle where the CPU does not require use of the system buses. An f cycle is always one cycle of the system bus clock and is always a read cycle.
p	Progryam fetch; read from next consecutive location in program memory
r	Read 8-bit operand
s	Push (write) one byte onto stack
u	Pop (read) one byte from stack
v	Read vector from $FFxx (high byte first)
w	Write 8-bit operand

CCR Bits:

V	Overflow bit
H	Half-carry bit
I	Interrupt mask
N	Negative bit
Z	Zero bit
C	Carry/borrow bit

CCR Effects:

↕	Set or cleared
–	Not affected
U	Undefined

Table 7-3. Opcode Map (Sheet 1 of 2)

Bit-Manipulation		Branch	Read-Modify-Write						Control		Register/Memory					
00 5 BRSET0 3 DIR	10 5 BSET0 2 DIR	20 3 BRA 2 REL	30 5 NEG 2 DIR	40 1 NEGA 1 INH	50 1 NEGX 1 INH	60 5 NEG 2 IX1	70 4 NEG 1 IX	80 9 RTI 1 INH	90 3 BGE 2 REL	A0 2 SUB 2 IMM	B0 3 SUB 2 DIR	C0 4 SUB 3 EXT	D0 4 SUB 3 IX2	E0 3 SUB 2 IX1	F0 3 SUB 1 IX
01 5 BRCLR0 3 DIR	11 5 BCLR0 2 DIR	21 3 BRN 2 REL	31 5 CBEQ 3 DIR	41 4 CBEQA 3 IMM	51 4 CBEQX 3 IMM	61 5 CBEQ 3 IX1+	71 5 CBEQ 2 IX+	81 6 RTS 1 INH	91 3 BLT 2 REL	A1 2 CMP 2 IMM	B1 3 CMP 2 DIR	C1 4 CMP 3 EXT	D1 4 CMP 3 IX2	E1 3 CMP 2 IX1	F1 3 CMP 1 IX
02 5 BRSET1 3 DIR	12 5 BSET1 2 DIR	22 3 BHI 2 REL	32 5 LDHX 3 EXT	42 5 MUL 1 INH	52 6 DIV 1 INH	62 1 NSA 1 INH	72 1 DAA 1 INH	82 5+ BGND 1 INH	92 3 BGT 2 REL	A2 2 SBC 2 IMM	B2 3 SBC 2 DIR	C2 4 SBC 3 EXT	D2 4 SBC 3 IX2	E2 3 SBC 2 IX1	F2 3 SBC 1 IX
03 5 BRCLR1 3 DIR	13 5 BCLR1 2 DIR	23 3 BLS 2 REL	33 5 COM 2 DIR	43 1 COMA 1 INH	53 1 COMX 1 INH	63 5 COM 2 IX1	73 4 COM 1 IX	83 11 SWI 1 INH	93 3 BLE 2 REL	A3 2 CPX 2 IMM	B3 3 CPX 2 DIR	C3 4 CPX 3 EXT	D3 4 CPX 3 IX2	E3 3 CPX 2 IX1	F3 3 CPX 1 IX
04 5 BRSET2 3 DIR	14 5 BSET2 2 DIR	24 3 BCC 2 REL	34 5 LSR 2 DIR	44 1 LSRA 1 INH	54 1 LSRX 1 INH	64 5 LSR 2 IX1	74 4 LSR 1 IX	84 1 TAP 1 INH	94 2 TXS 1 INH	A4 2 AND 2 IMM	B4 3 AND 2 DIR	C4 4 AND 3 EXT	D4 4 AND 3 IX2	E4 3 AND 2 IX1	F4 3 AND 1 IX
05 5 BRCLR2 3 DIR	15 5 BCLR2 2 DIR	25 3 BCS 2 REL	35 4 STHX 2 DIR	45 3 LDHX 3 IMM	55 4 LDHX 2 DIR	65 3 CPHX 3 IMM	75 5 CPHX 2 DIR	85 1 TPA 1 INH	95 2 TSX 1 INH	A5 2 BIT 2 IMM	B5 3 BIT 2 DIR	C5 4 BIT 3 EXT	D5 4 BIT 3 IX2	E5 3 BIT 2 IX1	F5 3 BIT 1 IX
06 5 BRSET3 3 DIR	16 5 BSET3 2 DIR	26 3 BNE 2 REL	36 5 ROR 2 DIR	46 1 RORA 1 INH	56 1 RORX 1 INH	66 5 ROR 2 IX1	76 4 ROR 1 IX	86 3 PULA 1 INH	96 5 STHX 3 EXT	A6 2 LDA 2 IMM	B6 3 LDA 2 DIR	C6 4 LDA 3 EXT	D6 4 LDA 3 IX2	E6 3 LDA 2 IX1	F6 3 LDA 1 IX
07 5 BRCLR3 3 DIR	17 5 BCLR3 2 DIR	27 3 BEQ 2 REL	37 5 ASR 2 DIR	47 1 ASRA 1 INH	57 1 ASRX 1 INH	67 5 ASR 2 IX1	77 4 ASR 1 IX	87 2 PSHA 1 INH	97 1 TAX 1 INH	A7 2 AIS 2 IMM	B7 3 STA 2 DIR	C7 4 STA 3 EXT	D7 4 STA 3 IX2	E7 3 STA 2 IX1	F7 2 STA 1 IX
08 5 BRSET4 3 DIR	18 5 BSET4 2 DIR	28 3 BHCC 2 REL	38 5 LSL 2 DIR	48 1 LSLA 1 INH	58 1 LSLX 1 INH	68 5 LSL 2 IX1	78 4 LSL 1 IX	88 3 PULX 1 INH	98 1 CLC 1 INH	A8 2 EOR 2 IMM	B8 3 EOR 2 DIR	C8 4 EOR 3 EXT	D8 4 EOR 3 IX2	E8 3 EOR 2 IX1	F8 3 EOR 1 IX
09 5 BRCLR4 3 DIR	19 5 BCLR4 2 DIR	29 3 BHCS 2 REL	39 5 ROL 2 DIR	49 1 ROLA 1 INH	59 1 ROLX 1 INH	69 5 ROL 2 IX1	79 4 ROL 1 IX	89 2 PSHX 1 INH	99 1 SEC 1 INH	A9 2 ADC 2 IMM	B9 3 ADC 2 DIR	C9 4 ADC 3 EXT	D9 4 ADC 3 IX2	E9 3 ADC 2 IX1	F9 3 ADC 1 IX
0A 5 BRSET5 3 DIR	1A 5 BSET5 2 DIR	2A 3 BPL 2 REL	3A 5 DEC 2 DIR	4A 1 DECA 1 INH	5A 1 DECX 1 INH	6A 5 DEC 2 IX1	7A 4 DEC 1 IX	8A 3 PULH 1 INH	9A 1 CLI 1 INH	AA 2 ORA 2 IMM	BA 3 ORA 2 DIR	CA 4 ORA 3 EXT	DA 4 ORA 3 IX2	EA 3 ORA 2 IX1	FA 3 ORA 1 IX
0B 5 BRCLR5 3 DIR	1B 5 BCLR5 2 DIR	2B 3 BMI 2 REL	3B 7 DBNZ 3 DIR	4B 4 DBNZA 2 INH	5B 4 DBNZX 2 INH	6B 7 DBNZ 3 IX1	7B 6 DBNZ 2 IX	8B 2 PSHH 1 INH	9B 1 SEI 1 INH	AB 2 ADD 2 IMM	BB 3 ADD 2 DIR	CB 4 ADD 3 EXT	DB 4 ADD 3 IX2	EB 3 ADD 2 IX1	FB 3 ADD 1 IX
0C 5 BRSET6 3 DIR	1C 5 BSET6 2 DIR	2C 3 BMC 2 REL	3C 5 INC 2 DIR	4C 1 INCA 1 INH	5C 1 INCX 1 INH	6C 5 INC 2 IX1	7C 4 INC 1 IX	8C 1 CLRH 1 INH	9C 1 RSP 1 INH		BC 3 JMP 2 DIR	CC 4 JMP 3 EXT	DC 4 JMP 3 IX2	EC 3 JMP 2 IX1	FC 3 JMP 1 IX
0D 5 BRCLR6 3 DIR	1D 5 BCLR6 2 DIR	2D 3 BMS 2 REL	3D 4 TST 2 DIR	4D 1 TSTA 1 INH	5D 1 TSTX 1 INH	6D 4 TST 2 IX1	7D 3 TST 1 IX		9D 1 NOP 1 INH	AD 5 BSR 2 REL	BD 5 JSR 2 DIR	CD 6 JSR 3 EXT	DD 6 JSR 3 IX2	ED 5 JSR 2 IX1	FD 5 JSR 1 IX
0E 5 BRSET7 3 DIR	1E 5 BSET7 2 DIR	2E 3 BIL 2 REL	3E 6 CPHX 3 EXT	4E 5 MOV 3 DD	5E 5 MOV 2 DIX+	6E 4 MOV 3 IMD	7E 5 MOV 2 IX+D	8E 2+ STOP 1 INH	9E Page 2	AE 2 LDX 2 IMM	BE 3 LDX 2 DIR	CE 4 LDX 3 EXT	DE 4 LDX 3 IX2	EE 3 LDX 2 IX1	FE 3 LDX 1 IX
0F 5 BRCLR7 3 DIR	1F 5 BCLR7 2 DIR	2F 3 BIH 2 REL	3F 5 CLR 2 DIR	4F 1 CLRA 1 INH	5F 1 CLRX 1 INH	6F 5 CLR 2 IX1	7F 4 CLR 1 IX	8F 2+ WAIT 1 INH	9F 1 TXA 1 INH	AF 2 AIX 2 IMM	BF 3 STX 2 DIR	CF 4 STX 3 EXT	DF 4 STX 3 IX2	EF 3 STX 2 IX1	FF 2 STX 1 IX

INH	**Inherent**	**REL**	**Relative**	**SP1**	**Stack Pointer, 8-Bit Offset**
IMM	**Immediate**	**IX**	**Indexed, No Offset**	**SP2**	**Stack Pointer, 16-Bit Offset**
DIR	**Direct**	**IX1**	**Indexed, 8-Bit Offset**	**IX+**	**Indexed, No Offset with Post Increment**
EXT	**Extended**	**IX2**	**Indexed, 16-Bit Offset**	**IX1+**	**Indexed, 1-Byte Offset with Post Increment**
DD	**DIR to DIR**	**IMD**	**IMM to DIR**		
IX+D	**IX+ to DIR**	**DIX+**	**DIR to IX+**		

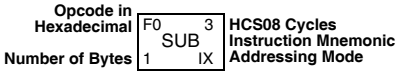


Table 7-3. Opcode Map (Sheet 2 of 2)

Bit-Manipulation		Branch	Read-Modify-Write					Control		Register/Memory					
						9E60 6 NEG 3 SP1							9ED0 5 SUB 4 SP2	9EE0 4 SUB 3 SP1	
						9E61 6 CBEQ 4 SP1							9ED1 5 CMP 4 SP2	9EE1 4 CMP 3 SP1	
													9ED2 5 SBC 4 SP2	9EE2 4 SBC 3 SP1	
						9E63 6 COM 3 SP1							9ED3 5 CPX 4 SP2	9EE3 4 CPX 3 SP1	9EF3 6 CPHX 3 SP1
						9E64 6 LSR 3 SP1							9ED4 5 AND 4 SP2	9EE4 4 AND 3 SP1	
													9ED5 5 BIT 4 SP2	9EE5 4 BIT 3 SP1	
						9E66 6 ROR 3 SP1							9ED6 5 LDA 4 SP2	9EE6 4 LDA 3 SP1	
						9E67 6 ASR 3 SP1							9ED7 5 STA 4 SP2	9EE7 4 STA 3 SP1	
						9E68 6 LSL 3 SP1							9ED8 5 EOR 4 SP2	9EE8 4 EOR 3 SP1	
						9E69 6 ROL 3 SP1							9ED9 5 ADC 4 SP2	9EE9 4 ADC 3 SP1	
						9E6A 6 DEC 3 SP1							9EDA 5 ORA 4 SP2	9EEA 4 ORA 3 SP1	
						9E6B 8 DBNZ 4 SP1							9EDB 5 ADD 4 SP2	9EEB 4 ADD 3 SP1	
						9E6C 6 INC 3 SP1									
						9E6D 5 TST 3 SP1									
										9EAE 5 LDHX 2 IX	9EBE 6 LDHX 4 IX2	9ECE 5 LDHX 3 IX1	9EDE 5 LDX 4 SP2	9EEE 4 LDX 3 SP1	9EFE 5 LDHX 3 SP1
						9E6F 6 CLR 3 SP1							9EDF 5 STX 4 SP2	9EEF 4 STX 3 SP1	9EFF 5 STHX 3 SP1

INH	**Inherent**	**REL**	**Relative**	**SP1**	**Stack Pointer, 8-Bit Offset**
IMM	**Immediate**	**IX**	**Indexed, No Offset**	**SP2**	**Stack Pointer, 16-Bit Offset**
DIR	**Direct**	**IX1**	**Indexed, 8-Bit Offset**	**IX+**	**Indexed, No Offset with**
EXT	**Extended**	**IX2**	**Indexed, 16-Bit Offset**		**Post Increment**
DD	**DIR to DIR**	**IMD**	**IMM to DIR**	**IX1+**	**Indexed, 1-Byte Offset with**
IX+D	**IX+ to DIR**	**DIX+**	**DIR to IX+**		**Post Increment**

Note: All Sheet 2 Opcodes are Preceded by the Page 2 Prebyte (9E)

Prebyte (9E) and Opcode in Hexadecimal
Number of Bytes

9E60 6
NEG
3 SP1

HCS08 Cycles
Instruction Mnemonic
Addressing Mode

Chapter 8
5 V Analog Comparator (S08ACMPV2)

8.1 Introduction

The analog comparator module (ACMP) provides a circuit for comparing two analog input voltages or for comparing one analog input voltage to an internal reference voltage. The comparator circuit is designed to operate across the full range of the supply voltage (rail to rail operation).

NOTE

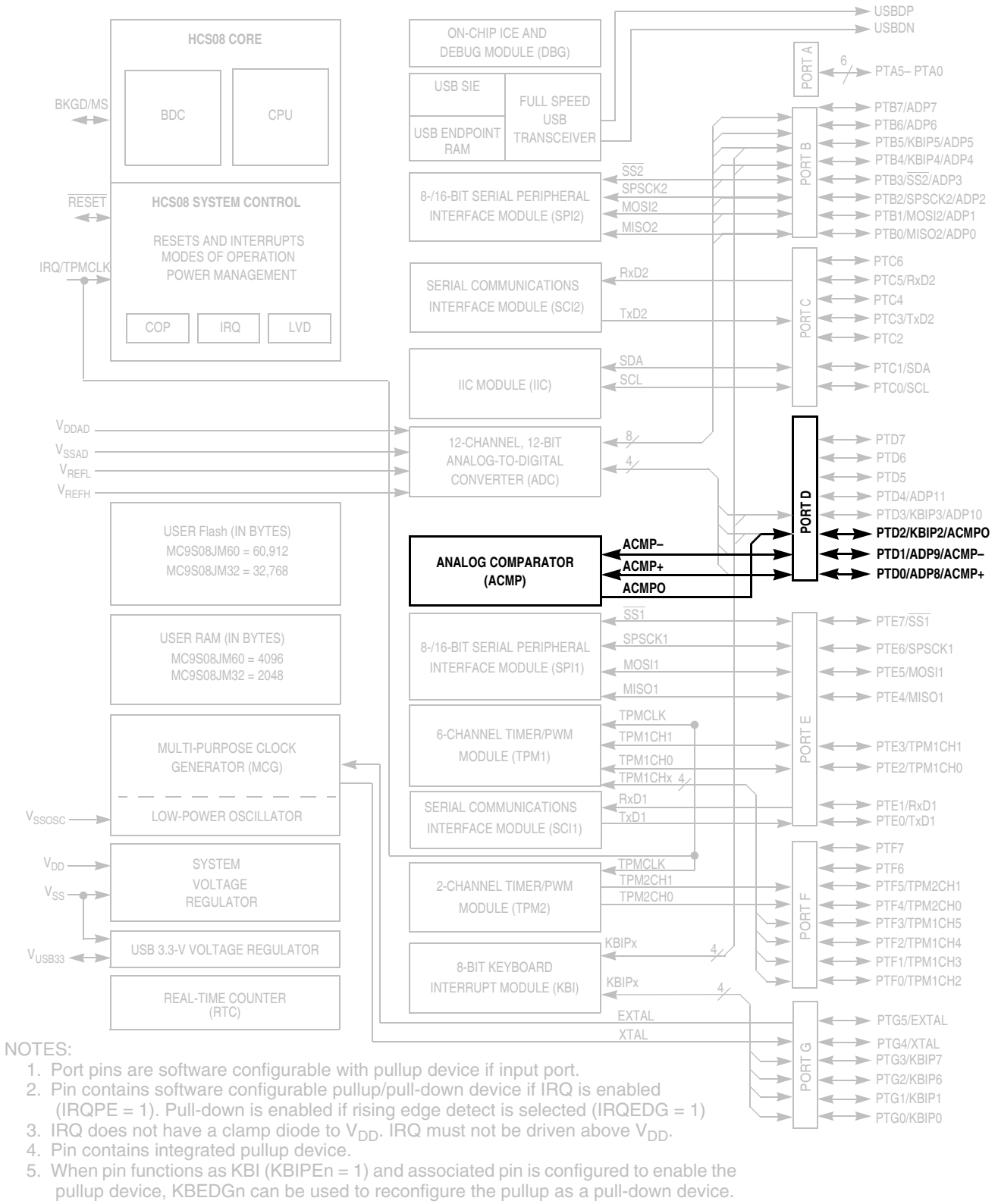
MC9S08JM60 series devices operate at a higher voltage range (2.7 V to 5.5 V) and do not include stop1 mode. Therefore, please disregard references to stop1.

8.1.1 ACMP Configuration Information

When using the bandgap reference voltage for input to ACMP+, the user must enable the bandgap buffer by setting BGBE =1 in SPMSC1 see Section 5.7.7, "System Power Management Status and Control 1 Register (SPMSC1)." For value of bandgap voltage reference see Appendix A.6, "DC Characteristics."

8.1.2 ACMP/TPM Configuration Information

The ACMP module can be configured to connect the output of the analog comparator to TPM input capture channel 0 by setting ACIC in SOPT2. With ACIC set, the TPM1CH0 pin is not available externally regardless of the configuration of the TPM module.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 8-1. MC9S08JM60 Series Block Diagram Highlighting ACMP Block and Pins

8.1.3 Features

The ACMP has the following features:

- Full rail to rail supply operation.
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output.
- Option to compare to fixed internal bandgap reference voltage.
- Option to allow comparator output to be visible on a pin, ACMPO.
- Can operate in stop3 mode

8.1.4 Modes of Operation

This section defines the ACMP operation in wait, stop and background debug modes.

8.1.4.1 ACMP in Wait Mode

The ACMP continues to run in wait mode if enabled before executing the WAIT instruction. Therefore, the ACMP can be used to bring the MCU out of wait mode if the ACMP interrupt, ACIE is enabled. For lowest possible current consumption, the ACMP must be disabled by software if not required as an interrupt source during wait mode.

8.1.4.2 ACMP in Stop Modes

8.1.4.2.1 Stop3 Mode Operation

The ACMP continues to operate in stop3 mode if enabled and compare operation remains active. If ACOPE is enabled, comparator output operates as in the normal operating mode and comparator output is placed onto the external pin. The MCU is brought out of stop when a compare event occurs and ACIE is enabled; ACF flag sets accordingly.

If stop is exited with a reset, the ACMP will be put into its reset state.

8.1.4.2.2 Stop2 and Stop1 Mode Operation

During either stop2 and stop1 mode, the ACMP module will be fully powered down. Upon wake-up from stop2 or stop1 mode, the ACMP module will be in the reset state.

8.1.4.3 ACMP in Active Background Mode

When the microcontroller is in active background mode, the ACMP will continue to operate normally.

8.1.5 Block Diagram

The block diagram for the Analog Comparator module is shown Figure 8-2.

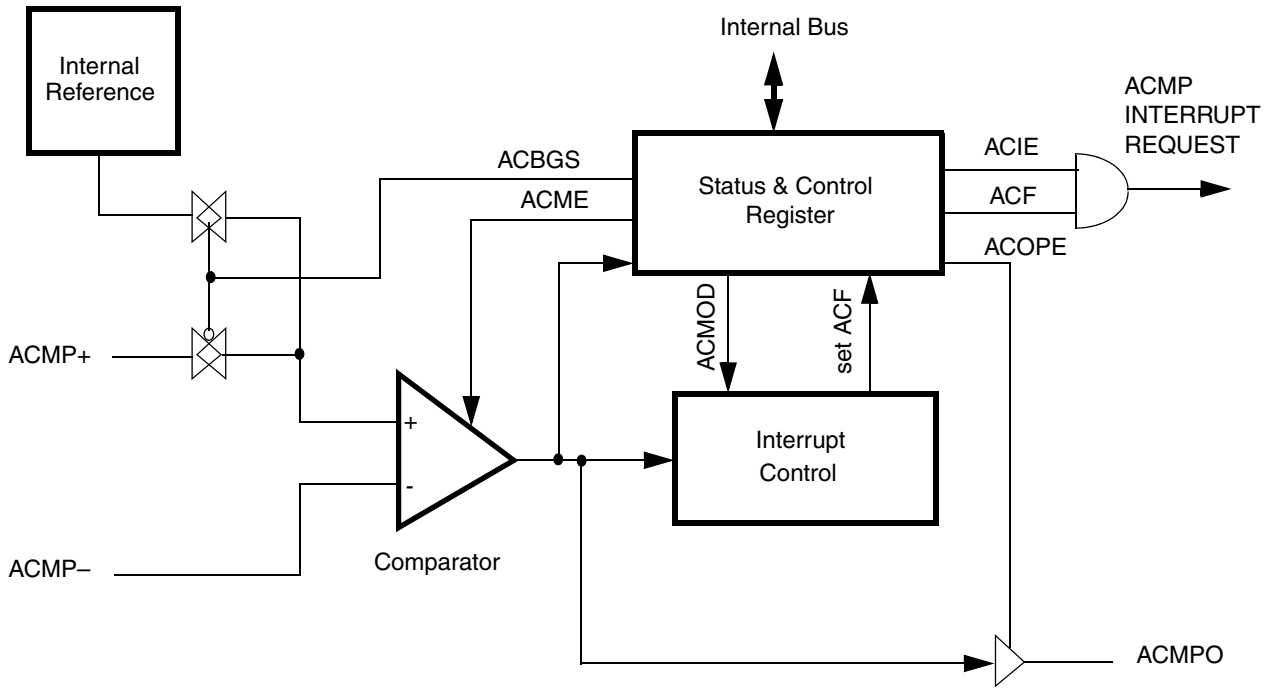


Figure 8-2. Analog Comparator 5V (ACMP5) Block Diagram

8.2 External Signal Description

The ACMP has two analog input pins, ACMP+ and ACMP– and one digital output pin ACMPO. Each of these pins can accept an input voltage that varies across the full operating voltage range of the MCU. As shown in Figure 8-2, the ACMP– pin is connected to the inverting input of the comparator, and the ACMP+ pin is connected to the comparator non-inverting input if ACBGS is a 0. As shown in Figure 8-2, the ACMPO pin can be enabled to drive an external pin.

The signal properties of ACMP are shown in Table 8-1.

Table 8-1. Signal Properties

Signal	Function	I/O
ACMP–	Inverting analog input to the ACMP. (Minus input)	I
ACMP+	Non-inverting analog input to the ACMP. (Positive input)	I
ACMPO	Digital output of the ACMP.	O

8.3 Memory Map

8.3.1 Register Descriptions

The ACMP includes one register:

- An 8-bit status and control register

Refer to the direct-page register summary in the memory section of this data sheet for the absolute address assignments for all ACMP registers.This section refers to registers and control bits only by their names.

Some MCUs may have more than one ACMP, so register names include placeholder characters to identify which ACMP is being referenced.

8.3.1.1 ACMP Status and Control Register (ACMPSC)

ACMPSC contains the status flag and control bits which are used to enable and configure the ACMP.

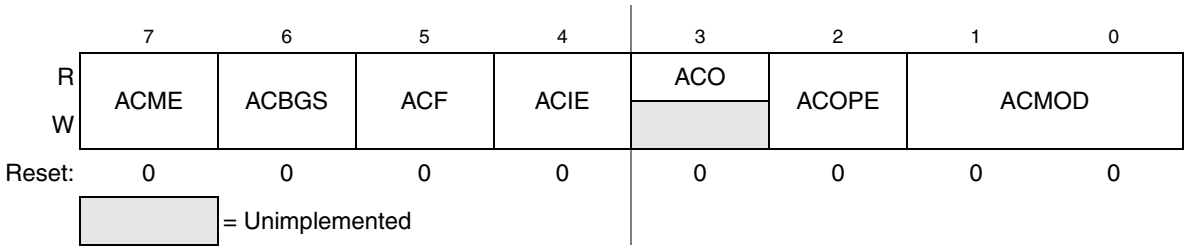


Figure 8-3. ACMP Status and Control Register

Table 8-2. ACMP Status and Control Register Field Descriptions

Field	Description
7 ACME	**Analog Comparator Module Enable** — ACME enables the ACMP module. 0 ACMP not enabled 1 ACMP is enabled
6 ACBGS	**Analog Comparator Bandgap Select** — ACBGS is used to select between the bandgap reference voltage or the ACMP+ pin as the input to the non-inverting input of the analog comparatorr. 0 External pin ACMP+ selected as non-inverting input to comparator 1 Internal reference select as non-inverting input to comparator Note: refer to this chapter introduction to verify if any other config bits are necessary to enable the bandgap reference in the chip level.
5 ACF	**Analog Comparator Flag** — ACF is set when a compare event occurs. Compare events are defined by ACMOD. ACF is cleared by writing a one to ACF. 0 Compare event has not occurred 1 Compare event has occurred
4 ACIE	**Analog Comparator Interrupt Enable** — ACIE enables the interrupt from the ACMP. When ACIE is set, an interrupt will be asserted when ACF is set. 0 Interrupt disabled 1 Interrupt enabled
3 ACO	**Analog Comparator Output** — Reading ACO will return the current value of the analog comparator output. ACO is reset to a 0 and will read as a 0 when the ACMP is disabled (ACME = 0).

Table 8-2. ACMP Status and Control Register Field Descriptions (continued)

Field	Description
2 ACOPE	**Analog Comparator Output Pin Enable** — ACOPE is used to enable the comparator output to be placed onto the external pin, ACMPO. 0 Analog comparator output not available on ACMPO 1 Analog comparator output is driven out on ACMPO
1:0 ACMOD	**Analog Comparator Mode** — ACMOD selects the type of compare event which sets ACF. 00 Encoding 0 — Comparator output falling edge 01 Encoding 1 — Comparator output rising edge 10 Encoding 2 — Comparator output falling edge 11 Encoding 3 — Comparator output rising or falling edge

8.4 Functional Description

The analog comparator can be used to compare two analog input voltages applied to ACMP+ and ACMP–; or it can be used to compare an analog input voltage applied to ACMP– with an internal bandgap reference voltage. ACBGS is used to select between the bandgap reference voltage or the ACMP+ pin as the input to the non-inverting input of the analog comparator. The comparator output is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. ACMOD is used to select the condition which will cause ACF to be set. ACF can be set on a rising edge of the comparator output, a falling edge of the comparator output, or either a rising or a falling edge (toggle). The comparator output can be read directly through ACO. The comparator output can be driven onto the ACMPO pin using ACOPE.

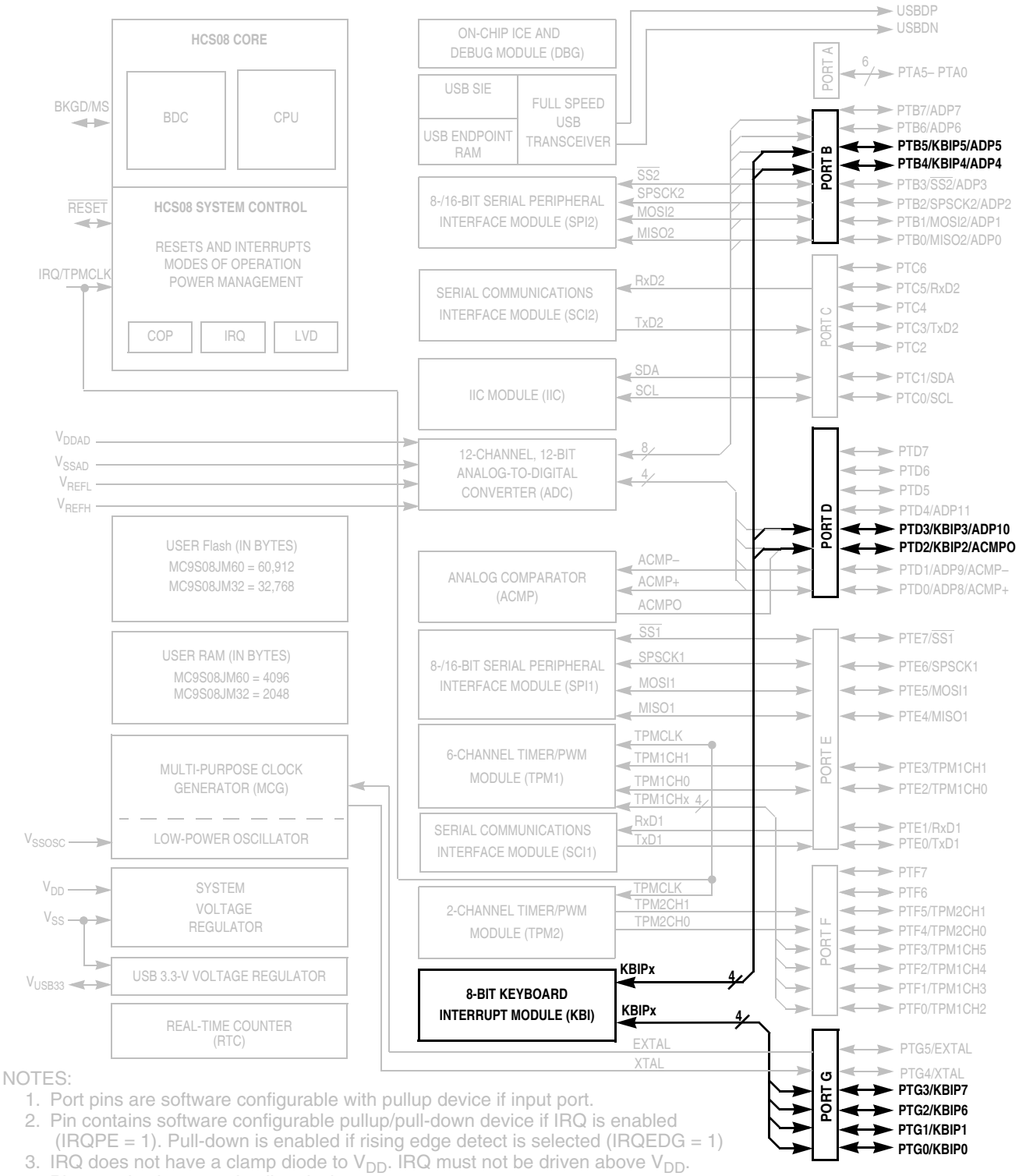
Chapter 9
Keyboard Interrupt (S08KBIV2)

9.1 Introduction

The MC9S08JM60 series have one KBI module with eight keyboard interrupt inputs. See Chapter 2, "Pins and Connections," for more information about the logic and hardware aspects of these pins.

NOTE

MC9S08JM60 series devices operate at a higher voltage range (2.7 V to 5.5 V) and do not include stop1 mode. Therefore, please disregard references to stop1.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 9-1. MC9S08JM60 Series Block Diagram Highlighting KBI Block and Pins

9.1.1 Features

The KBI features include:

- Up to eight keyboard interrupt pins with individual pin enable bits.
- Each keyboard interrupt pin is programmable as falling edge (or rising edge) only, or both falling edge and low level (or both rising edge and high level) interrupt sensitivity.
- One software enabled keyboard interrupt.
- Exit from low-power modes.

9.1.2 Modes of Operation

This section defines the KBI operation in wait, stop, and background debug modes.

9.1.2.1 KBI in Wait Mode

The KBI continues to operate in wait mode if enabled before executing the WAIT instruction. Therefore, an enabled KBI pin (KBPEx = 1) can be used to bring the MCU out of wait mode if the KBI interrupt is enabled (KBIE = 1).

9.1.2.2 KBI in Stop Modes

The KBI operates asynchronously in stop3 mode if enabled before executing the STOP instruction. Therefore, an enabled KBI pin (KBPEx = 1) can be used to bring the MCU out of stop3 mode if the KBI interrupt is enabled (KBIE = 1).

During either stop1 or stop2 mode, the KBI is disabled. In some systems, the pins associated with the KBI may be sources of wakeup from stop1 or stop2, see the stop modes section in the Modes of Operation chapter. Upon wake-up from stop1 or stop2 mode, the KBI module will be in the reset state.

9.1.2.3 KBI in Active Background Mode

When the microcontroller is in active background mode, the KBI will continue to operate normally.

9.1.3 Block Diagram

The block diagram for the keyboard interrupt module is shown Figure 9-2.

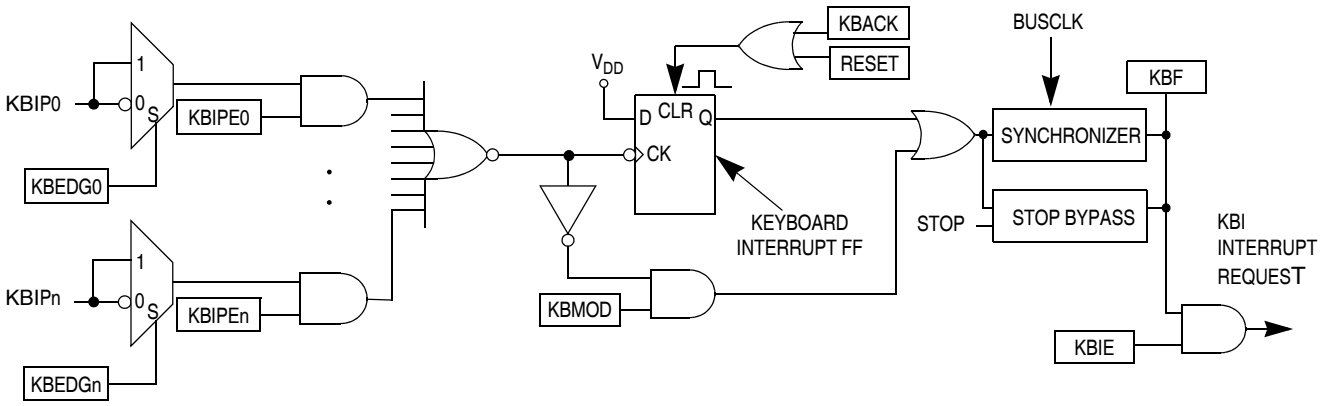


Figure 9-2. KBI Block Diagram

9.2 External Signal Description

The KBI input pins can be used to detect either falling edges, or both falling edge and low level interrupt requests. The KBI input pins can also be used to detect either rising edges, or both rising edge and high level interrupt requests.

The signal properties of KBI are shown in Table 9-1.

Table 9-1. Signal Properties

Signal	Function	I/O
KBIPn	Keyboard interrupt pins	I

9.3 Register Definition

The KBI includes three registers:

- An 8-bit pin status and control register.
- An 8-bit pin enable register.
- An 8-bit edge select register.

Refer to the direct-page register summary in the Memory chapter for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names.

Some MCUs may have more than one KBI, so register names include placeholder characters to identify which KBI is being referenced.

9.3.1 KBI Status and Control Register (KBISC)

KBISC contains the status flag and control bits, which are used to configure the KBI.

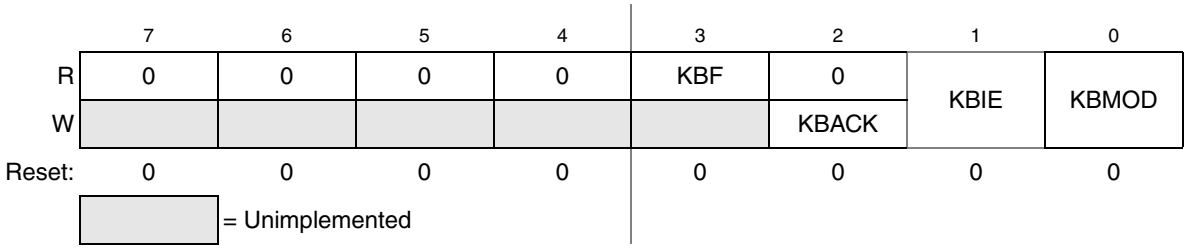


Figure 9-3. KBI Status and Control Register

Table 9-2. KBISC Register Field Descriptions

Field	Description
7:4	Unused register bits, always read 0.
3 KBF	**Keyboard Interrupt Flag** — KBF indicates when a keyboard interrupt is detected. Writes have no effect on KBF. 0 No keyboard interrupt detected. 1 Keyboard interrupt detected.
2 KBACK	**Keyboard Acknowledge** — Writing a 1 to KBACK is part of the flag clearing mechanism. KBACK always reads as 0.
1 KBIE	**Keyboard Interrupt Enable** — KBIE determines whether a keyboard interrupt is requested. 0 Keyboard interrupt request not enabled. 1 Keyboard interrupt request enabled.
0 KBMOD	**Keyboard Detection Mode** — KBMOD (along with the KBEDG bits) controls the detection mode of the keyboard interrupt pins.0Keyboard detects edges only. 1 Keyboard detects both edges and levels.

9.3.2 KBI Pin Enable Register (KBIPE)

KBIPE contains the pin enable control bits.

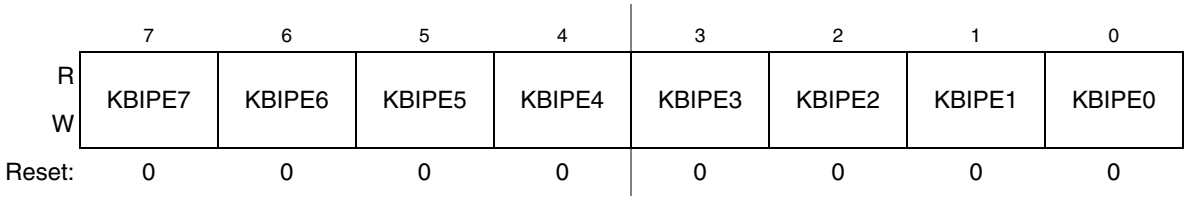


Figure 9-4. KBI Pin Enable Register

Table 9-3. KBIPE Register Field Descriptions

Field	Description
7:0 KBIPEn	**Keyboard Pin Enables** — Each of the KBIPEn bits enable the corresponding keyboard interrupt pin. 0 Pin not enabled as keyboard interrupt. 1 Pin enabled as keyboard interrupt.

9.3.3 KBI Edge Select Register (KBIES)

KBIES contains the edge select control bits.

	7	6	5	4	3	2	1	0
R W	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
Reset:	0	0	0	0	0	0	0	0

Figure 9-5. KBI Edge Select Register

Table 9-4. KBIES Register Field Descriptions

Field	Description
7:0 KBEDGn	**Keyboard Edge Selects** — Each of the KBEDGn bits selects the falling edge/low level or rising edge/high level function of the corresponding pin). 0 Falling edge/low level. 1 Rising edge/high level.

9.4 Functional Description

This on-chip peripheral module is called a keyboard interrupt (KBI) module because originally it was designed to simplify the connection and use of row-column matrices of keyboard switches. However, these inputs are also useful as extra external interrupt inputs and as an external means of waking the MCU from stop or wait low-power modes.

The KBI module allows up to eight pins to act as additional interrupt sources. Writing to the KBIPEn bits in the keyboard interrupt pin enable register (KBIPE) independently enables or disables each KBI pin. Each KBI pin can be configured as edge sensitive or edge and level sensitive based on the KBMOD bit in the keyboard interrupt status and control register (KBISC). Edge sensitive can be software programmed to be either falling or rising; the level can be either low or high. The polarity of the edge or edge and level sensitivity is selected using the KBEDGn bits in the keyboard interrupt edge select register (KBIES).

9.4.1 Edge Only Sensitivity

Synchronous logic is used to detect edges. A falling edge is detected when an enabled keyboard interrupt (KBIPEn=1) input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input signal is seen as a logic 0 (the deasserted level) during one bus cycle and then a logic 1 (the asserted level) during the next cycle.Before the first edge is detected, all enabled keyboard interrupt input signals must be at the deasserted logic levels. After any edge is detected, all enabled keyboard interrupt input signals must return to the deasserted level before any new edge can be detected.

A valid edge on an enabled KBI pin will set KBF in KBISC. If KBIE in KBISC is set, an interrupt request will be presented to the CPU. Clearing of KBF is accomplished by writing a 1 to KBACK in KBISC.

9.4.2 Edge and Level Sensitivity

A valid edge or level on an enabled KBI pin will set KBF in KBISC. If KBIE in KBISC is set, an interrupt request will be presented to the CPU. Clearing of KBF is accomplished by writing a 1 to KBACK in

KBISC provided all enabled keyboard inputs are at their deasserted levels. KBF will remain set if any enabled KBI pin is asserted while attempting to clear by writing a 1 to KBACK.

9.4.3 KBI Pullup/Pulldown Resistors

The KBI pins can be configured to use an internal pullup/pulldown resistor using the associated I/O port pullup enable register. If an internal resistor is enabled, the KBIES register is used to select whether the resistor is a pullup (KBEDGn = 0) or a pulldown (KBEDGn = 1).

9.4.4 KBI Initialization

When a keyboard interrupt pin is first enabled it is possible to get a false keyboard interrupt flag. To prevent a false interrupt request during keyboard initialization, the user should do the following:

1. Mask keyboard interrupts by clearing KBIE in KBISC.
2. Enable the KBI polarity by setting the appropriate KBEDGn bits in KBIES.
3. If using internal pullup/pulldown device, configure the associated pullup enable bits in PTxPE.
4. Enable the KBI pins by setting the appropriate KBIPEn bits in KBIPE.
5. Write to KBACK in KBISC to clear any false interrupts.
6. Set KBIE in KBISC to enable interrupts.

Chapter 10
Analog-to-Digital Converter (S08ADC12V1)

10.1 Overview

The 12-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

NOTE

MC9S08JM60 series devices operate at a higher voltage range (2.7 V to 5.5 V) and do not include stop1 mode. Therefore, please disregard references to stop1.

10.1.1 Module Configurations

This section provides information for configuring the ADC on this device.

10.1.1.1 Channel Assignments

The ADC channel assignments for the MC9S08JM60 Series devices are shown in the table below. Reserved channels convert to an unknown value.

Table 10-1. ADC Channel Assignment

ADCH	Channel	Input	Pin Control
00000	AD0	PTB0/MISO2/ADP0	ADPC0
00001	AD1	PTB1/MOSI2/ADP1	ADPC1
00010	AD2	PTB2/SPSCK2/ADP2	ADPC2
00011	AD3	PTB3/$\overline{SS2}$/ADP3	ADPC3
00100	AD4	PTB4/KBIP4/ADP4	ADPC4
00101	AD5	PTB5/KBIP5/ADP5	ADPC5
00110	AD6	PTB6/ADP6	ADPC6
00111	AD7	PTB7/ADP7	ADPC7
01000	AD8	PTD0/ADP8/ACMP+	ADPC8
01001	AD9	PTD1/ADP9/ACMP-	ADPC9
01010	AD10	PTD3/KBIP3/ADP10	ADPC10
01011	AD11	PTD4/ADP11	ADPC11
01100	AD12	V_{REFL}	ADPC12
01101	AD13	V_{REFL}	ADPC13
01110	AD14	V_{REFL}	ADPC14
01111	AD15	V_{REFL}	ADPC15

ADCH	Channel	Input	Pin Control
10000	AD16	V_{REFL}	N/A
10001	AD17	V_{REFL}	N/A
10010	AD18	V_{REFL}	N/A
10011	AD19	V_{REFL}	N/A
10100	AD20	V_{REFL}	N/A
10101	AD21	Reserved	N/A
10110	AD22	Reserved	N/A
10111	AD23	Reserved	N/A
11000	AD24	Reserved	N/A
11001	AD25	Reserved	N/A
11010	AD26	Temperature Sensor[1]	N/A
11011	AD27	Internal Bandgap	N/A
11100		Reserved	N/A
11101	V_{REFH}	V_{REFH}	N/A
11110	V_{REFL}	V_{REFL}	N/A
11111	module disabled	None	N/A

[1] For more information, see Section 10.1.1.5, “Temperature Sensor.”

NOTE

Selecting the internal bandgap channel requires BGBE =1 in SPMSC1 see Section 5.7.7, “System Power Management Status and Control 1 Register (SPMSC1).” For value of bandgap voltage reference see Appendix A.8, “Analog Comparator (ACMP) Electricals.”

10.1.1.2 Alternate Clock

The ADC is capable of performing conversions using the MCU bus clock, the bus clock divided by two, the local asynchronous clock (ADACK) within the module, or the alternate clock (ALTCLK). The

ALTCLK on this device is the MCGERCLK.

The selected clock source must run at a frequency such that the ADC conversion clock (ADCK) runs at a frequency within its specified range (f_{ADCK}) after being divided down from the ALTCLK input as determined by the ADIV bits.

ALTCLK is active while the MCU is in wait mode provided the conditions described above are met. This allows ALTCLK to be used as the conversion clock source for the ADC while the MCU is in wait mode.

ALTCLK cannot be used as the ADC conversion clock source while the MCU is in stop3.

10.1.1.3 Hardware Trigger

The RTC on this device can be enabled as a hardware trigger for the ADC module by setting the

ADCSC2[ADTRG] bit. When enabled, the ADC will be triggered every time RTCINT matches RTCMOD. The RTC interrupt does not have to be enabled to trigger the ADC.

The RTC can be configured to cause a hardware trigger in MCU run, wait, and stop3.

10.1.1.4 Analog Pin Enables

The ADC on MC9S08JM60 series contains only two analog pin enable registers, APCTL1 and APCTL2.

10.1.1.5 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 10-1 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{TEMP} - V_{TEMP25}) \div m) \qquad \textit{Eqn. 10-1}$$

where:

- V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.
- V_{TEMP25} is the voltage of the temperature sensor channel at 25°C.
- m is the hot or cold voltage versus temperature slope in V/°C.

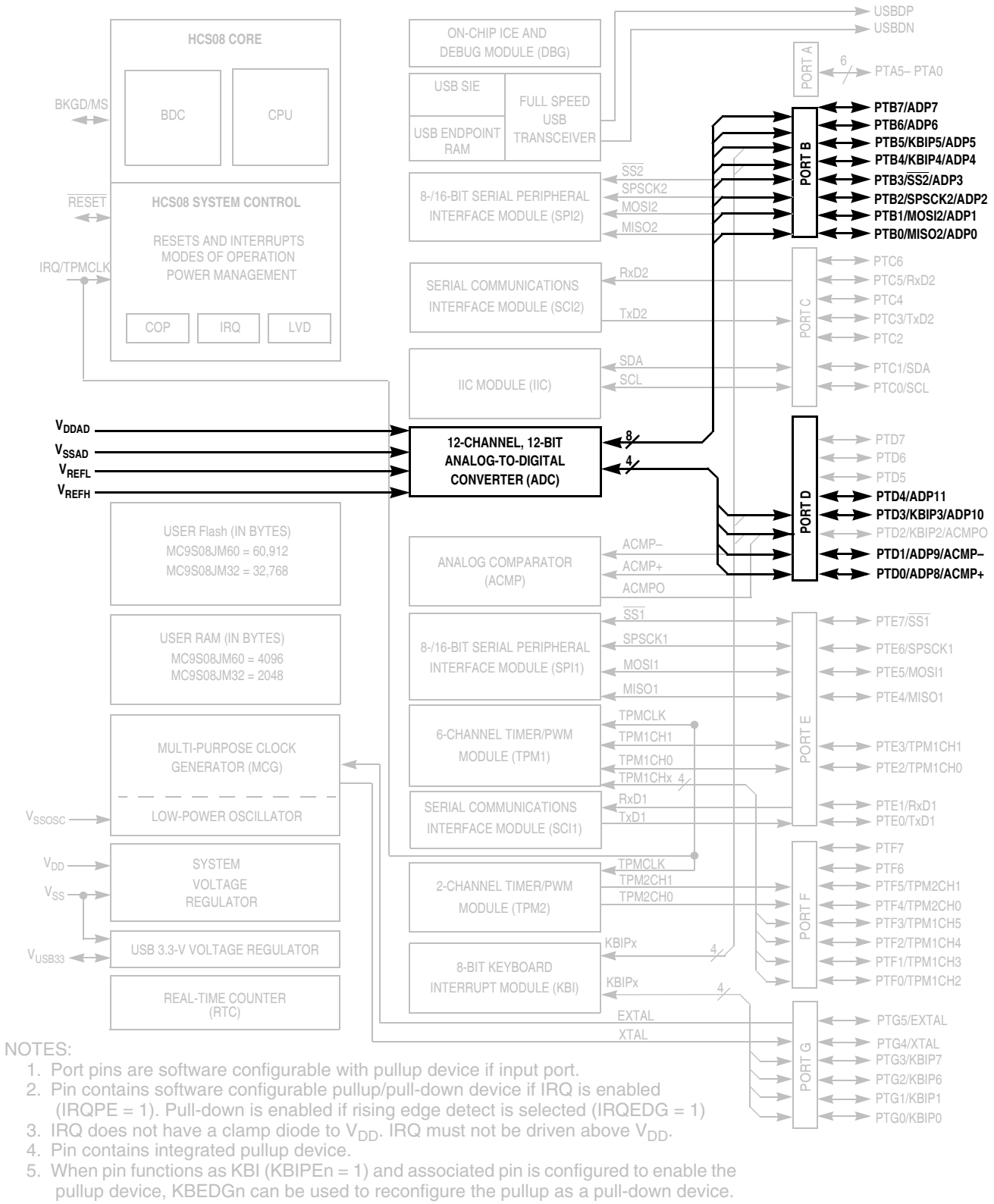
For temperature calculations, use the V_{TEMP25} and m values from the ADC Electricals table.

In application code, the user reads the temperature sensor channel, calculates V_{TEMP}, and compares to V_{TEMP25}. If V_{TEMP} is greater than V_{TEMP25}, the cold slope value is applied in Equation 10-1. If V_{TEMP} is less than V_{TEMP25} the hot slope value is applied in Equation 10-1.

To improve accuracy, calibrate the bandgap voltage reference and temperature sensor. Calibrating at 25 °C will improve accuracy to ±4.5°C. Calibration at 3 points, –40°C, 25°C, and 125°C will improve accuracy to ±2.5°C. Once calibration has been completed, the user will need to calculate the slope for both hot and cold. In application code, the user would then calculate the temperature using Equation 10-1 as detailed above and then determine if the temperature is above or below 25°C. Once determined, if the temperature is above or below 25°C, the user can recalculate the temperature using the hot or cold slope value obtained during calibration.

10.1.2 Low-Power Mode Operation

The ADC is capable of running in stop3 mode but requires LVDSE and LVDE in SPMSC1 to be set.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 10-1. MC9S08JM60 Series Block Diagram Highlighting ADC Block and Pins

10.1.3 Features

Features of the ADC module include:

- Linear successive approximation algorithm with 12-bit resolution
- Up to 28 analog inputs
- Output formatted in 12-, 10-, or 8-bit right-justified unsigned format
- Single or continuous conversion (automatic return to idle after single conversion)
- Configurable sample time and conversion speed/power
- Conversion complete flag and interrupt
- Input clock selectable from up to four sources
- Operation in wait or stop3 modes for lower noise operation
- Asynchronous clock source for lower noise operation
- Selectable asynchronous hardware conversion trigger
- Automatic compare with interrupt for less-than, or greater-than or equal-to, programmable value
- Temperature sensor

10.1.4 ADC Module Block Diagram

Figure 10-2 provides a block diagram of the ADC module.

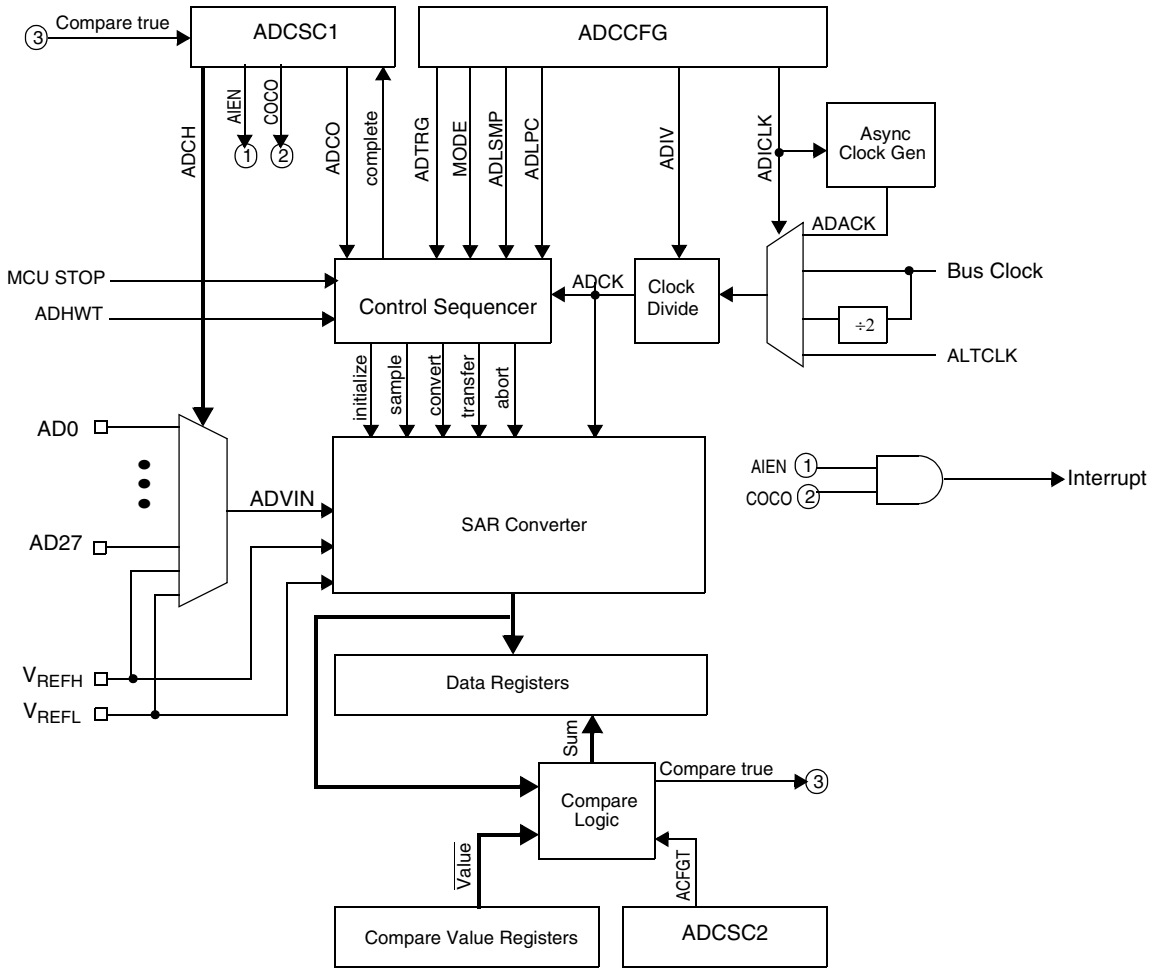


Figure 10-2. ADC Block Diagram

10.2 External Signal Description

The ADC module supports up to 28 separate analog inputs. It also requires four supply/reference/ground connections.

Table 10-2. Signal Properties

Name	Function
AD27–AD0	Analog Channel inputs
V_{REFH}	High reference voltage
V_{REFL}	Low reference voltage
V_{DDAD}	Analog power supply
V_{SSAD}	Analog ground

10.2.1 Analog Power (V_{DDAD})

The ADC analog portion uses V_{DDAD} as its power connection. In some packages, V_{DDAD} is connected internally to V_{DD}. If externally available, connect the V_{DDAD} pin to the same voltage potential as V_{DD}. External filtering may be necessary to ensure clean V_{DDAD} for good results.

10.2.2 Analog Ground (V_{SSAD})

The ADC analog portion uses V_{SSAD} as its ground connection. In some packages, V_{SSAD} is connected internally to V_{SS}. If externally available, connect the V_{SSAD} pin to the same voltage potential as V_{SS}.

10.2.3 Voltage Reference High (V_{REFH})

V_{REFH} is the high reference voltage for the converter. In some packages, V_{REFH} is connected internally to V_{DDAD}. If externally available, V_{REFH} may be connected to the same potential as V_{DDAD} or may be driven by an external source between the minimum V_{DDAD} spec and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}).

10.2.4 Voltage Reference Low (V_{REFL})

V_{REFL} is the low-reference voltage for the converter. In some packages, V_{REFL} is connected internally to V_{SSAD}. If externally available, connect the V_{REFL} pin to the same voltage potential as V_{SSAD}.

10.2.5 Analog Channel Inputs (ADx)

The ADC module supports up to 28 separate analog inputs. An input is selected for conversion through the ADCH channel select bits.

10.3 Register Definition

These memory-mapped registers control and monitor operation of the ADC:

- Status and control register, ADCSC1
- Status and control register, ADCSC2
- Data result registers, ADCRH and ADCRL
- Compare value registers, ADCCVH and ADCCVL
- Configuration register, ADCCFG
- Pin control registers, APCTL1, APCTL2, APCTL3

10.3.1 Status and Control Register 1 (ADCSC1)

This section describes the function of the ADC status and control register (ADCSC1). Writing ADCSC1 aborts the current conversion and initiates a new conversion (if the ADCH bits are equal to a value other than all 1s).

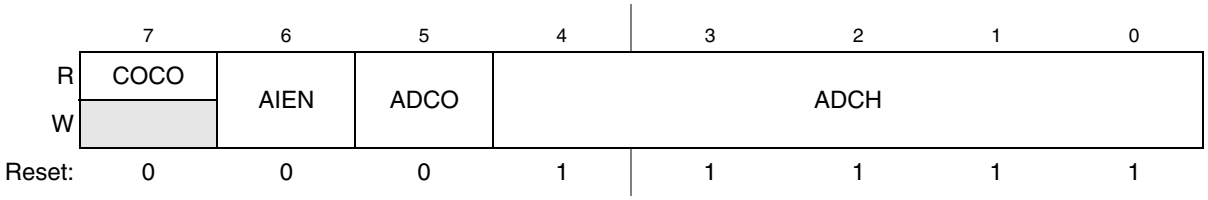


Figure 10-3. Status and Control Register (ADCSC1)

Table 10-3. ADCSC1 Field Descriptions

Field	Description
7 COCO	Conversion Complete Flag. The COCO flag is a read-only bit set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1), the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared when ADCSC1 is written or when ADCRL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	Interrupt Enable AIEN enables conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 ADCO	Continuous Conversion Enable. ADCO enables continuous conversions. 0 One conversion following a write to the ADCSC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected. 1 Continuous conversions initiated following a write to ADCSC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected.
4:0 ADCH	Input Channel Select. The ADCH bits form a 5-bit field that selects one of the input channels. The input channels are detailed in Table 10-4. The successive approximation converter subsystem is turned off when the channel select bits are all set. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional, single conversion from being performed. It is not necessary to set the channel select bits to all ones to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

Table 10-4. Input Channel Select

ADCH	Input Select
00000–01111	AD0–15
10000–11011	AD16–27
11100	Reserved
11101	V_{REFH}
11110	V_{REFL}
11111	Module disabled

10.3.2 Status and Control Register 2 (ADCSC2)

The ADCSC2 register controls the compare function, conversion trigger, and conversion active of the ADC module.

	7	6	5	4	3	2	1	0
R	ADACT	ADTRG	ACFE	ACFGT	0	0	R[1]	R[1]
W								
Reset:	0	0	0	0	0	0	0	0

[1] Bits 1 and 0 are reserved bits that must always be written to 0.

Figure 10-4. Status and Control Register 2 (ADCSC2)

Table 10-5. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	Conversion Active. Indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress 1 Conversion in progress
6 ADTRG	Conversion Trigger Select. Selects the type of trigger used for initiating a conversion. Two types of triggers are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input. 0 Software trigger selected 1 Hardware trigger selected
5 ACFE	Compare Function Enable. Enables the compare function. 0 Compare function disabled 1 Compare function enabled
4 ACFGT	Compare Function Greater Than Enable. Configures the compare function to trigger when the result of the conversion of the input being monitored is greater than or equal to the compare value. The compare function defaults to triggering when the result of the compare of the input being monitored is less than the compare value. 0 Compare triggers when input is less than compare value 1 Compare triggers when input is greater than or equal to compare value

10.3.3 Data Result High Register (ADCRH)

In 12-bit operation, ADCRH contains the upper four bits of the result of a 12-bit conversion. In 10-bit mode, ADCRH contains the upper two bits of the result of a 10-bit conversion. When configured for 10-bit mode, ADR[11:10] are cleared. When configured for 8-bit mode, ADR[11:8] are cleared.

In 12-bit and 10-bit mode, ADCRH is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. When a compare event does occur, the value is the addition of the conversion result and the two's complement of the compare value. In 12-bit and 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion is completed, the intermediate conversion result is lost. In 8-bit mode, there is no interlocking with ADCRL.

If the MODE bits are changed, any data in ADCRH becomes invalid.

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADR11	ADR10	ADR9	ADR8
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-5. Data Result High Register (ADCRH)

10.3.4 Data Result Low Register (ADCRL)

ADCRL contains the lower eight bits of the result of a 12-bit or 10-bit conversion, and all eight bits of an 8-bit conversion. This register is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. In 12-bit and 10-bit mode, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until the after next conversion is completed, the intermediate conversion results are lost. In 8-bit mode, there is no interlocking with ADCRH. If the MODE bits are changed, any data in ADCRL becomes invalid.

	7	6	5	4	3	2	1	0
R	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-6. Data Result Low Register (ADCRL)

10.3.5 Compare Value High Register (ADCCVH)

In 12-bit mode, the ADCCVH register holds the upper four bits of the 12-bit compare value. When the compare function is enabled, these bits are compared to the upper four bits of the result following a conversion in 12-bit mode.

	7	6	5	4	3	2	1	0
R	0	0	0	0	ADCV11	ADCV10	ADCV9	ADCV8
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-7. Compare Value High Register (ADCCVH)

In 10-bit mode, the ADCCVH register holds the upper two bits of the 10-bit compare value (ADCV[9:8]). These bits are compared to the upper two bits of the result following a conversion in 10-bit mode when the compare function is enabled.

In 8-bit mode, ADCCVH is not used during compare.

10.3.6 Compare Value Low Register (ADCCVL)

This register holds the lower 8 bits of the 12-bit or 10-bit compare value or all 8 bits of the 8-bit compare value. When the compare function is enabled, bits ADCV[7:0] are compared to the lower 8 bits of the result following a conversion in 12-bit, 10-bit or 8-bit mode.

	7	6	5	4	3	2	1	0
R W	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
Reset:	0	0	0	0	0	0	0	0

Figure 10-8. Compare Value Low Register (ADCCVL)

10.3.7 Configuration Register (ADCCFG)

ADCCFG selects the mode of operation, clock source, clock divide, and configures for low power and long sample time.

	7	6	5	4	3	2	1	0
R W	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
Reset:	0	0	0	0	0	0	0	0

Figure 10-9. Configuration Register (ADCCFG)

Table 10-6. ADCCFG Register Field Descriptions

Field	Description
7 ADLPC	Low-Power Configuration. ADLPC controls the speed and power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration:The power is reduced at the expense of maximum clock speed.
6:5 ADIV	Clock Divide Select. ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. Table 10-7 shows the available clock configurations.
4 ADLSMP	Long Sample Time Configuration. ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 0 Short sample time 1 Long sample time

Table 10-6. ADCCFG Register Field Descriptions (continued)

Field	Description
3:2 MODE	Conversion Mode Selection. MODE bits are used to select between 12-, 10-, or 8-bit operation. See Table 10-8.
1:0 ADICLK	Input Clock Select. ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 10-9.

Table 10-7. Clock Divide Select

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock ÷ 2
10	4	Input clock ÷ 4
11	8	Input clock ÷ 8

Table 10-8. Conversion Modes

MODE	Mode Description
00	8-bit conversion (N=8)
01	12-bit conversion (N=12)
10	10-bit conversion (N=10)
11	Reserved

Table 10-9. Input Clock Select

ADICLK	Selected Clock Source
00	Bus clock
01	Bus clock divided by 2
10	Alternate clock (ALTCLK)
11	Asynchronous clock (ADACK)

10.3.8 Pin Control 1 Register (APCTL1)

The pin control registers disable the I/O port control of MCU pins used as analog inputs. APCTL1 is used to control the pins associated with channels 0–7 of the ADC module.

	7	6	5	4	3	2	1	0
R W	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
Reset:	0	0	0	0	0	0	0	0

Figure 10-10. Pin Control 1 Register (APCTL1)

Table 10-10. APCTL1 Register Field Descriptions

Field	Description
7 ADPC7	ADC Pin Control 7. ADPC7 controls the pin associated with channel AD7. 0 AD7 pin I/O control enabled 1 AD7 pin I/O control disabled
6 ADPC6	ADC Pin Control 6. ADPC6 controls the pin associated with channel AD6. 0 AD6 pin I/O control enabled 1 AD6 pin I/O control disabled
5 ADPC5	ADC Pin Control 5. ADPC5 controls the pin associated with channel AD5. 0 AD5 pin I/O control enabled 1 AD5 pin I/O control disabled
4 ADPC4	ADC Pin Control 4. ADPC4 controls the pin associated with channel AD4. 0 AD4 pin I/O control enabled 1 AD4 pin I/O control disabled
3 ADPC3	ADC Pin Control 3. ADPC3 controls the pin associated with channel AD3. 0 AD3 pin I/O control enabled 1 AD3 pin I/O control disabled
2 ADPC2	ADC Pin Control 2. ADPC2 controls the pin associated with channel AD2. 0 AD2 pin I/O control enabled 1 AD2 pin I/O control disabled
1 ADPC1	ADC Pin Control 1. ADPC1 controls the pin associated with channel AD1. 0 AD1 pin I/O control enabled 1 AD1 pin I/O control disabled
0 ADPC0	ADC Pin Control 0. ADPC0 controls the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

10.3.9 Pin Control 2 Register (APCTL2)

APCTL2 controls channels 8–15 of the ADC module.

	7	6	5	4	3	2	1	0
R W	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
Reset:	0	0	0	0	0	0	0	0

Figure 10-11. Pin Control 2 Register (APCTL2)

Table 10-11. APCTL2 Register Field Descriptions

Field	Description
7 ADPC15	ADC Pin Control 15. ADPC15 controls the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	ADC Pin Control 14. ADPC14 controls the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control disabled
5 ADPC13	ADC Pin Control 13. ADPC13 controls the pin associated with channel AD13. 0 AD13 pin I/O control enabled 1 AD13 pin I/O control disabled
4 ADPC12	ADC Pin Control 12. ADPC12 controls the pin associated with channel AD12. 0 AD12 pin I/O control enabled 1 AD12 pin I/O control disabled
3 ADPC11	ADC Pin Control 11. ADPC11 controls the pin associated with channel AD11. 0 AD11 pin I/O control enabled 1 AD11 pin I/O control disabled
2 ADPC10	ADC Pin Control 10. ADPC10 controls the pin associated with channel AD10. 0 AD10 pin I/O control enabled 1 AD10 pin I/O control disabled
1 ADPC9	ADC Pin Control 9. ADPC9 controls the pin associated with channel AD9. 0 AD9 pin I/O control enabled 1 AD9 pin I/O control disabled
0 ADPC8	ADC Pin Control 8. ADPC8 controls the pin associated with channel AD8. 0 AD8 pin I/O control enabled 1 AD8 pin I/O control disabled

10.3.10 Pin Control 3 Register (APCTL3)

APCTL3 controls channels 16–23 of the ADC module.

	7	6	5	4	3	2	1	0
R W	ADPC23	ADPC22	ADPC21	ADPC20	ADPC19	ADPC18	ADPC17	ADPC16
Reset:	0	0	0	0	0	0	0	0

Figure 10-12. Pin Control 3 Register (APCTL3)

Table 10-12. APCTL3 Register Field Descriptions

Field	Description
7 ADPC23	ADC Pin Control 23. ADPC23 controls the pin associated with channel AD23. 0 AD23 pin I/O control enabled 1 AD23 pin I/O control disabled
6 ADPC22	ADC Pin Control 22. ADPC22 controls the pin associated with channel AD22. 0 AD22 pin I/O control enabled 1 AD22 pin I/O control disabled
5 ADPC21	ADC Pin Control 21. ADPC21 controls the pin associated with channel AD21. 0 AD21 pin I/O control enabled 1 AD21 pin I/O control disabled
4 ADPC20	ADC Pin Control 20. ADPC20 controls the pin associated with channel AD20. 0 AD20 pin I/O control enabled 1 AD20 pin I/O control disabled
3 ADPC19	ADC Pin Control 19. ADPC19 controls the pin associated with channel AD19. 0 AD19 pin I/O control enabled 1 AD19 pin I/O control disabled
2 ADPC18	ADC Pin Control 18. ADPC18 controls the pin associated with channel AD18. 0 AD18 pin I/O control enabled 1 AD18 pin I/O control disabled
1 ADPC17	ADC Pin Control 17. ADPC17 controls the pin associated with channel AD17. 0 AD17 pin I/O control enabled 1 AD17 pin I/O control disabled
0 ADPC16	ADC Pin Control 16. ADPC16 controls the pin associated with channel AD16. 0 AD16 pin I/O control enabled 1 AD16 pin I/O control disabled

10.4 Functional Description

The ADC module is disabled during reset or when the ADCH bits are all high. The module is idle when a conversion has completed and another conversion has not been initiated. When idle, the module is in its lowest power state.

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. In 12-bit and 10-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 12-bit digital result. In 8-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 9-bit digital result.

When the conversion is completed, the result is placed in the data registers (ADCRH and ADCRL). In 10-bit mode, the result is rounded to 10 bits and placed in the data registers (ADCRH and ADCRL). In 8-bit mode, the result is rounded to 8 bits and placed in ADCRL. The conversion complete flag (COCO) is then set and an interrupt is generated if the conversion complete interrupt has been enabled (AIEN = 1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of its compare registers. The compare function is enabled by setting the ACFE bit and operates with any of the conversion modes and configurations.

10.4.1 Clock Select and Divide Control

One of four clock sources can be selected as the clock source for the ADC module. This clock source is then divided by a configurable value to generate the input clock to the converter (ADCK). The clock is selected from one of the following sources by means of the ADICLK bits.

- The bus clock, which is equal to the frequency at which software is executed. This is the default selection following reset.
- The bus clock divided by two. For higher bus clock rates, this allows a maximum divide by 16 of the bus clock.
- ALTCLK, as defined for this MCU (See module section introduction).
- The asynchronous clock (ADACK). This clock is generated from a clock source within the ADC module. When selected as the clock source, this clock remains active while the MCU is in wait or stop3 mode and allows conversions in these modes for lower noise operation.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC do not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

10.4.2 Input Select and Pin Control

The pin control registers (APCTL3, APCTL2, and APCTL1) disable the I/O port control of the pins used as analog inputs.When a pin control register bit is set, the following conditions are forced for the associated MCU pin:

- The output buffer is forced to its high impedance state.
- The input buffer is disabled. A read of the I/O port returns a zero for any pin with its input buffer disabled.
- The pullup is disabled.

10.4.3 Hardware Trigger

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when the ADTRG bit is set. This source is not available on all MCUs. Consult the module introduction for information on the ADHWT source specific to this MCU.

When ADHWT source is available and hardware trigger is enabled (ADTRG=1), a conversion is initiated on the rising edge of ADHWT. If a conversion is in progress when a rising edge occurs, the rising edge is ignored. In continuous convert configuration, only the initial rising edge to launch continuous conversions is observed. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

10.4.4 Conversion Control

Conversions can be performed in 12-bit mode, 10-bit mode, or 8-bit mode as determined by the MODE bits. Conversions can be initiated by a software or hardware trigger. In addition, the ADC module can be

configured for low power operation, long sample time, continuous conversion, and automatic compare of the conversion result to a software determined compare value.

10.4.4.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADCSC1 (with ADCH bits not all 1s) if software triggered operation is selected.
- Following a hardware trigger (ADHWT) event if hardware triggered operation is selected.
- Following the transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADCSC1 is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

10.4.4.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADCRH and ADCRL. This is indicated by the setting of COCO. An interrupt is generated if AIEN is high at the time that COCO is set.

A blocking mechanism prevents a new result from overwriting previous data in ADCRH and ADCRL if the previous data is in the process of being read while in 12-bit or 10-bit MODE (the ADCRH register has been read but the ADCRL register has not). When blocking is active, the data transfer is blocked, COCO is not set, and the new result is lost. In the case of single conversions with the compare function enabled and the compare condition false, blocking has no effect and ADC operation is terminated. In all other cases of operation, when a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

If single conversions are enabled, the blocking mechanism could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

10.4.4.3 Aborting Conversions

Any conversion in progress is aborted when:

- A write to ADCSC1 occurs (the current conversion will be aborted and a new conversion will be initiated, if ADCH are not all 1s).
- A write to ADCSC2, ADCCFG, ADCCVH, or ADCCVL occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.
- The MCU is reset.
- The MCU enters stop mode with ADACK not enabled.

When a conversion is aborted, the contents of the data registers, ADCRH and ADCRL, are not altered. However, they continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset, ADCRH and ADCRL return to their reset states.

10.4.4.4 Power Control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source, the ADACK clock generator is also enabled.

Power consumption when active can be reduced by setting ADLPC. This results in a lower maximum value for f_{ADCK} (see the electrical specifications).

10.4.4.5 Sample Time and Total Conversion Time

The total conversion time depends on the sample time (as determined by ADLSMP), the MCU bus frequency, the conversion mode (8-bit, 10-bit or 12-bit), and the frequency of the conversion clock (f_{ADCK}). After the module becomes active, sampling of the input begins. ADLSMP selects between short (3.5 ADCK cycles) and long (23.5 ADCK cycles) sample times.When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The result of the conversion is transferred to ADCRH and ADCRL upon completion of the conversion algorithm.

If the bus frequency is less than the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when short sample is enabled (ADLSMP=0). If the bus frequency is less than 1/11th of the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed when long sample is enabled (ADLSMP=1).

The maximum total conversion time for different conditions is summarized in Table 10-13.

Table 10-13. Total Conversion Time vs. Control Conditions

Conversion Type	ADICLK	ADLSMP	Max Total Conversion Time
Single or first continuous 8-bit	0x, 10	0	20 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	0	23 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	0x, 10	1	40 ADCK cycles + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	0x, 10	1	43 ADCK cycles + 5 bus clock cycles
Single or first continuous 8-bit	11	0	5 μs + 20 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	0	5 μs + 23 ADCK + 5 bus clock cycles
Single or first continuous 8-bit	11	1	5 μs + 40 ADCK + 5 bus clock cycles
Single or first continuous 10-bit or 12-bit	11	1	5 μs + 43 ADCK + 5 bus clock cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	17 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}$	xx	0	20 ADCK cycles
Subsequent continuous 8-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	37 ADCK cycles
Subsequent continuous 10-bit or 12-bit; $f_{BUS} \geq f_{ADCK}/11$	xx	1	40 ADCK cycles

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK bits, and the divide ratio is specified by the ADIV bits. For example, in 10-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 8 MHz, then the conversion time for a single conversion is:

$$\text{Conversion time} = \frac{\text{23 ADCK Cyc}}{\text{8 MHz/1}} + \frac{\text{5 bus Cyc}}{\text{8 MHz}} = 3.5\ \mu\text{s}$$

Number of bus cycles = 3.5 μs x 8 MHz = 28 cycles

NOTE

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

10.4.5 Automatic Compare Function

The compare function can be configured to check for an upper or lower limit. After the input is sampled and converted, the result is added to the two's complement of the compare value (ADCCVH and ADCCVL). When comparing to an upper limit (ACFGT = 1), if the result is greater-than or equal-to the compare value, COCO is set. When comparing to a lower limit (ACFGT = 0), if the result is less than the compare value, COCO is set. The value generated by the addition of the conversion result and the two's complement of the compare value is transferred to ADCRH and ADCRL.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCO is not set and no data is transferred to the result registers. An ADC interrupt is generated upon the setting of COCO if the ADC interrupt is enabled (AIEN = 1).

NOTE

The compare function can monitor the voltage on a channel while the MCU is in wait or stop3 mode. The ADC interrupt wakes the MCU when the compare condition is met.

10.4.6 MCU Wait Mode Operation

Wait mode is a lower power-consumption standby mode from which recovery is fast because the clock sources remain active. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, and ADACK are available as conversion clock sources while in wait mode. The use of ALTCLK as the conversion clock source in wait is dependent on the definition of ALTCLK for this MCU. Consult the module introduction for information on ALTCLK specific to this MCU.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from wait mode if the ADC interrupt is enabled (AIEN = 1).

10.4.7 MCU Stop3 Mode Operation

Stop mode is a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

10.4.7.1 Stop3 Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its idle state. The contents of ADCRH and ADCRL are unaffected by stop3 mode. After exiting from stop3 mode, a software or hardware trigger is required to resume conversions.

10.4.7.2 Stop3 Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop3 mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop3 mode. Consult the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters stop3 mode, it continues until completion. Conversions can be initiated while the MCU is in stop3 mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from stop3 mode if the ADC interrupt is enabled (AIEN = 1).

NOTE

The ADC module can wake the system from low-power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure the data transfer blocking mechanism (discussed in Section 10.4.4.2, "Completing Conversions,") is cleared when entering stop3 and continuing ADC conversions.

10.4.8 MCU Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters stop2 mode. All module registers contain their reset values following exit from stop2. Therefore, the module must be re-enabled and re-configured following exit from stop2.

10.5 Initialization Information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. You can configure the module for 8-, 10-, or 12-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to Table 10-7, Table 10-8, and Table 10-9 for information used in this example.

NOTE

Hexadecimal values designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

10.5.1 ADC Module Initialization Example

10.5.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.
2. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.
3. Update status and control register 1 (ADCSC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

10.5.1.2 Pseudo-Code Example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion at low power with a long sample time on input channel 1, where the internal ADCK clock is derived from the bus clock divided by 1.

ADCCFG = 0x98 (%10011000)

```
Bit 7    ADLPC    1        Configures for low power (lowers maximum clock speed)
Bit 6:5  ADIV     00       Sets the ADCK to the input clock ÷ 1
Bit 4    ADLSMP   1        Configures for long sample time
Bit 3:2  MODE     10       Sets mode at 10-bit conversions
Bit 1:0  ADICLK   00       Selects bus clock as input clock source
```

ADCSC2 = 0x00 (%00000000)

```
Bit 7    ADACT    0        Flag indicates if a conversion is in progress
Bit 6    ADTRG    0        Software trigger selected
Bit 5    ACFE     0        Compare function disabled
Bit 4    ACFGT    0        Not used in this example
Bit 3:2           00       Reserved, always reads zero
Bit 1:0           00       Reserved for Freescale's internal use; always write zero
```

ADCSC1 = 0x41 (%01000001)

```
Bit 7    COCO     0        Read-only flag which is set when a conversion completes
Bit 6    AIEN     1        Conversion complete interrupt enabled
Bit 5    ADCO     0        One conversion only (continuous conversions disabled)
Bit 4:0  ADCH     00001    Input channel 1 selected as ADC input channel
```

ADCRH/L = 0xxx

```
Holds results of conversion. Read high byte (ADCRH) before low byte (ADCRL) so that
conversion data cannot be overwritten with data from the next conversion.
```

ADCCVH/L = 0xxx

Holds compare value when compare function enabled

APCTL1=0x02

AD1 pin I/O control disabled. All other AD pins remain general purpose I/O pins

APCTL2=0x00

All other AD pins remain general purpose I/O pins

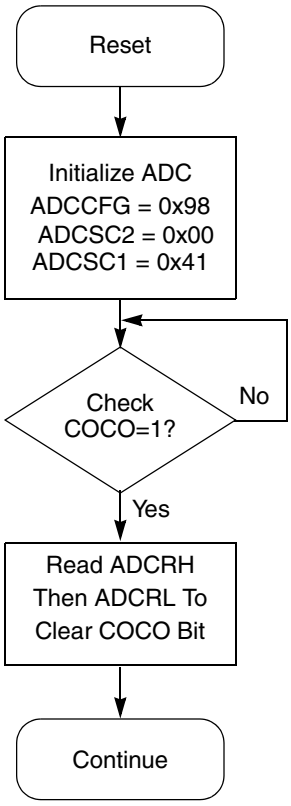


Figure 10-13. Initialization Flowchart for Example

10.6 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

10.6.1 External Pins and Routing

The following sections discuss the external pins associated with the ADC module and how they should be used for best results.

10.6.1.1 Analog Supply Pins

The ADC module has analog power and ground supplies (V_{DDAD} and V_{SSAD}) available as separate pins on some devices. V_{SSAD} is shared on the same pin as the MCU digital V_{SS} on some devices. On other devices, V_{SSAD} and V_{DDAD} are shared with the MCU digital supply pins. In these cases, there are separate pads for the analog supplies bonded to the same pin as the corresponding digital supply so that some degree of isolation between the supplies is maintained.

When available on a separate pin, both V_{DDAD} and V_{SSAD} must be connected to the same voltage potential as their corresponding MCU digital supply (V_{DD} and V_{SS}) and must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

If separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the V_{SSAD} pin. This should be the only ground connection between these supplies if possible. The V_{SSAD} pin makes a good single point ground location.

10.6.1.2 Analog Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs. The high reference is V_{REFH}, which may be shared on the same pin as V_{DDAD} on some devices. The low reference is V_{REFL}, which may be shared on the same pin as V_{SSAD} on some devices.

When available on a separate pin, V_{REFH} may be connected to the same potential as V_{DDAD}, or may be driven by an external source between the minimum V_{DDAD} spec and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}). When available on a separate pin, V_{REFL} must be connected to the same voltage potential as V_{SSAD}. V_{REFH} and V_{REFL} must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 μF capacitor with good high frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current causes a voltage drop that could result in conversion errors. Inductance in this path must be minimum (parasitic only).

10.6.1.3 Analog Input Pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer is in its high impedance state and the pullup is disabled. Also, the input buffer draws DC current when its input is not at V_{DD} or V_{SS}. Setting the pin control register bits for all pins used as analog inputs should be done to achieve lowest operating current.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 μF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used they must be placed as near as possible to the package pins and be referenced to V_{SSA}.

For proper conversion, the input voltage must fall between V_{REFH} and V_{REFL}. If the input is equal to or exceeds V_{REFH}, the converter circuit converts the signal to 0xFFF (full scale 12-bit representation), 0x3FF (full scale 10-bit representation) or 0xFF (full scale 8-bit representation). If the input is equal to or less than V_{REFL}, the converter circuit converts it to 0x000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. There is a brief current associated with V_{REFL} when the sampling capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins should not be transitioning during conversions.

10.6.2 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

10.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7kΩ and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 12-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below 2 kΩ.

Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP (to increase the sample window to 23.5 cycles) or decreasing ADCK frequency to increase sample time.

10.6.2.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than V_{DDAD} / (2^N*I_{LEAK}) for less than 1/4LSB leakage error (N = 8 in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

10.6.2.3 Noise-Induced Errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 μF low-ESR capacitor from V_{REFH} to V_{REFL}.
- There is a 0.1 μF low-ESR capacitor from V_{DDAD} to V_{SSAD}.
- If inductive isolation is used from the primary supply, an additional 1 μF capacitor is placed from V_{DDAD} to V_{SSAD}.
- V_{SSAD} (and V_{REFL}, if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or stop3 mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
  - For software triggered conversions, immediately follow the write to ADCSC1 with a wait instruction or stop instruction.

— For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces V_{DD} noise but increases effective conversion time due to stop recovery.

- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop3 or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01 μF capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSAD} (this improves noise issues, but affects the sample rate based on the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

10.6.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1\ lsb = (V_{REFH} - V_{REFL}) / 2^N \qquad \textit{Eqn. 10-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2 lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

For 12-bit conversions the code transitions only after the full code width is present, so the quantization error is –1 lsb to 0 lsb and the code width of each step is 1 lsb.

10.6.2.5 Linearity Errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the system should be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 lsb in 8-bit or 10-bit modes and 1 lsb in 12-bit mode). If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 lsb) is used.
- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5 lsb in 8-bit or 10-bit modes and 1LSB in 12-bit mode). If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1LSB) is used.

- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

10.6.2.6 Code Jitter, Non-Monotonicity, and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 lsb in 8-bit or 10-bit mode, or around 2 lsb in 12-bit mode, and increases with noise.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 10.6.2.3 reduces this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

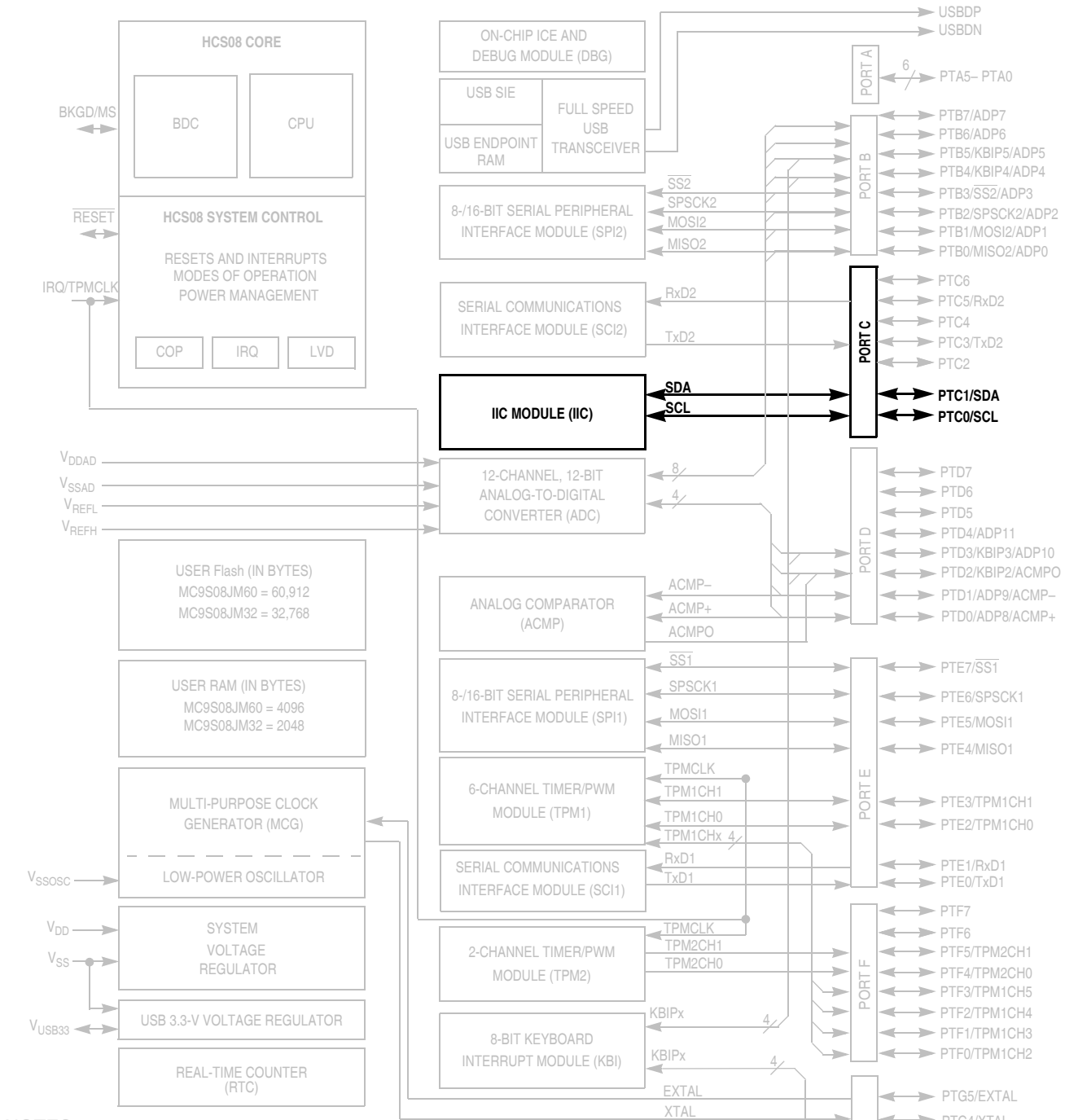
Chapter 11
Inter-Integrated Circuit (S08IICV2)

11.1 Introduction

The MC9S08JM60 series of microcontrollers has an inter-integrated circuit (IIC) module for communication with other integrated circuits. The two pins associated with this module, SCL and SDA, are shared with PTC0 and PTC1, respectively.

NOTE

MC9S08JM60 series devices operate at a higher voltage range (2.7 V to 5.5 V) and do not include stop1 mode. Therefore, please disregard references to stop1.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pull-up as a pull-down device.

Figure 11-1. MC9S08JM60 Series Block Diagram Highlighting the IIC Block and Pins

11.1.1 Features

The IIC includes these distinctive features:

- Compatible with IIC bus standard
- Multi-master operation
- Software programmable for one of 64 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
- General call recognition
- 10-bit address extension

11.1.2 Modes of Operation

A brief description of the IIC in the various MCU modes is given here.

- **Run mode** — This is the basic mode of operation. To conserve power in this mode, disable the module.
- **Wait mode** — The module continues to operate while the MCU is in wait mode and can provide a wake-up interrupt.
- **Stop mode** — The IIC is inactive in stop3 mode for reduced power consumption. The stop instruction does not affect IIC register states. Stop2 resets the register contents.

11.1.3 Block Diagram

Figure 11-2 is a block diagram of the IIC.

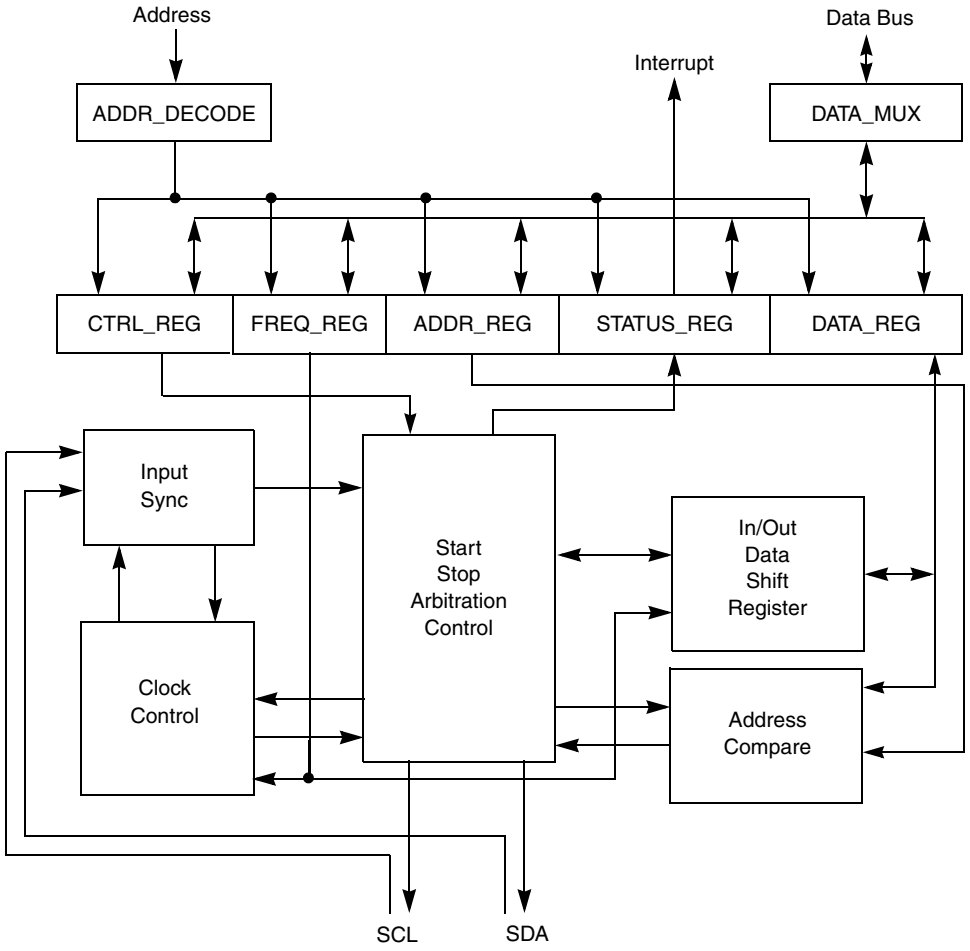


Figure 11-2. IIC Functional Block Diagram

11.2 External Signal Description

This section describes each user-accessible pin signal.

11.2.1 SCL — Serial Clock Line

The bidirectional SCL is the serial clock line of the IIC system.

11.2.2 SDA — Serial Data Line

The bidirectional SDA is the serial data line of the IIC system.

11.3 Register Definition

This section consists of the IIC register descriptions in address order.

Refer to the direct-page register summary in the memory chapter of this document for the absolute address assignments for all IIC registers. This section refers to registers and control bits only by their names. A

Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

11.3.1 IIC Address Register (IICA)

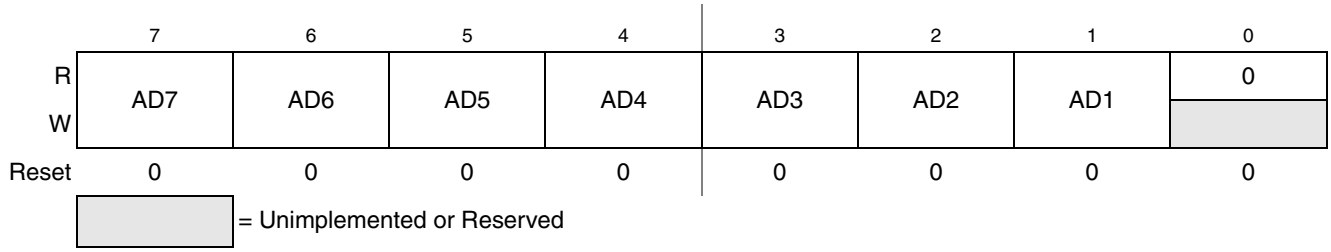


Figure 11-3. IIC Address Register (IICA)

Table 11-1. IICA Field Descriptions

Field	Description
7–1 AD[7:1]	**Slave Address.** The AD[7:1] field contains the slave address to be used by the IIC module. This field is used on the 7-bit address scheme and the lower seven bits of the 10-bit address scheme.

11.3.2 IIC Frequency Divider Register (IICF)

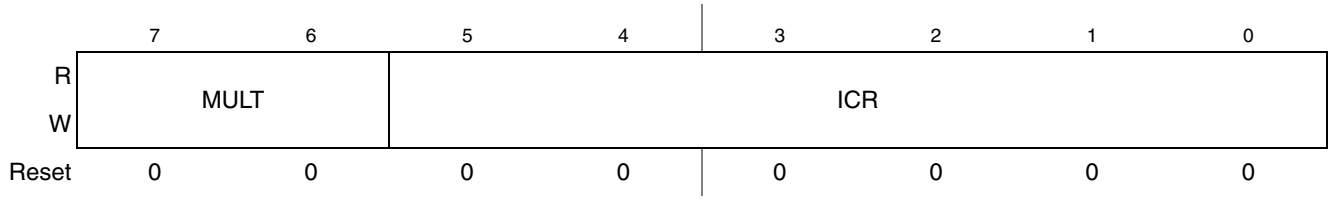


Figure 11-4. IIC Frequency Divider Register (IICF)

Table 11-2. IICF Field Descriptions

Field	Description
7–6 MULT	**IIC Multiplier Factor**. The MULT bits define the multiplier factor, mul. This factor, along with the SCL divider, generates the IIC baud rate. The multiplier factor mul as defined by the MULT bits is provided below. 00 mul = 01 01 mul = 02 10 mul = 04 11 Reserved
5–0 ICR	**IIC Clock Rate**. The ICR bits are used to prescale the bus clock for bit rate selection. These bits and the MULT bits determine the IIC baud rate, the SDA hold time, the SCL Start hold time, and the SCL Stop hold time. Table 11-4 provides the SCL divider and hold values for corresponding values of the ICR. The SCL divider multiplied by multiplier factor mul generates IIC baud rate. $\text{IIC baud rate} = \frac{\text{bus speed (Hz)}}{\text{mul} \times \text{SCLdivider}}$ ***Eqn. 11-1*** SDA hold time is the delay from the falling edge of SCL (IIC clock) to the changing of SDA (IIC data). SDA hold time = bus period (s) × mul × SDA hold value ***Eqn. 11-2*** SCL start hold time is the delay from the falling edge of SDA (IIC data) while SCL is high (Start condition) to the falling edge of SCL (IIC clock). SCL Start hold time = bus period (s) × mul × SCL Start hold value ***Eqn. 11-3*** SCL stop hold time is the delay from the rising edge of SCL (IIC clock) to the rising edge of SDA SDA (IIC data) while SCL is high (Stop condition). SCL Stop hold time = bus period (s) × mul × SCL Stop hold value ***Eqn. 11-4***

For example, if the bus speed is 8 MHz, the table below shows the possible hold time values with different ICR and MULT selections to achieve an IIC baud rate of 100kbps.

Table 11-3. Hold Time Values for 8 MHz Bus Speed

MULT	ICR	Hold Times (μs)		
		SDA	SCL Start	SCL Stop
0x2	0x00	3.500	3.000	5.500
0x1	0x07	2.500	4.000	5.250
0x1	0x0B	2.250	4.000	5.250
0x0	0x14	2.125	4.250	5.125
0x0	0x18	1.125	4.750	5.125

Table 11-4. IIC Divider and Hold Values

ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SCL Hold (Stop) Value
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
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

ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SCL Hold (Stop) Value
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
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
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

11.3.3 IIC Control Register (IICC1)

	7	6	5	4	3	2	1	0
R	IICEN	IICIE	MST	TX	TXAK	0	0	0
W						RSTA		
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 11-5. IIC Control Register (IICC1)

Table 11-5. IICC1 Field Descriptions

Field	Description
7 IICEN	**IIC Enable.** The IICEN bit determines whether the IIC module is enabled. 0 IIC is not enabled 1 IIC is enabled
6 IICIE	**IIC Interrupt Enable.** The IICIE bit determines whether an IIC interrupt is requested. 0 IIC interrupt request not enabled 1 IIC interrupt request enabled
5 MST	**Master Mode Select.** The MST bit changes from a 0 to a 1 when a start signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0 a stop signal is generated and the mode of operation changes from master to slave. 0 Slave mode 1 Master mode
4 TX	**Transmit Mode Select.** The TX bit selects the direction of master and slave transfers. In master mode, this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit is always high. When addressed as a slave, this bit should be set by software according to the SRW bit in the status register. 0 Receive 1 Transmit
3 TXAK	**Transmit Acknowledge Enable.** This bit specifies the value driven onto the SDA during data acknowledge cycles for master and slave receivers. 0 An acknowledge signal is sent out to the bus after receiving one data byte 1 No acknowledge signal response is sent
2 RSTA	**Repeat start.** Writing a 1 to this bit generates a repeated start condition provided it is the current master. This bit is always read as cleared. Attempting a repeat at the wrong time results in loss of arbitration.

11.3.4 IIC Status Register (IICS)

	7	6	5	4	3	2	1	0
R	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
W								
Reset	1	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 11-6. IIC Status Register (IICS)

Table 11-6. IICS Field Descriptions

Field	Description
7 TCF	**Transfer Complete Flag.** This bit is set on the completion of a byte transfer. This bit is only valid during or immediately following a transfer to the IIC module or from the IIC module.The TCF bit is cleared by reading the IICD register in receive mode or writing to the IICD in transmit mode. 0 Transfer in progress 1 Transfer complete
6 IAAS	**Addressed as a Slave.** The IAAS bit is set when the calling address matches the programmed slave address or when the GCAEN bit is set and a general call is received. Writing the IICC register clears this bit. 0 Not addressed 1 Addressed as a slave
5 BUSY	**Bus Busy.** The BUSY bit indicates the status of the bus regardless of slave or master mode. The BUSY bit is set when a start signal is detected and cleared when a stop signal is detected. 0 Bus is idle 1 Bus is busy
4 ARBL	**Arbitration Lost.** This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software by writing a 1 to it. 0 Standard bus operation 1 Loss of arbitration
2 SRW	**Slave Read/Write.** When addressed as a slave, the SRW bit indicates the value of the R/W command bit of the calling address sent to the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
1 IICIF	**IIC Interrupt Flag.** The IICIF bit is set when an interrupt is pending. This bit must be cleared by software, by writing a 1 to it in the interrupt routine. One of the following events can set the IICIF bit: • One byte transfer completes • Match of slave address to calling address • Arbitration lost 0 No interrupt pending 1 Interrupt pending
0 RXAK	**Receive Acknowledge**. When the RXAK bit is low, it indicates an acknowledge signal has been received after the completion of one byte of data transmission on the bus. If the RXAK bit is high it means that no acknowledge signal is detected. 0 Acknowledge received 1 No acknowledge received

11.3.5 IIC Data I/O Register (IICD)

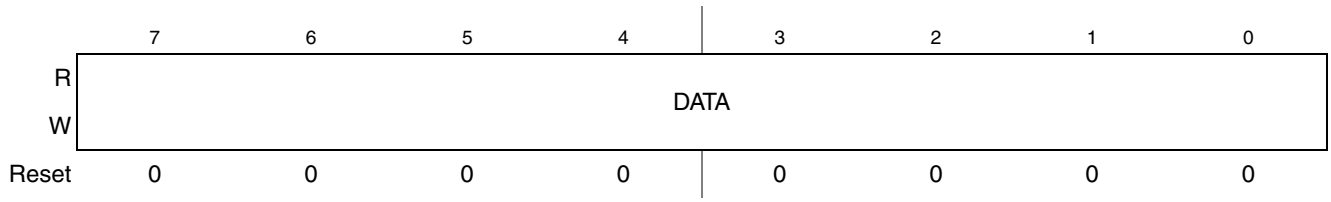


Figure 11-7. IIC Data I/O Register (IICD)

Table 11-7. IICD Field Descriptions

Field	Description
7–0 DATA	**Data** — In master transmit mode, when data is written to the IICD, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates receiving of the next byte of data.

NOTE

When transitioning out of master receive mode, the IIC mode should be switched before reading the IICD register to prevent an inadvertent initiation of a master receive data transfer.

In slave mode, the same functions are available after an address match has occurred.

The TX bit in IICC must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the IIC is configured for master transmit but a master receive is desired, reading the IICD does not initiate the receive.

Reading the IICD returns the last byte received while the IIC is configured in master receive or slave receive modes. The IICD does not reflect every byte transmitted on the IIC bus, nor can software verify that a byte has been written to the IICD correctly by reading it back.

In master transmit mode, the first byte of data written to IICD following assertion of MST is used for the address transfer and should comprise of the calling address (in bit 7 to bit 1) concatenated with the required R/$\overline{W}$ bit (in position bit 0).

11.3.6 IIC Control Register 2 (IICC2)

	7	6	5	4	3	2	1	0
R	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 11-8. IIC Control Register (IICC2)

Table 11-8. IICC2 Field Descriptions

Field	Description
7 GCAEN	**General Call Address Enable.** The GCAEN bit enables or disables general call address. 0 General call address is disabled 1 General call address is enabled
6 ADEXT	**Address Extension.** The ADEXT bit controls the number of bits used for the slave address. 0 7-bit address scheme 1 10-bit address scheme
2–0 AD[10:8]	Slave Address. The AD[10:8] field contains the upper three bits of the slave address in the 10-bit address scheme. This field is only valid when the ADEXT bit is set.

11.4 Functional Description

This section provides a complete functional description of the IIC module.

11.4.1 IIC Protocol

The IIC 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 logic 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
- Stop signal

The stop signal should not be confused with the CPU stop instruction. The IIC bus system communication is described briefly in the following sections and illustrated in Figure 11-9.

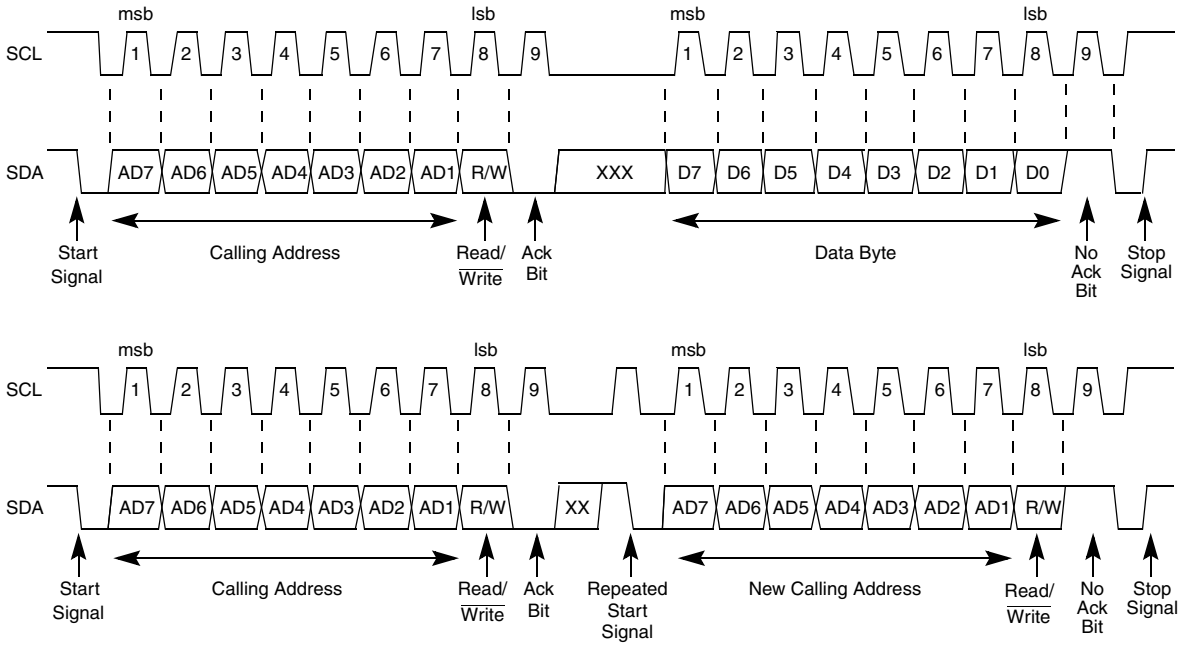


Figure 11-9. IIC Bus Transmission Signals

11.4.1.1 Start Signal

When the bus is free, no master device is engaging the bus (SCL and SDA lines are at logical high), a master may initiate communication by sending a start signal. As shown in Figure 11-9, 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.

11.4.1.2 Slave Address Transmission

The first byte of data transferred immediately after the start signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/$\overline{W}$ bit. The R/$\overline{W}$ 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 responds by sending back an acknowledge bit. This is done by pulling the SDA low at the ninth clock (see Figure 11-9).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address equal to its own slave address. The IIC cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the IIC reverts to slave mode and operates correctly even if it is being addressed by another master.

11.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/$\overline{W}$ 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 11-9. There is one clock pulse on SCL for each data bit, the msb being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the ninth bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets this as an end of data transfer and releases the SDA line.

In either case, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a stop signal.
- Commences a new calling by generating a repeated start signal.

11.4.1.4 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 at logical 1 (see Figure 11-9).

The master can generate a stop even if the slave has generated an acknowledge at which point the slave must release the bus.

11.4.1.5 Repeated Start Signal

As shown in Figure 11-9, 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.

11.4.1.6 Arbitration Procedure

The IIC 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 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.

11.4.1.7 Clock Synchronization

Because 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 after 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 11-10). 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.

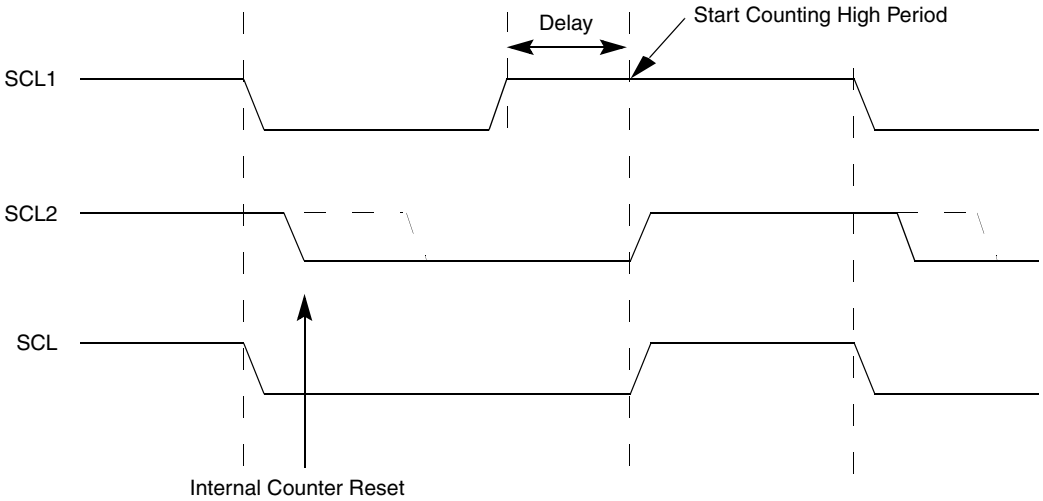


Figure 11-10. IIC Clock Synchronization

11.4.1.8 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 a case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

11.4.1.9 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.

11.4.2 10-bit Address

For 10-bit addressing, 0x11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

11.4.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 11-9). When a 10-bit address follows a start condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/$\overline{W}$ direction bit) is 0. More than one device can find a match and generate an acknowledge (A1). Then, each slave that finds a match compares the eight bits of the second byte of the slave address with its own address. Only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Data	A	...	Data	A/A	P

Table 11-9. Master-Transmitter Addresses Slave-Receiver with a 10-bit Address

After the master-transmitter has sent the first byte of the 10-bit address, the slave-receiver sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

11.4.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/$\overline{W}$ bit (see Table 11-10). Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated start condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the start condition (S) and tests whether the eighth (R/$\overline{W}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

After a repeated start condition (Sr), all other slave devices also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/$\overline{W}$) bit. However, none of them are addressed because R/$\overline{W}$ = 1 (for 10-bit devices) or the 11110XX slave address (for 7-bit devices) does not match.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Sr	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 1	A3	Data	A	...	Data	A	P

Table 11-10. Master-Receiver Addresses a Slave-Transmitter with a 10-bit Address

After the master-receiver has sent the first byte of the 10-bit address, the slave-transmitter sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

11.4.3 General Call Address

General calls can be requested in 7-bit address or 10-bit address. If the GCAEN bit is set, the IIC matches the general call address as well as its own slave address. When the IIC responds to a general call, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software must read the IICD register after the first byte transfer to determine whether the address matches is its own slave address or a general call. If the value is 00, the match is a general call. If the GCAEN bit is clear, the IIC ignores any data supplied from a general call address by not issuing an acknowledgement.

11.5 Resets

The IIC is disabled after reset. The IIC cannot cause an MCU reset.

11.6 Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in Table 11-11 occur, provided the IICIE bit is set. The interrupt is driven by bit IICIF (of the IIC status register) and masked with bit IICIE (of the IIC control register). The IICIF bit must be cleared by software by writing a 1 to it in the interrupt routine. You can determine the interrupt type by reading the status register.

Table 11-11. Interrupt Summary

Interrupt Source	Status	Flag	Local Enable
Complete 1-byte transfer	TCF	IICIF	IICIE
Match of received calling address	IAAS	IICIF	IICIE
Arbitration Lost	ARBL	IICIF	IICIE

11.6.1 Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the ninth clock to indicate the completion of byte transfer.

11.6.2 Address Detect Interrupt

When the calling address matches the programmed slave address (IIC address register) or when the GCAEN bit is set and a general call is received, the IAAS bit in the status register is set. The CPU is interrupted, provided the IICIE is set. The CPU must check the SRW bit and set its Tx mode accordingly.

11.6.3 Arbitration Lost Interrupt

The IIC 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, the relative priority of the contending masters is determined by a data arbitration procedure. The IIC module asserts this interrupt when it loses the data arbitration process and the ARBL bit in the status register is set.

Arbitration is lost in the following circumstances:

- SDA sampled as a low when the master drives a high during an address or data transmit cycle.
- SDA sampled as a low when the master drives a high during the acknowledge bit of a data receive cycle.
- A start cycle is attempted when the bus is busy.
- A repeated start cycle is requested in slave mode.
- A stop condition is detected when the master did not request it.

This bit must be cleared by software writing a 1 to it.

11.7 Initialization/Application Information

Module Initialization (Slave)

1. Write: IICC2
   — to enable or disable general call
   — to select 10-bit or 7-bit addressing mode
2. Write: IICA
   — to set the slave address
3. Write: IICC1
   — to enable IIC and interrupts
4. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
5. Initialize RAM variables used to achieve the routine shown in Figure 11-12

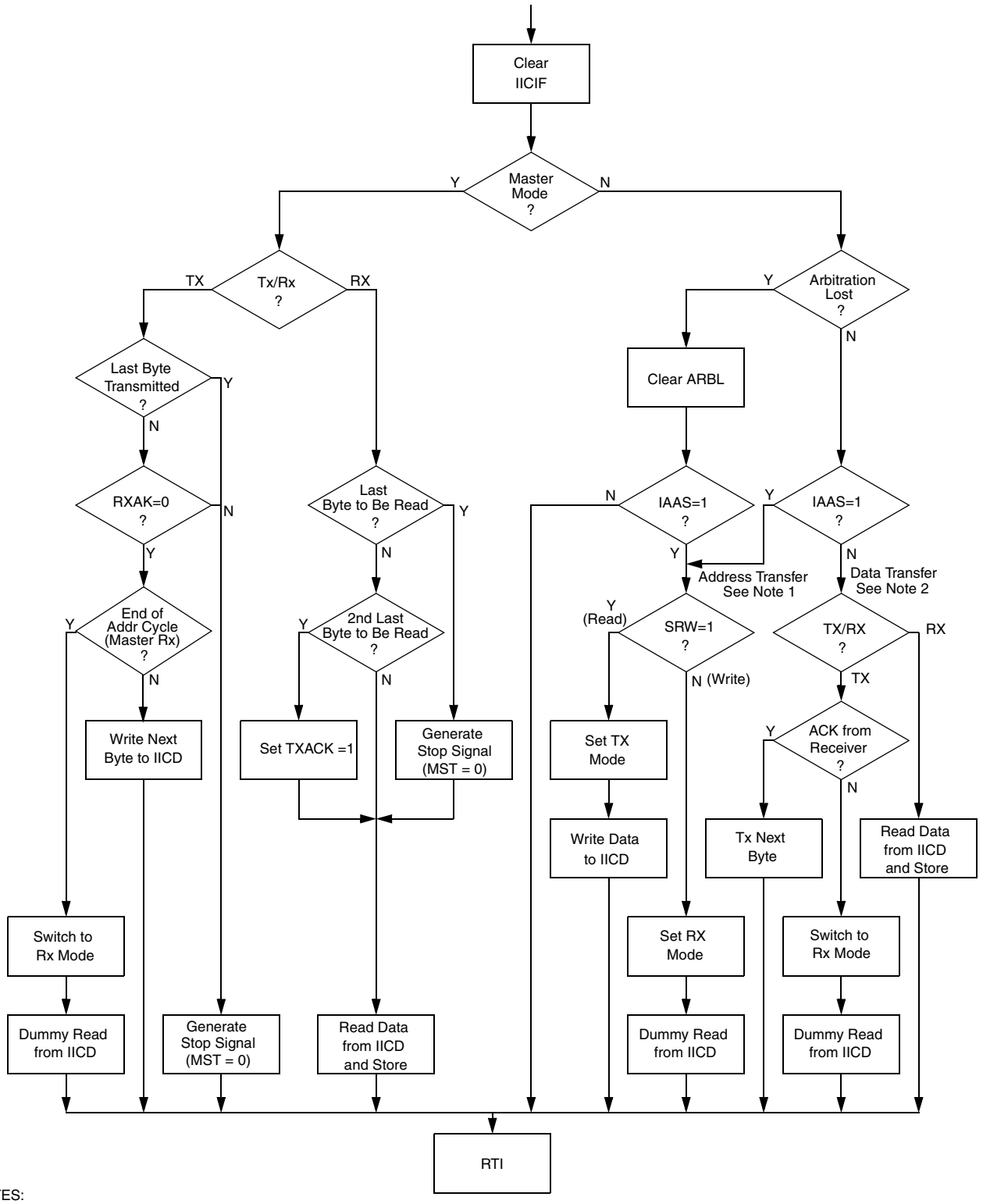
Module Initialization (Master)

1. Write: IICF
   — to set the IIC baud rate (example provided in this chapter)
2. Write: IICC1
   — to enable IIC and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in Figure 11-12
5. Write: IICC1
   — to enable TX

Register Model

IICA	AD[7:1]							0
	When addressed as a slave (in slave mode), the module responds to this address							
IICF	MULT		ICR					
	Baud rate = BUSCLK / (2 x MULT x (SCL DIVIDER))							
IICC1	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
	Module configuration							
IICS	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
	Module status flags							
IICD	DATA							
	Data register; Write to transmit IIC data read to read IIC data							
IICC2	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
	Address configuration							

Figure 11-11. IIC Module Quick Start



NOTES:

1. If general call is enabled, a check must be done to determine whether the received address was a general call address (0x00). If the received address was a general call address, then the general call must be handled by user software.
2. When 10-bit addressing is used to address a slave, the slave sees an interrupt following the first byte of the extended address.

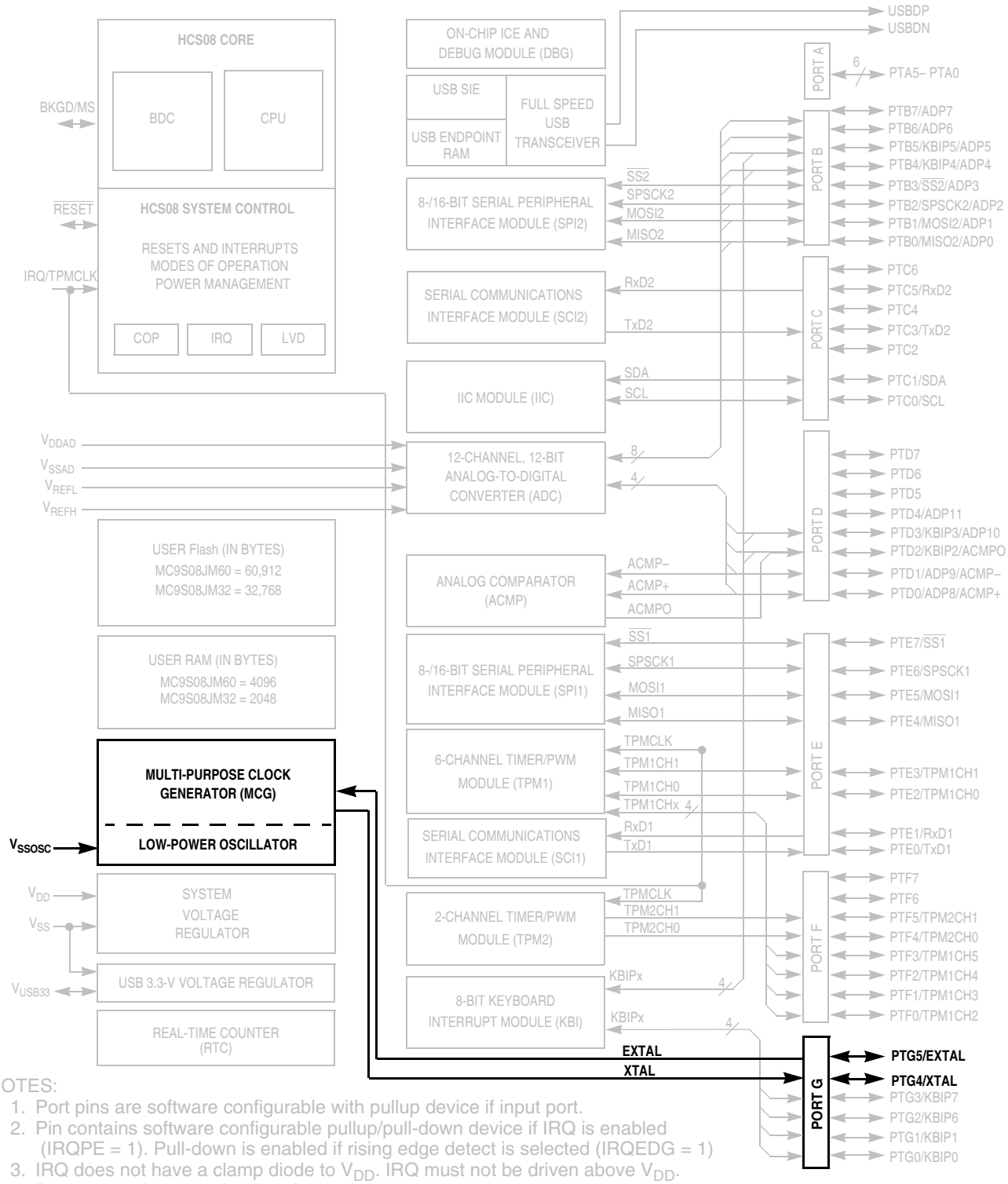
Figure 11-12. Typical IIC Interrupt Routine

Chapter 12
Multi-Purpose Clock Generator (S08MCGV1)

12.1 Introduction

The multi-purpose clock generator (MCG) module provides several clock source choices for the MCU. which contains a frequency-locked loop (FLL) and a phase-locked loop (PLL). The module can select either of the FLL or PLL clocks, or either of the internal or external reference clocks as a source for the MCU system clock. Whichever clock source is chosen, it is passed through a reduced bus divider which allows a lower output clock frequency to be derived. The MCG also controls an external oscillator (XOSC) for the use of a crystal or resonator as the external reference clock.

For USB operation on the MC9S08JM60 series, the MCG must be configured for PLL engaged external (PEE) mode in order to achieve a MCGOUT frequency of 48 MHz



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 12-1. MC9S08JM60 Series Block Diagram Highlighting MCG Block and Pins

12.1.1 Features

Key features of the MCG module are:

- Frequency-locked loop (FLL)
  - 0.2% resolution using internal 32-kHz reference
  - 2% deviation over voltage and temperature using internal 32-kHz reference
  - Internal or external reference can be used to control the FLL
- Phase-locked loop (PLL)
  - Voltage-controlled oscillator (VCO)
  - Modulo VCO frequency divider
  - Phase/Frequency detector
  - Integrated loop filter
  - Lock detector with interrupt capability
- Internal reference clock
  - Nine trim bits for accuracy
  - Can be selected as the clock source for the MCU
- External reference clock
  - Control for external oscillator
  - Clock monitor with reset capability
  - Can be selected as the clock source for the MCU
- Reference divider is provided
- Clock source selected can be divided down by 1, 2, 4, or 8
- BDC clock (MCGLCLK) is provided as a constant divide by 2 of the DCO output whether in an FLL or PLL mode.

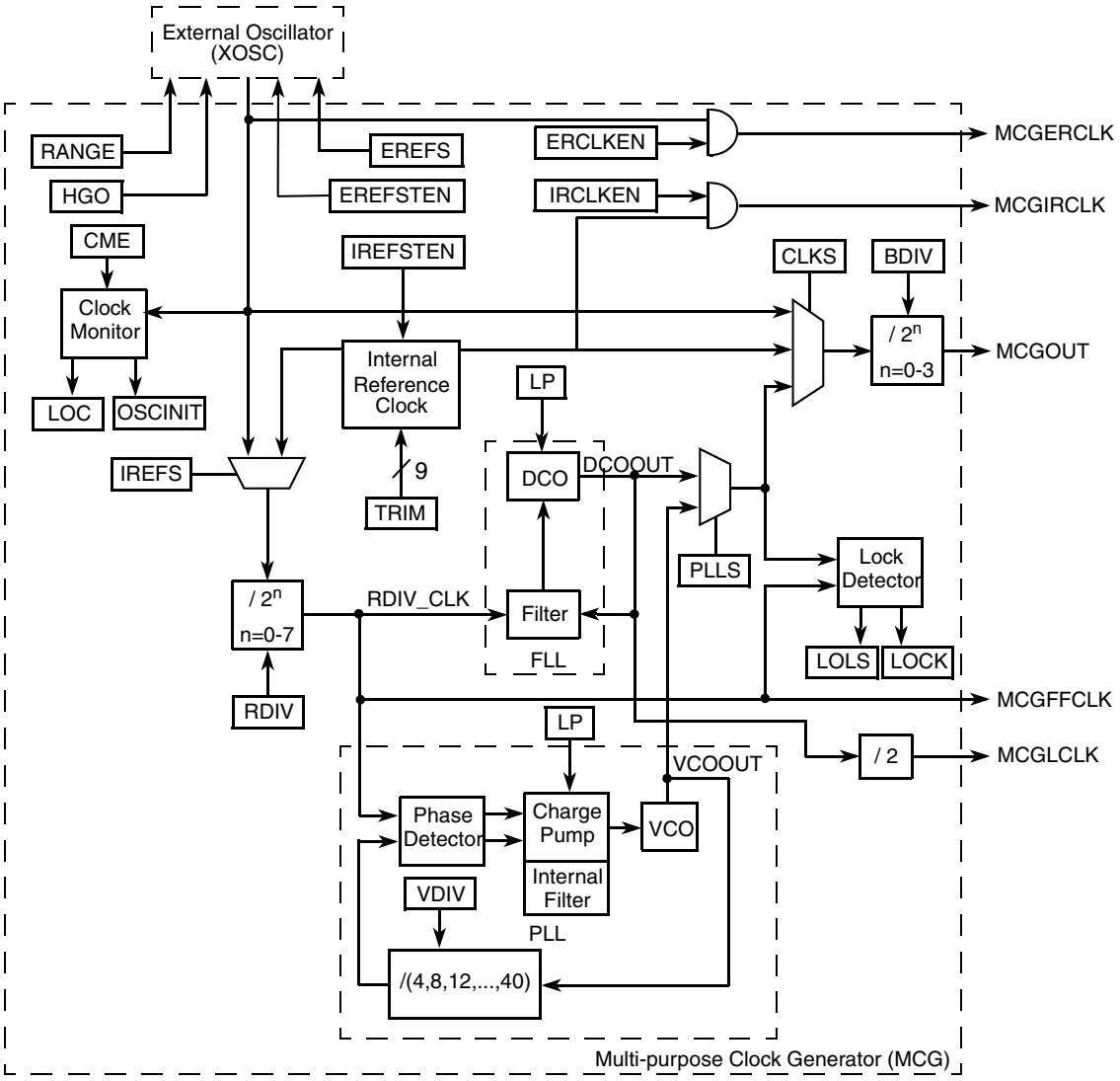


Figure 12-2. Multi-Purpose Clock Generator (MCG) Block Diagram

12.1.2 Modes of Operation

There are nine modes of operation for the MCG:

- FLL Engaged Internal (FEI)
- FLL Engaged External (FEE)
- FLL Bypassed Internal (FBI)
- FLL Bypassed External (FBE)
- PLL Engaged External (PEE)
- PLL Bypassed External (PBE)
- Bypassed Low Power Internal (BLPI)
- Bypassed Low Power External (BLPE)
- Stop

For details see Section 12.4.1, “Operational Modes.”

12.2 External Signal Description

There are no MCG signals that connect off chip.

12.3 Register Definition

12.3.1 MCG Control Register 1 (MCGC1)

	7	6	5	4	3	2	1	0
R W	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
Reset:	0	0	0	0	0	1	0	0

Figure 12-3. MCG Control Register 1 (MCGC1)

Table 12-1. MCG Control Register 1 Field Descriptions

Field	Description
7:6 CLKS	**Clock Source Select** — Selects the system clock source. 00 Encoding 0 — Output of FLL or PLL is selected. 01 Encoding 1 — Internal reference clock is selected. 10 Encoding 2 — External reference clock is selected. 11 Encoding 3 — Reserved, defaults to 00.
5:3 RDIV	**Reference Divider** — Selects the amount to divide down the reference clock selected by the IREFS bit. If the FLL is selected, the resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. If the PLL is selected, the resulting frequency must be in the range 1 MHz to 2 MHz. 000 Encoding 0 — Divides reference clock by 1 (reset default) 001 Encoding 1 — Divides reference clock by 2 010 Encoding 2 — Divides reference clock by 4 011 Encoding 3 — Divides reference clock by 8 100 Encoding 4 — Divides reference clock by 16 101 Encoding 5 — Divides reference clock by 32 110 Encoding 6 — Divides reference clock by 64 111 Encoding 7 — Divides reference clock by 128
2 IREFS	**Internal Reference Select** — Selects the reference clock source. 1 Internal reference clock selected 0 External reference clock selected
1 IRCLKEN	**Internal Reference Clock Enable** — Enables the internal reference clock for use as MCGIRCLK. 1 MCGIRCLK active 0 MCGIRCLK inactive
0 IREFSTEN	**Internal Reference Stop Enable** — Controls whether or not the internal reference clock remains enabled when the MCG enters stop mode. 1 Internal reference clock stays enabled in stop if IRCLKEN is set or if MCG is in FEI, FBI, or BLPI mode before entering stop 0 Internal reference clock is disabled in stop

12.3.2 MCG Control Register 2 (MCGC2)

	7	6	5	4	3	2	1	0
R W	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
Reset:	0	1	0	0	0	0	0	0

Figure 12-4. MCG Control Register 2 (MCGC2)

Table 12-2. MCG Control Register 2 Field Descriptions

Field	Description
7:6 BDIV	**Bus Frequency Divider** — Selects the amount to divide down the clock source selected by the CLKS bits in the MCGC1 register. This controls the bus frequency. 00 Encoding 0 — Divides selected clock by 1 01 Encoding 1 — Divides selected clock by 2 (reset default) 10 Encoding 2 — Divides selected clock by 4 11 Encoding 3 — Divides selected clock by 8
5 RANGE	**Frequency Range Select** — Selects the frequency range for the external oscillator or external clock source. 1 High frequency range selected for the external oscillator of 1 MHz to 16 MHz (1 MHz to 40 MHz for external clock source) 0 Low frequency range selected for the external oscillator of 32 kHz to 100 kHz (32 kHz to 1 MHz for external clock source)
4 HGO	**High Gain Oscillator Select** — Controls the external oscillator mode of operation. 1 Configure external oscillator for high gain operation 0 Configure external oscillator for low power operation
3 LP	**Low Power Select** — Controls whether the FLL (or PLL) is disabled in bypassed modes. 1 FLL (or PLL) is disabled in bypass modes (lower power). 0 FLL (or PLL) is not disabled in bypass modes.
2 EREFS	**External Reference Select** — Selects the source for the external reference clock. 1 Oscillator requested 0 External Clock Source requested
1 ERCLKEN	**External Reference Enable** — Enables the external reference clock for use as MCGERCLK. 1 MCGERCLK active 0 MCGERCLK inactive
0 EREFSTEN	**External Reference Stop Enable** — Controls whether or not the external reference clock remains enabled when the MCG enters stop mode. 1 External reference clock stays enabled in stop if ERCLKEN is set or if MCG is in FEE, FBE, PEE, PBE, or BLPE mode before entering stop 0 External reference clock is disabled in stop

12.3.3 MCG Trim Register (MCGTRM)

	7	6	5	4	3	2	1	0
R W	TRIM							
POR:	1	0	0	0	0	0	0	0
Reset:	U	U	U	U	U	U	U	U

Figure 12-5. MCG Trim Register (MCGTRM)

Table 12-3. MCG Trim Register Field Descriptions

Field	Description
7:0 TRIM	MCG **Trim Setting** — Controls the internal reference clock frequency by controlling the internal reference clock period. The TRIM bits are binary weighted (i.e., bit 1 will adjust twice as much as bit 0). Increasing the binary value in TRIM will increase the period, and decreasing the value will decrease the period. An additional fine trim bit is available in MCGSC as the FTRIM bit. If a TRIM[7:0] value stored in nonvolatile memory is to be used, it's the user's responsibility to copy that value from the nonvolatile memory location to this register.

12.3.4 MCG Status and Control Register (MCGSC)

	7	6	5	4	3	2	1	0
R	LOLS	LOCK	PLLST	IREFST	CLKST		OSCINIT	FTRIM
W								
POR:	0	0	0	1	0	0	0	0
Reset:	0	0	0	1	0	0	0	U

Figure 12-6. MCG Status and Control Register (MCGSC)

Table 12-4. MCG Status and Control Register Field Descriptions

Field	Description
7 LOLS	**Loss of Lock Status** — This bit is a sticky indication of lock status for the FLL or PLL. LOLS is set when lock detection is enabled and after acquiring lock, the FLL or PLL output frequency has fallen outside the lock exit frequency tolerance, D_{unl}. LOLIE determines whether an interrupt request is made when set. LOLS is cleared by reset or by writing a logic 1 to LOLS when LOLS is set. Writing a logic 0 to LOLS has no effect. 0 FLL or PLL has not lost lock since LOLS was last cleared. 1 FLL or PLL has lost lock since LOLS was last cleared.
6 LOCK	**Lock Status** — Indicates whether the FLL or PLL has acquired lock. Lock detection is disabled when both the FLL and PLL are disabled. If the lock status bit is set then changing the value of any of the following bits IREFS, PLLS, RDIV[2:0], TRIM[7:0] (if in FEI or FBI modes), or VDIV[3:0] (if in PBE or PEE modes), will cause the lock status bit to clear and stay cleared until the FLL or PLL has reacquired lock. Stop mode entry will also cause the lock status bit to clear and stay cleared until the FLL or PLL has reacquired lock. Entry into BLPI or BLPE mode will also cause the lock status bit to clear and stay cleared until the MCG has exited these modes and the FLL or PLL has reacquired lock. 0 FLL or PLL is currently unlocked. 1 FLL or PLL is currently locked.
5 PLLST	**PLL Select Status** — The PLLST bit indicates the current source for the PLLS clock. The PLLST bit does not update immediately after a write to the PLLS bit due to internal synchronization between clock domains. 0 Source of PLLS clock is FLL clock. 1 Source of PLLS clock is PLL clock.
4 IREFST	**Internal Reference Status** — The IREFST bit indicates the current source for the reference clock. The IREFST bit does not update immediately after a write to the IREFS bit due to internal synchronization between clock domains. 0 Source of reference clock is external reference clock (oscillator or external clock source as determined by the EREFS bit in the MCGC2 register). 1 Source of reference clock is internal reference clock.
3:2 CLKST	**Clock Mode Status** — The CLKST bits indicate the current clock mode. The CLKST bits do not update immediately after a write to the CLKS bits due to internal synchronization between clock domains. 00 Encoding 0 — Output of FLL is selected. 01 Encoding 1 — Internal reference clock is selected. 10 Encoding 2 — External reference clock is selected. 11 Encoding 3 — Output of PLL is selected.

Table 12-4. MCG Status and Control Register Field Descriptions (continued)

Field	Description
1 OSCINIT	**OSC Initialization** — If the external reference clock is selected by ERCLKEN or by the MCG being in FEE, FBE, PEE, PBE, or BLPE mode, and if EREFS is set, then this bit is set after the initialization cycles of the external oscillator clock have completed. This bit is only cleared when either EREFS is cleared or when the MCG is in either FEI, FBI, or BLPI mode and ERCLKEN is cleared.
0 FTRIM	MCG **Fine Trim** — Controls the smallest adjustment of the internal reference clock frequency. Setting FTRIM will increase the period and clearing FTRIM will decrease the period by the smallest amount possible. If an FTRIM value stored in nonvolatile memory is to be used, it's the user's responsibility to copy that value from the nonvolatile memory location to this register's FTRIM bit.

12.3.5 MCG Control Register 3 (MCGC3)

	7	6	5	4	3	2	1	0
R	LOLIE	PLLS	CME	0	VDIV			
W								
Reset:	0	0	0	0	0	0	0	1

Figure 12-7. MCG PLL Register (MCGPLL)

Table 12-5. MCG PLL Register Field Descriptions

Field	Description
7 LOLIE	**Loss of Lock Interrupt Enable** — Determines if an interrupt request is made following a loss of lock indication. The LOLIE bit only has an effect when LOLS is set. 0 No request on loss of lock. 1 Generate an interrupt request on loss of lock.
6 PLLS	**PLL Select** — Controls whether the PLL or FLL is selected. If the PLLS bit is clear, the PLL is disabled in all modes. If the PLLS is set, the FLL is disabled in all modes. 1 PLL is selected 0 FLL is selected

Table 12-5. MCG PLL Register Field Descriptions (continued)

Field	Description
5 CME	**Clock Monitor Enable** — Determines if a reset request is made following a loss of external clock indication. The CME bit should only be set to a logic 1 when either the MCG is in an operational mode that uses the external clock (FEE, FBE, PEE, PBE, or BLPE) or the external reference is enabled (ERCLKEN=1 in the MCGC2 register). Whenever the CME bit is set to a logic 1, the value of the RANGE bit in the MCGC2 register should not be changed. 0 Clock monitor is disabled. 1 Generate a reset request on loss of external clock.
3:0 VDIV	**VCO Divider** — Selects the amount to divide down the VCO output of PLL. The VDIV bits establish the multiplication factor (M) applied to the reference clock frequency. 0000 Encoding 0 — Reserved. 0001 Encoding 1 — Multiply by 4. 0010 Encoding 2 — Multiply by 8. 0011 Encoding 3 — Multiply by 12. 0100 Encoding 4 — Multiply by 16. 0101 Encoding 5 — Multiply by 20. 0110 Encoding 6 — Multiply by 24. 0111 Encoding 7 — Multiply by 28. 1000 Encoding 8 — Multiply by 32. 1001 Encoding 9 — Multiply by 36. 1010 Encoding 10 — Multiply by 40. 1011 Encoding 11 — Reserved (default to M=40). 11xx Encoding 12-15 — Reserved (default to M=40).

12.4 Functional Description

12.4.1 Operational Modes

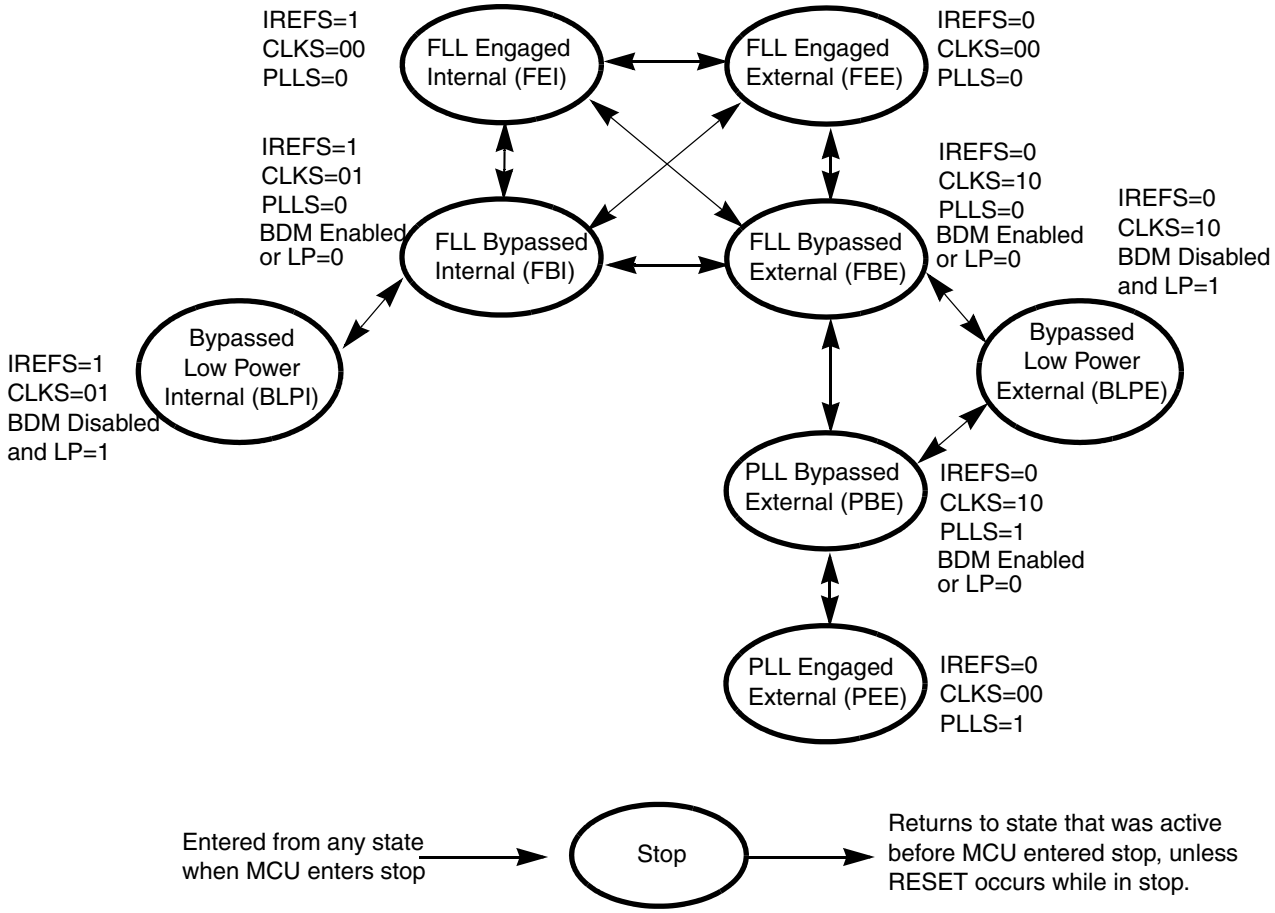


Figure 12-8. Clock Switching Modes

The nine states of the MCG are shown as a state diagram and are described below. The arrows indicate the allowed movements between the states.

12.4.1.1 FLL Engaged Internal (FEI)

FLL engaged internal (FEI) is the default mode of operation and is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 1
- PLLS bit is written to 0
- RDIV bits are written to 000. Since the internal reference clock frequency should already be in the range of 31.25 kHz to 39.0625 kHz after it is trimmed, no further frequency divide is necessary.

In FLL engaged internal mode, the MCGOUT clock is derived from the FLL clock, which is controlled by the internal reference clock. The FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

12.4.1.2 FLL Engaged External (FEE)

The FLL engaged external (FEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 0
- PLLS bit is written to 0
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz

In FLL engaged external mode, the MCGOUT clock is derived from the FLL clock which is controlled by the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source.The FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

12.4.1.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal (FBI) mode, the MCGOUT clock is derived from the internal reference clock and the FLL is operational but its output clock is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUT clock is driven from the internal reference clock.

The FLL bypassed internal mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1
- PLLS bit is written to 0
- RDIV bits are written to 000. Since the internal reference clock frequency should already be in the range of 31.25 kHz to 39.0625 kHz after it is trimmed, no further frequency divide is necessary.
- LP bit is written to 0

In FLL bypassed internal mode, the MCGOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

12.4.1.4 FLL Bypassed External (FBE)

In FLL bypassed external (FBE) mode, the MCGOUT clock is derived from the external reference clock and the FLL is operational but its output clock is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUT clock is driven from the external reference clock.

The FLL bypassed external mode is entered when all the following conditions occur:

- CLKS bits are written to 10
- IREFS bit is written to 0
- PLLS bit is written to 0
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz
- LP bit is written to 0

In FLL bypassed external mode, the MCGOUT clock is derived from the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source.The FLL clock is controlled by the external reference clock, and the FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

NOTE

It is possible to briefly operate in FBE mode with an FLL reference clock frequency that is greater than the specified maximum frequency. This can be necessary in applications that operate in PEE mode using an external crystal with a frequency above 5 MHz. Please see 12.5.2.4, “Example # 4: Moving from FEI to PEE Mode: External Crystal = 8 MHz, Bus Frequency = 8 MHz for a detailed example.

12.4.1.5 PLL Engaged External (PEE)

The PLL engaged external (PEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 0
- PLLS bit is written to 1
- RDIV bits are written to divide reference clock to be within the range of 1 MHz to 2 MHz

In PLL engaged external mode, the MCGOUT clock is derived from the PLL clock which is controlled by the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source The PLL clock frequency locks to a multiplication factor, as selected by the VDIV bits, times the reference frequency, as selected by the RDIV bits. If BDM is enabled then the MCGLCLK is derived from the DCO (open-loop mode) divided by two. If BDM is not enabled then the FLL is disabled in a low power state.

12.4.1.6 PLL Bypassed External (PBE)

In PLL bypassed external (PBE) mode, the MCGOUT clock is derived from the external reference clock and the PLL is operational but its output clock is not used. This mode is useful to allow the PLL to acquire its target frequency while the MCGOUT clock is driven from the external reference clock.

The PLL bypassed external mode is entered when all the following conditions occur:

- CLKS bits are written to 10
- IREFS bit is written to 0
- PLLS bit is written to 1
- RDIV bits are written to divide reference clock to be within the range of 1 MHz to 2 MHz
- LP bit is written to 0

In PLL bypassed external mode, the MCGOUT clock is derived from the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source. The PLL clock frequency locks to a multiplication factor, as selected by the VDIV bits, times the reference frequency, as selected by the RDIV bits. If BDM is enabled then the MCGLCLK is derived from the DCO (open-loop mode) divided by two. If BDM is not enabled then the FLL is disabled in a low power state.

12.4.1.7 Bypassed Low Power Internal (BLPI)

The bypassed low power internal (BLPI) mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1
- PLLS bit is written to 0 or 1
- LP bit is written to 1
- BDM mode is not active

In bypassed low power internal mode, the MCGOUT clock is derived from the internal reference clock.

The PLL and the FLL are disabled at all times in BLPI mode and the MCGLCLK will not be available for BDC communications If the BDM becomes active the mode will switch to one of the bypassed internal modes as determined by the state of the PLLS bit.

12.4.1.8 Bypassed Low Power External (BLPE)

The bypassed low power external (BLPE) mode is entered when all the following conditions occur:

- CLKS bits are written to 10
- IREFS bit is written to 0
- PLLS bit is written to 0 or 1
- LP bit is written to 1
- BDM mode is not active

In bypassed low power external mode, the MCGOUT clock is derived from the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source.

The PLL and the FLL are disabled at all times in BLPE mode and the MCGLCLK will not be available for BDC communications. If the BDM becomes active the mode will switch to one of the bypassed external modes as determined by the state of the PLLS bit.

12.4.1.9 Stop

Stop mode is entered whenever the MCU enters a STOP state. In this mode, the FLL and PLL are disabled and all MCG clock signals are static except in the following cases:

MCGIRCLK will be active in stop mode when all the following conditions occur:

- IRCLKEN = 1
- IREFSTEN = 1

MCGERCLK will be active in stop mode when all the following conditions occur:

- ERCLKEN = 1
- EREFSTEN = 1

12.4.2 Mode Switching

When switching between engaged internal and engaged external modes the IREFS bit can be changed at anytime, but the RDIV bits must be changed simultaneously so that the reference frequency stays in the range required by the state of the PLLS bit (31.25 kHz to 39.0625 kHz if the FLL is selected, or 1 MHz to 2 MHz if the PLL is selected). After a change in the IREFS value the FLL or PLL will begin locking again after the switch is completed. The completion of the switch is shown by the IREFST bit.

For the special case of entering stop mode immediately after switching to FBE mode, if the external clock and the internal clock are disabled in stop mode, (EREFSTEN = 0 and IREFSTEN = 0), it is necessary to allow 100us after the IREFST bit is cleared to allow the internal reference to shutdown. For most cases the delay due to instruction execution times will be sufficient.

The CLKS bits can also be changed at anytime, but in order for the MCGLCLK to be configured correctly the RDIV bits must be changed simultaneously so that the reference frequency stays in the range required by the state of the PLLS bit (31.25 kHz to 39.0625 kHz if the FLL is selected, or 1 MHz to 2MHz if the PLL is selected). The actual switch to the newly selected clock will be shown by the CLKST bits. If the newly selected clock is not available, the previous clock will remain selected.

For details see Figure 12-8.

12.4.3 Bus Frequency Divider

The BDIV bits can be changed at anytime and the actual switch to the new frequency will occur immediately.

12.4.4 Low Power Bit Usage

The low power bit (LP) is provided to allow the FLL or PLL to be disabled and thus conserve power when these systems are not being used. However, in some applications it may be desirable to enable the FLL or PLL and allow it to lock for maximum accuracy before switching to an engaged mode. Do this by writing the LP bit to 0.

12.4.5 Internal Reference Clock

When IRCLKEN is set the internal reference clock signal will be presented as MCGIRCLK, which can be used as an additional clock source. The MCGIRCLK frequency can be re-targeted by trimming the period of the internal reference clock. This can be done by writing a new value to the TRIM bits in the MCGTRM register. Writing a larger value will decrease the MCGIRCLK frequency, and writing a smaller value to the MCGTRM register will increase the MCGIRCLK frequency. The TRIM bits will effect the MCGOUT frequency if the MCG is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or bypassed low power internal (BLPI) mode. The TRIM and FTRIM value is initialized by POR but is not affected by other resets.

Until MCGIRCLK is trimmed, programming low reference divider (RDIV) factors may result in MCGOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications (see the Device Overview chapter).

If IREFSTEN and IRCLKEN bits are both set, the internal reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

12.4.6 External Reference Clock

The MCG module can support an external reference clock with frequencies between 31.25 kHz to 5 MHz in FEE and FBE modes, 1 MHz to 16 MHz in PEE and PBE modes, and 0 to 40 MHz in BLPE mode. When ERCLKEN is set, the external reference clock signal will be presented as MCGERCLK, which can be used as an additional clock source. When IREFS = 1, the external reference clock will not be used by the FLL or PLL and will only be used as MCGERCLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications will support (see the Device Overview chapter).

If EREFSTEN and ERCLKEN bits are both set or the MCG is in FEE, FBE, PEE, PBE or BLPE mode, the external reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

If CME bit is written to 1, the clock monitor is enabled. If the external reference falls below a certain frequency (f_{loc_high} or f_{loc_low} depending on the RANGE bit in the MCGC2), the MCU will reset. The LOC bit in the System Reset Status (SRS) register will be set to indicate the error.

12.4.7 Fixed Frequency Clock

The MCG presents the divided reference clock as MCGFFCLK for use as an additional clock source. The MCGFFCLK frequency must be no more than 1/4 of the MCGOUT frequency to be valid. Because of this requirement, the MCGFFCLK is not valid in bypass modes for the following combinations of BDIV and RDIV values:

- BDIV=00 (divide by 1), RDIV < 010

BDIV=01 (divide by 2), RDIV < 011

12.5 Initialization / Application Information

This section describes how to initialize and configure the MCG module in application. The following sections include examples on how to initialize the MCG and properly switch between the various available modes.

12.5.1 MCG Module Initialization Sequence

The MCG comes out of reset configured for FEI mode with the BDIV set for divide-by-2. The internal reference will stabilize in t_{irefst} microseconds before the FLL can acquire lock. As soon as the internal reference is stable, the FLL will acquire lock in t_{fll_lock} milliseconds.

Upon POR, the internal reference will require trimming to guarantee an accurate clock. Freescale recommends using FLASH location 0xFFAE for storing the fine trim bit, FTRIM in the MCGSC register, and 0xFFAF for storing the 8-bit trim value in the MCGTRM register. The MCU will not automatically copy the values in these FLASH locations to the respective registers. Therefore, user code must copy these values from FLASH to the registers.

NOTE

The BDIV value should not be changed to divide-by-1 without first trimming the internal reference. Failure to do so could result in the MCU running out of specification.

12.5.1.1 Initializing the MCG

Because the MCG comes out of reset in FEI mode, the only MCG modes which can be directly switched to upon reset are FEE, FBE, and FBI modes (see Figure 12-8). Reaching any of the other modes requires first configuring the MCG for one of these three initial modes. Care must be taken to check relevant status bits in the MCGSC register reflecting all configuration changes within each mode.

To change from FEI mode to FEE or FBE modes, follow this procedure:

1. Enable the external clock source by setting the appropriate bits in MCGC2.
2. Write to MCGC1 to select the clock mode.
   - — If entering FEE, set RDIV appropriately, clear the IREFS bit to switch to the external reference, and leave the CLKS bits at %00 so that the output of the FLL is selected as the system clock source.
   - — If entering FBE, clear the IREFS bit to switch to the external reference and change the CLKS bits to %10 so that the external reference clock is selected as the system clock source. The RDIV bits should also be set appropriately here according to the external reference frequency because although the FLL is bypassed, it is still on in FBE mode.
   - — The internal reference can optionally be kept running by setting the IRCLKEN bit. This is useful if the application will switch back and forth between internal and external modes. For

minimum power consumption, leave the internal reference disabled while in an external clock mode.

3. After the proper configuration bits have been set, wait for the affected bits in the MCGSC register to be changed appropriately, reflecting that the MCG has moved into the proper mode.
   - If ERCLKEN was set in step 1 or the MCG is in FEE, FBE, PEE, PBE, or BLPE mode, and EREFS was also set in step 1, wait here for the OSCINIT bit to become set indicating that the external clock source has finished its initialization cycles and stabilized. Typical crystal startup times are given in Appendix A, “Electrical Characteristics”.
   - If in FEE mode, check to make sure the IREFST bit is cleared and the LOCK bit is set before moving on.
   - If in FBE mode, check to make sure the IREFST bit is cleared, the LOCK bit is set, and the CLKST bits have changed to %10 indicating the external reference clock has been appropriately selected. Although the FLL is bypassed in FBE mode, it is still on and will lock in FBE mode.

To change from FEI clock mode to FBI clock mode, follow this procedure:

1. Change the CLKS bits to %01 so that the internal reference clock is selected as the system clock source.
2. Wait for the CLKST bits in the MCGSC register to change to %01, indicating that the internal reference clock has been appropriately selected.

12.5.2 MCG Mode Switching

When switching between operational modes of the MCG, certain configuration bits must be changed in order to properly move from one mode to another. Each time any of these bits are changed (PLLS, IREFS, CLKS, or EREFS), the corresponding bits in the MCGSC register (PLLST, IREFST, CLKST, or OSCINIT) must be checked before moving on in the application software.

Additionally, care must be taken to ensure that the reference clock divider (RDIV) is set properly for the mode being switched to. For instance, in PEE mode, if using a 4 MHz crystal, RDIV must be set to %001 (divide-by-2) or %010 (divide -by-4) in order to divide the external reference down to the required frequency between 1 and 2 MHz.

The RDIV and IREFS bits should always be set properly before changing the PLLS bit so that the FLL or PLL clock has an appropriate reference clock frequency to switch to.

The table below shows MCGOUT frequency calculations using RDIV, BDIV, and VDIV settings for each clock mode. The bus frequency is equal to MCGOUT divided by 2.

Table 12-6. MCGOUT Frequency Calculation Options

Clock Mode	f_{MCGOUT}[1]	Note
FEI (FLL engaged internal)	(f_{int} * 1024) / B	Typical f_{MCGOUT} = 16 MHz immediately after reset. RDIV bits set to %000.
FEE (FLL engaged external)	(f_{ext} / R *1024) / B	f_{ext} / R must be in the range of 31.25 kHz to 39.0625 kHz
FBE (FLL bypassed external)	f_{ext} / B	f_{ext} / R must be in the range of 31.25 kHz to 39.0625 kHz
FBI (FLL bypassed internal)	f_{int} / B	Typical f_{int} = 32 kHz
PEE (PLL engaged external)	[(f_{ext} / R) * M] / B	f_{ext} / R must be in the range of 1 MHz to 2 MHz
PBE (PLL bypassed external)	f_{ext} / B	f_{ext} / R must be in the range of 1 MHz to 2 MHz
BLPI (Bypassed low power internal)	f_{int} / B	
BLPE (Bypassed low power external)	f_{ext} / B	

[1] R is the reference divider selected by the RDIV bits, B is the bus frequency divider selected by the BDIV bits, and M is the multiplier selected by the VDIV bits.

This section will include 3 mode switching examples using a 4 MHz external crystal. If using an external clock source less than 1 MHz, the MCG should not be configured for any of the PLL modes (PEE and PBE).

12.5.2.1 Example # 1: Moving from FEI to PEE Mode: External Crystal = 4 MHz, Bus Frequency = 8 MHz

In this example, the MCG will move through the proper operational modes from FEI to PEE mode until the 4 MHz crystal reference frequency is set to achieve a bus frequency of 8 MHz. Because the MCG is in FEI mode out of reset, this example also shows how to initialize the MCG for PEE mode out of reset. First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, FEI must transition to FBE mode:
   a) MCGC2 = 0x36 (%00110110)
      – BDIV (bits 7 and 6) set to %00, or divide-by-1
      – RANGE (bit 5) set to 1 because the frequency of 4 MHz is within the high frequency range
      – HGO (bit 4) set to 1 to configure external oscillator for high gain operation
      – EREFS (bit 2) set to 1, because a crystal is being used
      – ERCLKEN (bit 1) set to 1 to ensure the external reference clock is active
   b) Loop until OSCINIT (bit 1) in MCGSC is 1, indicating the crystal selected by the EREFS bit has been initialized.

c) MCGC1 = 0xB8 (%10111000)
   - CLKS (bits 7 and 6) set to %10 in order to select external reference clock as system clock source
   - RDIV (bits 5-3) set to %111, or divide-by-128 because 4 MHz / 128 = 31.25 kHz which is in the 31.25 kHz to 39.0625 kHz range required by the FLL
   - IREFS (bit 2) cleared to 0, selecting the external reference clock

d) Loop until IREFST (bit 4) in MCGSC is 0, indicating the external reference is the current source for the reference clock

e) Loop until CLKST (bits 3 and 2) in MCGSC are %10, indicating that the external reference clock is selected to feed MCGOUT

2. Then, FBE must transition either directly to PBE mode or first through BLPE mode and then to PBE mode:

a) BLPE: If a transition through BLPE mode is desired, first set LP (bit 3) in MCGC2 to 1.

b) BLPE/PBE: MCGC1 = 0x90 (%10010000)
   - RDIV (bits 5-3) set to %010, or divide-by-4 because 4 MHz / 4 = 1 MHz which is in the 1 MHz to 2 MHz range required by the PLL. In BLPE mode, the configuration of the RDIV does not matter because both the FLL and PLL are disabled. Changing them only sets up the the dividers for PLL usage in PBE mode

c) BLPE/PBE: MCGC3 = 0x44 (%01000100)
   - PLLS (bit 6) set to 1, selects the PLL. In BLPE mode, changing this bit only prepares the MCG for PLL usage in PBE mode
   - VDIV (bits 3-0) set to %0100, or multiply-by-16 because 1 MHz reference * 16 = 16 MHz. In BLPE mode, the configuration of the VDIV bits does not matter because the PLL is disabled. Changing them only sets up the multiply value for PLL usage in PBE mode

d) BLPE: If transitioning through BLPE mode, clear LP (bit 3) in MCGC2 to 0 here to switch to PBE mode

e) PBE: Loop until PLLST (bit 5) in MCGSC is set, indicating that the current source for the PLLS clock is the PLL

f) PBE: Then loop until LOCK (bit 6) in MCGSC is set, indicating that the PLL has acquired lock

3. Last, PBE mode transitions into PEE mode:

a) MCGC1 = 0x10 (%00010000)
   - CLKS (bits7 and 6) in MCGSC1 set to %00 in order to select the output of the PLL as the system clock source
   - Loop until CLKST (bits 3 and 2) in MCGSC are %11, indicating that the PLL output is selected to feed MCGOUT in the current clock mode

b) Now, With an RDIV of divide-by-4, a BDIV of divide-by-1, and a VDIV of multiply-by-16, MCGOUT = [(4 MHz / 4) * 16] / 1 = 16 MHz, and the bus frequency is MCGOUT / 2, or 8 MHz

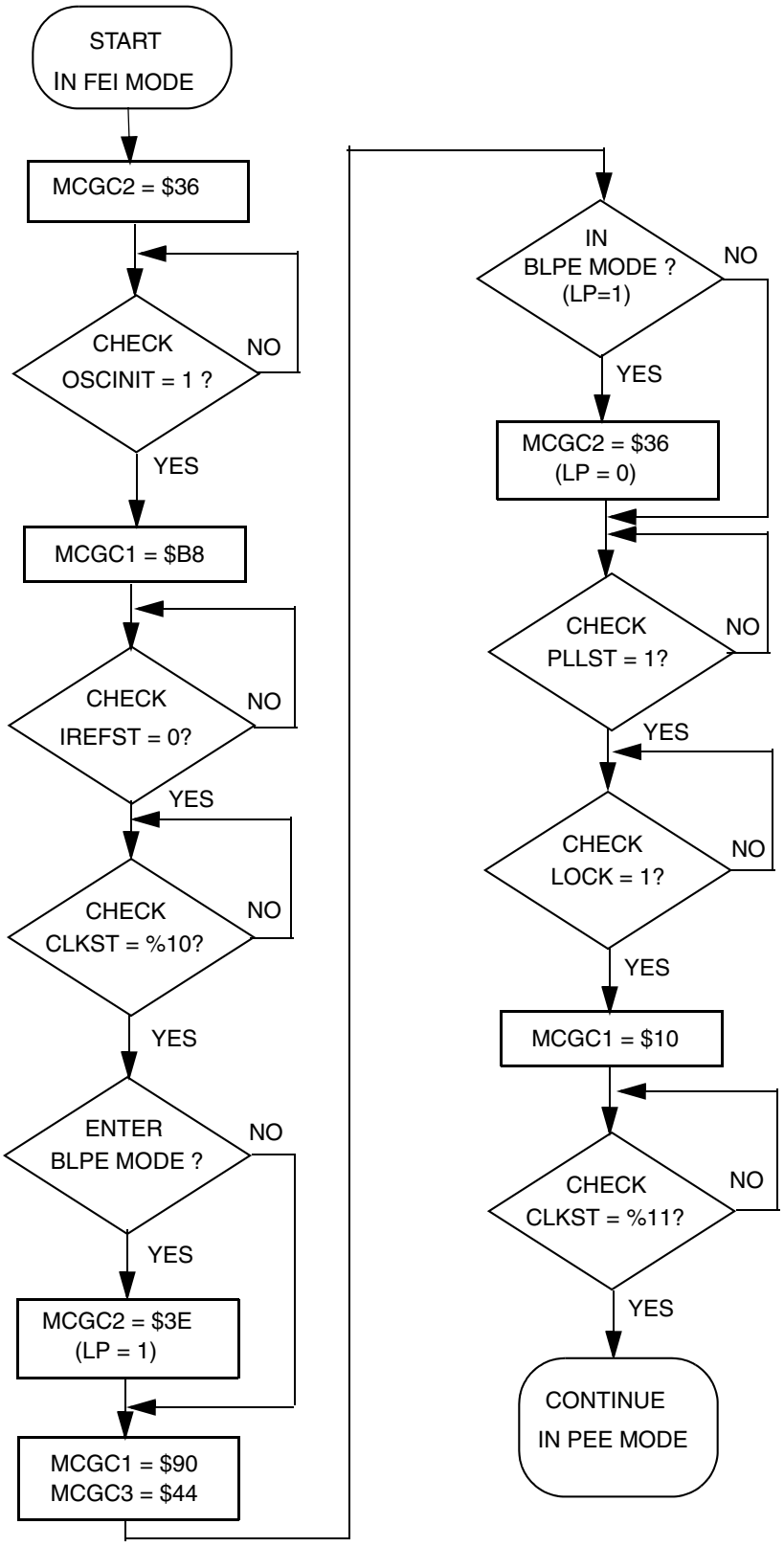


Figure 12-9. Flowchart of FEI to PEE Mode Transition using a 4 MHz Crystal

12.5.2.2 Example # 2: Moving from PEE to BLPI Mode: External Crystal = 4 MHz, Bus Frequency =16 kHz

In this example, the MCG will move through the proper operational modes from PEE mode with a 4 MHz crystal configured for an 8 MHz bus frequency (see previous example) to BLPI mode with a 16 kHz bus frequency.First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, PEE must transition to PBE mode:
   a) MCGC1 = 0x90 (%10010000)
      – CLKS (bits 7 and 6) set to %10 in order to switch the system clock source to the external reference clock
   b) Loop until CLKST (bits 3 and 2) in MCGSC are %10, indicating that the external reference clock is selected to feed MCGOUT
2. Then, PBE must transition either directly to FBE mode or first through BLPE mode and then to FBE mode:
   a) BLPE: If a transition through BLPE mode is desired, first set LP (bit 3) in MCGC2 to 1
   b) BLPE/FBE: MCGC1 = 0xB8 (%10111000)
      – RDIV (bits 5-3) set to %111, or divide-by-128 because 4 MHz / 128 = 31.25 kHz which is in the 31.25 kHz to 39.0625 kHz range required by the FLL. In BLPE mode, the configuration of the RDIV does not matter because both the FLL and PLL are disabled. Changing them only sets up the dividers for FLL usage in FBE mode
   c) BLPE/FBE: MCGC3 = 0x04 (%00000100)
      – PLLS (bit 6) clear to 0 to select the FLL. In BLPE mode, changing this bit only prepares the MCG for FLL usage in FBE mode. With PLLS = 0, the VDIV value does not matter.
   d) BLPE: If transitioning through BLPE mode, clear LP (bit 3) in MCGC2 to 0 here to switch to FBE mode
   e) FBE: Loop until PLLST (bit 5) in MCGSC is clear, indicating that the current source for the PLLS clock is the FLL
   f) FBE: Optionally, loop until LOCK (bit 6) in the MCGSC is set, indicating that the FLL has acquired lock. Although the FLL is bypassed in FBE mode, it is still enabled and running.
3. Next, FBE mode transitions into FBI mode:
   a) MCGC1 = 0x44 (%01000100)
      – CLKS (bits7 and 6) in MCGSC1 set to %01 in order to switch the system clock to the internal reference clock
      – IREFS (bit 2) set to 1 to select the internal reference clock as the reference clock source
      – RDIV (bits 5-3) set to %000, or divide-by-1 because the trimmed internal reference should be within the 31.25 kHz to 39.0625 kHz range required by the FLL
   b) Loop until IREFST (bit 4) in MCGSC is 1, indicating the internal reference clock has been selected as the reference clock source
   c) Loop until CLKST (bits 3 and 2) in MCGSC are %01, indicating that the internal reference clock is selected to feed MCGOUT

4. Lastly, FBI transitions into FBILP mode.
   a) MCGC2 = 0x08 (%00001000)
      – LP (bit 3) in MCGSC is 1

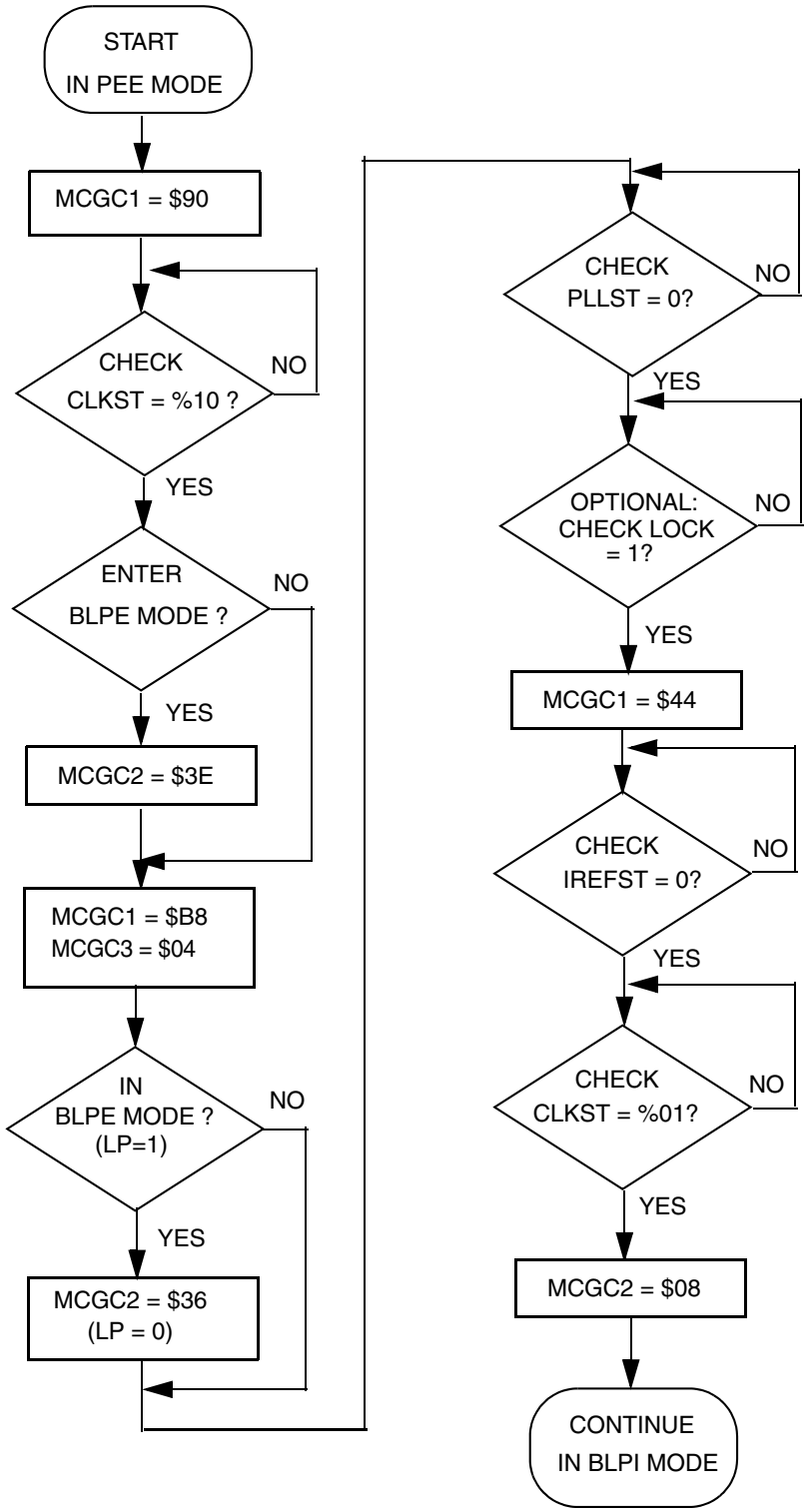


Figure 12-10. Flowchart of PEE to BLPI Mode Transition using a 4 MHz Crystal

12.5.2.3 Example #3: Moving from BLPI to FEE Mode: External Crystal = 4 MHz, Bus Frequency = 16 MHz

In this example, the MCG will move through the proper operational modes from BLPI mode at a 16 kHz bus frequency running off of the internal reference clock (see previous example) to FEE mode using a 4 MHz crystal configured for a 16 MHz bus frequency. First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, BLPI must transition to FBI mode.
   a) MCGC2 = 0x00 (%00000000)
      – LP (bit 3) in MCGSC is 0
   b) Optionally, loop until LOCK (bit 6) in the MCGSC is set, indicating that the FLL has acquired lock. Although the FLL is bypassed in FBI mode, it is still enabled and running.
2. Next, FBI will transition to FEE mode.
   a) MCGC2 = 0x36 (%00110110)
      – RANGE (bit 5) set to 1 because the frequency of 4 MHz is within the high frequency range
      – HGO (bit 4) set to 1 to configure external oscillator for high gain operation
      – EREFS (bit 2) set to 1, because a crystal is being used
      – ERCLKEN (bit 1) set to 1 to ensure the external reference clock is active
   b) Loop until OSCINIT (bit 1) in MCGSC is 1, indicating the crystal selected by the EREFS bit has been initialized.
   c) MCGC1 = 0x38 (%00111000)
      – CLKS (bits 7 and 6) set to %00 in order to select the output of the FLL as system clock source
      – RDIV (bits 5-3) set to %111, or divide-by-128 because 4 MHz / 128 = 31.25 kHz which is in the 31.25 kHz to 39.0625 kHz range required by the FLL
      – IREFS (bit 1) cleared to 0, selecting the external reference clock
   d) Loop until IREFST (bit 4) in MCGSC is 0, indicating the external reference clock is the current source for the reference clock
   e) Optionally, loop until LOCK (bit 6) in the MCGSC is set, indicating that the FLL has reacquired lock.
   f) Loop until CLKST (bits 3 and 2) in MCGSC are %00, indicating that the output of the FLL is selected to feed MCGOUT

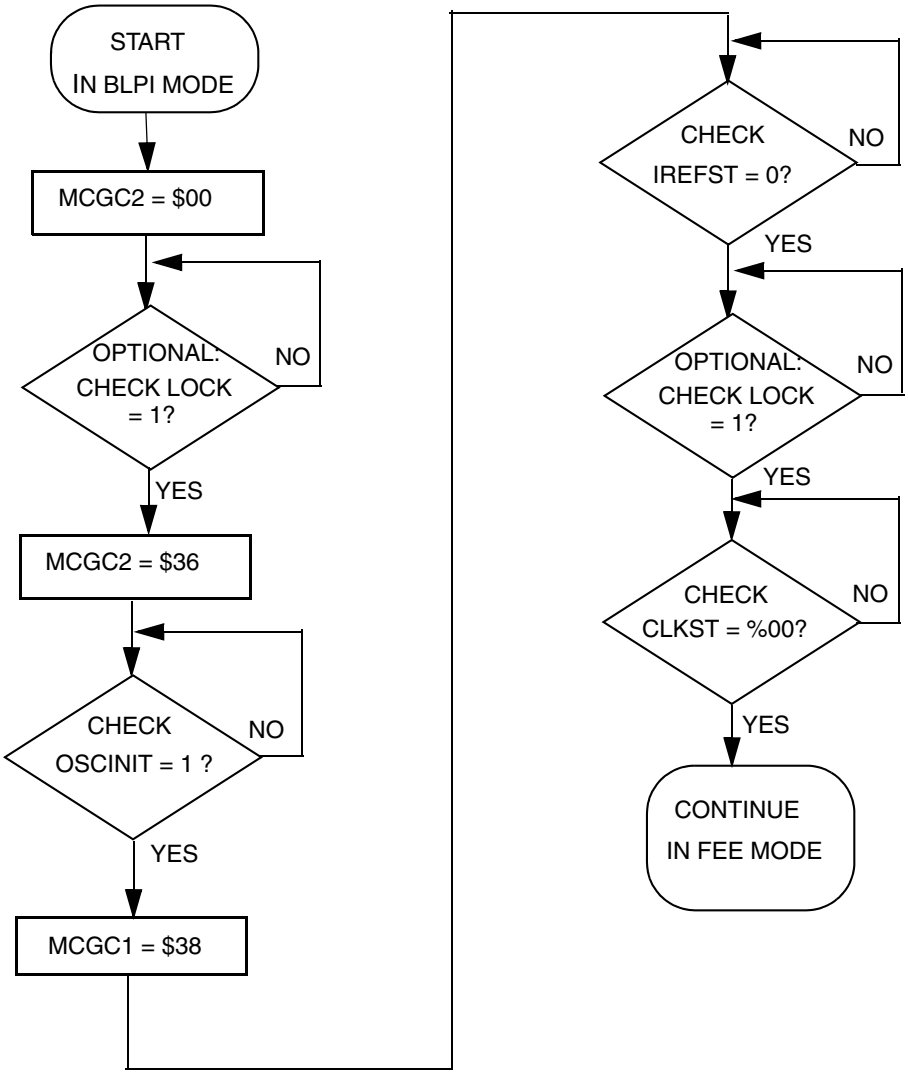


Figure 12-11. Flowchart of BLPI to FEE Mode Transition using a 4 MHz Crystal

12.5.2.4 Example # 4: Moving from FEI to PEE Mode: External Crystal = 8 MHz, Bus Frequency = 8 MHz

In this example, the MCG will move through the proper operational modes from FEI to PEE mode until the 8 MHz crystal reference frequency is set to achieve a bus frequency of 8 MHz.

This example is similar to example number one except that in this case the frequency of the external crystal is 8 MHz instead of 4 MHz. Special consideration must be taken with this case since there is a period of time along the way from FEI mode to PEE mode where the FLL operates based on a reference clock with a frequency that is greater than the maximum allowed for the FLL. This occurs because with an 8 MHz

external crystal and a maximum reference divider factor of 128, the resulting frequency of the reference clock for the FLL is 62.5 kHz (greater than the 39.0625 kHz maximum allowed).

Care must be taken in the software to minimize the amount of time spent in this state where the FLL is operating in this condition.

The following code sequence describes how to move from FEI mode to PEE mode until the 8 MHz crystal reference frequency is set to achieve a bus frequency of 8 MHz. Because the MCG is in FEI mode out of reset, this example also shows how to initialize the MCG for PEE mode out of reset. First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, FEI must transition to FBE mode:
   a) MCGC2 = 0x36 (%00110110)
      – BDIV (bits 7 and 6) set to %00, or divide-by-1
      – RANGE (bit 5) set to 1 because the frequency of 8 MHz is within the high frequency range
      – HGO (bit 4) set to 1 to configure external oscillator for high gain operation
      – EREFS (bit 2) set to 1, because a crystal is being used
      – ERCLKEN (bit 1) set to 1 to ensure the external reference clock is active
   b) Loop until OSCINIT (bit 1) in MCGSC is 1, indicating the crystal selected by the EREFS bit has been initialized.
   c) Block Interrupts (If applicable by setting the interrupt bit in the CCR).
   d) MCGC1 = 0xB8 (%10111000)
      – CLKS (bits 7 and 6) set to %10 in order to select external reference clock as system clock source
      – RDIV (bits 5-3) set to %111, or divide-by-128.

      NOTE

      8 MHz / 128 = 62.5 kHz which is greater than the 31.25 kHz to 39.0625 kHz range required by the FLL. Therefore after the transition to FBE is complete, software must progress through to BLPE mode immediately by setting the LP bit in MCGC2.

      – IREFS (bit 2) cleared to 0, selecting the external reference clock
   e) Loop until IREFST (bit 4) in MCGSC is 0, indicating the external reference is the current source for the reference clock
   f) Loop until CLKST (bits 3 and 2) in MCGSC are %10, indicating that the external reference clock is selected to feed MCGOUT
2. Then, FBE mode transitions into BLPE mode:
   a) MCGC2 = 0x3E (%00111110)
      – LP (bit 3) in MCGC2 to 1 (BLPE mode entered)

      NOTE

      There must be no extra steps (including interrupts) between steps 1d and 2a.

   b) Enable Interrupts (if applicable by clearing the interrupt bit in the CCR).

c) MCGC1 = 0x98 (%10011000)
   - RDIV (bits 5-3) set to %011, or divide-by-8 because 8 MHz / 8= 1 MHz which is in the 1 MHz to 2 MHz range required by the PLL. In BLPE mode, the configuration of the RDIV does not matter because both the FLL and PLL are disabled. Changing them only sets up the the dividers for PLL usage in PBE mode

d) MCGC3 = 0x44 (%01000100)
   - PLLS (bit 6) set to 1, selects the PLL. In BLPE mode, changing this bit only prepares the MCG for PLL usage in PBE mode
   - VDIV (bits 3-0) set to %0100, or multiply-by-16 because 1 MHz reference * 16 = 16 MHz. In BLPE mode, the configuration of the VDIV bits does not matter because the PLL is disabled. Changing them only sets up the multiply value for PLL usage in PBE mode

e) Loop until PLLST (bit 5) in MCGSC is set, indicating that the current source for the PLLS clock is the PLL

3. Then, BLPE mode transitions into PBE mode:
   a) Clear LP (bit 3) in MCGC2 to 0 here to switch to PBE mode
   b) Then loop until LOCK (bit 6) in MCGSC is set, indicating that the PLL has acquired lock
4. Last, PBE mode transitions into PEE mode:
   a) MCGC1 = 0x18 (%00011000)
      - CLKS (bits7 and 6) in MCGSC1 set to %00 in order to select the output of the PLL as the system clock source
      - Loop until CLKST (bits 3 and 2) in MCGSC are %11, indicating that the PLL output is selected to feed MCGOUT in the current clock mode
   b) Now, With an RDIV of divide-by-8, a BDIV of divide-by-1, and a VDIV of multiply-by-16, MCGOUT = [(8 MHz / 8) * 16] / 1 = 16 MHz, and the bus frequency is MCGOUT / 2, or 8 MHz

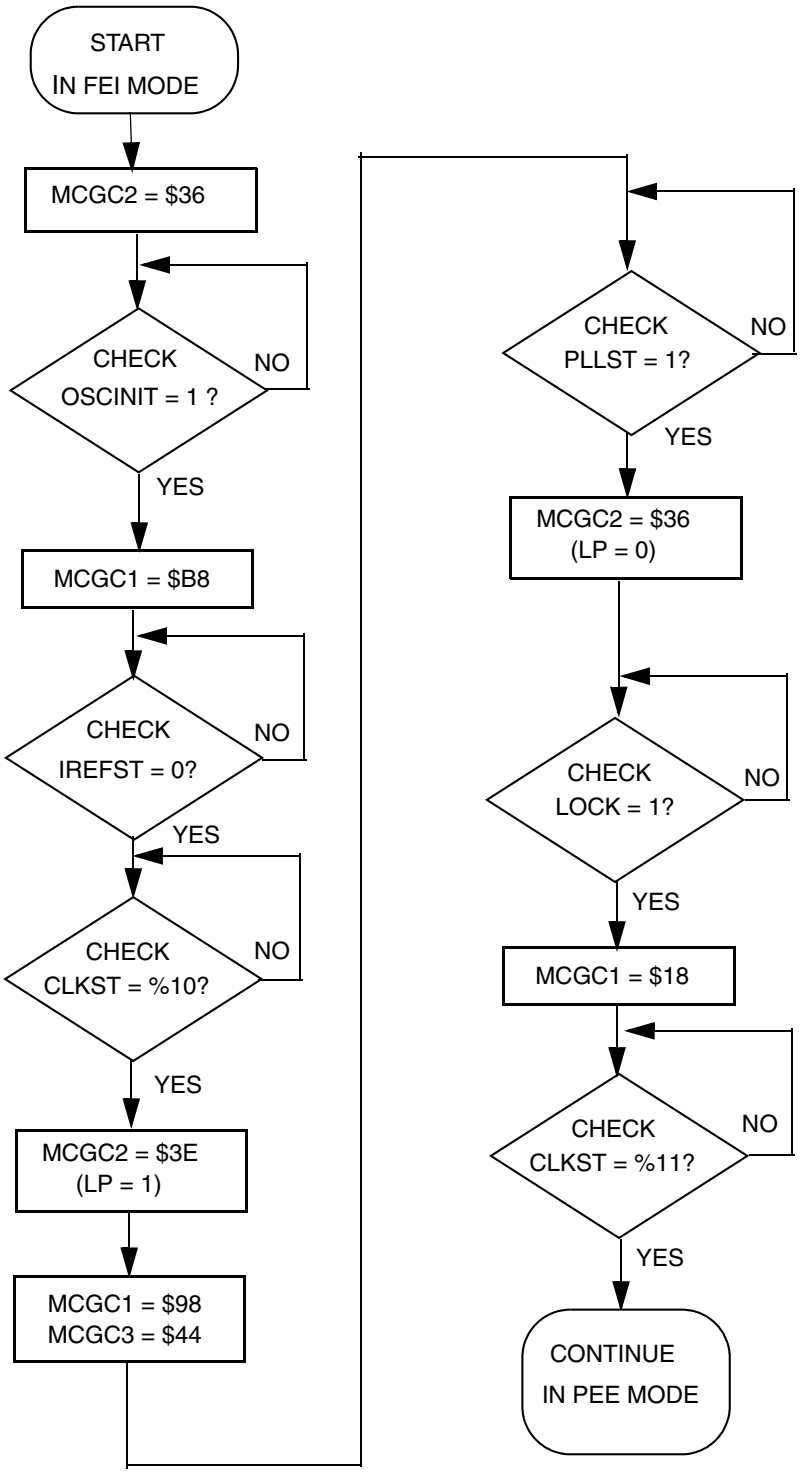


Figure 12-12. Flowchart of FEI to PEE Mode Transition using a 8 MHz Crystal

12.5.3 Calibrating the Internal Reference Clock (IRC)

The IRC is calibrated by writing to the MCGTRM register first, then using the FTRIM bit to “fine tune” the frequency. We will refer to this total 9-bit value as the trim value, ranging from 0x000 to 0x1FF, where the FTRIM bit is the LSB.

The trim value after a POR is always 0x100 (MCGTRM = 0x80 and FTRIM = 0). Writing a larger value will decrease the frequency and smaller values will increase the frequency. The trim value is linear with the period, except that slight variations in wafer fab processing produce slight non-linearities between trim value and period. These non-linearities are why an iterative trimming approach to search for the best trim value is recommended. In example #4 later in this section, this approach will be demonstrated.

After a trim value has been found for a device, this value can be stored in FLASH memory to save the value. If power is removed from the device, the IRC can easily be re-trimmed by copying the saved value from FLASH to the MCG registers. Freescale identifies recommended FLASH locations for storing the trim value for each MCU. Consult the memory map in the data sheet for these locations. On devices that are factory trimmed, the factory trim value will be stored in these locations.

12.5.3.1 Example #5: Internal Reference Clock Trim

For applications that require a tight frequency tolerance, a trimming procedure is provided that will allow a very accurate internal clock source. This section outlines one example of trimming the internal oscillator. Many other possible trimming procedures are valid and can be used.

In the example below, the MCG trim will be calibrated for the 9-bit MCGTRM and FTRIM collective value. This value will be referred to as TRMVAL.

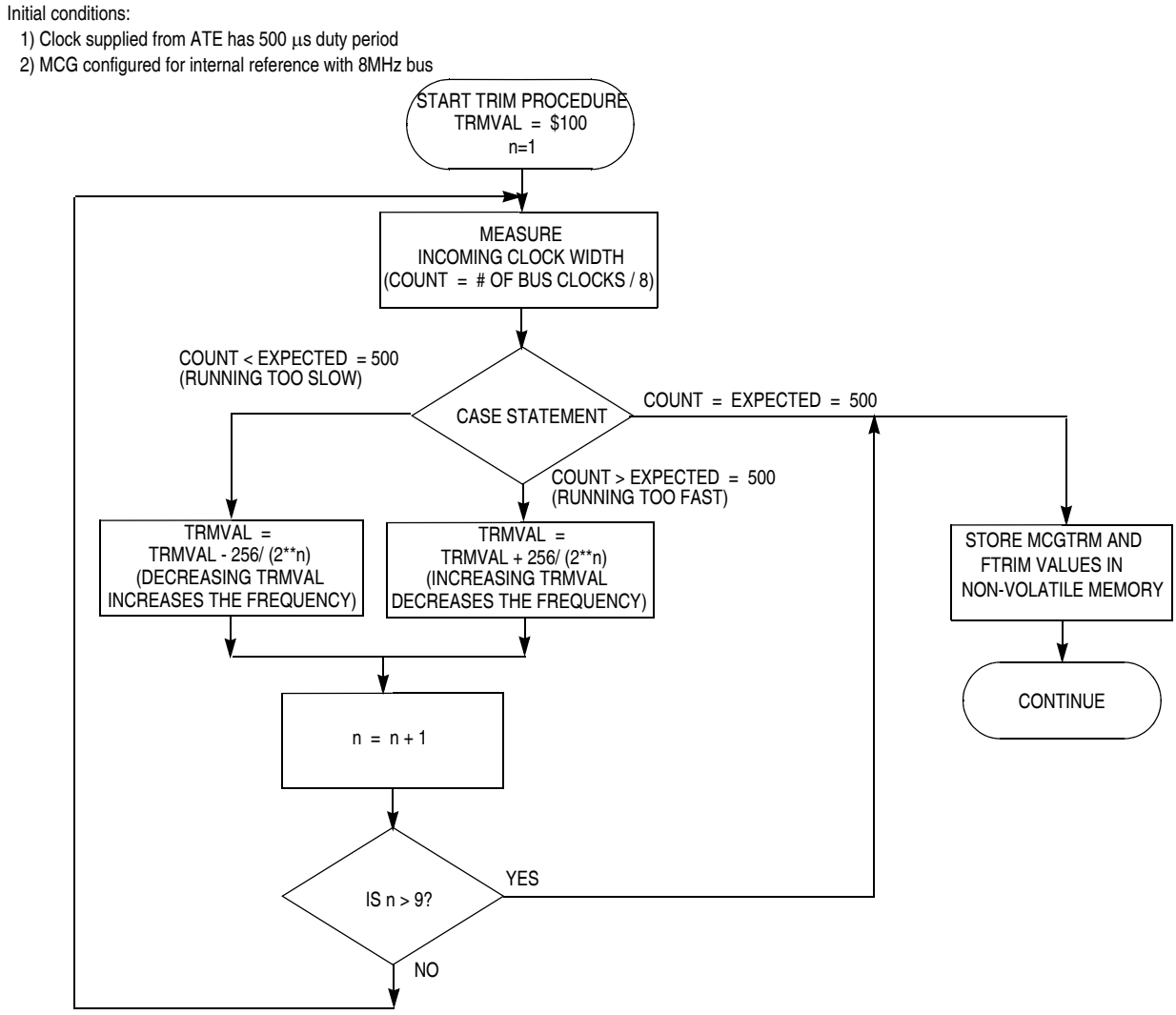


Figure 12-13. Trim Procedure

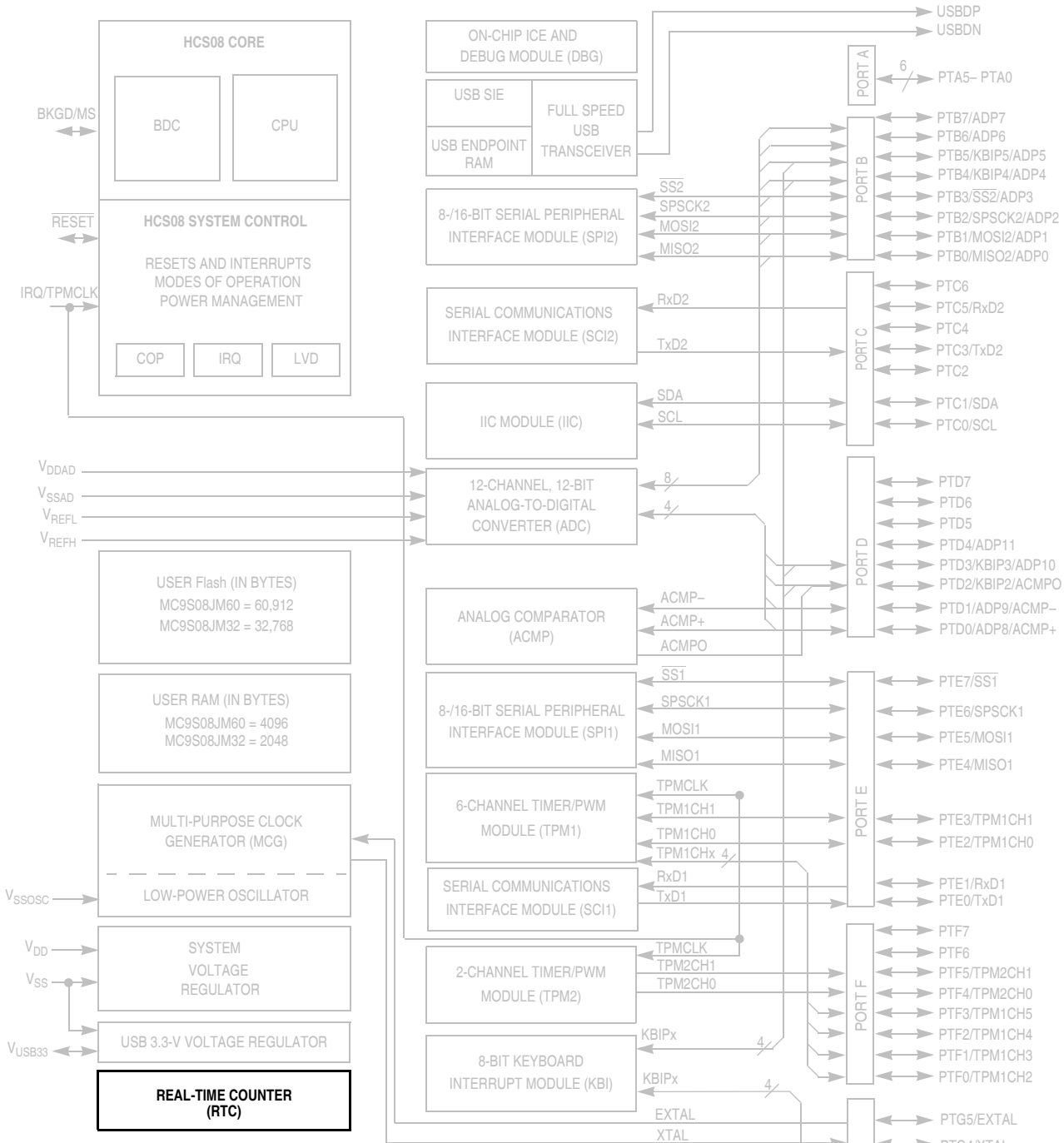
In this particular case, the MCU has been attached to a PCB and the entire assembly is undergoing final test with automated test equipment. A separate signal or message is provided to the MCU operating under user provided software control. The MCU initiates a trim procedure as outlined in Figure 12-13 while the tester supplies a precision reference signal.

If the intended bus frequency is near the maximum allowed for the device, it is recommended to trim using a reference divider value (RDIV setting) of twice the final value. After the trim procedure is complete, the reference divider can be restored. This will prevent accidental overshoot of the maximum clock frequency.

Chapter 13
Real-Time Counter (S08RTCV1)

13.1 Introduction

The real-time counter (RTC) consists of one 8-bit counter, one 8-bit comparator, several binary-based and decimal-based prescaler dividers, two clock sources, and one programmable periodic interrupt. This module can be used for time-of-day, calendar or any task scheduling functions. It can also serve as a cyclic wake up from low power modes without the need of external components.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 13-1. MC9S08JM60 Series Block Diagram Highlighting RTC Block

13.1.1 Features

Features of the RTC module include:

- 8-bit up-counter
  - 8-bit modulo match limit
  - Software controllable periodic interrupt on match
- Three software selectable clock sources for input to prescaler with selectable binary-based and decimal-based divider values
  - 1 kHz internal low-power oscillator (LPO)
  - External clock (ERCLK)
  - 32 kHz internal clock (IRCLK)

13.1.2 Modes of Operation

This section defines the operation in stop, wait and background debug modes.

13.1.2.1 Wait Mode

The RTC continues to run in wait mode if enabled before executing the appropriate instruction. Therefore, the RTC can bring the MCU out of wait mode if the real-time interrupt is enabled. For lowest possible current consumption, the RTC should be stopped by software if not needed as an interrupt source during wait mode.

13.1.2.2 Stop Modes

The RTC continues to run in stop2 or stop3 mode if the RTC is enabled before executing the STOP instruction. Therefore, the RTC can bring the MCU out of stop modes with no external components, if the real-time interrupt is enabled.

The LPO clock can be used in stop2 and stop3 modes. ERCLK and IRCLK clocks are only available in stop3 mode.

Power consumption is lower when all clock sources are disabled, but in that case, the real-time interrupt cannot wake up the MCU from stop modes.

13.1.2.3 Active Background Mode

The RTC suspends all counting during active background mode until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as the RTCMOD register is not written and the RTCPS and RTCLKS bits are not altered.

13.1.3 Block Diagram

The block diagram for the RTC module is shown in Figure 13-2.

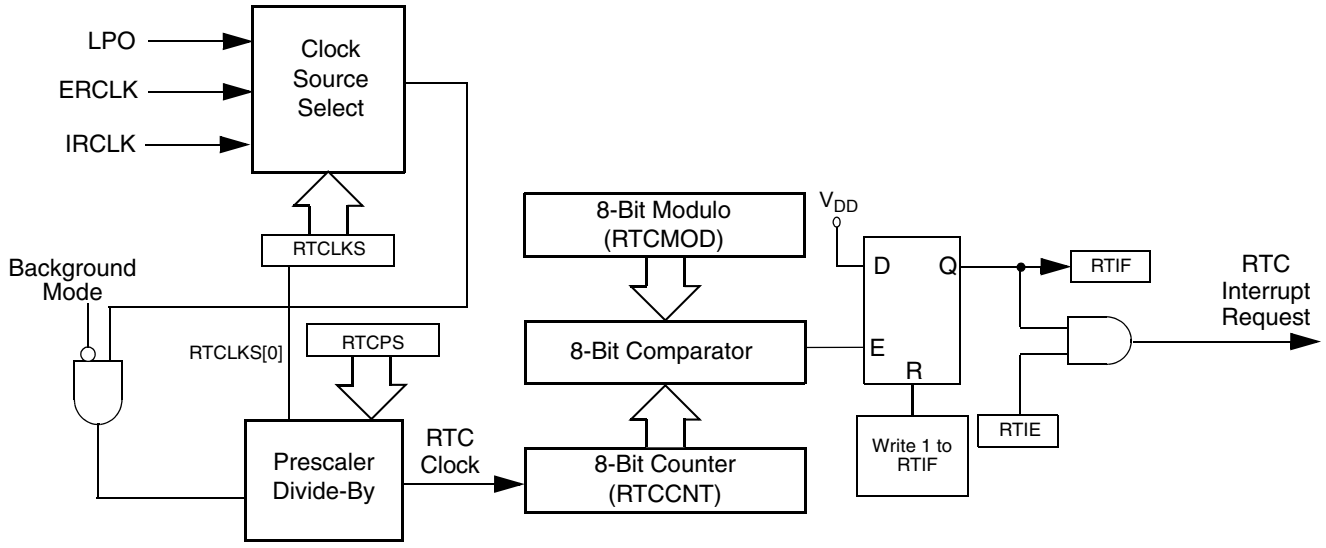


Figure 13-2. Real-Time Counter (RTC) Block Diagram

13.2 External Signal Description

The RTC does not include any off-chip signals.

13.3 Register Definition

The RTC includes a status and control register, an 8-bit counter register, and an 8-bit modulo register.

Refer to the direct-page register summary in the memory section of this document for the absolute address assignments for all RTC registers.This section refers to registers and control bits only by their names and relative address offsets.

Table 13-1 is a summary of RTC registers.

Table 13-1. RTC Register Summary

Name		7	6	5	4	3	2	1	0
RTCSC	R	RTIF	RTCLKS		RTIE	RTCPS			
	W								
RTCCNT	R	RTCCNT							
	W								
RTCMOD	R	RTCMOD							
	W								

13.3.1 RTC Status and Control Register (RTCSC)

RTCSC contains the real-time interrupt status flag (RTIF), the clock select bits (RTCLKS), the real-time interrupt enable bit (RTIE), and the prescaler select bits (RTCPS).

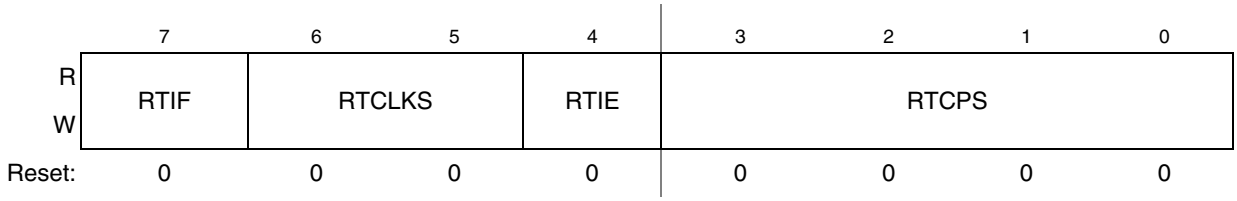


Figure 13-3. RTC Status and Control Register (RTCSC)

Table 13-2. RTCSC Field Descriptions

Field	Description
7 RTIF	Real-Time Interrupt Flag This status bit indicates the RTC counter register reached the value in the RTC modulo register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RTIF. 0 RTC counter has not reached the value in the RTC modulo register. 1 RTC counter has reached the value in the RTC modulo register.
6–5 RTCLKS	Real-Time Clock Source Select. These two read/write bits select the clock source input to the RTC prescaler. Changing the clock source clears the prescaler and RTCCNT counters. When selecting a clock source, ensure that the clock source is properly enabled (if applicable) to ensure correct operation of the RTC. Reset clears RTCLKS. 00 Real-time clock source is the 1-kHz low power oscillator (LPO) 01 Real-time clock source is the external clock (ERCLK) 1x Real-time clock source is the internal clock (IRCLK)
4 RTIE	Real-Time Interrupt Enable. This read/write bit enables real-time interrupts. If RTIE is set, then an interrupt is generated when RTIF is set. Reset clears RTIE. 0 Real-time interrupt requests are disabled. Use software polling. 1 Real-time interrupt requests are enabled.
3–0 RTCPS	Real-Time Clock Prescaler Select. These four read/write bits select binary-based or decimal-based divide-by values for the clock source. See Table 13-3. Changing the prescaler value clears the prescaler and RTCCNT counters. Reset clears RTCPS.

Table 13-3. RTC Prescaler Divide-by values

RTCLKS[0]	RTCPS															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	Off	2^3	2^5	2^6	2^7	2^8	2^9	2^{10}	1	2	2^2	10	2^4	10^2	$5x10^2$	10^3
1	Off	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	10^3	$2x10^3$	$5x10^3$	10^4	$2x10^4$	$5x10^4$	10^5	$2x10^5$

13.3.2 RTC Counter Register (RTCCNT)

RTCCNT is the read-only value of the current RTC count of the 8-bit counter.

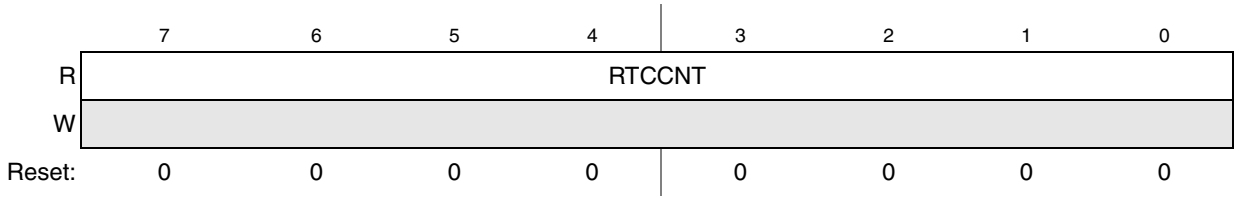


Figure 13-4. RTC Counter Register (RTCCNT)

Table 13-4. RTCCNT Field Descriptions

Field	Description
7:0 RTCCNT	RTC Count. These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset, writing to RTCMOD, or writing different values to RTCLKS and RTCPS clear the count to 0x00.

13.3.3 RTC Modulo Register (RTCMOD)

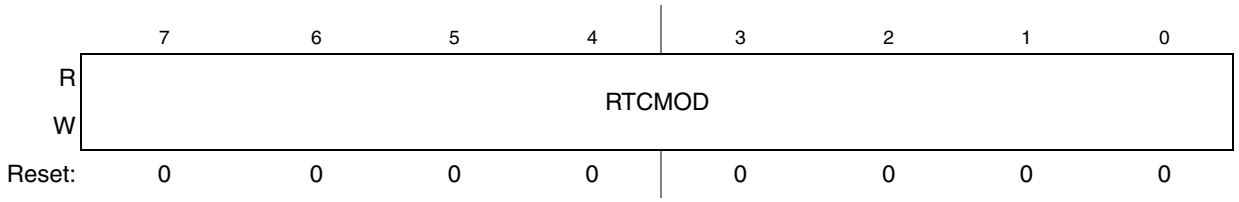


Figure 13-5. RTC Modulo Register (RTCMOD)

Table 13-5. RTCMOD Field Descriptions

Field	Description
7:0 RTCMOD	RTC Modulo. These eight read/write bits contain the modulo value used to reset the count to 0x00 upon a compare match and set the RTIF status bit. A value of 0x00 sets the RTIF bit on each rising edge of the prescaler output. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00. Reset sets the modulo to 0x00.

13.4 Functional Description

The RTC is composed of a main 8-bit up-counter with an 8-bit modulo register, a clock source selector, and a prescaler block with binary-based and decimal-based selectable values. The module also contains software selectable interrupt logic.

After any MCU reset, the counter is stopped and reset to 0x00, the modulus register is set to 0x00, and the prescaler is off. The 1-kHz internal oscillator clock is selected as the default clock source. To start the prescaler, write any value other than zero to the prescaler select bits (RTCPS).

Three clock sources are software selectable: the low power oscillator clock (LPO), the external clock (ERCLK), and the internal clock (IRCLK). The RTC clock select bits (RTCLKS) select the desired clock source. If a different value is written to RTCLKS, the prescaler and RTCCNT counters are reset to 0x00.

RTCPS and the RTCLKS[0] bit select the desired divide-by value. If a different value is written to RTCPS, the prescaler and RTCCNT counters are reset to 0x00. Table 13-6 shows different prescaler period values.

Table 13-6. Prescaler Period

RTCPS	1-kHz Internal Clock (RTCLKS = 00)	1-MHz External Clock (RTCLKS = 01)	32-kHz Internal Clock (RTCLKS = 10)	32-kHz Internal Clock (RTCLKS = 11)
0000	Off	Off	Off	Off
0001	8 ms	1.024 ms	250 μs	32 ms
0010	32 ms	2.048 ms	1 ms	64 ms
0011	64 ms	4.096 ms	2 ms	128 ms
0100	128 ms	8.192 ms	4 ms	256 ms
0101	256 ms	16.4 ms	8 ms	512 ms
0110	512 ms	32.8 ms	16 ms	1.024 s
0111	1.024 s	65.5 ms	32 ms	2.048 s
1000	1 ms	1 ms	31.25 μs	31.25 ms
1001	2 ms	2 ms	62.5 μs	62.5 ms
1010	4 ms	5 ms	125 μs	156.25 ms
1011	10 ms	10 ms	312.5 μs	312.5 ms
1100	16 ms	20 ms	0.5 ms	0.625 s
1101	0.1 s	50 ms	3.125 ms	1.5625 s
1110	0.5 s	0.1 s	15.625 ms	3.125 s
1111	1 s	0.2 s	31.25 ms	6.25 s

The RTC modulo register (RTCMOD) allows the compare value to be set to any value from 0x00 to 0xFF. When the counter is active, the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter resets to 0x00 and continues counting. The real-time interrupt flag (RTIF) is set when a match occurs. The flag sets on the transition from the modulo value to 0x00. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00.

The RTC allows for an interrupt to be generated when RTIF is set. To enable the real-time interrupt, set the real-time interrupt enable bit (RTIE) in RTCSC. RTIF is cleared by writing a 1 to RTIF.

13.4.1 RTC Operation Example

This section shows an example of the RTC operation as the counter reaches a matching value from the modulo register.

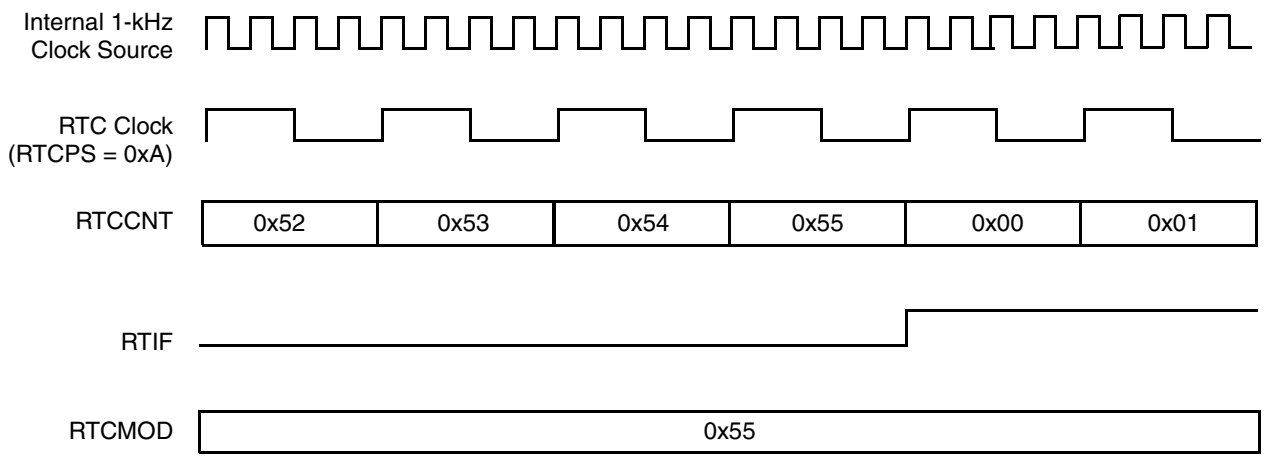


Figure 13-6. RTC Counter Overflow Example

In the example of Figure 13-6, the selected clock source is the 1-kHz internal oscillator clock source. The prescaler (RTCPS) is set to 0xA or divide-by-4. The modulo value in the RTCMOD register is set to 0x55. When the counter, RTCCNT, reaches the modulo value of 0x55, the counter overflows to 0x00 and continues counting. The real-time interrupt flag, RTIF, sets when the counter value changes from 0x55 to 0x00. A real-time interrupt is generated when RTIF is set, if RTIE is set.

13.5 Initialization/Application Information

This section provides example code to give some basic direction to a user on how to initialize and configure the RTC module. The example software is implemented in C language.

The example below shows how to implement time of day with the RTC using the 1-kHz clock source to achieve the lowest possible power consumption. Because the 1-kHz clock source is not as accurate as a crystal, software can be added for any adjustments. For accuracy without adjustments at the expense of additional power consumption, the external clock (ERCLK) or the internal clock (IRCLK) can be selected with appropriate prescaler and modulo values.

```
/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;

/* Configure RTC to interrupt every 1 second from 1-kHz clock source */
RTCMOD.byte = 0x00;
RTCSC.byte = 0x1F;

/*********************************************************************
Function Name : RTC_ISR
Notes : Interrupt service routine for RTC module.
*********************************************************************/
#pragma TRAP_PROC
void RTC_ISR(void)
{
        /* Clear the interrupt flag */
```

```
RTCSC.byte = RTCSC.byte | 0x80;
/* RTC interrupts every 1 Second */
Seconds++;
/* 60 seconds in a minute */
if (Seconds > 59){
Minutes++;
Seconds = 0;
}
/* 60 minutes in an hour */
if (Minutes > 59){
Hours++;
Minutes = 0;
}
/* 24 hours in a day */
if (Hours > 23){
Days ++;
Hours = 0;
}
```

Chapter 14
Serial Communications Interface (S08SCIV4)

14.1 Introduction

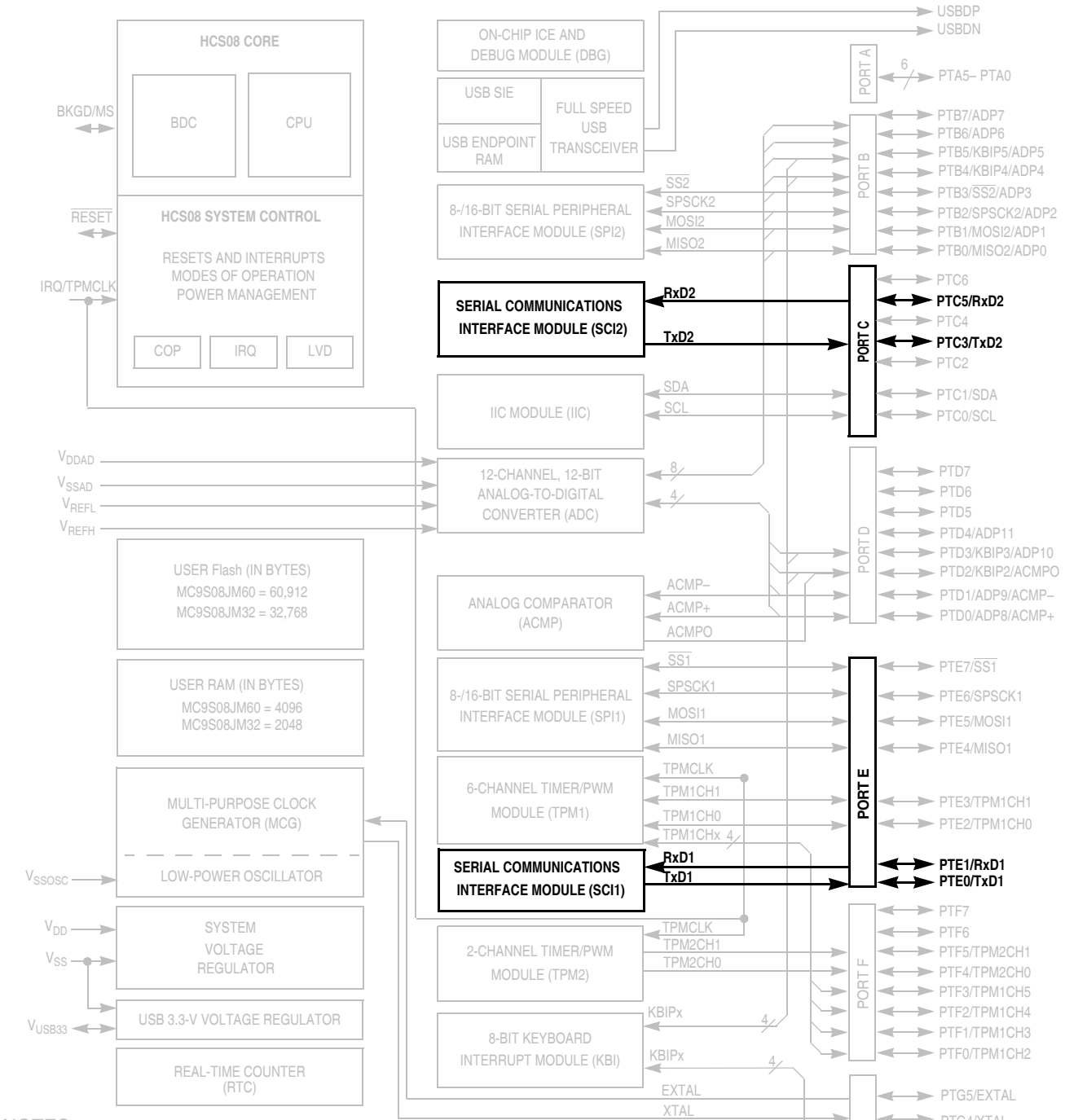
The MC9S08JM60 series include two independent serial communications interface (SCI) modules which are sometimes called universal asynchronous receiver/transmitters (UARTs). Typically, these systems are used to connect to the RS232 serial input/output (I/O) port of a personal computer or workstation, but they can also be used to communicate with other embedded controllers.

A flexible, 13-bit, modulo-based baud rate generator supports a broad range of standard baud rates beyond 115.2 kbaud. Transmit and receive within the same SCI use a common baud rate, and each SCI module has a separate baud rate generator.

This SCI system offers many advanced features not commonly found on other asynchronous serial I/O peripherals on other embedded controllers. The receiver employs an advanced data sampling technique that ensures reliable communication and noise detection. Hardware parity, receiver wakeup, and double buffering on transmit and receive are also included.

NOTE

MC9S08JM60 series devices operate at a higher voltage range (2.7 V to 5.5 V) and do not include stop1 mode. Therefore, please disregard references to stop1.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 14-1. MC9S08JM60 Series Block Diagram Highlighting the SCI Blocks and Pins

14.1.1 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

14.1.2 Modes of Operation

See Section 14.3, “Functional Description,” For details concerning SCI operation in these modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

14.1.3 Block Diagram

Figure 14-2 shows the transmitter portion of the SCI.

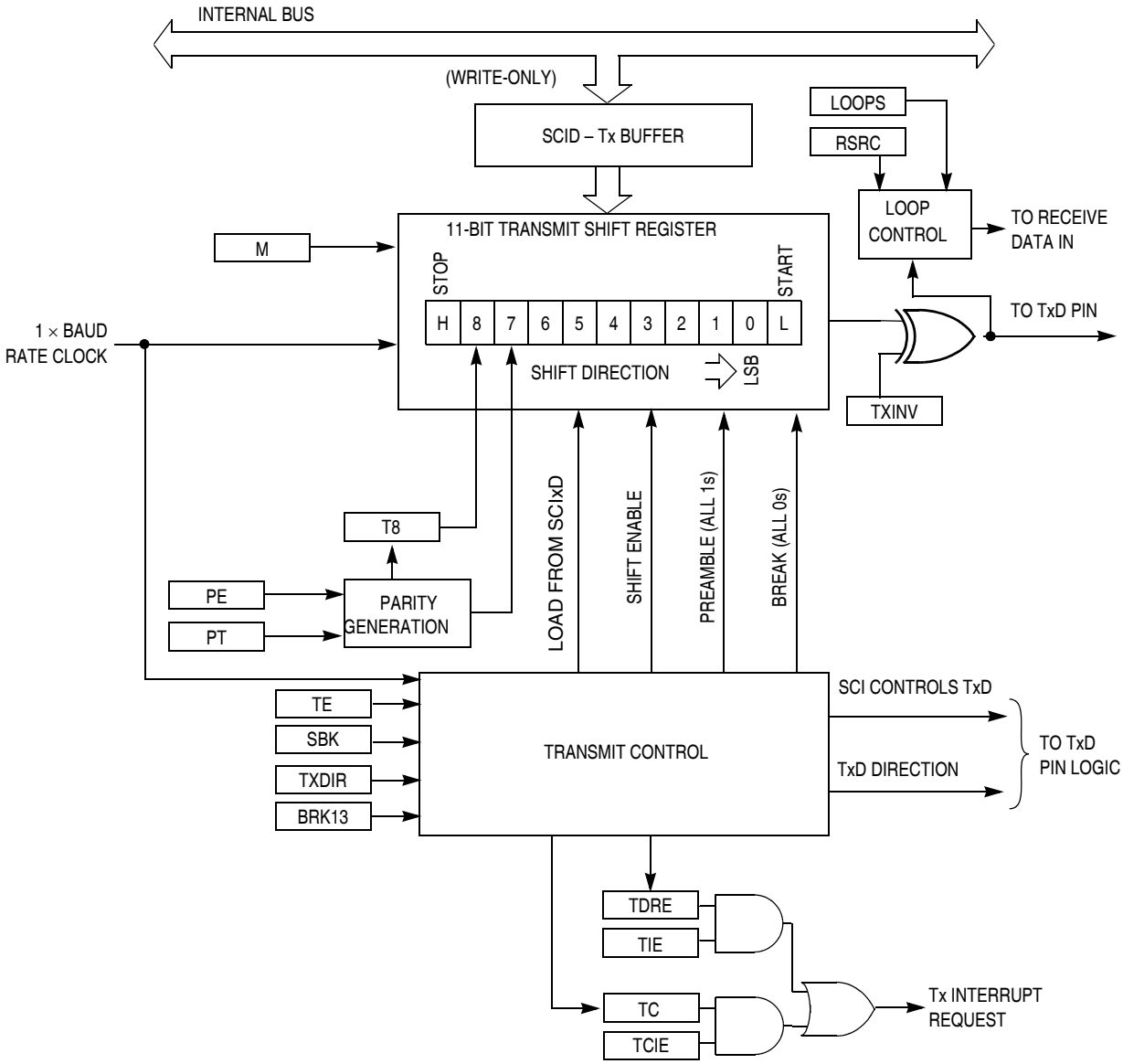


Figure 14-2. SCI Transmitter Block Diagram

Figure 14-3 shows the receiver portion of the SCI.

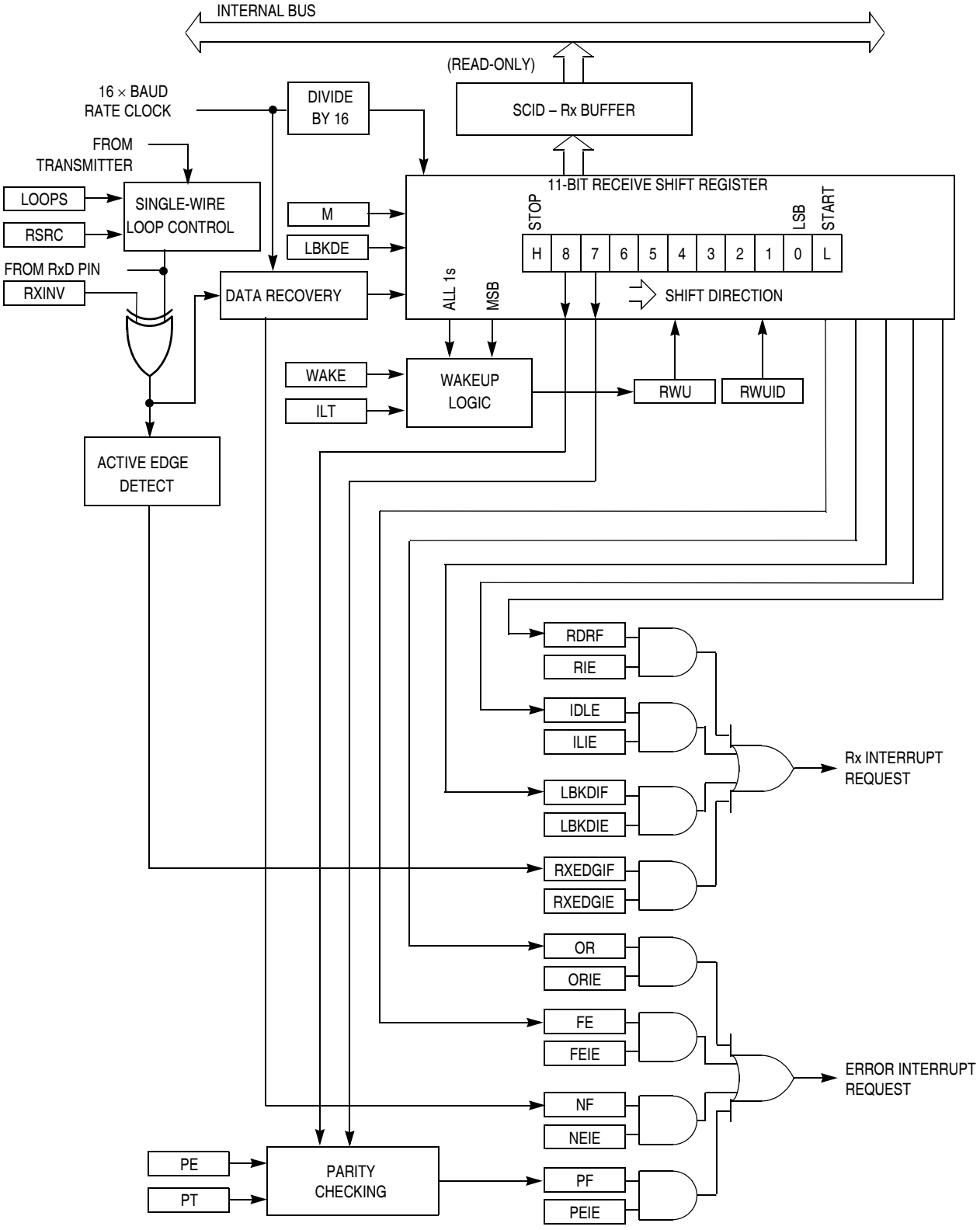


Figure 14-3. SCI Receiver Block Diagram

14.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all SCI registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

14.2.1 SCI Baud Rate Registers (SCIxBDH, SCIxBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIxBDH to buffer the high half of the new value and then write to SCIxBDL. The working value in SCIxBDH does not change until SCIxBDL is written.

SCIxBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIxC2 are written to 1).

	7	6	5	4	3	2	1	0
R	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 14-4. SCI Baud Rate Register (SCIxBDH)

Table 14-1. SCIxBDH Field Descriptions

Field	Description
7 LBKDIE	**LIN Break Detect Interrupt Enable (for LBKDIF)** 0 Hardware interrupts from LBKDIF disabled (use polling). 1 Hardware interrupt requested when LBKDIF flag is 1.
6 RXEDGIE	**RxD Input Active Edge Interrupt Enable (for RXEDGIF)** 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4:0 SBR[12:8]	**Baud Rate Modulo Divisor** — The 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 14-2.

	7	6	5	4	3	2	1	0
R W	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
Reset	0	0	0	0	0	1	0	0

Figure 14-5. SCI Baud Rate Register (SCIxBDL)

Table 14-2. SCIxBDL Field Descriptions

Field	Description
7:0 SBR[7:0]	**Baud Rate Modulo Divisor** — These 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 14-1.

14.2.2 SCI Control Register 1 (SCIxC1)

This read/write register is used to control various optional features of the SCI system.

	7	6	5	4	3	2	1	0
R W	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0

Figure 14-6. SCI Control Register 1 (SCIxC1)

Table 14-3. SCIxC1 Field Descriptions

Field	Description
7 LOOPS	**Loop Mode Select** — Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input. 0 Normal operation — RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See RSRC bit.) RxD pin is not used by SCI.
6 SCISWAI	**SCI Stops in Wait Mode** 0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU. 1 SCI clocks freeze while CPU is in wait mode.
5 RSRC	**Receiver Source Select** — This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 0 Provided LOOPS = 1, RSRC = 0 selects internal loop back mode and the SCI does not use the RxD pins. 1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	**9-Bit or 8-Bit Mode Select** 0 Normal — start + 8 data bits (LSB first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (LSB first) + 9th data bit + stop.

Table 14-3. SCIxC1 Field Descriptions (continued)

Field	Description
3 WAKE	**Receiver Wakeup Method Select** — Refer to Section 14.3.3.2, "Receiver Wakeup Operation" for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	**Idle Line Type Select** — Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of logic high level needed by the idle line detection logic. Refer to Section 14.3.3.2.1, "Idle-Line Wakeup" for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.
1 PE	**Parity Enable** — Enables hardware parity generation and checking. When parity is enabled, the most significant bit (MSB) of the data character (eighth or ninth data bit) is treated as the parity bit. 0 No hardware parity generation or checking. 1 Parity enabled.
0 PT	**Parity Type** — Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0 Even parity. 1 Odd parity.

14.2.3 SCI Control Register 2 (SCIxC2)

This register can be read or written at any time.

	7	6	5	4	3	2	1	0
R W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset	0	0	0	0	0	0	0	0

Figure 14-7. SCI Control Register 2 (SCIxC2)

Table 14-4. SCIxC2 Field Descriptions

Field	Description
7 TIE	**Transmit Interrupt Enable (for TDRE)** 0 Hardware interrupts from TDRE disabled (use polling). 1 Hardware interrupt requested when TDRE flag is 1.
6 TCIE	**Transmission Complete Interrupt Enable (for TC)** 0 Hardware interrupts from TC disabled (use polling). 1 Hardware interrupt requested when TC flag is 1.
5 RIE	**Receiver Interrupt Enable (for RDRF)** 0 Hardware interrupts from RDRF disabled (use polling). 1 Hardware interrupt requested when RDRF flag is 1.
4 ILIE	**Idle Line Interrupt Enable (for IDLE)** 0 Hardware interrupts from IDLE disabled (use polling). 1 Hardware interrupt requested when IDLE flag is 1.

Table 14-4. SCIxC2 Field Descriptions (continued)

Field	Description
3 TE	**Transmitter Enable** 0 Transmitter off. 1 Transmitter on. TE must be 1 in order to use the SCI transmitter. When TE = 1, the SCI forces the TxD pin to act as an output for the SCI system. When the SCI is configured for single-wire operation (LOOPS = RSRC = 1), TXDIR controls the direction of traffic on the single SCI communication line (TxD pin). TE also can be used to queue an idle character by writing TE = 0 then TE = 1 while a transmission is in progress. Refer to Section 14.3.2.1, "Send Break and Queued Idle" for more details. When TE is written to 0, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin.
2 RE	**Receiver Enable** — When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS = 1 the RxD pin reverts to being a general-purpose I/O pin even if RE = 1. 0 Receiver off. 1 Receiver on.
1 RWU	**Receiver Wakeup Control** — This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is either an idle line between messages (WAKE = 0, idle-line wakeup), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to Section 14.3.3.2, "Receiver Wakeup Operation" for more details. 0 Normal SCI receiver operation. 1 SCI receiver in standby waiting for wakeup condition.
0 SBK	**Send Break** — Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 (13 or 14 if BRK13 = 1) bit times of logic 0 are queued as long as SBK = 1. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to Section 14.3.2.1, "Send Break and Queued Idle" for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

14.2.4 SCI Status Register 1 (SCIxS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) are used to clear these status flags.

	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W								
Reset	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 14-8. SCI Status Register 1 (SCIxS1)

Table 14-5. SCIxS1 Field Descriptions

Field	Description
7 TDRE	**Transmit Data Register Empty Flag** — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIxS1 with TDRE = 1 and then write to the SCI data register (SCIxD). 0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.
6 TC	**Transmission Complete Flag** — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted. 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete). TC is cleared automatically by reading SCIxS1 with TC = 1 and then doing one of the following three things: • Write to the SCI data register (SCIxD) to transmit new data • Queue a preamble by changing TE from 0 to 1 • Queue a break character by writing 1 to SBK in SCIxC2
5 RDRF	**Receive Data Register Full Flag** — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCIxD). To clear RDRF, read SCIxS1 with RDRF = 1 and then read the SCI data register (SCIxD). 0 Receive data register empty. 1 Receive data register full.
4 IDLE	**Idle Line Flag** — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. To clear IDLE, read SCIxS1 with IDLE = 1 and then read the SCI data register (SCIxD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period. 0 No idle line detected. 1 Idle line was detected.
3 OR	**Receiver Overrun Flag** — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCIxD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCIxD. To clear OR, read SCIxS1 with OR = 1 and then read the SCI data register (SCIxD). 0 No overrun. 1 Receive overrun (new SCI data lost).
2 NF	**Noise Flag** — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCIxS1 and then read the SCI data register (SCIxD). 0 No noise detected. 1 Noise detected in the received character in SCIxD.

Table 14-5. SCIxS1 Field Descriptions (continued)

Field	Description
1 FE	**Framing Error Flag** — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCIxS1 with FE = 1 and then read the SCI data register (SCIxD). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	**Parity Error Flag** — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCIxS1 and then read the SCI data register (SCIxD). 0 No parity error. 1 Parity error.

14.2.5 SCI Status Register 2 (SCIxS2)

This register has one read-only status flag.

	7	6	5	4	3	2	1	0
R	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 14-9. SCI Status Register 2 (SCIxS2)

Table 14-6. SCIxS2 Field Descriptions

Field	Description
7 LBKDIF	**LIN Break Detect Interrupt Flag** — LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. LBKDIF is cleared by writing a “1” to it. 0 No LIN break character has been detected. 1 LIN break character has been detected.
6 RXEDGIF	**RxD Pin Active Edge Interrupt Flag** — RXEDGIF is set when an active edge (falling if RXINV = 0, rising if RXINV=1) on the RxD pin occurs. RXEDGIF is cleared by writing a “1” to it. 0 No active edge on the receive pin has occurred. 1 An active edge on the receive pin has occurred.
4 RXINV[1]	**Receive Data Inversion** — Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	**Receive Wake Up Idle Detect**— RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	**Break Character Generation Length** — BRK13 is used to select a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)

Table 14-6. SCIxS2 Field Descriptions (continued)

Field	Description
1 LBKDE	**LIN Break Detection Enable**— LBKDE is used to select a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 0 Break character is detected at length of 10 bit times (11 if M = 1). 1 Break character is detected at length of 11 bit times (12 if M = 1).
0 RAF	**Receiver Active Flag** — RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxD input not idle).

[1] Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold by one bit time. Under the worst case timing conditions allowed in LIN, it is possible that a 0x00 data character can appear to be 10.26 bit times long at a slave which is running 14% faster than the master. This would trigger normal break detection circuitry which is designed to detect a 10 bit break symbol. When the LBKDE bit is set, framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.

14.2.6 SCI Control Register 3 (SCIxC3)

	7	6	5	4	3	2	1	0
R	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 14-10. SCI Control Register 3 (SCIxC3)

Table 14-7. SCIxC3 Field Descriptions

Field	Description
7 R8	**Ninth Data Bit for Receiver** — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the SCIxD register. When reading 9-bit data, read R8 before reading SCIxD because reading SCIxD completes automatic flag clearing sequences which could allow R8 and SCIxD to be overwritten with new data.
6 T8	**Ninth Data Bit for Transmitter** — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the SCIxD register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCIxD is written so T8 should be written (if it needs to change from its previous value) before SCIxD is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCIxD is written.
5 TXDIR	**TxD Pin Direction in Single-Wire Mode** — When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin. 0 TxD pin is an input in single-wire mode. 1 TxD pin is an output in single-wire mode.

Table 14-7. SCIxC3 Field Descriptions (continued)

Field	Description
4 TXINV[1]	**Transmit Data Inversion** — Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	**Overrun Interrupt Enable** — This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0 OR interrupts disabled (use polling). 1 Hardware interrupt requested when OR = 1.
2 NEIE	**Noise Error Interrupt Enable** — This bit enables the noise flag (NF) to generate hardware interrupt requests. 0 NF interrupts disabled (use polling). 1 Hardware interrupt requested when NF = 1.
1 FEIE	**Framing Error Interrupt Enable** — This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0 FE interrupts disabled (use polling). 1 Hardware interrupt requested when FE = 1.
0 PEIE	**Parity Error Interrupt Enable** — This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0 PF interrupts disabled (use polling). 1 Hardware interrupt requested when PF = 1.

[1] Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

14.2.7 SCI Data Register (SCIxD)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the SCI status flags.

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
Reset	0	0	0	0	0	0	0	0

Figure 14-11. SCI Data Register (SCIxD)

14.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

14.3.1 Baud Rate Generation

As shown in Figure 14-12, the clock source for the SCI baud rate generator is the bus-rate clock.

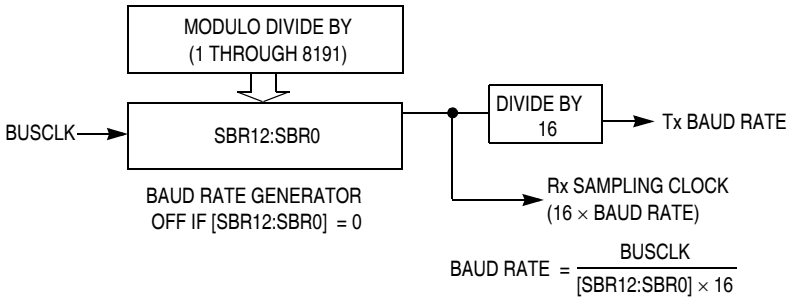


Figure 14-12. SCI Baud Rate Generation

SCI communications require the transmitter and receiver (which typically derive baud rates from independent clock sources) to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition, but in the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For a Freescale Semiconductor SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about ±4.5 percent for 8-bit data format and about ±4 percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

14.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in Figure 14-2.

The transmitter output (TxD) idle state defaults to logic high (TXINV = 0 following reset). The transmitter output is inverted by setting TXINV = 1. The transmitter is enabled by setting the TE bit in SCIxC2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCIxD).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, we will assume M = 0, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCIxD.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity that is in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

14.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIxC2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13 = 1. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin that is shared with TxD is an output driving a logic 1. This ensures that the TxD line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

Table 14-8. Break Character Length

BRK13	M	Break Character Length
0	0	10 bit times
0	1	11 bit times
1	0	13 bit times
1	1	14 bit times

14.3.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 14-3) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV = 1. The receiver is enabled by setting the RE bit in SCIxC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section 14.3.5.1, “8- and 9-Bit Data Modes.” For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF)

status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to Section 14.3.4, "Interrupts and Status Flags," for more details about flag clearing.

14.3.3.1 Data Sampling Technique

The SCI receiver uses a 16× baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The 16× baud rate clock is used to divide the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic level for that bit, the noise flag (NF) will be set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges, and if an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE is still set.

14.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIxC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant

message characters. At the end of a message, or at the beginning of the next message, all receivers automatically force RWU to 0 so all receivers wake up in time to look at the first character(s) of the next message.

14.3.3.2.1 Idle-Line Wakeup

When WAKE = 0, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits).

When RWU is one and RWUID is zero, the idle condition that wakes up the receiver does not set the IDLE flag. The receiver wakes up and waits for the first data character of the next message which will set the RDRF flag and generate an interrupt if enabled. When RWUID is one, any idle condition sets the IDLE flag and generates an interrupt if enabled, regardless of whether RWU is zero or one.

The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT = 0, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT = 1, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

14.3.3.2.2 Address-Mark Wakeup

When WAKE = 1, the receiver is configured for address-mark wakeup. In this mode, RWU is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character (eighth bit in M = 0 mode and ninth bit in M = 1 mode).

Address-mark wakeup allows messages to contain idle characters but requires that the MSB be reserved for use in address frames. The logic 1 MSB of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case the character with the MSB set is received even though the receiver was sleeping during most of this character time.

14.3.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF and LBKDIF events, and a third vector is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can still be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that optionally can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCIxD. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt will be requested whenever TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever TC = 1.

Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared by reading SCIxS1 while RDRF = 1 and then reading SCIxD.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIxS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD line remains idle for an extended period of time. IDLE is cleared by reading SCIxS1 while IDLE = 1 and then reading SCIxD. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the RXEDGIF flag to set. The RXEDGIF flag is cleared by writing a “1” to it. This function does depend on the receiver being enabled (RE = 1).

14.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

14.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIxC1. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIxC3. For the receiver, the ninth bit is held in R8 in SCIxC3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCIxD.

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCIxD to the shifter.

9-bit data mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

14.3.5.2 Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes. No SCI module registers are affected in stop3 mode.

The receive input active edge detect circuit is still active in stop3 mode, but not in stop2.. An active edge on the receive input brings the CPU out of stop3 mode if the interrupt is not masked (RXEDGIE = 1).

Note, because the clocks are halted, the SCI module will resume operation upon exit from stop (only in stop3 mode). Software should ensure stop mode is not entered while there is a character being transmitted out of or received into the SCI module.

14.3.5.3 Loop Mode

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RxD pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

14.3.5.4 Single-Wire Operation

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIxC3 controls the direction of serial data on the TxD pin. When TXDIR = 0, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When TXDIR = 1, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.

Chapter 15
16-Bit Serial Peripheral Interface (S08SPI16V1)

15.1 Introduction

The 8- or 16-bit selectable serial peripheral interface (SPI) module provides for full-duplex, synchronous, serial communication between the MCU and peripheral devices. These peripheral devices can include other microcontrollers, analog-to-digital converters, shift registers, sensors, memories, etc.

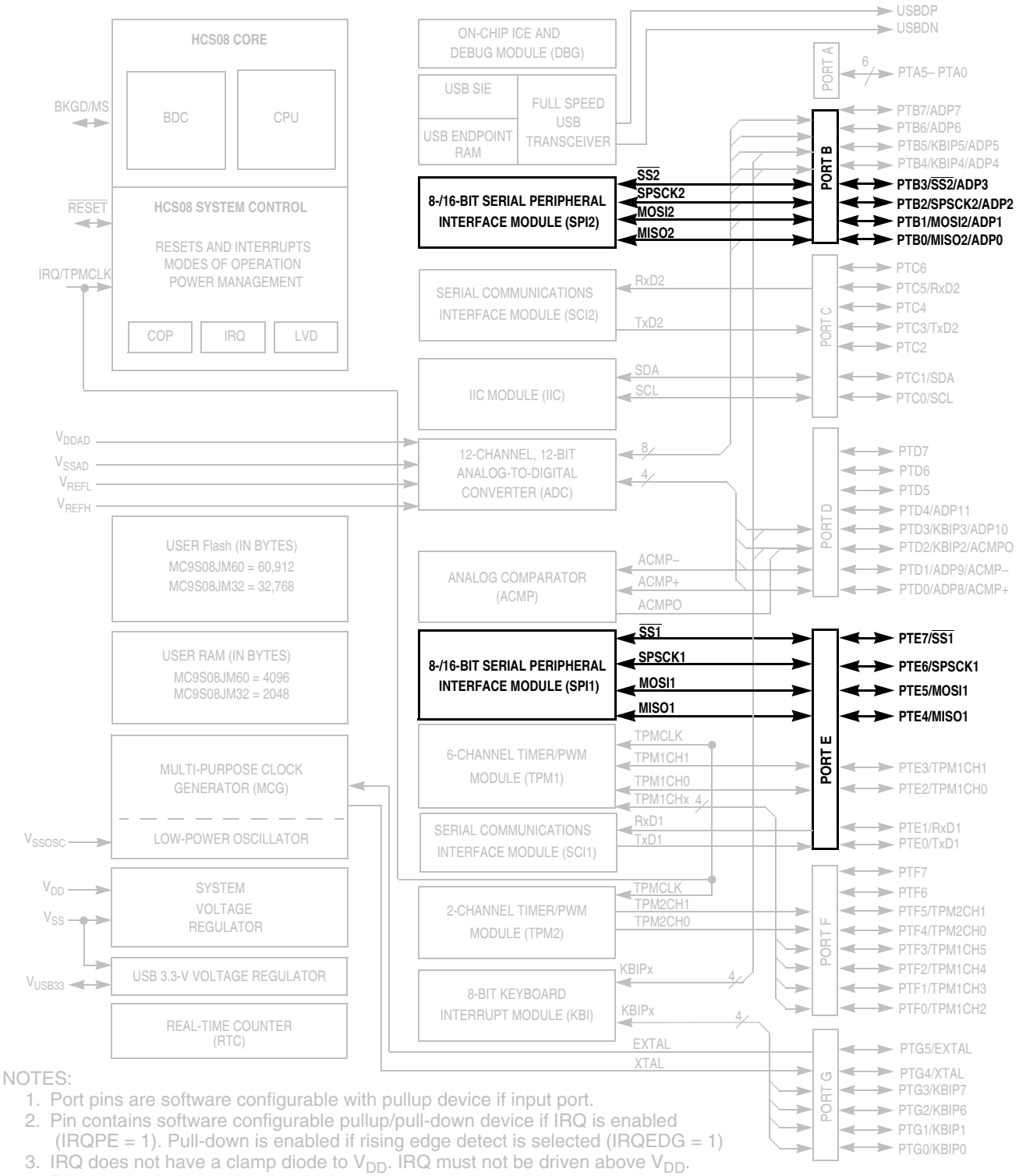
The SPI runs at a baud rate up to the bus clock divided by two in master mode and up to the bus clock divided by four in slave mode. Software can poll the status flags, or SPI operation can be interrupt driven.

The SPI also supports a data length of 8 or 16 bits and includes a hardware match feature for the receive data buffer.

The MC9S08JM60 series have two serial peripheral interface modules (SPI1 and SPI2). The four pins associated with SPI functionality are shared with PTB[3:0] and PTE[7:4]. See Appendix A, “Electrical Characteristics,” for SPI electrical parametric information.

15.1.1 SPI Port Configuration Information

By default, the input filters on the SPI port pins will be enabled (SPIxFE=1), which restricts the SPI data rate to 6 MHz, but protects the SPI from noise during data transfers.To configure the SPI at a baud rate of 6MHz or greater, the input filters on the SPI port pins must be disabled by clearing the SPIxFE in SOPT2. and also enable the high output drive strength selection on the affected SPI port pins.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 15-1. MC9S08JM60 Series Block Diagram Highlighting the SPI Blocks and Pins

Module Initialization (Slave):

Write:	SPIxC1	to configure	interrupts, set primary SPI options, slave mode select, and system enable.
Write:	SPIxC2	to configure	optional SPI features, hardware match interrupt enable, and 8- or 16-bit data transmission length
Write:	SPIxMH:SPIxML	to set	hardware compare value that triggers SPMF (optional) when value in receive data buffer equals this value.

Module Initialization (Master):

Write:	SPIxC1	to configure	interrupts, set primary SPI options, master mode select, and system enable.
Write:	SPIxC2	to configure	optional SPI features, hardware match interrupt enable, and 8- or 16-bit data transmission length
Write:	SPIxBR	to set	baud rate
Write:	SPIxMH:SPIxML	to set	hardware compare value that triggers SPMF (optional) when value in receive data buffer equals this value.

Module Use:

After SPI master initiates transfer by checking that SPTEF = 1 and then writing data to SPIDH/L:

Wait for SPRF, then read from SPIDH/L
Wait for SPTEF, then write to SPIDH/L

Data transmissions can be 8- or 16-bits long, and mode fault detection can be enabled for master mode in cases where more than one SPI device might become a master at the same time. Also, some applications may utilize the receive data buffer hardware match feature to trigger specific actions, such as when command data can be sent through the SPI or to indicate the end of an SPI transmission.

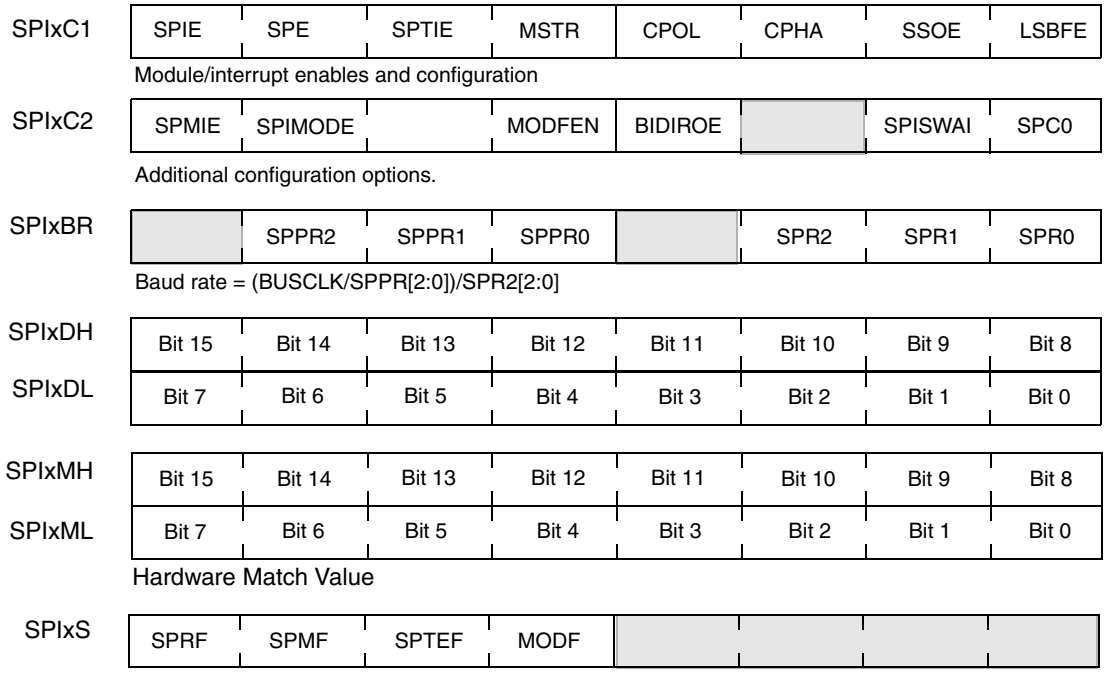


Figure 15-2. SPI Module Quick Start

15.1.2 Features

The SPI includes these distinctive features:

- Master mode or slave mode operation
- Full-duplex or single-wire bidirectional mode
- Programmable transmit bit rate
- Double-buffered transmit and receive data register
- Serial clock phase and polarity options
- Slave select output
- Mode fault error flag with CPU interrupt capability
- Control of SPI operation during wait mode
- Selectable MSB-first or LSB-first shifting
- Programmable 8- or 16-bit data transmission length
- Receive data buffer hardware match feature

15.1.3 Modes of Operation

The SPI functions in three modes, run, wait, and stop.

- Run Mode

  This is the basic mode of operation.
- Wait Mode

  SPI operation in wait mode is a configurable low power mode, controlled by the SPISWAI bit located in the SPIxC2 register. In wait mode, if the SPISWAI bit is clear, the SPI operates like in Run Mode. If the SPISWAI bit is set, the SPI goes into a power conservative state, with the SPI clock generation turned off. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU goes into Run Mode. If the SPI is configured as a slave, reception and transmission of a byte continues, so that the slave stays synchronized to the master.
- Stop Mode

  The SPI is inactive in stop3 mode for reduced power consumption. If the SPI is configured as a master, any transmission in progress stops, but is resumed after the CPU goes into Run Mode. If the SPI is configured as a slave, reception and transmission of a data continues, so that the slave stays synchronized to the master.

The SPI is completely disabled in all other stop modes. When the CPU wakes from these stop modes, all SPI register content will be reset.

This is a high level description only, detailed descriptions of operating modes are contained in section Section 15.4.9, “Low Power Mode Options.”

15.1.4 Block Diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

15.1.4.1 SPI System Block Diagram

Figure 15-3 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ($\overline{SS}$ pin). In this system, the master device has configured its $\overline{SS}$ pin as an optional slave select output.

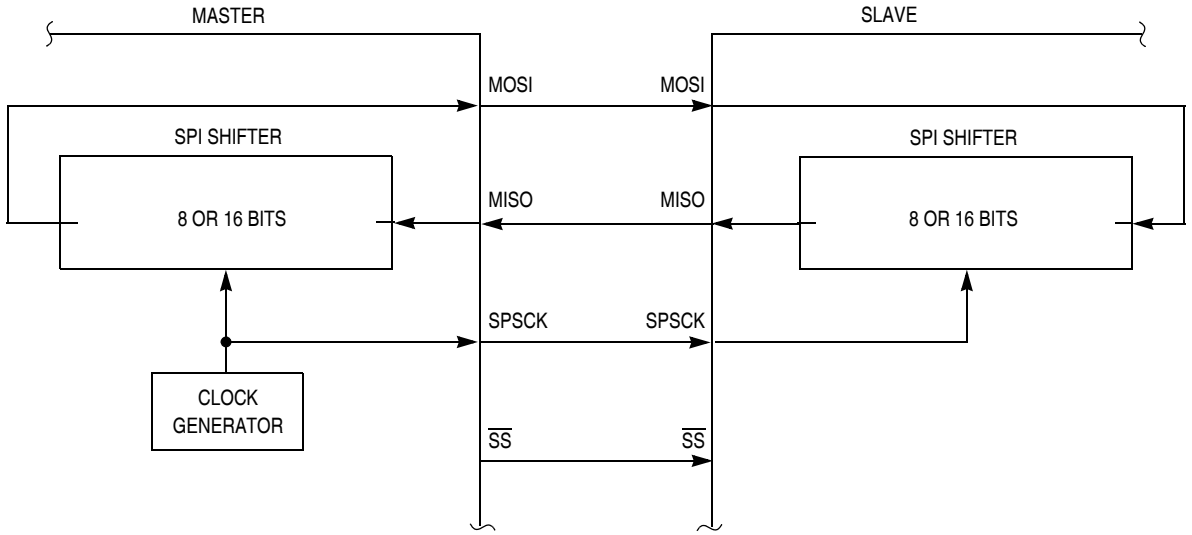


Figure 15-3. SPI System Connections

15.1.4.2 SPI Module Block Diagram

Figure 15-4 is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPIxDH:SPIxDL) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in 8 or 16 bits (as determined by SPIMODE bit) of data, the data is transferred into the double-buffered receiver where it can be read (read from SPIxDH:SPIxDL). Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

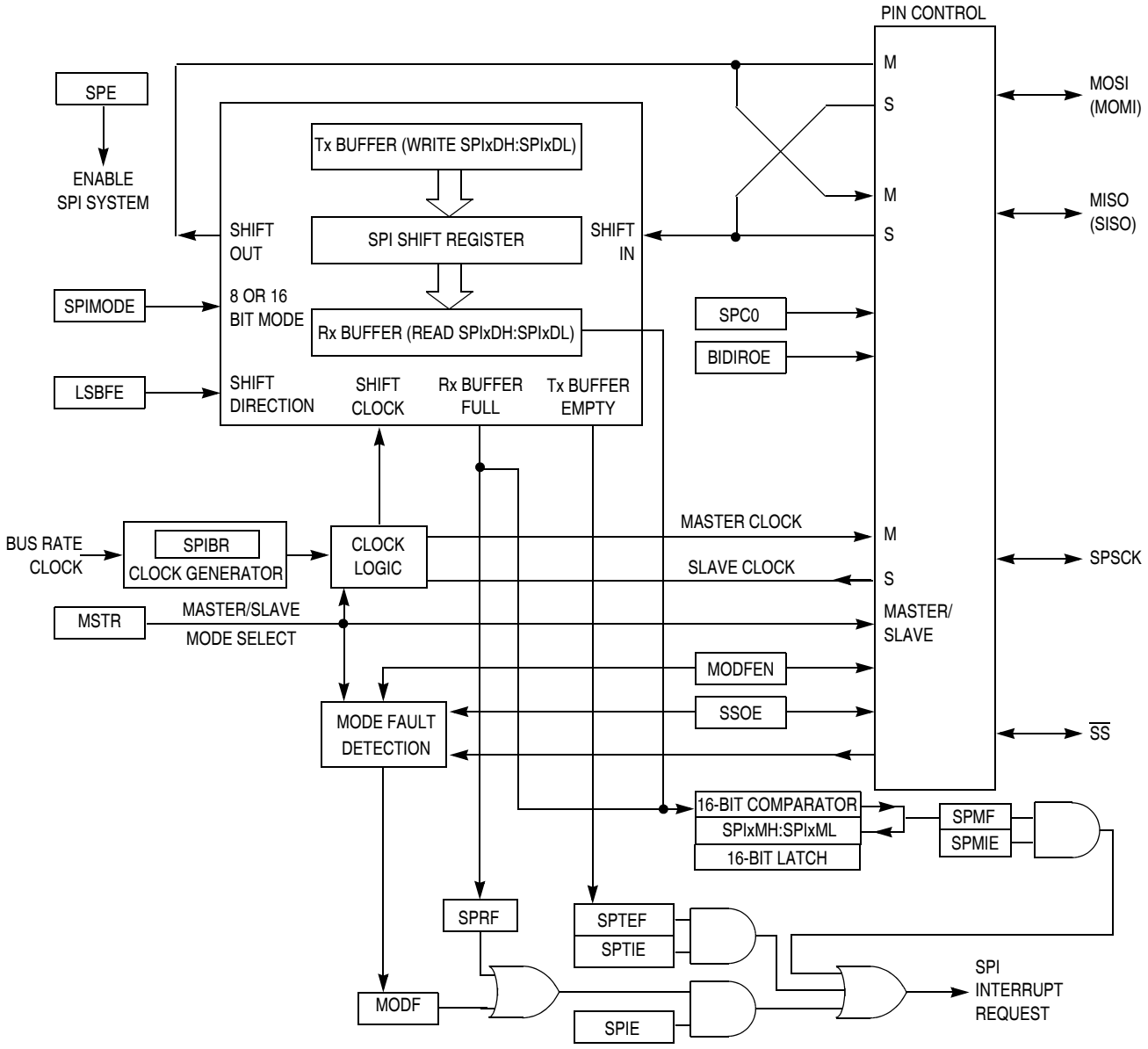


Figure 15-4. SPI Module Block Diagram

15.2 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled (SPE = 0), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

15.2.1 SPSCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

15.2.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data input. If SPC0 = 1 to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and slave mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

15.2.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data output. If SPC0 = 1 to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

15.2.4 $\overline{SS}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (MODFEN = 0), this pin is not used by the SPI and reverts to being a general-purpose port I/O pin. When the SPI is enabled as a master and MODFEN = 1, the slave select output enable bit determines whether this pin acts as the mode fault input (SSOE = 0) or as the slave select output (SSOE = 1).

15.3 Register Definition

The SPI has eight 8-bit registers to select SPI options, control baud rate, report SPI status, hold an SPI data match value, and for transmit/receive data.

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all SPI registers. This section refers to registers and control bits only by their names, and a Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

15.3.1 SPI Control Register 1 (SPIxC1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

	7	6	5	4	3	2	1	0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset	0	0	0	0	0	1	0	0

Figure 15-5. SPI Control Register 1 (SPIxC1)

Table 15-1. SPIxC1 Field Descriptions

Field	Description
7 SPIE	**SPI Interrupt Enable (for SPRF and MODF)** — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF inhibited (use polling) 1 When SPRF or MODF is 1, request a hardware interrupt
6 SPE	**SPI System Enable** — This bit enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, SPI is disabled and forced into idle state, and all status bits in the SPIxS register are reset. 0 SPI system inactive 1 SPI system enabled
5 SPTIE	**SPI Transmit Interrupt Enable** — This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). 0 Interrupts from SPTEF inhibited (use polling) 1 When SPTEF is 1, hardware interrupt requested
4 MSTR	**Master/Slave Mode Select —** This bit selects master or slave mode operation. 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	**Clock Polarity** — This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to Section 15.4.5, “SPI Clock Formats” for more details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)
2 CPHA	**Clock Phase** — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to Section 15.4.5, “SPI Clock Formats” for more details. 0 First edge on SPSCK occurs at the middle of the first cycle of a data transfer 1 First edge on SPSCK occurs at the start of the first cycle of a data transfer
1 SSOE	**Slave Select Output Enable** — This bit is used in combination with the mode fault enable (MODFEN) bit in SPIxC2 and the master/slave (MSTR) control bit to determine the function of the $\overline{SS}$ pin as shown in Table 15-2.
0 LSBFE	**LSB First (Shifter Direction) —** This bit does not affect the position of the MSB and LSB in the data register. Reads and writes of the data register always have the MSB in bit 7 (or bit 15 in 16-bit mode). 0 SPI serial data transfers start with most significant bit 1 SPI serial data transfers start with least significant bit

Table 15-2. $\overline{SS}$ Pin Function

MODFEN	SSOE	Master Mode	Slave Mode
0	0	General-purpose I/O (not SPI)	Slave select input
0	1	General-purpose I/O (not SPI)	Slave select input
1	0	$\overline{SS}$ input for mode fault	Slave select input
1	1	Automatic $\overline{SS}$ output	Slave select input

15.3.2 SPI Control Register 2 (SPIxC2)

This read/write register is used to control optional features of the SPI system. Bits 6 and 5 are not implemented and always read 0.

	7	6	5	4	3	2	1	0
R	SPMIE	SPIMODE	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 15-6. SPI Control Register 2 (SPIxC2)

Table 15-3. SPIxC2 Register Field Descriptions

Field	Description
7 SPMIE	**SPI Match Interrupt Enable** — This is the interrupt enable for the SPI receive data buffer hardware match (SPMF) function. 0 Interrupts from SPMF inhibited (use polling). 1 When SPMF = 1, requests a hardware interrupt.
6 SPIMODE	**SPI 8- or 16-bit Mode** — This bit allows the user to select either an 8-bit or 16-bit SPI data transmission length. In master mode, a change of this bit will abort a transmission in progress, force the SPI system into idle state, and reset all status bits in the SPIxS register. Refer to section Section 15.4.4, “Data Transmission Length,” for details. 0 8-bit SPI shift register, match register, and buffers. 1 16-bit SPI shift register, match register, and buffers.
4 MODFEN	**Master Mode-Fault Function Enable** — When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used (refer to Table 15-2 for details) 0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI 1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output
3 BIDIROE	**Bidirectional Mode Output Enable** — When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect. 0 Output driver disabled so SPI data I/O pin acts as an input 1 SPI I/O pin enabled as an output
1 SPISWAI	**SPI Stop in Wait Mode —** This bit is used for power conservation while in wait. 0 SPI clocks continue to operate in wait mode 1 SPI clocks stop when the MCU enters wait mode
0 SPC0	**SPI Pin Control 0** — This bit enables bidirectional pin configurations as shown in Table 15-4. 0 SPI uses separate pins for data input and data output. 1 SPI configured for single-wire bidirectional operation.

Table 15-4. Bidirectional Pin Configurations

Pin Mode	SPC0	BIDIROE	MISO	MOSI
Master Mode of Operation				
Normal	0	X	Master In	Master Out
Bidirectional	1	0	MISO not used by SPI	Master In
		1		Master I/O
Slave Mode of Operation				
Normal	0	X	Slave Out	Slave In
Bidirectional	1	0	Slave In	MOSI not used by SPI
		1	Slave I/O	

15.3.3 SPI Baud Rate Register (SPIxBR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

	7	6	5	4	3	2	1	0
R	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 15-7. SPI Baud Rate Register (SPIxBR)

Table 15-5. SPIxBR Register Field Descriptions

Field	Description
6:4 SPPR[2:0]	**SPI Baud Rate Prescale Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in Table 15-6. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see Figure 15-15). See Section 15.4.6, “SPI Baud Rate Generation,” for details.
2:0 SPR[2:0]	**SPI Baud Rate Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown in Table 15-7. The input to this divider comes from the SPI baud rate prescaler (see Figure 15-15). See Section 15.4.6, “SPI Baud Rate Generation,” for details.

Table 15-6. SPI Baud Rate Prescaler Divisor

SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

Table 15-7. SPI Baud Rate Divisor

SPR2:SPR1:SPR0	Rate Divisor
0:0:0	2
0:0:1	4
0:1:0	8
0:1:1	16
1:0:0	32
1:0:1	64
1:1:0	128
1:1:1	256

15.3.4 SPI Status Register (SPIxS)

This register has four read-only status bits. Bits 3 through 0 are not implemented and always read 0. Writes have no meaning or effect.

	7	6	5	4	3	2	1	0
R	SPRF	SPMF	SPTEF	MODF	0	0	0	0
W								
Reset	0	0	1	0	0	0	0	0

= Unimplemented or Reserved

Figure 15-8. SPI Status Register (SPIxS)

Table 15-8. SPIxS Register Field Descriptions

Field	Description
7 SPRF	**SPI Read Buffer Full Flag** — SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data register (SPIxDH:SPIxDL). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register. 0 No data available in the receive data buffer. 1 Data available in the receive data buffer.
6 SPMF	**SPI Match Flag** — SPMF is set after SPRF = 1 when the value in the receive data buffer matches the value in SPIMH:SPIML. To clear the flag, read SPMF when it is set, then write a 1 to it. 0 Value in the receive data buffer does not match the value in SPIxMH:SPIxML registers. 1 Value in the receive data buffer matches the value in SPIxMH:SPIxML registers.
5 SPTEF	**SPI Transmit Buffer Empty Flag** — This bit is set when the transmit data buffer is empty. It is cleared by reading SPIxS with SPTEF set, followed by writing a data value to the transmit buffer at SPIxDH:SPIxDL. SPIxS must be read with SPTEF = 1 before writing data to SPIxDH:SPIxDL or the SPIxDH:SPIxDL write will be ignored. SPTEF is automatically set when all data from the transmit buffer transfers into the transmit shift register. For an idle SPI, data written to SPIxDH:SPIxDL is transferred to the shifter almost immediately so SPTEF is set within two bus cycles allowing a second data to be queued into the transmit buffer. After completion of the transfer of the data in the shift register, the queued data from the transmit buffer will automatically move to the shifter and SPTEF will be set to indicate there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter. 0 SPI transmit buffer not empty 1 SPI transmit buffer empty
4 MODF	**Master Mode Fault Flag** — MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The $\overline{SS}$ pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPIxC1). 0 No mode fault error 1 Mode fault error detected

15.3.5 SPI Data Registers (SPIxDH:SPIxDL)

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

Figure 15-9. SPI Data Register High (SPIxDH)

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 15-10. SPI Data Register Low (SPIxDL)

The SPI data registers (SPIxDH:SPIxDL) are both the input and output register for SPI data. A write to these registers writes to the transmit data buffer, allowing data to be queued and transmitted.

When the SPI is configured as a master, data queued in the transmit data buffer is transmitted immediately after the previous transmission has completed.

The SPI transmit buffer empty flag (SPTEF) in the SPIxS register indicates when the transmit data buffer is ready to accept new data. SPIxS must be read when SPTEF is set before writing to the SPI data registers, or the write will be ignored.

Data may be read from SPIxDH:SPIxDL any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

In 8-bit mode, only SPIxDL is available. Reads of SPIxDH will return all 0s. Writes to SPIxDH will be ignored.

In 16-bit mode, reading either byte (SPIxDH or SPIxDL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIxDH or SPIxDL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the transmit data buffer.

15.3.6 SPI Match Registers (SPIxMH:SPIxML)

These read/write registers contain the hardware compare value, which sets the SPI match flag (SPMF) when the value received in the SPI receive data buffer equals the value in the SPIxMH:SPIxML registers.

In 8-bit mode, only SPIxML is available. Reads of SPIxMH will return all 0s. Writes to SPIxMH will be ignored.

In 16-bit mode, reading either byte (SPIxMH or SPIxML) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIxMH or SPIxML) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent value into the SPI match registers.

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

Figure 15-11. SPI Match Register High (SPIxMH)

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 15-12. SPI Match Register Low (SPIxML)

15.4 Functional Description

15.4.1 General

The SPI system is enabled by setting the SPI enable (SPE) bit in SPI Control Register 1. While the SPE bit is set, the four associated SPI port pins are dedicated to the SPI function as:

- Slave select ($\overline{SS}$)
- Serial clock (SPSCK)
- Master out/slave in (MOSI)
- Master in/slave out (MISO)

An SPI transfer is initiated in the master SPI device by reading the SPI status register (SPIxS) when SPTEF = 1 and then writing data to the transmit data buffer (write to SPIxDH:SPIxDL). When a transfer is complete, received data is moved into the receive data buffer. The SPIxDH:SPIxDL registers act as the SPI receive data buffer for reads and as the SPI transmit data buffer for writes.

The clock phase control bit (CPHA) and a clock polarity control bit (CPOL) in the SPI Control Register 1 (SPIxC1) select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects a non-inverted or inverted clock. The CPHA bit is used to accommodate two fundamentally different protocols by sampling data on odd numbered SPSCK edges or on even numbered SPSCK edges.

The SPI can be configured to operate as a master or as a slave. When the MSTR bit in SPI control register 1 is set, master mode is selected, when the MSTR bit is clear, slave mode is selected.

15.4.2 Master Mode

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by reading the SPIxS register while SPTEF = 1 and writing to the master SPI data registers. If the shift register is empty, the byte immediately transfers to the shift register. The data begins shifting out on the MOSI pin under the control of the serial clock.

- SPSCK

The SPR2, SPR1, and SPR0 baud rate selection bits in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI Baud Rate register control the baud rate generator and determine the speed of the transmission. The SPSCK pin is the SPI clock output. Through the SPSCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.

- MOSI, MISO pin

In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.

- $\overline{SS}$ pin

If MODFEN and SSOE bit are set, the $\overline{SS}$ pin is configured as slave select output. The $\overline{SS}$ output becomes low during each transmission and is high when the SPI is in idle state.

If MODFEN is set and SSOE is cleared, the $\overline{SS}$ pin is configured as input for detecting mode fault error. If the $\overline{SS}$ input becomes low this indicates a mode fault error where another master tries to drive the MOSI

and SPSCK lines. In this case, the SPI immediately switches to slave mode, by clearing the MSTR bit and also disables the slave output buffer MISO (or SISO in bidirectional mode). So the result is that all outputs are disabled and SPSCK, MOSI and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state.

This mode fault error also sets the mode fault (MODF) flag in the SPI Status Register (SPIxS). If the SPI interrupt enable bit (SPIE) is set when the MODF flag gets set, then an SPI interrupt sequence is also requested.

When a write to the SPI Data Register in the master occurs, there is a half SPSCK-cycle delay. After the delay, SPSCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI Control Register 1 (see Section 15.4.5, "SPI Clock Formats.")

NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, BIDIROE with SPC0 set, SPIMODE, SPPR2-SPPR0 and SPR2-SPR0 in master mode will abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master has to ensure that the remote slave is set back to idle state.

15.4.3 Slave Mode

The SPI operates in slave mode when the MSTR bit in SPI Control Register1 is clear.

- SPSCK

In slave mode, SPSCK is the SPI clock input from the master.

- MISO, MOSI pin

In slave mode, the function of the serial data output pin (MISO) and serial data input pin (MOSI) is determined by the SPC0 bit and BIDIROE bit in SPI Control Register 2.

- $\overline{SS}$ pin

The $\overline{SS}$ pin is the slave select input. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be low. $\overline{SS}$ must remain low until the transmission is complete. If $\overline{SS}$ goes high, the SPI is forced into idle state.

The $\overline{SS}$ input also controls the serial data output pin, if $\overline{SS}$ is high (not selected), the serial data output pin is high impedance, and, if $\overline{SS}$ is low the first bit in the SPI Data Register is driven out of the serial data output pin. Also, if the slave is not selected ($\overline{SS}$ is high), then the SPSCK input is ignored and no internal shifting of the SPI shift register takes place.

Although the SPI is capable of duplex operation, some SPI peripherals are capable of only receiving SPI data in a slave mode. For these simpler devices, there is no serial data out pin.

NOTE

When peripherals with duplex capability are used, take care not to simultaneously enable two receivers whose serial outputs drive the same system slave's serial data output line.

As long as no more than one slave device drives the system slave's serial data output line, it is possible for several slaves to receive the same transmission from a master, although the master would not receive return information from all of the receiving slaves.

If the CPHA bit in SPI Control Register 1 is clear, odd numbered edges on the SPSCK input cause the data at the serial data input pin to be latched. Even numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

If the CPHA bit is set, even numbered edges on the SPSCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

When CPHA is set, the first edge is used to get the first data bit onto the serial data output pin. When CPHA is clear and the $\overline{SS}$ input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the eighth (SPIMODE = 0) or sixteenth (SPIMODE = 1) shift, the transfer is considered complete and the received data is transferred into the SPI data registers. To indicate transfer is complete, the SPRF flag in the SPI Status Register is set.

NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0 and BIDIROE with SPC0 set and SPIMODE in slave mode will corrupt a transmission in progress and has to be avoided.

15.4.4 Data Transmission Length

The SPI can support data lengths of 8 or 16 bits. The length can be configured with the SPIMODE bit in the SPIxC2 register.

In 8-bit mode (SPIMODE = 0), the SPI Data Register is comprised of one byte: SPIxDL. The SPI Match Register is also comprised of only one byte: SPIxML. Reads of SPIxDH and SPIxMH will return zero. Writes to SPIxDH and SPIxMH will be ignored.

In 16-bit mode (SPIMODE = 1), the SPI Data Register is comprised of two bytes: SPIxDH and SPIxDL. Reading either byte (SPIxDH or SPIxDL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIxDH or SPIxDL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the transmit data buffer.

In 16-bit mode, the SPI Match Register is also comprised of two bytes: SPIxMH and SPIxML. Reading either byte (SPIxMH or SPIxML) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIxMH or SPIxML) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the transmit data buffer.

Any switching between 8- and 16-bit data transmission length (controlled by SPIMODE bit) in master mode will abort a transmission in progress, force the SPI system into idle state, and reset all status bits in the SPIxS register. To initiate a transfer after writing to SPIMODE, the SPIxS register must be read with SPTEF = 1, and data must be written to SPIxDH:SPIxDL in 16-bit mode (SPIMODE = 1) or SPIxDL in 8-bit mode (SPIMODE = 0).

In slave mode, user software should write to SPIMODE only once to prevent corrupting a transmission in progress.

NOTE

Data can be lost if the data length is not the same for both master and slave devices.

15.4.5 SPI Clock Formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a clock polarity (CPOL) bit and a clock phase (CPHA) control bit to select one of four clock formats for data transfers. CPOL selectively inserts an inverter in series with the clock. CPHA chooses between two different clock phase relationships between the clock and data.

Figure 15-13 shows the clock formats when SPIMODE = 0 (8-bit mode) and CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the sixteenth SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

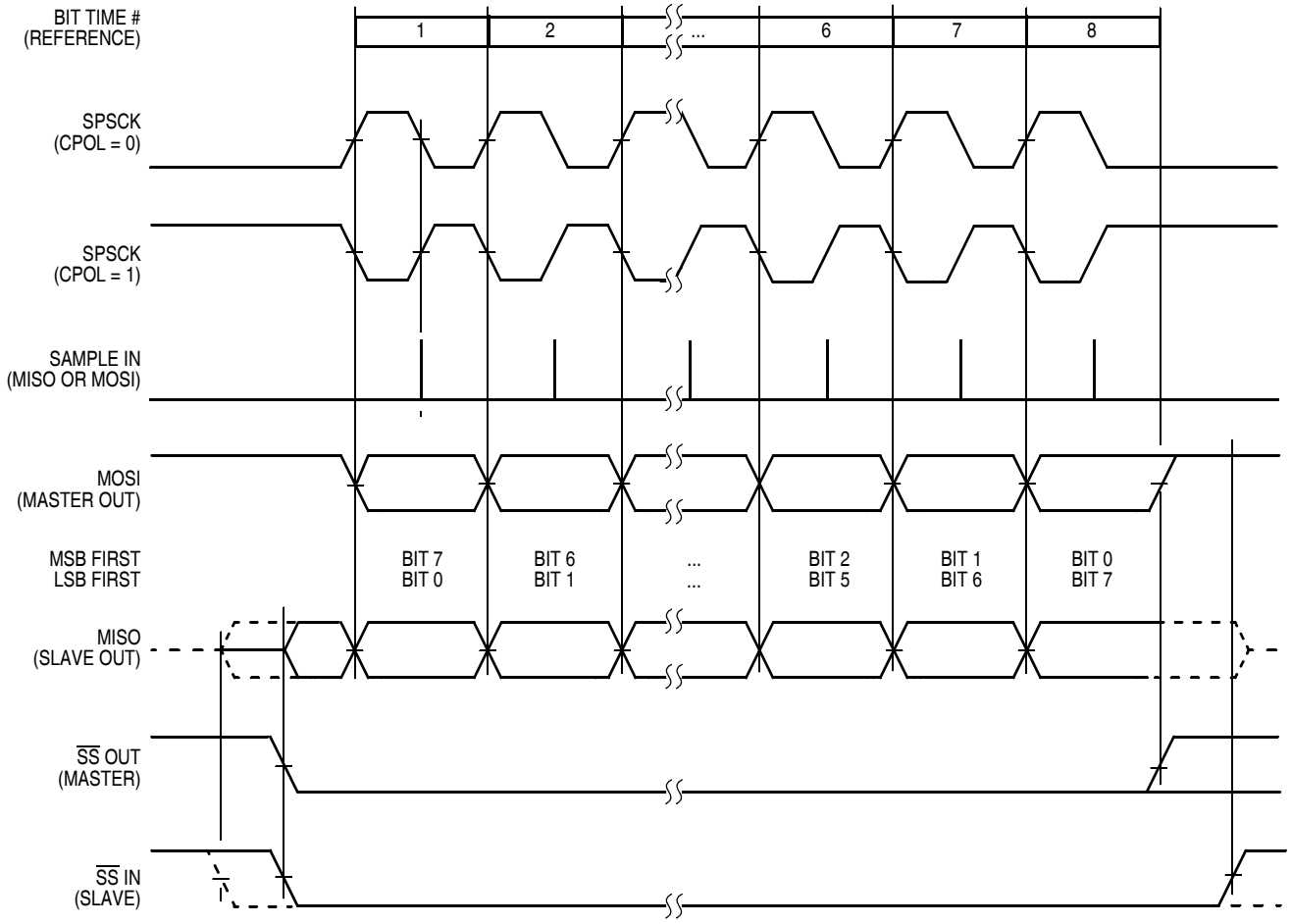


Figure 15-13. SPI Clock Formats (CPHA = 1)

When CPHA = 1, the slave begins to drive its MISO output when $\overline{SS}$ goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 1, the slave's $\overline{SS}$ input is not required to go to its inactive high level between transfers.

Figure 15-14 shows the clock formats when SPIMODE = 0 and CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ($\overline{SS}$ IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low at the start of the first bit time of the transfer and goes back high one-half

SPSCK cycle after the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

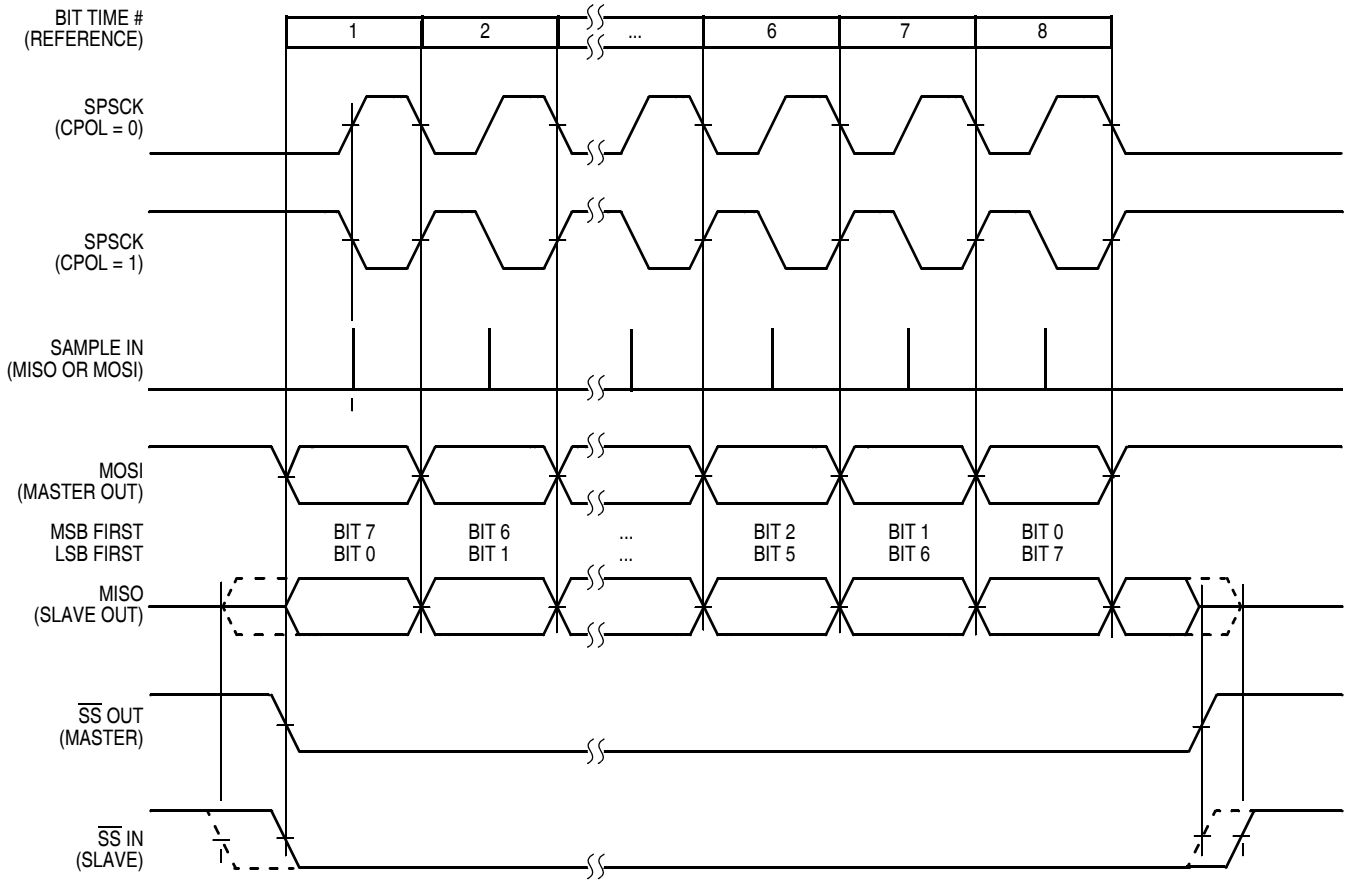


Figure 15-14. SPI Clock Formats (CPHA = 0)

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when $\overline{SS}$ goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's $\overline{SS}$ input must go to its inactive high level between transfers.

15.4.6 SPI Baud Rate Generation

As shown in Figure 15-15, the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, or 256 to get the internal SPI master mode bit-rate clock.

The baud rate generator is activated only when the SPI is in the master mode and a serial transfer is taking place. In the other cases, the divider is disabled to decrease I_{DD} current.

The baud rate divisor equation is as follows:

$$BaudRateDivisor = (SPPR + 1)\xi 2^{(SPR + 1)}$$

The baud rate can be calculated with the following equation:

$$Baud\ Rate = \frac{BusClock}{BaudRateDivisor}$$

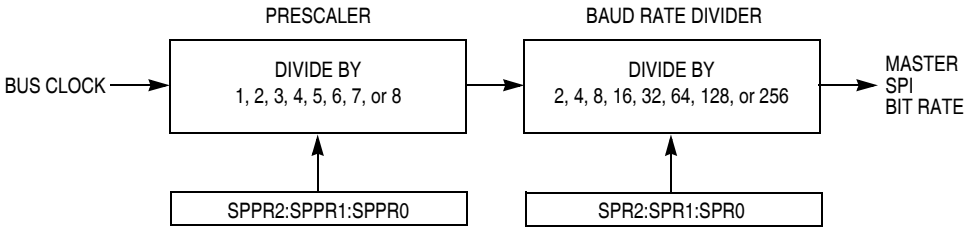


Figure 15-15. SPI Baud Rate Generation

15.4.7 Special Features

15.4.7.1 $\overline{SS}$ Output

The $\overline{SS}$ output feature automatically drives the $\overline{SS}$ pin low during transmission to select external devices and drives it high during idle to deselect external devices. When $\overline{SS}$ output is selected, the $\overline{SS}$ output pin is connected to the $\overline{SS}$ input pin of the external device.

The $\overline{SS}$ output is available only in master mode during normal SPI operation by asserting the SSOE and MODFEN bits as shown in Table 15-2.

The mode fault feature is disabled while $\overline{SS}$ output is enabled.

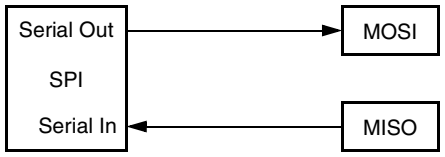
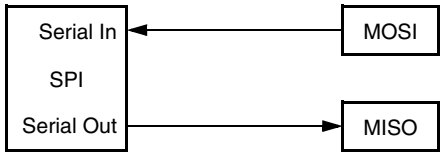
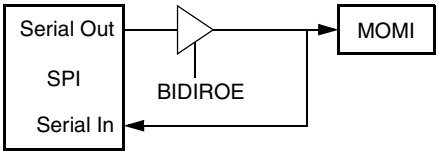
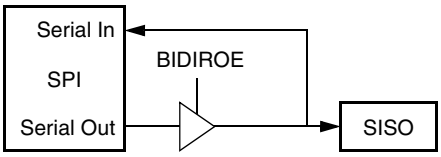
NOTE

Care must be taken when using the $\overline{SS}$ output feature in a multi-master system since the mode fault feature is not available for detecting system errors between masters.

15.4.7.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI Control Register 2 (see Table 15-9.) In this mode, the SPI uses only one serial data pin for the interface with external device(s). The MSTR bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes serial data I/O (SISO) pin for the slave mode. The MISO pin in master mode and MOSI pin in slave mode are not used by the SPI.

Table 15-9. Normal Mode and Bidirectional Mode

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
Normal Mode SPC0 = 0		
Bidirectional Mode SPC0 = 1		

The direction of each serial I/O pin depends on the BIDIROE bit. If the pin is configured as an output, serial data from the shift register is driven out on the pin. The same pin is also the serial input to the shift register.

The SPSCK is output for the master mode and input for the slave mode.

The $\overline{SS}$ is the input or output for the master mode, and it is always the input for the slave mode.

The bidirectional mode does not affect SPSCK and $\overline{SS}$ functions.

NOTE

In bidirectional master mode, with mode fault enabled, both data pins MISO and MOSI can be occupied by the SPI, though MOSI is normally used for transmissions in bidirectional mode and MISO is not used by the SPI. If a mode fault occurs, the SPI is automatically switched to slave mode, in this case MISO becomes occupied by the SPI and MOSI is not used. This has to be considered, if the MISO pin is used for another purpose.

15.4.8 Error Conditions

The SPI has one error condition:

- Mode fault error

15.4.8.1 Mode Fault Error

If the $\overline{SS}$ input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SPSCK lines simultaneously. This condition is not permitted in normal operation, and the MODF bit in the SPI status register is set automatically provided the MODFEN bit is set.

In the special case where the SPI is in master mode and MODFEN bit is cleared, the $\overline{SS}$ pin is not used by the SPI. In this special case, the mode fault error function is inhibited and MODF remains cleared. In case the SPI system is configured as a slave, the $\overline{SS}$ pin is a dedicated input pin. Mode fault error doesn't occur in slave mode.

If a mode fault error occurs the SPI is switched to slave mode, with the exception that the slave output buffer is disabled. So SPSCK, MISO and MOSI pins are forced to be high impedance inputs to avoid any possibility of conflict with another output driver. A transmission in progress is aborted and the SPI is forced into idle state.

If the mode fault error occurs in the bidirectional mode for a SPI system configured in master mode, output enable of the MOMI (MOSI in bidirectional mode) is cleared if it was set. No mode fault error occurs in the bidirectional mode for the SPI system configured in slave mode.

The mode fault flag is cleared automatically by a read of the SPI Status Register (with MODF set) followed by a write to SPI Control Register 1. If the mode fault flag is cleared, the SPI becomes a normal master or slave again.

15.4.9 Low Power Mode Options

15.4.9.1 SPI in Run Mode

In run mode with the SPI system enable (SPE) bit in the SPI control register clear, the SPI system is in a low-power, disabled state. SPI registers can still be accessed, but clocks to the core of this module are disabled.

15.4.9.2 SPI in Wait Mode

SPI operation in wait mode depends upon the state of the SPISWAI bit in SPI Control Register 2.

- If SPISWAI is clear, the SPI operates normally when the CPU is in wait mode
- If SPISWAI is set, SPI clock generation ceases and the SPI module enters a power conservation state when the CPU is in wait mode.
  - — If SPISWAI is set and the SPI is configured for master, any transmission and reception in progress stops at wait mode entry. The transmission and reception resumes when the SPI exits wait mode.
  - – If SPISWAI is set and the SPI is configured as a slave, any transmission and reception in progress continues if the SPSCK continues to be driven from the master. This keeps the slave synchronized to the master and the SPSCK.

    If the master transmits data while the slave is in wait mode, the slave will continue to send out data consistent with the operation mode at the start of wait mode (i.e., if the slave is currently sending its SPIxDH:SPIxDL to the master, it will continue to send the same byte. Otherwise, if the slave is currently sending the last data received byte from the master, it will continue to send each previously receive data from the master byte).

NOTE

Care must be taken when expecting data from a master while the slave is in wait or stop3 mode. Even though the shift register will continue to operate, the rest of the SPI is shut down (i.e. a SPRF interrupt will not be generated until exiting stop or wait mode). Also, the data from the shift register will not be copied into the SPIxDH:SPIxDL registers until after the slave SPI has exited wait or stop mode. A SPRF flag and SPIxDH:SPIxDL copy is only generated if wait mode is entered or exited during a tranmission. If the slave enters wait mode in idle mode and exits wait mode in idle mode, neither a SPRF nor a SPIxDH:SPIxDL copy will occur.

15.4.9.3 SPI in Stop Mode

Stop3 mode is dependent on the SPI system. Upon entry to stop3 mode, the SPI module clock is disabled (held high or low). If the SPI is in master mode and exchanging data when the CPU enters stop mode, the transmission is frozen until the CPU exits stop mode. After stop, data to and from the external SPI is exchanged correctly. In slave mode, the SPI will stay synchronized with the master.

The stop mode is not dependent on the SPISWAI bit.

In all other stop modes, the SPI module is completely disabled. After stop, all registers are reset to their default values, and the SPI module must be re-initialized.

15.4.9.4 Reset

The reset values of registers and signals are described in Section 15.3, “Register Definition.” which details the registers and their bit-fields.

- If a data transmission occurs in slave mode after reset without a write to SPIxDH:SPIxDL, it will transmit garbage, or the data last received from the master before the reset.
- Reading from the SPIxDH:SPIxDL after reset will always read zeros.

15.4.9.5 Interrupts

The SPI only originates interrupt requests when the SPI is enabled (SPE bit in SPIxC1 set). The following is a description of how the SPI makes a request and how the MCU should acknowledge that request. The interrupt vector offset and interrupt priority are chip dependent.

15.4.10 SPI Interrupts

There are four flag bits, three interrupt mask bits, and one interrupt vector associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). The SPI match interrupt enable mask bit (SPIMIE) enables interrupts from the SPI match flag (SPMF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) should check

the flag bits to determine what event caused the interrupt. The service routine should also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

15.4.10.1 MODF

MODF occurs when the master detects an error on the $\overline{SS}$ pin. The master SPI must be configured for the MODF feature (see Table 15-2). Once MODF is set, the current transfer is aborted and the following bit is changed:

- MSTR=0, The master bit in SPIxC1 resets.

The MODF interrupt is reflected in the status register MODF flag. Clearing the flag will also clear the interrupt. This interrupt will stay active while the MODF flag is set. MODF has an automatic clearing process which is described in Section 15.3.4, “SPI Status Register (SPIxS).”

15.4.10.2 SPRF

SPRF occurs when new data has been received and copied to the SPI receive data buffer. In 8-bit mode, SPRF is set only after all 8 bits have been shifted out of the shift register and into SPIxDL. In 16-bit mode, SPRF is set only after all 16 bits have been shifted out of the shift register and into SPIxDH:SPIxDL.

Once SPRF is set, it does not clear until it is serviced. SPRF has an automatic clearing process which is described in Section 15.3.4, “SPI Status Register (SPIxS).” In the event that the SPRF is not serviced before the end of the next transfer (i.e. SPRF remains active throughout another transfer), the latter transfers will be ignored and no new data will be copied into the SPIxDH:SPIxDL.

15.4.10.3 SPTEF

SPTEF occurs when the SPI transmit buffer is ready to accept new data. In 8-bit mode, SPTEF is set only after all 8 bits have been moved from SPIxDL into the shifter. In 16-bit mode, SPTEF is set only after all 16 bits have been moved from SPIxDH:SPIxDL into the shifter.

Once SPTEF is set, it does not clear until it is serviced. SPTEF has an automatic clearing process which is described in Section 15.3.4, “SPI Status Register (SPIxS).

15.4.10.4 SPMF

SPMF occurs when the data in the receive data buffer is equal to the data in the SPI match register. In 8-bit mode, SPMF is set only after bits 8–0 in the receive data buffer are determined to be equivalent to the value in SPIxML. In 16-bit mode, SPMF is set after bits 15–0 in the receive data buffer are determined to be equivalent to the value in SPIxMH:SPIxML.

15.5 Initialization/Application Information

15.5.1 SPI Module Initialization Example

15.5.1.1 Initialization Sequence

Before the SPI module can be used for communication, an initialization procedure must be carried out, as follows:

1. Update control register 1 (SPIxC1) to enable the SPI and to control interrupt enables. This register also sets the SPI as master or slave, determines clock phase and polarity, and configures the main SPI options.
2. Update control register 2 (SPIxC2) to enable additional SPI functions such as the SPI match interrupt feature, the master mode-fault function, and bidirectional mode output. 8- or 16-bit mode select and other optional features are controlled here as well.
3. Update the baud rate register (SPIxBR) to set the prescaler and bit rate divisor for an SPI master.
4. Update the hardware match register (SPIxMH:SPIxML) with the value to be compared to the receive data register for triggering an interrupt if hardware match interrupts are enabled.
5. In the master, read SPIxS while SPTEF = 1, and then write to the transmit data register (SPIxDH:SPIxDL) to begin transfer.

15.5.1.2 Pseudo—Code Example

In this example, the SPI module will be set up for master mode with only hardware match interrupts enabled. The SPI will run in 16-bit mode at a maximum baud rate of bus clock divided by 2. Clock phase and polarity will be set for an active-high SPI clock where the first edge on SPSCK occurs at the start of the first cycle of a data transfer.

SPIxC1=0x54(%01010100)

Bit 7	SPIE	= 0	Disables receive and mode fault interrupts
Bit 6	SPE	= 1	Enables the SPI system
Bit 5	SPTIE	= 0	Disables SPI transmit interrupts
Bit 4	MSTR	= 1	Sets the SPI module as a master SPI device
Bit 3	CPOL	= 0	Configures SPI clock as active-high
Bit 2	CPHA	= 1	First edge on SPSCK at start of first data transfer cycle
Bit 1	SSOE	= 0	Determines $\overline{SS}$ pin function when mode fault enabled
Bit 0	LSBFE	= 0	SPI serial data transfers start with most significant bit

SPIxC2 = 0xC0(%11000000)

Bit 7	SPMIE	= 1	SPI hardware match interrupt enabled
Bit 6	SPIMODE	= 1	Configures SPI for 16-bit mode
Bit 5		= 0	Unimplemented
Bit 4	MODFEN	= 0	Disables mode fault function
Bit 3	BIDIROE	= 0	SPI data I/O pin acts as input
Bit 2		= 0	Unimplemented
Bit 1	SPISWAI	= 0	SPI clocks operate in wait mode
Bit 0	SPC0	= 0	uses separate pins for data input and output

SPIxBR = 0x00(%00000000)

Bit 7	= 0	Unimplemented
Bit 6:4	= 000	Sets prescale divisor to 1
Bit 3	= 0	Unimplemented
Bit 2:0	= 000	Sets baud rate divisor to 2

SPIxS = 0x00(%00000000)

Bit 7	SPRF	= 0	Flag is set when receive data buffer is full
Bit 6	SPMF	= 0	Flag is set when SPIMH/L = receive data buffer
Bit 5	SPTEF	= 0	Flag is set when transmit data buffer is empty
Bit 4	MODF	= 0	Mode fault flag for master mode
Bit 3:0		= 0	Unimplemented

SPIxMH = 0xXX

In 16-bit mode, this register holds bits 8–15 of the hardware match buffer. In 8-bit mode, writes to this register will be ignored.

SPIxML = 0xXX

Holds bits 0–7 of the hardware match buffer.

SPIxDH = 0xxx

In 16-bit mode, this register holds bits 8–15 of the data to be transmitted by the transmit buffer and received by the receive buffer.

SPIxDL = 0xxx

Holds bits 0–7 of the data to be transmitted by the transmit buffer and received by the receive buffer.

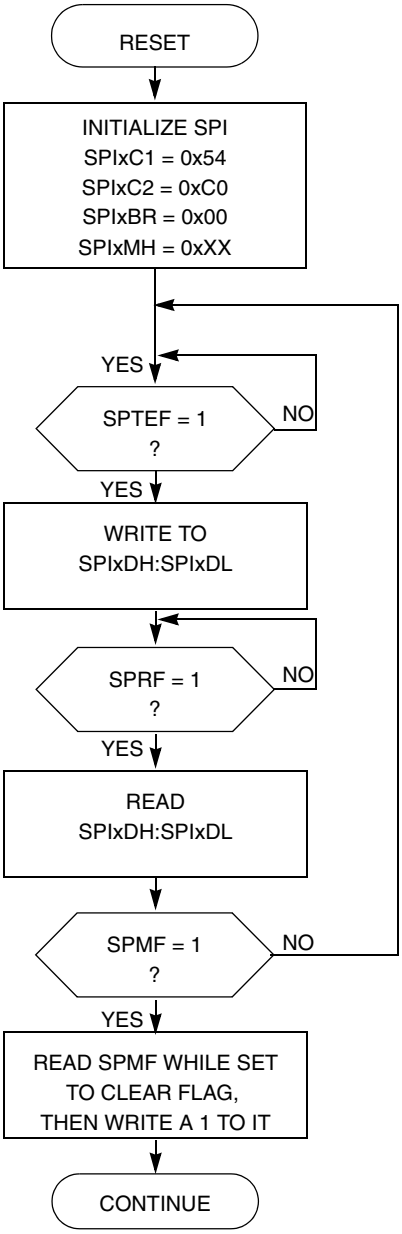
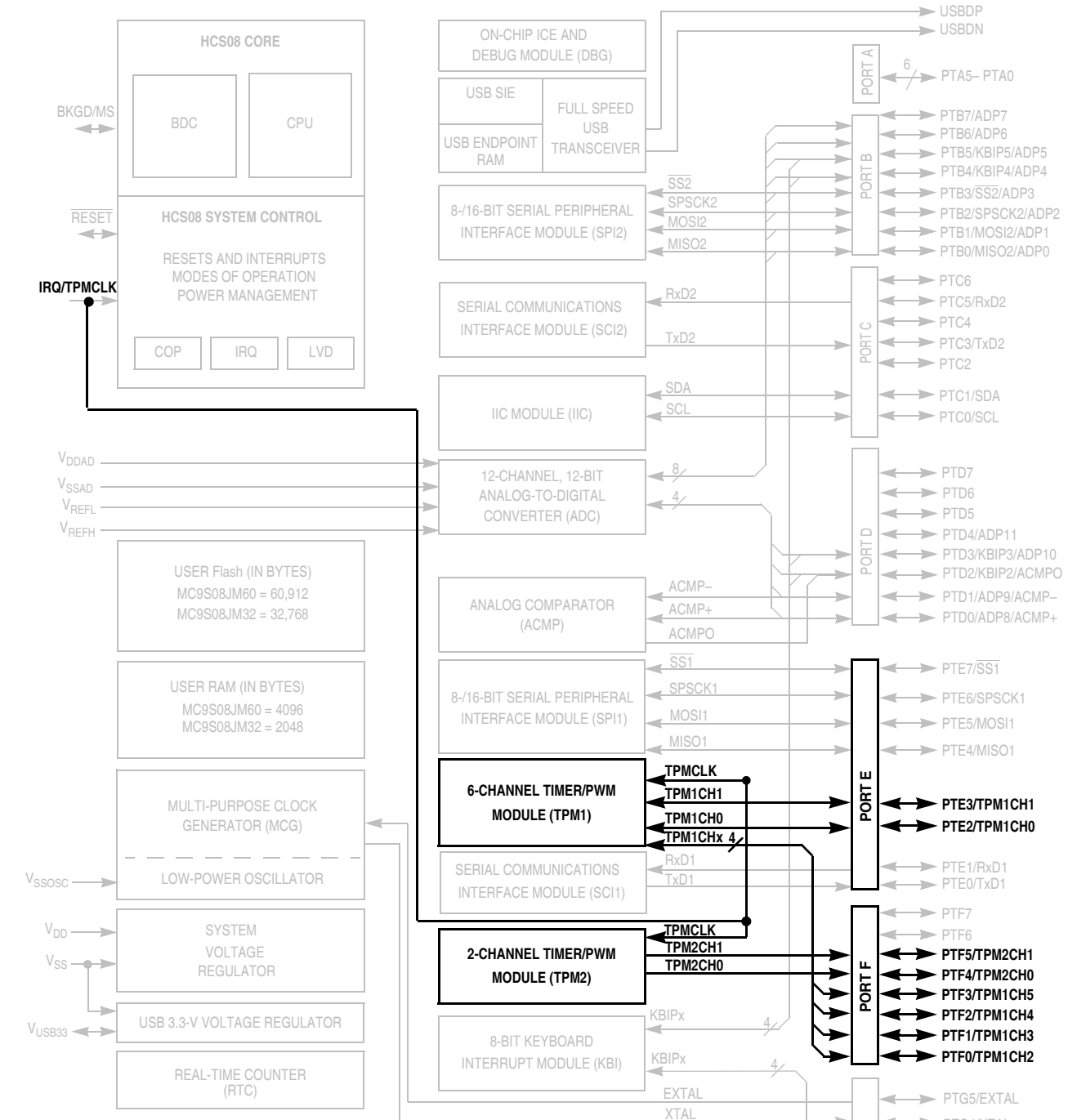


Figure 15-16. Initialization Flowchart Example for SPI Master Device in 16-bit Mode

Chapter 16
Timer/Pulse-Width Modulator (S08TPMV3)

16.1 Introduction

The MC9S08JM60 series includes two independent timer/PWM (TPM) modules which support traditional input capture, output compare, or buffered edge-aligned pulse-width modulation (PWM) on each channel. A control bit in each TPM configures all channels in that timer to operate as center-aligned PWM functions. In each of these two TPMs, timing functions are based on a separate 16-bit counter with prescaler and modulo features to control frequency and range (period between overflows) of the time reference.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device if IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 16-1. MC9S08JM60 Series Block Diagram Highlighting the TPM Blocks and Pins

16.1.1 Features

The TPM includes these distinctive features:

- One to eight channels:
  - Each channel may be input capture, output compare, or edge-aligned PWM
  - Rising-Edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
  - Selectable polarity on PWM outputs
- Module may be configured for buffered, center-aligned pulse-width-modulation (CPWM) on all channels
- Timer clock source selectable as prescaled bus clock, fixed system clock, or an external clock pin
  - Prescale taps for divide-by 1, 2, 4, 8, 16, 32, 64, or 128
  - Fixed system clock source are synchronized to the bus clock by an on-chip synchronization circuit
  - External clock pin may be shared with any timer channel pin or a separated input pin
- 16-bit free-running or modulo up/down count operation
- Timer system enable
- One interrupt per channel plus terminal count interrupt

16.1.2 Modes of Operation

In general, TPM channels may be independently configured to operate in input capture, output compare, or edge-aligned PWM modes. A control bit allows the whole TPM (all channels) to switch to center-aligned PWM mode. When center-aligned PWM mode is selected, input capture, output compare, and edge-aligned PWM functions are not available on any channels of this TPM module.

When the microcontroller is in active BDM background or BDM foreground mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all system clocks, including the main oscillator, are stopped; therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally. Provided the TPM does not need to produce a real time reference or provide the interrupt source(s) needed to wake the MCU from wait mode, the user can save power by disabling TPM functions before entering wait mode.

- Input capture mode

  When a selected edge event occurs on the associated MCU pin, the current value of the 16-bit timer counter is captured into the channel value register and an interrupt flag bit is set. Rising edges, falling edges, any edge, or no edge (disable channel) may be selected as the active edge which triggers the input capture.
- Output compare mode

  When the value in the timer counter register matches the channel value register, an interrupt flag bit is set, and a selected output action is forced on the associated MCU pin. The output compare action may be selected to force the pin to zero, force the pin to one, toggle the pin, or ignore the pin (used for software timing functions).

- Edge-aligned PWM mode

  The value of a 16-bit modulo register plus 1 sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. The user may also choose the polarity of the PWM output signal. Interrupts are available at the end of the period and at the duty-cycle transition point. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within a TPM.
- Center-aligned PWM mode

  Twice the value of a 16-bit modulo register sets the period of the PWM output, and the channel-value register sets the half-duty-cycle duration. The timer counter counts up until it reaches the modulo value and then counts down until it reaches zero. As the count matches the channel value register while counting down, the PWM output becomes active. When the count matches the channel value register while counting up, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. This type of PWM is required for types of motors used in small appliances.

This is a high-level description only. Detailed descriptions of operating modes are in later sections.

16.1.3 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMxCHn (timer channel n) where n is the channel number (1-8). The TPM shares its I/O pins with general purpose I/O port pins (refer to I/O pin descriptions in full-chip specification for the specific chip implementation).

Figure 16-2 shows the TPM structure. The central component of the TPM is the 16-bit counter that can operate as a free-running counter or a modulo up/down counter. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMxMODH:TPMxMODL, control the modulo value of the counter (the values 0x0000 or 0xFFFF effectively make the counter free running). Software can read the counter value at any time without affecting the counting sequence. Any write to either half of the TPMxCNT counter resets the counter, regardless of the data value written.

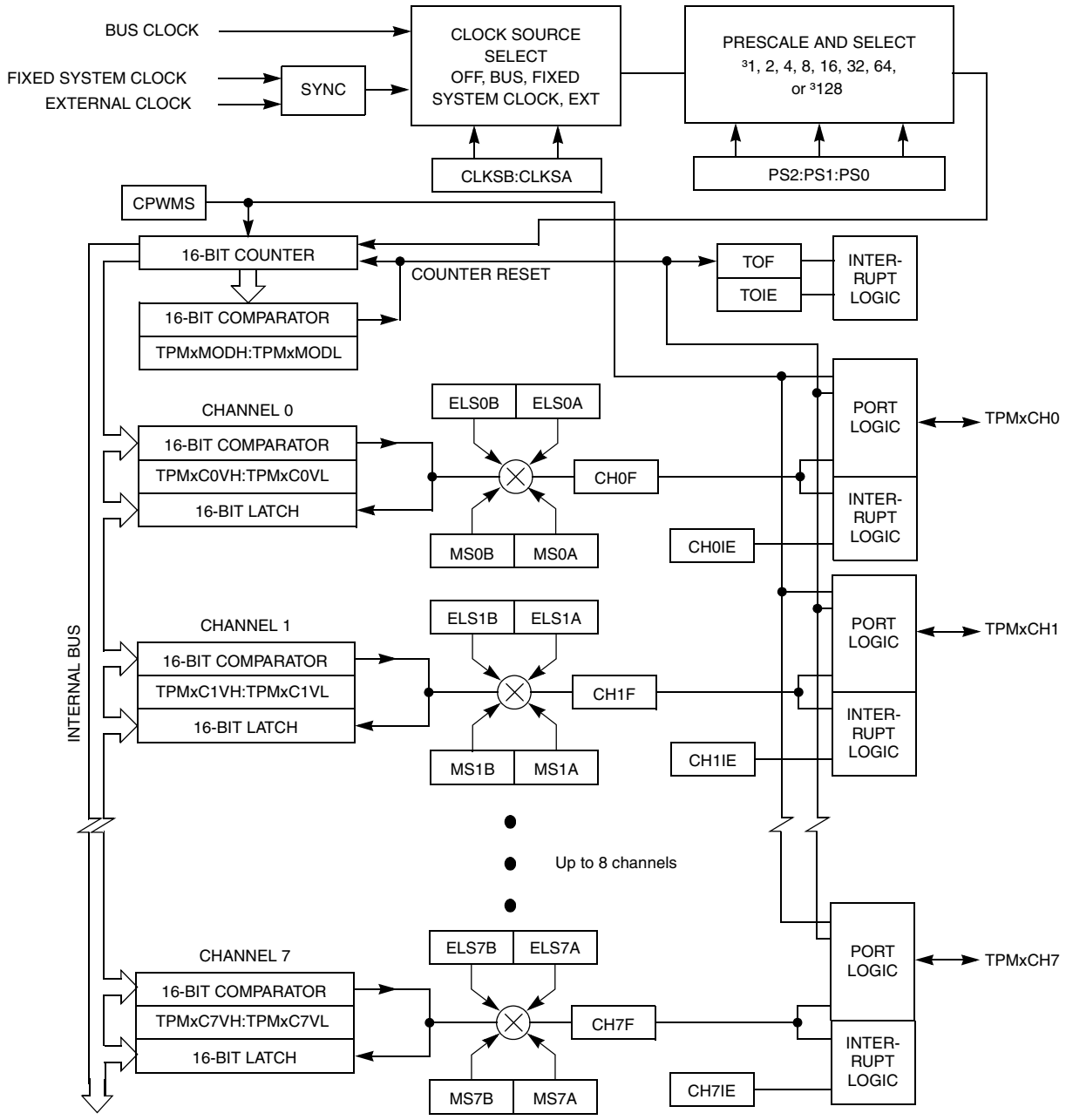


Figure 16-2. TPM Block Diagram

The TPM channels are programmable independently as input capture, output compare, or edge-aligned PWM channels. Alternately, the TPM can be configured to produce CPWM outputs on all channels. When the TPM is configured for CPWMs, the counter operates as an up/down counter; input capture, output compare, and EPWM functions are not practical.

If a channel is configured as input capture, an internal pullup device may be enabled for that channel. The details of how a module interacts with pin controls depends upon the chip implementation because the I/O pins and associated general purpose I/O controls are not part of the module. Refer to the discussion of the I/O port logic in a full-chip specification.

Because center-aligned PWMs are usually used to drive 3-phase AC-induction motors and brushless DC motors, they are typically used in sets of three or six channels.

16.2 Signal Description

Table 16-1 shows the user-accessible signals for the TPM. The number of channels may be varied from one to eight. When an external clock is included, it can be shared with the same pin as any TPM channel; however, it could be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

Table 16-1. Signal Properties

Name	Function
EXTCLK[1]	External clock source which may be selected to drive the TPM counter.
TPMxCHn[2]	I/O pin associated with TPM channel n

1 When preset, this signal can share any channel pin; however depending upon full-chip implementation, this signal could be connected to a separate external pin.

2 n=channel number (1 to 8)

Refer to documentation for the full-chip for details about reset states, port connections, and whether there is any pullup device on these pins.

TPM channel pins can be associated with general purpose I/O pins and have passive pullup devices which can be enabled with a control bit when the TPM or general purpose I/O controls have configured the associated pin as an input. When no TPM function is enabled to use a corresponding pin, the pin reverts to being controlled by general purpose I/O controls, including the port-data and data-direction registers. Immediately after reset, no TPM functions are enabled, so all associated pins revert to general purpose I/O control.

16.2.1 Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 16-1 grouped all channel pins together, any TPM pin can be shared with the external clock source signal. Since I/O pin logic is not part of the TPM, refer to full-chip documentation for a specific derivative for more details about the interaction of TPM pin functions and general purpose I/O controls including port data, data direction, and pullup controls.

16.2.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow the user to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock which drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer; the frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock, to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin will not be usable for channel I/O function when selected as the external clock source. It is the user's responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can still be used in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

16.2.1.2 TPMxCHn — TPM Channel n I/O Pin(s)

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled whenever a port pin is acting as an input.

The TPM channel does not control the I/O pin when (ELSnB:ELSnA = 0:0) or when (CLKSB:CLKSA = 0:0) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the TPMxCHn pins are all controlled by the TPM system. When CPWMS=0, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPMxCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges will trigger input-capture events. A synchronizer based on the bus clock is used to synchronize input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA = 0:1 and ELSnB:ELSnA not = 0:0), the associated data direction control is overridden, the TPMxCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPMxCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event—then the pin is toggled.

When a channel is configured for edge-aligned PWM (CPWMS=0, MSnB=1 and ELSnB:ELSnA not = 0:0), the data direction is overridden, the TPMxCHn pin is forced to be an output controlled by the TPM, and ELSnA controls the polarity of the PWM output signal on the pin. When ELSnB:ELSnA=1:0, the TPMxCHn pin is forced high at the start of each new period (TPMxCNT=0x0000), and the pin is forced low when the channel value register matches the timer counter. When ELSnA=1, the TPMxCHn pin is forced low at the start of each new period (TPMxCNT=0x0000), and the pin is forced high when the channel value register matches the timer counter.

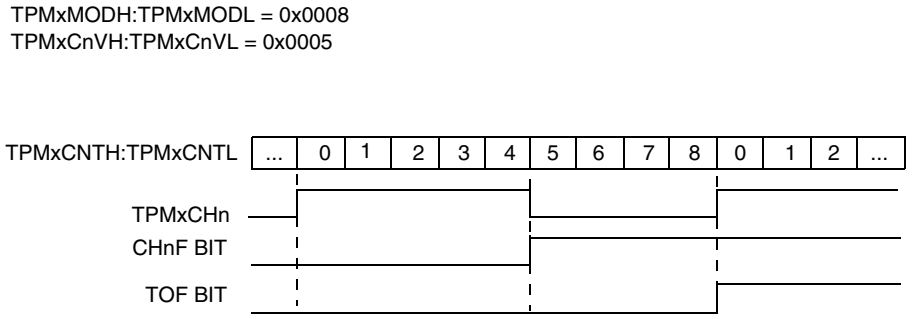


Figure 16-3. High-True Pulse of an Edge-Aligned PWM

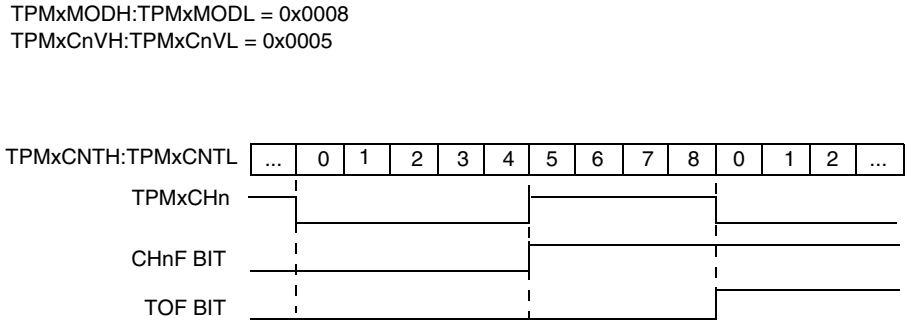


Figure 16-4. Low-True Pulse of an Edge-Aligned PWM

When the TPM is configured for center-aligned PWM (and ELSnB:ELSnA not = 0:0), the data direction for all channels in this TPM are overridden, the TPMxCHn pins are forced to be outputs controlled by the TPM, and the ELSnA bits control the polarity of each TPMxCHn output. If ELSnB:ELSnA=1:0, the corresponding TPMxCHn pin is cleared when the timer counter is counting up, and the channel value register matches the timer counter; the TPMxCHn pin is set when the timer counter is counting down, and the channel value register matches the timer counter. If ELSnA=1, the corresponding TPMxCHn pin is set when the timer counter is counting up and the channel value register matches the timer counter; the TPMxCHn pin is cleared when the timer counter is counting down and the channel value register matches the timer counter.

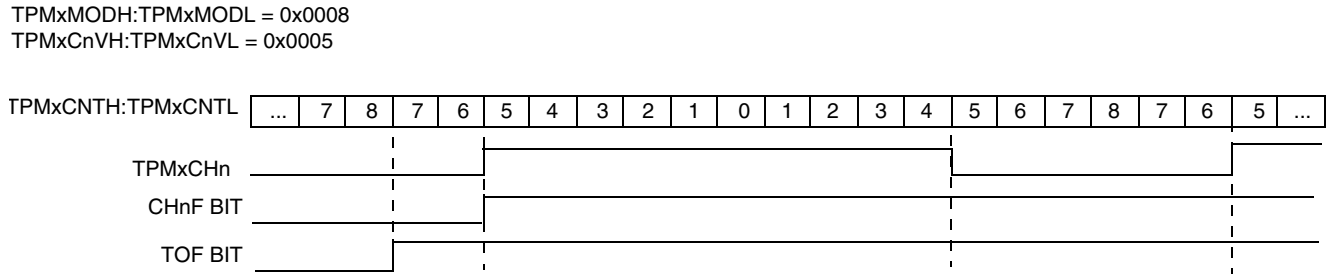


Figure 16-5. High-True Pulse of a Center-Aligned PWM

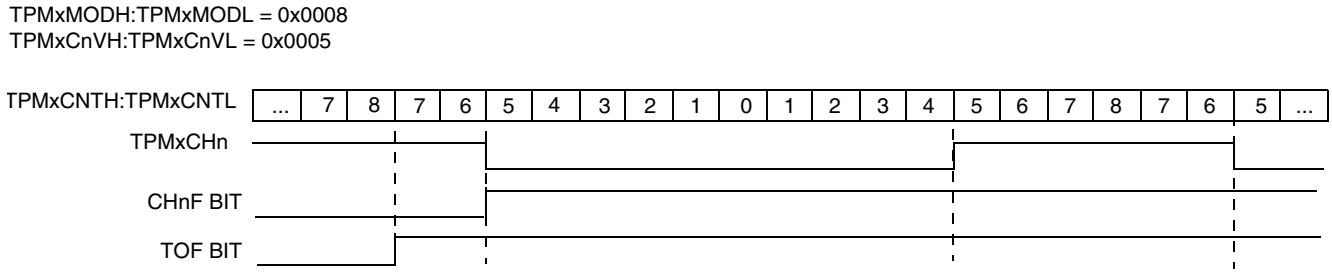


Figure 16-6. Low-True Pulse of a Center-Aligned PWM

16.3 Register Definition

This section consists of register descriptions in address order. A typical MCU system may contain multiple TPMs, and each TPM may have one to eight channels, so register names include placeholder characters to identify which TPM and which channel is being referenced. For example, TPMxCnSC refers to timer (TPM) x, channel n. TPM1C2SC would be the status and control register for channel 2 of timer 1.

16.3.1 TPM Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits used to configure the interrupt enable, TPM configuration, clock source, and prescale factor. These controls relate to all channels within this timer module.

	7	6	5	4	3	2	1	0
R	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
W	0							
Reset	0	0	0	0	0	0	0	0

Figure 16-7. TPM Status and Control Register (TPMxSC)

Table 16-2. TPMxSC Field Descriptions

Field	Description
7 TOF	Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed
6 TOIE	Timer overflow interrupt enable. This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals one. Reset clears TOIE. 0 TOF interrupts inhibited (use for software polling) 1 TOF interrupts enabled
5 CPWMS	Center-aligned PWM select. When present, this read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS. 0 All channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register. 1 All channels operate in center-aligned PWM mode.

Table 16-2. TPMxSC Field Descriptions (continued)

Field	Description
4–3 CLKS[B:A]	Clock source selects. As shown in Table 16-3, this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The fixed system clock source is only meaningful in systems with a PLL-based system clock. When there is no PLL, the fixed-system clock source is the same as the bus rate clock. The external source is synchronized to the bus clock by TPM module, and the fixed system clock source (when a PLL is present) is synchronized to the bus clock by an on-chip synchronization circuit. When a PLL is present but not enabled, the fixed-system clock source is the same as the bus-rate clock.
2–0 PS[2:0]	Prescale factor select. This 3-bit field selects one of 8 division factors for the TPM clock input as shown in Table 16-4. This prescaler is located after any clock source synchronization or clock source selection so it affects the clock source selected to drive the TPM system. The new prescale factor will affect the clock source on the next system clock cycle after the new value is updated into the register bits.

Table 16-3. TPM-Clock-Source Selection

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disable)
01	Bus rate clock
10	Fixed system clock
11	External source

Table 16-4. Prescale Factor Selection

PS2:PS1:PS0	TPM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

16.3.2 TPM-Counter Registers (TPMxCNTH:TPMxCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMxCNTH or TPMxCNTL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This allows coherent 16-bit reads in either big-endian or little-endian order which makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the timer status/control register (TPMxSC).

Reset clears the TPM counter registers. Writing any value to TPMxCNTH or TPMxCNTL also clears the TPM counter (TPMxCNTH:TPMxCNTL) and resets the coherency mechanism, regardless of the data involved in the write.

	7	6	5	4	3	2	1	0
R	Bit 15	14	13	12	11	10	9	Bit 8
W	Any write to TPMxCNTH clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0

Figure 16-8. TPM Counter Register High (TPMxCNTH)

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	Any write to TPMxCNTL clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0

Figure 16-9. TPM Counter Register Low (TPMxCNTL)

When BDM is active, the timer counter is frozen (this is the value that will be read by user); the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both counter halves are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution.

In BDM mode, writing any value to TPMxSC, TPMxCNTH or TPMxCNTL registers resets the read coherency mechanism of the TPMxCNTH:L registers, regardless of the data involved in the write.

16.3.3 TPM Counter Modulo Registers (TPMxMODH:TPMxMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock, and the overflow flag (TOF) becomes set. Writing to TPMxMODH or TPMxMODL inhibits the TOF bit and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000 which results in a free running timer counter (modulo disabled).

Writing to either byte (TPMxMODH or TPMxMODL) latches the value into a buffer and the registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), then the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), then the registers are updated after both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF

The latching mechanism may be manually reset by writing to the TPMxSC address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.

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

Figure 16-10. TPM Counter Modulo Register High (TPMxMODH)

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 16-11. TPM Counter Modulo Register Low (TPMxMODL)

Reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

16.3.4 TPM Channel n Status and Control Register (TPMxCnSC)

TPMxCnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

	7	6	5	4	3	2	1	0
R	CHnF	CHnIE	MSnB	MSnA	ELSnB	ELSnA	0	0
W	0							
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 16-12. TPM Channel n Status and Control Register (TPMxCnSC)

Table 16-5. TPMxCnSC Field Descriptions

Field	Description
7 CHnF	Channel n flag. When channel n is an input-capture channel, this read/write bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF will not be set even when the value in the TPM counter registers matches the value in the TPM channel n value registers. A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPMxCnSC while CHnF is set and then writing a logic 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF remains set after the clear sequence completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost due to clearing a previous CHnF. Reset clears the CHnF bit. Writing a logic 1 to CHnF has no effect. 0 No input capture or output compare event occurred on channel n 1 Input capture or output compare event on channel n
6 CHnIE	Channel n interrupt enable. This read/write bit enables interrupts from channel n. Reset clears CHnIE. 0 Channel n interrupt requests disabled (use for software polling) 1 Channel n interrupt requests enabled
5 MSnB	Mode select B for TPM channel n. When CPWMS=0, MSnB=1 configures TPM channel n for edge-aligned PWM mode. Refer to the summary of channel mode and setup controls in Table 16-6.
4 MSnA	Mode select A for TPM channel n. When CPWMS=0 and MSnB=0, MSnA configures TPM channel n for input-capture mode or output compare mode. Refer to Table 16-6 for a summary of channel mode and setup controls. **Note:** If the associated port pin is not stable for at least two bus clock cycles before changing to input capture mode, it is possible to get an unexpected indication of an edge trigger.
3–2 ELSnB ELSnA	Edge/level select bits. Depending upon the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 16-6, these bits select the polarity of the input edge that triggers an input capture event, select the level that will be driven in response to an output compare match, or select the polarity of the PWM output. Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general purpose I/O pin not related to any timer functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin.

Table 16-6. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin not used for TPM - revert to general purpose I/O or other peripheral control	
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	01	Output compare	Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
	1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)
		X1		Low-true pulses (set output on compare)
1	XX	10	Center-aligned PWM	High-true pulses (clear output on compare-up)
		X1		Low-true pulses (set output on compare-up)

16.3.5 TPM Channel Value Registers (TPMxCnVH:TPMxCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel registers are cleared by reset.

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

Figure 16-13. TPM Channel Value Register High (TPMxCnVH)

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 16-14. TPM Channel Value Register Low (TPMxCnVL)

In input capture mode, reading either byte (TPMxCnVH or TPMxCnVL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This latching mechanism also resets (becomes unlatched) when the TPMxCnSC register is written (whether BDM mode is active or not). Any write to the channel registers will be ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxCnSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the TPMxCnVH and TPMxCnVL registers in BDM mode is the value of these registers and not the value of their read buffer.

In output compare or PWM modes, writing to either byte (TPMxCnVH or TPMxCnVL) latches the value into a buffer. After both bytes are written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKSB:CLKSA bits and the selected mode, so:

- If (CLKSB:CLKSA = 0:0), then the registers are updated when the second byte is written.
- If (CLKSB:CLKSA not = 0:0 and in output compare mode) then the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).
- If (CLKSB:CLKSA not = 0:0 and in EPWM or CPWM modes), then the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

The latching mechanism may be manually reset by writing to the TPMxCnSC register (whether BDM mode is active or not). This latching mechanism allows coherent 16-bit writes in either big-endian or little-endian order which is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active even if one or both halves of the channel register are written while BDM is active. Any write to the channel registers bypasses the buffer latches and directly write to the channel register while BDM is active. The values written to the channel register while BDM is active are used for PWM & output compare operation once normal execution resumes. Writes to the channel registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

16.4 Functional Description

All TPM functions are associated with a central 16-bit counter which allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM

and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

The following sections describe the main counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend upon the operating mode, these topics will be covered in the associated mode explanation sections.

16.4.1 Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock source, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

16.4.1.1 Counter Clock Source

The 2-bit field, CLKSB:CLKSA, in the timer status and control register (TPMxSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See Table 16-3. After any MCU reset, CLKSB:CLKSA=0:0 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKSB:CLKSA field) does not affect the values in the counter or other timer registers.

Table 16-7. TPM Clock Source Selection

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disabled)
01	Bus rate clock
10	Fixed system clock
11	External source

The bus rate clock is the main system bus clock for the MCU. This clock source requires no synchronization because it is the clock that is used for all internal MCU activities including operation of the CPU and buses.

In MCUs that have no PLL or the PLL is not engaged, the fixed system clock source is the same as the bus-rate-clock source, and it does not go through a synchronizer. When a PLL is present and engaged, a synchronizer is required between the crystal divided-by two clock source and the timer counter so counter transitions will be properly aligned to bus-clock transitions. A synchronizer will be used at chip level to synchronize the crystal-related source clock to the bus clock.

The external clock source may be connected to any TPM channel pin. This clock source always has to pass through a synchronizer to assure that counter transitions are properly aligned to bus clock transitions. The bus-rate clock drives the synchronizer; therefore, to meet Nyquist criteria even with jitter, the frequency of the external clock source must not be faster than the bus rate divided-by four. With ideal clocks the external clock can be as fast as bus clock divided by four.

When the external clock source shares the TPM channel pin, this pin should not be used for other channel timing functions. For example, it would be ambiguous to configure channel 0 for input capture when the TPM channel 0 pin was also being used as the timer external clock source. (It is the user's responsibility to avoid such settings.) The TPM channel could still be used in output compare mode for software timing functions (pin controls set not to affect the TPM channel pin).

16.4.1.2 Counter Overflow and Modulo Reset

An interrupt flag and enable are associated with the 16-bit main counter. The flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE=0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE=1) where a static hardware interrupt is generated whenever the TOF flag is equal to one.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS=1). In the simplest mode, there is no modulus limit and the TPM is not in CPWMS=1 mode. In this case, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS=1), the TOF flag gets set as the counter changes direction at the end of the count value set in the modulus register (that is, at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

16.4.1.3 Counting Modes

The main timer counter has two counting modes. When center-aligned PWM is selected (CPWMS=1), the counter operates in up/down counting mode. Otherwise, the counter operates as a simple up counter. As an up counter, the timer counter counts from 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMxMODH:TPMxMODL.

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. Both 0x0000 and the terminal count value are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) becomes set at the end of the terminal-count period (as the count changes to the next lower count value).

16.4.1.4 Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to either half of TPMxCNTH or TPMxCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only half of the counter was read before resetting the count.

16.4.2 Channel Mode Selection

Provided CPWMS=0, the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

16.4.2.1 Input Capture Mode

With the input-capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input-capture channel, the TPM latches the contents of the TPM counter into the channel-value registers (TPMxCnVH:TPMxCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

In input capture mode, the TPMxCnVH and TPMxCnVL registers are read only.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An input capture event sets a flag bit (CHnF) which may optionally generate a CPU interrupt request.

While in BDM, the input capture function works as configured by the user. When an external event occurs, the TPM latches the contents of the TPM counter (which is frozen because of the BDM mode) into the channel value registers and sets the flag bit.

16.4.2.2 Output Compare Mode

With the output-compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel-value registers of an output-compare channel, the TPM can set, clear, or toggle the channel pin.

In output compare mode, values are transferred to the corresponding timer channel registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) which may optionally generate a CPU-interrupt request.

16.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS=0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the value of the modulus register (TPMxMODH:TPMxMODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 16-15). The time between the modulus overflow and the output compare is the pulse width. If ELSnA=0, the counter overflow forces the PWM signal high, and the output compare forces the PWM signal low. If ELSnA=1, the counter overflow forces the PWM signal low, and the output compare forces the PWM signal high.

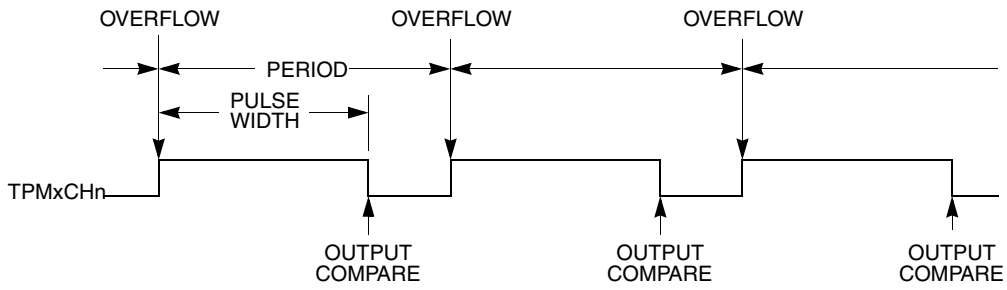


Figure 16-15. PWM Period and Pulse Width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. 100% duty cycle can be achieved by setting the timer-channel register (TPMxCnVH:TPMxCnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF in order to get 100% duty cycle.

Because the TPM may be used in an 8-bit MCU, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMxCnVH and TPMxCnVL, actually write to buffer registers. In edge-aligned PWM mode, values are transferred to the corresponding timer-channel registers according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

16.4.2.4 Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The output compare value in TPMxCnVH:TPMxCnVL determines the pulse width (duty cycle) of the PWM signal while the period is determined by the value in TPMxMODH:TPMxMODL. TPMxMODH:TPMxMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

pulse width = 2 x (TPMxCnVH:TPMxCnVL)

period = 2 x (TPMxMODH:TPMxMODL); TPMxMODH:TPMxMODL=0x0001-0x7FFF

If the channel-value register TPMxCnVH:TPMxCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMxCnVH:TPMxCnVL is a positive value (bit 15 clear) and is greater than the (non-zero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if you do not need to generate 100% duty cycle). This is not a significant limitation. The resulting period would be much longer than required for normal applications.

TPMxMODH:TPMxMODL=0x0000 is a special case that should not be used with center-aligned PWM mode. When CPWMS=0, this case corresponds to the counter running free from 0x0000 through 0xFFFF, but when CPWMS=1 the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

The output compare value in the TPM channel registers (times 2) determines the pulse width (duty cycle) of the CPWM signal (Figure 16-16). If ELSnA=0, a compare occurred while counting up forces the CPWM output signal low and a compare occurred while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPMxMODH:TPMxMODL, then counts down until it reaches zero. This sets the period equal to two times TPMxMODH:TPMxMODL.

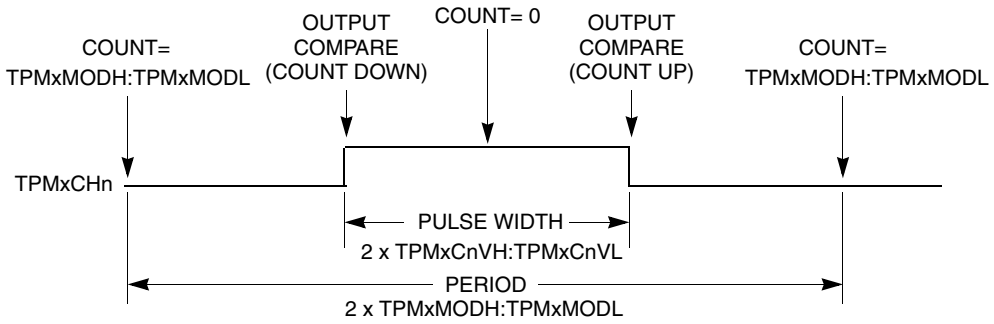


Figure 16-16. CPWM Period and Pulse Width (ELSnA=0)

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Input capture, output compare, and edge-aligned PWM functions do not make sense when the counter is operating in up/down counting mode so this implies that all active channels within a TPM must be used in CPWM mode when CPWMS=1.

The TPM may be used in an 8-bit MCU. The settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMxMODH, TPMxMODL, TPMxCnVH, and TPMxCnVL, actually write to buffer registers.

In center-aligned PWM mode, the TPMxCnVH:L registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

When TPMxCNTH:TPMxCNTL=TPMxMODH:TPMxMODL, the TPM can optionally generate a TOF interrupt (at the end of this count).

Writing to TPMxSC cancels any values written to TPMxMODH and/or TPMxMODL and resets the coherency mechanism for the modulo registers. Writing to TPMxCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMxCnVH:TPMxCnVL.

16.5 Reset Overview

16.5.1 General

The TPM is reset whenever any MCU reset occurs.

16.5.2 Description of Reset Operation

Reset clears the TPMxSC register which disables clocks to the TPM and disables timer overflow interrupts (TOIE=0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared which configures all TPM channels for input-capture operation with the associated pins disconnected from I/O pin logic (so all MCU pins related to the TPM revert to general purpose I/O pins).

16.6 Interrupts

16.6.1 General

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on each channel's mode of operation. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register.

All TPM interrupts are listed in Table 16-8 which shows the interrupt name, the name of any local enable that can block the interrupt request from leaving the TPM and getting recognized by the separate interrupt processing logic.

Table 16-8. Interrupt Summary

Interrupt	Local Enable	Source	Description
TOF	TOIE	Counter overflow	Set each time the timer counter reaches its terminal count (at transition to next count value which is usually 0x0000)
CHnF	CHnIE	Channel event	An input capture or output compare event took place on channel n

The TPM module will provide a high-true interrupt signal. Vectors and priorities are determined at chip integration time in the interrupt module so refer to the user's guide for the interrupt module or to the chip's complete documentation for details.

16.6.2 Description of Interrupt Operation

For each interrupt source in the TPM, a flag bit is set upon recognition of the interrupt condition such as timer overflow, channel-input capture, or output-compare events. This flag may be read (polled) by software to determine that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set

to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt will generate whenever the associated interrupt flag equals one. The user's software must perform a sequence of steps to clear the interrupt flag before returning from the interrupt-service routine.

TPM interrupt flags are cleared by a two-step process including a read of the flag bit while it is set (1) followed by a write of zero (0) to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

16.6.2.1 Timer Overflow Interrupt (TOF) Description

The meaning and details of operation for TOF interrupts varies slightly depending upon the mode of operation of the TPM system (general purpose timing functions versus center-aligned PWM operation). The flag is cleared by the two step sequence described above.

16.6.2.1.1 Normal Case

Normally TOF is set when the timer counter changes from 0xFFFF to 0x0000. When the TPM is not configured for center-aligned PWM (CPWMS=0), TOF gets set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. This case corresponds to the normal meaning of counter overflow.

16.6.2.1.2 Center-Aligned PWM Case

When CPWMS=1, TOF gets set when the timer counter changes direction from up-counting to down-counting at the end of the terminal count (the value in the modulo register). In this case the TOF corresponds to the end of a PWM period.

16.6.2.2 Channel Event Interrupt Description

The meaning of channel interrupts depends on the channel's current mode (input-capture, output-compare, edge-aligned PWM, or center-aligned PWM).

16.6.2.2.1 Input Capture Events

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select no edge (off), rising edges, falling edges or any edge as the edge which triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the two-step sequence described in Section 16.6.2, "Description of Interrupt Operation."

16.6.2.2.2 Output Compare Events

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the two-step sequence described Section 16.6.2, "Description of Interrupt Operation."

16.6.2.2.3 PWM End-of-Duty-Cycle Events

For channels configured for PWM operation there are two possibilities. When the channel is configured for edge-aligned PWM, the channel flag gets set when the timer counter matches the channel value register which marks the end of the active duty cycle period. When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle period which are the times when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described Section 16.6.2, “Description of Interrupt Operation.”

Chapter 17
Universal Serial Bus Device Controller (S08USBV1)

17.1 Introduction

This chapter describes an universal serial bus device controller (S08USBV1) module that is based on the Universal Serial Bus Specification Rev 2.0. The USB bus is designed to replace existing bus interfaces such as RS-232, PS/2, and IEEE 1284 for PC peripherals.

The S08USBV1 module provides a single-chip solution for full-speed (12 Mbps) USB device applications, and integrates the required transceiver with Serial Interface Engine (SIE), 3.3 V regualtor, Endpoint RAM and other control logics.

17.1.1 Clocking Requirements

The S08USBV1 requires two clock sources, the 24 MHz bus clock and a 48 MHz reference clock. The 48 MHz clock is sourced directly from MCGOUT. To achieve the 48 MHz clock rate, the MCG must be configured properly for PLL engaged external (PEE) mode with an external crystal.

For USB operation, examples of MCG configuration using PEE mode include:

- 2 MHz crystal – RDIV = 000 and VDIV = 0110
- 4 MHz crystal – RDIV = 001 and VDIV = 0110

17.1.2 Current Consumption in USB Suspend

In USB suspend mode, the USB device current consumption is limited to 500 μA. When the USB device goes into suspend mode, the firmware typically enters stop3 to meet the USB suspend requirements on current consumption.

NOTE

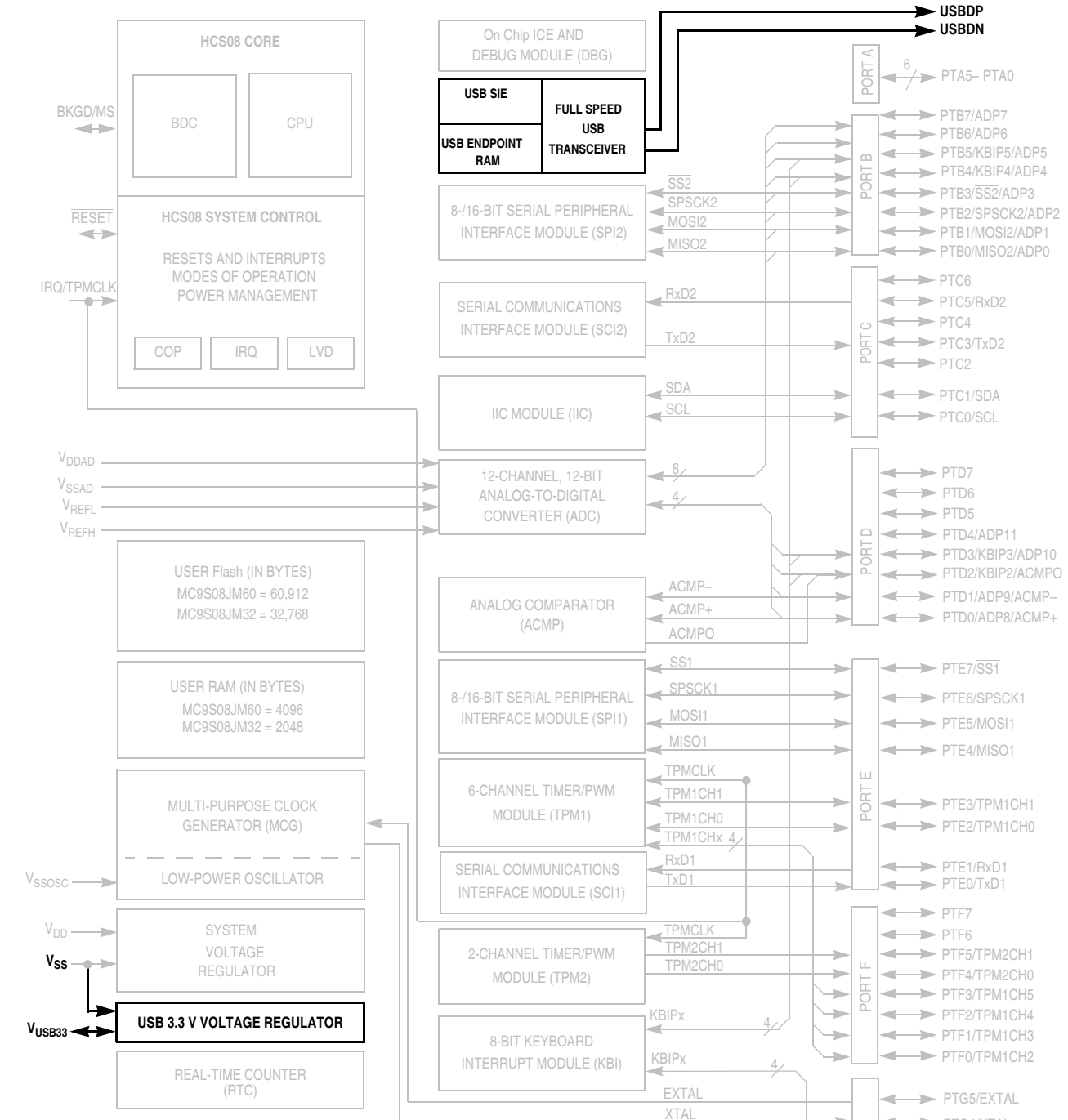
Enabling LVD increases current consumption in stop3. Consequently, when trying to satisfy USB suspend requirements, disabling LVD before entering stop3.

17.1.3 3.3 V Regulator

If using an external 3.3 V regulator as an input to V_{USB33} (only when USBVREN = 0), the supply voltage, V_{DD}, must not fall below the input voltage at the V_{USB33} pin. If using the internal 3.3 V regulator (USBVREN = 1), do not connect an external supply to the V_{USB33} pin. In this case, V_{DD} must fall between 3.9 V and 5.5 V for the internal 3.3 V regulator to operate correctly.

Table 17-1. USBVREN Configuration

USBVREN	3.3-V Regulator	VDD Supply Voltage Range
0	External 3.3-V Regulator (as input to V_{USB33} pin)	$V_{USB33} \leq V_{DD}$ Supply Voltage
1	Internal 3.3-V Regulator (no external supply connected to V_{USB33} pin)	3.9 V $\leq V_{DD}$ Supply Voltage $\leq$ 5.5V



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pull-down device ifpullup IRQ is enabled (IRQPE = 1). Pull-down is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pull-down device.

Figure 17-1. MC9S08JM60 Series Block Diagram Highlighting USB Blocks and Pins

17.1.4 Features

Features of the USB module include:

- USB 2.0 compliant
  - 12 Mbps full-speed (FS) data rate
  - USB data control logic:
    - Packet identification and decoding/generation
    - CRC generation and checking
    - NRZI (non-return-to-zero inverted) encoding/decoding
    - Bit-stuffing
    - Sync detection
    - End-of-packet detection
- Seven USB endpoints
  - Bidirectional endpoint 0
  - Six unidirectional data endpoints configurable as interrupt, bulk, or isochronous
  - Endpoints 5 and 6 support double-buffering
- USB RAM
  - 256 bytes of buffer RAM shared between system and USB module
  - RAM may be allocated as buffers for USB controller or extra system RAM resource
- USB reset options
  - USB module reset generated by MCU
  - Bus reset generated by the host, which triggers a CPU interrupt
- Suspend and resume operations with remote wakeup support
- Transceiver features
  - Converts USB differential voltages to digital logic signal levels
- On-chip USB pullup resistor
- On-chip 3.3-V regulator

17.1.5 Modes of Operation

Table 17-2. Operating Modes

Mode	Description
Stop1	USB module is not functional. Before entering stop1, the internal USB voltage regulator and USB transceiver enter shutdown mode; therefore, the USB voltage regulator and USB transceiver must be disabled by firmware.
Stop2	USB module is not functional. Before entering stop2, the internal USB voltage regulator and USB transceiver enter shutdown mode; therefore, the USB voltage regulator and USB transceiver must be disabled by firmware.

Table 17-2. Operating Modes (continued)

Mode	Description
Stop3	The USB module is optionally available in stop3. A reduced current consumption mode may be required for USB suspend mode per USB Specification Rev. 2.0, and stop3 mode is useful for achieving lower current consumption for the MCU and hence the overall USB device. Before entering stop3 via firmware, the user must ensure that the device settings are configured for stop3 to achieve USB suspend current consumption targets. The USB module is notified about entering suspend mode when the SLEEPF flag is set; this occurs after the USB bus is idle for 3 ms. The device USB suspend mode current consumption level requirements are defined by the USB Specification Rev. 2.0 (500 μA for low-power and 2.5 mA for high-power with remote-wakeup enabled). If USBRESMEN in USBCTL0 is set, and a K-state (resume signaling) is detected on the USB bus, the LPRESF bit in USBCTL0 will be set. This triggers an asynchronous interrupt that will wakeup the MCU from stop3 mode and enable clocks to the USB module. The USBRESMEN bit must then be cleared immediately after stop3 recovery to clear the LPRESF flag bit.
Wait	USB module is operational.

17.1.6 Block Diagram

Figure 17-2 is a block diagram of the USB module.

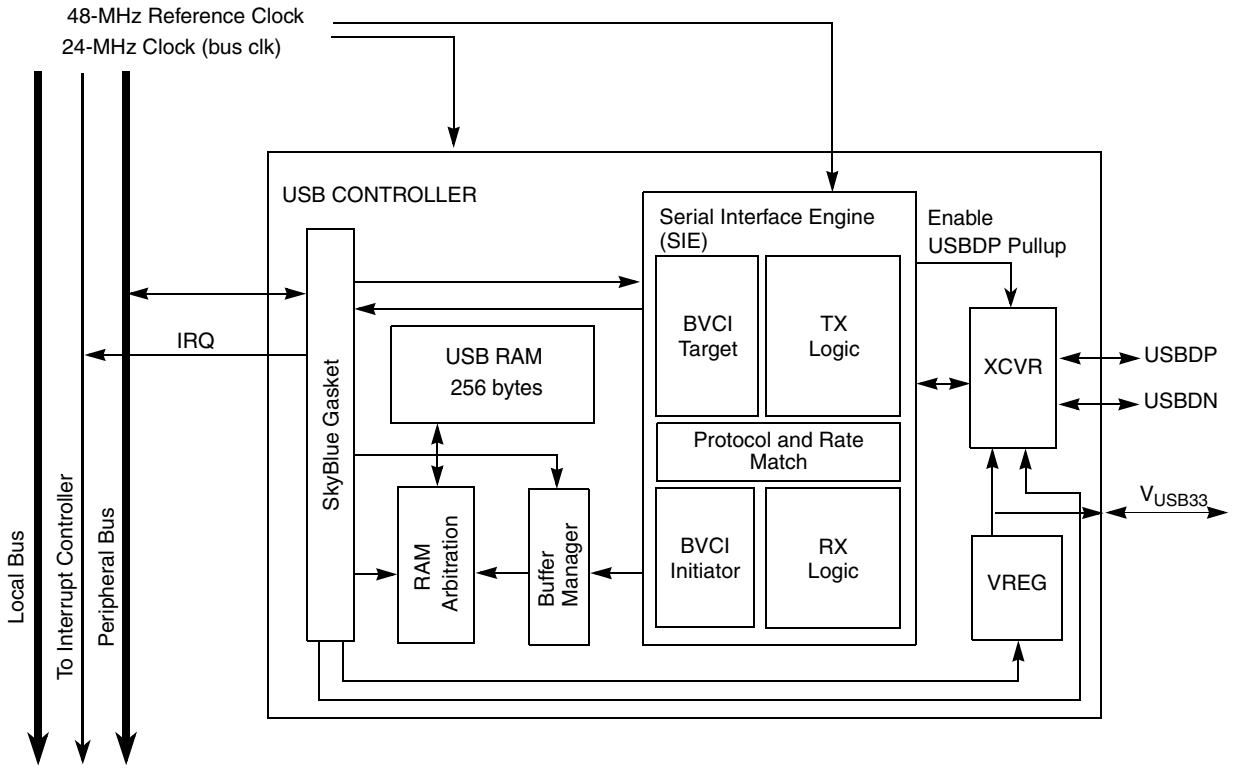


Figure 17-2. USB Module Block Diagram

17.2 External Signal Description

The USB module requires both data and power pins. Table 17-3 describes each of the USB external pin

Table 17-3. USB External Pins

Name	Port	Direction	Function	Reset State
Positive USB differential signal	USBDP	I/O	Differential USB signaling.	High impedance
Negative USB differential signal	USBDN	I/O	Differential USB signaling.	High impedance
USB voltage regulator power pin	V_{USB33}	Power	3.3 V USB voltage regulator output or 3.3 V USB transceiver/resistor supply input.	—

17.2.1 USBDP

USBDP is the positive USB differential signal. In a USB peripheral application, connect an external 33 Ω ±1% resistor in series with this signal in order to meet the USB Specification Rev. 2.0 impedance requirement.

17.2.2 USBDN

USBDN is the negative USB differential signal. In a USB peripheral application, connect an external 33 Ω ±1% resistor in series with this signal in order to meet the USB Specification, Rev. 2.0 impedance requirement.

17.2.3 V_{USB33}

V_{USB33} is connected to the on-chip 3.3-V voltage regulator (VREG). V_{USB33} maintains an output voltage of 3.3 V and can only source enough current for USB internal transceiver (XCVR) and USB pullup resistor. If the VREG is disabled by software, the application must input an external 3.3 V power supply to the USB module via V_{USB33}.

17.3 Register Definition

This section describes the memory map and control/status registers for the USB module.

17.3.1 USB Control Register 0 (USBCTL0)

	7	6	5	4	3	2	1	0
R	0	USBPU	USBRESMEN	LPRESF	0	USBVREN	0	USBPHYEN
W	USBRESET							
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 17-3. USB Transceiver and Regulator Control Register 0 (USBCTL0)

Table 17-4. USBCTL0 Field Descriptions

Field	Description
7 USBRESET	**USB Reset** — This bit generates a hard reset of the USB module, USBPHYEN and USBVREGEN bits will also be cleared. (need remember to restart USB Transceiver and USB voltage regulator). When set to 1, this bit automatically clears when the reset occurs. 0 USB module normal operation 1 Returns the USB module to its reset state
6 USBPU	**Pull Up Source** — This bit determines the source of the pullup resistor on the USBDP line. 0 Internal USBDP pullup resistor is disabled; The application can use an external pullup resistor 1 Internal USBDP pullup resistor is enabled
5 USBRESMEN	**USB Low-Power Resume Event Enable** — This bit, when set, enables the USB module to send an asynchronous wakeup interrupt to the MCU upon detection that the LPRESF bit has been set, indicating a K-state on the USB bus. This bit must be set before entering low-power stop3 mode only after SLEEPF=1 (USB is entering suspend mode). It must be cleared immediately after stop3 recovery in order to clear the Low-Power Resume Flag. 0 USB asynchronous wakeup from suspend mode disabled 1 USB asynchronous wakeup from suspend mode enabled
4 LPRESF	**Low-Power Resume Flag** — This bit becomes set in USB suspend mode if USBRESMEN=1 and a K-state is detected on the USB bus, indicating resume signaling while the device is in a low-power stop3 mode. This flag bit will trigger an asynchronous interrupt, which will wake the device from stop3. Firmware must then clear the USBRESMEN bit in order to clear the LPRESF bit. 0 No K-state detected on the USB bus while the device is in stop3 and the USB is suspended. 1 K-state detected on the USB bus when USBRESMEN=1, the device is in stop3, and the USB is suspended.
2 USBVREN	**USB Voltage Regulator Enable** — This bit enables the on-chip 3.3V USB voltage regulator. 0 On-chip USB voltage regulator is disabled (OFF MODE) 1 On-chip USB voltage regulator is enabled for active or standby mode
0 USBPHYEN	**USB Transceiver Enable** — When the USB Transceiver (XCVR) is disabled, USBDP and USBDN are hi-Z. It is recommended that the XCVR be enabled before setting the USBEN bit in the CTL register. The firmware must ensure that the XCVR remains enabled when entering USB SUSPEND mode. 0 On-chip XCVR is disabled 1 On-chip XCVR is enabled

17.3.2 Peripheral ID Register (PERID)

The PERID reads back the value of 0x04. This value is defined for the USB module peripheral.

	7	6	5	4	3	2	1	0
R	0	0	ID5	ID4	ID3	ID2	ID1	ID0
W								
Reset	0	0	0	0	0	1	0	0

= Unimplemented or Reserved

Figure 17-4. Peripheral ID Register (PERID)

Table 17-5. PERID Field Descriptions

Field	Description
5:0 ID[5:0]	**Peripheral Configuration Number** —This number is set to 0x04 and indicates that the peripheral is the full-speed USB module.

17.3.3 Peripheral ID Complement Register (IDCOMP)

The IDCOMP reads back the complement of the peripheral ID register. For the USB module peripheral this will be 0xFB.

	7	6	5	4	3	2	1	0
R	1	1	NID5	NID4	NID3	NID2	NID1	NID0
W								
Reset	1	1	1	1	1	0	1	1

= Unimplemented or Reserved

Figure 17-5. Peripheral ID Complement Register (IDCOMP)

Table 17-6. IDCOMP Field Descriptions

Field	Description
5:0 NID[5:0]	**Compliment ID Number** — One's complement version of ID[5:0].

17.3.4 Peripheral Revision Register (REV)

The REV reads back the value of the USB peripheral revision.

	7	6	5	4	3	2	1	0
R	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 17-6. Peripheral Revision Register (REV)

Table 17-7. REV Field Descriptions

Field	Description
8–0 REV[7:0]	**Revision** — Revision number of the USB module.

17.3.5 Interrupt Status Register (INTSTAT)

The INTSTAT contains bits for each of the interrupt source within the USB module. Each of these bits is qualified with its respective interrupt enable bits (see the interrupt enable register). All bits of the register are logically OR'ed together to form a single interrupt source for the microcontroller. Once an interrupt bit has been set, it may only be cleared by writing a 1 to the respective interrupt bit. This register will contain the value of 0x00 after a reset.

	7	6	5	4	3	2	1	0
R	STALLF	0	RESUMEF	SLEEPF	TOKDNEF	SOFTOKF	ERRORF	USBRSTF
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 17-8. Interrupt Status Register (INTSTAT)

Table 17-9. INTSTAT Field Descriptions

Field	Description
7 STALLF	**Stall Flag** — The stall interrupt is used in device mode. In device mode the stall flag is asserted when a STALL handshake is sent by the serial interface engine (SIE). 0 A STALL handshake has not been sent 1 A STALL handshake has been sent
5 RESUMEF	**Resume Flag** — This bit is set 2.5 μs after clocks to the USB module have restarted following resume signaling. It can be used to indicate remote wakeup signaling on the USB bus. This interrupt is enabled only when the USB module is about to enter suspend mode (usually when SLEEPF interrupt detected). 0 No RESUME observed 1 RESUME detected (K-state is observed on the USBDP/USBDN signals for 2.5 μs)
4 SLEEPF	**Sleep Flag** — This bit is set if the USB module has detected a constant idle on the USB bus for 3 ms, indicating that the USB module will go into suspend mode. The sleep timer is reset by activity on the USB bus. 0 No constant idle state of 3 ms has been detected on the USB bus 1 A constant idle state of 3 ms has been detected on the USB bus
3 TOKDNEF	**Token Complete Flag** — This bit is set when the current transaction is completed. The firmware must immediately read the STAT register to determine the endpoint and BD information. Clearing this bit (by setting it to 1) causes the STAT register to be cleared or the STAT FIFO holding register to be loaded into the STAT register. 0 No tokens being processed are complete 1 Current token being processed is complete
2 SOFTOKF	**SOF Token Flag** — This bit is set if the USB module has received a start of frame (SOF) token. 0 The USB module has not received an SOF token 1 The USB module has received an SOF token

Table 17-9. INTSTAT Field Descriptions (continued)

Field	Description
1 ERRORF	**Error Flag** — This bit is set when any of the error conditions within the ERRSTAT register has occurred. The firmware must then read the ERRSTAT register to determine the source of the error. 0 No error conditions within the ERRSTAT register have been detected 1 Error conditions within the ERRSTAT register have been detected
0 USBRSTF	**USB Reset Flag** —This bit is set when the USB module has decoded a valid USB reset. When asserted, this bit will inform the MCU to automatically write 0x00 to the address register and to enable endpoint 0. USBRSTF is set once a USB reset has been detected for 2.5 μs. It will not be asserted again until the USB reset condition has been removed, and then reasserted. 0 No USB reset observed 1 USB reset detected

17.3.6 Interrupt Enable Register (INTENB)

The INTENB contains enabling bits for each of the interrupt sources within the USB module. Setting any of these bits will enable the respective interrupt source in the INTSTAT register. This register will contain the value of 0x00 after a reset, i.e. all interrupts disabled.

	7	6	5	4	3	2	1	0
R	STALL	0	RESUME	SLEEP	TOKDNE	SOFTOK	ERROR	USBRST
W								
Reset	0	0	0	0	0	0	0	0

Figure 17-9. Interrupt Enable Register (INTENB)

Table 17-10. INTENB Field Descriptions

Field	Description
7 STALL	**STALL Interrupt Enable** — Setting this bit will enable STALL interrupts. 0 Interrupt disabled 1 Interrupt enabled
5 RESUME	**RESUME Interrupt Enable** — Setting this bit will enable RESUME interrupts. 0 Interrupt disabled 1 Interrupt enabled
4 SLEEP	**SLEEP Interrupt Enable** — Setting this bit will enable SLEEP interrupts. 0 Interrupt disabled 1 Interrupt enabled
3 TOKDNE	**TOKDNE Interrupt Enable** — Setting this bit will enable TOKDNE interrupts. 0 Interrupt disabled 1 Interrupt enabled
2 SOFTOK	**SOFTOK Interrupt Enable** — Setting this bit will enable SOFTOK interrupts. 0 Interrupt disabled 1 Interrupt enabled

Table 17-10. INTENB Field Descriptions (continued)

Field	Description
1 ERROR	**ERROR Interrupt Enable** — Setting this bit will enable ERROR interrupts. 0 Interrupt disabled 1 Interrupt enabled
0 USBRST	**USBRST Interrupt Enable** — Setting this bit will enable USBRST interrupts. 0 Interrupt disabled 1 Interrupt enabled

17.3.7 Error Interrupt Status Register (ERRSTAT)

The ERRSTAT contains bits for each of the error sources within the USB module. Each of these bits corresponds to its respective error enable bit (See Section 17.3.8, “Error Interrupt Enable Register (ERRENB)”.) The result is OR'ed together and sent to the ERROR bit of the INTSTAT register. Once an interrupt bit has been set, it may only be cleared by writing a 1 to the corresponding flag bit. Each bit is set as soon as the error condition is detected. Thus, the interrupt will typically not correspond with the end of a token being processed. This register will contain the value of 0x00 after reset.

	7	6	5	4	3	2	1	0
R W	BTSERRF	Reserved	BUFERRF	BTOERRF	DFN8F	CRC16F	CRC5F	PIDERRF
Reset	0	0	0	0	0	0	0	0

Figure 17-10. Error Interrupt Status Register (ERRSTAT)

Table 17-11. ERRSTAT Field Descriptions

Field	Description
7 BTSERRF	**Bit Stuff Error Flag** — A bit stuff error has been detected. If set, the corresponding packet will be rejected due to a bit stuff error. 0 No bit stuff error detected 1 Bit stuff error flag set
5 BUFERRF	**Buffer Error Flag** — This bit is set if the USB module has requested a memory access to read a new BD but has not been given the bus before the USB module needs to receive or transmit data. If processing a TX (IN endpoint) transfer, this would cause a transmit data underflow condition. Or if processing an Rx (OUT endpoint) transfer, this would cause a receive data overflow condition. This bit is also set if a data packet to or from the host is larger than the buffer size that is allocated in the BD. In this case the data packet is truncated as it is put into buffer memory. 0 No buffer error detected 1 A buffer error has occurred
4 BTOERRF	**Bus Turnaround Error Timeout Flag** — This bit is set if a bus turnaround timeout error has occurred. The USB module uses a bus turnaround timer to keep track of the amount of time elapsed between the token and data phases of a SETUP or OUT TOKEN or the data and handshake phases of an IN TOKEN. If more than 16-bit times are counted from the previous EOP before a transition from IDLE, a bus turnaround timeout error will occur. 0 No bus turnaround timeout error has been detected 1 A bus turnaround timeout error has occurred

Table 17-11. ERRSTAT Field Descriptions (continued)

Field	Description
3 DFN8F	**Data Field Error Flag** — The data field received was not an interval of 8 bits. The USB Specification specifies that the data field must be an integer number of bytes. If the data field was not an integer number of bytes, this bit will be set. 0 The data field was an integer number of bytes 1 The data field was not an integer number of bytes
2 CRC16F	**CRC16 Error Flag** — The CRC16 failed. If set, the data packet was rejected due to a CRC16 error. 0 No CRC16 error detected 1 CRC16 error detected
1 CRC5F	**CRC5 Error Flag** — This bit will detect a CRC5 error in the token packets generated by the host. If set, the token packet was rejected due to a CRC5 error. 0 No CRC5 error detected 1 CRC5 error detected, and the token packet was rejected.
0 PIDERRF	**PID Error Flag** — The PID check failed. 0 No PID check error detected 1 PID check error detected

17.3.8 Error Interrupt Enable Register (ERRENB)

	7	6	5	4	3	2	1	0
R	BTSERR	0	BUFERR	BTOERR	DFN8	CRC16	CRC5	PIDERR
W								
Reset	0	0	0	0	0	0	0	0

Figure 17-11. Error Interrupt Enable Register (ERRENB)

Table 17-12. ERRSTAT Field Descriptions

Field	Description
7 BTSERR	**BTSERR Interrupt Enable** — Setting this bit will enable BTSERR interrupts. 0 Interrupt disabled 1 Interrupt enabled
5 BUFERR	**BUFERR Interrupt Enable** — Setting this bit will enable BUFERR interrupts. 0 Interrupt disabled 1 Interrupt enabled
4 BTOERR	**BTOERR Interrupt Enable** — Setting this bit will enable BTOERR interrupts. 0 Interrupt disabled 1 Interrupt enabled
3 DFN8	**DFN8 Interrupt Enable** — Setting this bit will enable DFN8 interrupts. 0 Interrupt disabled 1 Interrupt enabled
2 CRC16	**CRC16 Interrupt Enable** — Setting this bit will enable CRC16 interrupts. 0 Interrupt disabled 1 Interrupt enabled

Table 17-12. ERRSTAT Field Descriptions (continued)

Field	Description
1 CRC5	**CRC5 Interrupt Enable** — Setting this bit will enable CRC5 interrupts. 0 Interrupt disabled 1 Interrupt enabled
0 PIDERR	**PIDERR Interrupt Enable** — Setting this bit will enable PIDERR interrupts. 0 Interrupt disabled 1 Interrupt enabled

17.3.9 Status Register (STAT)

The STAT reports the transaction status within the USB module. When the MCU receives a TOKDNE interrupt, the STAT is read to determine the status of the previous endpoint communication. The data in the status register is valid only when the TOKDNEF interrupt flag is asserted. The STAT register is actually a read window into a status FIFO maintained by the USB module. When the USB module uses a BD, it updates the status register. If another USB transaction is performed before the TOKDNE interrupt is serviced, the USB module will store the status of the next transaction in the STAT FIFO. Thus, the STAT register is actually a four byte FIFO which allows the microcontroller to process one transaction while the serial interface engine (SIE) is processing the next. Clearing the TOKDNEF bit in the INTSTAT register causes the SIE to update the STAT register with the contents of the next STAT value. If the next data in the STAT FIFO holding register is valid, the SIE will immediately reassert the TOKDNE interrupt.

	7	6	5	4	3	2	1	0
R	ENDP[3:0]				IN	ODD	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 17-12. Status Register (STAT)

Table 17-13. STAT Field Descriptions

Field	Description
7–4 ENDP[3:0]	**Endpoint Number** — These four bits encode the endpoint address that received or transmitted the previous token. This allows the microcontroller to determine which BDT entry was updated by the last USB transaction. 0000 Endpoint 0 0001 Endpoint 1 0010 Endpoint 2 0011 Endpoint 3 0100 Endpoint 4 0101 Endpoint 5 0110 Endpoint 6

Table 17-13. STAT Field Descriptions (continued)

Field	Description
3 IN	**In/Out Transaction** — This bit indicates whether the last BDT updated was for a transmit (IN) transfer or a receive (OUT) data transfer. 0 Last transaction was a receive (OUT) data transfer 1 Last BDT updated was for transmit (IN) transfer
2 ODD	**Odd/Even Transaction** —This bit indicates whether the last buffer descriptor updated was in the odd bank of the BDT or the even bank of the BDT, See earlier section for more information on BDT address generation. 0 Last buffer descriptor updated was in the EVEN bank 1 Last buffer descriptor updated was in the ODD bank

17.3.10 Control Register (CTL)

The CTL provides various control and configuration information for the USB module.

	7	6	5	4	3	2	1	0
R W			TSUSPEND			CRESUME	ODDRST	USBEN
Reset	0	0	0	0	0	0	0	0

Figure 17-13. Control Register (CTL)

Table 17-14. CTL Field Descriptions

Field	Description
5 TSUSPEND	**Transaction Suspend** — This bit is set by the serial interface engine (SIE) when a setup token is received, allowing software to dequeue any pending packet transactions in the BDT before resuming token processing. The TSUSPEND bit informs the processor that the SIE has disabled packet transmission and reception. Clearing this bit allows the SIE to continue token processing. 0 Allows the SIE to continue token processing 1 Set by the SIE when a setup token is received; SIE has disabled packet transmission and reception.
2 CRESUME	**Resume Signaling** — Setting this bit will allow the USB module to execute resume signaling. This will allow the USB module to perform remote wakeup. Software must set CRESUME to 1 for the amount of time required by the USB Specification Rev. 2.0 and then clear it to 0. 0 Do not execute remote wakeup 1 Execute resume signaling - remote wakeup
1 ODDRST	**Odd Reset** — Setting this bit will reset all the buffer descriptor ODD ping-pong bits to 0 which will then specify the EVEN descriptor bank. This bit is used with double-buffered endpoints 5 and 6. This bit has no effect on endpoints 0 through 4. 0 Do not reset 1 Reset all the buffer descriptor ODD ping/pong bits to 0 which will then specify the EVEN descriptor bank
0 USBEN	**USB Enable** Setting this bit will enable the USB module to operate. Setting this bit causes the SIE to reset all of its ODD bits to the BDTs. Thus, setting this bit will reset much of the logic in the SIE. 0 Disable the USB module 1 Enable the USB module for operation, will not affect Transceiver and VREG.

17.3.11 Address Register (ADDR)

The ADDR register contains the unique 7-bit address the device will be recognized as through USB. The register is reset to 0x00 after the reset input has gone active or the USB module has decoded USB reset signaling. That will initialize the address register to decode address 0x00 as required by the USB specification. Firmware will change the value when it processes a SET_ADDRESS request.

	7	6	5	4	3	2	1	0
R	0	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
W								
Reset	0	0	0	0	0	0	0	0

Figure 17-14. Address Register (ADDR)

Table 17-15. ADDR Field Descriptions

Field	Description
6–0 ADDR[6:0]	**USB Address** — This 7-bit value defines the USB address that the USB module will decode

17.3.12 Frame Number Register (FRMNUML, FRMNUMH)

The frame number registers contains the 11-bit frame number. The frame number registers require two 8-bit registers to implement. The low order byte is contained in FRMNUML, and the high order byte is contained in FRMNUMH. These registers are updated with the current frame number whenever a SOF TOKEN is received.

	7	6	5	4	3	2	1	0
R	FRM7	FRM6	FRM5	FRM4	FRM3	FRM2	FRM1	FRM0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 17-15. Frame Number Register Low (FRMNUML)

Table 17-16. FRMNUML Field Descriptions

Field	Description
7–0 FRM[7:0]	**Frame Number** — These bits represent the low order bits of the 11 bit frame number.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	FRM10	FRM9	FRM8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 17-16. Frame Number Register High (FRMNUMH)

Table 17-17. FRMNUMH Field Descriptions

Field	Description
2–0 FRM[10:8]	**Frame Number** — These bits represent the high order bits of the 11-bit frame number.

17.3.13 Endpoint Control Register (EPCTLn, n=0-6)

The endpoint control registers contains the endpoint control bits (EPCTLDIS, EPRXEN, EPTXEN, and EPHSHK) for each endpoint available within the USB module for a decoded address. These four bits define all of the control necessary for any one endpoint. The formats for these registers are shown in the tables below. Endpoint 0 (ENDP0) is associated with control pipe 0 which is required by the USB for all functions. Therefore, after a USBRST interrupt has been received, the microcontroller must set EPCTL0 to contain 0x0D.

	7	6	5	4	3	2	1	0
R	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
W								
Reset (EP0-6)	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 17-17. Endpoint Control Register (EPCTLn)

Table 17-18. EPCTLn Field Descriptions

Field	Description
4 EPCTLDIS	**Endpoint Control** — This bit defines if an endpoint is enabled and the direction of the endpoint. The endpoint enable/direction control is defined in Table 17-19.
3 EPRXEN	**Endpoint Rx Enable** — This bit defines if an endpoint is enabled for OUT transfers. The endpoint enable/direction control is defined in Table 17-19.
2 EPTXEN	**Endpoint Tx Enable** — This bit defines if an endpoint is enabled for IN transfers. The endpoint enable/direction control is defined in Table 17-19.

Table 17-18. EPCTLn Field Descriptions (continued)

Field	Description
1 EPSTALL	**Endpoint Stall** — When set, this bit indicates that the endpoint is stalled. This bit has priority over all other control bits in the endpoint control register, but is only valid if EPTXEN=1 or EPRXEN=1. Any access to this endpoint will cause the USB module to return a STALL handshake. Once an endpoint is stalled it requires intervention from the host controller. 0 Endpoint n is not stalled 1 Endpoint n is stalled
0 EPHSHK	**Endpoint Handshake** — This bit determines if the endpoint will perform handshaking during a transaction to the endpoint. This bit will generally be set unless the endpoint is isochronous. 0 No handshaking performed during a transaction to this endpoint (usually for isochronous endpoints) 1 Handshaking performed during a transaction to this endpoint

Table 17-19. Endpoint Enable/Direction Control

Bit Name			Endpoint Enable/Direction Control
4 EPCTLDIS	3 EPRXEN	2 EPTXEN	
X	0	0	Disable endpoint
X	0	1	Enable endpoint for IN(TX) transfers only
X	1	0	Enable endpoint for OUT(RX) transfers only
0	1	1	Enable endpoint for IN, OUT and SETUP transfers.
1	1	1	RESERVED

17.4 Functional Description

This section describes the functional behavior of the USB module. It documents data packet processing for endpoint 0 and data endpoints, USB suspend and resume states, SOF token processing, reset conditions and interrupts.

17.4.1 Block Descriptions

Figure 17-2 is the block diagram. The module's sub-blocks and external signals are described in the following sections. The module involves several major blocks — USB transceiver (XCVR), USB serial interface engine (SIE), a 3.3 V regulator (VREG), endpoint buffer manager, shared RAM arbitration, USB RAM and the SkyBlue gasket.

17.4.1.1 USB Serial Interface Engine (SIE)

The SIE is composed of two major functions: TX Logic and RX Logic. These major functions are described below in more detail. The TX and RX logic are connected by a USB protocol engine which manages packet flow to and from the USB module. The SIE is connected to the rest of the system via

internal basic virtual component interface (BVCI) compliant target and initiator buses. The BVCI target interface is used to configure the USB SIE and to provide status and interrupts to CPU. The BVCI initiator interface provides the integrated DMA controller access to the buffer descriptor table (BDT), and transfers USB data to or from USB RAM memory.

17.4.1.1.1 Serial Interface Engine (SIE) Transmitter Logic

The SIE transmitter logic has two primary functions. The first is to format the USB data packets that have been stored in the endpoint buffers. The second is to transmit data packets via the USB transceiver.

All of the necessary USB data formatting is performed by the SIE transmitter logic, including:

- NRZI encoding
- bit-stuffing
- CRC computation
- addition of the SYNC field
- addition of the End-of-packet (EOP)

The CPU typically places data in the endpoint buffers as part of the application. When the buffer is configured as an IN buffer and the USB host requests a packet, the SIE responds with a properly formatted data packet.

The transmitter logic is also used to generate responses to packets received from the USB host. When a properly formatted packet is received from the USB host, the transmitter logic responds with the appropriate ACK, NAK or STALL handshake.

When the SIE transmitter logic is transmitting data from the buffer space for a particular endpoint, CPU access to that endpoint buffer space is not recommended.

17.4.1.1.2 Serial Interface Engine (SIE) Receiver Logic

The SIE receiver logic receives USB data and stores USB packets in USB RAM for processing by the CPU and the application software. Serial data from the transceiver is converted to a byte-wide parallel data stream, checked for proper packet framing, and stored in the USB RAM memory.

Received bitstream processing includes the following operations:

- decodes an NRZI USB serial data stream
- Sync detection
- Bit-stuff removal (and error detection)
- End-of-packet (EOP) detection
- CRC validation
- PID check
- other USB protocol layer checks.

The SIE receiver logic provides error detection including:

- Bad CRC
- Timeout detection for EOP

- Bit stuffing violation

If a properly formatted packet is received, the receiver logic initiates a handshake response to the host. If the packet is not decoded correctly due to bit stuff violation, CRC error or other packet level problem, the receiver ignores it. The USB host will eventually time-out waiting for a response, and retransmit the packet.

When the SIE receiver logic is receiving data in the buffer space for a particular endpoint, CPU access to that buffer space is not recommended.

17.4.1.2 MCU/Memory Interfaces

17.4.1.2.1 SkyBlue Gasket

The SkyBlue gasket connects the USB module to the SoC internal peripheral bus. The gasket maps accesses to the endpoint buffer descriptors or the endpoint buffers into the shared RAM block, and it also maps accesses to the peripherals register set into the serial interface engine (SIE) register space. The SkyBlue gasket interface includes registers to control the USB transceiver and voltage regulator.

17.4.1.2.2 Endpoint Buffer Manager

Each endpoint supported by the USB device transmits data to and from buffers stored in the shared buffer memory. The serial interface engine (SIE) uses a table of descriptors, the Buffer Descriptor Table (BDT), which is also stored in the USB RAM to describe the characteristics of each endpoint. The endpoint buffer manager is responsible for mapping requests to access endpoint buffer descriptors into physical addresses within the USB RAM block.

17.4.1.2.3 RAM Arbitration

The arbitration block allows access to the USB RAM block from the SkyBlue gasket block and from the SIE.

17.4.1.3 USB RAM

The USB module includes 256 bytes of high speed RAM, accessible by the USB serial interface engine (SIE) and the CPU. The USB RAM runs at twice the speed of the bus clock to allow interleaved non-blocked access by the CPU and SIE. The USB RAM is used for storage of the buffer descriptor table (BDT) and endpoint buffers. USB RAM that is not allocated for the BDT and endpoint buffers can be used as system memory. If the USB module is not enabled, then the entire USB RAM may be used as unsecured system memory.

17.4.1.4 USB Transceiver (XCVR)

The USB transceiver is electrically compliant to the *Universal Serial Bus Specification 2.0.* This block provides the necessary 2-wire differential NRZI signaling for USB communication. The transceiver is on-chip to provide a cost effective single chip USB peripheral solution.

17.4.1.5 USB On-Chip Voltage Regulator (VREG)

The on-chip 3.3-V regulator provides a stable power source to power the USB internal transceiver and provide for the termination of an internal or external pullup resistor. When the on-chip regulator is enabled, it requires a voltage supply input in the range from 3.9 V to 5.5 V, and the voltage regulator output will be in the range of 3.0 V to 3.6 V.

With a dedicated on-chip USB 3.3-V regulator and a separate power supply for the MCU, the MCU and USB can operate at different voltages (See the USB electricals regarding the USB voltage regulator electrical characteristics). When the on-chip 3.3-V regulator is disabled, a 3.3-V source must be provided through the V_{USB33} pin to power the USB transceiver. In this case, the power supply voltage to the MCU must not fall below the input voltage at the V_{USB33} pin.

The 3.3-V regulator has 3 modes including:

- Active mode — This mode is entered when USB is active. Current requirement is sufficient to power the transceiver and the USBDP pullup resistor.
- Standby — The voltage regulator standby mode is entered automatically when the USB device is in suspend mode. When the USB device is forced into suspend mode by the USB bus, the firmware must configure the MCU for stop3 mode. In standby mode, the requirement is to maintain the USBDP pin voltage at 3.0 V to 3.6 V, with a 900 Ω (worst-case) pullup.
- Power off — This mode is entered anytime when stop2 or stop1 is entered or when the voltage regulator is disabled.

17.4.1.6 USB On-Chip USBDP Pullup Resistor

The pullup resistor on the USBDP line required for full-speed operation by the USB Specification Rev. 2.0 can be internal or external to the MCU, depending on the application requirements. An on-chip pullup resistor, implemented as specified in the USB 2.0 resistor ECN, is optionally available via firmware configuration. Alternatively, this on-chip pullup resistor can be disabled, and the USB module can be configured to use an external pullup resistor for the USBDP line instead. If using an external pullup resistor on the USBDP line, the resistor must comply with the requirements in the USB 2.0 resistor ECN found at http://www.usb.org.

The USBPU bit in the USBCTL0 register can be used to indicate if the pullup resistor is internal or external to the MCU. If USBPU is clear, the internal pullup resistor on USBDP is disabled, and an external USBDP pullup can be used. When using an external USBDP pullup, if the voltage regulator is enabled, the V_{USB33} voltage output can be used with the USBDP pullup. While the use of the internal USBDP pullup resistor is generally recommended, the figure below shows the USBDP pullup resistor configuration for a USB device using an external resistor tied to V_{USB33}.

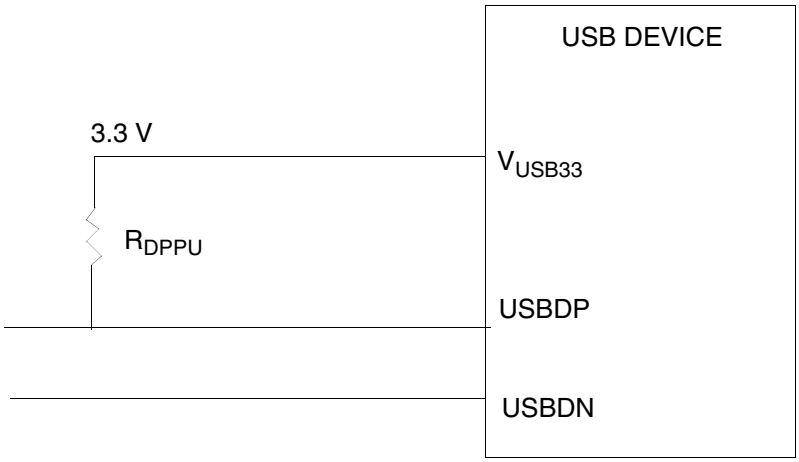


Figure 17-18. USBDP/USBDN Pullup Resistor Configuration for USB module

17.4.1.7 USB Powering and USBDP Pullup Enable Options

The USB module provides a single-chip solution for USB device applications that are self-powered or bus-powered. The USB device needs to know when it has a valid USB connection in order to enable or disable the pullup resistor on the USBDP line. For the USB module on this device, the pullup on USBDP is only applied when a valid VBUS connection is sensed, as required by the USB specification.

In bus-powered applications, system power must be derived from VBUS. Because VBUS is only available when a valid USB connection from host to device is made, the VBUS sensing is built-in, and the USBDP pullup can be enabled accordingly.

With self-powered applications, determining when a valid USB connection is made is different from that of bus-powered applications. In self-powered applications, VBUS sensing must be built into the application. For instance, a KBI pin interrupt can be utilized (if available). When a valid VBUS connection is made, the KBI interrupt can notify the application that a valid USB connection is available, and the internal pullup resistor can be enabled using the USBPU bit. If an external pullup resistor is used instead of the internal one, the VBUS sensing mechanism must be included in the system design.

Table 17-20 summarizes the differences in enabling the USBDP pullup for different USB power modes.

Table 17-20. USBDP Pullup Enable for Different USB Power Modes

Power	USBDP Pullup	Pullup Enable
Bus Power (Built-in VBUS sense)	Internal	Set USBPU bit
	External	Build into application
Self Power (Build VBUS sense into application)	Internal	Set USBPU bit
	External	Build into application

17.4.2 Buffer Descriptor Table (BDT)

To efficiently manage USB endpoint communications, the USB module implements a buffer descriptor table (BDT) comprised of buffer descriptors (BD) in the local USB RAM. The BD entries provide status or control information for a corresponding endpoint. The BD entries also provide an address to the endpoint's buffer. A single BD for an endpoint direction requires 3-bytes. A detailed description of the BDT format is provided in the next sections.

The software API intelligently manages buffers for the USB module by updating the BDT when needed. This allows the USB module to efficiently handle data transmission and reception, while the microcontroller performs communication overhead processing and other function dependent applications.

Because the buffers are shared between the microcontroller and the USB module, a simple semaphore mechanism is used to distinguish who is allowed to update the BDT and buffers in buffer memory. A semaphore bit, the OWN bit, is cleared to 0 when the BD entry is owned by the microcontroller. The microcontroller is allowed read and write access to the BD entry and the data buffer when the OWN bit is 0. When the OWN bit is set to 1, the BD entry and the data buffer are owned by the USB module. The USB module now has full read and write access and the microcontroller must not modify the BD or its corresponding data buffer.

17.4.2.1 Multiple Buffer Descriptor Table Entries for a Single Endpoint

Every endpoint direction requires at least one three-byte Buffer Descriptor entry. Thus, endpoint 0, a bidirectional control endpoint, requires one BDT entry for the IN direction, and one for the OUT direction.

Using two BD entries also allows for double-buffering. Double-buffering BDs allows the USB module to easily transfer data at the maximum throughput provided by the USB module. Double buffering allows the MCU to process one BD while the USB module is processing the other BD.

To facilitate double-buffering, two buffer descriptor (BD) entries are needed for each endpoint direction. One BD entry is the EVEN BD and the other is the ODD BD.

17.4.2.2 Addressing Buffer Descriptor Table Entries

The BDT addressing is hardwired into the module. The BDT occupies the first portion of the USB RAM. To access endpoint data via the USB or MCU, the addressing mechanism of the buffer descriptor table must be understood.

All enabled IN and OUT endpoint BD entries are indexed into the BDT to allow easy access via the USB module or the MCU. The figure below shows the USB RAM organization. The figure shows that the first entries in the USB RAM are dedicated to storage of the BDT entries - i.e. the first 30 bytes of the USB RAM (0x00 to 0x1D) are used to implement the BDT.

Table 17-21. USB RAM Organization

USB RAM Offset	USB RAM Description of Contents	
0x00	BDT	Endpoint 0 IN
		Endpoint 0, OUT
		Endpoint 1
		Endpoint 2
		Endpoint 3
		Endpoint 4
		Endpoint 5, Buffer EVEN
		Endpoint 5, Buffer ODD
		Endpoint 6, Buffer EVEN
0x1D		Endpoint 6, Buffer ODD
0x1E	RESERVED	
0x1F	RESERVED	
0x20	USB RAM available for endpoint buffers	
0xFF		

When the USB module receives a USB token on an enabled endpoint, it interrogates the BDT. The USB module reads the corresponding endpoint BD entry and determines if it owns the BD and corresponding data buffer.

17.4.2.3 Buffer Descriptor Formats

The buffer descriptors (BDs) are groups of registers that provide endpoint buffer control information for the USB module and the MCU. The BDs have different meanings based on who is reading the BD in memory.

The USB module uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- Release Own upon packet completion
- Data toggle synchronization enable
- How much data to be transmitted or received
- Where the buffer resides in the buffer RAM.

The microcontroller uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- The received TOKEN PID

- How much data was transmitted or received.
- Where the buffer resides in buffer memory

The BDT is composed of buffer descriptors (BD) which are used to define and control the actual buffers in the USB RAM space. BDs always occur as a 3-bytes block. See Figure 17-19 for the BD example of Endpoint 0 IN start from USB RAM offset 0x00.

The format for the buffer descriptor is shown in Table 17-22.

Offset		7	6	5	4	3	2	1	0
0x00	R	OWN	DATA0/1	BDTKPID[3]	BDTKPID[3]	BDTKPID[1]	BDTKPID[0]	0	0
	W			0	0	DTS	BDTSTALL		
0x01	R	BC[7:0]							
	W								
0x02	R	EPADR[9:4]							
	W								

Figure 17-19. Buffer Descriptor Example

Table 17-22. Buffer Descriptor Table Fields

Field	Description
OWN	**OWN** — This OWN bit determines who currently owns the buffer. The USB SIE generally writes a 0 to this bit when it has completed a token. The USB module ignores all other fields in the BD when OWN=0. Once the BD has been assigned to the USB module (OWN=1), the MCU must not change it in any way. This byte of the BD must always be the last byte the MCU (firmware) updates when it initializes a BD. Although the hardware will not block the MCU from accessing the BD while owned by the USB SIE, doing so may cause undefined behavior and is generally not recommended. 0 The MCU has exclusive access to the entire BD 1 The USB module has exclusive access to the BD
DATA0/1	**Data Toggle** — This bit defines if a DATA0 field (DATA0/1=0) or a DATA1 (DATA0/1=1) field was transmitted or received. It is unchanged by the USB module. 0 Data 0 packet 1 Data 1 packet
BDTKPID[3:0]	The current token PID is written back to the BD by the USB module when a transfer completes. The values written back are the token PID values from the USB specification: 0x1 for an OUT token, 0x9 for and IN token or 0xd for a SETUP token.
DTS	**Data Toggle Synchronization**— This bit enables data toggle synchronization. 0 No data toggle synchronization is performed. 1 Data toggle synchronization is performed.
BDTSTALL	**BDT Stall** — Setting this bit will cause the USB module to issue a STALL handshake if a token is received by the SIE that would use the BDT in this location. The BDT is not consumed by the SIE (the OWN bit remains and the rest of the BD is unchanged) when the BDTSTALL bit is set. 0 BDT stall is disabled 1 USB will issue a STALL handshake if a token is received by the SIE that would use the BDT in this location

Table 17-22. Buffer Descriptor Table Fields (continued)

Field	Description
BC[7:0]	**Byte Count** — The Byte Count bits represent the 8-bit byte count. The USB module serial interface engine (SIE) will change this field upon the completion of a RX transfer with the byte count of the data received. Note that while USB supports packets as large as 1023 bytes for isochronous endpoints, this module limits packet size to 64 bytes.
EPADR[9:4]	**Endpoint Address**— The endpoint address bits represent the upper 6 bits of the 10-bit buffer address within the module's local USB RAM. Bits [3:0] of EPADR are always zero, therefore the address of the buffer must always start on a 16-byte aligned address within the local RAM. These bits are unchanged by the USB module. This is NOT the address of the memory on the system bus. EPADR is relative to the start of the local USB RAM.

17.4.3 USB Transactions

When the USB module transmits or receives data, it will first compute the BDT address based on the endpoint number, data direction, and which buffer is being used (even or odd), then it will read the BD.

Once the BD has been read, and if the OWN bit equals 1, the serial interface engine (SIE) will transfer the packet data to or receive the packet data from the buffer pointed to by the EPADR field of the BD. When the USB TOKEN is complete, the USB module will update the BDT and change the OWN bit to 0.

The STAT register is updated and the TOKDNE interrupt is set. When the microcontroller processes the TOKDNE interrupt, it reads the status register. This gives the microcontroller all the information it needs to process the endpoint. At this point the microcontroller can allocate a new BD, so additional USB data can be transmitted or received for that endpoint, and it can process the previous BD. Figure 17-20 shows a timeline for how a typical USB token would be processed.

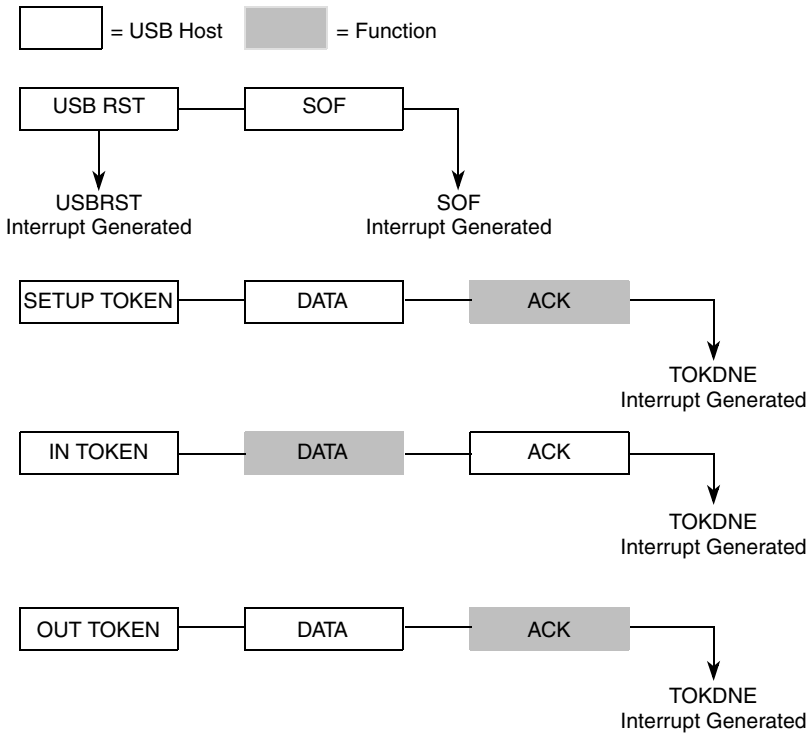


Figure 17-20. USB Packet Flow

The USB has two sources of data overrun error:

- The memory latency to the local USB RAM interface may be too high and cause the receive buffer to overflow. This is predominantly a hardware performance issue, usually caused by transient memory access issues.
- The packet received may be larger than the negotiated MAXPACKET size. This is caused by a software bug.

In the first case, the USB will respond with a NAK or bus timeout (BTO) as appropriate for the class of transaction. The BTOERR bit will be set in the ERRSTAT register. Depending on the values of the INTENB and ERRENB register, USB module may assert an interrupt to notify the CPU of the error. In device mode the BDT is not written back nor is the TOKDNE interrupt triggered because it is assumed that a second attempt will be queued at future time and will succeed.

In the second case of oversized data packets, the USB specification assumes correct software drivers on both sides. The overrun is not due to memory latency but to a lack of space to put the excess data. NAK'ing the packet will likely cause another retransmission of the already oversized packet data. In response to oversized packets, the USB module will still ACK the packet for non-isochronous transfers. The data written to memory is clipped to the MAXPACKET size so as not to corrupt the buffer space. The USB module will assert the BUFERRF bit of the ERRSTAT register (which could trigger an interrupt, as above) and a TOKDNE interrupt fails. The BDTKPID field of the BDT will not be “1111” because the BUFERRF is *not* due to latency. The packet length field written back to the BDT will be the MAXPACKET value to represent the length of the clipped data actually written to memory. From here the software can decide an

appropriate course of action for future transactions — stalling the endpoint, canceling the transfer, disabling the endpoint, etc.

17.4.4 USB Packet Processing

Packet processing for a USB device consists of managing buffers for IN (to the USB Host) and OUT (to the USB device) transactions. Packet processing is further divided into request processing on Endpoint 0, and data packet processing on the data endpoints.

17.4.4.1 USB Data Pipe Processing

Data pipe processing is essentially a buffer management task. The firmware is responsible for managing the shared buffer RAM to ensure that a BD is always ready for the hardware to process (OWN bit = 1).

The device allocates buffers within the shared RAM, sets up the buffer descriptors, and waits for interrupts. On receipt of a TOKDNE interrupt, the firmware reads the STAT register to determine which endpoint is affected, then reads the corresponding BDT entry to determine what to do next.

When processing data packets, firmware is responsible for managing the size of the packet buffers to be in compliance with the USB specification, and the physical limitations of this module. Packet sizes up to 64 bytes are supported on all endpoints. Isochronous endpoints also can only specify packet sizes up to 64 bytes.

Firmware is also responsible for setting the appropriate bits in the BDT. For most applications using bulk packets (control, bulk, and interrupt-type transfers), the firmware will set the DTS, BC and EPADR fields for each BD. For isochronous packets, firmware will set BC and EPADR fields. In all cases, firmware will set the OWN bit to enable the endpoint for data transfers.

17.4.4.2 Request Processing on Endpoint 0

In most cases, commands to the USB device are directed to Endpoint 0. The host uses the “Standard Requests” described in Chapter 9 of the USB specification to enumerate and configure the device. Class drivers or product specific drivers running on the host send class (HID, Mass Storage, Imaging) and vendor specific commands to the device on endpoint 0.

USB requests always follow a specific format:

- Host sends a SETUP token, followed by an 8-byte setup packet, and the device hardware can send a handshake packet.
- If the setup packet specifies a data phase, the host and device may transfer up to 64 Kbytes of data (either IN or OUT, not both).
- The request is terminated by a status phase.

Device firmware monitors the INTSTAT and STAT registers, the endpoint 0 buffer descriptors (BD’s), and the contents of the setup packet to correctly execute the host’s request.

The flow for processing endpoint 0 requests is as follows:

1. Allocate 8-byte buffers for endpoint 0 OUT.

2. Create BDT entries for Endpoint 0 OUT, and set the DTS and OWN bits to 1.
3. Wait for interrupt TOKDNE.
4. Read STAT register.
   — The status register must show Endpoint 0, RX. If it does not, then assert the EPSTALL bit in the endpoint control register.
5. Read Endpoint 0 OUT BD.
   — Verify that the token type is a SETUP token. If it is not, then assert the EPSTALL bit in the endpoint control register.
6. Decode and process the setup packet.
   — If the direction field in the setup packet indicates an OUT transfer, then process the out data phase to receive exactly the number of bytes specified in the wLength field of the setup packet.
   — If the direction field in the setup packet indicates an IN transfer, then process the in data phase to deliver no more than the number of bytes specified in the wLength field. Note that it is common for the host to request more bytes than it needs, expecting the device to only send as much as it needs to.
7. After processing the data phase (if there was one), create a zero-byte status phase transaction.
   — This is accomplished for an OUT data phase (IN status phase) by setting the BC to 0 in the next BD, while also setting OWN=1. For an IN data phase (OUT status phase), the host will send a zero-byte packet to the device.
   — Firmware can verify completion of the data phase by verifying the received token in the BD on receipt of the TOKDNE interrupt. If the data phase was of type IN, then the status phase token will be OUT. If the data phase was of type OUT, then the status phase token will be IN.

17.4.4.3 Endpoint 0 Exception Conditions

The USB includes a number of error checking and recovery mechanisms to ensure reliable data transfer. One such exception occurs when the host sends a SETUP packet to a device, and the host never receives the acknowledge handshake from the device. In this case, the host will retry the SETUP packet.

Endpoint 0 request handlers on the device must be aware of the possibility that after receiving a correct SETUP packet, they could receive another SETUP packet before the data phase actually begins.

17.4.5 Start of Frame Processing

The USB host allocates time in 1.0 ms chunks called “Frames” for the purposes of packet scheduling. The USB host starts each frame with a broadcast token called SOF (start of frame) that includes an 11-bit sequence number. The TOKSOF interrupt is used to notify firmware when an SOF token was received.

Firmware can read the current frame number from the FRMNUML/FRMNUMH registers.

In general, the SOF interrupt is only monitored by devices using isochronous endpoints to help ensure that the device and host remain synchronized.

17.4.6 Suspend/Resume

The USB supports a single low-power mode called suspend. Getting into and out of the suspend state is described in the following sections.

17.4.6.1 Suspend

The USB host can put a single device or the entire bus into the suspend state at any time. The MCU supports suspend mode for low power. Suspend mode will be entered when the USB data lines are in the idle state for more than 3 ms. Entry into suspend mode is announced by the SLEEPF bit in the INTSTAT register.

Per the USB specification, a low-power bus-powered USB device is required to draw less than 500 μA in suspend state. A high-power device that supports remote wakeup and has its remote wake-up feature enabled by the host can draw up to 2.5 mA of current. After the initial 3-ms idle, the USB device will reach this state within 7 ms. This low-current requirement means that firmware is responsible for entering stop3 mode once the SLEEPF flag has been set and before the USB module has been placed in the suspend state.

On receipt of resume signaling from the USB, the module can generate an asynchronous interrupt to the MCU which brings the device out of stop mode and wakes up the clocks. Setting the USBRESMEN bit in the USBCTL0 register immediately after the SLEEPF bit is set enables this asynchronous notification feature. The USB resume signaling will then cause the LPRESF bit to be set, indicating a low-power SUSPEND resume, which will wake the CPU from stop3 mode.

During normal operation, while the host is sending SOF packets, the USB module will not enter suspend mode.

17.4.6.2 Resume

There are three ways to get out of the suspend state. When the USB module is in suspend state, the resume detection is active even if all the clocks are disabled and the MCU is in stop3 mode. The MCU can be activated from the suspend state by normal bus activity, a USB reset signal, or upstream resume (remote wakeup).

17.4.6.2.1 Host Initiated Resume

The host signals a resume from suspend by initiating resume signaling (K state) for at least 20 ms followed by a standard low-speed EOP signal. This 20 ms ensures that all devices in the USB network are awakened. After resuming the bus, the host must begin sending bus traffic within 3 ms to prevent the device from re-entering suspend mode.

Depending on the power mode the device is in while suspended, the notification for a host initiated resume will be different:

- Run mode - RESUME must be set after SLEEPF becomes set to enable the RESUMEF interrupt. Then, upon resume signaling, the RESUMEF interrupt will trigger after a K-state has been observed on the USBDP/USBDN lines for 2.5 μs.
- Stop3 mode - USBRESMEN must be set after SLEEPF becomes set to arm the LPRESF bit. Then, upon a K-state on the bus while the device is in stop3 mode, the LPRESF bit will be set, indicating

a resume from low-power suspend. This will trigger an asynchronous interrupt to wake the CPU from stop3 mode and resume clocks to the USB module.

NOTE

As a precaution, after LPRESF is set, firmware must check the state of the USB bus to see if the K-state was a result of a transient event and not a true host-initiated resume. If this is the case, then the device can drop back into stop3 if necessary. To do this, the RESUME interrupt can be enabled in conjunction with the USBRESMEN feature. Then, after LPRESF is set, and a K-state is still detected approximately 2.5 µs after clocks have restarted, firmware can check that the RESUMEF interrupt has triggered, indicating resume signaling from the host.

17.4.6.2.2 USB Reset Signaling

Reset can wake a device from the suspend state.

17.4.6.2.3 Remote Wakeup

The USB device can send a resume event to the host by writing to the CRESUME bit. Firmware must first set the bit for the time period required by the USB Specification Rev. 2.0 (Section 7.1.7.7) and then clear it to 0.

17.4.7 Resets

The module supports multiple types of resets. The first is a bus reset generated by the USB Host, the second is a module reset generated by the MCU.

17.4.7.1 USB Bus Reset

At any time, the USB host may issue a reset to one or all of the devices attached to the bus. A USB reset is defined as a period of single ended zero (SE0) on the cable for greater than 2.5 µs. When the device detects reset signaling, it resets itself to the unconfigured state, and sets its USB address zero. The USB host uses reset signaling to force one or all connected devices into a known state prior to commencing enumeration.

The USB module responds to reset signaling by asserting the USBRST interrupt in the INTSTAT register. Software is required to service this interrupt to ensure correct operation of the USB.

17.4.7.2 USB Module Reset

USB module resets are initiated on-chip. During a module reset, the USB module is configured in the default mode. The USB module can also be forced into its reset state by setting the USBRESET bit in the USBCTL0 register. The default mode includes the following settings:

- Interrupts masked.
- USB clock enabled
- USB voltage regulator disabled

- USB transceiver disabled
- USBDP pullup disabled
- Endpoints disabled
- USB address register set to zero

17.4.8 Interrupts

Interrupts from the INTSTAT register signify events which occur during normal operation — USB start of frame tokens (TOKSOF), packet completion (TOKDNE), USB bus reset (USBRST), endpoint errors (ERROR), suspend and resume (SLEEP and RESUME), and endpoint stalled (STALL).

The ERRSTAT interrupts carry information about specific types of errors, which is needed on an application specific basis. Using ERRSTAT, an application can determine exactly why a packet transfer failed — due to CRC error, PID check error and so on.

Both registers are maskable via the INTENB and ERRENB registers. The INTSTAT and ERRSTAT are used to signal interrupts in a two-level structure. Unmasked interrupts from the ERRSTAT register are reported in the INTSTAT register.

Note that the interrupt registers work in concert with the STAT register. On receipt of an INTSTAT interrupt, software can check the STAT register and determine which BDT entry was affected by the transaction.

Chapter 18
Development Support

18.1 Introduction

Development support systems in the HCS08 include the background debug controller (BDC) and the on-chip debug module (DBG). The BDC provides a single-wire debug interface to the target MCU that provides a convenient interface for programming the on-chip flash and other nonvolatile memories. The BDC is also the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modify, breakpoints, and single instruction trace commands.

In the HCS08 family, address and data bus signals are not available on external pins (not even in test modes). Debug is done through commands fed into the target MCU via the single-wire background debug interface. The debug module provides a means to selectively trigger and capture bus information so an external development system can reconstruct what happened inside the MCU on a cycle-by-cycle basis without having external access to the address and data signals.

The alternate BDC clock source for MC9S08JM60 Series is the MCGLCLK. See Chapter 12, “Multi-Purpose Clock Generator (S08MCGV1),” for more information about MCGLCLK and how to select clock sources.

18.1.1 Features

Features of the BDC module include:

- Single pin for mode selection and background communications
- BDC registers are not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from stop or wait modes
- One hardware address breakpoint built into BDC
- Oscillator runs in stop mode, if BDC enabled
- COP watchdog disabled while in active background mode

Features of the ICE system include:

- Two trigger comparators: Two address + read/write (R/W) or one full address + data + R/W
- Flexible 8-word by 16-bit FIFO (first-in, first-out) buffer for capture information:
  - Change-of-flow addresses or
  - Event-only data
- Two types of breakpoints:
  - Tag breakpoints for instruction opcodes
  - Force breakpoints for any address access
- Nine trigger modes:
  - Basic: A-only, A OR B
  - Sequence: A then B
  - Full: A AND B data, A AND NOT B data
  - Event (store data): Event-only B, A then event-only B
  - Range: Inside range ($A \leq address \leq B$), outside range ($address < A$ or $address > B$)

18.2 Background Debug Controller (BDC)

All MCUs in the HCS08 family contain a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this system does not interfere with normal application resources. It does not use any user memory or locations in the memory map and does not share any on-chip peripherals.

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.

- Non-intrusive commands can be executed at any time even while the user's program is running. Non-intrusive commands allow a user to read or write MCU memory locations or access status and control registers within the background debug controller.

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, $\overline{\text{RESET}}$, and sometimes V_{DD}. An open-drain connection to reset allows the host to force a target system reset, which is useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes V_{DD} can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

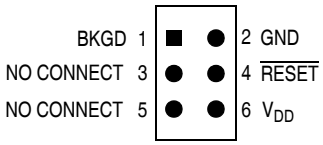


Figure 18-1. BDM Tool Connector

18.2.1 BKGD Pin Description

BKGD is the single-wire background debug interface pin. The primary function of this pin is for bidirectional serial communication of active background mode commands and data. During reset, this pin is used to select between starting in active background mode or starting the user's application program. This pin is also used to request a timed sync response pulse to allow a host development tool to determine the correct clock frequency for background debug serial communications.

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers. This protocol assumes the host knows the communication clock rate that is determined by the target BDC clock rate. All communication is initiated and controlled by the host that drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit first (MSB first). For a detailed description of the communications protocol, refer to Section 18.2.2, "Communication Details."

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed sync response signal from which the host can determine the correct communication speed.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively driven speedup pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to Section 18.2.2, "Communication Details," for more detail.

When no debugger pod is connected to the 6-pin BDM interface connector, the internal pullup on BKGD chooses normal operating mode. When a debug pod is connected to BKGD it is possible to force the MCU into active background mode after reset. The specific conditions for forcing active background depend upon the HCS08 derivative (refer to the introduction to this Development Support section). It is not necessary to reset the target MCU to communicate with it through the background debug interface.

18.2.2 Communication Details

The BDC serial interface requires the external controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 BDC clock cycles per bit (nominal speed). The interface times out if 512 BDC clock cycles occur between falling edges from the host. Any BDC command that was in progress when this timeout occurs is aborted without affecting the memory or operating mode of the target MCU system.

The custom serial protocol requires the debug pod to know the target BDC communication clock speed.

The clock switch (CLKSW) control bit in the BDC status and control register allows the user to select the BDC clock source. The BDC clock source can either be the bus or the alternate BDC clock source.

The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

Figure 18-2 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target HCS08 MCU. The host is asynchronous to the target so there is a 0-to-1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target BDC clock cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Because the target does not drive the BKGD pin during the host-to-target transmission period, there is no need to treat the line as an open-drain signal during this period.

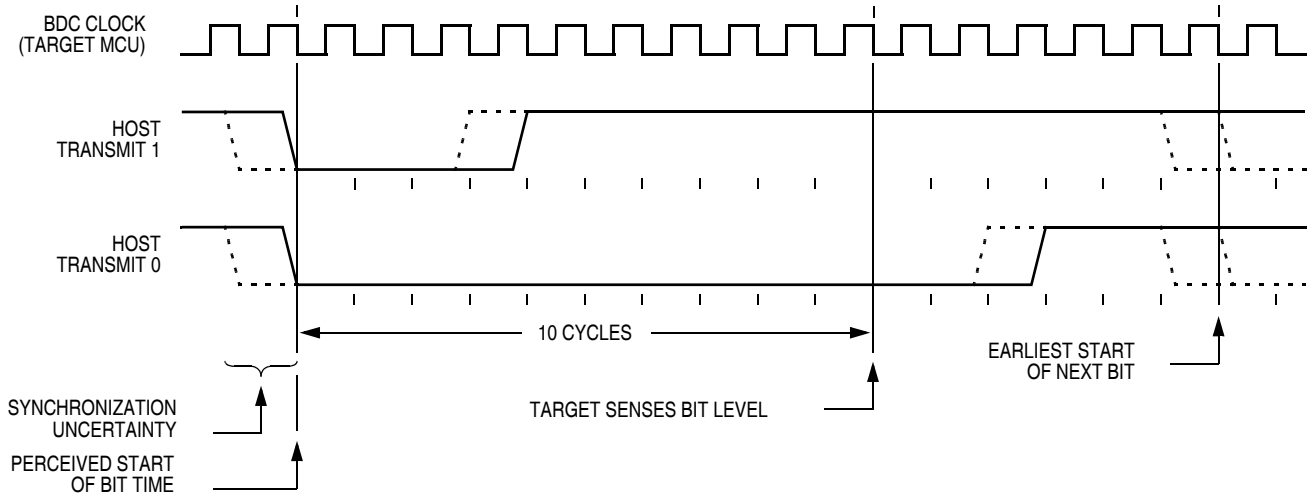


Figure 18-2. BDC Host-to-Target Serial Bit Timing

Figure 18-3 shows the host receiving a logic 1 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

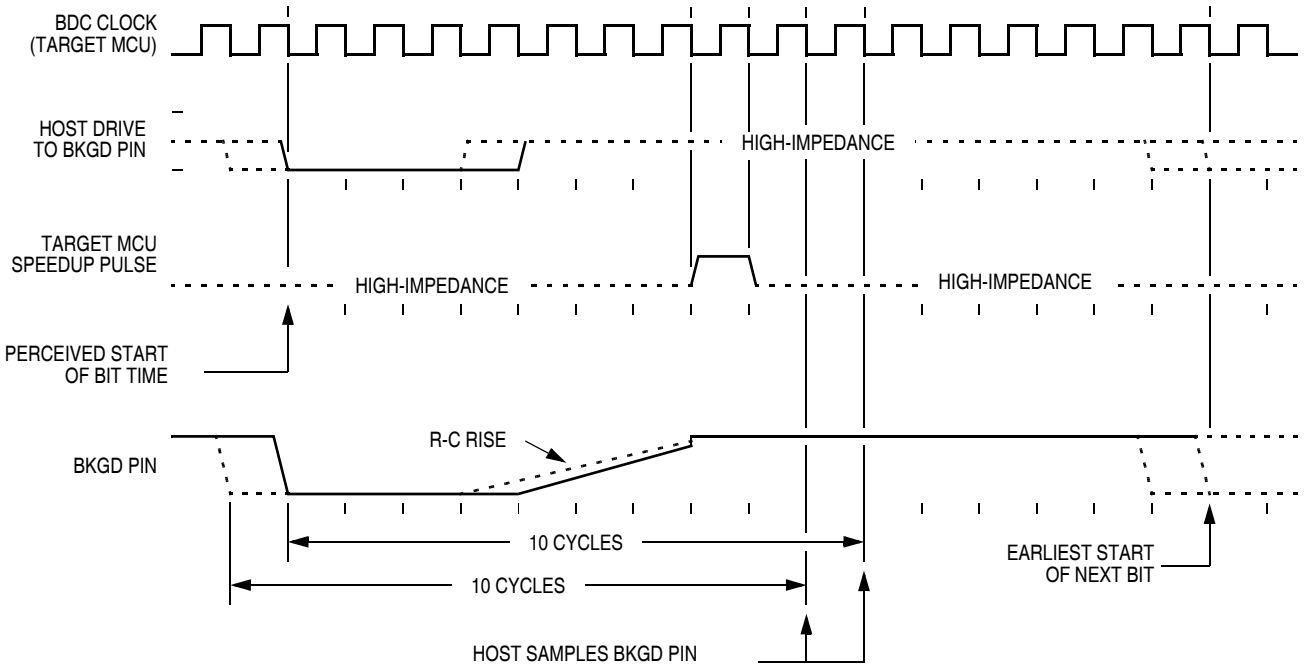


Figure 18-3. BDC Target-to-Host Serial Bit Timing (Logic 1)

Figure 18-4 shows the host receiving a logic 0 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time but the target HCS08 finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.

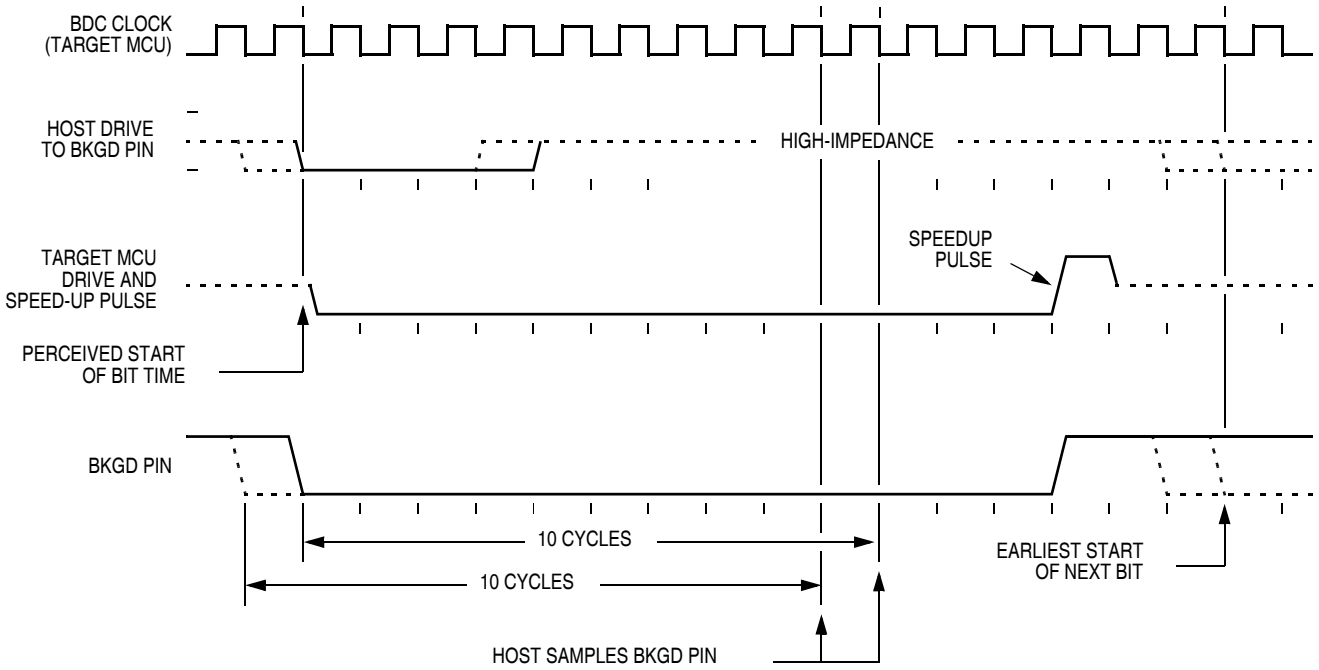


Figure 18-4. BDM Target-to-Host Serial Bit Timing (Logic 0)

18.2.3 BDC Commands

BDC commands are sent serially from a host computer to the BKGD pin of the target HCS08 MCU. All commands and data are sent MSB-first using a custom BDC communications protocol. Active background mode commands require that the target MCU is currently in the active background mode while non-intrusive commands may be issued at any time whether the target MCU is in active background mode or running a user application program.

Table 18-1 shows all HCS08 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

Coding Structure Nomenclature

This nomenclature is used in Table 18-1 to describe the coding structure of the BDC commands.

		Commands begin with an 8-bit hexadecimal command code in the host-to-target direction (most significant bit first)
/	=	separates parts of the command
d	=	delay 16 target BDC clock cycles
AAAA	=	a 16-bit address in the host-to-target direction
RD	=	8 bits of read data in the target-to-host direction
WD	=	8 bits of write data in the host-to-target direction
RD16	=	16 bits of read data in the target-to-host direction
WD16	=	16 bits of write data in the host-to-target direction
SS	=	the contents of BDCSCR in the target-to-host direction (STATUS)
CC	=	8 bits of write data for BDCSCR in the host-to-target direction (CONTROL)
RBKP	=	16 bits of read data in the target-to-host direction (from BDCBKPT breakpoint register)
WBKP	=	16 bits of write data in the host-to-target direction (for BDCBKPT breakpoint register)

Table 18-1. BDC Command Summary

Command Mnemonic	Active BDM/ Non-intrusive	Coding Structure	Description
SYNC	Non-intrusive	n/a[1]	Request a timed reference pulse to determine target BDC communication speed
ACK_ENABLE	Non-intrusive	D5/d	Enable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D.
ACK_DISABLE	Non-intrusive	D6/d	Disable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D.
BACKGROUND	Non-intrusive	90/d	Enter active background mode if enabled (ignore if ENBDM bit equals 0)
READ_STATUS	Non-intrusive	E4/SS	Read BDC status from BDCSCR
WRITE_CONTROL	Non-intrusive	C4/CC	Write BDC controls in BDCSCR
READ_BYTE	Non-intrusive	E0/AAAA/d/RD	Read a byte from target memory
READ_BYTE_WS	Non-intrusive	E1/AAAA/d/SS/RD	Read a byte and report status
READ_LAST	Non-intrusive	E8/SS/RD	Re-read byte from address just read and report status
WRITE_BYTE	Non-intrusive	C0/AAAA/WD/d	Write a byte to target memory
WRITE_BYTE_WS	Non-intrusive	C1/AAAA/WD/d/SS	Write a byte and report status
READ_BKPT	Non-intrusive	E2/RBKP	Read BDCBKPT breakpoint register
WRITE_BKPT	Non-intrusive	C2/WBKP	Write BDCBKPT breakpoint register
GO	Active BDM	08/d	Go to execute the user application program starting at the address currently in the PC
TRACE1	Active BDM	10/d	Trace 1 user instruction at the address in the PC, then return to active background mode
TAGGO	Active BDM	18/d	Same as GO but enable external tagging (HCS08 devices have no external tagging pin)
READ_A	Active BDM	68/d/RD	Read accumulator (A)
READ_CCR	Active BDM	69/d/RD	Read condition code register (CCR)
READ_PC	Active BDM	6B/d/RD16	Read program counter (PC)
READ_HX	Active BDM	6C/d/RD16	Read H and X register pair (H:X)
READ_SP	Active BDM	6F/d/RD16	Read stack pointer (SP)
READ_NEXT	Active BDM	70/d/RD	Increment H:X by one then read memory byte located at H:X
READ_NEXT_WS	Active BDM	71/d/SS/RD	Increment H:X by one then read memory byte located at H:X. Report status and data.
WRITE_A	Active BDM	48/WD/d	Write accumulator (A)
WRITE_CCR	Active BDM	49/WD/d	Write condition code register (CCR)
WRITE_PC	Active BDM	4B/WD16/d	Write program counter (PC)
WRITE_HX	Active BDM	4C/WD16/d	Write H and X register pair (H:X)
WRITE_SP	Active BDM	4F/WD16/d	Write stack pointer (SP)
WRITE_NEXT	Active BDM	50/WD/d	Increment H:X by one, then write memory byte located at H:X
WRITE_NEXT_WS	Active BDM	51/WD/d/SS	Increment H:X by one, then write memory byte located at H:X. Also report status.

[1] The SYNC command is a special operation that does not have a command code.

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct communications speed to use for BDC communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

- Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (The slowest clock is normally the reference oscillator/64 or the self-clocked rate/64.)
- Drives BKGD high for a brief speedup pulse to get a fast rise time (This speedup pulse is typically one cycle of the fastest clock in the system.)
- Removes all drive to the BKGD pin so it reverts to high impedance
- Monitors the BKGD pin for the sync response pulse

The target, upon detecting the SYNC request from the host (which is a much longer low time than would ever occur during normal BDC communications):

- Waits for BKGD to return to a logic high
- Delays 16 cycles to allow the host to stop driving the high speedup pulse
- Drives BKGD low for 128 BDC clock cycles
- Drives a 1-cycle high speedup pulse to force a fast rise time on BKGD
- Removes all drive to the BKGD pin so it reverts to high impedance

The host measures the low time of this 128-cycle sync response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the communication protocol can easily tolerate speed errors of several percent.

18.2.4 BDC Hardware Breakpoint

The BDC includes one relatively simple hardware breakpoint that compares the CPU address bus to a 16-bit match value in the BDCBKPT register. This breakpoint can generate a forced breakpoint or a tagged breakpoint. A forced breakpoint causes the CPU to enter active background mode at the first instruction boundary following any access to the breakpoint address. The tagged breakpoint causes the instruction opcode at the breakpoint address to be tagged so that the CPU will enter active background mode rather than executing that instruction if and when it reaches the end of the instruction queue. This implies that tagged breakpoints can only be placed at the address of an instruction opcode while forced breakpoints can be set at any address.

The breakpoint enable (BKPTEN) control bit in the BDC status and control register (BDCSCR) is used to enable the breakpoint logic (BKPTEN = 1). When BKPTEN = 0, its default value after reset, the breakpoint logic is disabled and no BDC breakpoints are requested regardless of the values in other BDC breakpoint registers and control bits. The force/tag select (FTS) control bit in BDCSCR is used to select forced (FTS = 1) or tagged (FTS = 0) type breakpoints.

The on-chip debug module (DBG) includes circuitry for two additional hardware breakpoints that are more flexible than the simple breakpoint in the BDC module.

18.3 On-Chip Debug System (DBG)

Because HCS08 devices do not have external address and data buses, the most important functions of an in-circuit emulator have been built onto the chip with the MCU. The debug system consists of an 8-stage FIFO that can store address or data bus information, and a flexible trigger system to decide when to capture bus information and what information to capture. The system relies on the single-wire background debug system to access debug control registers and to read results out of the eight stage FIFO.

The debug module includes control and status registers that are accessible in the user's memory map. These registers are located in the high register space to avoid using valuable direct page memory space.

Most of the debug module's functions are used during development, and user programs rarely access any of the control and status registers for the debug module. The one exception is that the debug system can provide the means to implement a form of ROM patching. This topic is discussed in greater detail in Section 18.3.6, "Hardware Breakpoints."

18.3.1 Comparators A and B

Two 16-bit comparators (A and B) can optionally be qualified with the R/W signal and an opcode tracking circuit. Separate control bits allow you to ignore R/W for each comparator. The opcode tracking circuitry optionally allows you to specify that a trigger will occur only if the opcode at the specified address is actually executed as opposed to only being read from memory into the instruction queue. The comparators are also capable of magnitude comparisons to support the inside range and outside range trigger modes. Comparators are disabled temporarily during all BDC accesses.

The A comparator is always associated with the 16-bit CPU address. The B comparator compares to the CPU address or the 8-bit CPU data bus, depending on the trigger mode selected. Because the CPU data bus is separated into a read data bus and a write data bus, the RWAEN and RWA control bits have an additional purpose, in full address plus data comparisons they are used to decide which of these buses to use in the comparator B data bus comparisons. If RWAEN = 1 (enabled) and RWA = 0 (write), the CPU's write data bus is used. Otherwise, the CPU's read data bus is used.

The currently selected trigger mode determines what the debugger logic does when a comparator detects a qualified match condition. A match can cause:

- Generation of a breakpoint to the CPU
- Storage of data bus values into the FIFO
- Starting to store change-of-flow addresses into the FIFO (begin type trace)
- Stopping the storage of change-of-flow addresses into the FIFO (end type trace)

18.3.2 Bus Capture Information and FIFO Operation

The usual way to use the FIFO is to setup the trigger mode and other control options, then arm the debugger. When the FIFO has filled or the debugger has stopped storing data into the FIFO, you would read the information out of it in the order it was stored into the FIFO. Status bits indicate the number of words of valid information that are in the FIFO as data is stored into it. If a trace run is manually halted by writing 0 to ARM before the FIFO is full (CNT = 1:0:0:0), the information is shifted by one position and

the host must perform ((8 – CNT) – 1) dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see Section 18.3.5, “Trigger Modes”), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

In trigger modes where the FIFO is storing change-of-flow addresses, there is a delay between CPU addresses and the input side of the FIFO. Because of this delay, if the trigger event itself is a change-of-flow address or a change-of-flow address appears during the next two bus cycles after a trigger event starts the FIFO, it will not be saved into the FIFO. In the case of an end-trace, if the trigger event is a change-of-flow, it will be saved as the last change-of-flow entry for that debug run.

The FIFO can also be used to generate a profile of executed instruction addresses when the debugger is not armed. When ARM = 0, reading DBGFL causes the address of the most-recently fetched opcode to be saved in the FIFO. To use the profiling feature, a host debugger would read addresses out of the FIFO by reading DBGFH then DBGFL at regular periodic intervals. The first eight values would be discarded because they correspond to the eight DBGFL reads needed to initially fill the FIFO. Additional periodic reads of DBGFH and DBGFL return delayed information about executed instructions so the host debugger can develop a profile of executed instruction addresses.

18.3.3 Change-of-Flow Information

To minimize the amount of information stored in the FIFO, only information related to instructions that cause a change to the normal sequential execution of instructions is stored. With knowledge of the source and object code program stored in the target system, an external debugger system can reconstruct the path of execution through many instructions from the change-of-flow information stored in the FIFO.

For conditional branch instructions where the branch is taken (branch condition was true), the source address is stored (the address of the conditional branch opcode). Because BRA and BRN instructions are not conditional, these events do not cause change-of-flow information to be stored in the FIFO.

Indirect JMP and JSR instructions use the current contents of the H:X index register pair to determine the destination address, so the debug system stores the run-time destination address for any indirect JMP or JSR. For interrupts, RTI, or RTS, the destination address is stored in the FIFO as change-of-flow information.

18.3.4 Tag vs. Force Breakpoints and Triggers

Tagging is a term that refers to identifying an instruction opcode as it is fetched into the instruction queue, but not taking any other action until and unless that instruction is actually executed by the CPU. This distinction is important because any change-of-flow from a jump, branch, subroutine call, or interrupt causes some instructions that have been fetched into the instruction queue to be thrown away without being executed.

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the DBGT register is set to select tag-type operation, the output from comparator A or B is qualified by a block of logic in the debug module that tracks opcodes and only produces a trigger to the debugger if the opcode at the compare address is actually executed. There is separate opcode tracking logic for each comparator so more than one compare event can be tracked through the instruction queue at a time.

18.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGT register selects one of nine trigger modes. When TRGSEL = 1 in the DBGT register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGT chooses whether the FIFO begins storing data when the qualified trigger is detected (begin trace), or the FIFO stores data in a circular fashion from the time it is armed until the qualified trigger is detected (end trigger).

A debug run is started by writing a 1 to the ARM bit in the DBGC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGS. A begin-trace debug run ends when the FIFO gets full. An end-trace run ends when the selected trigger event occurs. Any debug run can be stopped manually by writing a 0 to ARM or DBGEN in DBGC.

In all trigger modes except event-only modes, the FIFO stores change-of-flow addresses. In event-only trigger modes, the FIFO stores data in the low-order eight bits of the FIFO.

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following trigger mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

A-Only — Trigger when the address matches the value in comparator A

A OR B — Trigger when the address matches either the value in comparator A or the value in comparator B

A Then B — Trigger when the address matches the value in comparator B but only after the address for another cycle matched the value in comparator A. There can be any number of cycles after the A match and before the B match.

A AND B Data (Full Mode) — This is called a full mode because address, data, and R/W (optionally) must match within the same bus cycle to cause a trigger event. Comparator A checks address, the low byte of comparator B checks data, and R/W is checked against RWA if RWAEN = 1. The high-order half of comparator B is not used.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

A AND NOT B Data (Full Mode) — Address must match comparator A, data must not match the low half of comparator B, and R/W must match RWA if RWAEN = 1. All three conditions must be met within the same bus cycle to cause a trigger.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

Event-Only B (Store Data) — Trigger events occur each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

A Then Event-Only B (Store Data) — After the address has matched the value in comparator A, a trigger event occurs each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

Inside Range (A ≤ Address ≤ B) — A trigger occurs when the address is greater than or equal to the value in comparator A and less than or equal to the value in comparator B at the same time.

Outside Range (Address < A or Address > B) — A trigger occurs when the address is either less than the value in comparator A or greater than the value in comparator B.

18.3.6 Hardware Breakpoints

The BRKEN control bit in the DBGC register may be set to 1 to allow any of the trigger conditions described in Section 18.3.5, “Trigger Modes,” to be used to generate a hardware breakpoint request to the CPU. TAG in DBGC controls whether the breakpoint request will be treated as a tag-type breakpoint or a force-type breakpoint. A tag breakpoint causes the current opcode to be marked as it enters the instruction queue. If a tagged opcode reaches the end of the pipe, the CPU executes a BGND instruction to go to active background mode rather than executing the tagged opcode. A force-type breakpoint causes the CPU to finish the current instruction and then go to active background mode.

If the background mode has not been enabled (ENBDM = 1) by a serial WRITE_CONTROL command through the BKGD pin, the CPU will execute an SWI instruction instead of going to active background mode.

18.4 Register Definition

This section contains the descriptions of the BDC and DBG registers and control bits.

Refer to the high-page register summary in the device overview chapter of this data sheet for the absolute address assignments for all DBG registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

18.4.1 BDC Registers and Control Bits

The BDC has two registers:

- The BDC status and control register (BDCSCR) is an 8-bit register containing control and status bits for the background debug controller.
- The BDC breakpoint match register (BDCBKPT) holds a 16-bit breakpoint match address.

These registers are accessed with dedicated serial BDC commands and are not located in the memory space of the target MCU (so they do not have addresses and cannot be accessed by user programs).

Some of the bits in the BDCSCR have write limitations; otherwise, these registers may be read or written at any time. For example, the ENBDM control bit may not be written while the MCU is in active background mode. (This prevents the ambiguous condition of the control bit forbidding active background mode while the MCU is already in active background mode.) Also, the four status bits (BDMACT, WS, WSF, and DVF) are read-only status indicators and can never be written by the WRITE_CONTROL serial BDC command. The clock switch (CLKSW) control bit may be read or written at any time.

18.4.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ_STATUS and WRITE_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

	7	6	5	4	3	2	1	0
R	ENBDM	BDMACT	BKPTEN	FTS	CLKSW	WS	WSF	DVF
W								
Normal Reset	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	1	0	0	0

= Unimplemented or Reserved

Figure 18-5. BDC Status and Control Register (BDCSCR)

Table 18-2. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	**Enable BDM (Permit Active Background Mode)** — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it. 0 BDM cannot be made active (non-intrusive commands still allowed) 1 BDM can be made active to allow active background mode commands
6 BDMACT	**Background Mode Active Status** — This is a read-only status bit. 0 BDM not active (user application program running) 1 BDM active and waiting for serial commands
5 BKPTEN	**BDC Breakpoint Enable** — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored. 0 BDC breakpoint disabled 1 BDC breakpoint enabled
4 FTS	**Force/Tag Select** — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters active background mode rather than executing the tagged opcode. 0 Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction 1 Breakpoint match forces active background mode at next instruction boundary (address need not be an opcode)
3 CLKSW	**Select Source for BDC Communications Clock** — CLKSW defaults to 0, which selects the alternate BDC clock source. 0 Alternate BDC clock source 1 MCU bus clock

Table 18-2. BDCSCR Register Field Descriptions (continued)

Field	Description
2 WS	**Wait or Stop Status** — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands. 0 Target CPU is running user application code or in active background mode (was not in wait or stop mode when background became active) 1 Target CPU is in wait or stop mode, or a BACKGROUND command was used to change from wait or stop to active background mode
1 WSF	**Wait or Stop Failure Status** — This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.) 0 Memory access did not conflict with a wait or stop instruction 1 Memory access command failed because the CPU entered wait or stop mode
0 DVF	**Data Valid Failure Status** — This status bit is not used in the MC9S08JM60 Series because it does not have any slow access memory. 0 Memory access did not conflict with a slow memory access 1 Memory access command failed because CPU was not finished with a slow memory access

18.4.1.2 BDC Breakpoint Match Register (BDCBKPT)

This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ_BKPT and WRITE_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU. Breakpoints are normally set while the target MCU is in active background mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to Section 18.2.4, “BDC Hardware Breakpoint.”

18.4.2 System Background Debug Force Reset Register (SBDFR)

This register contains a single write-only control bit. A serial background mode command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return 0x00.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								BDFR[1]
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

[1] BDFR is writable only through serial background mode debug commands, not from user programs.

Figure 18-6. System Background Debug Force Reset Register (SBDFR)

Table 18-3. SBDFR Register Field Description

Field	Description
0 BDFR	**Background Debug Force Reset** — A serial active background mode command such as WRITE_BYTE allows an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program.

18.4.3 DBG Registers and Control Bits

The debug module includes nine bytes of register space for three 16-bit registers and three 8-bit control and status registers. These registers are located in the high register space of the normal memory map so they are accessible to normal application programs. These registers are rarely if ever accessed by normal user application programs with the possible exception of a ROM patching mechanism that uses the breakpoint logic.

18.4.3.1 Debug Comparator A High Register (DBGCAH)

This register contains compare value bits for the high-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

18.4.3.2 Debug Comparator A Low Register (DBGCAL)

This register contains compare value bits for the low-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

18.4.3.3 Debug Comparator B High Register (DBGCBH)

This register contains compare value bits for the high-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

18.4.3.4 Debug Comparator B Low Register (DBGCBL)

This register contains compare value bits for the low-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

18.4.3.5 Debug FIFO High Register (DBGFH)

This register provides read-only access to the high-order eight bits of the FIFO. Writes to this register have no meaning or effect. In the event-only trigger modes, the FIFO only stores data into the low-order byte of each FIFO word, so this register is not used and will read 0x00.

Reading DBGFH does not cause the FIFO to shift to the next word. When reading 16-bit words out of the FIFO, read DBGFH before reading DBGFL because reading DBGFL causes the FIFO to advance to the next word of information.

18.4.3.6 Debug FIFO Low Register (DBGFL)

This register provides read-only access to the low-order eight bits of the FIFO. Writes to this register have no meaning or effect.

Reading DBGFL causes the FIFO to shift to the next available word of information. When the debug module is operating in event-only modes, only 8-bit data is stored into the FIFO (high-order half of each FIFO word is unused). When reading 8-bit words out of the FIFO, simply read DBGFL repeatedly to get successive bytes of data from the FIFO. It isn't necessary to read DBGFH in this case.

Do not attempt to read data from the FIFO while it is still armed (after arming but before the FIFO is filled or ARMF is cleared) because the FIFO is prevented from advancing during reads of DBGFL. This can interfere with normal sequencing of reads from the FIFO.

Reading DBGFL while the debugger is not armed causes the address of the most-recently fetched opcode to be stored to the last location in the FIFO. By reading DBGFH then DBGFL periodically, external host software can develop a profile of program execution. After eight reads from the FIFO, the ninth read will return the information that was stored as a result of the first read. To use the profiling feature, read the FIFO eight times without using the data to prime the sequence and then begin using the data to get a delayed picture of what addresses were being executed. The information stored into the FIFO on reads of DBGFL (while the FIFO is not armed) is the address of the most-recently fetched opcode.

18.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

	7	6	5	4	3	2	1	0
R W	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
Reset	0	0	0	0	0	0	0	0

Figure 18-7. Debug Control Register (DBGC)

Table 18-4. DBGC Register Field Descriptions

Field	Description
7 DBGEN	**Debug Module Enable** — Used to enable the debug module. DBGEN cannot be set to 1 if the MCU is secure. 0 DBG disabled 1 DBG enabled
6 ARM	**Arm Control** — Controls whether the debugger is comparing and storing information in the FIFO. A write is used to set this bit (and ARMF) and completion of a debug run automatically clears it. Any debug run can be manually stopped by writing 0 to ARM or to DBGEN. 0 Debugger not armed 1 Debugger armed
5 TAG	**Tag/Force Select** — Controls whether break requests to the CPU will be tag or force type requests. If BRKEN = 0, this bit has no meaning or effect. 0 CPU breaks requested as force type requests 1 CPU breaks requested as tag type requests
4 BRKEN	**Break Enable** — Controls whether a trigger event will generate a break request to the CPU. Trigger events can cause information to be stored in the FIFO without generating a break request to the CPU. For an end trace, CPU break requests are issued to the CPU when the comparator(s) and R/W meet the trigger requirements. For a begin trace, CPU break requests are issued when the FIFO becomes full. TRGSEL does not affect the timing of CPU break requests. 0 CPU break requests not enabled 1 Triggers cause a break request to the CPU
3 RWA	**R/W Comparison Value for Comparator A** — When RWAEN = 1, this bit determines whether a read or a write access qualifies comparator A. When RWAEN = 0, RWA and the R/W signal do not affect comparator A. 0 Comparator A can only match on a write cycle 1 Comparator A can only match on a read cycle
2 RWAEN	**Enable R/W for Comparator A** — Controls whether the level of R/W is considered for a comparator A match. 0 R/W is not used in comparison A 1 R/W is used in comparison A
1 RWB	**R/W Comparison Value for Comparator B** — When RWBEN = 1, this bit determines whether a read or a write access qualifies comparator B. When RWBEN = 0, RWB and the R/W signal do not affect comparator B. 0 Comparator B can match only on a write cycle 1 Comparator B can match only on a read cycle
0 RWBEN	**Enable R/W for Comparator B** — Controls whether the level of R/W is considered for a comparator B match. 0 R/W is not used in comparison B 1 R/W is used in comparison B

18.4.3.8 Debug Trigger Register (DBGT)

This register can be read any time, but may be written only if ARM = 0, except bits 4 and 5 are hard-wired to 0s.

	7	6	5	4	3	2	1	0
R	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 18-8. Debug Trigger Register (DBGT)

Table 18-5. DBGT Register Field Descriptions

Field	Description
7 TRGSEL	**Trigger Type** — Controls whether the match outputs from comparators A and B are qualified with the opcode tracking logic in the debug module. If TRGSEL is set, a match signal from comparator A or B must propagate through the opcode tracking logic and a trigger event is only signalled to the FIFO logic if the opcode at the match address is actually executed. 0 Trigger on access to compare address (force) 1 Trigger if opcode at compare address is executed (tag)
6 BEGIN	**Begin/End Trigger Select** — Controls whether the FIFO starts filling at a trigger or fills in a circular manner until a trigger ends the capture of information. In event-only trigger modes, this bit is ignored and all debug runs are assumed to be begin traces. 0 Data stored in FIFO until trigger (end trace) 1 Trigger initiates data storage (begin trace)
3:0 TRG[3:0]	**Select Trigger Mode** — Selects one of nine triggering modes, as described below. 0000 A-only 0001 A OR B 0010 A Then B 0011 Event-only B (store data) 0100 A then event-only B (store data) 0101 A AND B data (full mode) 0110 A AND NOT B data (full mode) 0111 Inside range: A ≤ address ≤ B 1000 Outside range: address < A or address > B 1001 – 1111 (No trigger)

18.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.

	7	6	5	4	3	2	1	0
R	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 18-9. Debug Status Register (DBGS)

Table 18-6. DBGS Register Field Descriptions

Field	Description
7 AF	**Trigger Match A Flag** — AF is cleared at the start of a debug run and indicates whether a trigger match A condition was met since arming. 0 Comparator A has not matched 1 Comparator A match
6 BF	**Trigger Match B Flag** — BF is cleared at the start of a debug run and indicates whether a trigger match B condition was met since arming. 0 Comparator B has not matched 1 Comparator B match
5 ARMF	**Arm Flag** — While DBGEN = 1, this status bit is a read-only image of ARM in DBGC. This bit is set by writing 1 to the ARM control bit in DBGC (while DBGEN = 1) and is automatically cleared at the end of a debug run. A debug run is completed when the FIFO is full (begin trace) or when a trigger event is detected (end trace). A debug run can also be ended manually by writing 0 to ARM or DBGEN in DBGC. 0 Debugger not armed 1 Debugger armed
3:0 CNT[3:0]	**FIFO Valid Count** — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO. 0000 Number of valid words in FIFO = No valid data 0001 Number of valid words in FIFO = 1 0010 Number of valid words in FIFO = 2 0011 Number of valid words in FIFO = 3 0100 Number of valid words in FIFO = 4 0101 Number of valid words in FIFO = 5 0110 Number of valid words in FIFO = 6 0111 Number of valid words in FIFO = 7 1000 Number of valid words in FIFO = 8

Appendix A
Electrical Characteristics

A.1 Introduction

This appendix contains electrical and timing specifications for the MC9S08JM60 series of microcontrollers available at the time of publication.

A.2 Parameter Classification

The electrical parameters shown in this supplement are guaranteed by various methods. To give the customer a better understanding the following classification is used and the parameters are tagged accordingly in the tables where appropriate:

Table A-1. Parameter Classifications

P	Those parameters are guaranteed during production testing on each individual device.
C	Those parameters are achieved by the design characterization by measuring a statistically relevant sample size across process variations.
T	Those parameters are achieved by design characterization on a small sample size from typical devices under typical conditions unless otherwise noted. All values shown in the typical column are within this category.
D	Those parameters are derived mainly from simulations.

NOTE

The classification is shown in the column labeled “C” in the parameter tables where appropriate.

A.3 Absolute Maximum Ratings

Absolute maximum ratings are stress ratings only, and functional operation at the maxima is not guaranteed. Stress beyond the limits specified in Table A-2 may affect device reliability or cause permanent damage to the device. For functional operating conditions, refer to the remaining tables in this section.

This device contains circuitry protecting against damage due to high static voltage or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (for instance, either V_{SS} or V_{DD}).

Table A-2. Absolute Maximum Ratings

Rating	Symbol	Value	Unit
Supply voltage	V_{DD}	– 0.3 to + 5.8	V
Input voltage	V_{In}	– 0.3 to V_{DD} + 0.3	V
Instantaneous maximum current Single pin limit (applies to all port pins)[1, 2, 3]	I_D	± 25	mA
Maximum current into V_{DD}	I_{DD}	120	mA
Storage temperature	T_{stg}	–55 to +150	°C

1 Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive (V_{DD}) and negative (V_{SS}) clamp voltages, then use the larger of the two resistance values.

2 All functional non-supply pins are internally clamped to V_{SS} and V_{DD}.

3 Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current conditions. If positive injection current ($V_{In} > V_{DD}$) is greater than I_{DD}, the injection current may flow out of V_{DD} and could result in external power supply going out of regulation. Ensure external V_{DD} load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if the clock rate is very low which would reduce overall power consumption.

A.4 Thermal Characteristics

This section provides information about operating temperature range, power dissipation, and package thermal resistance. Power dissipation on I/O pins is usually small compared to the power dissipation in on-chip logic and it is user-determined rather than being controlled by the MCU design. In order to take $P_{I/O}$ into account in power calculations, determine the difference between actual pin voltage and V_{SS} or V_{DD} and multiply by the pin current for each I/O pin. Except in cases of unusually high pin current (heavy loads), the difference between pin voltage and V_{SS} or V_{DD} will be very small.

Table A-3. Thermal Characteristics

Num	C	Rating	Symbol	Value	Unit	Temp. Code
1	T	Operating temperature range (packaged)	T_A	–40 to 85	°C	C
2	D	Maximum junction temperature	T_J	135	°C	
3	T	Thermal resistance Single layer board 64-pin QFP 64-pin LQFP 48-pin QFN 44-pin LQFP Four layer board 64-pin QFP 64-pin LQFP 48-pin QFN 44-pin LQFP	θ_{JA}	 55 73 84 71 41 54 28 49	°C/W	—

The average chip-junction temperature (T_J) in °C can be obtained from:

$$T_J = T_A + (P_D \times \theta_{JA}) \quad \textit{Eqn. A-1}$$

where:

T_A = Ambient temperature, °C
θ_{JA} = Package thermal resistance, junction-to-ambient, °C/W
$P_D = P_{int} + P_{I/O}$
$P_{int} = I_{DD} \times V_{DD}$, Watts — chip internal power
$P_{I/O}$ = Power dissipation on input and output pins — user determined

For most applications, $P_{I/O} << P_{int}$ and can be neglected. An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad \textit{Eqn. A-2}$$

Solving equations 1 and 2 for K gives:

$$K = P_D \times (T_A + 273°C) + \theta_{JA} \times (P_D)^2 \quad \textit{Eqn. A-3}$$

where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations 1 and 2 iteratively for any value of T_A.

A.5 ESD Protection and Latch-Up Immunity

Although damage from electrostatic discharge (ESD) is much less common on these devices than on early CMOS circuits, normal handling precautions must be used to avoid exposure to static discharge. Qualification tests are performed to ensure that these devices can withstand exposure to reasonable levels of static without suffering any permanent damage.

All ESD testing is in conformity with AEC-Q100 Stress Test Qualification for Automotive Grade Integrated Circuits. During the device qualification ESD stresses were performed for the Human Body Model (HBM) and the Charge Device Model (CDM).

A device is defined as a failure if after exposure to ESD pulses the device no longer meets the device specification. Complete DC parametric and functional testing is performed per the applicable device specification at room temperature followed by hot temperature, unless specified otherwise in the device specification.

Table A-4. ESD and Latch-up Test Conditions

Model	Description	Symbol	Value	Unit
Human Body	Series Resistance	R1	1500	Ω
	Storage Capacitance	C	100	pF
	Number of Pulse per pin	—	3	
Latch-up	Minimum input voltage limit		–2.5	V
	Maximum input voltage limit		7.5	V

Table A-5. ESD and Latch-Up Protection Characteristics

Num	Rating	Symbol	Min	Max	Unit
1	Human Body Model (HBM)	V_{HBM}	±2000	—	V
2	Charge Device Model (CDM)	V_{CDM}	±500	—	V
3	Latch-up Current at T_A = 85°C	I_{LAT}	±100	—	mA

A.6 DC Characteristics

This section includes information about power supply requirements, I/O pin characteristics, and power supply current in various operating modes.

Table A-6. DC Characteristics

Num	C	Parameter	Symbol	Min	Typical[1]	Max.	Unit
1		Operating voltage[2]		2.7	—	5.5	V
2	P	Output high voltage — Low drive (PTxDSn = 0) 5 V, I_{Load} = –4 mA 3 V, I_{Load} = –2 mA 5 V, I_{Load} = –2 mA 3 V, I_{Load} = –1 mA	V_{OH}	 V_{DD} – 1.5 V_{DD} – 1.5 V_{DD} – 0.8 V_{DD} – 0.8	 — — — —	 — — — —	V
		Output high voltage — High drive (PTxDSn = 1) 5 V, I_{Load} = –15 mA 3 V, I_{Load} = –8 mA 5 V, I_{Load} = –8 mA 3 V, I_{Load} = –4 mA		 V_{DD} – 1.5 V_{DD} – 1.5 V_{DD} – 0.8 V_{DD} – 0.8	 — — — —	 — — — —	
3	P	Output low voltage — Low drive (PTxDSn = 0) 5 V, I_{Load} = 4 mA 3 V, I_{Load} = 2 mA 5 V, I_{Load} = 2 mA 3 V, I_{Load} = 1 mA	V_{OL}	 — — — —	 — — — —	 1.5 1.5 0.8 0.8	V
		Output low voltage — High drive (PTxDSn = 1) 5 V, I_{Load} = 15 mA 3 V, I_{Load} = 8 mA 5 V, I_{Load} = 8 mA 3 V, I_{Load} = 4 mA		 — — — —	 — — — —	 1.5 1.5 0.8 0.8	
4	P	Output high current — Max. total I_{OH} for all ports 5 V 3 V	I_{OHT}	 — —	 — —	 100 60	mA

Table A-6. DC Characteristics (continued)

Num	C	Parameter	Symbol	Min	Typical[1]	Max.	Unit
5	P	Output low current — Max. total I_{OL} for all ports 5 V 3 V	I_{OLT}	 — —	 — —	 100 60	mA
6	C	Input high voltage; all digital inputs 5 V 3 V	V_{IH}	 $0.65 \times V_{DD}$ $0.70 \times V_{DD}$	 —	 —	V
7	C	Input low voltage; all digital inputs	V_{IL}	—	—	$0.35 \times V_{DD}$	
8	C	Input hysteresis; all digital inputs	V_{hys}	$0.06 \times V_{DD}$			mV
9	C	Input leakage current (per pin); input only pins	$\lvert I_{In} \rvert$	—	0.1	1	μA
10	P	Hi-Z (off-state) leakage current (per pin)	$\lvert I_{OZ} \rvert$	—	0.1	1	μA
11	P	Internal pullup resistors[3]	R_{PU}	20	45	65	kΩ
12	P	Internal pulldown resistors[4]	R_{PD}	20	45	65	kΩ
13	T	Internal pullup resistor to USBDP (to V_{USB33}) Idle Transmit	R_{PUPD}	 900 1425	 1300 2400	 1575 3090	kΩ
14	D	DC injection current[5 6 7 8] (single pin limit) $V_{IN} > V_{DD}$ $V_{IN} < V_{SS}$	I_{IC}	 0 0	 — —	 2 –0.2	mA
		DC injection current (Total MCU limit, includes sum of all stressed pins) $V_{IN} > V_{DD}$ $V_{IN} < V_{SS}$		 0 0	 — —	 25 –5	mA
15	D	Input capacitance; all non-supply pins	C_{In}	—	—	8	pF
16	D	RAM retention voltage	V_{RAM}	—	0.6	1.0	V
17	D	POR re-arm voltage	V_{POR}	0.9	1.4	2.0	V
18	D	POR re-arm time	t_{POR}	10	—	—	μs

Table A-6. DC Characteristics (continued)

Num	C	Parameter	Symbol	Min	Typical[1]	Max.	Unit
19	P	Low-voltage detection threshold — High range V_{DD} falling V_{DD} rising	V_{LVD1}	 3.9 4.0	 4.0 4.1	 4.1 4.2	V
20	P	Low-voltage detection threshold — Low range V_{DD} falling V_{DD} rising	V_{LVD0}	 2.48 2.54	 2.56 2.62	 2.64 2.70	V
21	P	Low-voltage warning threshold — High range 1 V_{DD} falling V_{DD} rising	V_{LVW3}	 4.5 4.6	 4.6 4.7	 4.7 4.8	V
22	C	Low-voltage warning threshold — High range 0 V_{DD} falling V_{DD} rising	V_{LVW2}	 4.2 4.3	 4.3 4.4	 4.4 4.5	V
23	P	Low-voltage warning threshold Low range 1 V_{DD} falling V_{DD} rising	V_{LVW1}	 2.84 2.90	 2.92 2.98	 3.00 3.06	V
24	C	Low-voltage warning threshold — Low range 0 V_{DD} falling V_{DD} rising	V_{LVW0}	 2.66 2.72	 2.74 2.80	 2.82 2.88	V
25	T	Low-voltage inhibit reset/recover hysteresis +5V +3V	V_{hys}	 — —	 100 60	 — —	 mV mV
26	P	Bandgap voltage reference factory trimmed at V_{DD} = 5.0 V, Temp = 25°C	V_{BG}	1.19	1.20	1.21	V

1 Typical values are based on characterization data at 25°C unless otherwise stated.

2 Maximum is highest voltage that POR is guaranteed.

3 Measured with $V_{In} = V_{SS}$.

4 Measured with $V_{In} = V_{DD}$.

5 Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current conditions. If positive injection current ($V_{In} > V_{DD}$) is greater than I_{DD}, the injection current may flow out of V_{DD} and could result in external power supply going out of regulation. Ensure external V_{DD} load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if clock rate is very low (which would reduce overall power consumption).

6 All functional non-supply pins are internally clamped to V_{SS} and V_{DD}.

7 Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.

8 The $\overline{\text{RESET}}$ pin does not have a clamp diode to V_{DD}. Do not drive this pin above V_{DD}.

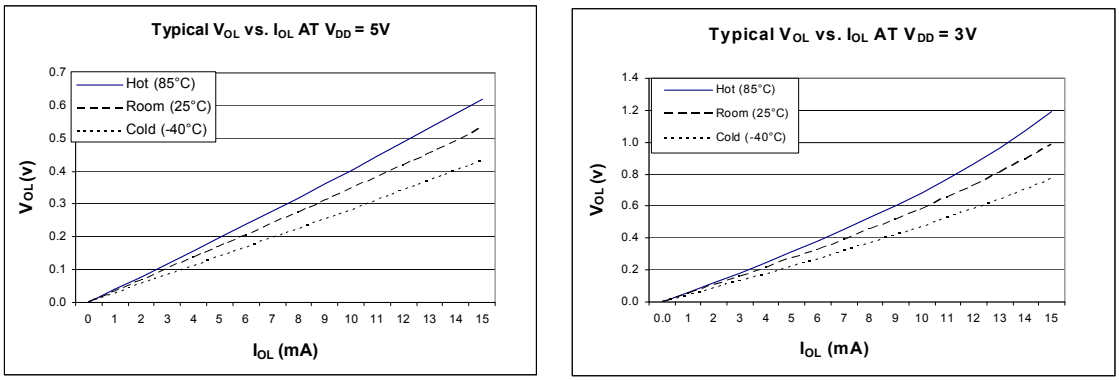


Figure A-1. Typical Low-side Drive (sink) characteristics – High Drive (PTxDSn = 1)

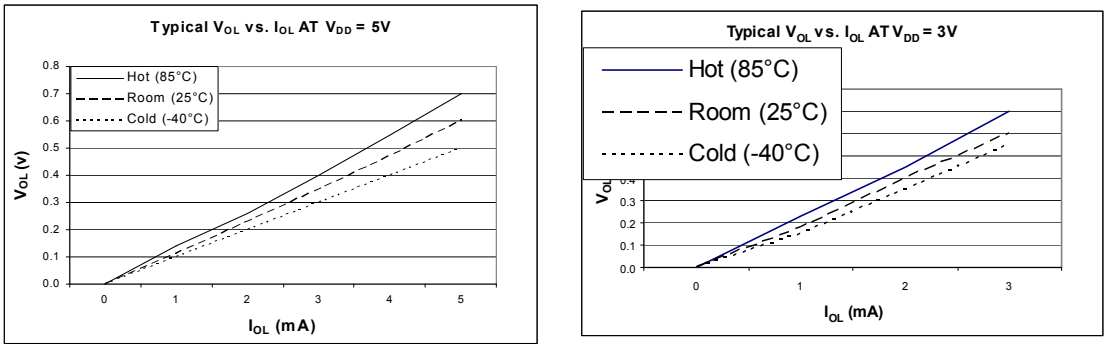


Figure A-2. Typical Low-side Drive (sink) characteristics – Low Drive (PTxDSn = 0)

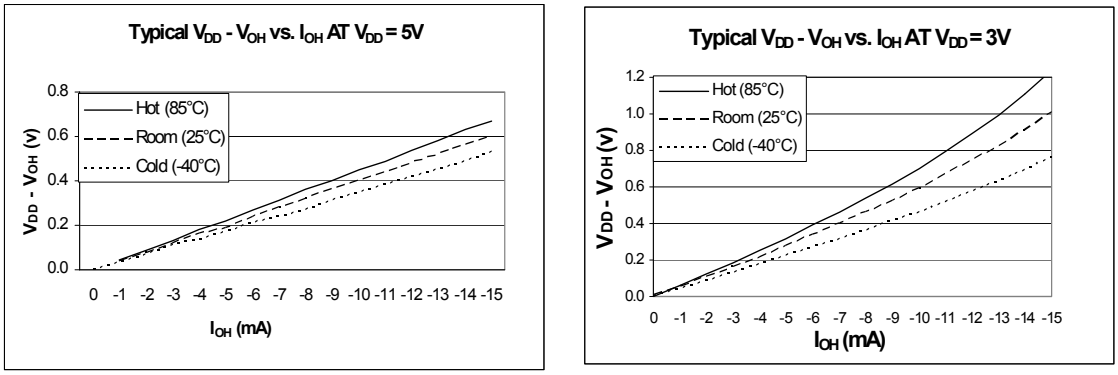


Figure A-3. Typical High-side Drive (source) characteristics – High Drive (PTxDSn = 1)

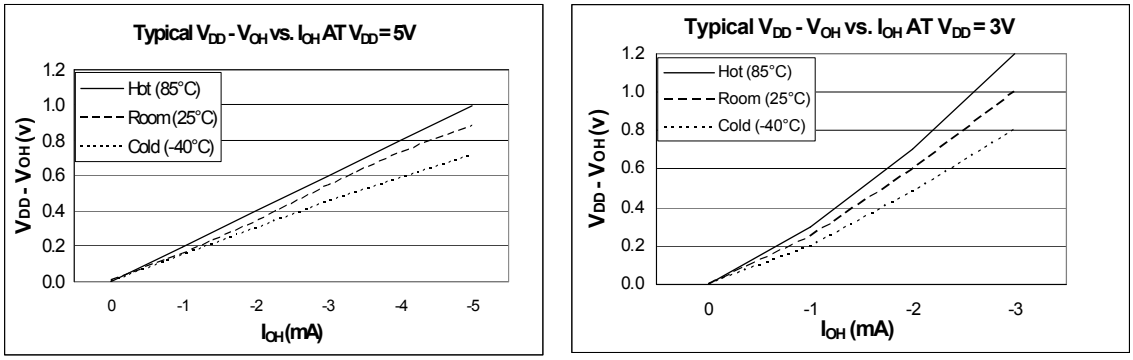


Figure A-4. Typical High-side Drive (source) characteristics – Low Drive (PTxDSn = 0)

A.7 Supply Current Characteristics

Table A-7. Supply Current Characteristics

Num	C	Parameter	Symbol	V_{DD} (V)	Typical[1]	Max[2]	Unit
1	C	Run supply current[3] measured at (Core clock = 2 MHz, f_{Bus} = 1 MHz, BLPE mode)	RI_{DD}	5	1.1	1.6	mA
				3	0.8	1.5	
2	P	Run supply current[3] measured at (Core clock = 8 MHz, f_{Bus} = 4 MHz, FBE mode)		5	4.9	8	mA
				3	4.3	7	
3	C	Run supply current[3] measured at (Core clock = 48 MHz, f_{Bus} = 24 MHz, PEE mode)		5	23	30	mA
				3	22	30	
4	P	Stop2 mode supply current –40 °C 25 °C 85 °C	$S2I_{DD}$	5	0.80	3 3 20	µA
		–40 °C 25 °C 85 °C		3	0.80	3 3 20	µA
5	P	Stop3 mode supply current –40 °C 25 °C 85 °C	$S3I_{DD}$	5	0.90	3 3 20	µA
		–40 °C 25 °C 85 °C		3	0.90	3 3 20	µA
6	P	Adder to stop2 or stop3 for RTC enabled[4], 25°C	ΔI_{SRTC}	5	300		nA
				3	300		nA
7	P	Adder to stop3 for LVD enabled (LVDE = LVDSE = 1)	ΔI_{SLVD}	5	110		µA
				3	90		µA
8	P	Adder to stop3 for oscillator enabled[5] (ERCLKEN = 1 and EREFSTEN = 1)	ΔI_{SOSC}	5	5		µA
				3	5		µA
9	T	USB module enable current[6]	ΔI_{USBE}	5	1.5		mA
10	T	USB suspend current[7]	I_{SUSP}	5	270	500	µA

1 Typicals are measured at 25°C.

2 Values given here are preliminary estimates prior to completing characterization.

3 All modules except USB and ADC active, Oscillator disabled (ERCLKEN = 0), using external clock resource for input, and does not include any dc loads on port pins.

4 Most customers are expected to find that auto-wakeup from stop2 or stop3 can be used instead of the higher current wait mode. Wait mode typical is 560 µA at 5 V and 422 µA at 3V with f_{Bus} = 1 MHz.

5 Values given under the following conditions: low range operation (RANGE = 0), low power mode (HGO = 0).

[6] Here USB module is enabled and clocked at 48 MHz (USBEN = 1, USBVREN =1, USBPHYEN = 1 and USBPU = 1), and D+ and D– pull down by two 15.1kΩ resisters independently. The current consumption may be much higher when the packets are being transmitted through the attached cable.

[7] MCU enters into Stop3 mode, USB bus in idle state. The USB suspend current will be dominated by the D+ pull up resister.

A.8 Analog Comparator (ACMP) Electricals

Table A-8. Analog Comparator Electrical Specifications

Num	C	Rating	Symbol	Min	Typical	Max	Unit
1	—	Supply voltage	V_{DD}	2.7	—	5.5	V
2	D	Supply current (active)	I_{DDAC}	—	20	35	μA
3	D	Analog input voltage	V_{AIN}	V_{SS} – 0.3	—	V_{DD}	V
4	D	Analog input offset voltage	V_{AIO}		20	40	mV
5	D	Analog Comparator hysteresis	V_H	3.0	6.0	20.0	mV
6	D	Analog input leakage current	I_{ALKG}			1.0	μA
7	D	Analog Comparator initialization delay	t_{AINIT}	—	—	1.0	μs

A.9 ADC Characteristics

Table A-9. 5 Volt 12-bit ADC Operating Conditions

Characteristic	Conditions	Symb	Min	Typ[1]	Max	Unit	Comment
Supply voltage	Absolute	V_{DDAD}	2.7	—	5.5	V	
	Delta to V_{DD} (V_{DD}-V_{DDAD})[2]	ΔV_{DDAD}	–100	0	+100	mV	
Ground voltage	Delta to V_{SS} (V_{SS}-V_{SSAD})[2]	ΔV_{SSAD}	–100	0	+100	mV	
Ref Voltage High		V_{REFH}	2.7	V_{DDAD}	V_{DDAD}	V	
Ref Voltage Low		V_{REFL}	V_{SSAD}	V_{SSAD}	V_{SSAD}	V	
Input Voltage		V_{ADIN}	V_{REFL}	—	V_{REFH}	V	
Input Capacitance		C_{ADIN}	—	4.5	5.5	pF	
Input Resistance		R_{ADIN}	—	3	5	kΩ	
Analog Source Resistance	12 bit mode f_{ADCK} > 4 MHz f_{ADCK} < 4 MHz	R_{AS}	 — —	 — —	 2 5	kΩ	External to MCU
	10 bit mode f_{ADCK} > 4 MHz f_{ADCK} < 4 MHz		 — —	 — —	 5 10		
	8 bit mode (all valid f_{ADCK})		—	—	10		

Table A-9. 5 Volt 12-bit ADC Operating Conditions (continued)

Characteristic	Conditions	Symb	Min	Typ[1]	Max	Unit	Comment
ADC Conversion Clock Freq.	High Speed (ADLPC=0)	f_{ADCK}	0.4	—	8.0	MHz	
	Low Power (ADLPC=1)		0.4	—	4.0		

1 Typical values assume V_{DDAD} = 5.0 V, Temp = 25°C, f_{ADCK} = 1.0 MHz unless otherwise stated. Typical values are for reference only and are not tested in production.

2 DC potential difference.

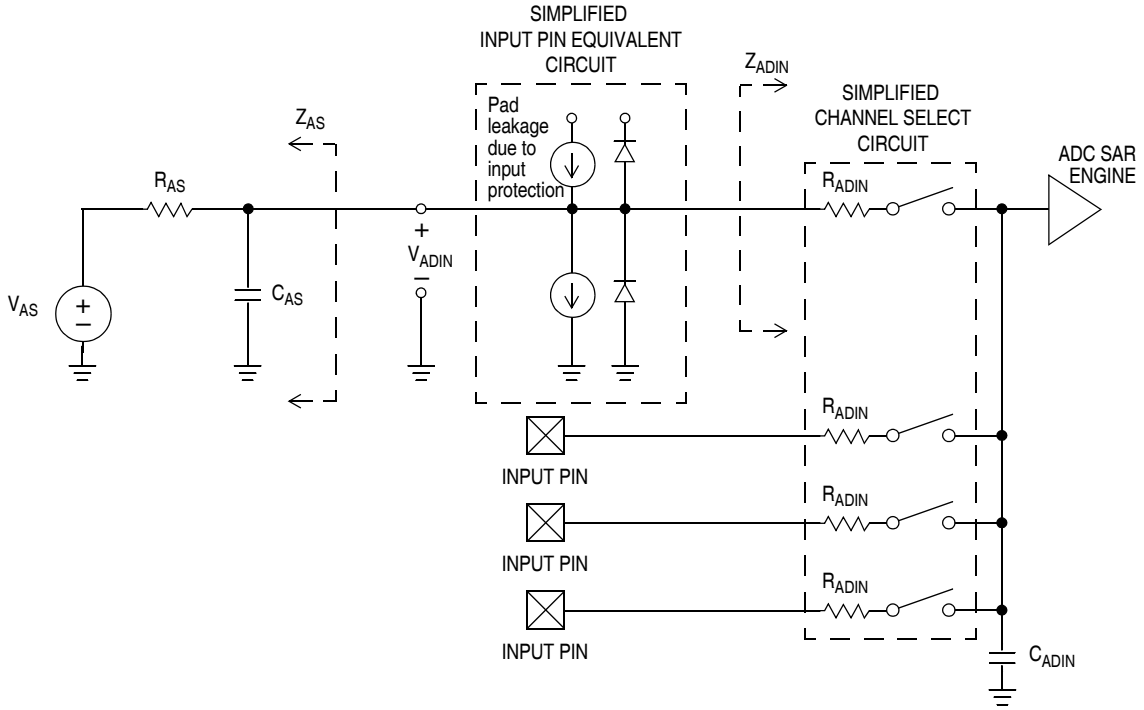


Figure A-5. ADC Input Impedance Equivalency Diagram

Table A-10. 5 Volt 12-bit ADC Characteristics ($V_{REFH} = V_{DDAD}$, $V_{REFL} = V_{SSAD}$)

Characteristic	Conditions	C	Symb	Min	Typ[1]	Max	Unit	Comment
Supply Current ADLPC=1 ADLSMP=1 ADCO=1		T	I_{DDAD}	—	133	—	μA	
Supply Current ADLPC=1 ADLSMP=0 ADCO=1		T	I_{DDAD}	—	218	—	μA	
Supply Current ADLPC=0 ADLSMP=1 ADCO=1		T	I_{DDAD}	—	327	—	μA	
Supply Current ADLPC=0 ADLSMP=0 ADCO=1		T	I_{DDAD}	—	0.582	1	mA	
Supply Current	Stop, Reset, Module Off		I_{DDAD}	—	0.011	1	μA	
ADC Asynchronous Clock Source	High Speed (ADLPC=0)	T	f_{ADACK}	2	3.3	5	MHz	$t_{ADACK} = 1/f_{ADACK}$
	Low Power (ADLPC=1)			1.25	2	3.3		
Conversion Time (Including sample time)	Short Sample (ADLSMP=0)	T	t_{ADC}	—	20	—	ADCK cycles	See Table 10.13 for conversion time variances
	Long Sample (ADLSMP=1)			—	40	—		
Sample Time	Short Sample (ADLSMP=0)	T	t_{ADS}	—	3.5	—	ADCK cycles	
	Long Sample (ADLSMP=1)			—	23.5	—		
Total Unadjusted Error	12 bit mode	T	E_{TUE}	—	±3.0	±10.0	LSB[2]	Includes quantization
	10 bit mode	P		—	±1	±2.5		
	8 bit mode	T		—	±0.5	±1.0		
Differential Non-Linearity	12 bit mode	T	DNL	—	±1.75	±4.0	LSB[2]	
	10 bit mode[3]	P		—	±0.5	±1.0		
	8 bit mode[2]	T		—	±0.3	±0.5		
Integral Non-Linearity	12 bit mode	T	INL	—	±1.5	±4.0	LSB[2]	
	10 bit mode	T		—	±0.5	±1.0		
	8 bit mode	T		—	±0.3	±0.5		
Zero-Scale Error	12 bit mode	T	E_{ZS}	—	±1.5	±6.0	LSB[2]	$V_{ADIN} = V_{SSAD}$
	10 bit mode	P		—	±0.5	±1.5		
	8 bit mode	T		—	±0.5	±0.5		

Table A-10. 5 Volt 12-bit ADC Characteristics ($V_{REFH} = V_{DDAD}$, $V_{REFL} = V_{SSAD}$) (continued)

Characteristic	Conditions	C	Symb	Min	Typ[1]	Max	Unit	Comment
Full-Scale Error	12 bit mode	T	E_{FS}	—	±1	±4.0	LSB[2]	$V_{ADIN} = V_{DDAD}$
	10 bit mode	P		—	±0.5	±1		
	8 bit mode	T		—	±0.5	±0.5		
Quantization Error	12 bit mode	D	E_Q	—	–1 to 0	–1 to 0	LSB[2]	
	10 bit mode			—	—	±0.5		
	8 bit mode			—	—	±0.5		
Input Leakage Error	12 bit mode	D	E_{IL}	—	±1	±10	LSB[2]	Pad leakage[4] * R_{AS}
	10 bit mode			—	±0.2	±2.5		
	8 bit mode			—	±0.1	±1		
Temp Sensor Voltage	25°C	D	V_{TEMP25}	—	1.396	—	V	
Temp Sensor Slope	–40 °C — 25 °C	D	m	—	3.266	—	mV/°C	
	25 °C — 125 °C			—	3.638	—		

1 Typical values assume V_{DDAD} = 5.0 V, Temp = 25°C, f_{ADCK} = 1.0 MHz unless otherwise stated. Typical values are for reference only and are not tested in production.

2 1 LSB = $(V_{REFH} - V_{REFL})/2^N$

3 Monotonicity and no-missing-codes guaranteed in 10 bit and 8 bit modes.

4 Based on input pad leakage current. Refer to pad electricals.

A.10 External Oscillator (XOSC) Characteristics

Table A-11. Oscillator Electrical Specifications (Temperature Range = –40 to 85°C Ambient)

Num	C	Rating	Symbol	Min	Typ[1]	Max	Unit
1	C	Oscillator crystal or resonator (EREFS = 1, ERCLKEN = 1) Low range (RANGE = 0) High range (RANGE = 1) FEE or FBE mode [2] High range (RANGE = 1) PEE or PBE mode [3] High range (RANGE = 1, HGO = 1) BLPE mode High range (RANGE = 1, HGO = 0) BLPE mode	 f_{lo} $f_{hi\text{-}fll}$ $f_{hi\text{-}pll}$ $f_{hi\text{-}hgo}$ $f_{hi\text{-}lp}$	 32 1 1 1 1	 — — — — —	 38.4 5 16 16 8	 kHz MHz MHz MHz MHz
2	—	Load capacitors	C_1, C_2	See crystal or resonator manufacturer's recommendation.			
3	—	Feedback resistor Low range (32 kHz to 38.4 kHz) High range (1 MHz to 16 MHz)	R_F		10 1		MΩ MΩ
4	—	Series resistor Low range, low gain (RANGE = 0, HGO = 0) Low range, high gain (RANGE = 0, HGO = 1) High range, low gain (RANGE = 1, HGO = 0) High range, high gain (RANGE = 1, HGO = 1) ≥ 8 MHz 4 MHz 1 MHz	R_S	 — — — — — —	 0 100 0 0 0 0	 — — — 0 10 20	kΩ
5	T	Crystal start-up time [4] Low range, low gain (RANGE = 0, HGO = 0) Low range, high gain (RANGE = 0, HGO = 1) High range, low gain (RANGE = 1, HGO = 0)[5] High range, high gain (RANGE = 1, HGO = 1)[5]	 $t_{CSTL\text{-}LP}$ $t_{CSTL\text{-}HGO}$ $t_{CSTH\text{-}LP}$ $t_{CSTH\text{-}HGO}$	 — — — —	 200 400 5 15	 — — — —	ms
6	T	Square wave input clock frequency (EREFS = 0, ERCLKEN = 1) FEE or FBE mode [2] PEE or PBE mode [3] BLPE mode	f_{extal}	 0.03125 1 0	 — — —	 5 16 40	 MHz MHz MHz

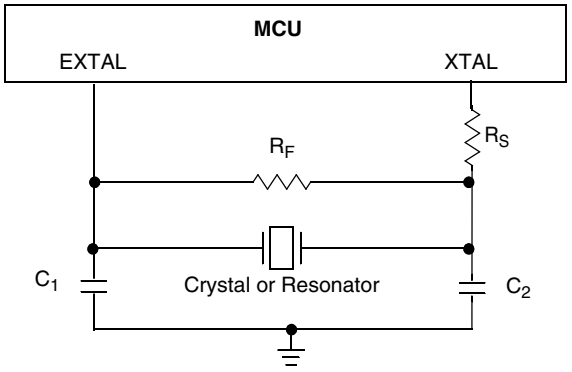
1 Typical data was characterized at 3.0 V, 25°C or is recommended value.

2 When MCG is configured for FEE or FBE mode, input clock source must be divided using RDIV to within the range of 31.25 kHz to 39.0625 kHz.

3 When MCG is configured for PEE or PBE mode, input clock source must be divided using RDIV to within the range of 1 MHz to 2 MHz.

4 This parameter is characterized and not tested on each device. Proper PC board layout procedures must be followed to achieve specifications.

5 4 MHz crystal



A.11 MCG Specifications

Table A-12. MCG Frequency Specifications (Temperature Range = –40 to 125°C Ambient)

Num	C	Rating	Symbol	Min	Typical	Max	Unit
1	P	Internal reference frequency - factory trimmed at V_{DD} = 5 V and temperature = 25 °C	f_{int_ft}	—	31.25	—	kHz
2	P	Average internal reference frequency – untrimmed [1]	f_{int_ut}	25	32.7	41.66	kHz
3	P	Average internal reference frequency Q – user trimmed	f_{int_t}	31.25	—	39.0625	kHz
4	D	Internal reference startup time	t_{irefst}	—	60	100	μs
5	—	DCO output frequency range - untrimmed [1] value provided for reference: f_{dco_ut} = 1024 X f_{int_ut}	f_{dco_ut}	25.6	33.48	42.66	MHz
6	P	DCO output frequency range - trimmed	f_{dco_t}	32	—	40	MHz
7	C	Resolution of trimmed DCO output frequency at fixed voltage and temperature (using FTRIM)	$\Delta f_{dco\_res\_t}$	—	±0.1	±0.2	$\%f_{dco}$
8	C	Resolution of trimmed DCO output frequency at fixed voltage and temperature (not using FTRIM)	$\Delta f_{dco\_res\_t}$	—	±0.2	±0.4	$\%f_{dco}$
9	P	Total deviation of trimmed DCO output frequency over voltage and temperature	Δf_{dco_t}	—	+0.5 –1.0	±2	$\%f_{dco}$
10	C	Total deviation of trimmed DCO output frequency over fixed voltage and temperature range of 0 –70 °C	Δf_{dco_t}	—	±0.5	±1	$\%f_{dco}$
11	C	FLL acquisition time [2]	$t_{fll\_acquire}$	—	—	1	ms
12	D	PLL acquisition time [3]	$t_{pll\_acquire}$	—	—	1	ms
13	C	Long term Jitter of DCO output clock (averaged over 2ms interval) [4]	C_{Jitter}	—	0.02	0.2	$\%f_{dco}$
14	D	VCO operating frequency	f_{vco}	7.0	—	55.0	MHz
15	D	PLL reference frequency range	f_{pll_ref}	1.0	—	2.0	MHz
16	T	Long term accuracy of PLL output clock (averaged over 2 ms)	$f_{pll\_jitter\_2ms}$	—	0.590[5]	—	$\%f_{pll}$
17	T	Jitter of PLL output clock measured over 625 ns	$f_{pll\_jitter\_625ns}$	—	0.566[5]	—	$\%f_{pll}$
18	D	Lock entry frequency tolerance [6]	D_{lock}	±1.49	—	±2.98	%
19	D	Lock exit frequency tolerance [7]	D_{unl}	±4.47	—	±5.97	%
20	D	Lock time – FLL	t_{fll_lock}	—	—	$t_{fll\_acquire}$+ $1075(1/f_{int\_t})$	s
21	D	Lock time – PLL	t_{pll_lock}	—	—	$t_{pll\_acquire}$+ $1075(1/f_{pll\_ref})$	s
22	D	Loss of external clock minimum frequency – RANGE = 0	f_{loc_low}	(3/5) x f_{int}	—	—	kHz
23	D	Loss of external clock minimum frequency – RANGE = 1	f_{loc_high}	(16/5) x f_{int}	—	—	kHz

1 TRIM register at default value (0x80) and FTRIM control bit at default value (0x0).

2 This specification applies to any time the FLL reference source or reference divider is changed, trim value changed or changing from FLL disabled (BLPE, BLPI) to FLL enabled (FEI, FEE, FBE, FBI). If a crystal/resonator is being used as the reference, this specification assumes it is already running.

3 This specification applies to any time the PLL VCO divider or reference divider is changed, or changing from PLL disabled (BLPE, BLPI) to PLL enabled (PBE, PEE). If a crystal/resonator is being used as the reference, this specification assumes it is already running.

4 Jitter is the average deviation from the programmed frequency measured over the specified interval at maximum f_{BUS}. Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the FLL circuitry via V_{DD} and V_{SS} and variation in crystal oscillator frequency increase the C_{Jitter} percentage for a given interval.

5 Jitter measurements are based upon a 48 MHz MCGOUT clock frequency..

6 Below D_{lock} minimum, the MCG is guaranteed to enter lock. Above D_{lock} maximum, the MCG will not enter lock. But if the MCG is already in lock, then the MCG may stay in lock.

7 Below D_{unl} minimum, the MCG will not exit lock if already in lock. Above D_{unl} maximum, the MCG is guaranteed to exit lock.

A.12 AC Characteristics

This section describes ac timing characteristics for each peripheral system.

A.12.1 Control Timing

Table A-13. Control Timing

Num	C	Parameter	Symbol	Min	Typ[1]	Max	Unit
1		Bus frequency ($t_{cyc} = 1/f_{Bus}$)	f_{Bus}	dc	—	24	MHz
2		Internal low-power oscillator period	t_{LPO}	700		1300	μs
3		External reset pulse width[2]	t_{extrst}	100		—	ns
4		Reset low drive	t_{rstdrv}	66 x t_{cyc}		—	ns
5		Active background debug mode latch setup time	t_{MSSU}	500		—	ns
6		Active background debug mode latch hold time	t_{MSH}	100		—	ns
7		IRQ pulse width Asynchronous path[2] Synchronous path[3]	t_{ILIH}, t_{IHIL}	 100 1.5 x t_{cyc}	—	—	ns
8		KBIPx pulse width Asynchronous path[2] Synchronous path[3]	t_{ILIH}, t_{IHIL}	 100 1.5 x t_{cyc}	—	—	ns
9		Port rise and fall time low output drive (PTxDS = 0), (load = 50 pF)[4] Slew rate control disabled (PTxSE = 0) Slew rate control enabled (PTxSE = 1) high output drive (PTxDS = 1), (load = 50 pF) Slew rate control disabled (PTxSE = 0) Slew rate control enabled (PTxSE = 1)	t_{Rise}, t_{Fall}	 — — — —	 40 75 11 35		ns

1 Typical values are based on characterization data at V_{DD} = 5.0 V, 25 °C unless otherwise stated.

2 This is the shortest pulse that is guaranteed to be recognized as a reset pin request. Shorter pulses are not guaranteed to override reset requests from internal sources.

3 This is the minimum pulse width that is guaranteed to pass through the pin synchronization circuitry. Shorter pulses may or may not be recognized. In stop mode, the synchronizer is bypassed so shorter pulses can be recognized in that case.

4 Timing is shown with respect to 20% V_{DD} and 80% V_{DD} levels. Temperature range –40 °C to 85 °C.

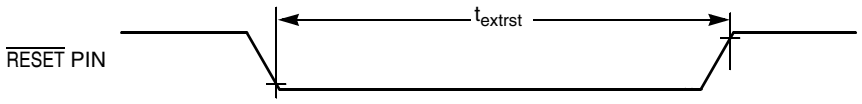


Figure A-6. Reset Timing

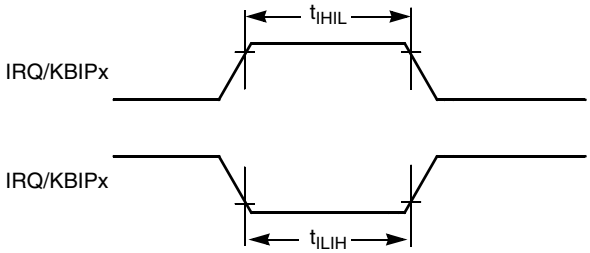


Figure A-7. IRQ/KBIPx Timing

A.12.2 Timer/PWM (TPM) Module Timing

Synchronizer circuits determine the shortest input pulses that can be recognized or the fastest clock that can be used as the optional external source to the timer counter. These synchronizers operate from the current bus rate clock.

Table A-14. TPM Input Timing

NUM	C	Function	Symbol	Min	Max	Unit
1	—	External clock frequency	f_{TPMext}	dc	$f_{Bus}/4$	MHz
2	—	External clock period	t_{TPMext}	4	—	t_{cyc}
3	D	External clock high time	t_{clkh}	1.5	—	t_{cyc}
4	D	External clock low time	t_{clkl}	1.5	—	t_{cyc}
5	D	Input capture pulse width	t_{ICPW}	1.5	—	t_{cyc}

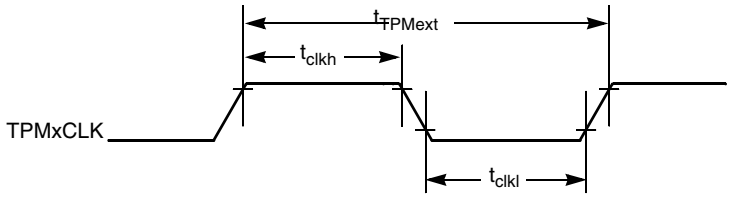


Figure A-8. Timer External Clock

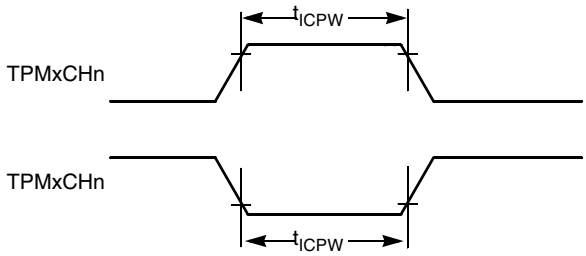


Figure A-9. Timer Input Capture Pulse

A.12.3 SPI Characteristics

Table A-15 and Figure A-10 through Figure A-13 describe the timing requirements for the SPI system.

Table A-15. SPI Electrical Characteristic

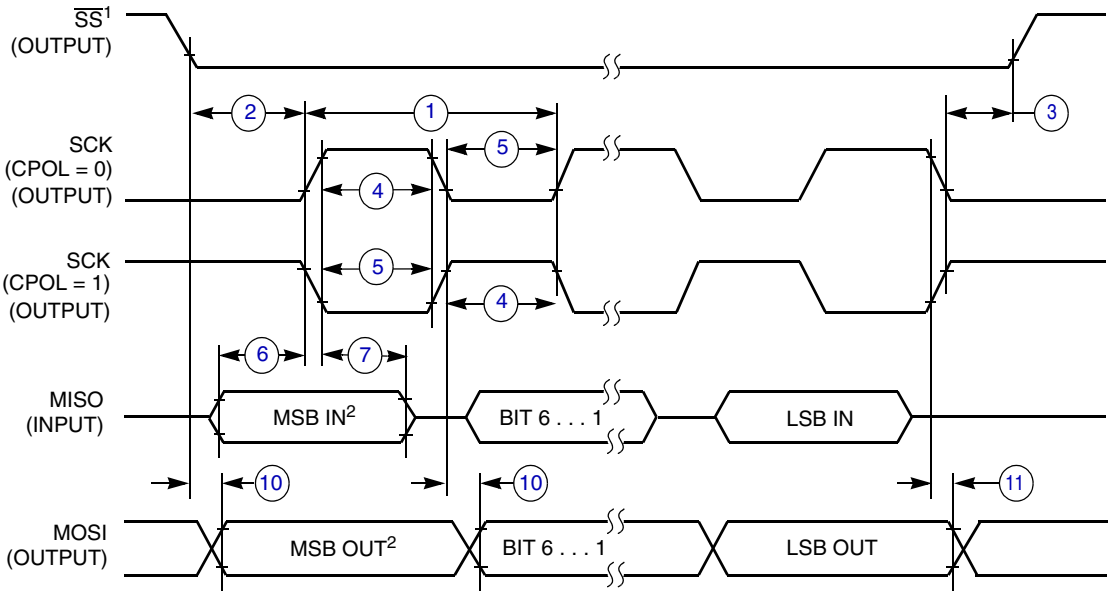
Num[1]	C	Characteristic[2]	Symbol	Min	Max	Unit
1	D	Operating frequency Master Slave	f_{op} f_{op}	$f_{Bus}/2048$ dc	$f_{Bus}/2$ $f_{Bus}/4$	Hz
2	D	Cycle time Master Slave	t_{SCK} t_{SCK}	2 4	2048 —	t_{cyc} t_{cyc}
3	D	Enable lead time Master Slave	t_{Lead} t_{Lead}	— 1/2	1/2 —	t_{SCK} t_{SCK}
4	D	Enable lag time Master Slave	t_{Lag} t_{Lag}	— 1/2	1/2 —	t_{SCK} t_{SCK}
5	D	Clock (SPSCK) high time Master and Slave	t_{SCKH}	1/2 t_{SCK} – 25	—	ns
6	D	Clock (SPSCK) low time Master and Slave	t_{SCKL}	1/2 t_{SCK} – 25	—	ns
7	D	Data setup time (inputs) Master Slave	$t_{SI(M)}$ $t_{SI(S)}$	30 30	— —	ns ns
8	D	Data hold time (inputs) Master Slave	$t_{HI(M)}$ $t_{HI(S)}$	30 30	— —	ns ns
9	D	Access time, slave[3]	t_A	0	40	ns
10	D	Disable time, slave[4]	t_{dis}	—	40	ns
11	D	Data setup time (outputs) Master Slave	t_{SO} t_{SO}	25 25	— —	ns ns
12	D	Data hold time (outputs) Master Slave	t_{HO} t_{HO}	–10 –10	— —	ns ns

1 Refer to Figure A-10 through Figure A-13.

2 All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; 100 pF load on all SPI pins. All timing assumes slew rate control disabled and high drive strength enabled for SPI output pins.

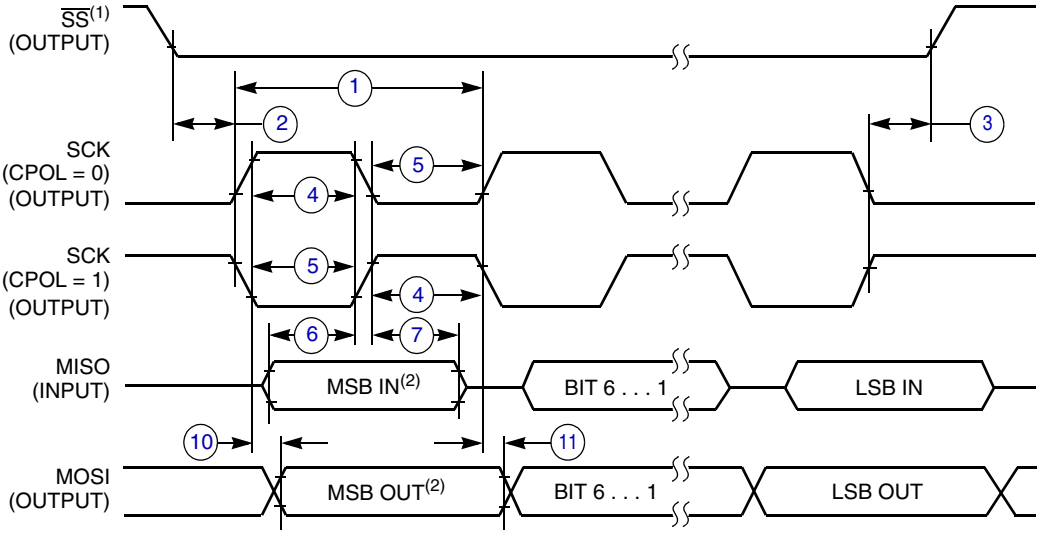
3 Time to data active from high-impedance state.

4 Hold time to high-impedance state.



NOTES:
1. $\overline{SS}$ output mode (MODFEN = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

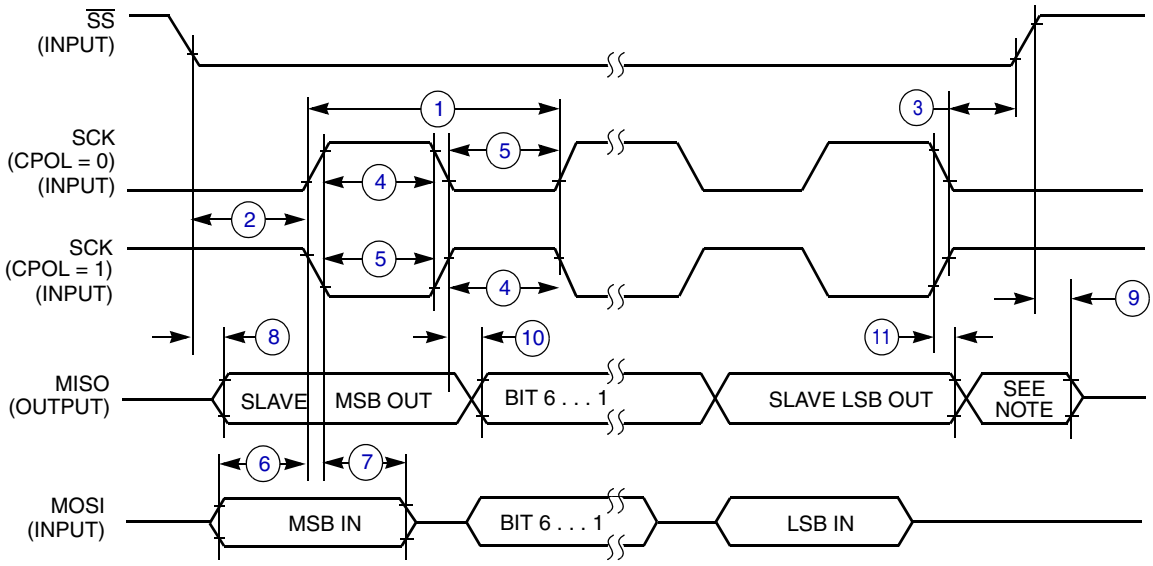
Figure A-10. SPI Master Timing (CPHA = 0)



NOTES:

1. $\overline{SS}$ output mode (MODFEN = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

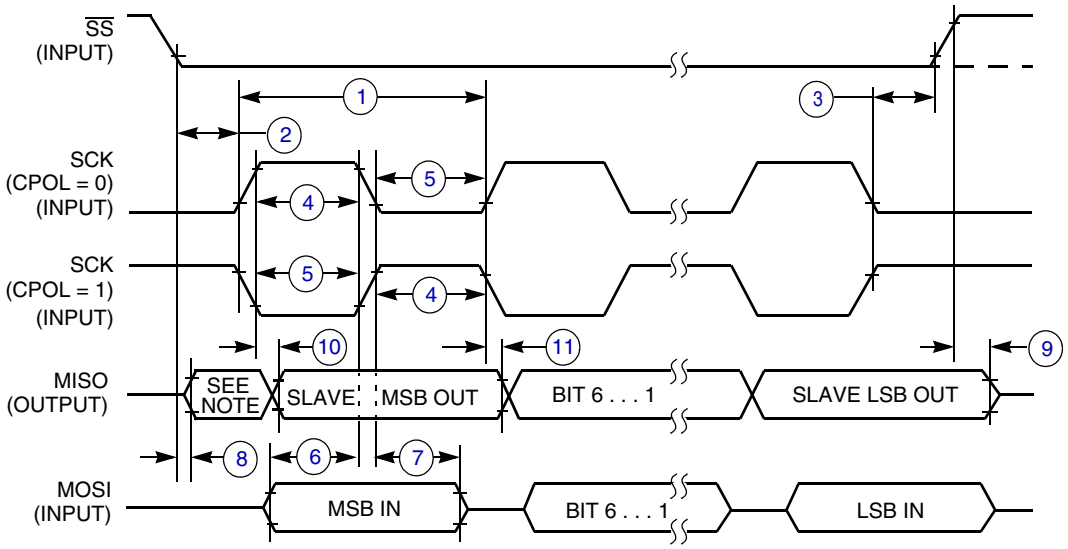
Figure A-11. SPI Master Timing (CPHA = 1)



NOTE:

1. Not defined but normally MSB of character just received

Figure A-12. SPI Slave Timing (CPHA = 0)



NOTE:
1. Not defined but normally LSB of character just received

Figure A-13. SPI Slave Timing (CPHA = 1)

A.13 Flash Specifications

This section provides details about program/erase times and program-erase endurance for the flash memory.

Program and erase operations do not require any special power sources other than the normal V_{DD} supply. For more detailed information about program/erase operations.

Table A-16. Flash Characteristics

Num	C	Characteristic	Symbol	Min	Typ[1]	Max	Unit
1		Supply voltage for program/erase	$V_{prog/erase}$	2.7		5.5	V
2		Supply voltage for read operation	V_{Read}	2.7		5.5	V
3		Internal FCLK frequency[2]	f_{FCLK}	150		200	kHz
4		Internal FCLK period (1/FCLK)	t_{Fcyc}	5		6.67	µs
5		Byte program time (random location)(2)	t_{prog}	9			t_{Fcyc}
6		Byte program time (burst mode)(2)	t_{Burst}	4			t_{Fcyc}
7		Page erase time[3]	t_{Page}	4000			t_{Fcyc}
8		Mass erase time(2)	t_{Mass}	20,000			t_{Fcyc}
9	C	Program/erase endurance[4] T_L to T_H = –40°C to + 85°C T = 25°C		10,000 —	— 100,000	— —	cycles
10		Data retention[5]	t_{D_ret}	15	100	—	years

[1] Typical values are based on characterization data at V_{DD} = 5.0 V, 25°C unless otherwise stated.

[2] The frequency of this clock is controlled by a software setting.

[3] These values are hardware state machine controlled. User code does not need to count cycles. This information supplied for calculating approximate time to program and erase.

[4] Typical endurance for Flash is based on the intrinsic bitcell performance. For additional information on how Freescale Semiconductor defines typical endurance, please refer to Engineering Bulletin EB619/D, *Typical Endurance for Nonvolatile Memory*.

[5] Typical data retention values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25 °C using the Arrhenius equation. For additional information on how Freescale Semiconductor defines typical data retention, please refer to Engineering Bulletin EB618/D, *Typical Data Retention for Nonvolatile Memory.*

A.14 USB Electricals

The USB electricals for the S08USBV1 module conform to the standards documented by the Universal Serial Bus Implementers Forum. For the most up-to-date standards, visit http://www.usb.org.

If the Freescale S08USBV1 implementation has electrical characteristics that deviate from the standard or require additional information, this space would be used to communicate that information.

Table A-17. Internal USB 3.3V Voltage Regulator Characteristics

	Symbol	Unit	Min	Typ	Max
Regulator operating voltage	V_{regin}	V	3.9	—	5.5
VREG output	V_{regout}	V	3	3.3	3.6
V_{USB33} input with internal VREG disabled	$V_{usb33in}$	V	3	3.3	3.6
VREG Quiescent Current	I_{VRQ}	mA	—	0.5	—

A.15 EMC Performance

Electromagnetic compatibility (EMC) performance is highly dependant on the environment in which the MCU resides. Board design and layout, circuit topology choices, location and characteristics of external components as well as MCU software operation all play a significant role in EMC performance. The system designer must consult Freescale applications notes such as AN2321, AN1050, AN1263, AN2764, and AN1259 for advice and guidance specifically targeted at optimizing EMC performance.

A.15.1 Radiated Emissions

Microcontroller radiated RF emissions are measured from 150 kHz to 1 GHz using the TEM/GTEM Cell method in accordance with the IEC 61967-2 and SAE J1752/3 standards. The measurement is performed with the microcontroller installed on a custom EMC evaluation board while running specialized EMC test software. The radiated emissions from the microcontroller are measured in a TEM cell in two package orientations (North and East). For more detailed information concerning the evaluation results, conditions and setup, please refer to the EMC Evaluation Report for this device.

The maximum radiated RF emissions of the tested configuration in all orientations are less than or equal to the reported emissions levels.

Table A-18. Radiated Emissions

Parameter	Symbol	Conditions	Frequency	f_{OSC}/f_{BUS}	Level[1] (Max)	Unit
Radiated emissions, electric field	V_{RE_TEM}	V_{DD} = 5.0 V T_A = +25°C	0.15 – 50 MHz	4 MHz crystal 24 MHz Bus	20	dBμV
			50 – 150 MHz		27	
			150 – 500 MHz		27	
			500 – 1000 MHz		16	
			IEC Level		1	—
			SAE Level		3	—

1 Data based on qualification test results.

Appendix B
Ordering Information and Mechanical Drawings

B.1 Ordering Information

This section contains ordering numbers for MC9S08JM60 series devices. See below for an example of the device numbering system.

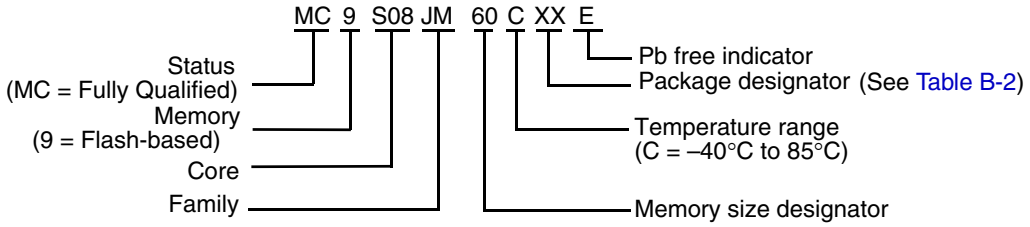
Table B-1. Device Numbering System

Device Number[1]	Memory		Available Packages[2]
	Flash	RAM	Type
MC9S08JM60	60,912	4096	64-pin LQFP 64-pin QFP 48-pin QFN 44-pin LQFP
MC9S08JM32	32,768	2048	

1 See Table 1-1 for a complete description of modules included on each device.

2 See Table B-2 for package information.

B.2 Orderable Part Numbering System



B.3 Mechanical Drawings

This following pages contain mechanical specifications for MC9S08JM60 series package options. See Table B-2 for the document numbers that correspond to each package type.

Table B-2. Package Information

Pin Count	Type	Designator	Document No.
44	LQFP	LD	98ASS23225W
48	QFN	GT	98ARH99048A
64	LQFP	LH	98ASS23234W
64	QFP	QH	98ASB42844B

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number MC9S08JM60
Revision 5, 10/2014

freescale™

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Freescale Semiconductor:

MC9S08JM60CGT MC9S08JM60CLD MC9S08JM60CLH MC9S08JM60CQH MC9S08JM60CLDR MC9S08JM32CQH MC9S08JM32CLH MC9S08JM32CLD MC9S08JM32CGT