MC9S08JE128
MC9S08JE64

Reference Manual

HCS08
Microcontrollers

freescale.com

freescale™
semiconductor

MC9S08JE128 series

8-Bit HCS08 Central Processor Unit (CPU)

- Up to 48-MHz CPU above 2.4 V, 40 MHz CPU above 2.1 V, and 20 MHz CPU above 1.8 V across temperature of -40°C to 105°C
- HCS08 instruction set with added BGND instruction
- Support for up to 33 interrupt/reset sources

On-Chip Memory

- 128 K Dual Array Flash read/program/erase over full operating voltage and temperature
- 12 KB Random-access memory (RAM)
- Security circuitry to prevent unauthorized access to RAM and Flash

Power-Saving Modes

- Peripheral clock enable register can disable clocks to unused modules to reduce currents
- Time of Day (TOD) — Ultra-low power 1/4 sec counter with up to 64s timeout.
- Ultra-low power external oscillator that can be used in stop modes to provide accurate clock source to the TOD. 6 usec typical wake up time from stop3 mode

Clock Source Options

- Oscillator (XOSC1) — Loop-control Pierce oscillator; 32.768 kHz crystal or ceramic resonator dedicated for TOD operation.
- Oscillator (XOSC2) — for high frequency crystal input for MCG reference to be used for system clock and USB operations.
- Multipurpose Clock Generator (MCG) — PLL and FLL; precision trimming of internal reference allows 0.2% resolution and 2% deviation over temperature and voltage; supports CPU frequencies from 4 kHz to 48 MHz.

System Protection

- Watchdog computer operating properly (COP) reset with option to run from dedicated 1-kHz internal clock source or bus clock with reset or interrupt; selectable trip points; separate low-voltage warning with optional interrupt; selectable trip pointsFlash block protection for each array to prevent accidental write/erasure
- Hardware CRC to support fast cyclic redundancy checks

Development Support

- Single-wire background debug interface
- Real-time debug with 6 hardware breakpoints (4 PC, 1 address and 1 data) Breakpoint capability to allow single breakpoint setting during in-circuit debugging
- On-chip in-circuit emulator (ICE) debug module containing 3 comparators and 9 trigger modes

Peripherals

- **CMT**— Carrier Modulator timer for remote control communications. Carrier generator, modulator and driver for dedicated infrared out. Can be used as an output compare timer.
- **IIC**— Up to 100 kbps with maximum bus loading; Multi-master operation; Programmable slave address; Interrupt driven byte-by-byte data transfer; supports broadcast mode and 11-bit addressing
- **PRACMP** — Analog comparator with selectable interrupt; compare option to programmable internal reference voltage; operation in stop3

- **SCI** — Two serial communications interfaces with optional 13-bit break; option to connect Rx input to PRACMP output on SCI1 and SCI2; High current drive on Tx on SCI1 and SCI2; wake-up from stop3 on Rx edge
- **SPI1**— Serial peripheral interface (SPI) with 64-bit FIFO buffer; 16-bit or 8-bit data transfers; full-duplex or single-wire bidirectional; double-buffered transmit and receive; master or slave mode; MSB-first or LSB-first shifting
- **SPI2**— Serial peripheral interface with full-duplex or single-wire bidirectional; Double-buffered transmit and receive; Master or Slave mode; MSB-first or LSB-first shifting
- **TPM** — Two 4-channel Timer/PWM Module; Selectable input capture, output compare, or buffered edge- or center-aligned PWM on each channel; external clock input/pulse accumulator
- **USB** — Supports USB in full-speed device configuration. On-chip transceiver and 3.3V regulator help save system cost, fully compliant with USB Specification 2.0. Allows control, bulk, interrupt and isochronous transfers.
- **ADC12** — 12-bit Successive approximation ADC with up to 4 dedicated differential channels and 8 single-ended channels; range compare function; 1.7 mV/°C temperature sensor; internal bandgap reference channel; operation in stop3; fully functional from 3.6V to 1.8V, Configurable hardware trigger for 8 Channel select and result registers
- **PDB** — Programmable delay block with 16-bit counter and modulus and prescale to set reference clock to bus divided by 1 to bus divided by 2048; 8 trigger outputs for ADC12 module provides periodic coordination of ADC sampling sequence with sequence completion interrupt; Back-to-Back mode and Timed mode
- **DAC** — 12-bit resolution; 16-word data buffers with configurable watermark.

Input/Output

- Up to 47 GPIOs and 2 output-only pin and 1 input-only pin.
- Voltage Reference output (VREFO).
- Dedicated infrared output pin (IRO) with high current sink capability.
- Up to 16 KBI pins with selectable polarity.

Package Options

- 81-MBGA 10x10 mm
- 80-LQFP 12x12 mm
- 64-LQFP 10x10 mm

MC9S08JE128 Reference Manual

Covers MC9S08JE128
MC9S08JE64

Related Documentation:

- **MC9S08JE128 (Data Sheet)**
  Contains descriptive feature set, block diagram, and electrical characteristics for the device.

Find the most current versions of all documents at:

http://www.freescale.com/

freescale™
semiconductor

Revision History

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

http://www.freescale.com/

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

Revision Number	Revision Date	Description of Changes
0	6/2009	Initial draft of the MC9S08JE128 book.
1	7/2009	Added missing register name FPROTD to memory map. Added missing FPROT register description and flash protection address range. Updated the PDB chapter to latest version of the block guide and corrected the PDB intro to reference ACMPO and ADCSC1A_COCO correctly as well as added the section, "PDB Trigger Acknowledgement Inputs." Updated the ADC Channel Assignment tables to latest version.
2	6/2010	Updated DAC, VREF, PDB, ADC, and USB.
3	12/2010	Updated to latest block guides.

Contents

Section	Title	Page

Preface

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
Keyboard Interrupt

Chapter 8
Central Processor Unit (S08CPUV5)

Chapter 9
Programmable Analog Comparator (S08PRACMPV1)

Chapter 10
Analog-to-Digital Converter (S08ADC12V1)

Chapter 11
Cyclic Redundancy Check (S08CRCV3)

Chapter 14
Inter-Integrated Circuit (S08IICV3)

Chapter 15
Multipurpose Clock Generator (S08MCGV3)

Chapter 16
Programmable Delay Block (S08PDBV1)

Chapter 17
Serial Communications Interface (S08SCIV4)

Chapter 18
16-bit Serial Peripheral Interface 1 (S08SPI16V2)

Chapter 19
8-bit Serial Peripheral Interface 2 (S08SPI8V5)

Chapter 20
Time Of Day Module (S08TODV1)

Chapter 21
Timer/Pulse-Width Modulator (S08TPMV3)

Chapter 22
Universal Serial Bus (S08USBV1)

Chapter 23
Voltage Reference Module (S08VREFV1)

Chapter 24
Development Support

Preface

About This Book

The primary objective of this reference manual is to define the processor for software and hardware developers. The information in this book is subject to change without notice, as described in the disclaimers on the title page. As with any technical documentation, use the most recent version of the documentation.

To locate any published errata or updates for this document, refer to the world-wide web at http://www.freescale.com/.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with this HCS08 processor. An understanding of operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the HCS08 architecture is recommended.

Suggested Reading

This section lists additional reading that provides background for the information in this manual as well as general information about HCS08 architecture.

General Information

Useful information about the HCS08 architecture and computer architecture in general:

- *HCS08 Family Reference Manual (enter keyword HCS08RMv1/D at www.freescale.com)*
- *Using Microprocessors and Microcomputers: The Motorola Family,* William C. Wray, Ross Bannatyne, Joseph D. Greenfield
- *Computer Architecture: A Quantitative Approach*, Second Edition, by John L. Hennessy and David A. Patterson.
- *Computer Organization and Design: The Hardware/Software Interface*, Second Edition, David A. Patterson and John L. Hennessy.

HCS08 Documentation

HCS08 documentation is available from the sources listed on the back cover of this manual, as well as our web site, http://www.freescale.com/.

- Reference manuals — These books provide details about individual HCS08 implementations and are intended to be used in conjunction with the device's data sheet.

- Data sheets — Data sheets provide specific data regarding pin-out diagrams, bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations.
- Product briefs — Each device has a product brief that provides an overview of its features. This document is roughly equivalent to the overview (Chapter 1) of an device's reference manual.
- Application notes — These short documents address specific design issues useful to programmers and engineers working with Freescale Semiconductor processors.

Additional literature is published as new processors become available. For a current list of HCS08 documentation, refer to http://www.freescale.com/.

Conventions

This document uses the following notational conventions:

cleared/set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
MNEMONICS	In text, instruction mnemonics are shown in uppercase.
mnemonics	In code and tables, instruction mnemonics are shown in lowercase.
italics	Italics indicate variable command parameters. Book titles in text are set in italics.
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
REG[FIELD]	Abbreviations for registers are shown in uppercase. Specific bits, fields, or ranges appear in brackets. For example, RAMBAR[BA] identifies the base address field in the RAM base address register.
nibble	A 4-bit data unit
byte	An 8-bit data unit
word	A 16-bit data unit[1]
x	In some contexts, such as signal encodings, x indicates a don't care.
n	Used to express an undefined numerical value
~	NOT logical operator
&	AND logical operator
|	OR logical operator
||	Field concatenation operator
$\overline{\text{OVERBAR}}$	An overbar indicates that a signal is active-low.

Register Figure Conventions

This document uses the following conventions for the register reset values:

—	Undefined at reset.

[1]The only exceptions to this appear in the discussion of serial communication modules that support variable-length data transmission units. To simplify the discussion these units are referred to as words regardless of length.

u   Unaffected by reset.

[*signal_name*]   Reset value is determined by the polarity of the indicated signal.

The following register fields are used:

R	0
W	

Indicates a reserved bit field in a memory-mapped register. These bits are always read as zeros.

R	1
W	

Indicates a reserved bit field in a memory-mapped register. These bits are always read as ones.

R	FIELDNAME
W	

Indicates a read/write bit.

R	FIELDNAME
W	

Indicates a read-only bit field in a memory-mapped register.

R	
W	FIELDNAME

Indicates a write-only bit field in a memory-mapped register.

R	FIELDNAME
W	w1c

Write 1 to clear: indicates that writing a 1 to this bit field clears it.

R	0
W	FIELDNAME

Indicates a self-clearing bit.

Chapter 1
Device Overview

1.1   Introduction

The MC9S08JE128 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.

The following table summarizes the peripheral availability per package type for the devices available in the MC9S08JE128 series.

Table 1-1. MC9S08JE128 series Features by MCU and Package

Feature	MC9S08JE128			MC9S08JE64
Pin quantity	81	80	64	64
FLASH size (bytes)	131072			65535
RAM size (bytes)	12K			12K
Programmable Analog Comparator (PRACMP)	yes			yes
Debug Module (DBG)	yes			yes
Multipurpose Clock Generator (MCG)	yes			yes
Inter-Integrated Communication (IIC)	yes			yes
Interrupt Request Pin (IRQ)	yes			yes
Keyboard Interrupt (KBI)	16	16	6	6
Port I/O[1]	47	46	33	33
Dedicated Analog Input Pins	12			12
Power and Ground Pins	8			8
Time Of Day (TOD)	yes			yes
Serial Communications (SCI1)	yes			yes
Serial Communications (SCI2)	yes			yes
Serial Peripheral Interface 1 (SPI1 (FIFO))	yes			yes
Serial Peripheral Interface 2 (SPI2)	yes			yes
Carrier Modulator Timer pin (IRO)	yes			yes
TPM input clock pin (TPMCLK)	yes			yes

Table 1-1. MC9S08JE128 series Features by MCU and Package (Continued)

Feature	MC9S08JE128			MC9S08JE64
TPM1 channels	4			4
TPM2 channels	4	4	2	2
XOSC1	yes			yes
XOSC2	yes			yes
USB	yes			yes
Programmable Delay Block (PDB)	yes			yes
SAR ADC differential channels[2]	4	4	3	3
SAR ADC single-ended channels	8	8	6	6
Voltage reference output pin (VREFO)	yes			yes

[1]   Port I/O count does not include two (2) output-only and one (1) input-only pins.

[2]   Each differential channel is comprised of 2 pin inputs.

1.2   MCU Block Diagram

The block diagram in Figure 1-1 shows the structure of the MC9S08JE128 series MCU.
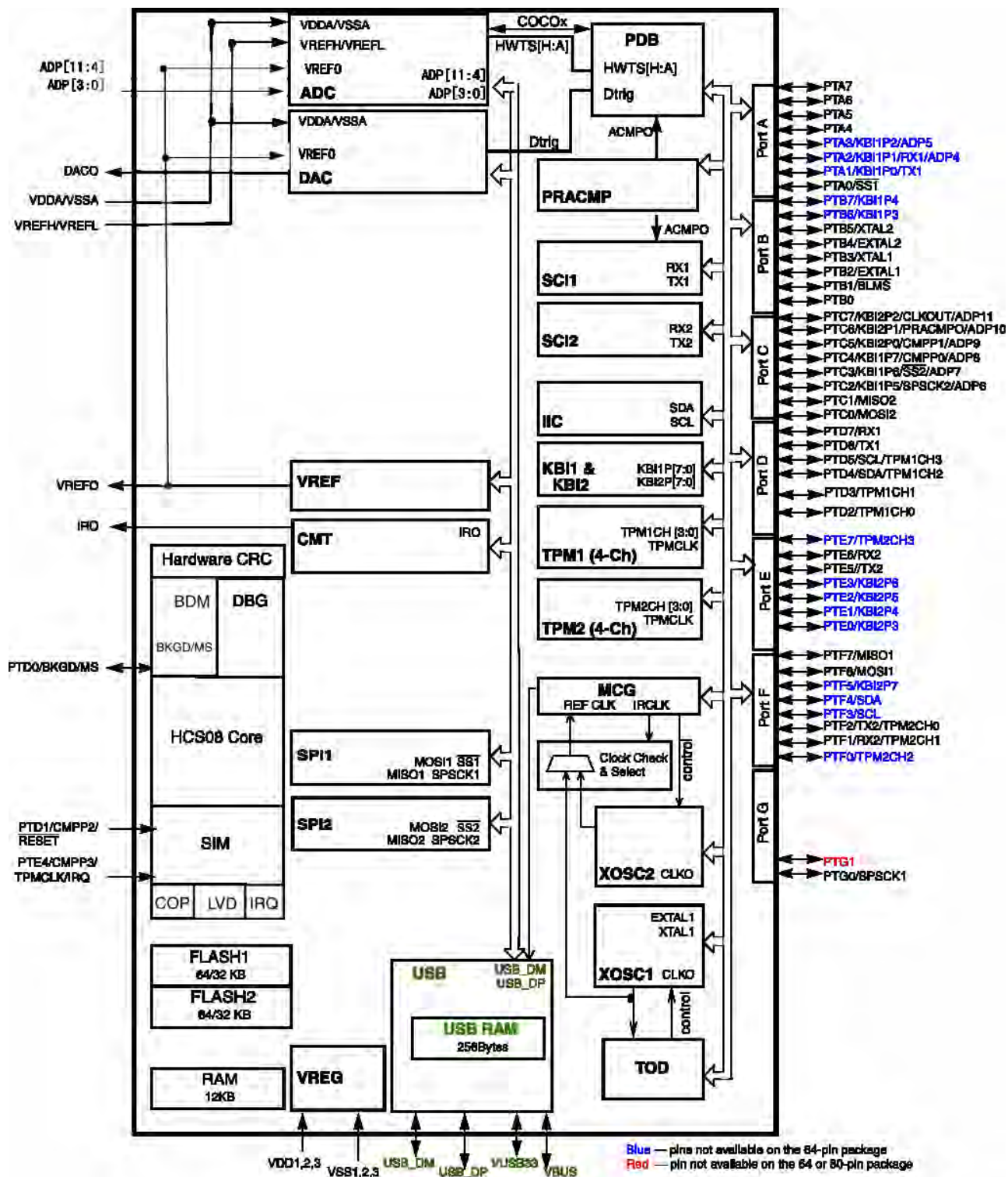
Figure 1-1. MC9S08JE128 series Block Diagram

The following table lists the functional versions of the on-chip modules.

Table 1-2. Versions of On-Chip Modules

Module	Version
Analog-to-Digital Converter (ADC12)	1
Digital to Analog Converter (DAC)	1
Programmable Delay Block	1
Inter-Integrated Circuit (IIC)	3
Central Processing Unit (CPU)	5
On-Chip In-Circuit Debug/Emulator (DBG)	3
Multi-Purpose Clock Generator (MCG)	3
Low Power Oscillator (XOSCVLP)	1
Carrier Modulator Timer (CMT)	1
Programable Analog Comparator (PRACMP)	1
Serial Communications Interface (SCI)	4
Serial Peripheral Interface (SPI)	5
Time of Day (TOD)	1
Universal Serial Bus (USB)	1
Timer Pulse-Width Modulator (TPM)	3
System Integration Module (SIM)	1
Cyclic Redundancy Check (CRC)	3
Keyboard Interrupt (KBI)	2
Voltage Reference (VREF)	1
Voltage Regulator (VREG)	1
Interrupt Request (IRQ)	3
Flash Wrapper	1
GPIO	2
Port Control	1

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.
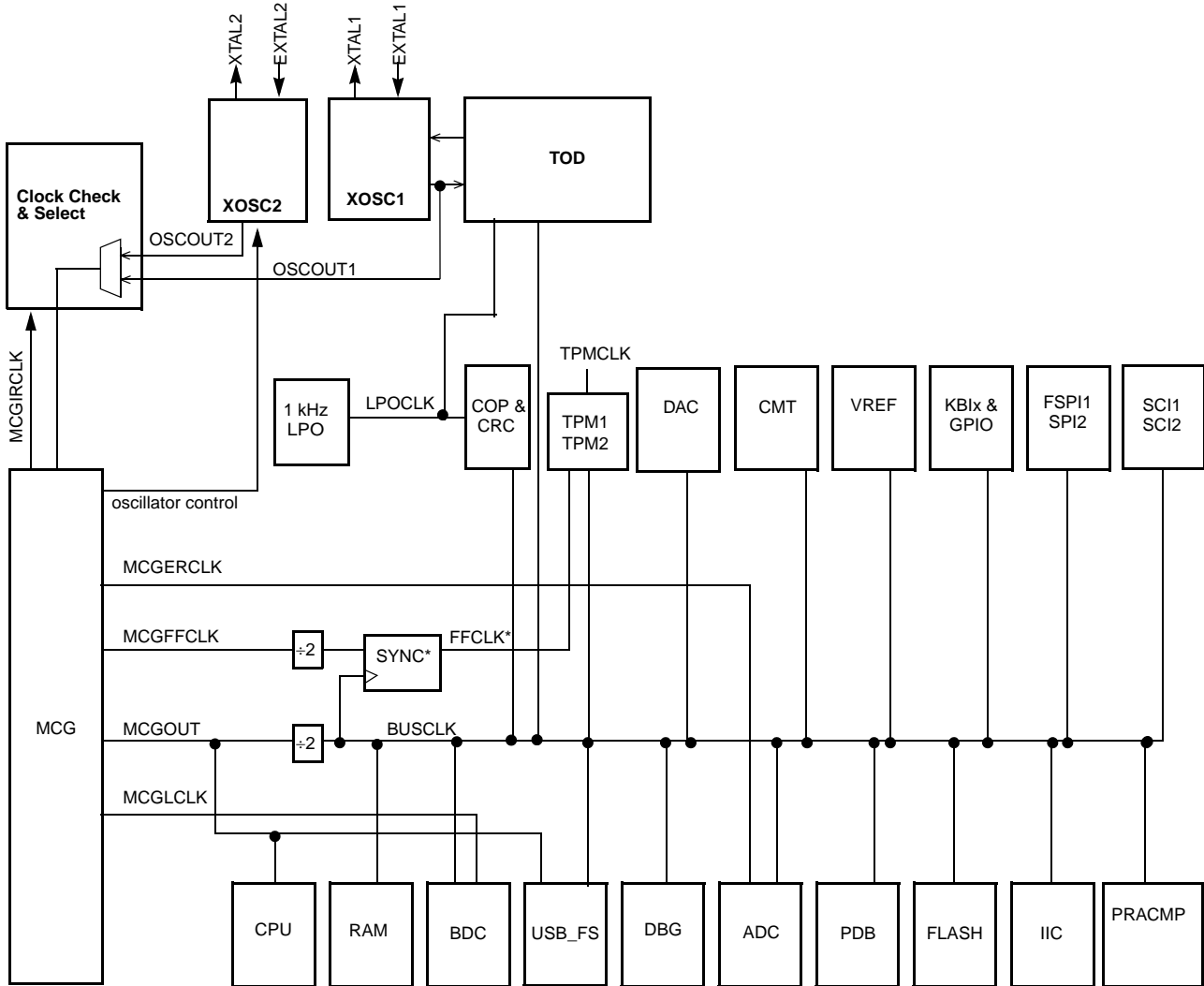
Note: The ADC has minimum and maximum frequency requirements. See the ADC chapter and the *MC9S08JE128 Data Sheet.* Flash memory has frequency requirements for program and erase operations. Each ADC also has it's own internal asynchronous clock source, which is not shown above.

* The fixed frequency clock (FFCLK) is internally synchronized to the bus clock (BUSCLK) and must not exceed one half of the bus clock frequency.

Figure 1-2. System Clock Distribution Diagram

1.3.1 System Clocks

Table 1-3 describes each of the system clocks.

Table 1-3. System Clocks

Clock	Description
MCGOUT	This clock source is used as the CPU clock and is divided by two to generate the peripheral bus clock. Control bits in the MCG control registers determine which of three clock sources is connected: • Internal reference clock • External reference clock • Frequency-locked loop (FLL) or phase-locked loop (PLL) output This clock drives the CPU, debug, RAM, and BDM directly and is divided by two to clock all peripherals (BUSCLK). See Chapter 15, "Multipurpose Clock Generator (S08MCGV3)," 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.
MCGERCLK	**MCG External Reference Clock**—This is the external reference clock and can be selected as the alternate clock for the ADC.
MCGIRCLK	**MCG Internal Reference Clock**—This is the internal reference clock and can be selected as the TOD clock source.
MCGFFCLK	**MCG Fixed-Frequency Clock**—This clock is divided by 2 to generate the fixed frequency clock (FFCLK) after being synchronized to the bus clock. It can be selected as clock source for the TPM modules. The frequency of the FFCLK is determined by the settings of the MCG.
LPOCLK	**Low-power Oscillator Clock**—This clock is generated from an internal low-power oscillator that is completely independent of the MCG module. The LPOCLK can be selected as the clock source to the TOD or COP.
TPMCLK	**TPM Clock**—An optional external clock source for the TPMs. This clock must be limited to one-quarter the frequency of the bus clock for synchronization.
ADACK (not shown)	The ADC module also has an internally generated asynchronous clock that allows it to run in STOP mode (ADACK). This signal is not available externally.
OSCOUT1	Low-power crystal oscillator output that can be used as the reference clock to the MCG or the TOD
OSCOUT2	Low-power crystal oscillator output that can be used as the reference clock to the MCG

1.3.2 Clock Gating

To save power, peripheral clocks can be shut off by programming the system clock gating registers. For details, refer to Section 5.7.8, "System Clock Gating Control 1 Register (SCGC1)".

1.3.3 MCG Modes of Operation

The MCG operates in one of the modes described in Table 1-4.

Table 1-4. MCG Modes

Mode	Related field values[1]	Description
FLL Engaged Internal (FEI)	• MCGC1[IREFS] = 1 • MCGC1[CLKS] = 00 • MCGC3[PLLS] = 0	Default. The MCG supplies a clock derived from one of the on-chip FLLs, which is sourced by the internal reference clock. Upon exiting reset, the default FLL is that which generates the x MHz bus / y MHz CPU clocks.
FLL Engaged External (FEE)	• MCGC1[IREFS] = 0 • MCGC1[CLKS] = 00 • MCGC3[PLLS] = 0	The MCG supplies a clock derived from the FLL, which is sourced from an external reference clock (or crystal oscillator).
FLL Bypassed Internal (FBI)	• MCGC1[IREFS] = 1 • MCGC1[CLKS] = 01 • MCGC3[PLLS] = 0 • XCSM[ENBDM] = 1 or MCGC2[LP] = 0	The FLL is enabled and sourced by the internal reference clock but is bypassed. The MCGOUT clock is derived from the internal reference clock.
FLL Bypassed External (FBE)	• MCGC1[IREFS] = 0 • MCGC1[CLKS] = 10 • MCGC3[PLLS] = 0 • XCSM[ENBDM] = 1 or MCGC2[LP] = 0	The FLL is enabled and controlled by an external reference clock but is bypassed. The MCGOUT clock is derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an XOSC1 or XOSC2 controlled by the MCG, or it can be another external clock source.
PLL Engaged External (PEE)	• MCGC1[IREFS] = 0 • MCGC1[CLKS] = 00 • MCGC3[PLLS] = 1	The MCG supplies a clock derived from the PLL, which is sourced from an external reference clock (or crystal oscillator).
PLL Bypassed External (PBE)	• MCGC1[IREFS] = 0 • MCGC1[CLKS] = 10 • MCGC3[PLLS] = 1 • XCSM[ENBDM] = 1 or MCGC2[LP] = 0	The MCG supplies a clock MCGOUT derived from the external reference clock (or crystal oscillator). The PLL is also sourced from the external clock source but is bypassed.
Bypassed Low Power Internal (BLPI)	• MCGC1[IREFS] = 1 • MCGC1[CLKS] = 01 • XCSM[ENBDM] = 0 and MCGC2[LP] = 1	The FLL and PLL are disabled and bypassed, and the MCG supplies MCGOUT derived from the internal reference clock.
Bypassed Low Power External (BLPE)	• MCGC1[IREFS] = 0 • MCGC1[CLKS] = 10 • XCSM[ENBDM] = 0 and MCGC2[LP] = 1	The FLL and PLL are disabled and bypassed, and the MCG supplies MCGOUT derived from the external reference clock.
STOP	—	The FLL and PLL are disabled, and the internal or external reference clocks can be selected to be enabled or disabled. The microcontroller does not provide a microcontroller clock source unless BDC[ENBDM] is enabled.

[1] For descriptions of the MCGC1, MCGC2, and MCGC3 registers, see Chapter 15, "Multipurpose Clock Generator (S08MCGV3)".

1.3.4 MCG Mode State Diagram

shows the valid state transitions for the MCG.

The IREFS and CLKS fields are contained within the MCG module definition. The LP bit is part of the on-chip power management controller (PMC) block.

The clock source for the BDC is controlled by the BDC clksw bit. Choices for the BDC clock are MCGOUT and the output from DCO.
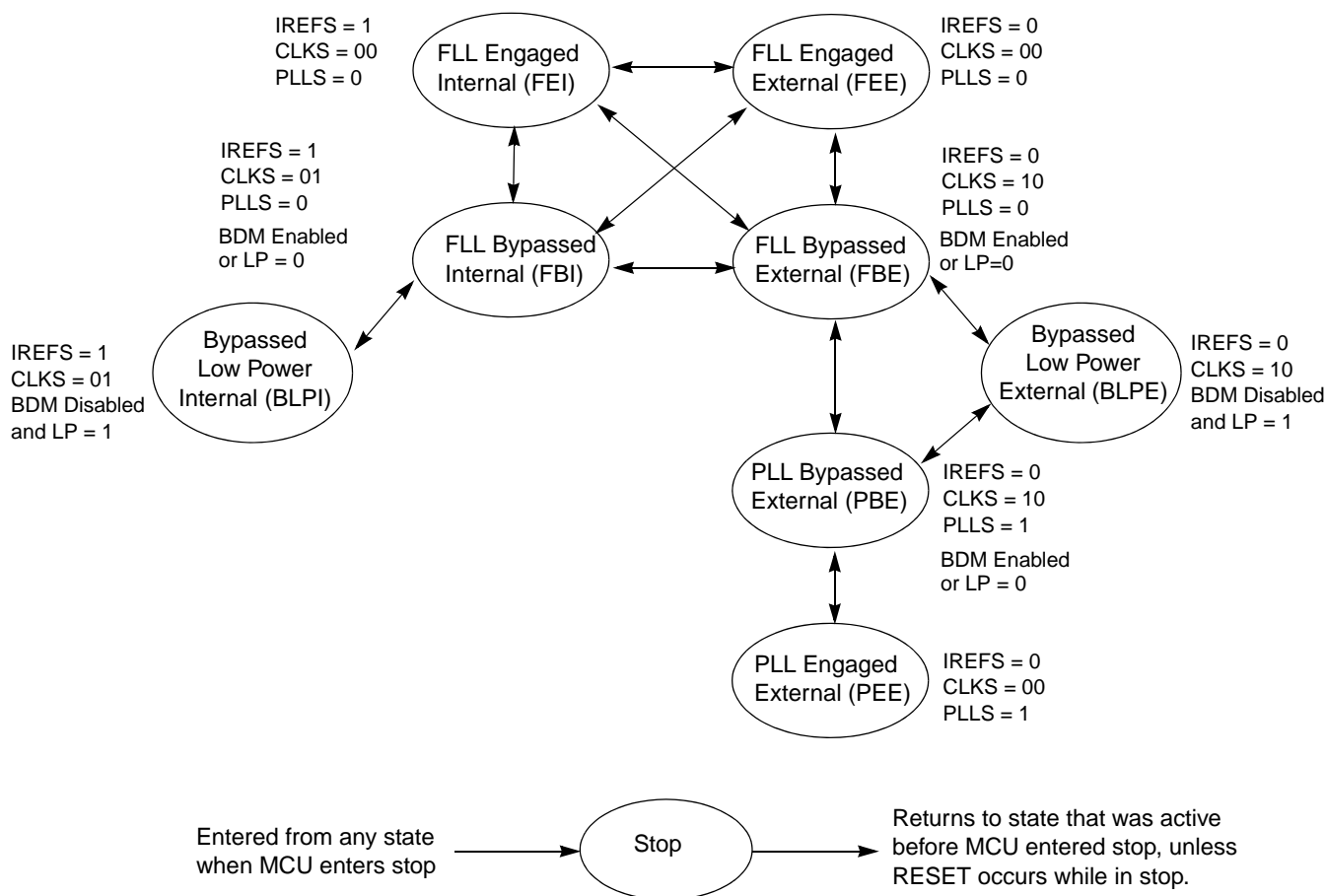
Figure 1-3. MCG Mode State Diagram

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

2.2.1   64-Pin LQFP

The following two figures show the 64-pin LQFP pinout configuration.
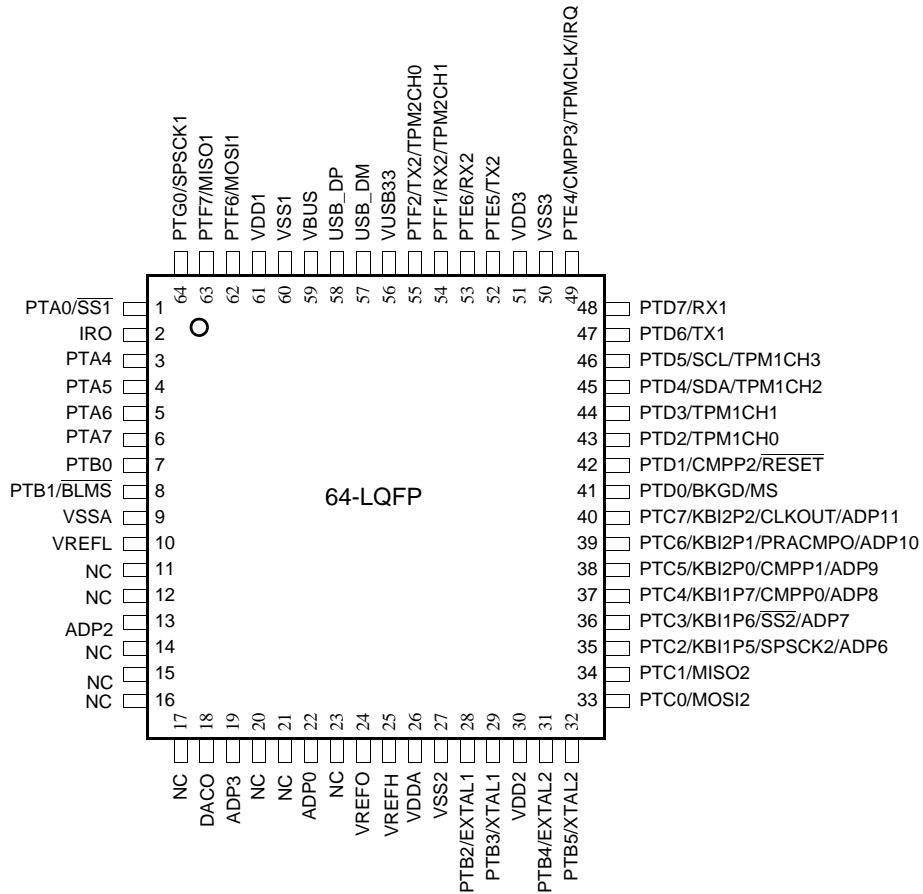
Figure 2-1. 64-Pin LQFP

NOTE

All NC pins should be unconnected and floating.

2.2.2 80-Pin LQFP

The following figure shows the 80-pin LQFP pinout configuration.
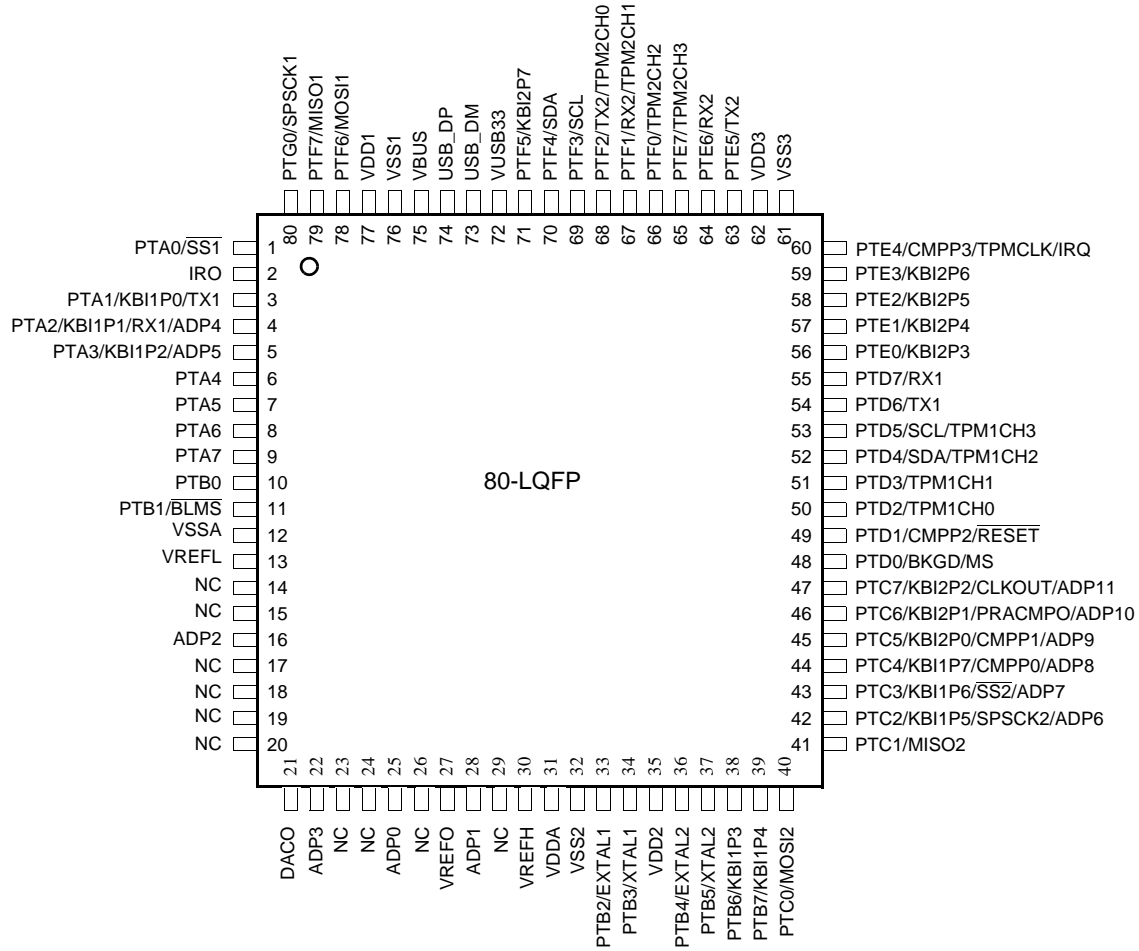
Figure 2-2. 80-Pin LQFP

NOTE

All NC pins should be unconnected and floating.

2.2.3 81-Pin MAPBGA

The following figure shows the 81-pin MAPBGA pinout configuration.

	1	2	3	4	5	6	7	8	9
A	IRO	PTG0	PTF6	USB_DP	VBUS	VUSB33	PTF4	PTF3	PTE4
B	PTF7	PTA0	PTG1	USB_DM	PTF5	PTE7	PTF1	PTF0	PTE3
C	PTA4	PTA5	PTA6	PTA1	PTF2	PTE6	PTE5	PTE2	PTE1
D		PTA7	PTB0	PTB1	PTA2	PTA3	PTD5	PTD7	PTE0
E				VDD2	VDD3	VDD1	PTD2	PTD3	PTD6
F		ADP2		VSS2	VSS3	VSS1	PTB7	PTC7	PTD4
G	ADP0	DACO	ADP3		VREFO	PTB6	PTC0	PTC1	PTC2
H			ADP1		PTC3	PTC4	PTD0	PTC5	PTC6
J	VSSA	VREFL	VREFH	VDDA	PTB2	PTB3	PTD1	PTB4	PTB5

Figure 2-3. 81-Pin MAPBGA

NOTE

All NC pins (indicated with blue shading) should be unconnected and floating.

2.2.4 Pin Assignments

Table 2-1 shows the package pin assignments.

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.

Table 2-1. Package Pin Assignments

Package			Default Function	ALT1	ALT2	ALT3	Composite Pin Name
81 MAPBGA	80 LQFP	64 LQFP					
B2	1	1	PTA0	$\overline{SS1}$	—	—	PTA0/$\overline{SS1}$
A1	2	2	IRO	—	—	—	IRO
C4	3	—	PTA1	KBI1P0	TX1	—	PTA1/KBI1P0/TX1
D5	4	—	PTA2	KBI1P1	RX1	ADP4	PTA2/KBI1P1/RX1/ADP4
D6	5	—	PTA3	KBI1P2	ADP5	—	PTA3/KBI1P2/ADP5
C1	6	3	PTA4	—	—	—	PTA4
C2	7	4	PTA5	—	—	—	PTA5
C3	8	5	PTA6	—	—	—	PTA6
D2	9	6	PTA7	—	—	—	PTA7
D3	10	7	PTB0	—	—	—	PTB0
D4	11	8	PTB1	$\overline{BLMS}$	—	—	PTB1/$\overline{BLMS}$
J1	12	9	VSSA	—	—	—	VSSA
J2	13	10	VREFL	—	—	—	VREFL
D1	14	11	NC	—	—	—	NC
E1	15	12	NC	—	—	—	NC
F2	16	13	ADP2	—	—	—	ADP2
F1	17	14	NC	—	—	—	NC
E2	18	15	NC	—	—	—	NC
F3	19	16	NC	—	—	—	NC

Table 2-1. Package Pin Assignments (Continued)

Package			Default Function	ALT1	ALT2	ALT3	Composite Pin Name
81 MAPBGA	80 LQFP	64 LQFP					
E3	20	17	NC	—	—	—	NC
G2	21	18	DACO	—	—	—	DACO
G3	22	19	ADP3	—	—	—	ADP3
H4	23	20	NC	—	—	—	NC
G4	24	21	NC	—	—	—	NC
G1	25	22	ADP0	—	—	—	ADP0
H1	26	23	NC	—	—	—	NC
G5	27	24	VREFO	—	—	—	VREFO
H3	28	—	ADP1	—	—	—	ADP1
H2	29	—	NC	—	—	—	NC
J3	30	25	VREFH	—	—	—	VREFH
J4	31	26	VDDA	—	—	—	VDDA
F4	32	27	VSS2	—	—	—	VSS2
J5	33	28	PTB2	EXTAL1	—	—	PTB2/EXTAL1
J6	34	29	PTB3	XTAL1	—	—	PTB3/XTAL1
E4	35	30	VDD2	—	—	—	VDD2
J8	36	31	PTB4	EXTAL2	—	—	PTB4/EXTAL2
J9	37	32	PTB5	XTAL2	—	—	PTB5/XTAL2
G6	38	—	PTB6	KBI1P3	—	—	PTB6/KBI1P3
F7	39	—	PTB7	KBI1P4	—	—	PTB7/KBI1P4
G7	40	33	PTC0	MOSI2	—	—	PTC0/MOSI2
G8	41	34	PTC1	MISO2	—	—	PTC1/MISO2
G9	42	35	PTC2	KBI1P5	SPSCK2	ADP6	PTC2/KBI1P5/SPSCK2/ADP6
H5	43	36	PTC3	KBI1P6	$\overline{SS2}$	ADP7	PTC3/KBI1P6/$\overline{SS2}$/ADP7
H6	44	37	PTC4	KBI1P7	CMPP0	ADP8	PTC4/KBI1P7/CMPP0/ADP8
H8	45	38	PTC5	KBI2P0	CMPP1	ADP9	PTC5/KBI2P0/CMPP1/ADP9
H9	46	39	PTC6	KBI2P1	PRACMPO	ADP10	PTC6/KBI2P1/PRACMPO/ADP10
F8	47	40	PTC7	KBI2P2	CLKOUT	ADP11	PTC7/KBI2P2/CLKOUT/ADP11
H7	48	41	PTD0	BKGD	MS	—	PTD0/BKGD/MS

Table 2-1. Package Pin Assignments (Continued)

Package			Default Function	ALT1	ALT2	ALT3	Composite Pin Name
81 MAPBGA	80 LQFP	64 LQFP					
J7	49	42	PTD1	CMPP2	RESET	—	PTD1/CMPP2/RESET
E7	50	43	PTD2	TPM1CH0	—	—	PTD2TPM1CH0
E8	51	44	PTD3	TPM1CH1	—	—	PTD3/TPM1CH1
F9	52	45	PTD4	SDA	TPM1CH2	—	PTD4/SDA/TPM1CH2
D7	53	46	PTD5	SCL	TPM1CH3	—	PTD5/SCL/TPM1CH3
E9	54	47	PTD6	TX1	—	—	PTD6/TX1
D8	55	48	PTD7	RX1	—	—	PTD7/RX1
D9	56	—	PTE0	KBI2P3	—	—	PTE0/KBI2P3
C9	57	—	PTE1	KBI2P4	—	—	PTE1/KBI2P4
C8	58	—	PTE2	KBI2P5	—	—	PTE2/KBI2P5
B9	59	—	PTE3	KBI2P6	—	—	PTE3/KBI2P6
A9	60	49	PTE4	CMPP3	TPMCLK	IRQ	PTE4/CMPP3/TPMCLK/IRQ
F5	61	50	VSS3	—	—	—	VSS3
E5	62	51	VDD3	—	—	—	VDD3
C7	63	52	PTE5	TX2	—	—	PTE5/TX2
C6	64	53	PTE6	RX2	—	—	PTE6/RX2
B6	65	—	PTE7	TPM2CH3	—	—	PTE7/TPM2CH3
B8	66	—	PTF0	TPM2CH2	—	—	PTF0/TPM2CH2
B7	67	54	PTF1	RX2	TPM2CH1	—	PTF1/RX2/TPM2CH1
C5	68	55	PTF2	TX2	TPM2CH0	—	PTF2/TX2/TPM2CH0
A8	69	—	PTF3	SCL	—	—	PTF3/SCL
A7	70	—	PTF4	SDA	—	—	PTF4/SDA
B5	71	—	PTF5	KBI2P7	—	—	PTF5/KBI2P7
A6	72	56	VUSB33	—	—	—	VUSB33
B4	73	57	USB_DM	—	—	—	USB_DM
A4	74	58	USB_DP	—	—	—	USB_DP
A5	75	59	VBUS	—	—	—	VBUS
F6	76	60	VSS1	—	—	—	VSS1
E6	77	61	VDD1	—	—	—	VDD1

Table 2-1. Package Pin Assignments (Continued)

Package			Default Function	ALT1	ALT2	ALT3	Composite Pin Name
81 MAPBGA	80 LQFP	64 LQFP					
A3	78	62	PTF6	MOSI1	—	—	PTF6/MOSI1
B1	79	63	PTF7	MISO1	—	—	PTF7/MISO1
A2	80	64	PTG0	SPSCK1	—	—	PTG0/SPSCK1
B3	—	—	PTG1	—	—	—	PTG1

2.2.5   Pinout Summary

The following tables identify pin options on a peripheral-by-peripheral basis.

Table 2-2. PTA Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	PTA
B2	1	1	PTA0	PTA0/$\overline{SS1}$	PTA0
C4	3	—	PTA1	PTA1/KBI1P0/TX1	PTA1
D5	4	—	PTA2	PTA2/KBI1P1/RX1/ADP4	PTA2
D6	5	—	PTA3	PTA3/KBI1P2/ADP5	PTA3
C1	6	3	PTA4	PTA4	PTA4
C2	7	4	PTA5	PTA5	PTA5
C3	8	5	PTA6	PTA6	PTA6
D2	9	6	PTA7	PTA7	PTA7

Table 2-3. PTB Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	PTB
D3	10	7	PTB0	PTB0	PTB0
D4	11	8	PTB1	PTB1/$\overline{BLMS}$	PTB1
J5	33	28	PTB2	PTB2/EXTAL1	PTB2
J6	34	29	PTB3	PTB3/XTAL1	PTB3
J8	36	31	PTB4	PTB4/EXTAL2	PTB4
J9	37	32	PTB5	PTB5/XTAL2	PTB5
G6	38	—	PTB6	PTB6/KBI1P3	PTB6
F7	39	—	PTB7	PTB7/KBI1P4	PTB7

Table 2-4. PTC Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	PTC
G7	40	33	PTC0	PTC0/MOSI2	PTC0
G8	41	34	PTC1	PTC1/MISO2	PTC1
G9	42	35	PTC2	PTC2/KBI1P5/SPSCK2/ADP6	PTC2
H5	43	36	PTC3	PTC3/KBI1P6/$\overline{SS2}$/ADP7	PTC3
H6	44	37	PTC4	PTC4/KBI1P7/CMPP0/ADP8	PTC4
H8	45	38	PTC5	PTC5/KBI2P0/CMPP1/ADP9	PTC5
H9	46	39	PTC6	PTC6/KBI2P1/PRACMPO/ADP10	PTC6
F8	47	40	PTC7	PTC7/KBI2P2/CLKOUT/ADP11	PTC7

Table 2-5. PTD Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	PTD
H7	48	41	PTD0	PTD0/BKGD/MS	PTD0
J7	49	42	PTD1	PTD1/CMPP2/$\overline{RESET}$	PTD1
E7	50	43	PTD2	PTD2/TPM1CH0	PTD2
E8	51	44	PTD3	PTD3/TPM1CH1	PTD3
F9	52	45	PTD4	PTD4/SDA/TPM1CH2	PTD4
D7	53	46	PTD5	PTD5/SCL/TPM1CH3	PTD5
E9	54	47	PTD6	PTD6/TX1	PTD6
D8	55	48	PTD7	PTD7/RX1	PTD7

Table 2-6. PTE Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	PTE
D9	56	—	PTE0	PTE0/KBI2P3	PTE0
C9	57	—	PTE1	PTE1/KBI2P4	PTE1
C8	58	—	PTE2	PTE2/KBI2P5	PTE2
B9	59	—	PTE3	PTE3/KBI2P6	PTE3
A9	60	49	PTE4[1]	PTE4/CMPP3/TPMCLK/IRQ	PTE4
C7	63	52	PTE5	PTE5/TX2	PTE5
C6	64	53	PTE6	PTE6/RX2	PTE6
B6	65	—	PTE7	PTE7/TPM2CH3	PTE7

[1] PTE4/CMPP3/TPMCLK/IRQ is limited to input-only for the port I/O function.

Table 2-7. PTF Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	PTF
B8	66	—	PTF0	PTF0/TPM2CH2	PTF0
B7	67	54	PTF1	PTF1/RX2/TPM2CH1	PTF1
C5	68	55	PTF2	PTF2/TX2/TPM2CH0	PTF2
A8	69	—	PTF3	PTF3/SCL	PTF3
A7	70	—	PTF4	PTF4/SDA	PTF4
B5	71	—	PTF5	PTF5/KBI2P7	PTF5
A3	78	62	PTF6	PTF6/MOSI1	PTF6
B1	79	63	PTF7	PTF7/MISO1	PTF7

Table 2-8. PTG Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	PTG
A2	80	64	PTG0	PTG0/SPSCK1	PTG0
B3	—	—	PTG1	PTG1	PTG1

Table 2-9. OSC1 Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	OSC1
J5	33	28	PTB2	PTB2/EXTAL1	EXTAL1
J6	34	29	PTB3	PTB3/XTAL1	XTAL1

Table 2-10. OSC2 Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	OSC2
J8	36	31	PTB4	PTB4/EXTAL2	EXTAL2
J9	37	32	PTB5	PTB5/XTAL2	XTAL2

Table 2-11. KBI1 Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	KBI1
C4	3	—	PTA1	PTA1/KBI1P0/TX1	KBI1P0
D5	4	—	PTA2	PTA2/KBI1P1/RX1/ADP4	KBI1P1

Table 2-11. KBI1 Pinout Summary (Continued)

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	KBI1
D6	5	—	PTA3	PTA3/KBI1P2/ADP5	KBI1P2
G6	38	—	PTB6	PTB6/KBI1P3	KBI1P3
F7	39	—	PTB7	PTB7/KBI1P4	KBI1P4
G9	42	35	PTC2	PTC2/KBI1P5/SPSCK2/ADP6	KBI1P5
H5	43	36	PTC3	PTC3/KBI1P6/$\overline{SS2}$/ADP7	KBI1P6
H6	44	37	PTC4	PTC4/KBI1P7/CMPP0/ADP8	KBI1P7

Table 2-12. KBI2 Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	KBI2
H8	45	38	PTC5	PTC5/KBI2P0/CMPP1/ADP9	KBI2P0
H9	46	39	PTC6	PTC6/KBI2P1/PRACMPO/ADP10	KBI2P1
F8	47	40	PTC7	PTC7/KBI2P2/CLKOUT/ADP11	KBI2P2
D9	56	—	PTE0	PTE0/KBI2P3	KBI2P3
C9	57	—	PTE1	PTE1/KBI2P4	KBI2P4
C8	58	—	PTE2	PTE2/KBI2P5	KBI2P5
B9	59	—	PTE3	PTE3/KBI2P6	KBI2P6
B5	71	—	PTF5	PTF5/KBI2P7	KBI2P7

Table 2-13. SPI Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	SPI1
B1	79	63	PTF7	PTF7/MISO1	MISO1
A3	78	62	PTF6	PTF6/MOSI1	MOSI1
A2	80	64	PTG0	PTG0/SPSCK1	SPSCK1
B2	1	1	PTA0	PTA0/$\overline{SS1}$	$\overline{SS1}$

Table 2-14. SPI2 Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	SPI2
G8	41	34	PTC1	PTC1/MISO2	MISO2
G7	40	33	PTC0	PTC0/MOSI2	MOSI2
G9	42	35	PTC2	PTC2/KBI1P5/SPSCK2/ADP6	SPSCK2
H5	43	36	PTC3	PTC3/KBI1P6/$\overline{SS2}$/ADP7	$\overline{SS2}$

Table 2-15. IIC Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	IIC
D7	53	46	PTD5	PTD5/SCL/TPM1CH3	SCL
A8	69	—	PTF3	PTF3/SCL	SCL
F9	52	45	PTD4	PTD4/SDA/TPM1CH2	SDA
A7	70	—	PTF4	PTF4/SDA	SDA

Table 2-16. Other Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	Other
H7	48	41	PTD0	PTD0/BKGD/MS	BKGD/MS
F8	47	40	PTC7	PTC7/KBI2P2/CLKOUT/ADP11	CLKOUT
A9	60	49	PTE4	PTE4/CMPP3/TPMCLK/IRQ	IRQ
J7	49	42	PTD1	PTD1/CMPP2/$\overline{\text{RESET}}$	$\overline{\text{RESET}}$

Table 2-17. VREF Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	VREF
G5	27	24	VREFO	VREFO	VREFO

Table 2-18. CMT Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	CMT
A1	2	2	IRO	IRO	IRO

Table 2-19. P/G Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	P/G
E6	77	61	VDD1	VDD1	VDD1
E4	35	30	VDD2	VDD2	VDD2
E5	62	51	VDD3	VDD3	VDD3
J4	31	26	VDDA	VDDA	VDDA
F6	76	60	VSS1	VSS1	VSS1
F4	32	27	VSS2	VSS2	VSS2
F5	61	50	VSS3	VSS3	VSS3
J1	12	9	VSSA	VSSA	VSSA

Table 2-20. USB Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	USB
B4	73	57	USB_DM	USB_DM	USB_DM
A4	74	58	USB_DP	USB_DP	USB_DP
A5	75	59	VBUS	VBUS	VBUS
A6	72	56	VUSB33	VUSB33	VUSB33

Table 2-21. SCI1 Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	SCI1
D5	4	—	PTA2	PTA2/KBI1P1/RX1/ADP4	RX1
D8	55	48	PTD7	PTD7/RX1	RX1
C4	3	—	PTA1	PTA1/KBI1P0/TX1	TX1
E9	54	47	PTD6	PTD6/TX1	TX1

Table 2-22. SCI2 Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	SCI2
C6	64	53	PTE6	PTE6/RX2	RX2
B7	67	54	PTF1	PTF1/RX2/TPM2CH1	RX2
C7	63	52	PTE5	PTE5/TX2	TX2
C5	68	55	PTF2	PTF2/TX2/TPM2CH0	TX2

Table 2-23. PRACMP Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	PRACMP
H6	44	37	PTC4	PTC4/KBI1P7/CMPP0/ADP8	CMPP0
H8	45	38	PTC5	PTC5/KBI2P0/CMPP1/ADP9	CMPP1
J7	49	42	PTD1	PTD1/CMPP2/$\overline{RESET}$	CMPP2
A9	60	49	PTE4	PTE4/CMPP3/TPMCLK/IRQ	CMPP3
H9	46	39	PTC6	PTC6/KBI2P1/PRACMPO/ADP10	PRACMPO

Table 2-24. ADC Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	ADC
H9	46	39	PTC6	PTC6/KBI2P1/PRACMPO/ADP10	ADP10
F8	47	40	PTC7	PTC7/KBI2P2/CLKOUT/ADP11	ADP11
D5	4	—	PTA2	PTA2/KBI1P1/RX1/ADP4	ADP4
D6	5	—	PTA3	PTA3/KBI1P2/ADP5	ADP5
G9	42	35	PTC2	PTC2/KBI1P5/SPSCK2/ADP6	ADP6
H5	43	36	PTC3	PTC3/KBI1P6/$\overline{SS2}$/ADP7	ADP7
H6	44	37	PTC4	PTC4/KBI1P7/CMPP0/ADP8	ADP8
H8	45	38	PTC5	PTC5/KBI2P0/CMPP1/ADP9	ADP9
H1	26	23	—	NC	—
H2	29	—	—	NC	—
E2	18	15	—	NC	—
G4	24	21	—	NC	—
G1	25	22	ADP0	ADP0	ADP0
H3	28	—	ADP1	ADP1	ADP1
F2	16	13	ADP2	ADP2	ADP2
G3	22	19	ADP3	ADP3	ADP3
J3	30	10	VREFH	VREFH	VREFH
J2	13	25	VREFL	VREFL	VREFL

Table 2-25. TPM1 Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	TPM1
E7	50	43	PTD2	PTD2/TPM1CH0	TPM1CH0
E8	51	44	PTD3	PTD3(D+)/TPM1CH1	TPM1CH1
F9	52	45	PTD4	PTD4/SDA/TPM1CH2	TPM1CH2
D7	53	46	PTD5	PTD5/SCL/TPM1CH3	TPM1CH3
A9	60	49	PTE4	PTE4/CMPP3/TPMCLK/IRQ	TPMCLK

Table 2-26. TPM2 Pinout Summary

81 MAPBGA	80 LQFP	64 LQFP	Default Function	Composite Pin Name	TPM2
C5	68	55	PTF2	PTF2/TX2/TPM2CH0	TPM2CH0
B7	67	54	PTF1	PTF1/RX2/TPM2CH1	TPM2CH1
B8	66	—	PTF0	PTF0/TPM2CH2	TPM2CH2
B6	65	—	PTE7	PTE7/TPM2CH3	TPM2CH3
A9	60	49	PTE4	PTE4/CMPP3/TPMCLK/IRQ	TPMCLK

2.3 Recommended System Connections

Figure 2-4 shows connections that are common to almost all MC9S08JE128 series application systems.
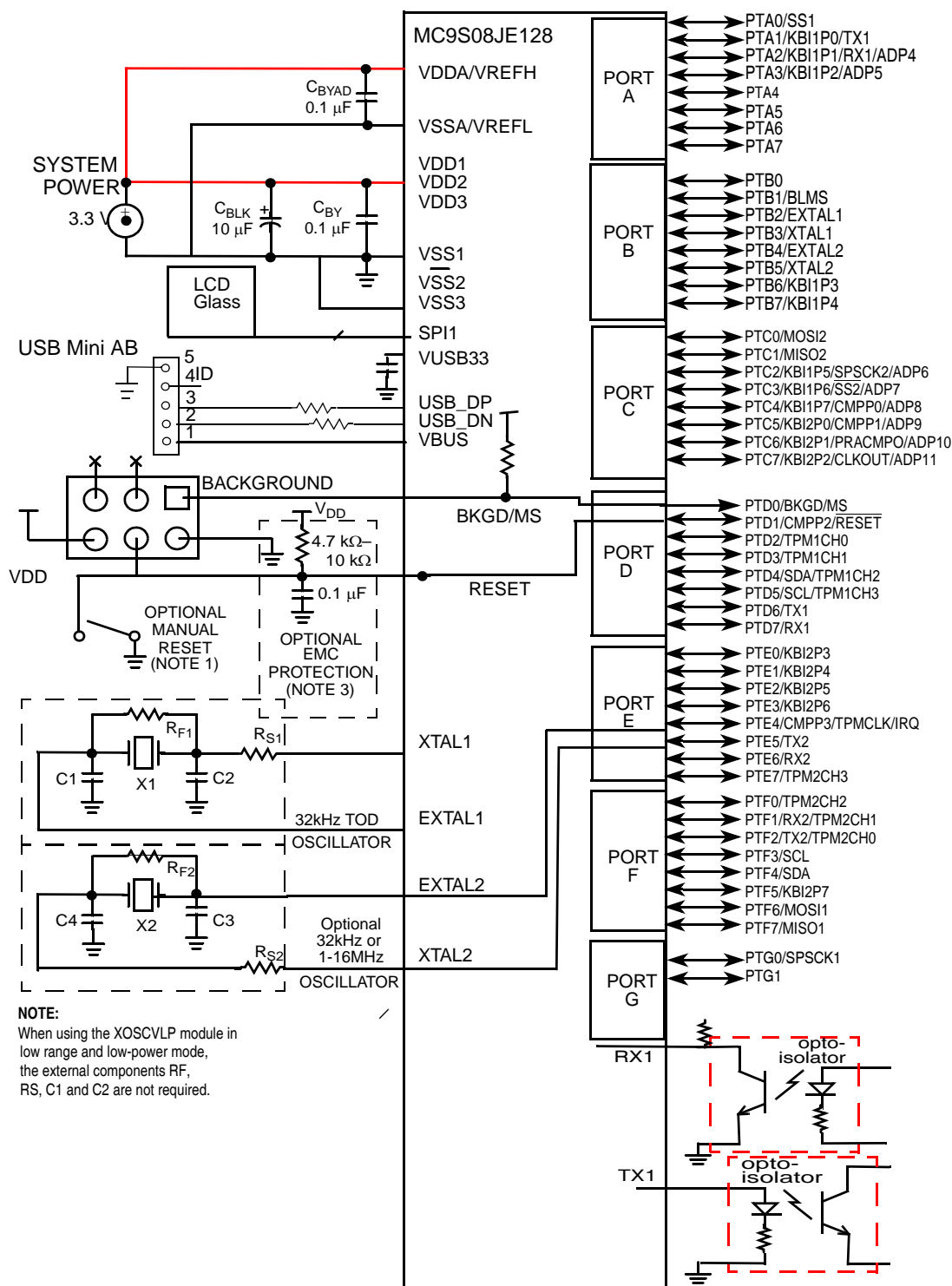
Figure 2-4. Recommended System Connections

2.3.1 Interfacing the SCIs to Off-Chip Opto-Isolators

SCI1 is designed with twice the normal I/O drive capability on the TX1 pin. The RX pin can either be fed directly from the digital I/O buffer, or those signals can be pre-conditioned using the comparators as shown in Figure 2-5 Similarly, the TX output can be modulated with the output of one of the timers before being passed off chip.
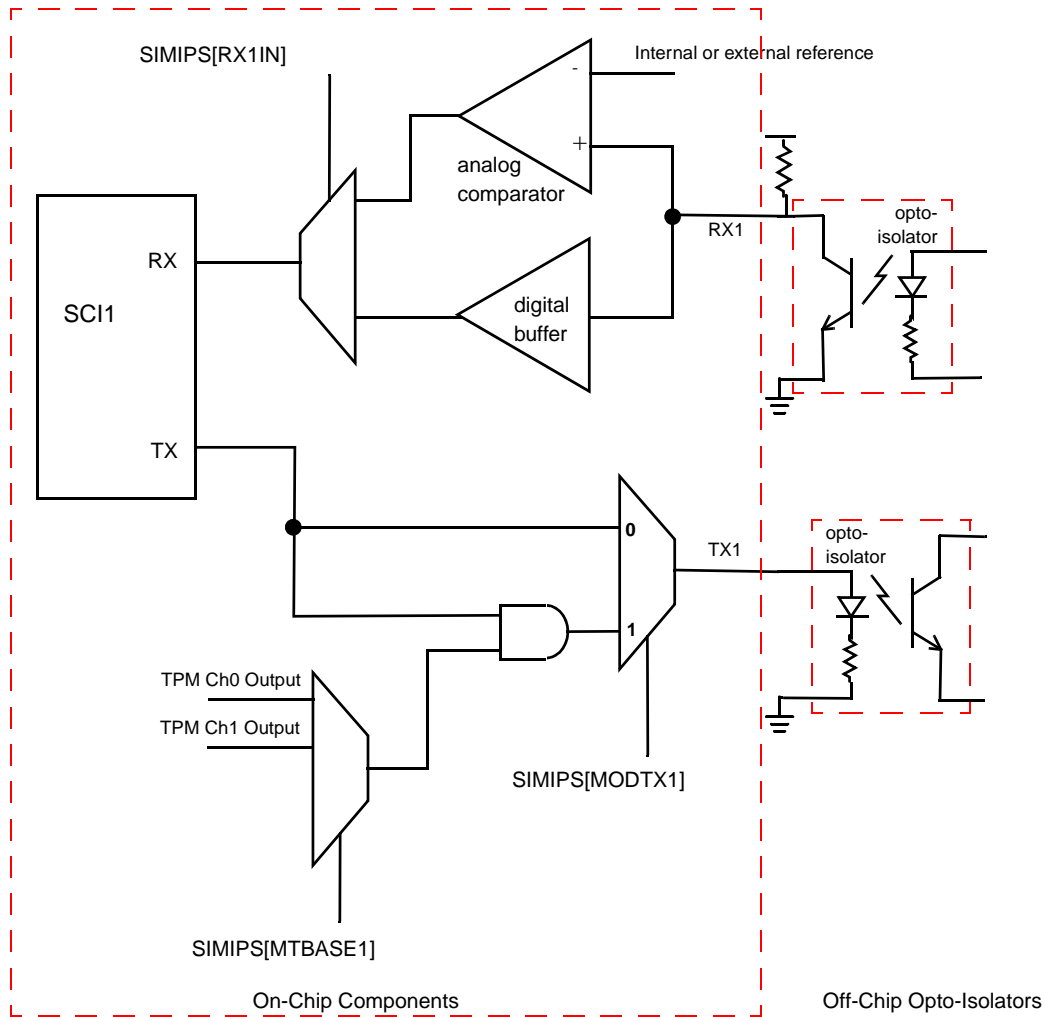
Figure 2-5. On-Chip Signal Conditioning Associated with SCI1 RX and TX Pins

Controls for the circuitry shown in Figure 2-5 are discussed in Section 5.7.14, "SIM Internal Peripheral Select Register (SIMIPS)."

2.3.2   Power

$V_{DD1,2,3}$ and $V_{SS1,2,3}$ are the primary power supply pins for the microcontroller. 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 microcontroller.

Typically, application systems have two separate capacitors across the power pins. In this case, there should 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 close to the microcontroller power pins as practical to suppress high-frequency noise. The MC9S08JE128 series has three V_{DD} pins. For the best noise suppression, connect each pin to a bypass capacitor.

V_{DDA} and V_{SSA} are the analog power supply pins for the microcontroller. This voltage source supplies power to the ADC and VREF modules. A 0.1 µF ceramic bypass capacitor should be located as close to the microcontroller power pins as practical to suppress high-frequency noise.

V_{USB33} maintains an output voltage of 3.3 V and sources enough current for the internal USB transceiver and USB pull-up resistor. The VBUS input supplies the voltage necessary to power the internal USB 3.3V regulator.

For V_{USB33}, connect two separate capacitors (4.7 µF bulk electrolytic stability capacitor and 0.47 µF ceramic bypass capacitors) across this pin to ground to decrease the output ripple of this voltage regulator when it is enabled.

2.3.3   Oscillator

Immediately after reset, the microcontroller uses an internally generated clock provided by the multipurpose clock generation (MCG) module.

The oscillator (XOSC1 or XOSC2) in this microcontroller is a Pierce oscillator that can accommodate a crystal or ceramic resonator. Optionally, an external clock source can be connected to the EXTAL input pin.

When using the oscillator module in low range and low-power mode, the external components RF, RS, C1 and C2 are not required.

For using the oscillator module in other modes, refer to Figure 2-4 for the following discussion. R_S (when used) and R_F should 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 should 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 microcontroller pin capacitance when selecting C1 and C2. The crystal manufacturer typically specifies a load capacitance that 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).

2.3.4   PTD1/CMPP2/$\overline{\text{RESET}}$

After a power-on reset (POR), the PTD1/CMPP2/$\overline{\text{RESET}}$ pin defaults to $\overline{\text{RESET}}$. Clearing RSTPE in SOPT1 allows the pin to be a GPIO pin or as an input to the PRACMP. When configured as a GPIO, the pin will remain as a GPIO until the next POR or LVD reset. When enabled, the $\overline{\text{RESET}}$ pin can be used to reset the MCU from an external source when the pin is driven low. 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. A manual external reset can be added by supplying a simple switch to ground (pull reset pin low to force a reset).

Whenever any non-POR reset is initiated (whether from an external signal or from an internal system), the enabled $\overline{\text{RESET}}$ pin is driven low for about 66bus cycles. The reset circuitry decodes the cause of reset and records it by setting a corresponding bit in the system 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.

2.3.5   PTE4/CMPP3/TPMCLK/IRQ

The IRQ pin is the input source for the IRQ interrupt. If the IRQ function is not enabled, this pin can be used for other functions. In EMC-sensitive applications, an external RC filter is recommended on the IRQ pin. See Figure 2-4 for an example.

NOTE

The voltage on the internally pulled up IRQ pin when measured is below V_{DD}. The internal gates connected to this pin are pulled to V_{DD}. If the IRQ pin is required to drive to a V_{DD} level, an external pull-up must be used.

2.3.6   Background / Mode Select (PTD0/BKGD/MS)

During a power-on-reset (POR) or background debug force reset (see bit BDFR in Section 5.7.3, "System Background Debug Force Reset Register (SBDFR)," for more information), the BKGD/MS pin functions as a mode select pin. Immediately after any reset, the pin functions as the background pin and can be used for background debug communication.

If the BKGD/MS pin is unconnected, the microcontroller enters normal operating mode at the rising edge of the internal reset after a POR or forced BDC reset. If a debug system is connected to the 6-pin standard background debug header, it can hold BKGD/MS low during a POR or immediately after issuing a background debug force reset[1], which forces the microcontroller to halt mode.

The BKGD/MS pin is used primarily for background debug controller (BDC) communications using a custom protocol that uses 16 clock cycles of the target microcontroller's BDC clock per bit time. The target

---

1. Specifically, BKGD must be held low through the first 16 cycles after deassertion of the internal reset.

microcontroller's BDC clock could be as fast as the bus clock rate, so there should never be any significant capacitance connected to the BKGD/MS pin that could interfere with background serial communications.

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

2.3.7   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.

2.3.8   Bootloader Mode Select (PTB1/$\overline{BLMS}$)

During a power-on-reset (POR), the CPU detects the state of the PTB1/$\overline{BLMS}$ pin that functions as a mode select pin. When the $\overline{BLMS}$ pin is held low and BKGD/MS is not pulled low, the CPU enters the bootloader mode. During a power-on-reset (POR), an internal pull-up device is automatically enabled in PTB1/$\overline{BLMS}$ pin. Immediately after reset rises the pin functions as a general-purpose output only pin and an internal pull-up device is automatically disabled.

2.3.9   USB Data Pins (USBDP, USBDN)

The USBDP (D+) and USBDN (D–) pins are the analog input/output lines to/from full-speed internal USB transceiver. An optional internal pull-up resistor for the USBDP pin, R_{PUDP}, is available. See Chapter 22, "Universal Serial Bus (S08USBV1)" for more details.

2.3.10   General-Purpose I/O and Peripheral Ports

The MC9S08JE128 series microcontrollers support up to 47 general-purpose I/O pins and two Output-only pins (PTB1 and PTD0) and one input-only pin (PTE4). I/O pins are shared with on-chip peripheral functions (timers, serial I/O, ADC, ACMP, 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 pull-up device. Immediately after reset, all of these pins are configured as high-impedance general-purpose inputs with internal pull-up devices enabled.

When an on-chip peripheral system is controlling a pin, data direction control bits determine what is read from the port data registers, even though the peripheral controls the pin direction via the pin's output buffer

enable. For information about controlling these pins as general-purpose I/O pins, see Chapter 6, "Parallel Input/Output."

NOTE

To avoid extra current drain from floating input pins, the reset initialization routine in the application program should enable on-chip pull-up devices or change the direction of unused or non-bonded pins to outputs so they do not float.

Chapter 3
Modes of Operation

3.1   Introduction

The operating modes of the MC9S08JE128 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
- Bootloader mode - Enables USB communications to external host for programming and erasing the Flash as an alternate to using the BDC.
- Run mode — CPU clocks can be run at full speed and the internal supply is fully regulated.
- LPRUN mode — CPU and peripheral clocks are restricted to 125kHz maximum and the internal voltage regulator is in standby
- Wait mode — CPU shuts down to conserve power; system clocks are running and full regulation is maintained
- LPWAIT mode — CPU shuts down to conserve power; peripheral clocks are restricted to 125kHz maximum and the internal voltage regulator is in standby
- Stop modes — System clocks are stopped and voltage regulator is in standby
    - Stop3 — All internal circuits are powered for fast recovery
    - Stop2 — Partial power down of internal circuits, RAM content is retained, I/O states held

3.3   Bootloader Mode

While executing out of the bootloader several qualification factors are examined to determine whether to continue executing bootloader code or begin executing user code. The bootloader ROM can be accessed in bootloader mode or user mode. This section describes the valid operations and protection mechanism in bootloader and user modes.

The following four items will be examined after each reset of the MCU.

- $\overline{\text{BLMS}}$ pin
- SIGNATURE semaphore byte
- Flash block CRC checksum
- CRC BYPASS byte

3.3.1 Entering Bootloader Mode

Bootloader mode can be entered in the following four conditions:

- When $\overline{\text{BLMS}}$ pin is low and BKGD/MS is not pulled low during power-on-reset (POR), the bootloader mode is entered directly with no other qualifications.
- When $\overline{\text{BLMS}}$ pin and BKGD/MS are high during power-on-reset (POR), a CHECKSUM BYPASS flash location is examined. If it is not equal to 0x00 or 0xFF, then the bootloader mode is entered.
- When $\overline{\text{BLMS}}$ pin and BKGD/MS are high during power-on-reset (POR), a CHECKSUM BYPASS flash location is examined. If it is equal to 0xFF, a flash CRC is calculated for the flash array and compared with a FLASHCRC 16-bit word. If the result does not match, then the bootloader mode is entered.
- After a reset (other than a power-on reset), the SIGNATURE semaphore byte is examined. If it is equal to 0xC3, then the bootloader mode is entered.

3.3.2 Entering User mode

User mode can be entered in the following three conditions:

- When $\overline{\text{BLMS}}$ pin and BKGD/MS are high during power-on-reset (POR), and the CHECKSUM BYPASS byte is equal to 0x00, the user mode is entered.
- When $\overline{\text{BLMS}}$ pin and BKGD/MS are high during power-on-reset (POR), and the CHECKSUM BYPASS byte is equal to 0xFF, a flash CRC is calculated for the flash array and compared with a FLASHCRC 16-bit word. If the result matches, the user mode is entered.
- When a reset occurs (other than a power-on reset), if the SIGNATURE semaphore is not equal to 0xC3, the user mode is entered.

3.3.3 Active Background Mode and Bootloader Mode Arbitrage

During POR, if both BKGD/MS and $\overline{\text{BLMS}}$ pins are low, active background mode is entered.

3.3.4 Bootloader Operation

This section describes the bootloader mechanism and bootloader flow chart.

The bootloader software is located in bootloader ROM. User can perform flash erasing and programming when:

- Bootloader mode is entered
- Flash block checksum that has been calculated and flash block checksum do not match after power-on reset
- SIGNATURE value in register matches

3.3.4.1 Flash Block Checksum

Upon power-on reset (POR), if BLMSS = 0 and the value of checksum bypass is 0xFF, the bootloader will calculate the flash checksum. The checksum is calculated for the Flash locations:

(1000:FB00)

The calculated checksum are verified with a checksum written to the two bytes of the flash (FLASHCRC). If the checksum matches, the previous bootloader operation was successful and the MCU jumps to the user code entry and starts to execute user code. If the checksum does not match, it jumps to bootloader entry to wait for commands.

Flash block checksum calculation uses 16-bit CRC.

3.3.4.2 SIGNATURE Semaphore Register

After a reset (other than a power-on reset), the bootloader verifies SIGNATURE semaphore register. If SIGNATURE = 0xC3, the MCU jumps to bootloader entry to wait for commands. If not, it jumps to the user code entry and starts to execute user code.

Users are required to provide a mechanism in their application code to set the SIGNATURE to 0xC3 and initiate a reset if they want to re-enter bootloader mode after a successful user code has been programmed. Alternatively, BKGD mode can be entered and SIGNATURE can be updated using BDM commands and reset initiated with BKGD pin high.

3.3.4.3 Flash Partial Erase Semaphore

The value of flash partial erase is programmed by the user. Only when flash partial erase is programmed to 0x00, can the partial erase flash array command be supported by bootloader.

The value of this byte is 0xFF when the device is shipped from Freescale.

Boot Mode Entry Flow Chart

Start

Jump to bootrom

Power-on reset?

Regular reset

NO

YES

SIGNATURE
=0xC3?

YES

NO

BLMSS=1?

NO

Checksum bypass
=0xFF?

YES

Calculate Flash
block checksum

Flash checksum
match?

YES

Jump to user code entry,
User code executed.

YES

NO

Bootloader Mode entered
Change SIGNATURE=0xC3

NO

Checksum bypass
=0x00?

NO

Change SIGNATURE=0xC3

YES

Jump to user code entry,
User code executed.

Initial bus frequency to
24 MHz,
Initial USB

Jump to bootloader entry

Waiting for CMD

Note: Only when FLASH PARTIAL ERASE=0x00,
this command is valid.

CMD=Partial erase?

YES

Put Pass/Fail on stack

NO

CMD=Mass erase?

YES

Put Pass/Fail on stack

NO

CMD=Program Flash?

YES

Put Pass/Fail on stack

NO

CMD=Reset?

NO

YES

Clear SIGNATURE

Figure 3-1. Bootloader Flow Chart

3.4 Run Mode

Run is the normal operating mode for the MC9S08JE128 series. This mode is selected after any internal reset including LVD and when both the BKGD/MS and $\overline{BLMS}$ pins are high after a POR exit or a BDC forced reset. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFE:0xFFFF after reset.

3.4.1 Low-power Run Mode (LPRun)

In the low-power run mode, the on-chip voltage regulator is put into its standby state. This state uses the minimum power consumption necessary for CPU functionality. Power consumption is most reduced by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGC1 and SCGC2 registers.

Before entering this mode, the following conditions must be met:

- BLPE is the selected clock mode for the MCG.
- The HGO bit in the MCGC2 register is clear.
- The bus frequency is less than 125 kHz.
- If enabled, the ADC must be configured to use the asynchronous clock source, ADACK, to meet the ADC minimum frequency requirements. The bandgap channel cannot be converted in low-power run mode.
- If enabled, the TOD must be configured to use an external clock source (OSCOUT) or the 1 kHz low-power oscillator.
- The LVDE or LVDSE bit in SPMSC1 register must be clear. LVD and LVW will automatically be disabled.
- Flash programming/erasing is not allowed.
- ACMP option to compare to internal bandgap reference is not allowed in LPRUN and LPWAIT.
- The MCU cannot be in active background mode.

Once these conditions are met, low-power run mode can be entered by setting the LPR bit in the SPMSC2 register.

To re-enter standard run mode, clear the LPR bit. The LPRS bit in the SPMSC2 register is a read-only status bit that can be used to determine if the regulator is in full regulation mode or not. When LPRS is '0', the regulator is in full regulation mode and the MCU can run at full speed in any clock mode.

3.4.1.1 Interrupts in Low-power Run Mode

Low-power run mode provides the option to return to full regulation if any interrupt occurs. This is done by setting the LPWUI bit in the SPMSC2 register. The MCG can then be set for full speed immediately in the interrupt service routine.

- If the LPWUI bit is clear, interrupts will be serviced in low-power run mode.
- If the LPWUI bit is set, LPR and LPRS bits will be cleared and interrupts will be serviced with the regulator in full regulation.

3.4.1.2 Resets in Low-power Run Mode

Any reset will exit low-power run mode, clear the LPR and LPRS bits, and return the device to normal run mode.

3.5 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 during POR or immediately after issuing a background debug force 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 MC9S08JE128 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 24, "Development Support."

3.6 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.

3.6.1 Low-power Wait Mode (LPWait)

Low-power wait mode is entered by executing a WAIT instruction while the MCU is in low-power run mode. In the low-power wait mode, the on-chip voltage regulator remains in its standby state (as in the low-power run mode). This state uses the minimum power consumption necessary for most modules to maintain functionality. Power consumption is most reduced by disabling the clocks to all unused peripherals by clearing the corresponding bits in the SCGC register.

The same restrictions on the low-power run mode apply to low-power wait mode.

3.6.1.1 Interrupts in Low-power Wait Mode

If the LPWUI bit is set when the WAIT instruction is executed, then the voltage regulator will return to full regulation when wait mode is exited. The MCG can be set for full speed immediately in the interrupt service routine.

If the LPWUI bit is clear when the WAIT instruction is executed, an interrupt will return the device to low-power run mode.

If the LPWUI bit is set when the WAIT instruction is executed, an interrupt will return the device to normal run mode with full regulation and the LPR and LPRS bits will be cleared.

3.6.1.2 Resets in Low-power Wait Mode

Any reset will exit low-power wait mode, clear the LPR and LPRS bits, and return the device to normal run mode.

3.7 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 15, "Multipurpose Clock Generator (S08MCGV3)," 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 1.8.1.1, "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.

3.7.1 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: $\overline{RESET}$ or IRQ.

NOTE

> IRQ 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 TOD 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.

3.7.2 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: TOD 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.

3.7.2.1 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.

3.7.2.2 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 24, "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.

3.7.3 Stop Modes in Low-power Run Mode

Stop2 mode cannot be entered from low-power run mode. If the PPDC bit is set, then the LPR bit cannot be set. Likewise, if the LPR bit is set, the PPDC bit cannot be set.

Stop3 mode can be entered from low-power run mode by executing the STOP instruction while in low-power run. Exiting stop3 with a reset will put the device back into normal run mode. If LPWUI is clear, interrupts will exit stop3 mode, return the device to low-power run mode, and then service the interrupt. If LPWUI is set, interrupts will exit stop3 mode, put the device into normal run mode, clear LPR and LPRS bits, and then service the interrupt.

3.8 Mode Selection

Several control signals are used to determine the current operating mode of the device. Table 3-2 shows the conditions for each of the device's operating modes.

Table 3-2. Power Mode Selections

Mode of Operation	BDCSCR BDM	SPMSC1 PMC		SPMSC2 PMC		CPU & Peripheral CLKs	Affects on Sub-System	
	ENBDM [1]	LVDE	LVDSE	LPR	PPDC		BDM Clock	Voltage Regulator
RUN mode	0	x	x	0	x	on. MCG in any mode.	off	on
		1	1	1				
	1	x	x	x			on	
LPRUN mode	0	0	x	1	0	low frequency required. MCG in BLPE mode only.	off	standby
		1	0					
WAIT mode — (Assumes WAIT instruction executed.)	0	x	x	0	x	CPU clock is off; peripheral clocks on. MCG state same as RUN mode.	off	on
		1	1	1				
	1	x	x	x			on	
LPWAIT mode — (Assumes WAIT instruction executed.)	0	0	x	1	0	CPU clock is off; peripheral clocks at low speed. MCG in BLPE mode.	off	standby
		1	0					
STOP3 — (Assumes STOPE bit is set and STOP instruction executed.) Note that STOP3 is used in place of STOP2 if the BDM or LVD is enabled.	0	0	x	x	0	MCG in STOP. MCGERCLK MCGIRCLK, and OSCOUT optionally on[2]	off	standby
	0	1	0	x	0		off	
	0	1	1	x	x		off	on - stop currents will be increased
	1	x	x	x	x	MCGLCLK still active.	on	

Table 3-2. Power Mode Selections (Continued)

Mode of Operation	BDCSCR BDM	SPMSC1 PMC		SPMSC2 PMC		CPU & Peripheral CLKs	Affects on Sub-System	
	ENBDM [1]	LVDE	LVDSE	LPR	PPDC		BDM Clock	Voltage Regulator
STOP2 — (Assumes STOPE bit is set and STOP instruction executed.) If BDM or LVD is enabled, STOP3 will be invoked rather than STOP2.	0	0	x	0	1	OSCOUT optionally on[2,3]	off	partial power down
		1	0					

[1] ENBDM is located in the BDC status and control register (BDCSCR) which is write accessible only through BDC commands.

[2] Configured within the MCG module based on the settings of IREFSTEN, EFRESTEN, IRCLKEN and ERCLKEN.

[3] In stop2, CPU, Flash, MCG and all peripheral modules are powered down except for the TOD and LCD.
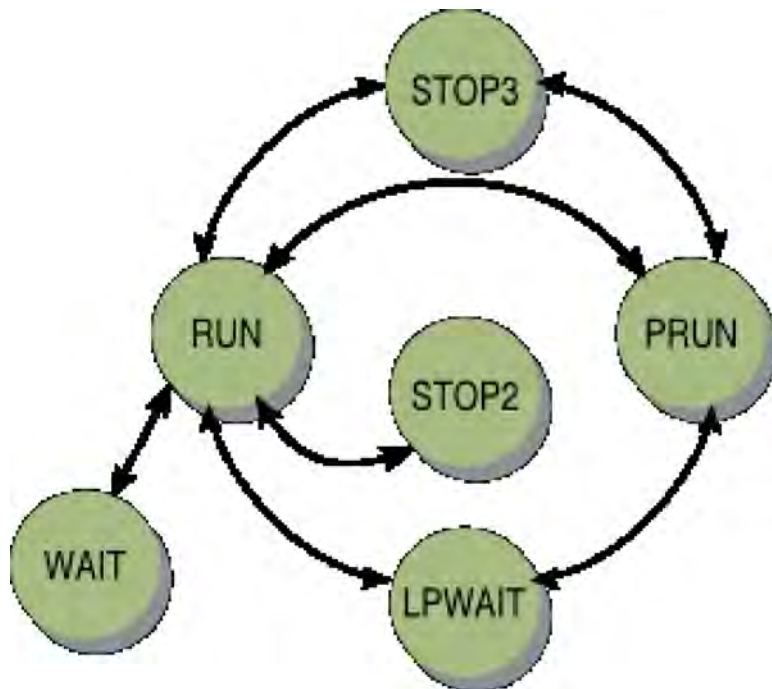
Mode	Regulator State
RUN	Full on
WAIT	Full on
LPRUN	Standby
LPWAIT	Standby
STOP3	Standby
STOP2	Partial power down

Figure 3-2. Allowable Power Mode Transitions for the MC9S08JE128 series

Figure 3-2 illustrates mode state transitions allowed between the legal states shown in Table 3-1. PTB2/$\overline{\text{RESET}}$ must be asserted low or a TOD interrupt must occur to exit STOP2. Interrupts suffice for the other STOP and WAIT modes.

Table 3-3 defines triggers for the various state transitions shown in Figure 3-2.

Table 3-3. Triggers for Transitions Shown in Figure 3-2.

Transition #	From	To	Trigger
1	RUN	LPRUN	Configure settings shown in Table 3-1, **switch LPR=1 last**
	LPRUN	RUN	Clear LPR
			Interrupt when LPWUI=1
2	RUN	STOP2	Pre-configure settings shown in Table 3-1, issue **STOP instruction**
	STOP2	RUN	**assert zero on PTB2/RESET**[1] **or a TOD interrupt**, reload environment from RAM
3	LPRUN	LPWAIT	**WAIT** instruction
	LPWAIT	LPRUN	**Interrupt** when LPWUI=0
4	LPRUN	STOP3	**STOP** instruction
	STOP3	LPRUN	**Interrupt** when LPWUI=0
5	LPWAIT	RUN	**Interrupt** when LPWUI=1
	RUN	LPWAIT	**NOT SUPPORTED**
6	RUN	WAIT	**WAIT instruction**
	WAIT	RUN	**Interrupt or reset**
7	STOP3	RUN	**Interrupt (**if LPR = 0, or LPR = 1 and LPWUI =1) **or reset**
	RUN	STOP3	**STOP instruction**

[1] An analog connection from this pin to the on-chip regulator will wake up the regulator, which will then initiate a power-on-reset sequence.

3.8.1   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 3.7.1, "Stop2 Mode," and Section 3.7.2, "Stop3 Mode," for specific information on system behavior in stop modes.

Table 3-4. Stop Mode Behavior

Peripheral	Mode	
	Stop2	Stop3
CPU	Off	Standby

Table 3-4. Stop Mode Behavior (Continued)

Peripheral	Mode	
	Stop2	**Stop3**
RAM	Standby	Standby
Flash	Off	Standby
Parallel Port Registers	Off	Standby
ADC	Off	Optionally On[1]
PRACMP	Off	Optionally On[2]
MCG	Off	Optionally On[3]
IIC	Off	Standby
DAC	Off	Standby
VREF	Optionally on[4]	Optionally on[4]
SCIx	Off	Standby
SPIx	Off	Standby
PDB	Off	Standby
TOD	Optionally On	Optionally On
CRC	Off	Standby
CMT	Off	Standby
TPMx	Off	Standby
System Voltage Regulator	Off	Standby
XOSC2	Off	Optionally On[5]
XOSC1	Optionally On	Optionally On[6]
I/O Pins	States Held	States Held
USB (SIE and Transceiver)	Off	Optionally On[7]
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] PS[3:0] in SC does not equal 0 before entering stop, else off.

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

[6] ERCLKEN and EREFSTEN set in MCGC2, else in standby. High-frequency range (RANGE in MCGC2 set) requires the LVD to also be enabled in stop3.

[7] USBEN in CTL is set and USBPHYEN in USBCTL0 is set, else off.

Chapter 4
Memory

4.1   MC9S08JE128 series Memory Map

Figure 4-1 shows the memory map for the MC9S08JE128 series. On-chip memory in the MC9S08JE128 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 0x191F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

Extended Address

CPU Address

0x00000

PPAGE=0

FLASH
16384 BYTES

When PPAGE 0 is accessed through the linear address pointer or through the paging window, the flash memory is read.

DIRECT PAGE REGISTERS 96 BYTES

RAM
5968 BYTES

HIGH
PAGE REGISTERS

USB RAM
256 BYTES

RAM
5776 BYTES

USB BOOT ROM
4096 BYTES

0x03FFF

0x0000

0x00AF
0x00B0

0x17FF
0x1800

0x191F
0x1920

0x1A1F
0x1A20

0x30AF
0x30B0

When the CPU accesses PPAGE 0 directly, RAM and registers, when present, take priority over flash memory.

PPAGE=7

PPAGE=6

PPAGE=5

PPAGE=4

FLASH
16384 BYTES

0x10000-0x13FFF

0x14000-0x17FFF

0x18000-0x1BFFF

0x1C000-0x1FFFF

Extended Address

0x04000

PPAGE=1

FLASH
16384 BYTES

0x07FFF
0x08000

Paging Window -
Extended addresses formed with PPAGE and A13:A0 of CPU address

0x0BFFF
0x0C000

PPAGE=3

FLASH
16384 BYTES

0x0FFFF

0x40B0

0x7FFF
0x8000

0xBFFF
0xC000

0xFFFF

PPAGE=3

PPAGE=2

PPAGE=1

PPAGE=0

FLASH
16384 BYTES

0x00000-0x03FFF

0x04000-0x07FFF

0x08000-0x0BFFF

0x0C000-0x0FFFF

Extended Address

Figure 4-1. MC9S08JE128 series Memory Map

Figure 4-2 shows the memory map for the MC9S08JE64 series.
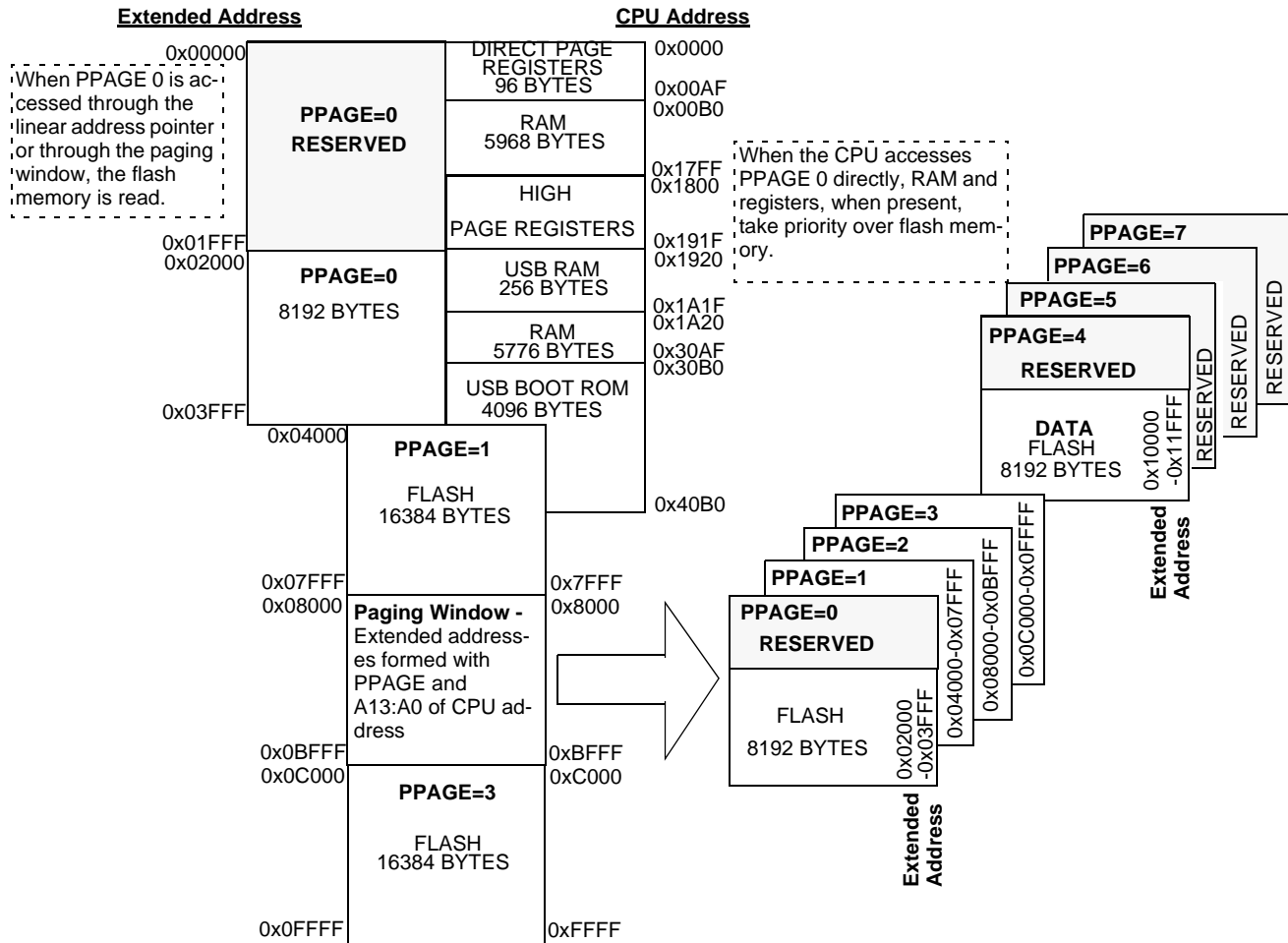
Figure 4-2. MC9S08JE64 Memory Map

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 MC9S08JE128 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
0xFF9E/0xFF9F	SCI2 transmit	Vsci2tx
0xFFC0/0xFFC1	SCI2 receive	Vsci2rx
0xFFC2/0xFFC3	SCI2 error	Vsci2err

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

Address (High/Low)	Vector	Vector Name
0xFFC4/0xFFC5	Time of Day interrupt	Vtod
0xFFC6/0xFFC7	Keyboard2 pins	Vkeyboard2
0xFFC8/0xFFC9	Keyboard1 pins	Vkeyboard1
0xFFCA/0xFFCB	Prog. Analog comparator	Vacmp
0xFFCC/0xFFCD	ADC	Vadc
0xFFCE/0xFFCF	IIC control	Viic
0xFFD0/0xFFD1	SCI1 transmit	Vsci1tx
0xFFD2/0xFFD3	SCI1 receive	Vsci1rx
0xFFD4/0xFFD5	SCI1 error	Vsci1err
0xFFD6/0xFFD7	CMT	Vcmt
0xFFD8/0xFFD9	SPI2	Vspi2
0xFFDA/0xFFDB	TPM2 overflow	Vtpm2ovf
0xFFDC/0xFFDD	TPM2 channel 3	Vtpm2ch3
0xFFDE/0xFFDF	TPM2 channel 2	Vtpm2ch2
0xFFE0/0xFFE1	TPM2 channel 1	Vtpm2ch1
0xFFE2/0xFFE3	TPM2 channel 0	Vtpm2ch0
0xFFE4/0xFFE5	TPM1 overflow	Vtpm1ovf
0xFFE6/0xFFE7	TPM1 channel 3	Vtpm1ch3
0xFFE8/0xFFE9	TPM1 channel 2	Vtpm1ch2
0xFFEA/0xFFEB	TPM1 channel 1	Vtpm1ch1
0xFFEC/0xFFED	TPM1 channel 0	Vtpm1ch0
0xFFEE/0xFFEF	DAC WaterMark	Vdac
0xFFF0/0xFFF1	PDB	Vpdb
0xFFF2/0xFFF3	USB Status	Vusb
0xFFF4/0xFFF5	FSPI1	Vspi1
0xFFF6/0xFFF7	MCG Loss of Lock	Vlol
0xFFF8/0xFFF9	Low-voltage detect, Low-voltage warning	Vlvd
0xFFFA/0xFFFB	IRQ pin	Virq
0xFFFC/0xFFFD	Software interrupt	Vswi
0xFFFE/0xFFFF	Watchdog timer Low-voltage detect External pin Illegal opcode Illegal address	Vreset

4.2 Register Addresses and Bit Assignments

The registers in the MC9S08JE128 series are divided into these three groups:

- Direct-page registers are located in the first 128 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:

  — A 2-byte CRC checksum location

  — 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	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	PTADD	PTADD7	PTADD6	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	PTCD7	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
0x0005	PTCDD	PTCDD7	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	PPAGE	0	0	0	0	0	XA16	XA15	XA14
0x0009	LAP2	0	0	0	0	0	0	0	LA16
0x000A	LAP1	LA15	LA14	LA13	LA12	LA11	LA10	LA9	LA8
0x000B	LAP0	LA7	LA6	LA5	LA4	LA3	LA2	LA1	LA0
0x000C	LWP	D7	D6	D5	D4	D3	D2	D1	D0
0x000D	LBP	D7	D6	D5	D4	D3	D2	D1	D0
0x000E	LB	D7	D6	D5	D4	D3	D2	D1	D0
0x000F	LAPAB	D7	D6	D5	D4	D3	D2	D1	D0
0x0010	DACDAT0H	DACDAT0[15:8]							
0x0011	DACDAT0L	DACDAT0[7:0]							
0x0012	DACDAT1H	DACDAT1[15:8]							
0x0013	DACDAT1L	DACDAT1[7:0]							
0x0014	DACDAT2H	DACDAT2[15:8]							
0x0015	DACDAT2L	DACDAT2[7:0]							
0x0016	DACDAT3H	DACDAT3[15:8]							
0x0017	DACDAT3L	DACDAT3[7:0]							
0x0018	DACDAT4H	DACDAT4[15:8]							
0x0019	DACDAT4L	DACDAT4[7:0]							
0x001A	DACDAT5H	DACDAT5[15:8]							
0x001B	DACDAT5L	DACDAT5[7:0]							
0x001C	DACDAT6H	DACDAT6[15:8]							
0x001D	DACDAT6L	DACDAT6[7:0]							
0x001E	DACDAT7H	DACDAT7[15:8]							
0x001F	DACDAT7L	DACDAT7[7:0]							
0x0020	DACDAT8H	DACDAT8[15:8]							
0x0021	DACDAT8L	DACDAT8[7:0]							
0x0022	DACDAT9H	DACDAT9[15:8]							
0x0023	DACDAT9L	DACDAT9[7:0]							
0x0024	DACDAT10H	DACDAT10[15:8]							
0x0025	DACDAT10L	DACDAT10[7:0]							
0x0026	DACDAT11H	DACDAT11[15:8]							
0x0027	DACDAT11L	DACDAT11[7:0]							
0x0028	DACDAT12H	DACDAT12[15:8]							
0x0029	DACDAT12L	DACDAT12[7:0]							

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	DACDAT13H				DACDAT13[15:8]				
0x002B	DACDAT13L				DACDAT13[7:0]				
0x002C	DACDAT14H				DACDAT14[15:8]				
0x002D	DACDAT14L				DACDAT14[7:0]				
0x002E	DACDAT15H				DACDAT15[15:8]				
0x002F	DACDAT15L				DACDAT15[7:0]				
0x0030	DACS	0	0	0	0	0	DACWM	DACRPT	DACRPB
0x0031	DACC0	DACEN	DACRFS	DACTSEL	DACSTRG	LPEN	DACWIE	DACTIE	DACBIE
0x0032	DACC1	0	0	0	DACBFWM		DACBFMD		DACBFE
0x0033	DACC2		DACBFRP				DACBFUP		
0x0034	PRACMPCS	ACEN	ACMPF	—	ACOPE	ACMPO	ACINTS		ACIEN
0x0035	PRACMPC0	—		ACPSEL		—		ACNSEL	
0x0036	PRACMPC1	PRGEN	PRGINS	—	PRGOS4	PRGOS3	PRGOS2	PRGOS1	PRGOS0
0x0037	PRACMPC2	—	ACIPE6	ACIPE5	ACIPE4	ACIPE3	ACIPE2	ACIPE1	ACIPE0
0x0038	MCGC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
0x0039	MCGC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
0x003A	MCGTRM				TRIM				
0x003B	MCGSC	LOLS	LOCK	PLLST	IREFST	CLKST		OSCINIT	FTRIM
0x003C	MCGC3	LOLIE	PLLS	CME	DIV32	VDIV			
0x003D	MCGC4	0	0	DMX32	0	0	0	DRST / DRS	
0x003E-0x003F	RESERVED	—	—	—	—	—	—	—	—
0x0040	ADCSC1	COCO	AIEN	DIFF	ADCH				
0x0041-0x0047	RESERVED	—	—	—	—	—	—	—	—
0x0048	ADCCFG1	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x0049	ADCCFG2	0	0	0	0	ADACKEN	ADHSC	ADLSTS	
0x004A	ADCRH				D[15:8]				
0x004B	ADCRL				D[7:0]				
0x004C-0x005B	RESERVED	—	—	—	—	—	—	—	—
0x005C	VREFTRM				TRM				
0x005D	VREFSC	VREFEN	0	0	0	0	VREFST	MODE	
0x005E	RESERVED	—	—	—	—	—	—	—	—
0x005F	IRQSC	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x0060	IICA1	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
0x0061	IICF	MULT		ICR					
0x0062	IICC1	IICEN	IICIE	MST	TX	TXAK	RSTA	0	0
0x0063	IICS	TCF	IAAS	BUSY	ARBL	0	SRW	IICIF	RXAK
0x0064	IICD				DATA				
0x0065	IICC2	GCAEN	ADEXT	0	0	0	AD10	AD9	AD8
0x0066	IICSMB	FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF	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
0x0067	IICA2	SAD7	SAD6	SAD5	SAD4	SAD3	SAD2	SAD1	0
0x0068	IICSLTH	SSLT15	SSLT14	SSLT13	SSLT12	SSLT11	SSLT10	SSLT9	SSLT8
0x0069	IICSLTL	SSLT7	SSLT6	SSLT5	SSLT4	SSLT3	SSLT2	SSLT1	SSLT0
0x006A	IICFLT	0	0	0	0	FLT3	FLT2	FLT1	FLT0
0x006B	RESERVED	—	—	—	—	—	—	—	—
0x006C	KBI1SC	0	0	0	0	KB1F	KB1ACK	KB1IE	KBI1MOD
0x006D	KBI1PE	KBI1PE7	KBI1PE6	KBI1PE5	KBI1PE4	KBI1PE3	KBI1PE2	KBI1PE1	KBI1PE0
0x006E	KBI1ES	KB1EDG7	KB1EDG6	KB1EDG5	KB1EDG4	KB1EDG3	KB1EDG2	KB1EDG1	KB1EDG0
0x006F	RESERVED	—	—	—	—	—	—	—	—
0x0070	SPI1C1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0071	SPI1C2	SPMIE	SPIMODE	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x0072	SPI1BR	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
0x0073	SPI1S	SPRF	SPMF	SPTEF	MODF	RNFULLF	TNEAREF	TXFULLF	RFIFOFEF
0x0074	SPI1DH	Bit 15	14	13	12	11	10	9	Bit 8
0x0075	SPI1DL	Bit 7	6	5	4	3	2	1	Bit 0
0x0076	SPI1MH	Bit 15	14	13	12	11	10	9	Bit 8
0x0077	SPI1ML	Bit 7	6	5	4	3	2	1	Bit 0
0x0078	SPI1C3	0	0	TNEAREFMARK	RNFULLMARK	INTCLR	TNEARIEN	RNFULLIEN	FIFOMODEN
0x0079	SPI1CI	TXFERR	RXFERR	TXFOF	RXFOF	TNEAREFCI	RNFULLFCI	SPTEFCI	SPRFCI
0x007A-0x007B	RESERVED	—	—	—	—	—	—	—	—
0x007C	KBI2SC	0	0	0	0	KB2F	KB2ACK	KB2IE	KBI2MOD
0x007D	KBI2PE	KBI2PE7	KBI2PE6	KBI2PE5	KBI2PE4	KBI2PE3	KBI2PE2	KBI2PE1	KBI2PE0
0x007E	KBI2ES	KB2EDG7	KB2EDG6	KB2EDG5	KB2EDG4	KB2EDG3	KB2EDG2	KB2EDG1	KB2EDG0
0x007F	RESERVED	—	—	—	—	—	—	—	—
0x0080	USBCTL0	USBRESET	USBPU	USBRESMEN	LPRESF	—	USBVREN	—	USBPHYEN
0x0081–0x0087	RESERVED	—	—	—	—	—	—	—	—
0x0088	PERID	0	0	ID5	ID4	ID3	ID2	ID1	ID0
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

MC9S08JE128 Series Reference Manual, Rev. 3 Freescale Semiconductor

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
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 6)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	LOC	LVD	0
0x1801	SBDFR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT1	COPT		STOPE	0	BLMSS	0	BKGDPE	RSTPE
0x1803	SOPT2	COPCLKS	COPW	0	CLKOUT_EN	0	0	0	ACIC
0x1804-0x1805	RESERVED	—	—	—	—	—	—	—	—
0x1806	SDIDH	REV				ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	SCGC1	CMT	TPM2	TPM1	ADC	DAC	IIC	SCI2	SCI1
0x1809	SCGC2	USB	PDB	IRQ	KBI	ACMP	TOD	SPI2	SPI1
0x180A	SCGC3	VREF	CRC	—	FLS1	—	—	—	—
0x180B	SOPT3	SCI2PS	SCI1PS	IICPS	—	—	—	SCI1_PAD	CMT_PAD
0x180C	SOPT4	—	—	—	—	—	—	IROSRE	IRODSE
0x180D	SOPT5[1]	—	—	—	—				
0x180E	SIMIPS	ADCTRS	RX1N	—	—	MTBASE1		—	MODTX1
0x180F	SIGNATURE	SIGNATURE SEMAPHORE							
0x1810	CCSCTRL	RANGE1	HGO1	ERCLKEN1	OSCINIT1	EREFS1	EN	TEST	SEL
0x1811	CCSTMR1	CNT1							
0x1812	CCSTMR2	CNT2							
0x1813	CCSTMRIR	CNTIR							
0x1814	FPROTD	—	—	—	—	—	—	—	FPDIS
0x1815-0x1818	RESERVED	—	—	—	—	—	—	—	—

Table 4-3. High-Page Register Summary (Sheet 2 of 6)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1819	SIMCO	—	—	—	—	—	CS		
0x181A–0x181B	RESERVED								
0x181C	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0	BGBE
0x181D	SPMSC2	LPR	LPRS	LPWUI	0	PPDF	PPDACK	PPDE	PPDC
0x181E	RESERVED	—	—	—	—	—	—	—	—
0x181F	SPMSC3	LVWF	LVWACK	LVDV	LVWV	LVWIE	0	0	0
0x1820	FCDIV	FDIVLD	PRDIV8	FDIV					
0x1821	FOPT	KEYEN		0	0	0	0	SEC	
0x1822	RESERVED	—	—	—	—	—	—	—	—
0x1823	FCNFG	0	0	KEYACC	0	0	0	0	0
0x1824	FPROT	FPS							FPOPEN
0x1825	FSTAT	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x1826	FCMD	FCMD							
0x1827–0x182F	RESERVED	—	—	—	—	—	—	—	—
0x1830	PTED	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
0x1831	PTEDD	PTEDD7	PTEDD6	PTEDD5	0	PTEDD3	PTEDD2	PTEDD1	PTEDD0
0x1832	PTFD	PTFD7	PTFD6	PTFD5	PTFD4	PTFD3	PTFD2	PTFD1	PTFD0
0x1833	PTFDD	PTFDD7	PTFDD6	PTFDD5	PTFDD4	PTFDD3	PTFDD2	PTFDD1	PTFDD0
0x1834–0x1837	RESERVED	—	—	—	—	—	—	—	—
0x1838	SCI2BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x1839	SCI2BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x183A	SCI2C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x183B	SCI2C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x183C	SCI2S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x183D	SCI2S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x183E	SCI2C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x183F	SCI2D	Bit 7	6	5	4	3	2	1	Bit 0
0x1840	SPI2C1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x1841	SPI2C2	SPMIE	0	0	MODFEN	BIDIROE	0	SPISWAI	SP2C0
0x1842	SPI2BR	0	SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
0x1843	SPI2S	SPRF	SPMF	SPTEF	MODF	0	0	0	0
0x1844	RESERVED	—	—	—	—	—	—	—	—
0x1845	SPI2D	Bit 7	6	5	4	3	2	1	Bit 0
0x1846	RESERVED	—	—	—	—	—	—	—	—
0x1847	SPI2MR	Bit 7	6	5	4	3	2	1	Bit 0
0x1848	PTGD	—	—	—	—	—	—	PTGD1	PTGD0
0x1849	PTGDD	—	—	—	—	—	—	PTGDD1	PTGDD0
0x184A–0X184F	RESERVED	—	—	—	—	—	—	—	—
0x1850	PTAPE	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0

Table 4-3. High-Page Register Summary (Sheet 3 of 6)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1851	PTASE	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x1852	PTADS	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x1853	PTAIFE	PTAIFE7	PTAIFE6	PTAIFE5	PTAIFE4	PTAIFE3	PTAIFE2	PTAIFE1	PTAIFE0
0x1854	PTBPE	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x1855	PTBSE	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x1856	PTBDS	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x1857	PTBIFE	PTBIFE7	PTBIFE6	PTBIFE5	PTBIFE4	PTBIFE3	PTBIFE2	PTBIFE1	PTBIFE0
0x1858	PTCPE	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x1859	PTCSE	PTCSE7	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x185A	PTCDS	PTCDS7	PTCDS6	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x185B	PTCIFE	PTCIFE7	PTCIFE6	PTCIFE5	PTCIFE4	PTCIFE3	PTCIFE2	PTCIFE1	PTCIFE0
0x185C	PTDPE	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0
0x185D	PTDSE	PTDSE7	PTDSE6	PTDSE5	PTDSE4	PTDSE3	PTDSE2	PTDSE1	PTDSE0
0x185E	PTDDS	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
0x185F	PTDIFE	PTDIFE7	PTDIFE6	PTDIFE5	PTDIFE4	PTDIFE3	PTDIFE2	PTDIFE1	PTDIFE0
0x1860	PTEPE	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
0x1861	PTESE	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0
0x1862	PTEDS	PTEDS7	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
0x1863	PTEIFE	PTEIFE7	PTEIFE6	PTEIFE5	PTEIFE4	PTEIFE3	PTEIFE2	PTEIFE1	PTEIFE0
0x1864	PTFPE	PTFPE7	PTFPE6	PTFPE5	PTFPE4	PTFPE3	PTFPE2	PTFPE1	PTFPE0
0x1865	PTFSE	PTFSE7	PTFSE6	PTFSE5	PTFSE4	PTFSE3	PTFSE2	PTFSE1	PTFSE0
0x1866	PTFDS	PTFDS7	PTFDS6	PTFDS5	PTFDS4	PTFDS3	PTFDS2	PTFDS1	PTFDS0
0x1867	PTFIFE	PTFIFE7	PTFIFE6	PTFIFE5	PTFIFE4	PTFIFE3	PTFIFE2	PTFIFE1	PTFIFE0
0x1868	PTGPE	—	—	—	—	—	—	PTGPE1	PTGPE0
0x1869	PTGSE	—	—	—	—	—	—	PTGSE1	PTGSE0
0x186A	PTGDS	—	—	—	—	—	—	PTGDS1	PTGDS0
0x186B	PTGIFE	PTGIFE7	PTGIFE6	PTGIFE5	PTGIFE4	PTGIFE3	PTGIFE2	PTGIFE1	PTGIFE0
0x186C-0x186F	RESERVED	—	—	—	—	—	—	—	—
0x1870	CMTCGH1	PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0
0x1871	CMTCGL1	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0
0x1872	CMTCGH2	SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
0x1873	CMTCGL2	SL7	SL6	SL5	SL4	SL3	SL2	SL1	SL0
0x1874	CMTOC	IROL	CMTPOL	IROPEN	0	0	0	0	0
0x1875	CMTMSC	EOCF	CMTDIV1	CMTDIV0	EXSPC	BASE	FSK	EOCIE	MCGEN
0x1876	CMTCMD1	MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8
0x1877	CMTCMD2	MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0
0x1878	CMTCMD3	SB15	SB14	SB13	SB12	SB11	SB10	SB9	SB8
0x1879	CMTCMD4	SB7	SB6	SB5	SB4	SB3	SB2	SB1	SB0
0x187A-0x187F	RESERVED	—	—	—	—	—	—	—	—
0x1880	DBGCAH	Bit 15	14	13	12	11	10	9	Bit 8
0x1881	DBGCAL	Bit 7	6	5	4	3	2	1	Bit 0

Table 4-3. High-Page Register Summary (Sheet 4 of 6)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1882	DBGCBH	Bit 15	14	13	12	11	10	9	Bit 8
0x1883	DBGCBL	Bit 7	6	5	4	3	2	1	Bit 0
0x1884	DBGCCH	Bit 15	14	13	12	11	10	9	Bit 8
0x1885	DBGCCL	Bit 7	6	5	4	3	2	1	Bit 0
0x1886	DBGFH	Bit 15	14	13	12	11	10	9	Bit 8
0x1887	DBGFL	Bit 7	6	5	4	3	2	1	Bit 0
0x1888	DBGCAX	RWAEN	RWA	PAGSEL	0	0	0	0	Bit 16
0x1889	DBGCBX	RWBEN	RWB	PAGSEL	0	0	0	0	Bit 16
0x188A	DBGCCX	RWCEN	RWC	PAGSEL	0	0	0	0	Bit 16
0x188B	DBGFX	PPACC	—	—	—	—	—	—	Bit 16
0x188C	DBGC	DBGEN	ARM	TAG	BRKEN	0	0	0	LOOP1
0x188D	DBGT	TRGSEL	BEGIN	0	0	TRG			
0x188E	DBGS	AF	BF	CF	0	0	0	0	ARMF
0x188F	DBGCNT	0	0	0	0	CNT			
0x1890	CRCH	Bit 15	14	13	12	11	10	9	Bit 8
0x1891	CRCL	Bit 7	6	5	4	3	2	1	Bit 0
0x1892	TRANSPOSE	Bit 7	6	5	4	3	2	1	Bit 0
0X1893-0x189B	RESERVED	—	—	—	—	—	—	—	—
0x189C	TODC	TODEN	TODCLKS		TODR	TODCLKEN	TODPS		
0x189D	TODSC	QSECF	SECF	MTCHF	QSECIE	SECIE	MTCHIE	MTCHEN	MTCHWC
0x189E	TODM	TODM						MQSEC	
0x189F	TODCNT	TODCNT							
0x18A0	TPM2SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x18A1	TPM2CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x18A2	TPM2CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x18A3	TPM2MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x18A4	TPM2MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x18A5	TPM2C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x18A6	TPM2C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x18A7	TPM2C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x18A8	TPM2C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x18A9	TPM2C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x18AA	TPM2C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x18AB	TPM2C2SC	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x18AC	TPM2C2VH	Bit 15	14	13	12	11	10	9	Bit 8
0x18AD	TPM2C2VL	Bit 7	6	5	4	3	2	1	Bit 0
0x18AE	TPM2C3SC	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	0	0
0x18AF	TPM2C3VH	Bit 15	14	13	12	11	10	9	Bit 8
0x18B0	TPM2C3VL	Bit 7	6	5	4	3	2	1	Bit 0
0x18B1-0X18B7	RESERVED	—	—	—	—	—	—	—	—
0x18B8	SCI1BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8

MC9S08JE128 Series Reference Manual, Rev. 3

Freescale Semiconductor

Table 4-3. High-Page Register Summary (Sheet 5 of 6)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x18B9	SCI1BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x18BA	SCI1C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x18BB	SCI1C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x18BC	SCI1S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x18BD	SCI1S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x18BE	SCI1C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x18BF	SCI1D	Bit 7	6	5	4	3	2	1	Bit 0
0x18C0	PDBSC	PDBEN	PDBIF	PDBIE	LDMOD	TOS		DACTOE	LDOK
0x18C1	PDBC1	PRESCALER			TRIGSEL			CONT	MULT
0x18C2	PDBC2	—	—	—	—	—	—	—	SWTRIG
0x18C3	PDBCHEN	CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1	CHEN0
0x18C4	PDBMODH	MOD[15:8]							
0x18C5	PDBMODL	MOD[7:0]							
0x18C6	PDBCNTH	COUNT[15:8]							
0x18C7	PDBCNTL	COUNT[7:0]							
0x18C8	PDBIDLYH	IDELAY[15:8]							
0x18C9	PDBIDLYL	IDELAY[7:0]							
0x18CA	DACINTH	DACINT[15:8]							
0x18CB	DACINTL	DACINT[7:0]							
0x18CC	PDBDLYH	DELAY[15:8]							
0x18CD	PDBDLYL	DELAY[7:0]							
0x18CE-0X18DF	RESERVED	—	—	—	—	—	—	—	—
0x18E0	TPM1SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x18E1	TPM1CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x18E2	TPM1CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x18E3	TPM1MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x18E4	TPM1MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x18E5	TPM1C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x18E6	TPM1C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x18E7	TPM1C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x18E8	TPM1C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x18E9	TPM1C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x18EA	TPM1C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x18EB	TPM1C2SC	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x18EC	TPM1C2VH	Bit 15	14	13	12	11	10	9	Bit 8
0x18ED	TPM1C2VL	Bit 7	6	5	4	3	2	1	Bit 0
0x18EE	TPM1C3SC	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	0	0
0x18EF	TPM1C3VH	Bit 15	14	13	12	11	10	9	Bit 8
0x18F0	TPM1C3VL	Bit 7	6	5	4	3	2	1	Bit 0
0x18F1-0x18F7	RESERVED	—	—	—	—	—	—	—	—
0x18F8	ADCCVH	CV[15:8]							

Table 4-3. High-Page Register Summary (Sheet 6 of 6)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x18F9	ADCCVL	CV[7:0]							
0x18FA-0x18FB	RESERVED	—	—	—	—	—	—	—	—
0x18FC	ADCSC2	ADACT	ADTRG	ACFE	ACFGT	—	0	—	—
0x18FD	ADCSC3	CAL	CALF	0	0	ADCO	—	—	—
0x18FE	ADCOFSH	OFS[15:8]							
0x18FF	ADCOFSL	OFS[7:0]							
0x1900	ADCPGH	PG[15:8]							
0x1901	ADCPGL	PG[7:0]							
0x1902	ADCMGH	MG[15:8]							
0x1903	ADCMGL	MG[7:0]							
0x1904	ADCCLPD	0	0	CLPD					
0x1905	ADCCLPS	0	0	CLPS					
0x1906	ADCCLP4H	0	0	0	0	0	0	CLP4[9:8]	
0x1907	ADCCLP4L	CLP4[7:0]							
0x1908	ADCCLP3H	0	0	0	0	0	0	0	CLP3[8]
0x1909	ADCCLP3L	CLP3[7:0]							
0x190A	ADCCLP2	CLP2							
0x190B	ADCCLP1	0	CLP1						
0x190C	ADCCLP0	0	0	CLP0					
0x190D	ADCTST	TEST	0	TESTSEL[5:0]					
0x190D	RESERVED	—	—	—	—	—	—	—	—
0x190E	ADCCLMD	0	0	CLMD					
0x190F	ADCCLMS	0	0	CLMS					
0x1910	ADCCLM4H	0	0	0	0	0	0	CLM4[9:8]	
0x1911	ADCCLM4L	CLM4[7:0]							
0x1912	ADCCLM3H	0	0	0	0	0	0	0	CLM3[8]
0x1913	ADCCLM3L	CLM3[7:0]							
0x1914	ADCCLM2	CLM2[7:0]							
0x1915	ADCCLM1	0	CLM1[6:0]						
0x1916	ADCCLM0	0	0	CLM0[5:0]					
0x1917	APCTL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x1918	APCTL2	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
0x1919	APCTL3	ADPC23	ADPC22	ADPC21	ADPC20	ADPC19	ADPC18	ADPC17	ADPC16
0x191A	APCTL4	ADPC31	ADPC30	ADPC29	ADPC28	ADPC27	ADPC26	ADPC25	ADPC24
0x191A-0x191F	RESERVED	—	—	—	—	—	—	—	—

[1] User must not write any value to this location. This location is reserved for Freescale internal testing. Writing a value to this location can affect on-chip LVD performance.

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	Reserved for storage of MCGTRM	TRIM							
0xFFB0–0xFFB7	NVBACKKEY	8-Byte Comparison Key							
0xFFB8	FCHKSH	CHECKSUM[15:8]							
0xFFB9	FCHKSL	CHECKSUM[7:0]							
0xFFBA	CHKSBYP	BYPASS							
0xFFB8–0xFFBB	Reserved	—	—	—	—	—	—	—	—
0xFFBC		Partial_Erase_Semaphore							
0xFFBD	NVPROT	FPS7	FPS6	FPS5	FPS4	FPS3	FPS2	FPS1	FPOPEN
0xFFBE	Reserved	—	—	—	—	—	—	—	—
0xFFBF	NVOPT	KEYEN		0	0	0	0	SEC	

Provided that 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 Memory Management Unit

The memory management unit (MMU) allows the program and data space for the HCS08 Family of Microcontrollers to be extended beyond the 64K byte CPU addressable memory map. The MMU utilizes a paging scheme similar to that seen on other MCU architectures, such as HCS12. The extended memory when used for data can also be accessed linearly using a linear address pointer and data access registers.

4.3.1 Features

Key features of the MMU module are:
- Memory Management Unit extends the HCS08 memory space
  — up to 4M bytes for program and data space
- Extended program space using paging scheme
  — PPAGE register used for page selection

— fixed 16K byte memory window

— architecture supports up to 256, 16K pages

- Extended data space using linear address pointer

  — up to 22-bit linear address pointer

  — linear address pointer and data register provided in direct page allows access of complete Flash memory map using direct page instructions

  — optional auto increment of pointer when data accessed

  — supports an 2s compliment addition/subtraction to address pointer without using any math instructions or memory resources

  — supports word accesses to any address specified by the linear address pointer when using LDHX, STHX instructions

4.3.2 Register Definition

Figure 4-5 is a summary of MMU registers.

Table 4-5. MMU Register Summary

Name		7	6	5	4	3	2	1	0
PPAGE	R	0	0	0	0	0	XA16	XA15	XA14
	W								
LAP2	R	0	0	0	0	0	0	0	LA16
	W								
LAP1	R	LA15	LA14	LA13	LA12	LA11	LA10	LA9	LA8
	W								
LAP0	R	LA7	LA6	LA5	LA4	LA3	LA2	LA1	LA0
	W								
LWP	R	D7	D6	D5	D4	D3	D2	D1	D0
	W								
LBP	R	D7	D6	D5	D4	D3	D2	D1	D0
	W								
LB	R	D7	D6	D5	D4	D3	D2	D1	D0
	W								
LAPAB	R	0	0	0	0	0	0	0	0
	W	D7	D6	D5	D4	D3	D2	D1	D0

4.3.2.1 Program Page Register (PPAGE)

The HCS08 Core architecture limits the CPU addressable space available to 64K bytes. The address space can be extended to 128K bytes using a paging window scheme. The Program Page (PPAGE) allows for

selecting one of the 16K byte blocks to be accessed through the Program Page Window located at 0x8000-0xBFFF. The CALL and RTC instructions can load or store the value of PPAGE onto or from the stack during program execution. After any reset, PPAGE is set to PAGE 2.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	XA16	XA15	XA14
W						XA16	XA15	XA14
Reset:	0	0	0	0	0	0	1	0
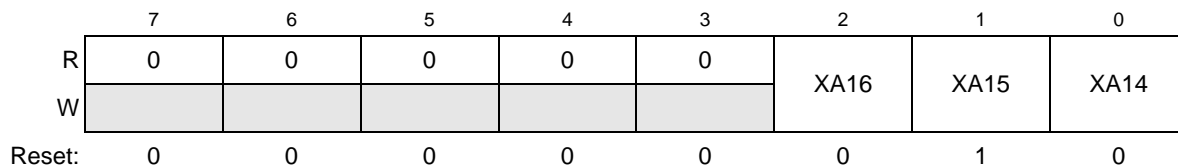
Figure 4-3. Program Page Register (PPAGE)

Table 4-6. Program Page Register Field Descriptions

Field	Description
2:0 XA16:XA14	When the CPU addresses the paging window, 0x8000-0xBFFF, the value in the PPAGE register along with the CPU addresses A13:A0 are used to create a 17-bit extended address.

4.3.2.2 Linear Address Pointer Registers 2:0 (LAP2:LAP0)

The three registers, LAP2:LAP0 contain the 17-bit linear address that allows the user to access any Flash location in the extended address map. This register is used in conjunction with the data registers, linear byte (LB), linear byte post increment (LBP) and linear word post increment (LWP). The contents of LAP2:LAP0 will auto-increment when accessing data using the LBP and LWP registers. The contents of LAP2:LAP0 can be increased by writing an 8-bit value to LAPAB.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	LA16
W								LA16
R	LA15	LA14	LA13	LA12	LA11	LA10	LA9	LA8
W	LA15	LA14	LA13	LA12	LA11	LA10	LA9	LA8
R	LA15	LA6	LA5	LA4	LA3	LA2	LA1	LA0
W	LA15	LA6	LA5	LA4	LA3	LA2	LA1	LA0
Reset:	0	0	0	0	0	0	0	0
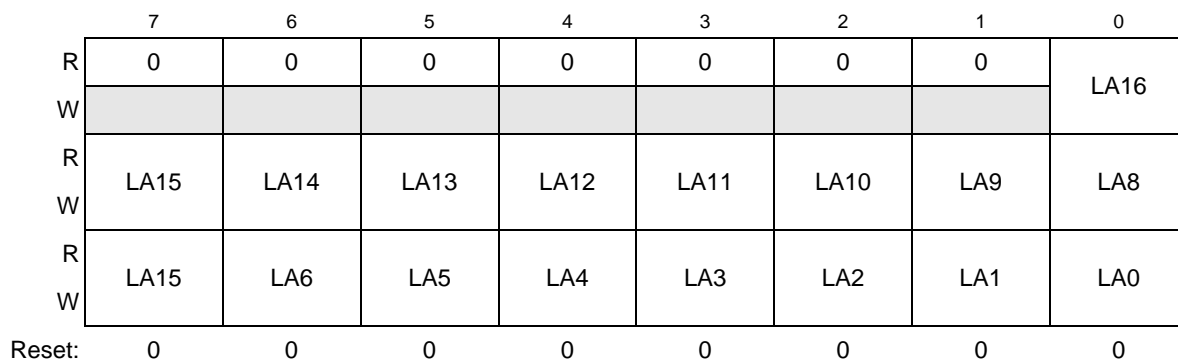
Figure 4-4. Linear Address Pointer Registers 2:0 (LAP2:LAP0)

Table 4-7. Linear Address Pointer Registers 2:0 Field Descriptions

Field	Description
16:0 LA21:LA0	The values in LAP2:LAP0 are used to create a 17-bit linear address pointer. The value in these registers are used as the extended address when accessing any of the data registers LB, LBP and LWP.

4.3.2.3 Linear Word Post Increment Register (LWP)

This register is one of three data registers that the user can use to access any Flash memory location in the extended address map. When LWP is accessed the contents of LAP2:LAP0 make up the extended address

of the Flash memory location to be addressed. When accessing data using LWP, the contents of LAP2:LAP0 will increment after the read or write is complete.

Accessing LWP does the same thing as accessing LBP. The MMU register ordering of LWP followed by LBP, allow the user to access data by words using the LDHX or STHX instructions of the LWP register.

	7	6	5	4	3	2	1	0
R	D7	D6	D5	D4	D3	D2	D1	D0
W								
Reset:	0	0	0	0	0	0	0	0
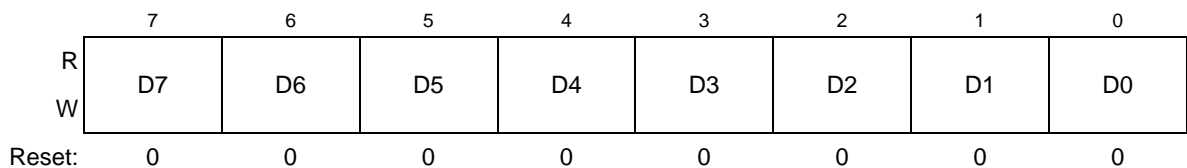
Figure 4-5. Linear Word Post Increment Register (LWP)

Table 4-8. Linear Word Post Increment Register Field Descriptions

Field	Description
7:0 D7:D0	Reads of this register will first return the data value pointed to by the linear address pointer, LAP2:LAP0 and then will increment LAP2:LAP0. Writes to this register will first write the data value to the memory location specified by the linear address pointer and then will increment LAP2:LAP0. Writes to this register are most commonly used when writing to the Flash block(s) during programming.

4.3.2.4 Linear Byte Post Increment Register (LBP)

This register is one of three data registers that the user can use to access any Flash memory location in the extended address map. When LBP is accessed the contents of LAP2:LAP0 make up the extended address of the Flash memory location to be addressed. When accessing data using LBP, the contents of LAP2:LAP0 will increment after the read or write is complete.

Accessing LBP does the same thing as accessing LWP. The MMU register ordering of LWP followed by LBP, allow the user to access data by words using the LDHX or STHX instructions with the address of the LWP register.

	7	6	5	4	3	2	1	0
R	D7	D6	D5	D4	D3	D2	D1	D0
W								
Reset:	0	0	0	0	0	0	0	0
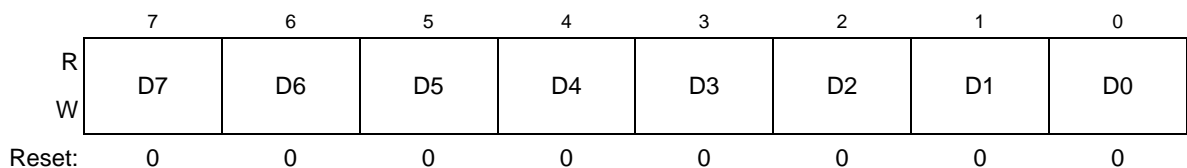
Figure 4-6. Linear Byte Post Increment Register (LBP)

Table 4-9. Linear Byte Post Increment Register Field Descriptions

Field	Description
7:0 D7:D0	Reads of this register will first return the data value pointed to by the linear address pointer, LAP2:LAP0 and then will increment LAP2:LAP0. Writes to this register will first write the data value to the memory location specified by the linear address pointer and then will increment LAP2:LAP0. Writes to this register are most commonly used when writing to the Flash block(s) during programming.

4.3.2.5 Linear Byte Register (LB)

This register is one of three data registers that the user can use to access any Flash memory location in the extended address map. When LB is accessed the contents of LAP2:LAP0 make up the extended address of the Flash memory location to be addressed.

	7	6	5	4	3	2	1	0
R	D7	D6	D5	D4	D3	D2	D1	D0
W								
Reset:	0	0	0	0	0	0	0	0
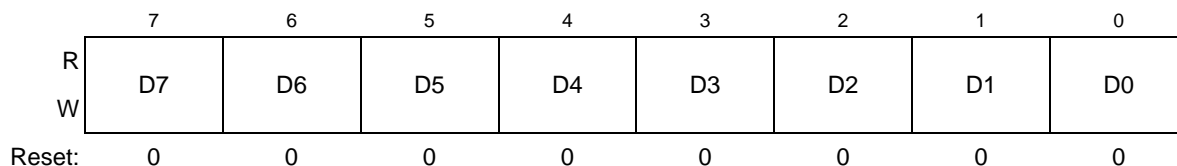
Figure 4-7. Linear Byte Register (LB)

Table 4-10. Linear Data Register Field Descriptions

Field	Description
7:0 D7:D0	Reads of this register returns the data value pointed to by the linear address pointer, LAP2:LAP0. Writes to this register will write the data value to the memory location specified by the linear address pointer. Writes to this register are most commonly used when writing to the Flash block(s) during programming.

4.3.2.6 Linear Address Pointer Add Byte Register (LAPAB)

The user can increase or decrease the contents of LAP2:LAP0 by writing a 2s compliment value to LAPAB. The value written will be added to the current contents of LAP2:LAP0.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	D7	D6	D5	D4	D3	D2	D1	D0
Reset:	0	0	0	0	0	0	0	0
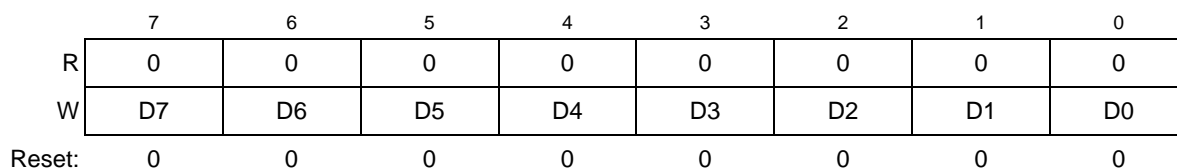
Figure 4-8. Linear Address Pointer Add Byte Register (LAPAB)

Table 4-11. Linear Address Pointer Add Byte Register Field Descriptions

Field	Description
7:0 D7:D0	The 2s compliment value written to LAPAB will be added to contents of the linear address pointer register, LAP2:LAP0. Writing a value of 0x7f to LAPAB will increase LAP by 127, a value of 0xff will decrease LAP by 1, and a value of 0x80 will decrease LAP by 128.

4.3.3 Functional Description

4.3.3.1 Memory Expansion

The HCS08 Core architecture limits the CPU addressable space available to 64K bytes. The Program Page (PPAGE) allows for integrating up to 4M byte of Flash into the system by selecting one of the 16K byte blocks to be accessed through the Paging Window located at 0x8000-0xBFFF. The MMU module also provides a linear address pointer that allows extension of data access up to 4M bytes.

4.3.3.1.1 Program Space

The PPAGE register holds the page select value for the Paging Window. The value in PPAGE can be manipulated by using normal read and write instructions as well as the CALL and RTC instructions. The user should not change PPAGE directly when running from paged memory, only CALL and RTC should be used.

When the MMU detects that the CPU is addressing the Paging Window, the value currently in PPAGE will be used to create an extended address that the MCU's decode logic will use to select the desired Flash location.

As seen in Figure 4-5, the Flash blocks in the CPU addressable memory can be accessed directly or using the Paging Window and PPAGE register. For example, the Flash from location 0x4000-0x7FFF can be accessed directly or using the paging window, PPAGE = 1, address 0x8000-0xBFFF.

4.3.3.1.2 CALL and RTC (Return from Call) Instructions

CALL and RTC are instructions that perform automated page switching when executed in the user program. CALL is similar to a JSR instruction, but the subroutine that is called can be located anywhere in the normal 64K byte address space or on any page of program memory.

During the execution of a CALL instruction, the CPU:

- Stacks the return address.
- Pushes the current PPAGE value onto the stack.
- Writes the new instruction-supplied PPAGE value into the PPAGE register.
- Transfers control to the subroutine of the new instruction-supplied address.

This sequence is not interruptible; there is no need to inhibit interrupts during CALL execution. A CALL can be executed from any address in memory to any other address.

The new PPAGE value is provided by an immediate operand in the instruction along with the address within the paging window, 0x8000-0xBFFF.

RTC is similar to an RTS instruction.

The RTC instruction terminates subroutines invoked by a CALL instruction.

During the execution of an RTC instruction, the CPU:

- Pulls the old PPAGE value from the stack and loads it into the PPAGE register
- Pulls the 16-bit return address from the stack and loads it into the PC
- Resumes execution at the return address

This sequence is not interruptible; there is no need to inhibit interrupts during RTC execution. An RTC can be executed from any address in memory.

4.3.3.1.3   Data Space

The linear address pointer registers, LAP2:LAP0 along with the linear data register allow the CPU to read or write any address in the extended Flash memory space. This linear address pointer may be used to access data from any memory location while executing code from any location in extended memory, including accessing data from a different PPAGE than the currently executing program.

To access data using the linear address pointer, the user would first setup the extended address in the 22-bit address pointer, LAP2:LAP0. Accessing one of the three linear data registers LB, LBP and LWP will access the extended memory location specified by LAP2:LAP0. The three linear data registers access the memory locations in the same way, however the LBP and LWP will also increment LAP2:LAP0. Accessing either the LBP or LWP registers allows a user program to read successive memory locations without re-writing the linear address pointer. Accessing LBP or LWP does the exact same function. However, because of the address mapping of the registers with LBP following LWP, a user can do word accesses in the extended address space using the LDHX or STHX instructions to access location LWP.

The MMU supports the addition of a 2s compliment value to the linear address pointer without using any math instructions or memory resources. Writes to LAPAB with a 2s compliment value will cause the MMU to add that value to the existing value in LAP2:LAP0.

4.3.3.1.4   PPAGE and Linear Address Pointer to Extended Address

See Figure 4-5, on how the program PPAGE memory pages and the Linear Address Pointer are mapped to extended address space.

4.4   RAM (System RAM)

The MC9S08JE128 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 MC9S08JE128 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 4.6.5, "Flash Module Security," for a detailed description of the security feature.

4.5   USB RAM

USB RAM is discussed in detail in Chapter 22, "Universal Serial Bus (S08USBV1)."

The Flash memory is intended primarily 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/D.

Because the MC9S08JE128 series contains two Flash arrays, program and erase operations can be conducted on one array while executing code from the other. The security and protection features treat the two arrays as a single memory entity. Programming and erasing of each Flash array is conducted through the same command interface detailed in the following sections.

It is not possible to page erase or program both arrays at the same time. The mass erase command will erase both arrays, and the blank check command will check both arrays.

4.6   Flash

The Flash memory is intended primarily 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.

The Flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The Flash module includes a memory controller that executes commands to modify Flash memory contents.

Array read access time is one bus cycle per byte. For Flash memory, an erased bit reads 1 and a programmed bit reads 0. It is not possible to read from a Flash block while any command is executing on that specific Flash block. It is possible to read from a Flash block while a command is executing on a different Flash block.

CAUTION
A Flash block address must be in the erased state before being programmed. Cumulative programming of bits within a Flash block address is not allowed except for status field updates required in EEPROM emulation applications.

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/D.

4.6.1   Features

Features of the Flash memory include:
- • Flash size

— MC9S08JE128: 131,072 bytes (256 sectors of 512 bytes each)

— MC9S08JE64: 65,536 bytes (128 sectors of 512 bytes each)

• Single power supply program and erase

• Automated program and erase algorithm

• Fast program and erase operation

• Burst program command for faster Flash array program times

• Up to 100,000 program/erase cycles at typical voltage and temperature

• Flexible protection scheme to prevent accidental program or erase

• Security feature to prevent unauthorized access to the Flash and RAM

• Auto power-down for low-frequency read accesses

4.6.2   Register Descriptions

The Flash module contains a set of 16 control and status registers. Detailed descriptions of each register bit are provided in the following sections.

4.6.2.1   Flash Clock Divider Register (FCDIV)

The FCDIV register is used to control the length of timed events in program and erase algorithms executed by the Flash memory controller.

	7	6	5	4	3	2	1	0
R	FDIVLD	PRDIV8			FDIV			
W								
Reset	0	0	0	0	0	0	0	0
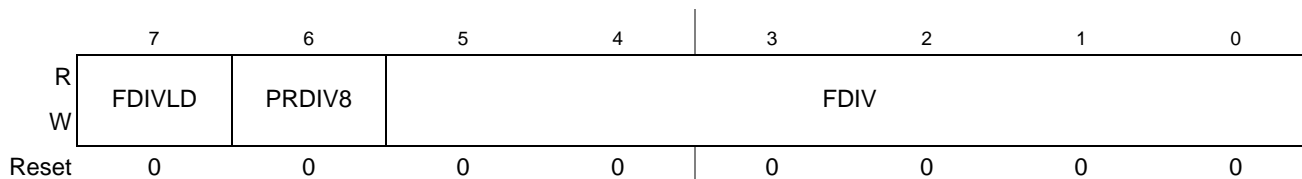
Figure 4-9. Flash Clock Divider Register (FCDIV)

All bits in the FCDIV register are readable and writable with restrictions as determined by the value of FDIVLD when writing to the FCDIV register (see Table 4-12).

Table 4-12. FCDIV Field Descriptions

Field	Description
7 FDIVLD	**Clock Divider Load Control** — When writing to the FCDIV register for the first time after a reset, the value of the FDIVLD bit written controls the future ability to write to the FCDIV register: 0 Writing a 0 to FDIVLD locks the FCDIV register contents; all future writes to FCDIV are ignored. 1 Writing a 1 to FDIVLD keeps the FCDIV register writable; next write to FCDIV is allowed. When reading the FCDIV register, the value of the FDIVLD bit read indicates the following: 0 FCDIV register has not been written to since the last reset. 1 FCDIV register has been written to since the last reset.

Table 4-12. FCDIV Field Descriptions

Field	Description
6 PRDIV8	**Enable Prescaler by 8.** 0 The bus clock is directly fed into the clock divider. 1 The bus clock is divided by 8 before feeding into the clock divider.
5:0 FDIV[5:0]	**Clock Divider Bits** — The combination of PRDIV8 and FDIV[5:0] must divide the bus clock down to a frequency of 150 kHz–200 kHz. The minimum divide ratio is 2 and the maximum divide ratio is 512. Please refer to Section 4.6.3.1.1, "Writing the FCDIV Register" for more information.

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

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

Table 4-13 shows the appropriate values for PRDIV8 and DIV for selected bus frequencies.

Table 4-13. Flash Clock Divider Settings

f_{Bus}	PRDIV8 (Binary)	DIV (Decimal)	f_{FCLK}	Program/Erase Timing Pulse (5 μs Min, 6.7 μs Max)
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

4.6.2.2   Flash Options Register (FOPT and NVOPT)

The FOPT register holds all bits associated with the security of the MCU and Flash module.

	7	6	5	4	3	2	1	0
R	KEYEN		0	0	0	0	SEC	
W								
Reset	1	1	0	0	0	0	1	1

= Unimplemented or Reserved
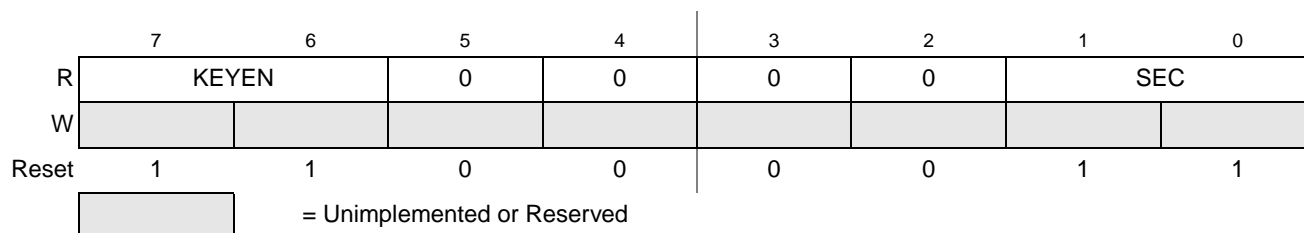
Figure 4-10. Flash Security Register (FOPT)

All bits in the FOPT register are readable but are not writable. To change the value in this register, erase and reprogram the NVSEC location in Flash memory as usual and then issue an MCU reset.

The FOPT register is loaded from the Flash location, NVSEC, during the reset sequence, indicated by F in Figure 4-10.

Table 4-14. FOPT Field Descriptions

Field	Description
7:6 KEYEN[1:0]	**Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of backdoor key access to the Flash module as shown in Table 4-15.
1:0 SEC[1:0]	**Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 4-16. If the Flash module is unsecured using backdoor key access, the SEC[1:0] bits are forced to the unsecured state.

Table 4-15. Flash KEYEN States

KEYEN[1:0]	Status of Backdoor Key Access
00	DISABLED
01[1]	DISABLED
10	ENABLED
11	DISABLED

[1] Preferred KEYEN state to disable Backdoor Key Access.

Table 4-16. Flash Security States

SEC[1:0]	Status of Security
00	SECURED
01[1]	SECURED
10	UNSECURED
11	SECURED

[1] Preferred SEC state to set MCU to secured state.

The security feature in the Flash module is described in Section 4.6.5, "Flash Module Security".

4.6.2.3 Flash Configuration Register (FCNFG)

The FCNFG register gates the security backdoor writes.

	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
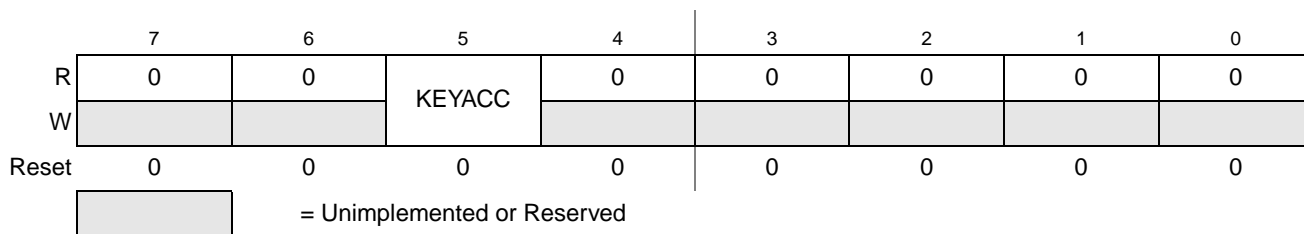
Figure 4-11. Flash Configuration Register (FCNFG)

The KEYACC bit is readable and writable while all remaining bits read 0 and are not writable. KEYACC is only writable if KEYEN is set to the enabled state (see Section 4.6.2.2, "Flash Options Register (FOPT and NVOPT)".

Table 4-17. FCNFG Field Descriptions

Field	Description
5 KEYACC	**Enable Security Key Writing** 0 Writes to the Flash block are interpreted as the start of a command write sequence. 1 Writes to the Flash block are interpreted as keys to open the backdoor.

NOTE

Flash array reads are allowed while KEYACC is set.

4.6.2.4   Flash Protection Register (FPROT and NVPROT)

The FPROT register defines which Flash sectors are protected against program or erase operations.

	7	6	5	4	3	2	1	0
R W				FPS				FPOPEN
Reset	1	1	1	1	1	1	1	1
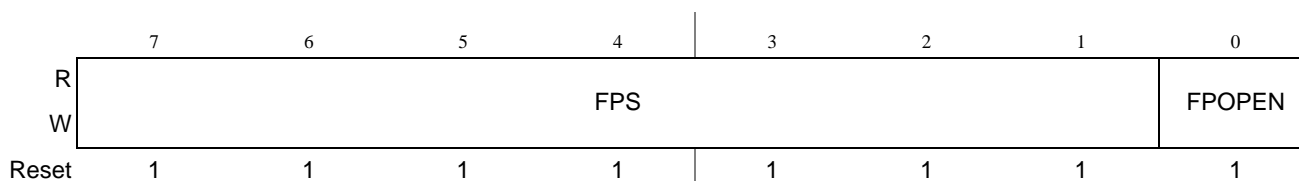
Figure 4-12. Flash Protection Register (FPROT)

FPROT bits are readable and writable as long as the size of the protected Flash memory is being increased. Any write to FPROT that attempts to decrease the size of the protected Flash memory will be ignored.

During the reset sequence, the FPROT register is loaded from the Flash protection byte, NVPROT. To change the Flash protection that will be loaded during the reset sequence, the Flash sector containing NVPROT must be unprotected and erased, then NVPROT can be reprogrammed.

Trying to alter data in any protected area in the Flash memory will result in a protection violation error and the FPVIOL flag will be set in the FSTAT register. The mass erase of the Flash array is not possible if any of the Flash sectors contained in the Flash array are protected.

Table 4-18. FPROT Field Descriptions

Field	Description
7:1 FPS[6:0]	**Flash Protection Size** — With FPOPEN set, the FPS bits determine the size of the protected Flash address range as shown in Table 4-19.
0 FPOPEN	**Flash Protection Open** 0 Flash array fully protected. 1 Flash array protected address range determined by FPS bits.

Table 4-19. Flash Protection Address Range

FPS[6:0]	FPOPEN	Protected Address Range Relative to Flash Array Base		Protected Size
		Flash Array 0	**Flash Array 1**	
—	0	0x0_0000–0x0_FFFF	0x1_0000–0x1_FFFF	128 Kbytes
0x00	1	0x0_0000–0x0_FFFF	0x1_0400–0x1_FFFF	127 Kbytes
0x01		0x0_0000–0x0_FFFF	0x1_0800–0x1_FFFF	126 Kbytes
0x02		0x0_0000–0x0_FFFF	0x1_0C00–0x1_FFFF	125 Kbytes
0x03		0x0_0000–0x0_FFFF	0x1_1000–0x1_FFFF	124 Kbytes
0x04		0x0_0000–0x0_FFFF	0x1_1400–0x1_FFFF	123 Kbytes
0x05		0x0_0000–0x0_FFFF	0x1_1800–0x1_FFFF	122 Kbytes
0x06		0x0_0000–0x0_FFFF	0x1_1C00–0x1_FFFF	121 Kbytes
…		...	...	…
0x37		0x0_0000–0x0_FFFF	0x1_E000–0x1_FFFF	72 Kbytes
0x38		0x0_0000–0x0_FFFF	0x1_E400–0x1_FFFF	71 Kbytes
0x39		0x0_0000–0x0_FFFF	0x1_E800–0x1_FFFF	70 Kbytes
0x3A		0x0_0000–0x0_FFFF	0x1_EC00–0x1_FFFF	69 Kbytes
0x3B		0x0_0000–0x0_FFFF	0x1_F000–0x1_FFFF	68 Kbytes
0x3C		0x0_0000–0x0_FFFF	0x1_F400–0x1_FFFF	67 Kbytes
0x3D		0x0_0000–0x0_FFFF	0x1_F800–0x1_FFFF	66 Kbytes
0x3E		0x0_0000–0x0_FFFF	0x1_FC00–0x1_FFFF	65 Kbytes
0x3F		0x0_0000–0x0_FFFF		64 Kbytes
0x40		0x0_0400–0x0_FFFF		63 Kbytes
0x41		0x0_0800–0x0_FFFF		62 Kbytes
0x42		0x0_0C00–0x0_FFFF		61 Kbytes
0x43		0x0_1000–0x0_FFFF		60 Kbytes
0x44		0x0_1400–0x0_FFFF		59 Kbytes
0x45		0x0_1800–0x0_FFFF		58 Kbytes
0x46		0x0_1C00–0x0_FFFF		57 Kbytes
…		…		…
0x77		0x0_E000–0x0_FFFF		8 Kbytes
0x78		0x0_E400–0x0_FFFF		7 Kbytes
0x79		0x0_E800–0x0_FFFF		6 Kbytes
0x7A		0x0_EC00–0x0_FFFF		5 Kbytes
0x7B		0x0_F000–0x0_FFFF		4 Kbytes
0x7C		0x0_F400–0x0_FFFF		3 Kbytes
0x7D		0x0_F800–0x0_FFFF		2 Kbytes
0x7E		0x0_FC00–0x0_FFFF		1 Kbyte
0x7F		No Protection		0 Kbytes

4.6.2.5 Flash Status Register (FSTAT)

The FSTAT register defines the operational status of the Flash module.

FCBEF, FPVIOL, and FACCERR are readable and writable, FCCF and FBLANK are readable and not writable, remaining bits read 0 and are not writable.

	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
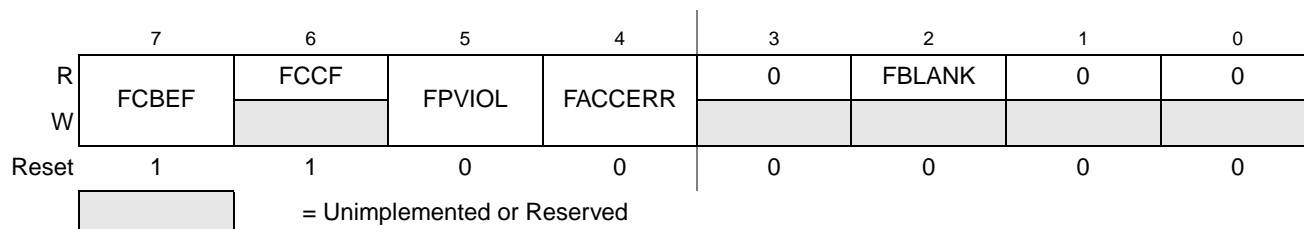
Figure 4-13. Flash Status Register (FSTAT)

Table 4-20. FSTAT Field Descriptions

Field	Description
7 FCBEF	**Flash Command Buffer Empty Flag** — The FCBEF flag indicates that the command buffer is empty so that a new command write sequence can be started when performing burst programming. Writing a 0 to the FCBEF flag has no effect on FCBEF. Writing a 0 to FCBEF after writing an aligned address to the Flash array memory, but before FCBEF is cleared, will abort a command write sequence and cause the FACCERR flag to be set. Writing a 0 to FCBEF outside of a command write sequence will not set the FACCERR flag. The FCBEF flag is cleared by writing a 1 to FCBEF. 0 Command buffers are full. 1 Command buffers are ready to accept a new command.
6 FCCF	**Flash Command Complete Interrupt Flag** — The FCCF flag indicates that there are no more commands pending. The FCCF flag is cleared when FCBEF is cleared and sets automatically upon completion of all active and pending commands. The FCCF flag does not set when an active program command completes and a pending burst program command is fetched from the command buffer. Writing to the FCCF flag has no effect on FCCF. 0 Command in progress. 1 All commands are completed.
5 FPVIOL	**Flash Protection Violation Flag** —The FPVIOL flag indicates an attempt was made to program or erase an address in a protected area of the Flash memory or Flash IFR during a command write sequence. Writing a 0 to the FPVIOL flag has no effect on FPVIOL. The FPVIOL flag is cleared by writing a 1 to FPVIOL. While FPVIOL is set, it is not possible to launch a command or start a command write sequence. 0 No protection violation detected. 1 Protection violation has occurred.

Table 4-20. FSTAT Field Descriptions

Field	Description
4 FACCERR	**Flash Access Error Flag** — The FACCERR flag indicates an illegal access has occurred to the Flash memory or Flash IFR caused by either a violation of the command write sequence (see Section 4.6.3.1.2, "Command Write Sequence"), issuing an illegal Flash command (see Table 4-22), or the execution of a CPU STOP instruction while a command is executing (FCCF = 0). Writing a 0 to the FACCERR flag has no effect on FACCERR. The FACCERR flag is cleared by writing a 1 to FACCERR.While FACCERR is set, it is not possible to launch a command or start a command write sequence. 0 No access error detected. 1 Access error has occurred.
2 FBLANK	**Flash Flag Indicating the Erase Verify Operation Status** — When the FCCF flag is set after completion of an erase verify command, the FBLANK flag indicates the result of the erase verify operation. The FBLANK flag is cleared by the Flash module when FCBEF is cleared as part of a new valid command write sequence. Writing to the FBLANK flag has no effect on FBLANK. 0 Flash block verified as not erased. 1 Flash block verified as erased.

4.6.2.6   Flash Command Register (FCMD)

The FCMD register is the Flash command register.

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

= Unimplemented or Reserved
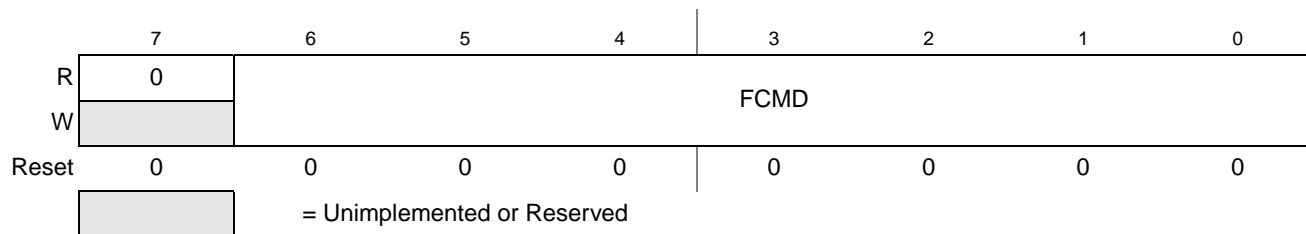
Figure 4-14. Flash Command Register (FCMD)

All FCMD bits are readable and writable during a command write sequence while bit 7 reads 0 and is not writable.

Table 4-21. FCMD Field Descriptions

Field	Description
6:0 FCMD[6:0]	**Flash Command** — Valid Flash commands are shown in Table 4-22. Writing any command other than those listed in Table 4-22 sets the FACCERR flag in the FSTAT register.

Table 4-22. Valid Flash Command List

FCMD[6:0]	NVM Command
0x05	Erase Verify
0x20	Program
0x25	Burst Program
0x40	Sector Erase
0x41	Mass Erase

4.6.3 Functional Description

4.6.3.1 Flash Command Operations

Flash command operations are used to execute program, erase, and erase verify algorithms described in this section. The program and erase algorithms are controlled by the Flash memory controller whose time base, FCLK, is derived from the bus clock via a programmable divider.

The next sections describe:

- How to write the FCDIV register to set FCLK
- Command write sequences to program, erase, and erase verify operations on the Flash memory
- Valid Flash commands
- Effects resulting from illegal Flash command write sequences or aborting Flash operations

4.6.3.1.1 Writing the FCDIV Register

Prior to issuing any Flash command after a reset, the user is required to write the FCDIV register to divide the bus clock down to within the 150 kHz to 200 kHz range. 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 4-23 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 4-23. 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

NOTE

Program and erase command execution time will increase proportionally with the period of FCLK. Programming or erasing the Flash memory with FCLK < 150 kHz should be avoided. Setting FCDIV to a value such that FCLK < 150 kHz can destroy the Flash memory due to overstress. Setting FCDIV to a value such that FCLK > 200 kHz can result in incomplete programming or erasure of the Flash memory cells.

If the FCDIV register is written, the FDIVLD bit is set automatically. If the FDIVLD bit is 0, the FCDIV register has not been written since the last reset. If the FCDIV register has not been written to, the Flash command loaded during a command write sequence will not execute and the FACCERR flag in the FSTAT register will set.

4.6.3.1.2   Command Write Sequence

The Flash command controller is used to supervise the command write sequence to execute program, erase, and erase verify algorithms.

Before starting a command write sequence, the FACCERR and FPVIOL flags in the FSTAT register must be clear and the FCBEF flag must be set (see Section 4.6.2.5).

A command write sequence consists of three steps which must be strictly adhered to with writes to the Flash module not permitted between the steps. However, Flash register and array reads are allowed during a command write sequence. The basic command write sequence is as follows:

1. Write to a valid address in the Flash array memory.
2. Write a valid command to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the command.

Once a command is launched, the completion of the command operation is indicated by the setting of the FCCF flag in the FSTAT register. The FCCF flag will set upon completion of all active and buffered burst program commands.

4.6.3.2   Flash Commands

Table 4-24 summarizes the valid Flash commands along with the effects of the commands on the Flash block.

Table 4-24. Flash Command Description

FCMDB	NVM Command	Function on Flash Memory
0x05	Erase Verify	Verify all memory bytes in the Flash array memory are erased. If the Flash array memory is erased, the FBLANK flag in the FSTAT register will set upon command completion.
0x20	Program	Program an address in the Flash array.
0x25	Burst Program	Program an address in the Flash array with the internal address incrementing after the program operation.
0x40	Sector Erase	Erase all memory bytes in a sector of the Flash array.
0x41	Mass Erase	Erase all memory bytes in the Flash array. A mass erase of the full Flash array is only possible when no protection is enabled prior to launching the command.

CAUTION

A Flash block address must be in the erased state before being programmed. Cumulative programming of bits within a Flash block address is not allowed except for status field updates required in EEPROM emulation applications.

4.6.3.2.1 Erase Verify Command

The erase verify operation will verify that a Flash block is erased.

An example flow to execute the erase verify operation is shown in Figure 4-15. The erase verify command write sequence is as follows:

1. Write to a Flash block address to start the command write sequence for the erase verify command. The address and data written will be ignored.
2. Write the erase verify command, 0x05, to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the erase verify command.

After launching the erase verify command, the FCCF flag in the FSTAT register will set after the operation has completed. The number of bus cycles required to execute the erase verify operation is equal to the number of addresses in the Flash array memory plus several bus cycles as measured from the time the FCBEF flag is cleared until the FCCF flag is set. Upon completion of the erase verify operation, the FBLANK flag in the FSTAT register will be set if all addresses in the Flash array memory are verified to be erased. If any address in the Flash array memory is not erased, the erase verify operation will terminate and the FBLANK flag in the FSTAT register will remain clear.
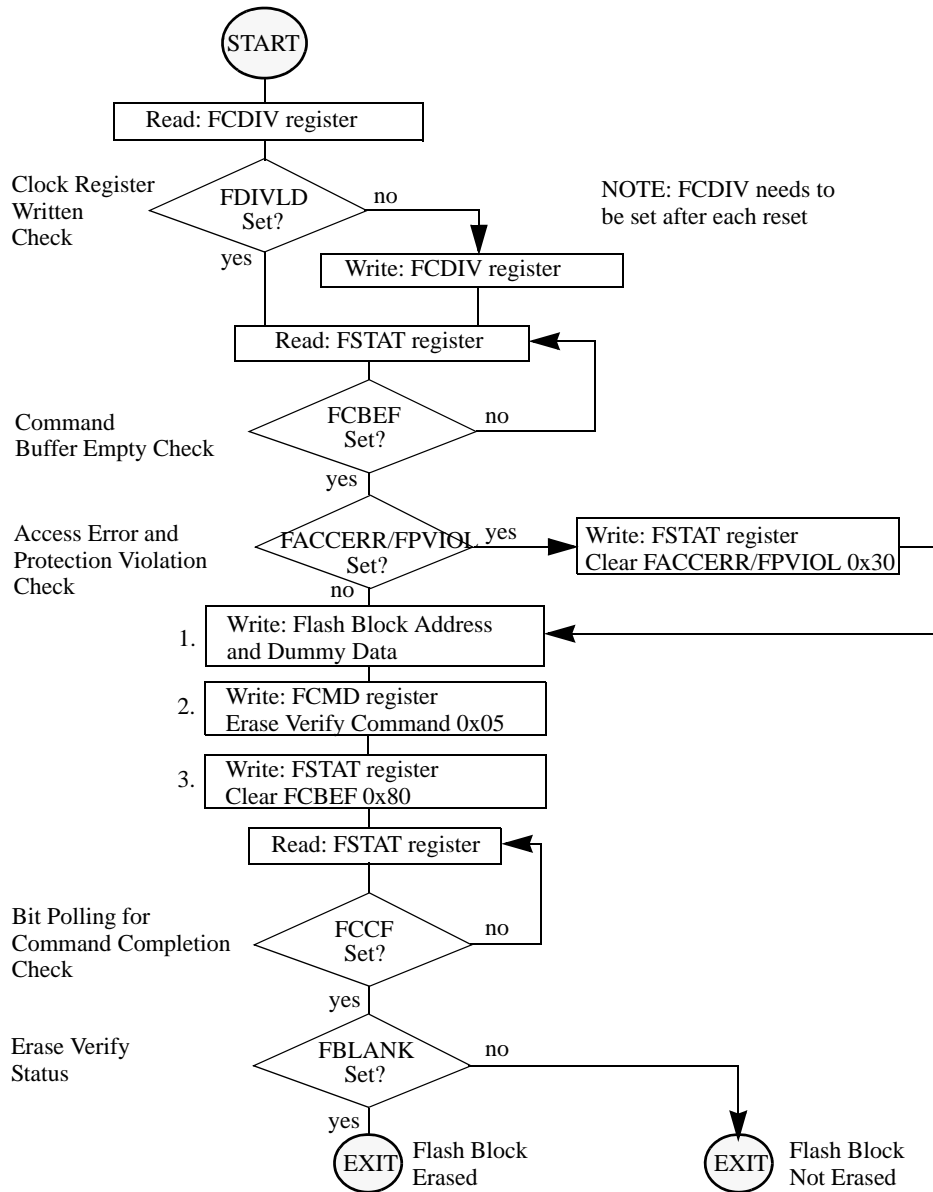
Figure 4-15. Example Erase Verify Command Flow

4.6.3.2.2 Program Command

The program operation will program a previously erased address in the Flash memory using an embedded algorithm.

An example flow to execute the program operation is shown in Figure 4-16. The program command write sequence is as follows:

1. Write to a Flash block address to start the command write sequence for the program command. The data written will be programmed to the address written.
2. Write the program command, 0x20, to the FCMD register.

3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the program command.

If an address to be programmed is in a protected area of the Flash block, the FPVIOL flag in the FSTAT register will set and the program command will not launch. Once the program command has successfully launched, the FCCF flag in the FSTAT register will set after the program operation has completed.
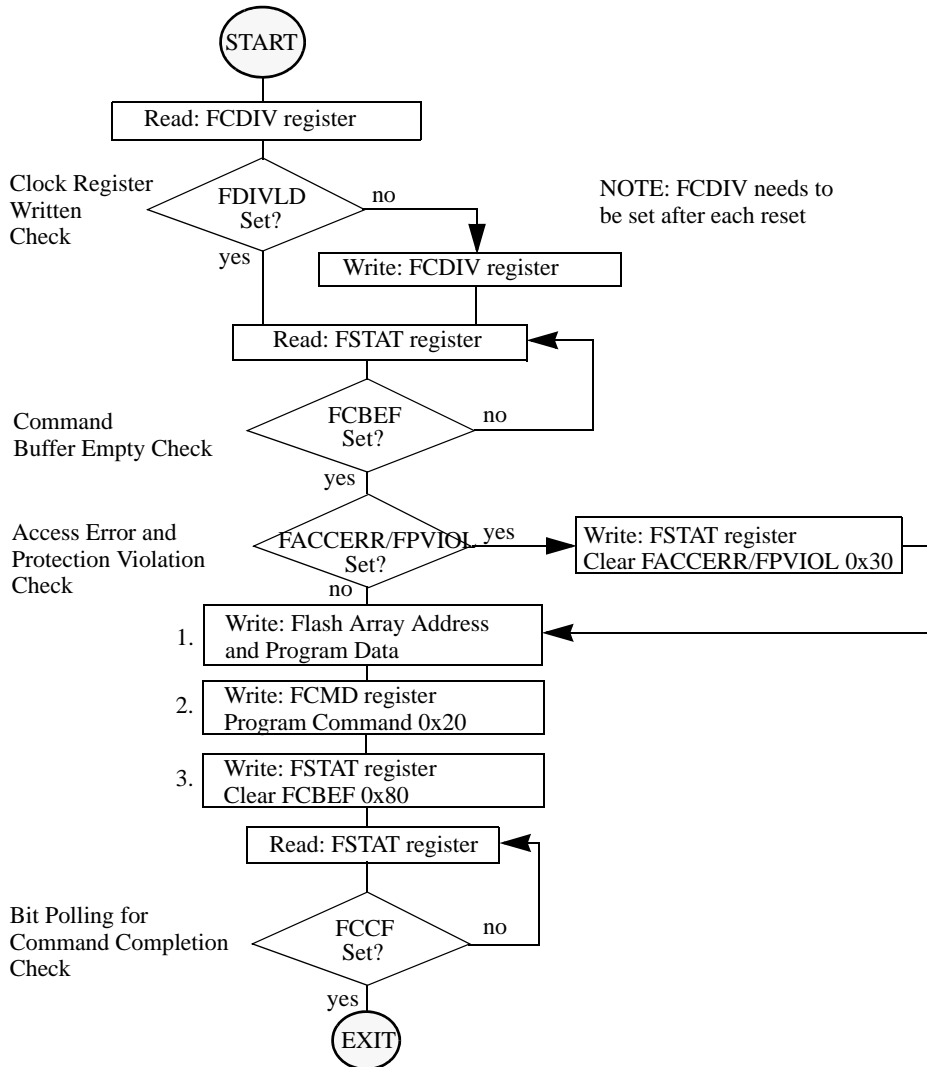
Figure 4-16. Example Program Command Flow

4.6.3.2.3 Burst Program Command

The burst program operation will program previously erased data in the Flash memory using an embedded algorithm.

While burst programming, two internal data registers operate as a buffer and a register (2-stage FIFO) so that a second burst programming command along with the necessary data can be stored to the buffers while the first burst programming command is still in progress. This pipelined operation allows a time

optimization when programming more than one consecutive address on a specific row in the Flash array as the high voltage generation can be kept active in between two programming commands.

An example flow to execute the burst program operation is shown in Figure 4-17. The burst program command write sequence is as follows:

1. Write to a Flash block address to start the command write sequence for the burst program command. The data written will be programmed to the address written.
2. Write the program burst command, 0x25, to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the program burst command.
4. After the FCBEF flag in the FSTAT register returns to a 1, repeat steps 1 through 3. The address written is ignored but is incremented internally.

The burst program procedure can be used to program the entire Flash memory even while crossing row boundaries within the Flash array. If data to be burst programmed falls within a protected area of the Flash array, the FPVIOL flag in the FSTAT register will set and the burst program command will not launch. Once the burst program command has successfully launched, the FCCF flag in the FSTAT register will set after the burst program operation has completed unless a new burst program command write sequence has been buffered. By executing a new burst program command write sequence on sequential addresses after the FCBEF flag in the FSTAT register has been set, greater than 50% faster programming time for the entire Flash array can be effectively achieved when compared to using the basic program command.
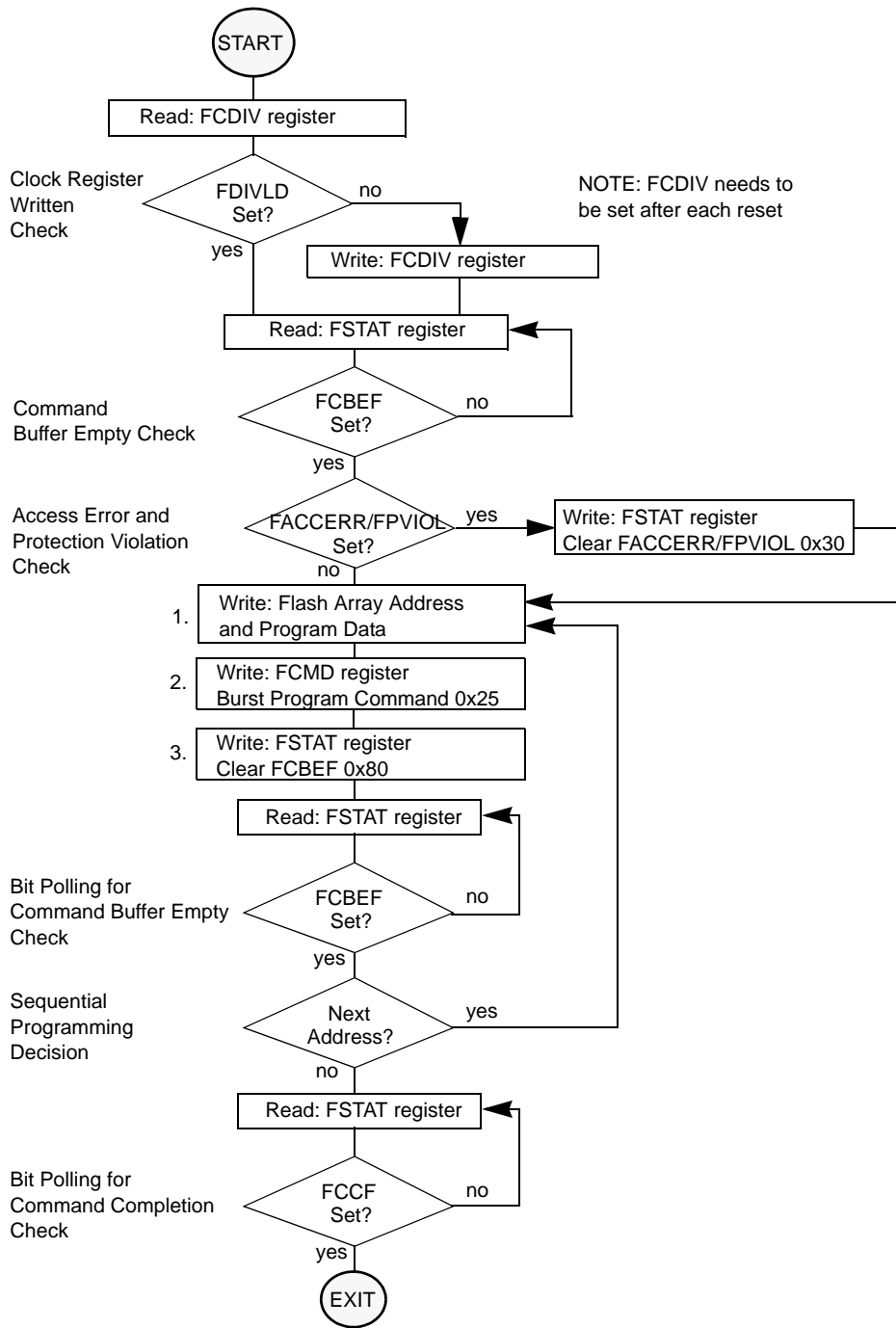
Figure 4-17. Example Burst Program Command Flow

4.6.3.2.4   Sector Erase Command

The sector erase operation will erase all addresses in a 1 Kbyte sector of Flash memory using an embedded algorithm.

An example flow to execute the sector erase operation is shown in Figure 4-18. The sector erase command write sequence is as follows:

1. Write to a Flash block address to start the command write sequence for the sector erase command. The Flash address written determines the sector to be erased while global address bits [8:0] and the data written are ignored.
2. Write the sector erase command, 0x40, to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the sector erase command.

If a Flash sector to be erased is in a protected area of the Flash block, the FPVIOL flag in the FSTAT register will set and the sector erase command will not launch. Once the sector erase command has successfully launched, the FCCF flag in the FSTAT register will set after the sector erase operation has completed.
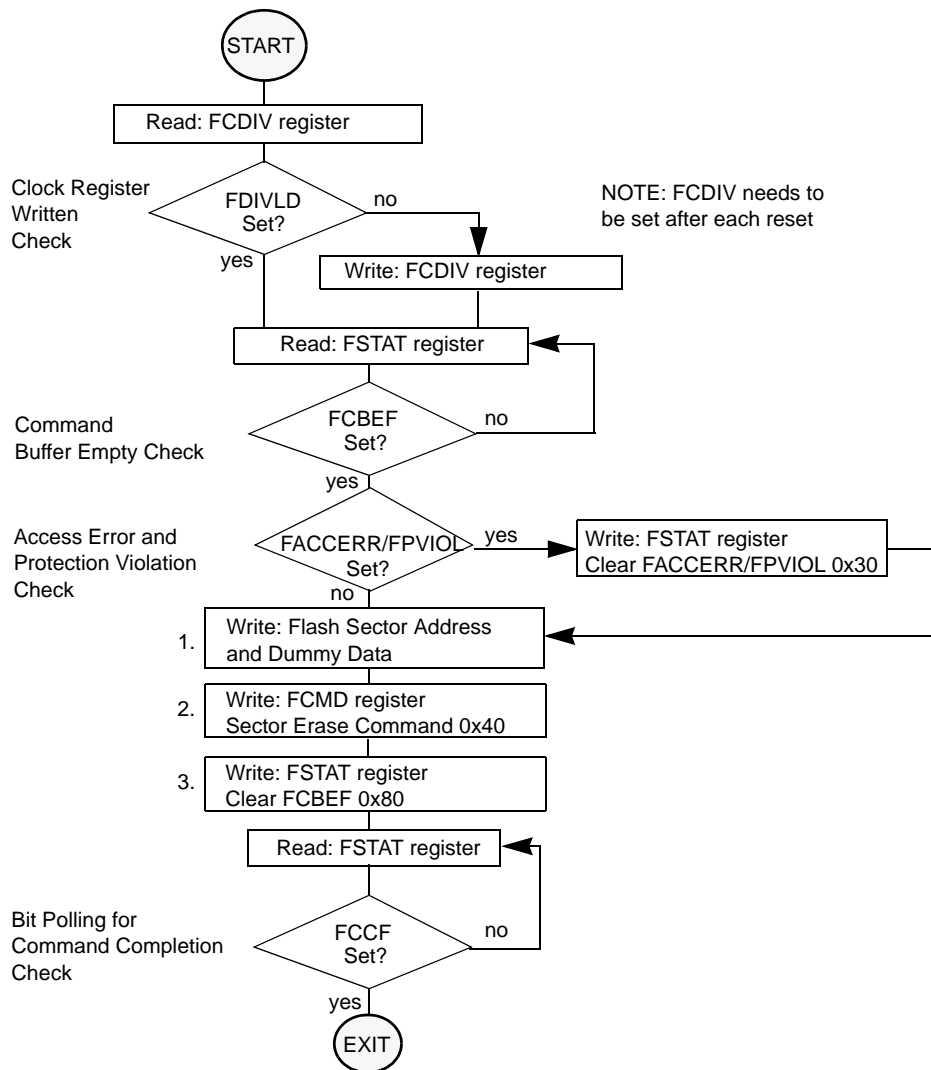
Figure 4-18. Example Sector Erase Command Flow

4.6.3.2.5 Mass Erase Command

The mass erase operation erases the entire flash array memory using an embedded algorithm. An example flow to execute the mass erase operation is shown in Figure 4-19. The mass erase command write sequence is as follows:

1. Write to a flash block address to start the command write sequence for the mass erase command. The address and data written is ignored.
2. Write the mass erase command, 0x41, to the FCMD register.
3. Clear the FCBEF flag in the FSTAT register by writing a 1 to FCBEF to launch the mass erase command.

If the flash array memory to be mass erased contains any protected area, FSTAT[FPVIOL] is set and the mass erase command does not launch. After the mass erase command has successfully launched and the mass erase operation has completed, FSTAT[FCCF] is set.
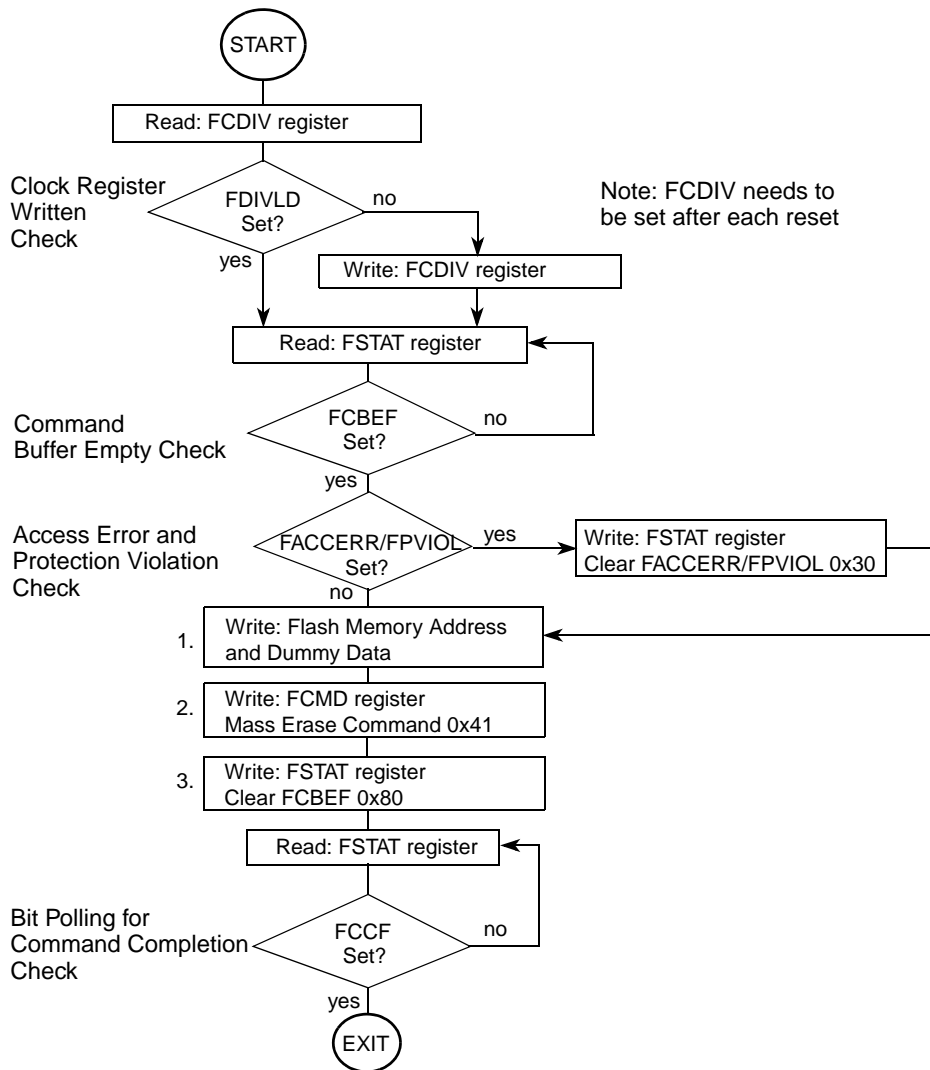
Figure 4-19. Example Mass Erase Command Flow

NOTE

The BDM can also perform a mass erase and verify command. See
Chapter 24, "Development Support," for details.

4.6.3.3   Illegal Flash Operations

4.6.3.3.1   Flash Access Violations

The FACCERR flag will be set during the command write sequence if any of the following illegal steps
are performed, causing the command write sequence to immediately abort:

- Writing to a Flash address before initializing the FCDIV register.
- Writing to any Flash register other than FCMD after writing to a Flash address.
- Writing to a second Flash address in the same command write sequence.
- Writing an invalid command to the FCMD register unless the address written was in a protected area of the Flash array.
- Writing a command other than burst program while FCBEF is set and FCCF is clear.
- When security is enabled, writing a command other than mass erase to the FCMD register when the write originates from a non-secure memory location or from the background debug mode.
- Writing to a Flash address after writing to the FCMD register.
- Writing to any Flash register other than FSTAT (to clear FCBEF) after writing to the FCMD register.
- Writing a 0 to the FCBEF flag in the FSTAT register to abort a command write sequence.

The FACCERR flag will also be set if the MCU enters stop mode while a program or erase operation is
active. The operation is aborted immediately and, if burst programming, any pending burst program
command is purged (see Section 4.6.4.2, "Stop Mode").

The FACCERR flag will not be set if any Flash register is read during a valid command write sequence.

If the Flash memory is read during execution of an algorithm (FCCF = 0), the read operation will return
invalid data and the FACCERR flag will not be set.

If the FACCERR flag is set in the FSTAT register, the user must clear the FACCERR flag before starting
another command write sequence (see Section 4.6.2.5, "Flash Status Register (FSTAT)").

4.6.3.3.2   Flash Protection Violations

The FPVIOL flag will be set after the command is written to the FCMD register during a command write
sequence if any of the following illegal operations are attempted, causing the command write sequence to
immediately abort:

- Writing the program command if the address written in the command write sequence was in a protected area of the Flash array.
- Writing the sector erase command if the address written in the command write sequence was in a protected area of the Flash array.
- Writing the mass erase command while any Flash protection is enabled.

- Writing an invalid command if the address written in the command write sequence was in a protected area of the Flash array.

If the FPVIOL flag is set in the FSTAT register, the user must clear the FPVIOL flag before starting another command write sequence (see Section 4.6.2.5, "Flash Status Register (FSTAT)").

4.6.4 Operating Modes

4.6.4.1 Wait Mode

If a command is active (FCCF = 0) when the MCU enters wait mode, the active command and any buffered command will be completed.

4.6.4.2 Stop Mode

If a command is active (FCCF = 0) when the MCU enters stop mode, the operation will be aborted and, if the operation is program or erase, the Flash array data being programmed or erased may be corrupted and the FCCF and FACCERR flags will be set. If active, the high voltage circuitry to the Flash array will immediately be switched off when entering stop mode. Upon exit from stop mode, the FCBEF flag is set and any buffered command will not be launched. The FACCERR flag must be cleared before starting a command write sequence (see Section 4.6.3.1.2, "Command Write Sequence").

NOTE

As active commands are immediately aborted when the MCU enters stop mode, it is strongly recommended that the user does not use the STOP instruction during program or erase operations.

4.6.4.3 Background Debug Mode

In background debug mode (BDM), the FPROT register is writable. If the MCU is unsecured, then all Flash commands listed in Table 4-24 can be executed.

4.6.5 Flash Module Security

The MC9S08JE128 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).

The Flash module provides the necessary security information to the MCU. During each reset sequence, the Flash module determines the security state of the MCU as defined in Section 4.6.2.2, "Flash Options Register (FOPT and NVOPT)".

The contents of the Flash security byte in NVSEC must be changed directly by programming the NVSEC location when the MCU is unsecured and the sector containing NVSEC is unprotected. If NVSEC is left in a secured state, any reset will cause the MCU to initialize into a secure operating mode.

4.6.5.1   Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (NVBACKKEY through NVBACKKEY+7, see Table 4-4 for specific addresses). If the KEYEN[1:0] bits are in the enabled state (see Section 4.6.2.2) and the KEYACC bit is set, a write to a backdoor key address in the Flash memory triggers a comparison between the written data and the backdoor key data stored in the Flash memory. If all backdoor keys are written to the correct addresses in the correct order and the data matches the backdoor keys stored in the Flash memory, the MCU will be unsecured. The data must be written to the backdoor keys sequentially. Values 0x0000 and 0xFFFF are not permitted as backdoor keys. While the KEYACC bit is set, reads of the Flash memory will return invalid data.

The user code stored in the Flash memory must have a method of receiving the backdoor keys from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If the KEYEN[1:0] bits are in the enabled state (see Section 4.6.2.2), the MCU can be unsecured by the backdoor key access sequence described below:

1. Set the KEYACC bit in the Flash configuration register (FCNFG).
2. Sequentially write the correct eight 8-bit bytes to the Flash addresses containing the backdoor keys.
3. Clear the KEYACC bit. Depending on the user code used to write the backdoor keys, a wait cycle (NOP) may be required before clearing the KEYACC bit.
4. If all data written match the backdoor keys, the MCU is unsecured and the SEC[1:0] bits in the FSEC register are forced to the unsecure state of 1:0.

The backdoor key access sequence is monitored by an internal security state machine. An illegal operation during the backdoor key access sequence will cause the security state machine to lock, leaving the MCU in the secured state. A reset of the MCU will cause the security state machine to exit the lock state and allow a new backdoor key access sequence to be attempted. The following operations during the backdoor key access sequence will lock the security state machine:

1. If any of the keys written does not match the backdoor keys programmed in the Flash array.
2. If the keys are written in the wrong sequence.
3. If more keys than are required are written.
4. If any of the keys written are all 0's or all 1's.
5. If the KEYACC bit does not remain set while the keys are written.
6. If any of the keys are written on successive MCU clock cycles.
7. Executing a STOP instruction while the KEYACC bit is set.

After the backdoor keys have been correctly matched, the MCU will be unsecured. Once the MCU is unsecured, the Flash security byte can be programmed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the backdoor keys by programming the associated addresses in NVBACKKEY through NVBACKKEY+7.

The security as defined in the Flash security byte is not changed by using the backdoor key access sequence to unsecure. The stored backdoor keys are unaffected by the backdoor key access sequence. After the next reset of the MCU, the security state of the Flash module is determined by the Flash security byte. The backdoor key access sequence has no effect on the program and erase protections defined in the Flash protection register (FPROT).

It is not possible to unsecure the MCU in special mode by using the backdoor key access sequence in background debug mode (BDM).

4.6.6   Resets

If a reset occurs while any Flash command is in progress, that command will be immediately aborted. The state of the Flash array address being programmed or the sector/block being erased is not guaranteed.

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 MC9S08JE128 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 pull-up 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 MC9S08JE128 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 1 (SOPT1) Register," 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 2 (SOPT2) Register," 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 5-7 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.

* High byte (H) of index register is not automatically stacked.
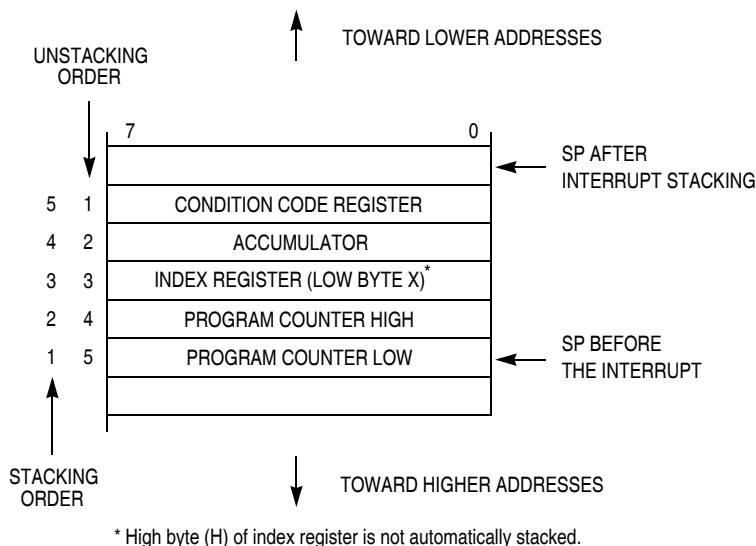
Figure 5-1. Interrupt Stack Frame

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 pull-up or pull-down depending on the polarity chosen. If the user desires to use an external pull-up 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. MC9S08JE128 Interrupt Vectors

Vector Priority	Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
Lowest							
	32	0xFF9E/0xFF9F	Vsci2tx	SCI2	TDRE, TC	TIE, TCIE	SCI2 transmit
	31	0xFFC0/0xFFC1	Vsci2rx	SCI2	IDLE, RDRF, LBKDIF, RXEDGIF	ILIE, RIE, LBKDIE, RXEDGIE	SCI2 receive
	30	0xFFC2/0xFFC3	Vsci2err	SCI2	OR, NF, FE, PF	ORIE, NFIE, FEIE, PFIE	SCI2 error
	29	0xFFC4/0xFFC5	Vtod	TOD	QSECF, SECF, MTCHF	QSECIE, SECIE, MTCHIE	Time of Day interrupt
	28	0xFFC6/0xFFC7	Vkeyboard2	KBI2	KBF	KBIE	Keyboard2 pins
	27	0xFFC8/0xFFC9	Vkeyboard1	KBI1	KBF	KBIE	Keyboard1 pins
	26	0xFFCA/0xFFCB	Vacmp	PRACMP	ACF	ACIE	Prog. Analog comparator
	25	0xFFCC/0xFFCD	Vadc	ADC12	COCO	AIEN	ADC
	24	0xFFCE/0xFFCF	Viic	IIC	IICIS	IICIE	IIC control
	23	0xFFD0/0xFFD1	Vsci1tx	SCI1	TDRE, TC	TIE, TCIE	SCI1 transmit
	22	0xFFD2/0xFFD3	Vsci1rx	SCI1	IDLE, RDRF, LBKDIF, RXEDGIF	ILIE, RIE, LBKDIE, RXEDGIE	SCI1 receive
	21	0xFFD4/0xFFD5	Vsci1err	SCI1	OR, NF, FE, PF	ORIE, NFIE, FEIE, PFIE	SCI1 error
	20	0xFFD6/0xFFD7	Vcmt	CMT	EOCF	EOCIE	CMT
	19	0xFFD8/0xFFD9	Vspi2	SPI2	SPIF, MODF, SPTEF	SPIE, SPE, SPTIE	SPI2
	18	0xFFDA/0xFFDB	Vtpm2ovf	TPM2	TOF	TOIE	TPM2 overflow
	17	0xFFDC/0xFFDD	Vtpm2ch3	TPM2	CH3F	CH3IE	TPM2 channel 3
	16	0xFFDE/0xFFDF	Vtpm2ch2	TPM2	CH2F	CH2IE	TPM2 channel 2
	15	0xFFE0/0xFFE1	Vtpm2ch1	TPM2	CH1F	CH1IE	TPM2 channel 1
	14	0xFFE2/0xFFE3	Vtpm2ch0	TPM2	CH0F	CH0IE	TPM2 channel 0
	13	0xFFE4/0xFFE5	Vtpm1ovf	TPM1	TOF	TOIE	TPM1 overflow
	12	0xFFE6/0xFFE7	Vtpm1ch3	TPM1	CH3F	CH3IE	TPM1 channel 3
	11	0xFFE8/0xFFE9	Vtpm1ch2	TPM1	CH2F	CH2IE	TPM1 channel 2
	10	0xFFEA/0xFFEB	Vtpm1ch1	TPM1	CH1F	CH1IE	TPM1 channel 1
	9	0xFFEC/0xFFED	Vtpm1ch0	TPM1	CH0F	CH0IE	TPM1 channel 0
	8	0xFFEE/0xFFEF	Vdac	DAC	DACWMF, DACRPTF, DACRPBF	DACWMIE DACTIE DACBIE	DAC Buffer
	7	0xFFF0/0xFFF1	Vpdb	PDB	PDB_SCR[IF]	PDB_SCR[IE]	PDB
	6	0xFFF2/0xFFF3	Vusb	USB	STALLF RESUMEF SLEEPF TOKDNEF SOFTOKF ERRORF USBRSTF	STALL RESUME SLEEP TOKDNE SOFTOK ERROR USBRST	USB Status
	5	0xFFF4/0xFFF5	Vspi1	SPI1	SPIF, MODF, SPTEF	SPIE, SPE, SPTIE	FSPI1
	4	0xFFF6/0xFFF7	Vlol	MCG	LOLS	LOLIE	MCG Loss of Lock
	3	0xFFF8/0xFFF9	Vlvd	System control	LVDF, LVWF	LVDIE, LVWIE	Low-voltage detect, Low-voltage warning
	2	0xFFFA/0xFFFB	Virq	IRQ	IRQF	IRQIE	IRQ pin
	1	0xFFFC/0xFFFD	Vswi	Core	SWI Instruction	—	Software interrupt
Highest	0	0xFFFE/0xFFFF	Vreset	System control	COP LVD RESET pin Illegal opcode Illegal address POR	COPE LVDRE RSTPE — —	Watchdog timer Low-voltage detect External pin Illegal opcode Illegal address

5.6 Low-Voltage Detect (LVD) System

The MC9S08JE128 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)

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

☐ = Unimplemented or Reserved
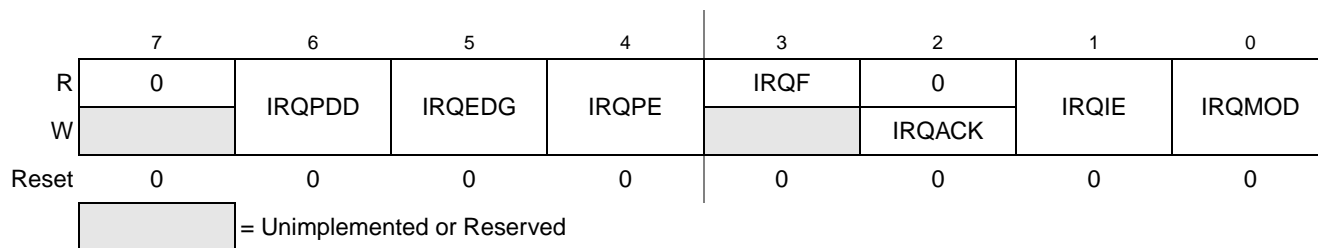
Figure 5-2. Interrupt Request Status and Control Register (IRQSC)

Table 5-2. Register Field Descriptions

Field	Description
6 IRQPDD	**Interrupt Request (IRQ) Pull Device Disable** — This read/write control bit is used to disable the internal pull-up 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 pull-up resistor is re-configured as an optional pull-down 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.
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 is set.

To reset the COP counter, write 0x55 and 0xAA (in this order) to the address of SRS during the selected timeout period. The reset state of these bits depends on what caused the microcontroller to reset.

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	ILAD	LOC	LVD	—
W	Writing back-to-back 0x55 and 0xAA in sequence clears COP the 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)	(1)	(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** — A 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.
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.
3 ILAD	**ILAD** — Reset was caused by an attempt to access an illegal address. 0 Reset not caused by an illegal address access. 1 Reset caused by an illegal address access.
2 LOC	**Loss-of-Clock Reset** — A 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
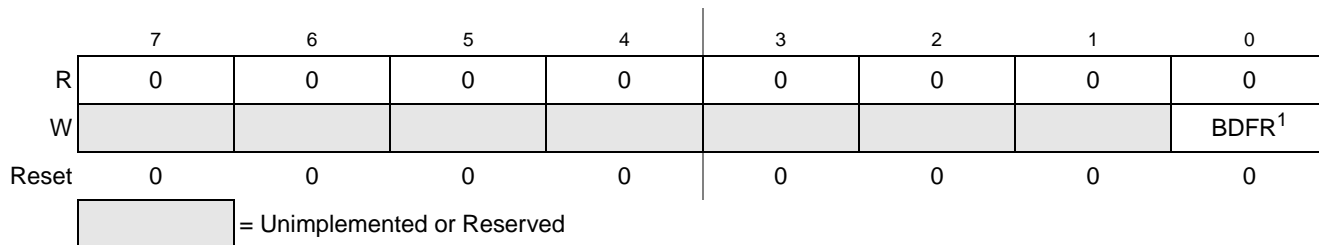
[1] BDFR is writable only through serial background debug commands, not from user programs.

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 1 (SOPT1) Register

This high-page register has five write-once bits and one write anytime bit. For the write-once bits, only the first write after reset is honored. All bits in the register can be read at any time. Any subsequent attempt to write a write-once bit is ignored to avoid accidental changes to these sensitive settings. SOPT1 should be written to during the 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	COPT[1]		STOPE[1]	0	BLMSS	0	BKGDPE[1]	RSTPE[1]
W								
POR	1	1	0	0	0	0	1	1
LVR	1	1	0	0	0	0	1	1
Any Other Reset:	1	1	0	0	0	0	1	u[2]
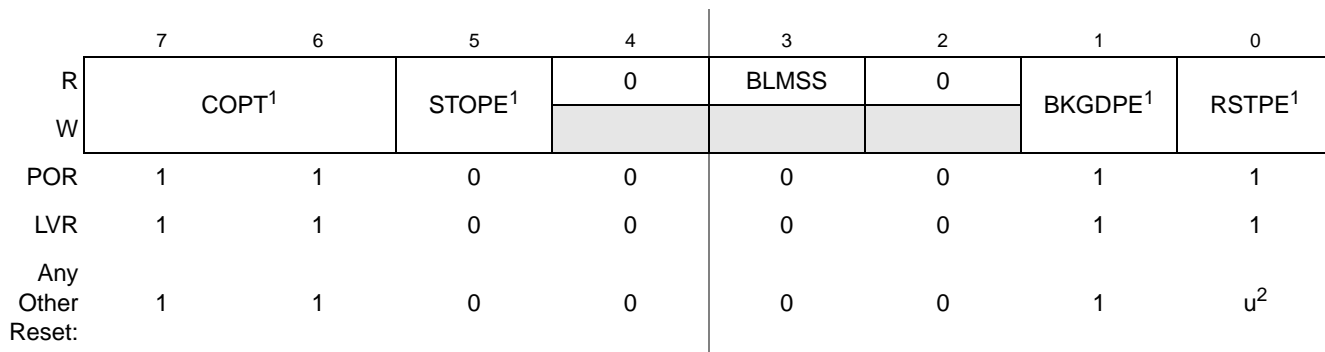
Figure 5-5. System Options 1 (SOPT1) Register

[1] These bits can be written only one time after reset. Subsequent writes are ignored.

[2] u = Unaffected

Table 5-5. SOPT1 Field Descriptions

Field	Description
7–6 COPT	**COP Watchdog Timeout** — These write-once bits select the timeout period of the COP. COPT along with SOPT2[COPCLKS] defines the COP timeout period as described in Table 5-7.
5 STOPE	**Stop Mode Enable** — This write-once bit is used to enable stop mode. If stop and wait modes are disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset may be generated.
3 BLMSS	**Boot Loader Mode Select Status**— This Read only bit shows the status of the $\overline{BLMS}$ bit during the last Power on Reset. 0 $\overline{BLMS}$ pin was high at last POR 1 $\overline{BLMS}$ pin was low at last POR.
1 BKGDPE	**Background Debug Mode Pin Enable** — This write-once bit when set enables the PTD0/BKGD/MS pin to function as BKGD/MS. When clear, the pin functions as one of its output only alternative functions. This pin defaults to the BKGD/MS function following any MCU reset. 0 PTD0/BKGD/MS pin functions as PTD0 1 PTD0/BKGD/MS pin functions as BKGD/MS.
0 RSTPE	$\overline{\text{RESET}}$ **Pin Enable** — This write-once bit when set enables the PTD1/CMPP2/$\overline{RESET}$ pin to function as $\overline{RESET}$. When clear, the pin functions as open drain output only. This pin defaults to its $\overline{RESET}$ function following an MCU POR or LVD. When RSTPE is set, an internal pull-up device is enabled on $\overline{RESET}$. 0 PTD1/CMPP2/$\overline{RESET}$ pin functions as PTD1. 1 PTD1/CMPP2/$\overline{RESET}$ pin functions as $\overline{RESET}$.

5.7.5   System Options 2 (SOPT2) Register

This high page register contains bits to configure microcontroller specific features on the MC9S08JE128 series microcontrollers.

	7	6	5	4	3	2	1	0
R	COPCLKS[1]	COPW[1]	0	CLKOUT_EN	0	0	0	ACIC
W								
Reset:	0	0	0	0	0	0	0	0
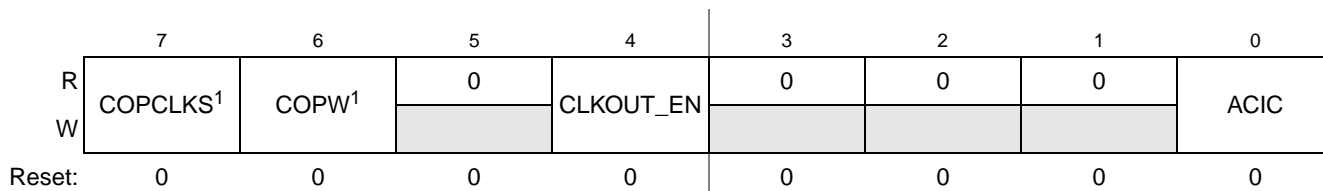
[1] This bit can be written to only one time after reset. Additional writes are ignored.

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

Table 5-6. 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 LPOCLK is source to COP. 1 Bus clock is source to COP.
6 COPW	**COP Window Mode** — This write-once bit specifies whether the COP operates in Normal or Window mode. In Window mode, the 0x55–0xAA write sequence to the SRS register must occur within the last 25% of the selected period; any write to the SRS register during the first 75% of the selected period resets the microcontroller. 0 Normal mode 1 Window mode

Table 5-6. SOPT2 Register Field Descriptions (Continued)

Field	Description
4 CLKOUT_EN	**Clock Output Enable** — This bit is used to mux out the BUSCLK clock to PTC7. 0 BUSCLK is not available on the external pin PTC7. 1 BUSCLK is available on the external pin PTC7.
0 ACIC	**Analog Comparator to Input Capture Enable**— This bit connects the output of the ACMP to TPM1 input channel 0. See Chapter 9, "Programmable Analog Comparator (S08PRACMPV1)," and Chapter 21, "Timer/Pulse-Width Modulator (S08TPMV3)," for more details on this feature. 0 ACMP output not connected to TPM1 input channel 0. 1 ACMP output connected to TPM1 input channel 0.

Table 5-7. COP Configuration Options

Control Bits		Clock Source	COP Window[1] Opens (SOPT2[COPW] = 1)	COP Overflow Count
SOPT2[COPCLKS]	SOPT1[COPT]			
N/A	00	N/A	N/A	COP is disabled
0	01	1 kHz LPOCLK	N/A	2^5 cycles (32 ms[2])
0	10	1 kHz LPOCLK	N/A	2^8 cycles (256 ms[2])
0	11	1 kHz LPOCLK	N/A	2^{10} cycles (1,024 ms[2])
1	01	BUSCLK	6,144 cycles	2^{13} cycles[1]
1	10	BUSCLK	49,152 cycles	2^{16} cycles[1]
1	11	BUSCLK	196,608 cycles	2^{18} cycles[1]

[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 (SOPT2[COPW] = 1).

[2] Values shown in milliseconds based on t_{LPO} = 1 ms.

5.7.6   SIM Clock Set and Select Register (SIMCO)

This register controls operation of the CLKOUT pin. The CS field in this register controls the output mux function for the CLKOUT pin. The various clock sources must be enabled/disabled via the appropriate controls elsewhere in the device.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0		CS	
W	—[1]							
Reset:	0	0	0	0	0	0	0	0

[1] Bit 7 of SIMCO register is reserved for internal Freescale testing. Writing a "1" to this bit impacts TPM1 external clock selection.

Table 5-8. SIMCO Bit Field Descriptions

Field	Description
7-3	**RESERVED**
2-0 CS	**CLKOUT Select**— 000 CLKOUT = 0 001 CLKOUT = Crystal oscillator 1 010 CLKOUT = Crystal oscillator 2 011 CLKOUT = internal RC oscillator 100 CLKOUT = BUSCLK 101 CLKOUT = CPUCLK 110 CLKOUT = LPOCLK 111 CLKOUT = ADC asynchronous clock

5.7.7 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
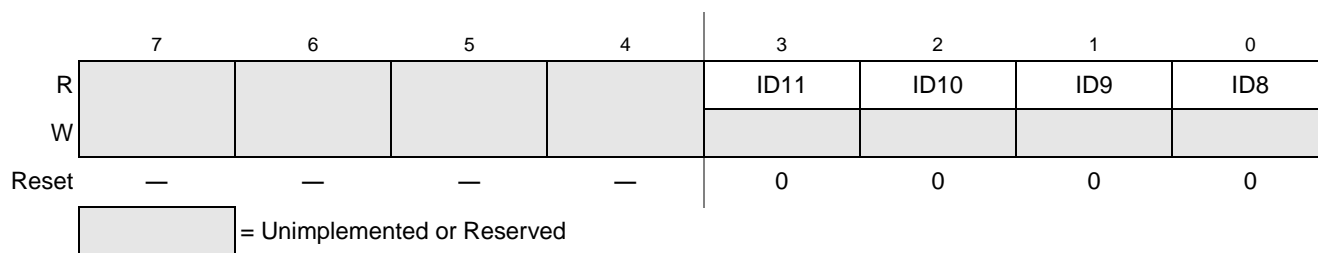
Figure 5-7. System Device Identification Register — High (SDIDH)

Table 5-9. 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 MC9S08JE128 series is hard coded to the value 0x02E. See also ID bits in Table 5-10.

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

= Unimplemented or Reserved
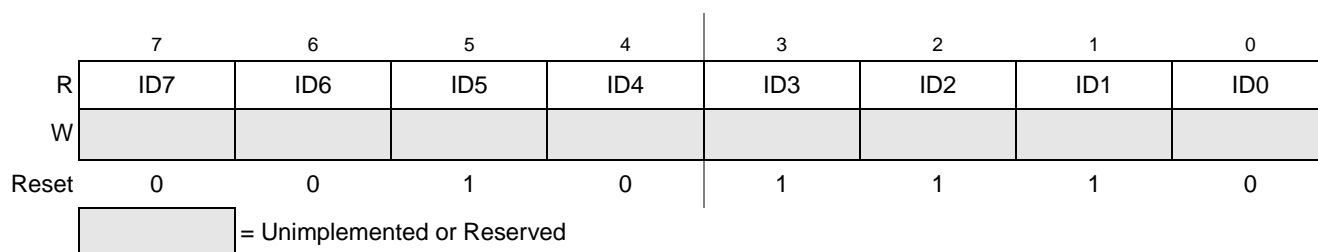
Figure 5-8. System Device Identification Register — Low (SDIDL)

Table 5-10. 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 MC9S08JE128 series is hard coded to the value 0x02E. See also ID bits in Table 5-9.

5.7.8 System Clock Gating Control 1 Register (SCGC1)

This high page register contains control bits to enable or disable the bus clock to the CMT, TPMx, ADC, DAC, IIC, and SCIx modules. Gating off the clocks to unused peripherals is used to reduce the microcontroller's run and wait currents. See the section on clock gating in the peripheral's chapter introduction for more information.

	7	6	5	4	3	2	1	0
R W	CMT	TPM2	TPM1	ADC	DAC	IIC	SCI2	SCI1
Reset:	1	1	1	1	1	1	1	1
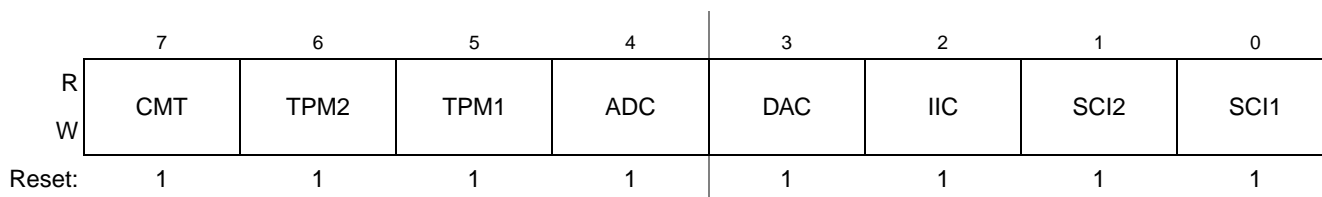
Figure 5-9. System Clock Gating Control 1 Register (SCGC1)

Table 5-11. SCGC1 Register Field Descriptions

Field	Description
7 CMT	**CMT Clock Gate Control** — This bit controls the clock gate to the CMT module. 0 Bus clock to the CMT module is disabled. 1 Bus clock to the CMT module is enabled.
6 TPM2	**TPM2 Clock Gate Control** — This bit controls the clock gate to the TPM2 module. 0 Bus clock to the TPM2 module is disabled. 1 Bus clock to the TPM2 module is enabled.
5 TPM1	**TPM1 Clock Gate Control** — This bit controls the clock gate to the TPM1 module. 0 Bus clock to the TPM1 module is disabled. 1 Bus clock to the TPM1 module is enabled.
4 ADC	**ADC Clock Gate Control** — This bit controls the clock gate to the ADC module. 0 Bus clock to the ADC module is disabled. 1 Bus clock to the ADC module is enabled.
3 DAC	**DAC Clock Gate Control** — This bit controls the clock gate to the DAC module. 0 Bus clock to the DAC module is disabled. 1 Bus clock to the DAC module is enabled.
2 IIC	**IIC Clock Gate Control** — This bit controls the clock gate to the IIC1 module. 0 Bus clock to the IIC module is disabled. 1 Bus clock to the IIC module is enabled.
1 SCI2	**SCI2 Clock Gate Control** — This bit controls the clock gate to the SCI2 module. 0 Bus clock to the SCI2 module is disabled. 1 Bus clock to the SCI2 module is enabled.
0 SCI1	**SCI1 Clock Gate Control** — This bit controls the clock gate to the SCI1 module. 0 Bus clock to the SCI1 module is disabled. 1 Bus clock to the SCI1 module is enabled.

5.7.9   System Clock Gating Control 2 Register (SCGC2)

This high page register contains control bits to enable or disable the bus clock to the USB, PDB, IRQ, KBIx, ACMP, TOD, and SPIx modules. Gating off the clocks to unused peripherals is used to reduce the microcontroller's run and wait currents. See the section on clock gating in the peripheral's chapter introduction for more information.

	7	6	5	4	3	2	1	0
R	USB	PDB	IRQ	KBI	PRACMP	TOD	SPI2	SPI1
W								
Reset:	1	1	1	1	1	1	1	1

Figure 5-10. System Clock Gating Control 2 Register (SCGC2)

Table 5-12. SCGC2 Register Field Descriptions

Field	Description
7 USB	**USB Clock Gate Control** — This bit controls the bus clock gate to the USB module. 0 Bus clock to the USB module is disabled. 1 Bus clock to the USB module is enabled.
6 PDB	**PDB Clock Gate Control** — This bit controls the bus clock gate to the PDB registers. 0 Bus clock to PDB registers is disabled. 1 Bus clock to PDB registers is enabled.
5 IRQ	**IRQ Clock Gate Control** — This bit controls the bus clock gate to the IRQ module. 0 Bus clock to the IRQ module is disabled. 1 Bus clock to the IRQ module is enabled.
4 KBI	**KBI Clock Gate Control** — This bit controls the clock gate to both of the KBI modules. 0 Bus clock to the KBI modules is disabled. 1 Bus clock to the KBI modules is enabled.
3 PRACMP	**ACMP Clock Gate Control** — This bit controls the clock gate to both of the PRACMP modules. 0 Bus clock to the PRACMP modules is disabled. 1 Bus clock to the PRACMP modules is enabled.
2 TOD	**TOD Clock Gate Control** — This bit controls the bus clock gate to the TOD module. Only the bus clock is gated; the clock source of TOD is not controlled by this bit. 0 Bus clock to the TOD module is disabled. 1 Bus clock to the TOD module is enabled.
1 SPI2	**SPI2 Clock Gate Control** — This bit controls the clock gate to the SPI2 module. 0 Bus clock to the SPI2 module is disabled. 1 Bus clock to the SPI2 module is enabled.
0 SPI1	**SPI1 Clock Gate Control** — This bit controls the clock gate to the SPI1 module. 0 Bus clock to the SPI1 module is disabled. 1 Bus clock to the SPI1 module is enabled.

5.7.10 System Clock Gating Control 3 Register (SCGC3)

This high page register contains control bits to enable or disable the bus clock to the CRC, FLSx, and VREF modules. Gating off the clocks to unused peripherals is used to reduce the microcontroller's run and wait currents. See the section on clock gating in the peripheral's chapter introduction for more information.

	7	6	5	4	3	2	1	0
R	VREF	CRC	—	FLS1	—	—	—	—
W								
Reset:	1	1	1	1	1	1	1	1
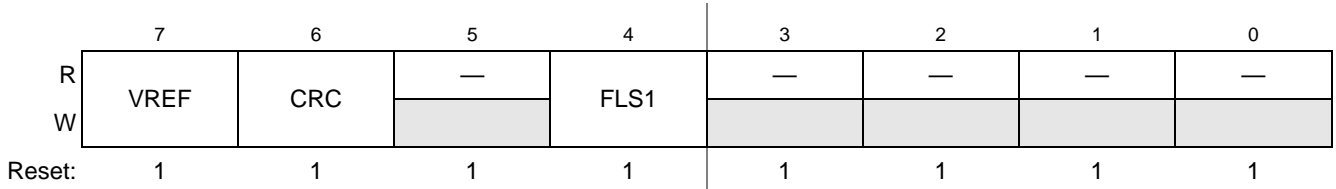
Figure 5-11. System Clock Gating Control 3 Register (SCGC3)

Table 5-13. SCGC3 Register Field Descriptions

Field	Description
7 VREF	**VREF Clock Gate Control** — This bit controls the clock gate to the VREF module. 0 Bus clock to the VREF module is disabled. 1 Bus clock to the VREF module is enabled.
6 CRC	**CRC Clock Gate Control** — This bit controls the clock gate to the CRC module. 0 Bus clock to the CRC module is disabled. 1 Bus clock to the CRC module is enabled.
4 FLS1	**FLS1 Clock Gate Control** — This bit controls the clock gate to the FLS1 module. 0 Bus clock to the FLS1 module is disabled. 1 Bus clock to the FLS1 module is enabled.

5.7.11 System Options 3 Register (SOPT3)

This register contains a bit that controls the pad drive strength for the CMT module.

	7	6	5	4	3	2	1	0
R	SCI2PS	SCI1PS	IICPS	0	0	0	SCI1_PAD	CMT_PAD
W								
Reset:	0	0	0	0	0	0	0	0
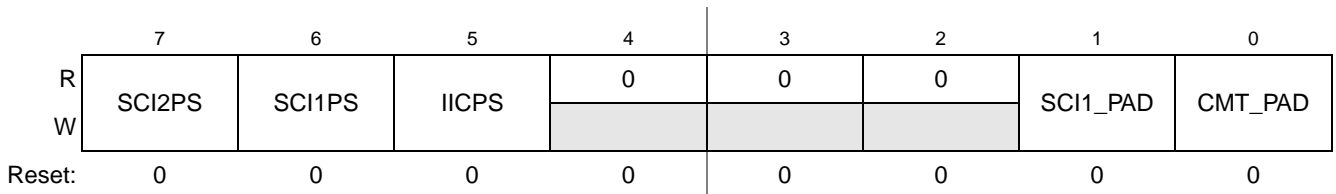
Figure 5-12. System Options 3 Register (SOPT3)

Table 5-14. SOPT3 Field Descriptions

Field	Description
7 SCI2PS	**SCI2PS** — Configures TX2 and RX2 to be on PTE5, PTE6 or PTF2, PTF1, by default this is PTE5, PTE6 0 PTE5, PTE6 1 PTF2, PTF1
6 SCI1PS	**SCI1PS** — Configures TX1 and RX1 to be on PTA1, PTA2 or PTD6, PTD7, by default this is PTA1, PTA2 0 PTA1, PTA2 1 PTD6, PTD7

Table 5-14. SOPT3 Field Descriptions (Continued)

Field	Description
5 IICPS	**IICPS** — Configures SDA and SCL to be on PTD4, PTD5 or PTF4, PTF3, by default this is PTD4, PTD5 0 PTD4, PTD5 1 PTF4, PTF3
1 SCI1_PAD	**SCI_PAD pad drive strength** — This bit controls the SCI1 TX pad drive strength by connecting two pads together. 0 single-pad drive strength 1 dual-pads bounding drive strength
0 CMT_PAD	**CMT pad drive strength** — This bit controls the CMT pad drive strength by connecting two pads together. 0 single-pad drive strength 1 dual-pads bounding drive strength

5.7.12 System Options 4 Register (SOPT4)

This register contains bits that control the drive strength, slew rate of IRO pin.

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	IROSRE	IRODSE
W								
Reset:	0	0	0	0	0	0	0	0
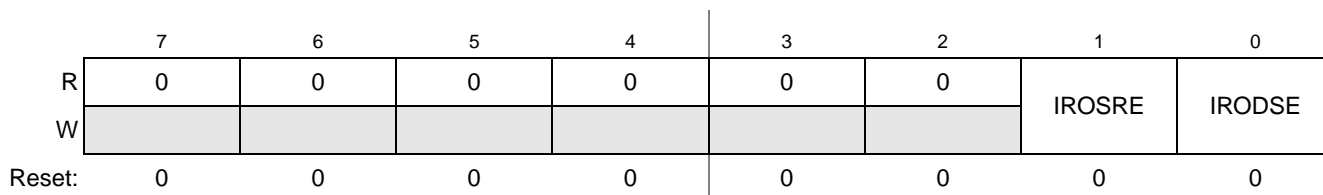
Figure 5-13. System Options 4 Register (SOPT4)

Table 5-15. SOPT4 Field Descriptions

Field	Description
1 IROSRE	**Output Slew Rate Control Enable for IRO Pin**— This bit determines whether output slew rate control is enabled for the output only IRO pin. 0 Output slew rate control disabled for IRO pin 1 Output slew rate control enabled for IRO pin
0 IRODSE	**Drive Strength Control Enable for IRO Pin**— This bit determines whether high drive strength or low drive strength is enabled for the IRO pin. 0 Low Drive strength selected for IRO pin 1 High drive strength selected for IRO pin

5.7.13 System Options 5 Register (SOPT5)

CAUTION

This location is reserved for Freescale internal testing. Do not write any value to this location. Writing a value to this location can affect on-chp LVD performance.

5.7.14 SIM Internal Peripheral Select Register (SIMIPS)

The fields in this register control source used for the SCI1 RX pin, as well as modulation choice for the SCI1 TX pin. The various clock sources used for modulation purposes must be enabled/disabled via the appropriate controls elsewhere in the device. See Figure 2-4 for an illustration of these controls in action.

	7	6	5	4	3	2	1	0
R	ADCTRS	RX1N	0	0	MTBASE1		0	MODTX1
W								
POR/LVD:	0	0	0	0	0	0	0	0
Any Other Reset:	u[1]	u[1]	0	0	u[1]	u[1]	0	u[1]

Figure 5-14. SIM Internal Peripheral Select Register 1 (SIMIPS)

[1] u = Unaffected

Table 5-16. SIMIPS Register Bit Fields

Field	Description
7 ADCTRS	**ADC Trigger Select** — 0 ADC ALT trigger is from PDB. 1 ADC ALT trigger is from TOD.
6 RX1N	**SCI1 RX Input Pin Select** — 0 RX1 is fed from the digital input pin (assuming the RX1 is enabled on that pin via the MC registers) 1 RX1 is fed from the output of comparator 1
3-2 MTBASE1	**SCI1 TX Modulation Time Base Select** — 00 TPM 1CH0 01 TPM 1CH1 10 TPM2CH0 11 TPM2CH1
0 MODTX1	**Modulate TX1** — 0 Do not modulate the output of SCI1 1 Modulate the output of SCI1 with the timebase selected via the MTBASE1 field

5.7.15 Flash Protection Defeat Register (FPROTD)

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

[ ] = Unimplemented or Reserved
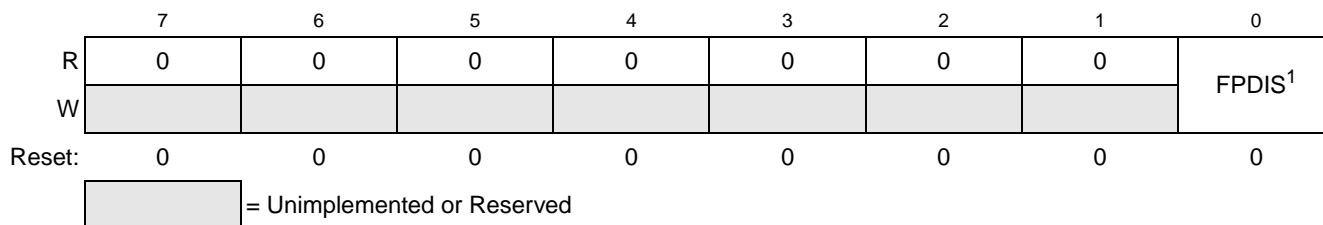
[1] Back to back writes of 0x55 and 0xAA, set FPDIS. Any other write clears FPDIS.

Figure 5-15. Flash Protection Defeat Register (FPROTD)

Table 5-17. FPROTD Register Field Descriptions

Field	Description
0 **FPDIS**	This bit is set when back-to-back writes of 0x55 and 0xAA occur. It is cleared, when back-to-back writes of non-(0x55,0xAA) occur. 0 Indicates that writes to the Flash block protection registers are not allowed. 1 Indicates that writes to the Flash block protection registers are allowed.

5.7.16 Signature Register (Signature)

The Signature Semaphore spans the entire 8-bit register.

	7	6	5	4	3	2	1	0
R								
W				Signature Semaphore				
POR:	0	0	0	0	0	0	0	0
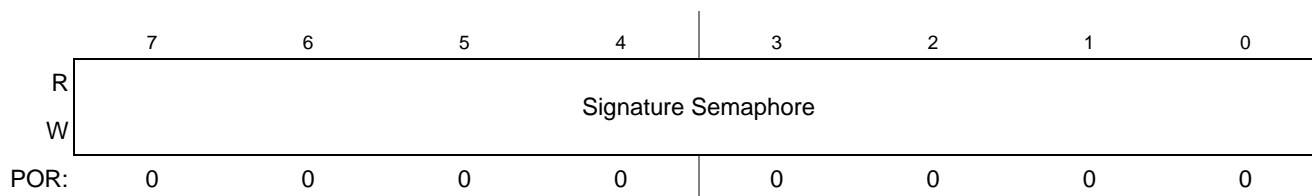
Figure 5-16. Signature Register (Signature)

Table 5-18. Signature Semaphore Register Field Descriptions

Field	Description
7:0 Signature [7:0]	**Signature Register Bits** — The Signature Semaphore is used as the semaphore to jump to bootloader mode (or not) after a regular Reset. This byte only can be cleared by a power-on reset (POR), content retains after regular reset.

5.7.17 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 should 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	LVDF[1]	0	LVDIE	LVDRE[2]	LVDSE	LVDE[2]	0	BGBE
W		LVDACK						
Reset:	0	0	0	1	1	1	0	0

[1] LVWF is 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 5-17. System Power Management Status and Control 1 Register (SPMSC1)

Table 5-19. SPMSC1 Register Field Descriptions

Field	Description
7 LVDF	**Low-Voltage Detection Flag** — The LVDF bit indicates the low-voltage detection status. 0 Low-voltage is not detected. 1 Low-voltage is detected or was detected.
6 LVDACK	**Low-Voltage Detection Acknowledge** — If LVDF = 1, a low-voltage condition has occurred. To acknowledge this low-voltage condition, write 1 to LVDACK, which automatically clears LVDF to 0 if the low-voltage is no longer detected.
5 LVDIE	**Low-Voltage Detection Interrupt Enable** — This bit enables hardware interrupt requests for LVDF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVDF = 1.
4 LVDRE	**Low-Voltage Detect Reset Enable** — This write-once bit enables LVDF events to generate a hardware reset (provided LVDE = 1). 0 LVDF does 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 the ADC module on one of its internal channels. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

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

This high page register contains status and control bits to configure the low-power run and wait modes as well as configure the stop mode behavior of the microcontroller. See Section 3.6.1, "Low-power Wait Mode (LPWait)," for more information.

SPMSC2 is not reset when exiting from STOP2.

	7	6	5	4	3	2	1	0
R	LPR	LPRS	LPWUI	0	PPDF	0	PPDE[1]	PPDC
W						PPDACK		
Reset:	0	0	0	0	—	0	0	0
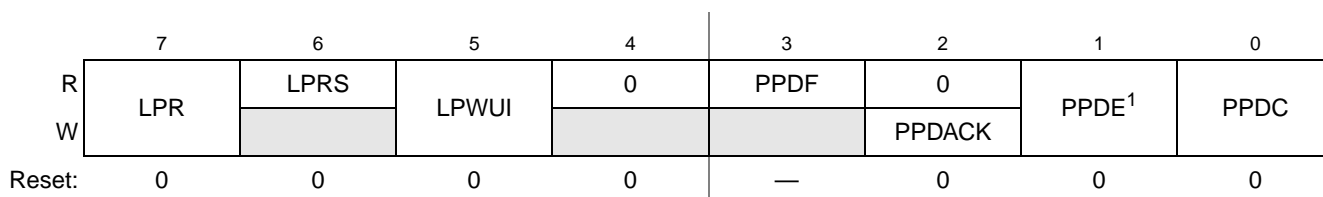
Figure 5-18. System Power Management Status and Control 2 Register (SPMSC2)

[1] PPDE is a write-once bit that can be used to permanently disable the PPDC bit.

Table 5-20. SPMSC2 Bit Field Descriptions

Field	Description
7 LPR	**Low-power Regulator Control.** The LPR bit controls entry into the low-power run and low-power wait modes in which the voltage regulator is put into standby. This bit cannot be set if PPDC=1. If PPDC and LPR are set in a single write instruction, only PPDC is actually set. LPR is cleared when an interrupt occurs in low-power mode and the LPWUI bit is 1. 0 Low-power run and low-power wait modes are disabled. 1 Low-power run and low-power wait modes are requested.
6 LPRS	**Low-power Regulator Status**. This read-only status bit indicates that the voltage regulator has entered into standby for the low-power run or wait mode. 0 The voltage regulator is not currently in standby. 1 The voltage regulator is currently in standby.
5 LPWUI	**Low-power Wake-Up on Interrupt.** This bit controls whether or not the voltage regulator exits standby when any active MCU interrupt occurs. 0 The voltage regulator remains in standby on an interrupt. 1 The voltage regulator exits standby on an interrupt. LPR is cleared.
4	**RESERVED**
3 PPDF	**Partial Power-Down Flag** — This read-only status bit indicates that the microcontroller has recovered from Stop2 mode. 0 Microcontroller has not recovered from Stop2 mode. 1 Microcontroller recovered from Stop2 mode.
2 PPDACK	**Partial Power-Down Acknowledge** — Writing a 1 to PPDACK clears the PPDF bit.
1 PPDE	Partial Power-Down Enable. The write-once PPDE bit can be used to lockout the partial power-down feature. This is a write-once bit. 0 Partial power-down is not enabled. 1 Partial power-down is enabled and controlled via the PPDC bit.
0 PPDC	**Partial Power-Down Control** — The PPDC bit controls which power-down mode is selected. This bit cannot be set if LPR = 1. If PPDC and LPR are set in a single write instruction, only PPDC will actually be set. PPDE must be set in order for PPDC to be set. 0 Stop3 low-power mode enabled. 1 Stop2 partial power-down mode enabled.

1 See the MC9S08JE128 Data Sheet Electrical Characteristics appendix for minimum and maximum values.

5.7.19 System Power Management Status and Control 3 Register (SPMSC3)

This register reports the status of the low voltage warning function and is used to select the low voltage detect trip voltage. SPMSC3 is not reset when exiting from stop2.

	7	6	5	4	3	2	1	0
R	LVWF	0	LVDV	LVWV	LVWIE	0	0	0
W		LVWACK						
POR:	0[1]	0	0	0	0	0	0	0
LVR:	0[1]	0	U	U	0	0	0	0
Any other reset:	0[1]	0	U	U	0	0	0	0

[1] LVWF is set when V_{Supply} transitions below the trip point or after reset and V_{Supply} is already below V_{LVW}.

Figure 5-19. System Power Management Status and Control 3 Register (SPMSC3)

Table 5-21. SPMSC3 Bit Field Descriptions

Field	Description
7 LVWF	Low-Voltage Warning Flag. The LVWF bit indicates the low voltage warning status. 0 Low voltage warning not present. 1 Low voltage warning is present or was present.
6 LVWACK	Low-Voltage Warning Acknowledge. Writing a 1 to LVWACK clears LVWF if a low voltage warning is not present.
5 LVDV	Low-Voltage Detect Voltage Select. The LVDV bit selects the LVD trip point voltage (V_{LVD}). 0 Low trip point selected ($V_{LVD} = V_{LVDL}$). 1 High trip point selected ($V_{LVD} = V_{LVDH}$).
4 LVWV	Low-Voltage Warning Voltage Select. The LVWV bit selects the LVW trip point voltage (V_{LVW}). 0 Low trip point selected ($V_{LVW} = V_{LVWL}$). 1 High trip point selected ($V_{LVW} = V_{LVWH}$).
3 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 is set.
2–0	Reserved, should be cleared.

Table 5-22. LVD and LVW Trip Point Typical Values[1]

LVDV:LVWV	LVW Trip Point	LVD Trip Point
00	$V_{LVWL} = 2.15$	$V_{LVDL} = 1.86$
01	$V_{LVWH} = 2.6$	
10 Not Recommended	$V_{LVWL} = 2.15$	$V_{LVDH} = 2.33$
11	$V_{LVWH} = 2.6$	

[1] Note: Data in this table may not be up-to-date. Please refer to the DC Characteristics table in the MC9S08JE128 Data Sheet for the latest values.

Chapter 6
Parallel Input/Output

6.1   Introduction

This chapter explains software controls related to parallel input/output (I/O). The MC9S08JE128 has seven I/O ports which include a total of 49 general-purpose I/O pins and one input only and two output only 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 pull-up 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.
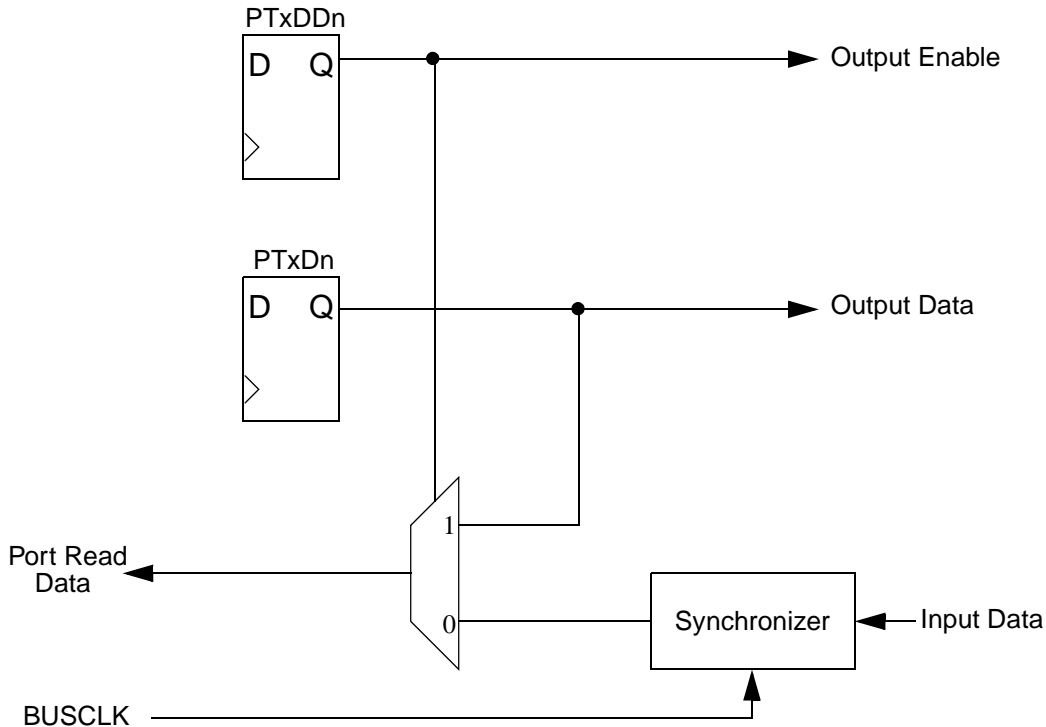
Figure 6-1. Parallel I/O Block Diagram

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 Pull-up Enable

An internal pull-up device can be enabled for each port pin by setting the corresponding bit in one of the pull-up enable registers (PTxPEn). The pull-up 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 pull-up enable register bit. The pull-up 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	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
Reset	0	0	0	0	0	0	0	0
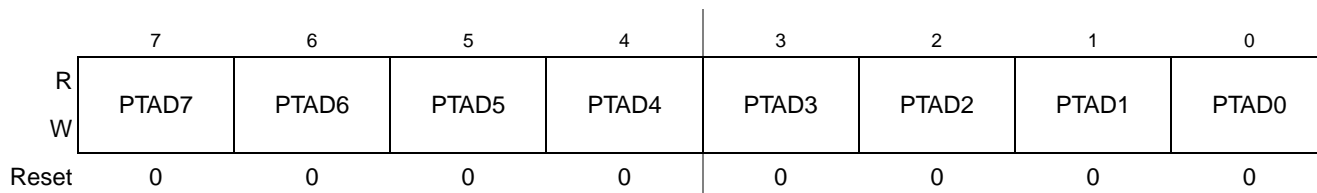
Figure 6-2. Port A Data Register (PTAD)

Table 6-1. PTAD Register Field Descriptions

Field	Description
7:0 PTAD[7:0]	**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	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
Reset	0	0	0	0	0	0	0	0
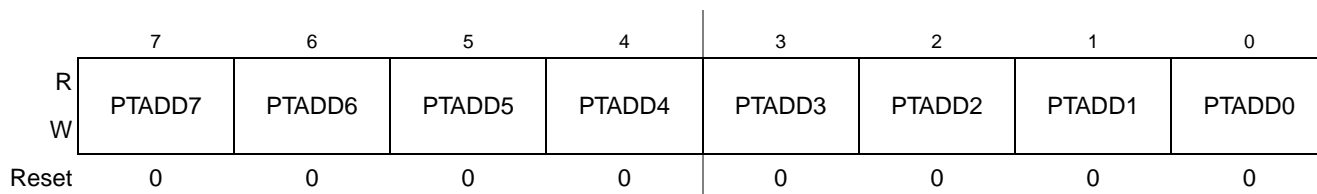
Figure 6-3. Data Direction for Port A Register (PTADD)

Table 6-2. PTADD Register Field Descriptions

Field	Description
7:0 PTADD[7:0]	**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	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
Reset	0	0	0	0	0	0	0	0
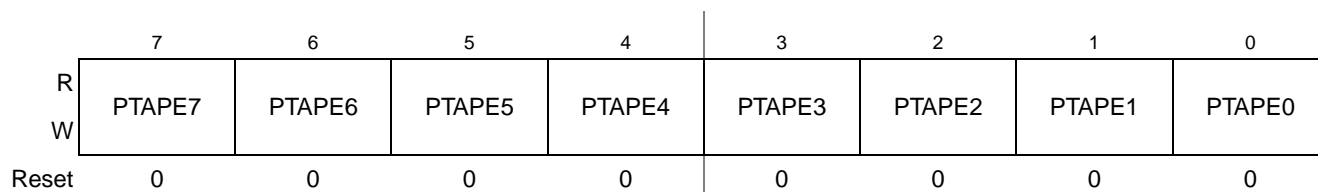
Figure 6-4. Internal Pull-up Enable for Port A (PTAPE)

Table 6-3. PTADD Register Field Descriptions

Field	Description
[7:0] PTAPE[7:0]	**Internal Pull-up Enable for Port A Bits** — Each of these control bits determines if the internal pull-up 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 pull-up devices are disabled. 0 Internal pull-up device disabled for port A bit n. 1 Internal pull-up device enabled for port A bit n.

	7	6	5	4	3	2	1	0
R W	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
Reset	0	0	0	0	0	0	0	0

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

Table 6-4. PTASE Register Field Descriptions

Field	Description
7:0 PTASE[7:0]	**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	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
Reset	0	0	0	0	0	0	0	0
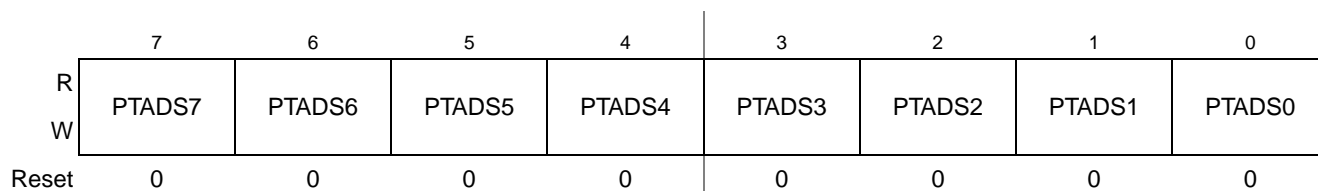
Figure 6-6. Output Drive Strength Selection for Port A (PTASE)

Table 6-5. PTASE Register Field Descriptions

Field	Description
7:0 PTADS[7:0]	**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.

6.5.2.1 Port A Input Filter Enable Register (PTAIFE)

The Port A pin incorporates an optional input low-pass filter. Set the associated PTAIFE bit during and after reset to enable the filter. To disable the filter, clear the associated PTAIFE bit through software control.

	7	6	5	4	3	2	1	0
R	PTAIFE7	PTAIFE6	PTAIFE5	PTAIFE4	PTAIFE3	PTAIFE2	PTAIFE1	PTAIFE0
W								
Reset:	1	1	1	1	1	1	1	1
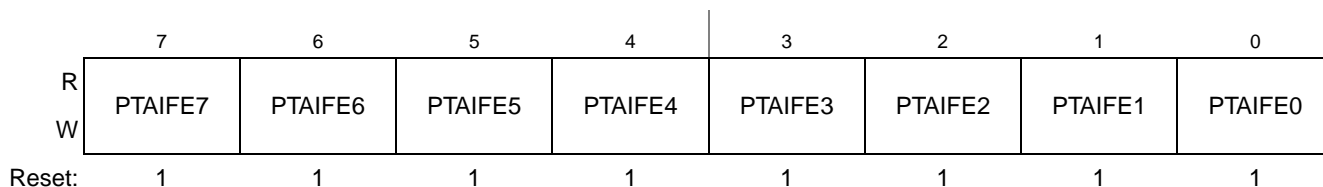
Figure 6-7. Input Filter Enable for Port A Register (PTAIFE)

Table 6-6. PTAIFE Register Field Descriptions

Field	Description
7–0 PTAIFEn	**Input Filter Enable for Port A Bits** — Input low-pass filter enable control bits for PTA pins. 0 Input filter disabled. 1 Input filter enabled.

6.5.3 Port B Registers

Port B is controlled by the following registers.

6.5.4 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	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1[1]	PTBD0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-8. Port B Data Register (PTBD)

[1] Reads of the PTBD1 bit always return the contents of PTBD1, regardless of the value stored in the PTBDD1 bit.

Table 6-7. 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	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1[1]	PTBDD0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-9. Data Direction for Port B (PTBDD)

[1] PTBDD1 has no effect on the output-only PTB1 pin.

Table 6-8. 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.

6.5.5   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	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1[1]	PTBPE0
W								
Reset	0	0	0	0	0	0	0	0
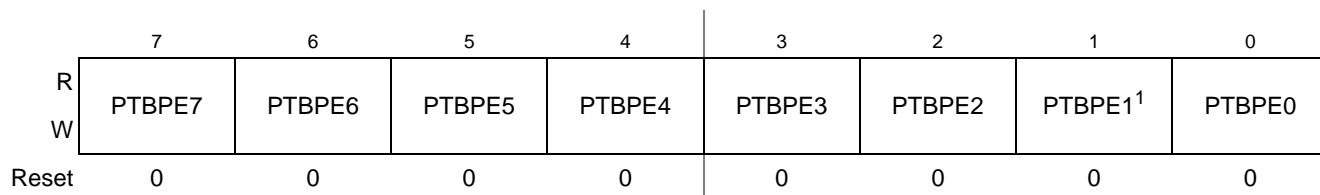
Figure 6-10. Internal Pull-up Enable for Port B (PTBPE)

[1] PTBPE1 has no effect on the output-only PTB1 pin.

Table 6-9. PTBPE Register Field Descriptions

Field	Description
7:0 PTBPE[7:0]	**Internal Pull-up Enable for Port B Bits** — Each of these control bits determines if the internal pull-up 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 pull-up devices are disabled. 0 Internal pull-up device disabled for port B bit n. 1 Internal pull-up device enabled for port B bit n.

	7	6	5	4	3	2	1	0
R	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
W								
Reset	0	0	0	0	0	0	0	0
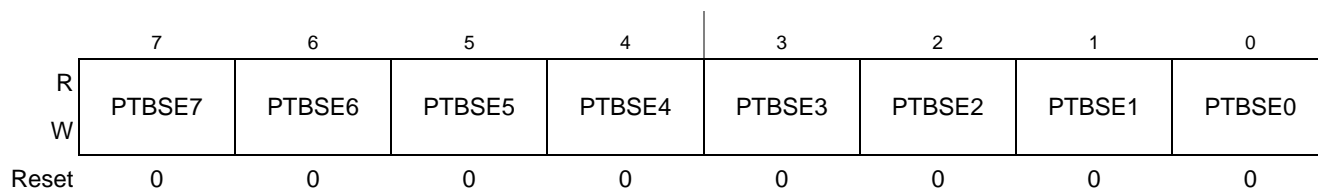
Figure 6-11. Output Slew Rate Control Enable (PTBSE)

Table 6-10. 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	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-12. Output Drive Strength Selection for Port B (PTBDS)

Table 6-11. 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.

6.5.5.1   Port B Input Filter Enable Register (PTBIFE)

The Port B pin incorporates an optional input low-pass filter. Set the associated PTBIFE bit during and after reset to enable the filter. To disable the filter, clear the associated PTBIFE bit through software control.

	7	6	5	4	3	2	1	0
R	PTBIFE7	PTBIFE6	PTBIFE5	PTBIFE4	PTBIFE3	PTBIFE2	PTBIFE1	PTBIFE0
W								
Reset:	1	1	1	1	1	1	1	1
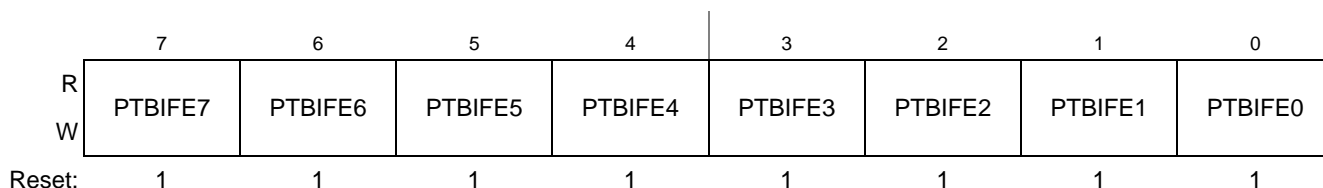
Figure 6-13. Input Filter Enable for Port B Register (PTBIFE)

Table 6-12. PTBIFE Register Field Descriptions

Field	Description
7–0 PTBIFE*n*	**Input Filter Enable for Port B Bits** — Input low-pass filter enable control bits for PTB pins. 0 Input filter disabled. 1 Input filter enabled.

6.5.6   Port C Registers

Port C is controlled by the following registers.

6.5.7   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	PTCD7	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
Reset	0	0	0	0	0	0	0	0
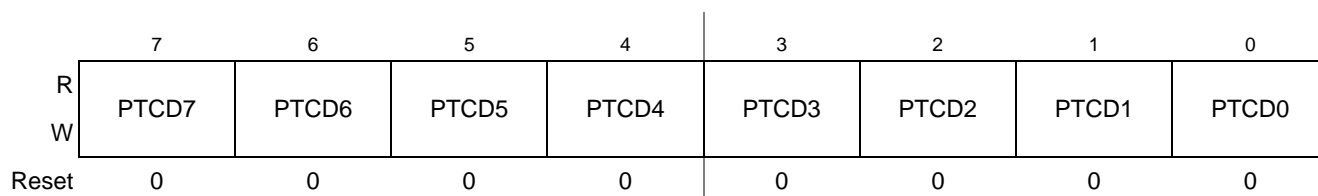
Figure 6-14. Port C Data Register (PTCD)

Table 6-13. PTCD Register Field Descriptions

Field	Description
7:0 PTCD[7:0]	**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	PTCDD7	PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
Reset	0	0	0	0	0	0	0	0
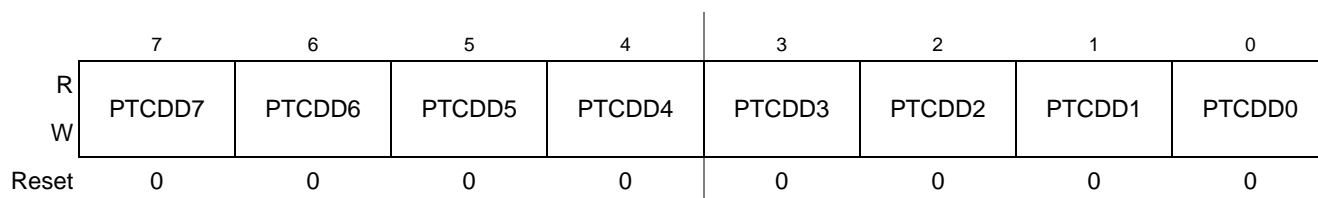
Figure 6-15. Data Direction for Port C (PTCDD)

Table 6-14. PTCDD Register Field Descriptions

Field	Description
7:0 PTCDD[7:0]	**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.

6.5.8 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	PTCPE7	PTCPE6	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-16. Internal Pull-up Enable for Port C (PTCPE)

Table 6-15. PTCPE Register Field Descriptions

Field	Description
7:0 PTCPE[7:0]	**Internal Pull-up Enable for Port C Bits** — Each of these control bits determines if the internal pull-up 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 pull-up devices are disabled. 0 Internal pull-up device disabled for port C bit n. 1 Internal pull-up device enabled for port C bit n.

	7	6	5	4	3	2	1	0
R	PTCSE7	PTCSE6	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-17. Output Slew Rate Control Enable for Port C (PTCSE)

Table 6-16. PTCSE Register Field Descriptions

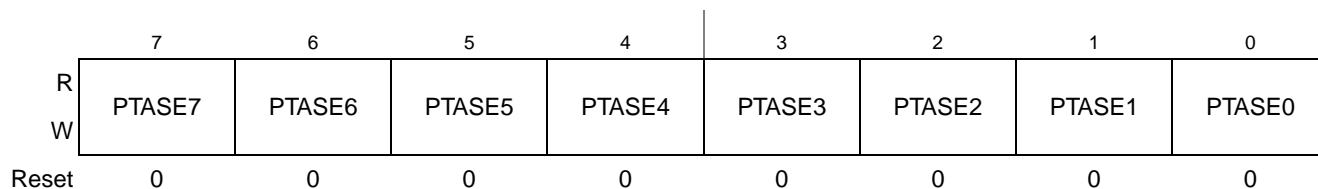| Field	Description
7:0 PTCSE[7:0]	**Output Slew Rate Control Enable for Port C Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTC pin. For port C pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled 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	PTCDS7	PTCDS6	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-18. Output Drive Strength Selection for Port C (PTCDS)

Table 6-17. PTCDS Register Field Descriptions

Field	Description
7:0 PTCDS[7:0]	**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.

6.5.8.1   Port C Input Filter Enable Register (PTCIFE)

The Port C pin incorporates an optional input low-pass filter. Set the associated PTCIFE bit during and after reset to enable the filter. To disable the filter, clear the associated PTCIFE bit through software control.

	7	6	5	4	3	2	1	0
R	PTCIFE7	PTCIFE6	PTCIFE5	PTCIFE4	PTCIFE3	PTCIFE2	PTCIFE1	PTCIFE0
W								
Reset:	1	1	1	1	1	1	1	1

Figure 6-19. Input Filter Enable for Port C Register (PTCIFE)

Table 6-18. PTCIFE Register Field Descriptions

Field	Description
7–0 PTCIFEn	**Input Filter Enable for Port C Bits** — Input low-pass filter enable control bits for PTC pins. 0 Input filter disabled. 1 Input filter enabled.

6.5.9   Port D Registers

Port D is controlled by the following registers.

PTD0 is an output only pin, so the control bits for input functions have no effect on this pin.

6.5.10   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	PTDD7	PTDD6	PTDD5	PTDD4	PTDD3	PTDD2	PTDD1	PTDD0[1]
W								
Reset	0	0	0	0	0	0	0	0
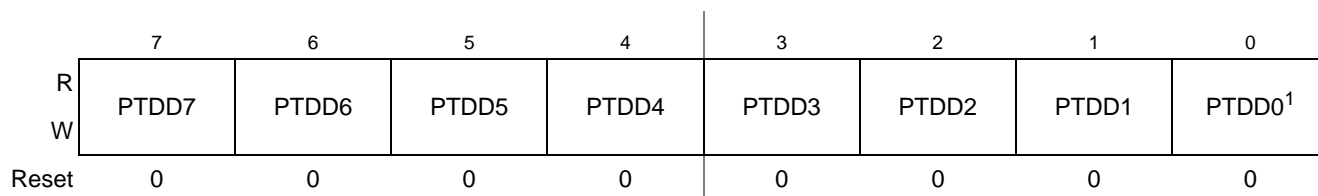
Figure 6-20. Port D Data Register (PTDD)

[1] Reads of bit PTDD0 always return the contents of PTDD0 regardless of the value stored in the PTDDD0 bit.

Table 6-19. PTDD Register Field Descriptions

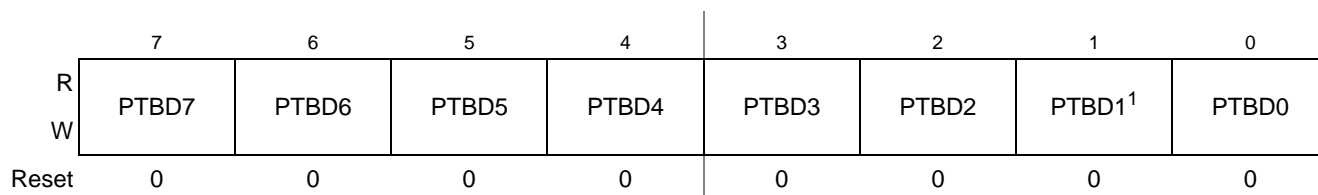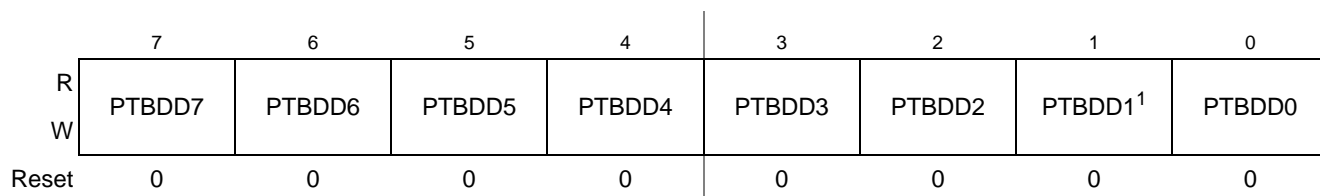| Field	Description
7:1 PTDD[7:1]	**Port D Data Register Bits** — For port D pins that are inputs, reads return the logic level on the pin. For port D pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port 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	PTDDD7	PTDDD6	PTDDD5	PTDDD4	PTDDD3	PTDDD2	PTDDD1	PTDDD0[1]
W								
Reset	0	0	0	0	0	0	0	1

Figure 6-21. Data Direction for Port D (PTDDD)

[1] PTDDD0 has no effect on the output-only PTD0 pin.

Table 6-20. 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.

6.5.11   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	PTDPE7	PTDPE6	PTDPE5	PTDPE4	PTDPE3	PTDPE2	PTDPE1	PTDPE0[1]
W								
Reset	0	0	0	0	0	0	0	0
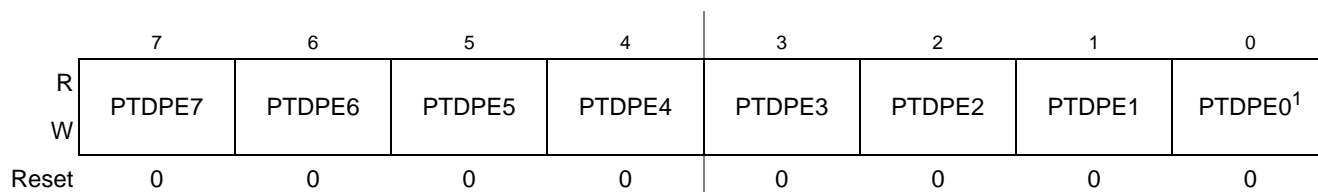
Figure 6-22. Internal Pull-up Enable for Port D (PTDPE)

[1] PTDPE0 has no effect on the output-only PTD0 pin.

Table 6-21. PTDPE Register Field Descriptions

Field	Description
7:0 PTDPE[7:0]	**Internal Pull-up Enable for Port D Bits** — Each of these control bits determines if the internal pull-up 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 pull-up devices are disabled. 0 Internal pull-up device disabled for port D bit n. 1 Internal pull-up device enabled for port D bit n.

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

Figure 6-23. Output Slew Rate Control Enable for Port D (PTDSE)

Table 6-22. 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	PTDDS7	PTDDS6	PTDDS5	PTDDS4	PTDDS3	PTDDS2	PTDDS1	PTDDS0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-24. Output Drive Strength Selection for Port D (PTDDS)

Table 6-23. PTDDS Register Field Descriptions

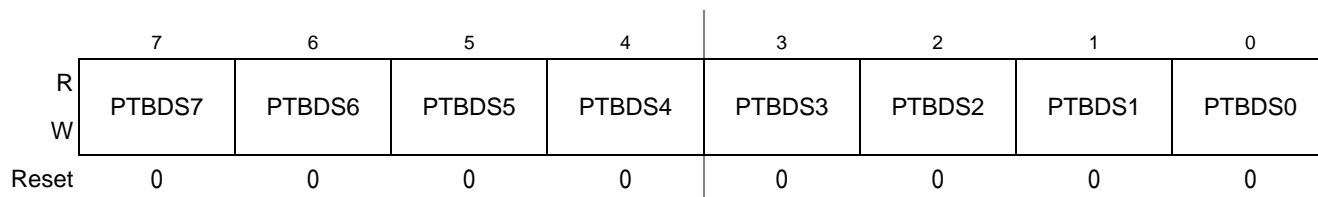| Field	Description
7:0 PTDDS[7:0]	**Output Drive Strength Selection for Port D Bits** — Each of these control bits selects between low and high output drive for the associated PTD pin. 0 Low output drive enabled for port D bit n. 1 High output drive enabled for port D bit n.

6.5.11.1   Port D Input Filter Enable Register (PTDIFE)

The Port D pin incorporates an optional input low-pass filter. Set the associated PTDIFE bit during and after reset to enable the filter. To disable the filter, clear the associated PTDIFE bit through software control.

	7	6	5	4	3	2	1	0
R	PTDIFE7	PTDIFE6	PTDIFE5	PTDIFE4	PTDIFE3	PTDIFE2	PTDIFE1	PTDIFE0
W								
Reset:	1	1	1	1	1	1	1	1

Figure 6-25. Input Filter Enable for Port D Register (PTDIFE)

Table 6-24. PTDIFE Register Field Descriptions

Field	Description
7–0 PTDIFEn	**Input Filter Enable for Port D Bits** — Input low-pass filter enable control bits for PTD pins. 0 Input filter disabled. 1 Input filter enabled.

6.5.12   Port E Registers

Port E is controlled by the following registers.

PTE4 is an input only pin. The control bits for input functions do not have an effect on this pin.

6.5.13   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	PTED7	PTED6	PTED5	PTED4[1]	PTED3	PTED2	PTED1	PTED0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-26. Port E Data Register (PTED)

[1] Reads of bit PTED4 always return the pin value.

Table 6-25. 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	PTEDD7	PTEDD6	PTEDD5	0	PTEDD3	PTEDD2	PTEDD1	PTEDD0
W								
Reset	0	0	0	0	0	0	0	0
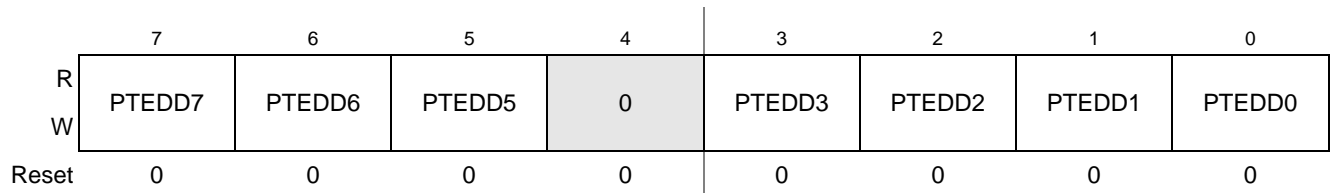
Figure 6-27. Data Direction for Port E (PTEDD)

Table 6-26. 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.

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

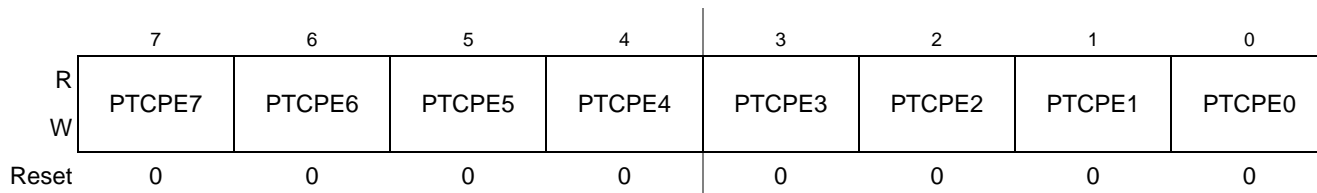In addition to the I/O control, port E pins are controlled by the registers listed below.

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

Figure 6-28. Internal Pull-up Enable for Port E (PTEPE)

Table 6-27. PTEPE Register Field Descriptions

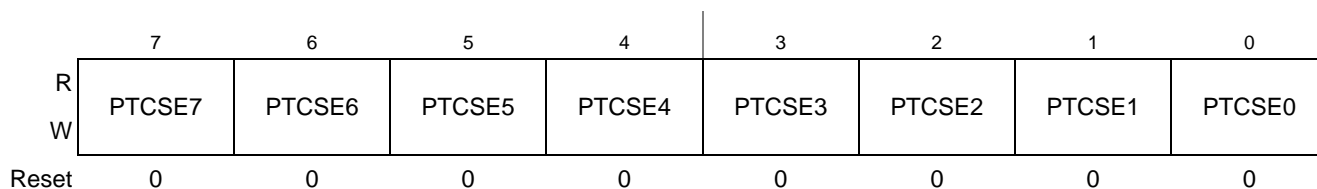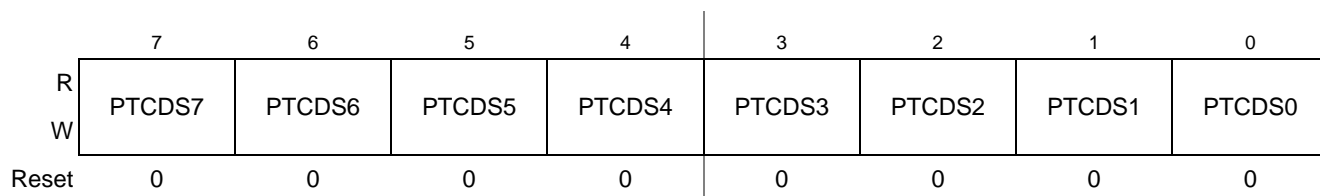| Field	Description
7:0 PTEPE[7:0]	**Internal Pull-up Enable for Port E Bits**— Each of these control bits determines if the internal pull-up device is enabled for the associated PTE pin. For port E pins that are configured as outputs, these bits have no effect and the internal pull-up devices are disabled.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

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

Figure 6-29. Output Slew Rate Control Enable for Port E (PTESE)

[1] PTESE4 has no effect on the PTE4 pin.

Table 6-28. PTESE Register Field Descriptions

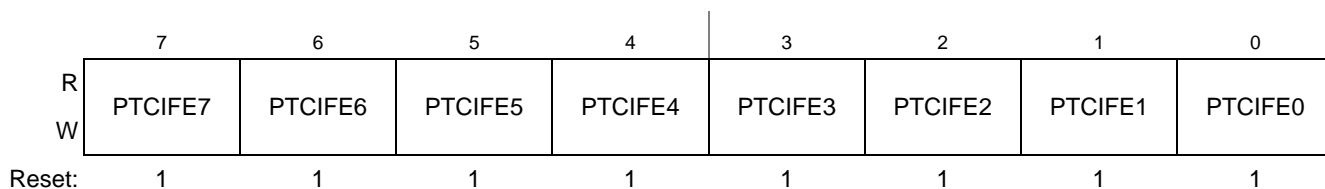| Field	Description
7:0 PTESE[7:0]	**Output Slew Rate Control Enable for Port E Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTE pin. For port E pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled 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	PTEDS7	PTEDS6	PTEDS5	PTEDS4[1]	PTEDS3	PTEDS2	PTEDS1	PTEDS0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-30. Output Drive Strength Selection for Port E (PTEDS)

[1] PTEDS4 has no effect on th PTE4 pin.

Table 6-29. 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.

6.5.14.1   Port E Input Filter Enable Register (PTEIFE)

The Port E pin incorporates an optional input low-pass filter. Set the associated PTEIFE bit during and after reset to enable the filter. To disable the filter, clear the associated PTEIFE bit through software control.

	7	6	5	4	3	2	1	0
R	PTEIFE7	PTEIFE6	PTEIFE5	PTEIFE4	PTEIFE3	PTEIFE2	PTEIFE1	PTEIFE0
W								
Reset:	1	1	1	1	1	1	1	1
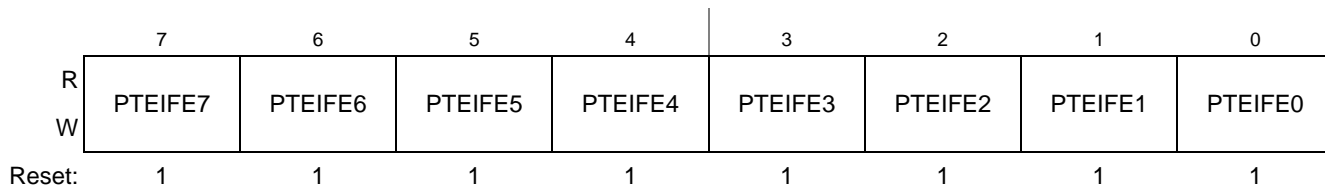
Figure 6-31. Input Filter Enable for Port E Register (PTEIFE)

Table 6-30. PTEIFE Register Field Descriptions

Field	Description
7–0 PTEIFEn	**Input Filter Enable for Port E Bits** — Input low-pass filter enable control bits for PTE pins. 0 Input filter disabled. 1 Input filter enabled.

6.5.15   Port F Registers

Port F is controlled by the following registers.

6.5.16   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	PTFD7	PTFD6	PTFD5	PTFD4	PTFD3	PTFD2	PTFD1	PTFD0
W								
Reset	0	0	0	0	0	0	0	0
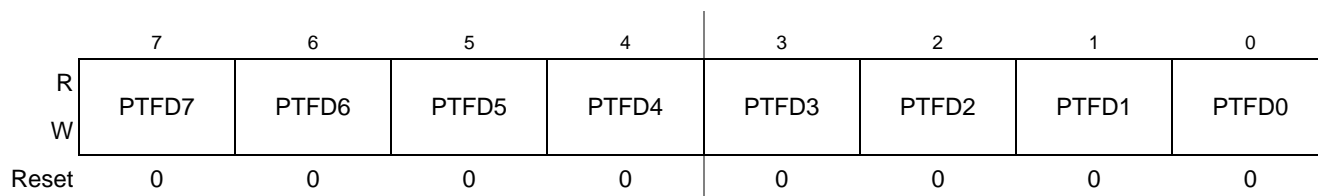
Figure 6-32. Port F Data Register (PTFD)

Table 6-31. PTFD Register Field Descriptions

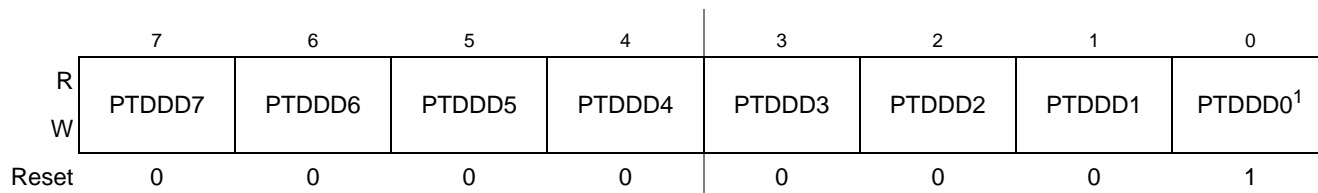| Field	Description
7:0 PTFD[7:0]	**Port F Data Register Bits** — For port F pins that are inputs, reads return the logic level on the pin. For port F pins, that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port 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	PTFDD7	PTFDD6	PTFDD5	PTFDD4	PTFDD3	PTFDD2	PTFDD1	PTFDD0
W								
Reset	0	0	0	0	0	0	0	0
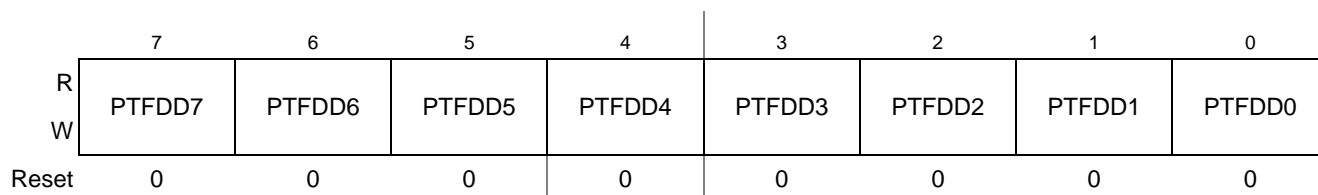
Figure 6-33. Data Direction for Port F (PTFDD)

Table 6-32. 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.

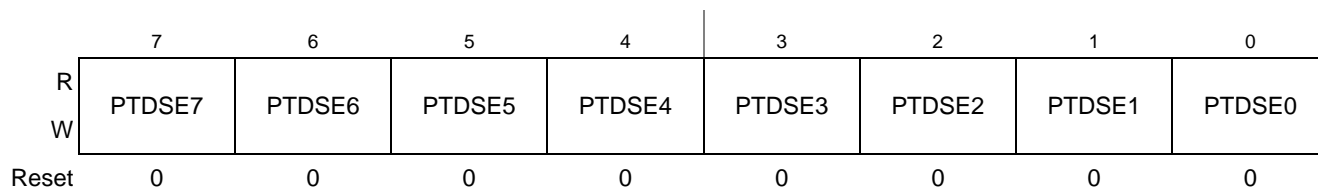## 6.5.17 Port F Pin Control Registers (PTFPE, PTFSE, PTFDS)

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	PTFPE7	PTFPE6	PTFPE5	PTFPE4	PTFPE3	PTFPE2	PTFPE1	PTFPE0
W								
Reset	0	0	0	0	0	0	0	0
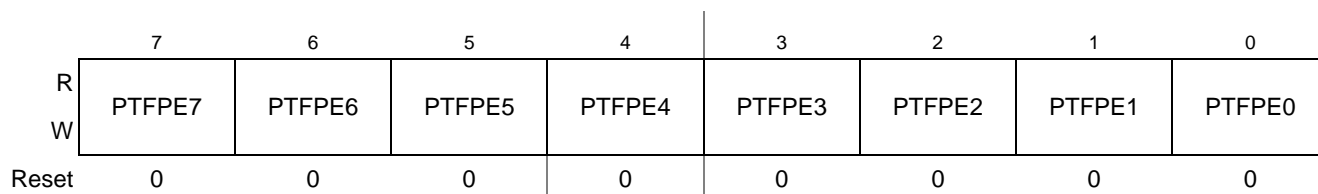
Figure 6-34. Internal Pull-up Enable for Port F (PTFPE)

Table 6-33. PTFPE Register Field Descriptions

Field	Description
7:0 PTFPE[7:0]	**Internal Pull-up Enable for Port F Bits** — Each of these control bits determines if the internal pull-up 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 pull-up devices are disabled. 0 Internal pull-up device disabled for port F bit n. 1 Internal pull-up device enabled for port F bit n.

	7	6	5	4	3	2	1	0
R	PTFSE7	PTFSE6	PTFSE5	PTFSE4	PTFSE3	PTFSE2	PTFSE1	PTFSE0
W								
Reset	0	0	0	0	0	0	0	0
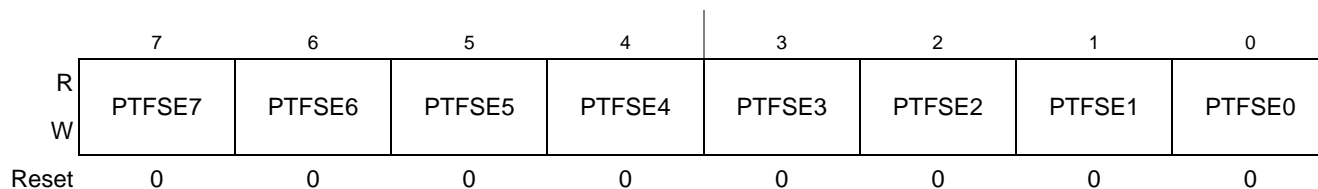
Figure 6-35. Output Slew Rate Control Enable for Port F (PTFSE)

Table 6-34. PTFSE Register Field Descriptions

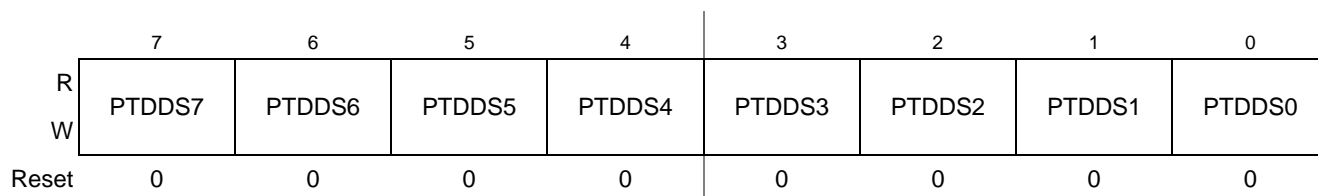| Field	Description
7:0 PTFSE[7:0]	**Output Slew Rate Control Enable for Port F Bits** — Each of these control bits determine whether output slew rate control is enabled for the associated PTF pin. For port F pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled 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	PTFDS7	PTFDS6	PTFDS5	PTFDS4	PTFDS3	PTFDS2	PTFDS1	PTFDS0
W								
Reset	0	0	0	0	0	0	0	0
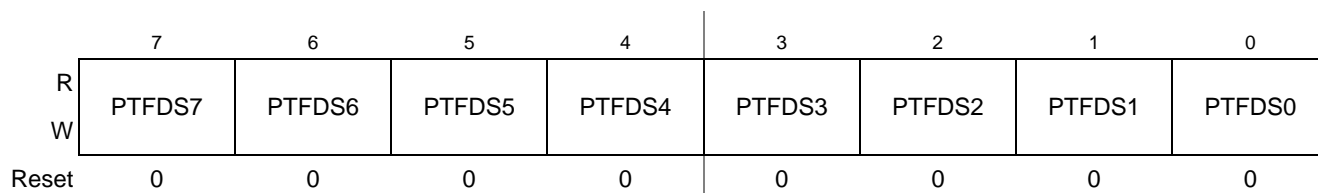
Figure 6-36. Output Drive Strength Selection for Port F (PTFDS)

Table 6-35. 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.

6.5.17.1 Port F Input Filter Enable Register (PTFIFE)

The Port F pin incorporates an optional input low-pass filter. Set the associated PTFIFE bit during and after reset to enable the filter. To disable the filter, clear the associated PTFIFE bit through software control.
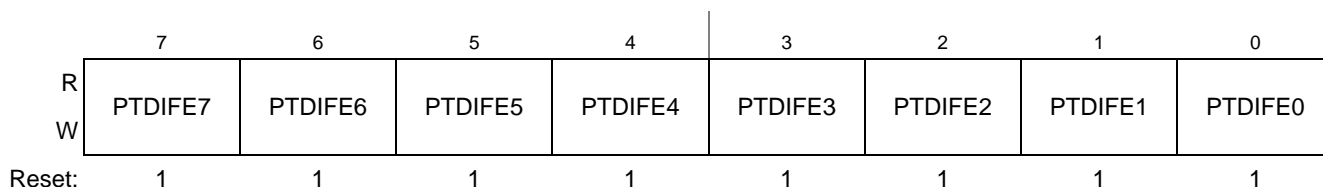
	7	6	5	4	3	2	1	0
R	PTFIFE7	PTFIFE6	PTFIFE5	PTFIFE4	PTFIFE3	PTFIFE2	PTFIFE1	PTFIFE0
W								
Reset:	1	1	1	1	1	1	1	1
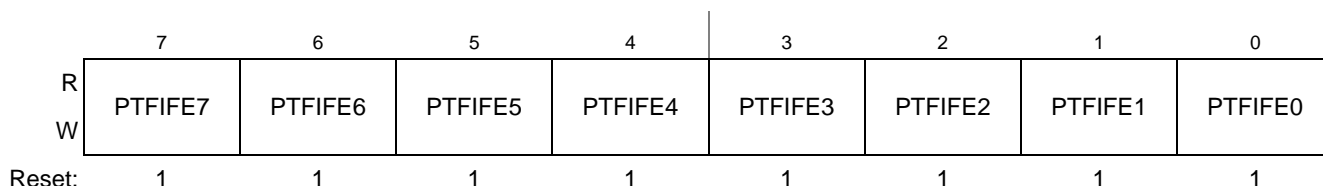
Figure 6-37. Input Filter Enable for Port F Register (PTFIFE)

Table 6-36. PTFIFE Register Field Descriptions

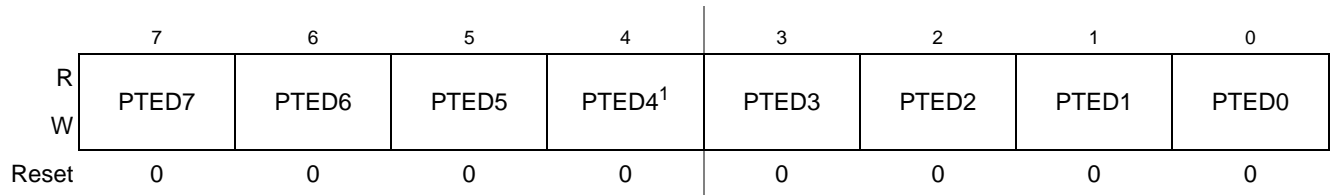| Field	Description
7–0 PTFIFE*n*	**Input Filter Enable for Port F Bits** — Input low-pass filter enable control bits for PTF pins. 0 Input filter disabled. 1 Input filter enabled.

6.5.18 Port G Registers

Port G is controlled by the following registers.

6.5.19 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							PTGD1	PTGD0
W								
Reset	0	0	0	0	0	0	0	0
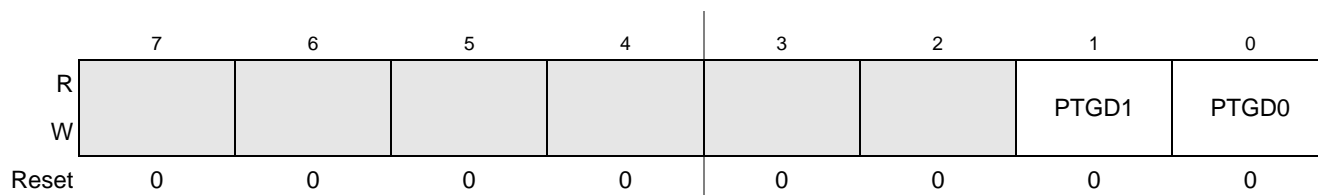
Figure 6-38. Port G Data Register (PTGD)

Table 6-37. PTGD Register Field Descriptions

Field	Description
1:0 PTGD[1:0]	**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							PTGDD1	PTGDD0
W								
Reset	0	0	0	0	0	0	0	0
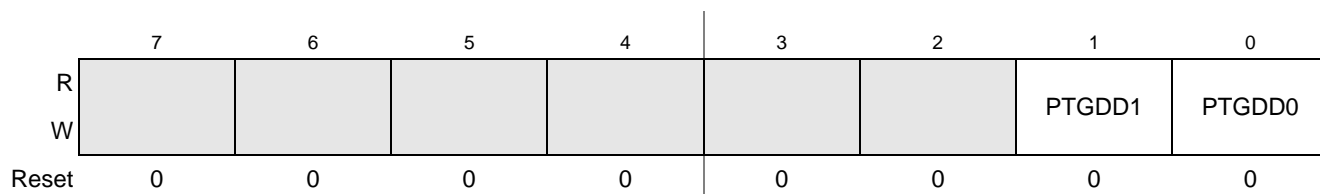
Figure 6-39. Data Direction for Port G (PTGDD)

Table 6-38. PTGDD Register Field Descriptions

Field	Description
1:0 PTGDD[1:0]	**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.

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

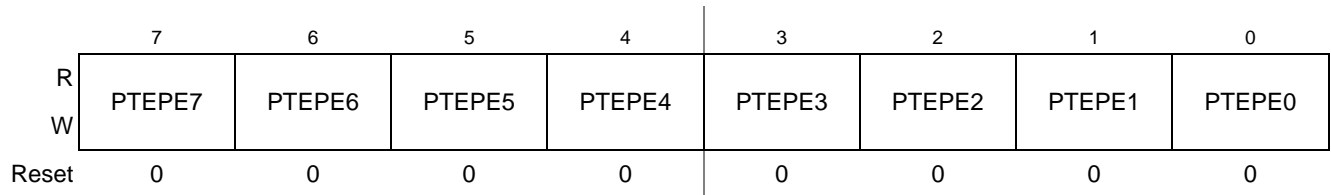In addition to the I/O control, port G pins are controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R							PTGPE1	PTGPE0
W								
Reset	0	0	0	0	0	0	0	0
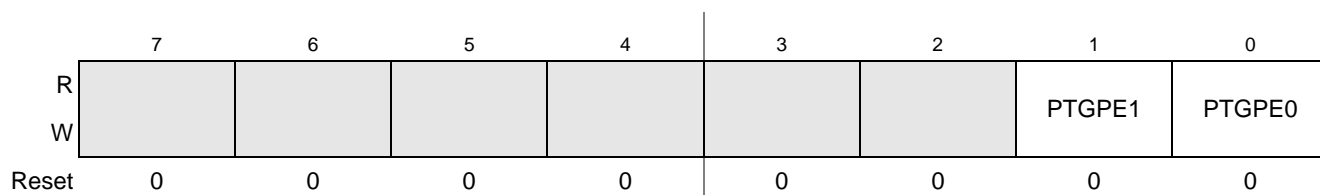
Figure 6-40. Internal Pull-up Enable for Port G Bits (PTGPE)

Table 6-39. PTGPE Register Field Descriptions

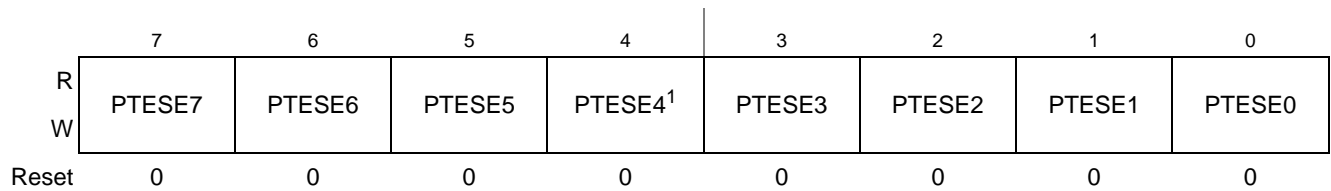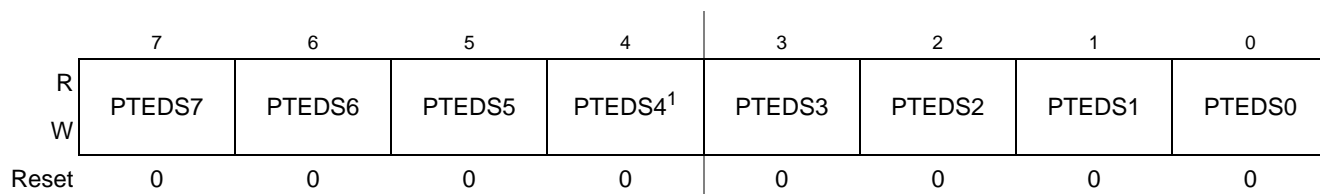| Field	Description
1:0 PTGPEn	**Internal Pull-up Enable for Port G Bits** — Each of these control bits determines if the internal pull-up device is enabled for the associated PTG pin. For port G pins that are configured as outputs, these bits have no effect and the internal pull-up devices are disabled.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

	7	6	5	4	3	2	1	0
R							PTGSE1	PTGSE0
W								
Reset	0	0	0	0	0	0	0	0

Figure 6-41. Output Slew Rate Control Enable for Port G Bits (PTGSE)

Table 6-40. 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							PTGDS1	PTGDS0
W								
Reset	0	0	0	0	0	0	0	0
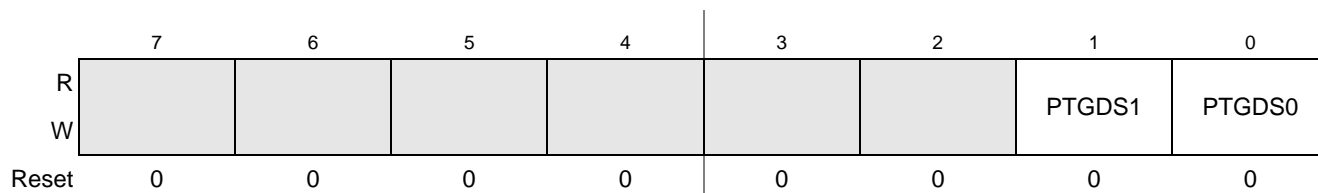
Figure 6-42. Output Drive Strength Selection for Port G (PTGDS)

Table 6-41. PTGDS Register Field Descriptions

Field	Description
1:0 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.

6.5.20.1   Port G Input Filter Enable Register (PTGIFE)

The Port G pin incorporates an optional input low-pass filter. Set the associated PTGIFE bit during and after reset to enable the filter. To disable the filter, clear the associated PTGIFE bit through software control.

	7	6	5	4	3	2	1	0
R	PTGIFE7	PTGIFE6	PTGIFE5	PTGIFE4	PTGIFE3	PTGIFE2	PTGIFE1	PTGIFE0
W								
Reset:	1	1	1	1	1	1	1	1
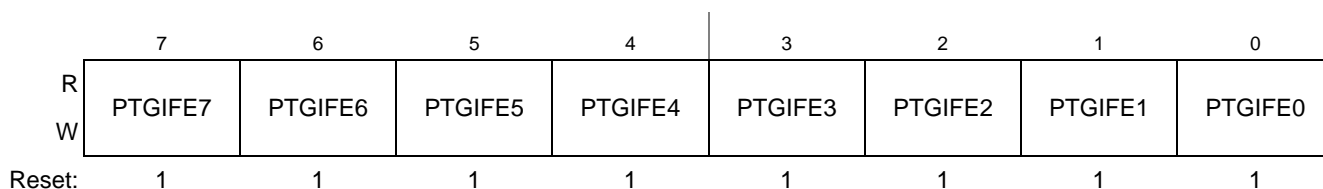
Figure 6-43. Input Filter Enable for Port G Register (PTGIFE)

Chapter 7
Keyboard Interrupt

7.1 Introduction

7.1.1 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.

7.1.2 Modes of Operation

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

7.1.2.1 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).

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

7.1.2.3 KBI in Active Background Mode

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

7.1.3 Block Diagram

The block diagram for the keyboard interrupt module is shown Figure 7-1.
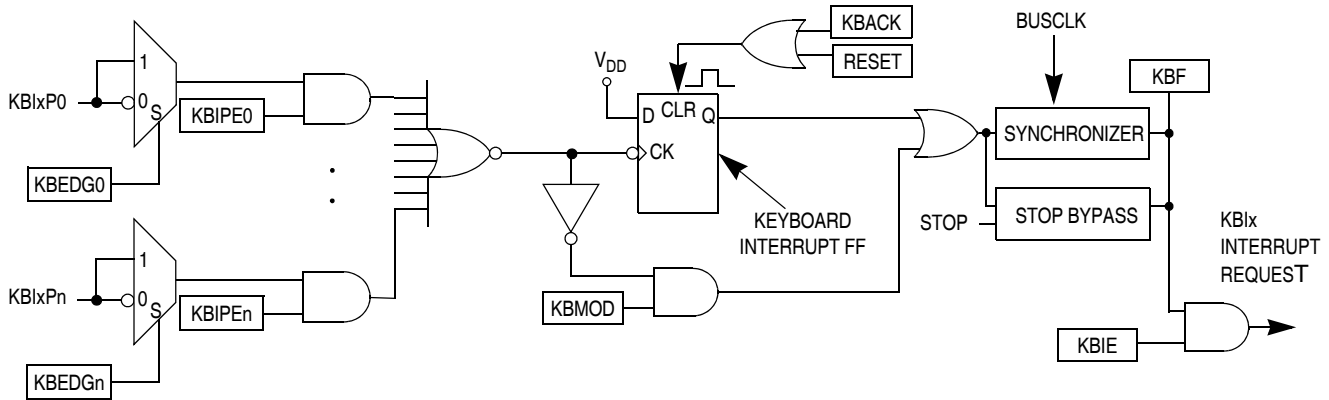
Figure 7-1. KBI Block Diagram

7.2 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 7-1.

Table 7-1. Signal Properties

Signal	Function	I/O
KBIxPn	Keyboard interrupt pins	I

7.3 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.

NOTE

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

7.3.1 KBIx Status and Control Register (KBIxSC)

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

	7	6	5	4	3	2	1	0
R	0	0	0	0	KBF	0	KBIE	KBMOD
W						KBACK		
Reset:	0	0	0	0	0	0	0	0

☐ = Unimplemented
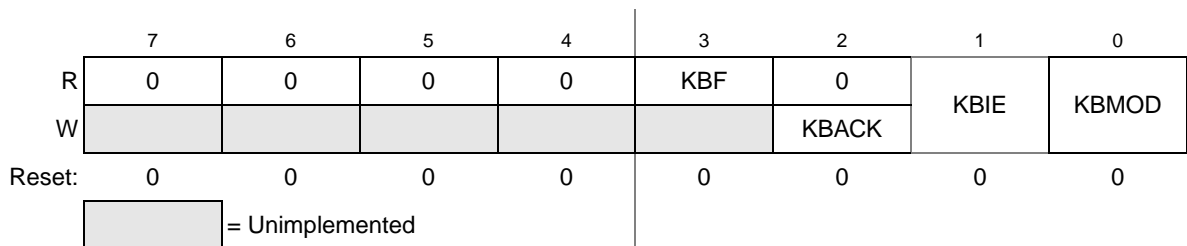
Figure 7-2. KBIx Status and Control Register

Table 7-2. KBIxSC 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. 0 Keyboard detects edges only. 1 Keyboard detects both edges and levels.

7.3.2 KBIx Pin Enable Register (KBIxPE)

KBIxPE contains the pin enable control bits.

	7	6	5	4	3	2	1	0
R W	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
Reset:	0	0	0	0	0	0	0	0

Figure 7-3. KBIx Pin Enable Register

Table 7-3. KBIxPE 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.

7.3.3 KBIx Edge Select Register (KBIxES)

KBIxES contains the edge select control bits.

	7	6	5	4	3	2	1	0
R	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
W								
Reset:	0	0	0	0	0	0	0	0
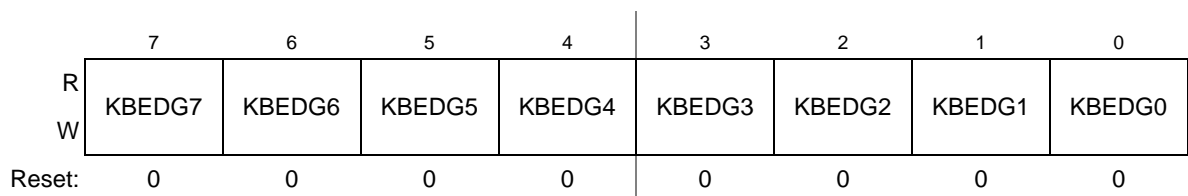
Figure 7-4. KBIx Edge Select Register

Table 7-4. KBIxES 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.

7.4   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 (KBIxPE) 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 (KBIxSC). 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 (KBIxES).

7.4.1   Edge Only Sensitivity

Synchronous logic is used to detect edges. Prior to detecting an edge, enabled keyboard inputs must be at the deasserted logic level. A falling edge is detected when an enabled keyboard 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 during one bus cycle and then a logic 1 during the next cycle.

A valid edge on an enabled KBI pin sets KBF in KBIxSC. If KBIE in KBIxSC is set, an interrupt request is presented to the CPU. To clear KBF, write a 1 to KBACK in KBIxSC.

7.4.2   Edge and Level Sensitivity

A valid edge or level on an enabled KBI pin sets KBF in KBIxSC. If KBIE in KBIxSC is set, an interrupt request is presented to the CPU. Clearing of KBF is accomplished by writing a 1 to KBACK in KBIxSC provided all enabled keyboard inputs are at their deasserted levels. KBF remains set if any enabled KBI pin is asserted while writing a 1 to KBACK.

7.4.3 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 KBIxES register is used to select whether the resistor is a pullup (KBEDGn = 0) or a pulldown (KBEDGn = 1).

7.4.4 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, do the following:

1. Clear KBIE in KBIxSC to mask keyboard interrupts.
2. Set the appropriate KBEDGn bits in KBIxES to enable the KBI polarity.
3. If using internal pullup/pulldown device, configure the associated pullup enable bits in PTxPE.
4. Set the appropriate KBIPEn bits in KBIxPE to enable the KBI pins.
5. Write to KBACK in KBIxSC to clear any false interrupts.
6. Set KBIE in KBIxSC to enable interrupts.

Chapter 8
Central Processor Unit (S08CPUV5)

8.1 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).

8.1.1 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-KB CPU 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

8.2 Programmer's Model and CPU Registers

Figure 8-1 shows the five CPU registers. CPU registers are not part of the memory map.
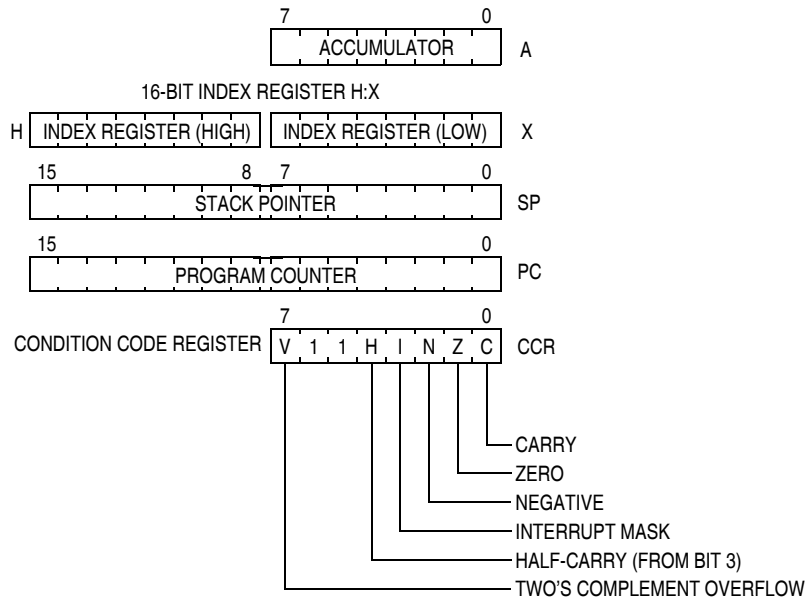
Figure 8-1. CPU Registers

8.2.1 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.

8.2.2 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.

8.2.3 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.

8.2.4 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.

8.2.5 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 8-2. Condition Code Register

Table 8-1. 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

8.3 Addressing Modes

Addressing modes define the way the CPU accesses operands and data. In the HCS08, memory, status and control registers, and input/output (I/O) ports share a single 64-Kbyte CPU address space. 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.

8.3.1 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.

8.3.2 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.

8.3.3 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.

8.3.4 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.

8.3.5 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).

8.3.6 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.

8.3.6.1 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.

8.3.6.2 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.

8.3.6.3 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.

8.3.6.4 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.

8.3.6.5 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.

8.3.6.6 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.

8.3.6.7   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.

8.4   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.

8.4.1   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.

8.4.2   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.

8.4.3 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.

8.4.4 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.

8.4.5 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.

8.5   HCS08 Instruction Set Summary

Table 8-2 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 8-2. Instruction Set Summary (Sheet 1 of 9)
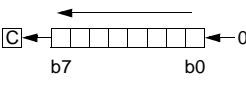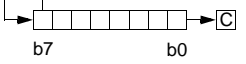
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	V 1 1 H	I N Z C
ADC #opr8i ADC opr8a ADC opr16a ADC oprx16,X ADC oprx8,X ADC ,X ADC oprx16,SP ADC oprx8,SP	Add with Carry A ← (A) + (M) + (C)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 ii B9 dd C9 hh ll D9 ee ff E9 ff F9 9E D9 ee ff 9E E9 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	Þ 1 1 Þ	– Þ Þ Þ

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
						Þ 1 1 Þ	– Þ Þ Þ

AIS #opr8i	Add Immediate Value (Signed) to Stack Pointer SP ← (SP) + (M)	IMM	A7 ii	2	pp	– 1 1 –	– – – –
AIX #opr8i	Add Immediate Value (Signed) to Index Register (H:X) H:X ← (H:X) + (M)	IMM	AF ii	2	pp	– 1 1 –	– – – –
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 1 1 –	– Þ Þ –
ASL opr8a ASLA ASLX ASL oprx8,X ASL ,X ASL oprx8,SP	Arithmetic Shift Left C ← [b7 ... b0] ← 0 (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	Þ 1 1 –	– Þ Þ Þ
ASR opr8a ASRA ASRX ASR oprx8,X ASR ,X ASR oprx8,SP	Arithmetic Shift Right [b7 ... b0] → C	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	Þ 1 1 –	– Þ Þ Þ
BCC rel	Branch if Carry Bit Clear (if C = 0)	REL	24 rr	3	ppp	– 1 1 –	– – – –

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR V 1 1 H	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	– 1 1 –	– – – –
BCS rel	Branch if Carry Bit Set (if C = 1) (Same as BLO)	REL	25 rr	3	ppp	– 1 1 –	– – – –
BEQ rel	Branch if Equal (if Z = 1)	REL	27 rr	3	ppp	– 1 1 –	– – – –
BGE rel	Branch if Greater Than or Equal To (if N ⊕ V = 0) (Signed)	REL	90 rr	3	ppp	– 1 1 –	– – – –
BGND	Enter active background if ENBDM=1 Waits for and processes BDM commands until GO, TRACE1, or TAGGO	INH	82	5+	fp...ppp	– 1 1 –	– – – –
BGT rel	Branch if Greater Than (if Z | (N ⊕ V) = 0) (Signed)	REL	92 rr	3	ppp	– 1 1 –	– – – –
BHCC rel	Branch if Half Carry Bit Clear (if H = 0)	REL	28 rr	3	ppp	– 1 1 –	– – – –
BHCS rel	Branch if Half Carry Bit Set (if H = 1)	REL	29 rr	3	ppp	– 1 1 –	– – – –
BHI rel	Branch if Higher (if C | Z = 0)	REL	22 rr	3	ppp	– 1 1 –	– – – –
BHS rel	Branch if Higher or Same (if C = 0) (Same as BCC)	REL	24 rr	3	ppp	– 1 1 –	– – – –
BIH rel	Branch if IRQ Pin High (if IRQ pin = 1)	REL	2F rr	3	ppp	– 1 1 –	– – – –
BIL rel	Branch if IRQ Pin Low (if IRQ pin = 0)	REL	2E rr	3	ppp	– 1 1 –	– – – –
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 1 1 –	– Þ Þ –
BLE rel	Branch if Less Than or Equal To (if Z | (N ⊕ V) = 1) (Signed)	REL	93 rr	3	ppp	– 1 1 –	– – – –
BLO rel	Branch if Lower (if C = 1) (Same as BCS)	REL	25 rr	3	ppp	– 1 1 –	– – – –
BLS rel	Branch if Lower or Same (if C | Z = 1)	REL	23 rr	3	ppp	– 1 1 –	– – – –
BLT rel	Branch if Less Than (if N ⊕ V = 1) (Signed)	REL	91 rr	3	ppp	– 1 1 –	– – – –
BMC rel	Branch if Interrupt Mask Clear (if I = 0)	REL	2C rr	3	ppp	– 1 1 –	– – – –
BMI rel	Branch if Minus (if N = 1)	REL	2B rr	3	ppp	– 1 1 –	– – – –
BMS rel	Branch if Interrupt Mask Set (if I = 1)	REL	2D rr	3	ppp	– 1 1 –	– – – –
BNE rel	Branch if Not Equal (if Z = 0)	REL	26 rr	3	ppp	– 1 1 –	– – – –
BPL rel	Branch if Plus (if N = 0)	REL	2A rr	3	ppp	– 1 1 –	– – – –
BRA rel	Branch Always (if I = 1)	REL	20 rr	3	ppp	– 1 1 –	– – – –

Table 8-2. Instruction Set Summary (Sheet 3 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR V 1 1 H	I N Z C
BRCLR *n,opr8a,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	– 1 1 –	– – – Þ
BRN *rel*	Branch Never (if I = 0)	REL	21 rr	3	ppp	– 1 1 –	– – – –
BRSET *n,opr8a,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	– 1 1 –	– – – Þ
BSET *n,opr8a*	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	– 1 1 –	– – – –
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	– 1 1 –	– – – –
CBEQ *opr8a,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	– 1 1 –	– – – –
CLC	Clear Carry Bit (C ← 0)	INH	98	1	p	– 1 1 –	– – – 0
CLI	Clear Interrupt Mask Bit (I ← 0)	INH	9A	1	p	– 1 1 –	0 – – –
CLR *opr8a* 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 1 1 –	– 0 1 –

Table 8-2. Instruction Set Summary (Sheet 4 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	V 1 1 H	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
						Þ 1 1 –	– Þ Þ Þ

COM opr8a COMA COMX COM oprx8,X COM ,X COM oprx8,SP	Complement M ← (M̄)= $FF – (M) (One's Complement) A ← (Ā) = $FF – (A) X ← (X̄) = $FF – (X) M ← (M̄) = $FF – (M) M ← (M̄) = $FF – (M) M ← (M̄) = $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 1 –	– Þ Þ 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	Þ 1 1 –	– Þ Þ Þ
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	Þ 1 1 –	– Þ Þ Þ
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	INH	72	1	p	U 1 1 –	– Þ Þ Þ
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	– 1 1 –	– – – –
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	Þ 1 1 –	– Þ Þ –
DIV	Divide A ← (H:A)÷(X); H ← Remainder	INH	52	6	fffffp	– 1 1 –	– – Þ Þ
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 1 1 –	– Þ Þ –

Table 8-2. Instruction Set Summary (Sheet 5 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	V 1 1 H	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	Þ 1 1 –	– Þ Þ –
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	– 1 1 –	– – – –
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	– 1 1 –	– – – –

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 1 1 –	– Þ Þ –

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 1 1 –	– Þ Þ –

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 1 1 –	– Þ Þ –

LSL opr8a LSLA LSLX LSL oprx8,X LSL ,X LSL oprx8,SP	Logical Shift Left C ← [ ] ← 0 b7 b0 (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	Þ 1 1 –	– Þ Þ Þ
LSR opr8a LSRA LSRX LSR oprx8,X LSR ,X LSR oprx8,SP	Logical Shift Right 0 → [ ] → C b7 b0	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	Þ 1 1 –	– 0 Þ Þ

Table 8-2. Instruction Set Summary (Sheet 6 of 9)
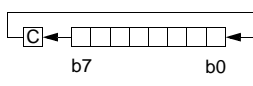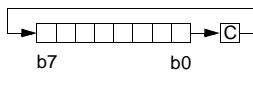
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	V 1 1 H	I N Z C

MOV *opr8a*,*opr8a*
MOV *opr8a*,X+
MOV #*opr8i*,*opr8a*
MOV ,X+,*opr8a*
	Move
	(M)$_{destination}$ ← (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 1 1 –	– Þ Þ –

MUL	Unsigned multiply X:A ← (X) × (A)	INH	42	5	ffffp	– 1 1 0	– – – 0
NEG *opr8a* 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	Þ 1 1 –	– Þ Þ Þ
NOP	No Operation — Uses 1 Bus Cycle	INH	9D	1	p	– 1 1 –	– – – –
NSA	Nibble Swap Accumulator A ← (A[3:0]:A[7:4])	INH	62	1	p	– 1 1 –	– – – –
ORA #*opr8i* ORA *opr8a* ORA *opr16a* 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 1 1 –	– Þ Þ –
PSHA	Push Accumulator onto Stack Push (A); SP ← (SP) – $0001	INH	87	2	sp	– 1 1 –	– – – –
PSHH	Push H (Index Register High) onto Stack Push (H); SP ← (SP) – $0001	INH	8B	2	sp	– 1 1 –	– – – –
PSHX	Push X (Index Register Low) onto Stack Push (X); SP ← (SP) – $0001	INH	89	2	sp	– 1 1 –	– – – –
PULA	Pull Accumulator from Stack SP ← (SP + $0001); Pull (A)	INH	86	3	ufp	– 1 1 –	– – – –
PULH	Pull H (Index Register High) from Stack SP ← (SP + $0001); Pull (H)	INH	8A	3	ufp	– 1 1 –	– – – –
PULX	Pull X (Index Register Low) from Stack SP ← (SP + $0001); Pull (X)	INH	88	3	ufp	– 1 1 –	– – – –
ROL *opr8a* 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	Þ 1 1 –	– Þ Þ Þ
ROR *opr8a* 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	Þ 1 1 –	– Þ Þ Þ

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	V 1 1 H	I N Z C
RSP	Reset Stack Pointer (Low Byte) SPL ← $FF (High Byte Not Affected)	INH	9C	1	p	– 1 1 –	– – – –
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	Þ 1 1 Þ	Þ Þ Þ Þ
RTS	Return from Subroutine SP ← SP + $0001; Pull (PCH) SP ← SP + $0001; Pull (PCL)	INH	81	5	ufppp	– 1 1 –	– – – –
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	Þ 1 1 –	– Þ Þ Þ
SEC	Set Carry Bit (C ← 1)	INH	99	1	p	– 1 1 –	– – – 1
SEI	Set Interrupt Mask Bit (I ← 1)	INH	9B	1	p	– 1 1 –	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 1 1 –	– Þ Þ –
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 1 1 –	– Þ Þ –
STOP	Enable Interrupts: Stop Processing Refer to MCU Documentation I bit ← 0; Stop Processing	INH	8E	2	fp...	– 1 1 –	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 1 1 –	– Þ Þ –

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	**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
						Þ 1 1 –	– Þ Þ Þ

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	sssssvvfppp	– 1 1 –	1 – – –
TAP	Transfer Accumulator to CCR CCR ← (A)	INH	84	1	p	Þ 1 1 Þ	Þ Þ Þ Þ
TAX	Transfer Accumulator to X (Index Register Low) X ← (A)	INH	97	1	p	– 1 1 –	– – – –
TPA	Transfer CCR to Accumulator A ← (CCR)	INH	85	1	p	– 1 1 –	– – – –
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 1 1 –	– Þ Þ –
TSX	Transfer SP to Index Reg. H:X ← (SP) + $0001	INH	95	2	fp	– 1 1 –	– – – –
TXA	Transfer X (Index Reg. Low) to Accumulator A ← (X)	INH	9F	1	p	– 1 1 –	– – – –

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	I N Z C
TXS	Transfer Index Reg. to SP SP ← (H:X) − $0001	INH	94	2	fp	– 1 1 –	– – – –
WAIT	Enable Interrupts; Wait for Interrupt I bit ← 0; Halt CPU	INH	8F	2+	fp...	– 1 1 –	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　　Program 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:

Þ　　Set or cleared
–　　Not affected
U　　Undefined

Table 8-3. Opcode Map (Sheet 1 of 2)

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

Opcode in Hexadecimal — F0 — 3 — HCS08 Cycles
SUB — Instruction Mnemonic
Number of Bytes — 1 — IX — Addressing Mode

Table 8-3. 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 Post Increment	
EXT	Extended	IX2	Indexed, 16-Bit Offset			
DD	DIR to DIR	IMD	IMM to DIR	IX1+	Indexed, 1-Byte Offset with Post Increment	
IX+D	IX+ to DIR	DIX+	DIR to IX+			

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

Prebyte (9E) and Opcode in Hexadecimal — 9E60 6 — HCS08 Cycles
NEG — Instruction Mnemonic
Number of Bytes — 3 SP1 — Addressing Mode

Chapter 9
Programmable Analog Comparator (S08PRACMPV1)

9.1 Introduction

The PRACMP is a CMOS comparator with a programmable reference input. For the MC9S08JE128 family, the comparator has up to 4 input pins, each of them can be compared with any input pins.There is also a internal programmable reference generator which divides the VIn into 32 levels, the VIn can be selected from two external sources. Output of this reference generator can be one of the eight inputs of comparator. The comparator circuit is designed to operate across the full range of the supply voltage (rail-to-rail operation).

9.1.1 PRACMP Configuration Information

When using the bandgap reference voltage for input to PRACMP, enable the VREF output to supply a reference signal to the PRACMP. See chapter VREF for information on how to enable the VREF output. For the value of the bandgap voltage reference, see the MC9S08JE128 series data sheet.

NOTE

For the MC9S08JE128 series devices,V_{in2} is connected to VREFO. Therefore, the PRGINS bit n the PRACMP1 register selects:

- VREFO when PRGINS=0
- V_{DD} when PRGINS=1

9.1.2 PRACMP/TPM Configuration Information

The PRACMP module can be configured to connect the output of the analog comparator to TPM1 input capture channel 0 by setting the ACIC bit in SOPT2. With ACIC set, the TPM1CH0 pin is not available externally regardless of the configuration of the TPM1 module.

9.1.3 PRACMP Clock Gating

The bus clock to the ACMP can be gated on and off using the PRACMP bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the PRACMP bit can be cleared to disable the clock to this module when not in use. See Section 5.7.9, "System Clock Gating Control 2 Register (SCGC2)," for details.
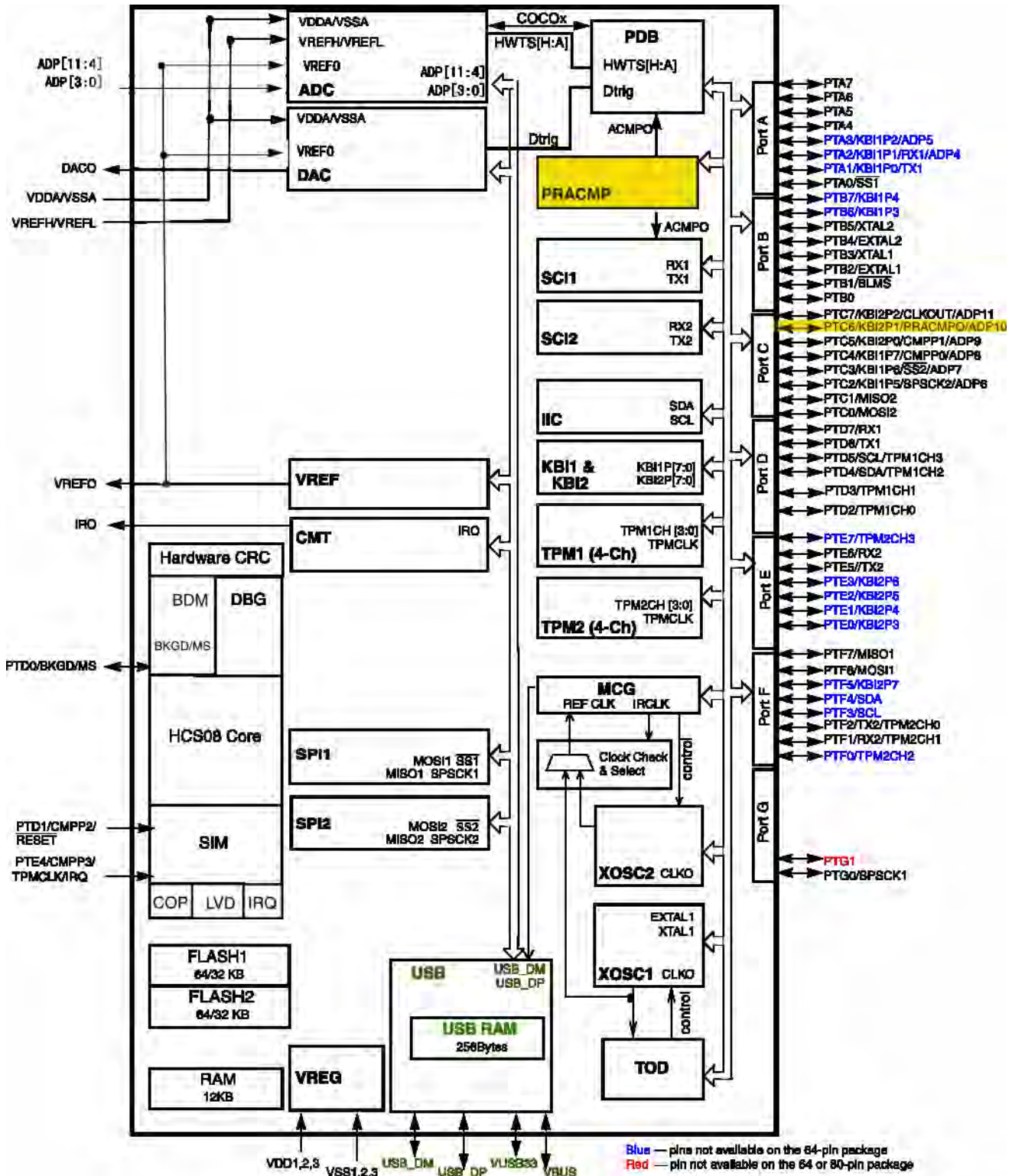
Figure 9-1. MC9S08JE128 series Block Diagram

9.1.4 Features

PRACMP features include:

- MC9S08JE128 series On-chip programmable reference generator output ($1/32\ V_{in}$ to V_{in}, step is $1/32\ V_{in}$, V_{in} can be selected from external V_{DD} and internal VREFO)
- Typically 5 mV of input offset
- Less than 40 μA in enable mode and less than 1 nA in disable mode (excluding programmable reference generator)
- Fixed ACMP hysteresis which is from 3 mV to 20 mV
- Up to eight selectable comparator inputs; each input can be compared with any input by any polarity sequence
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output
- Remains operational in stop3 mode

9.1.5 Modes of Operation

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

9.1.5.1 Operation in Wait Mode

The PRACMP continues to operate in wait mode if enabled. The interrupt can wake up the MCU if enabled.

9.1.5.2 Operation in Stop Mode

The PRACMP (including PRG and ACMP) continues to operate in stop3 mode if enabled. If ACIEN is set, a PRACMP interrupt still can be generated to wake the MCU up from stop3 mode.

If the stop3 is exited by an interrupt, the PRACMP remains the setting before entering the stop. If stop3 is exited with a reset, the PRACMP goes into its reset.

To conserve power, turn it off if its output is not used as a reference input of ACMP, because the PRG consumes additional power.

In stop2 mode, the PRACMP is shut down completely. Any waking up from stop2 brings PRACMP to its reset state.

9.1.5.3 Operation in Background Mode

When the MCU is in active background debug mode, the PRACMP continues operating normally.

9.1.6 Block Diagram

The block diagram of the PRACMP module is shown in Figure 9-2.
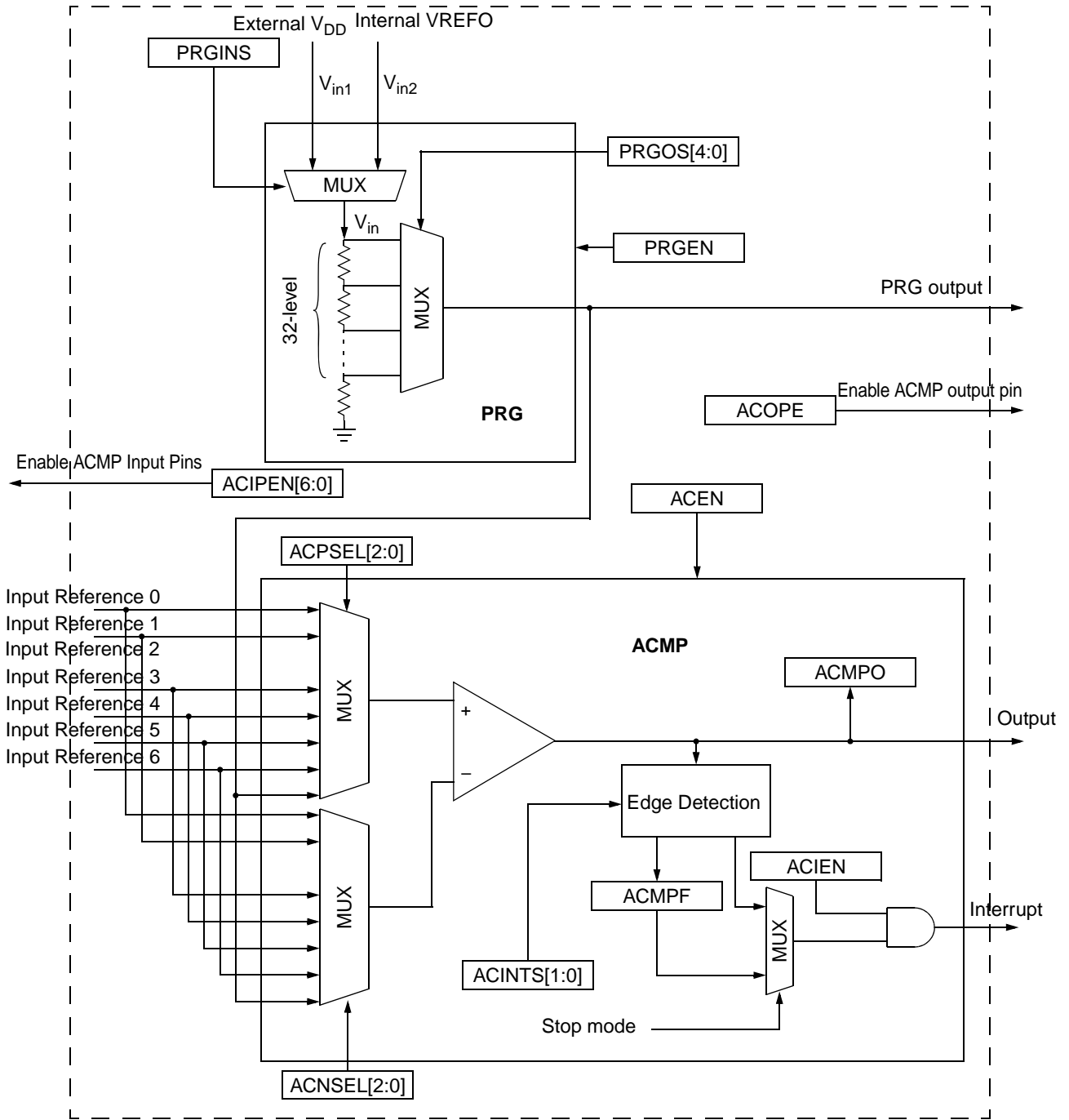
Figure 9-2. PRACMP Block Diagram

9.2 External Signal Description

The output of PRACMP can also be mapped to an external pin. When the output is mapped to an external pin, register bit ACOPE controls the pin to enable/disable the PRACMP output function.

9.3 Memory Map and Register Definition

Table 9-1 is the memory map of the programmable reference analog comparator (PRACMP).

Table 9-1. Module Memory Map

Address	Use	Access
Base + $0	PRACMP Control and Status Register (PRACMPCS)	Read/Write
Base + $1	PRACMP Control Register 0 (PRACMPC0)	Read/Write
Base + $2	PRACMP Control Register 1 (PRACMPC1)	Read/Write
Base + $3	PRACMP Control Register 2 (PRACMPC2)	Read/Write

Refer to the direct-page register summary in the memory chapter of this reference manual for the absolute address assignments for all PRACMP registers.

9.3.1 PRACMP Control and Status Register (PRACMPCS)

	7	6	5	4	3	2	1	0
R	ACEN	ACMPF		ACOPE	ACMPO	ACINTS		ACIEN
W								
Reset	0	0	0	0	0	0	0	0

Unimplemented or Reserved
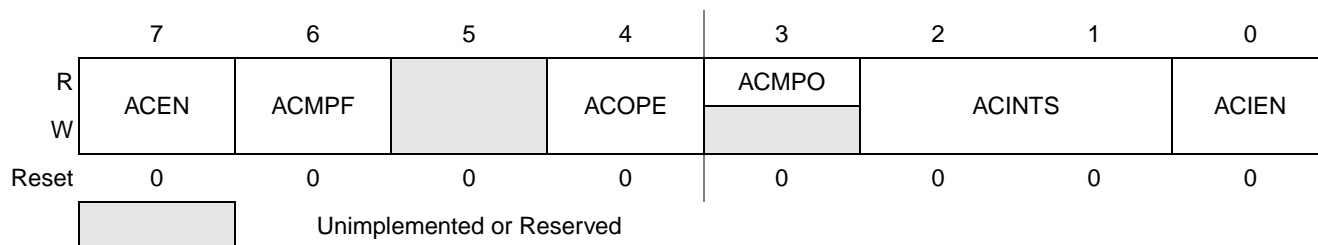
Figure 9-3. PRACMP Control and Status Register (PRACMPCS)

Table 9-2. PRACMPCS Descriptions

Field	Description
7 ACEN	**ACMP enable control bit** 0 The ACMP is disabled 1 The ACMP is enabled
6 ACMPF	**ACMP Interrupt Flag Bit** — Synchronously set by hardware when ACMP output has a valid edge defined by ACINTS[1:0]. The setting of this bit lags the ACMPO 2 bus clocks. Clear ACMPF bit by writing a 0 to this bit. Writing a 1 to this bit has not effect.
4 ACOPE	**ACMP Output Pin Enable** — ACOPE enables the pad logic so that the output can be placed onto an external pin 0 Output of ACMP can't be placed onto external pin 1 Output of ACMP can be placed onto external pin

Table 9-2. PRACMPCS Descriptions (Continued)

Field	Description
3 ACMPO	**ACMP Output Bit** — ACMP output is synchronized by bus clock to form this bit. It changes following the ACMP output. This bit is a read only bit. • Set when the output of the ACMP is high • Cleared when the output of the ACMP is low • After any reset or when the ACMP is disabled, this bit is read as 0.
2:1 ACINTS [1:0]	**ACMP Interrupt Select** — Determines the sensitivity modes of the interrupt trigger. 00 ACMP interrupt on output falling or rising edge 01 ACMP interrupt on output falling edge 10 ACMP interrupt on output rising edge 11 Reserved
0 ACIEN	**ACMP Interrupt Enable** — Enables an ACMP CPU interrupt 1 Enable the ACMP Interrupt 0 Disable the ACMP Interrupt

9.3.2   PRACMP Control Register 0 (PRACMPC0)

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

Unimplemented or Reserved
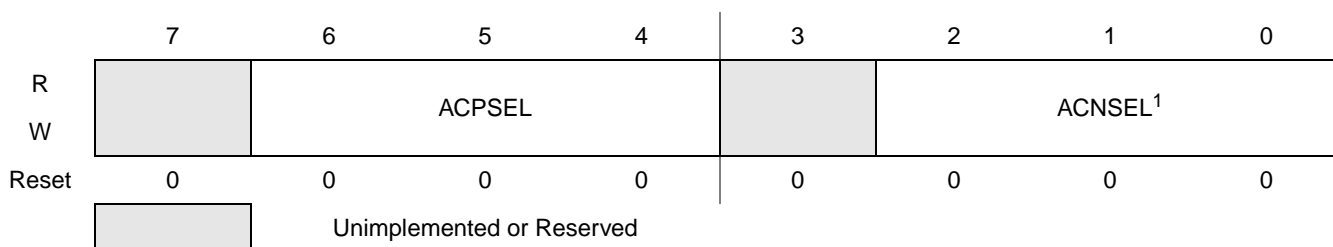
Figure 9-4. PRACMP Control Register 0 (PRACMPC0)

[1] Selects negative input, same as ACPSEL

Table 9-3. PRACMPC0 Field Descriptions

Field	Description
6:4 ACPSEL[2:0]	**ACMP Positive Input Select**[1] 000 External reference 0, CMPP0 001 External reference 1, CMPP1 010 External reference 2, CMPP2 011 External reference 3, CMPP3 100 RESERVED 101 RESERVED 110 Buffered Bandgap Voltage , PMC bandgap VREF 111 Internal PRG output, external V_{DD}, VREF out
2:0 ACNSEL[2:0]	**ACMP Negative Input Select**[1] 000 External reference 0, CMPP0 001 External reference 1, CMPP1 010 External reference 2, CMPP2 011 External reference 3, CMPP3 100 RESERVED 101 RESERVED 110 Buffered Bandgap Voltage , PMC bandgap VREF 111 Internal PRG output, external V_{DD}, VREF out

[1] Do not configure ACPSEL and ACNSEL to use the same value to operate the comparator. Selecting the same channel for the comparator may cause an unstable oscilliation on the output of the comparator.

9.3.3   PRACMP Control Register 1 (PRACMPC1)

	7	6	5	4	3	2	1	0
R	PRGEN	PRGINS		PRGOS4	PRGOS3	PRGOS2	PRGOS1	PRGOS0
W								
Reset	0	0	0	0	0	0	0	0

	Unimplemented or Reserved
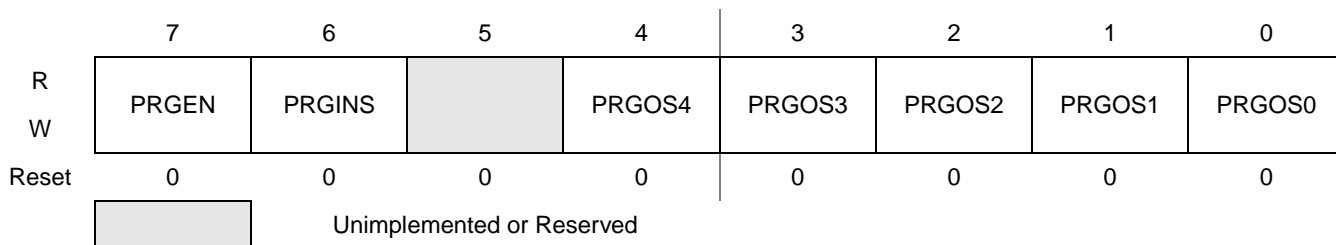
Figure 9-5. PRACMP Control Register 1 (PRACMPC1)

Table 9-4. PRACMPC1 Field Descriptions

Field	Description
7 PRGEN	**Programmable Reference Generator Enable** — The PRGEN bit starts the Programmable Reference Generator operation. 0 The PRG system is disabled 1 The PRG system is enabled
6 PRGINS	**Programmable Reference Generator Input Selection** 0 The PRG selects VREF out (~1.2V) as the reference voltage 1 The PRG selects V_{in1} (external power supply)as the reference voltage
4:0 PRGOS[4:0]	**Programmable Reference Generator Output Selection** — The output voltage is selected by the following formula: $V_{output}= (V_{in}/32)\times(PRGOS[4:0]+1)$ The V_{output} range is from $V_{in}/32$ to V_{in}, the step is $V_{in}/32$

Table 9-5 lists the output configuration of programmable reference generator (PRG).

Table 9-5. PRG Out Configuration

PRGOS[4:0]	Output Voltage of PRG
00000	$1V_{In}/32$
00001	$2V_{In}/32$
00010	$3V_{In}/32$
00011	$4V_{In}/32$
00100	$5V_{In}/32$
00101	$6V_{In}/32$
00110	$7V_{In}/32$
00111	$8V_{In}/32$
01000	$9V_{In}/32$
01001	$10V_{In}/32$
01010	$11V_{In}/32$
01011	$12V_{In}/32$

Table 9-5. PRG Out Configuration (Continued)

PRGOS[4:0]	Output Voltage of PRG
01100	$13V_{In}/32$
01101	$14V_{In}/32$
01110	$15V_{In}/32$
01111	$16V_{In}/32$
10000	$17V_{In}/32$
10001	$18V_{In}/32$
10010	$19V_{In}/32$
10011	$20V_{In}/32$
10100	$21V_{In}/32$
10101	$22V_{In}/32$
10110	$23V_{In}/32$
10111	$24V_{In}/32$
11000	$25V_{In}/32$
11001	$26V_{In}/32$
11010	$27V_{In}/32$
11011	$28V_{In}/32$
11100	$29V_{In}/32$
11101	$30V_{In}/32$
11110	$31V_{In}/32$
11111	V_{In}

9.3.4   PRACMP Control Register 2 (PRACMPC2)

	7	6	5	4	3	2	1	0
R	0	ACIPE6	ACIPE5	ACIPE4	ACIPE3	ACIPE2	ACIPE1	ACIPE0
W								
Reset	0	0	0	0	0	0	0	0

Unimplemented or Reserved
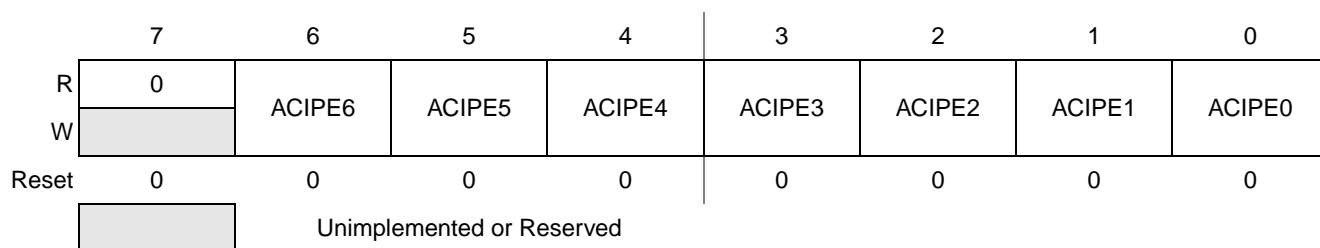
Figure 9-6. PRACMP Control Register 2 (PRACMPC2)

Table 9-6. PRACMPC2 Field Descriptions

Field	Description
6:0 ACIPE6:ACIPE0	ACMP Input Pin Enable — This 7-bit register controls if the corresponding PRACMP external pin can be driven an analog input. 0 The corresponding external analog input is not allowed 1 The corresponding external analog input is allowed

9.4 Functional Description

The PRACMP module is functionally composed of two parts: programmable reference generator (PRG) and analog comparator (ACMP).

The programmable reference generator (PRG) includes a 32-level DAC (digital to analog convertor) and relevant control logic. PRG can select one of two reference inputs, V_{in1}(external Vdd) or V_{in2}(internal regulated Vdd), as the DAC input V_{in} by setting PRGINS bit of PRACMPC1. After the DAC is enabled, it converts the data set in PRGOS[4:0] bits of PRACMPC1 to a stepped analog output which is fed into ACMP as an internal reference input. This stepped analog ouput is also mapped out of the module. The output voltage range is from $V_{in}/32$ to $V_{in.}$.The step size is $V_{in}/32$.

The ACMP can achieve the analog comparison between positive input and negative input, and then give out a digital output and relevant interrupt. Both the positive and negative input of ACMP can be selected from the eight common inputs: seven external reference inputs and one internal reference input from the PRG output. The positive input of ACMP is selected by ACPSEL[2:0] bits of PRACMPC0 and the negative input is selected by ACNSEL[2:0] bits of PRACMPC0. Any pair of the eight inputs can be compared by configuring the PRACMPC0 with the appropriate value.

After the ACMP is enabled by setting ACEN in PRACMPCS, the comparison result appears as a digital output. Whenever a valid edge defined in ACINTS[1:0] occurs, the ACMPF bit in PRACMPCS register is asserted. If ACIEN is set, a PRACMP CPU interrupt occurs. The valid edge is defined by ACINTS[1:0].When ACINTS[1:0] = 00, both the rising edge and falling edge on the ACMP output are valid. When ACINTS[1:0] = 01, only the falling edge on ACMP output is valid. When ACINTS[1:0] = 10, only rising edge on ACMP output is valid. ACINTS[1:0] = 11 is reserved.

The ACMP output is synchronized by the bus clock to generate ACMPO bit in PRACMPCS so that the CPU can read the comparison. In stop3 mode if the output of ACMP is changed, ACMPO can't be updated in time. The output can be synchronized and the ACMPO bit can be updated upon the waking up of the CPU because of the availability of the bus clock. The ACMPO changes following the comparison result, so it can serve as a tracking flag that continuously indicates the voltage delta on the inputs.

If a reference input external to the chip is selected as an input of ACMP, the corresponding ACIPE bit of PRACMPC2 should be set to enable the input from pad interface. If the output of the ACMP needs to be put onto the external pin, the ACOPE bit of PRACMPCS must enable the ACMP pin function of pad logic.

9.5 Setup and Operation of PRACMP

The two parts of PRACMP (PRG and ACMP) can be set up and operated independently. But if the PRG works as an input of the ACMP, the PRG must be configured before the ACMP is enabled.

Because the input-switching can cause problems on the ACMP inputs, the user should complete the input selection before enabling the ACMP and should not change the input selection setting when the ACMP is enabled to avoid unexpected output. Similarly, because the programmable reference generator (PRG) experiences a setup delay after the PRGOS[4:0] is changed, the user should complete the setting of PRGOS[4:0] before PRG is enabled.

9.6 Resets

During a reset the PRACMP is configured in the default mode. Both ACMP and PRG are disabled.

9.7 Interrupts

NOTE

If the bus clock is available when a valid edge (as defined in ACINTS[1:0]) occurs, the ACMPF bit in PRACMPCS register is asserted.

- If ACIEN is set, a PRACMP interrupt event occurs. The ACMPF bit remains asserted until the PRACMP interrupt is cleared by software
- When in stop3 mode, a valid edge on ACMP output generates an asynchronous interrupt which can wake the MCU up from stop3. To clear the interrupt, write a 0 to the ACMPF bit.

Chapter 10
Analog-to-Digital Converter (S08ADC12V1)

10.1   Introduction

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

NOTE

In this device, VDDA external pin is referred to as VDDAD and VSSA external pin is referred to as VSSAD.

Figure 10-1 shows the MC9S08JE128 series with the ADC module highlighted.

10.1.1   ADC Reference Selection

External VREFL and VREFH pins are the ADC conversion reference.

10.1.2   Module Configurations

This section provides device-specific information for configuring the ADC on MC9S08JE128 series.

10.1.2.1   Configurations for Stop Modes

The ADC, if enabled, must be configured to use the asynchronous clock source, ADACK, to meet the ADC minimum frequency requirements. The VREF output must be enabled in order to convert the bandgap channel in stop mode.

10.1.2.2   Single-Ended Channel Assignments

The ADC single-ended channel assignments for the MC9S08JE128 series devices are shown in Table 10-1. Reserved channels convert to an unknown value.

Table 10-1. ADC Single-Ended Channel Assignment

ADCH	Channel	Input	Pin Control	ADCH	Channel	Input	Pin Control
00000	DADP0	ADP0	ADPC0	10000	AD16	Reserved	N/A
00001	DADP1	ADP1	ADPC1	10001	AD17	Reserved	N/A
00010	DADP2	ADP2	ADPC2	10010	AD18	Reserved	N/A
00011	DADP3	ADP3	ADPC3	10011	AD19	Reserved	N/A
00100	AD4	PTA2/KBI1P1/RX1/ADP4	ADPC4	10100	AD20	Reserved	N/A

Table 10-1. ADC Single-Ended Channel Assignment (Continued)

ADCH	Channel	Input	Pin Control	ADCH	Channel	Input	Pin Control
00101	AD5	PTA3/KBI1P2/FB_D6/ADP5	ADPC5	10101	AD21	Reserved	N/A
00110	AD6	PTC2/KBI1P5/SPSCK2/ADP6	ADPC6	10110	AD22	Reserved	N/A
00111	AD7	PTC3/KBI1P6/SS2/ADP7	ADPC7	10111	AD23	Reserved	N/A
01000	AD8	PTC4/KBI1P7/CMPP0/ADP8	ADPC8	11000	AD24	Reserved	N/A
01001	AD9	PTC5/KBI2P0/CMPP1/ADP9	ADPC9	11001	R	Reserved	N/A
01010	AD10	PTC6/KBI2P1/PACMPO/ADP10	ADPC10	11010	Temp Single-ended	Temperature Sensor[1]	N/A
01011	AD11	PTC7/KBI2P2/CLKOUT/ADP11	ADPC11	11011	Bandgap	VREFO	N/A
01100	AD12	Reserved	N/A	11100	R	Reserved	N/A
01101	AD13	Reserved	N/A	11101	V_{REFH}	V_{REFH}	N/A
01110	AD14	Reserved	N/A	11110	Reserved		N/A
01111	AD15	Reserved	N/A	11111	Module Disabled	None	N/A

[1] 1 For information, see Section 10.1.2.5, "Temperature Sensor."

NOTE

Enable the VREF output to supply the bandgap voltage. See Chapter 23, "Voltage Reference Module (S08VREFV1)," for information on how to enable the VREF output. For the value of the bandgap voltage reference, see the data sheet.

10.1.2.3 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 the MC9S08JE128 series is connected to the MCGERCLK. See Chapter 15, "Multipurpose Clock Generator (S08MCGV3)" for more information.

10.1.2.4 Hardware Trigger

The ADC hardware trigger can be provided from the:

- Time of Day (TOD) Module
- Programmable Delay Block (PDB) Module

The ADCTRS bit from the SIMIPS register selects the hardware trigger source.

10.1.2.5   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.

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

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 in the data sheet.

In application code, the user reads the temperature sensor channel, calculates V_{TEMP}, and compares it 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.
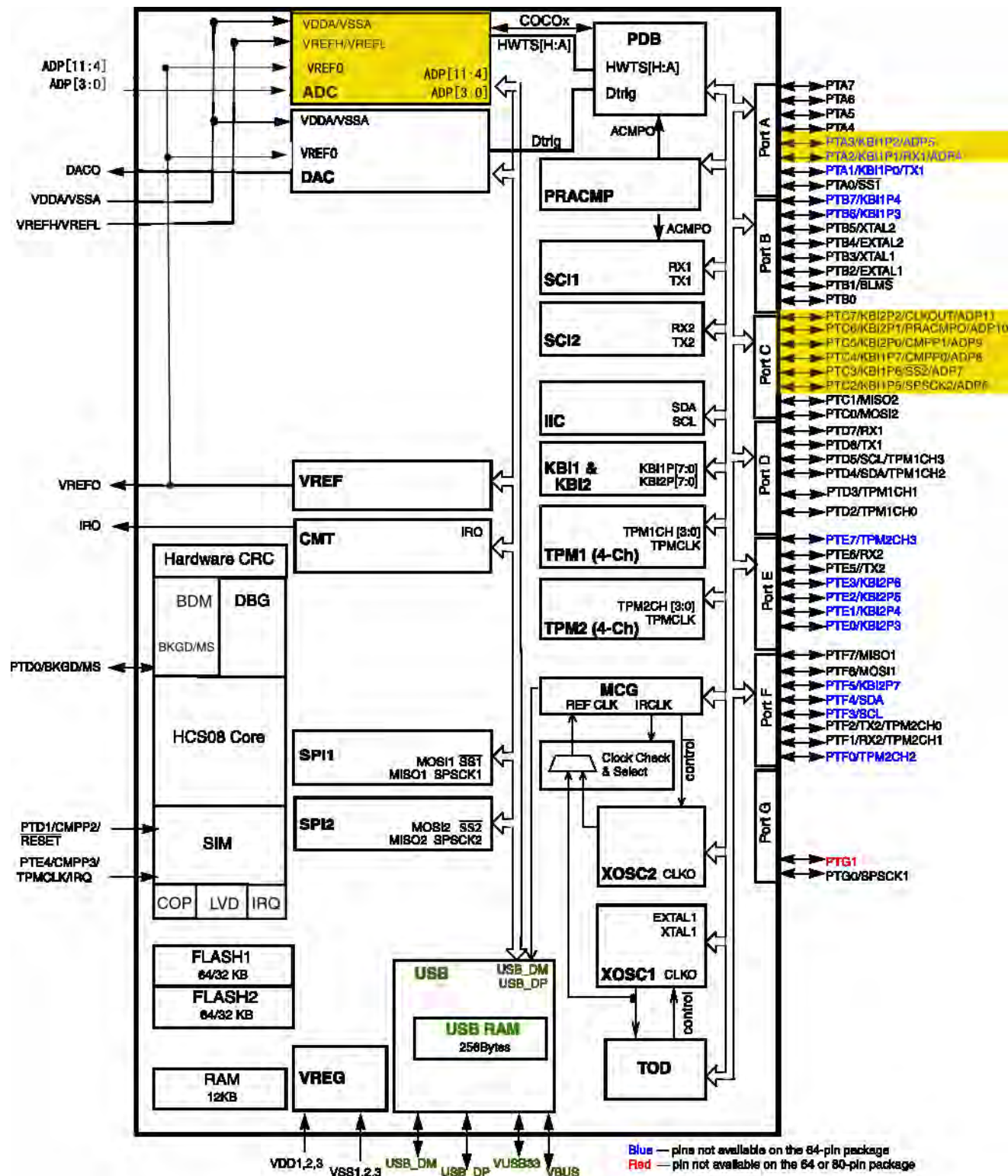
Figure 10-1. MC9S08JE128 Block Diagramwith ADC Highlighted
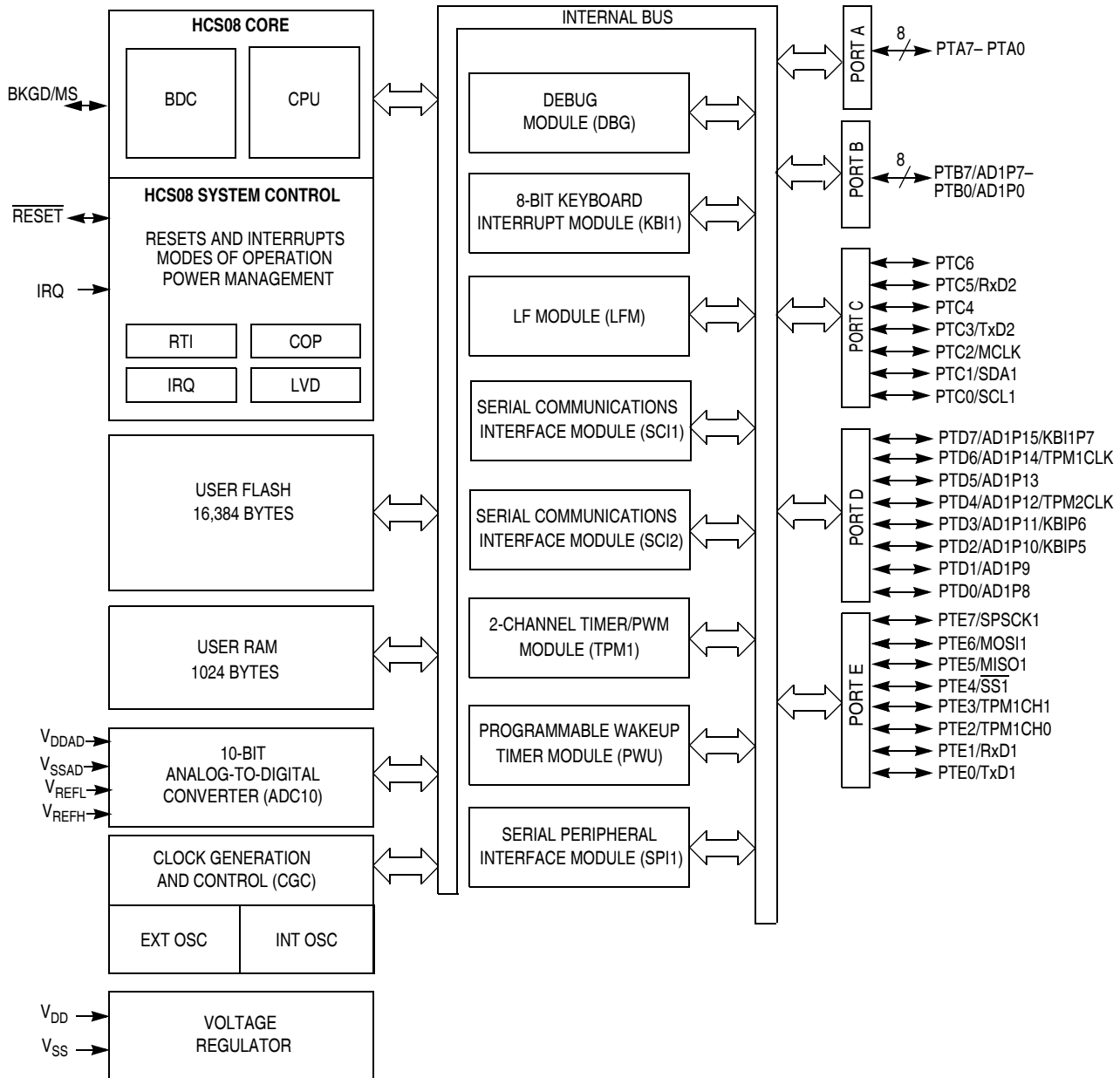
Figure 10-2. MC9S08JE128 Block Diagram

10.1.3 ADC Clock Gating

The bus clock to the ADC can be gated on and off using the ADC bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the ADC bit can be cleared to disable the clock to this module when not in use. See Section 5.7.8, "System Clock Gating Control 1 Register (SCGC1)," for details.

10.1.4 Features

Features of the ADC module include:

- Linear successive approximation algorithm with up to 12-bit resolution
- Output Modes: Single-ended 12-bit, 10-bit and 8-bit modes
- Output in right-justified unsigned format for single-ended
- Single or continuous conversion (automatic return to idle after single conversion)
- Configurable sample time and conversion speed/power
- Conversion complete / Hardware average 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 with option to output the clock
- Selectable asynchronous hardware conversion trigger with hardware channel select
- Automatic compare with interrupt for less-than, greater-than or equal-to programmable value
- Temperature sensor
- Self-Calibration mode

10.1.5 Block Diagram

Figure 10-3 provides a block diagram of the ADC module.
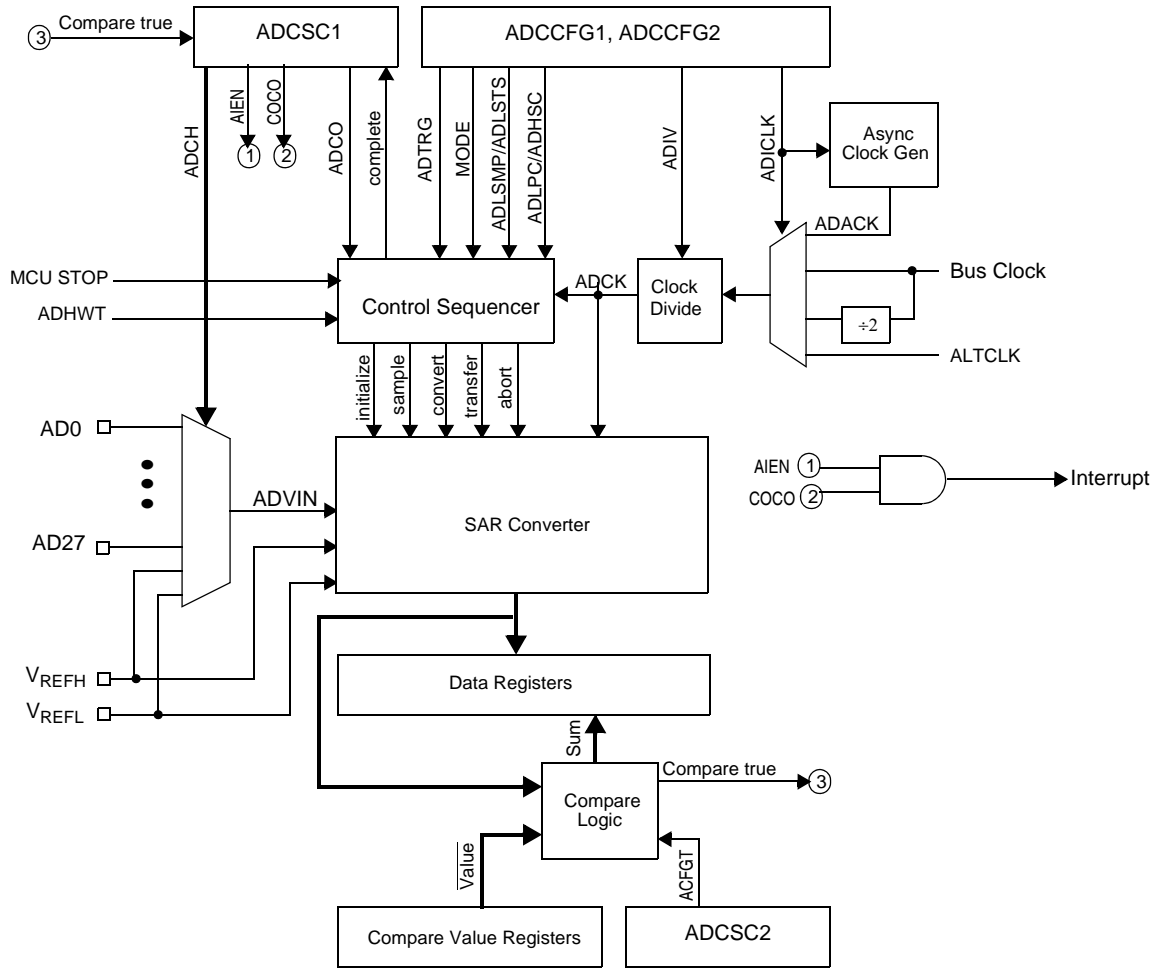
Figure 10-3. ADC Block Diagram

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 channel control registers, ADCSC1
- Configuration registers, ADCCFG1 and ADCCFG2
- Data result registers, ADCRH:ADCRL
- Compare value registers, ADCCVH, ADCCVL
- General status and control registers, ADCSC2 and ADCSC3
- Configuration registers, ADCCFG1 and ADCCFG2
- Pin enable registers, APCTL1, APCTL2, APCTL3, and APCTL4
- Offset Correction Registers, ADCOFSH and ADCOFSL
- Plus-input gain registers, ADCPGH and ADCPGL
- Minus-input gain registers, ADCMGH and ADCMGL

- Plus-side general calibration registers, ADCCLP0, ADCCLP1, ADCCLP2, ADCCLP3H, ADCCLP3L, ADCCLP4H, ADCCLP4L, ADCCLSP, ADDCLDP
- Minus-side general calibration registers, ADCCLM0, ADCCLM1, ADCCLM2, ADCCLM3H, ADCCLM3L, ADCCLM4H, ADCCLM4L, ADCCLSM, ADDCLDM

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)

	7	6	5	4	3	2	1	0
R	COCO	AIEN	0[1]			ADCH		
W								
Reset:	0	0	0	1	1	1	1	1

Figure 10-4. Status and Channel Control Register 1n (ADCSC1)

[1] This bit is reserved and must always be 0.

Table 10-3. ADCSC1:ADCSC1 Field Descriptions

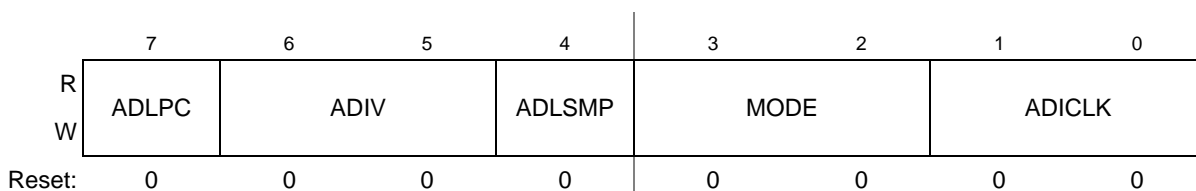| Field	Description
7 COCO	Conversion Complete Flag. The COCO flag is a read-only bit set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1), the COCO flag is set upon completion of a conversion only if the compare result 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
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	ADCH
00000–01111	AD0–15	00000–01111
10000–11011	AD16–27	10000–11011

Table 10-4. Input Channel Select (Continued)

ADCH	Input Select	ADCH
11100	Reserved	11100
11101	V_{REFH}	11101
11110	Reserved	11110
11111	Module disabled	11111

10.3.2 Configuration Register 1(ADCCFG1)

ADCCFG1 selects the mode of operation, clock source, clock divide, and configure for low power or long sample time.

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

Figure 10-5. Configuration Register (ADCCFG1)

Table 10-5. ADCCFG1 Register Field Descriptions

Field	Description
7 ADLPC	**Low-Power Configuration** - ADLPC controls the power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 Normal power 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 10-6 shows the available clock configurations.
4 ADLSMP	**Sample Time Configuration** - ADLSMP selects between different sample times based on the conversion mode selected. This bit 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. When ADLSMP=1, the Long Sample Time Select bits (ADLSTS[1:0]) can select the extent of the long sample time. 0 Short sample time 1 Long sample time (The ADLTS bits can select the extent of the long sample time)
3:2 MODE	**Conversion Mode Selection** - MODE bits are used to select between the ADC resolution mode. See Table 10-7.
1:0 ADICLK	**Input Clock Select** - ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 10-8.

Table 10-6. Clock Divide Select

ADIV	Divide Ratio	Clock Rate
00	1	Input clock
01	2	Input clock ÷ 2

Table 10-6. Clock Divide Select

ADIV	Divide Ratio	Clock Rate
10	4	Input clock ÷ 4
11	8	Input clock ÷ 8

Table 10-7. Conversion Modes

MODE	Conversion Mode Description
00	single-ended 8-bit conversion
01	single-ended 12-bit conversion
10	single-ended 10-bit conversion
11	Reserved

Table 10-8. Input Clock Select

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

10.3.3　Configuration Register 2 (ADCCFG2)

ADCCFG2 selects the special high speed configuration for very high speed conversions, and selects the long sample time duration during long sample mode.
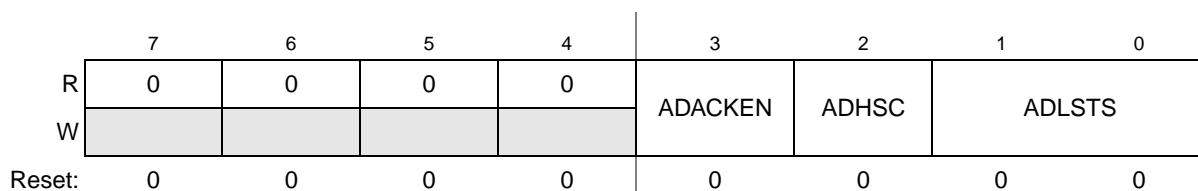
	7	6	5	4	3	2	1	0
R	0	0	0	0	ADACKEN	ADHSC	ADLSTS	
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-6. Configuration Register 2(ADCCFG2)

Table 10-9. ADCCFG2 Register Field Descriptions

Field	Description
3 ADACKEN	**Asynchronous clock output enable** - ADACKEN enables the ADC's asynchronous clock source and the clock source output regardless of the conversion and input clock select (ADICLK bits) status of the ADC. Based on MCU configuration the asynchronous clock may be used by other modules (see module introduction section). Setting this bit allows the clock to be used even while the ADC is idle or operating from a different clock source. Also, latency of initiating a single or first-continuous conversion with the asynchronous clock selected is reduced since the ADACK clock is already operational. 0 Asynchronous clock output disabled; Asynchronous clock only enabled if selected by ADICLK and a conversion is active 1 Asynchronous clock and clock output enabled regardless of the state of the ADC
2 ADHSC	**High Speed Configuration**- ADHSC configures the ADC for very high speed operation. The conversion sequence is altered (4 ADCK cycles added to the conversion time) to allow higher speed conversion clocks. 0 Normal conversion sequence selected 1 High speed conversion sequence selected (4 additional ADCK cycles to total conversion time)
1:0 ADLSTS [1:0]	**Long Sample Time Select** - ADLSTS selects between the extended sample times when long sample time is selected (ADLSMP=1). This allows 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. 00 Default longest sample time (20 extra ADCK cycles; 24 ADCK cycles total) 01 12 extra ADCK cycles; 16 ADCK cycles total sample time 10 6 extra ADCK cycles; 10 ADCK cycles total sample time 11 2 extra ADCK cycles; 6 ADCK cycles total sample time

10.3.4   Data Result Registers (ADCRH:ADCRL)

The Data Result Registers (ADCRH:ADCRL) contain the result of an ADC conversion of the channel selected by the status and channel control register (ADCSC1). 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 single-ended mode, there is no interlocking with ADCRL.

	7	6	5	4	3	2	1	0
R						**D[11:8]**		
W								
Reset:	0	0	0	0	0	0	0	0
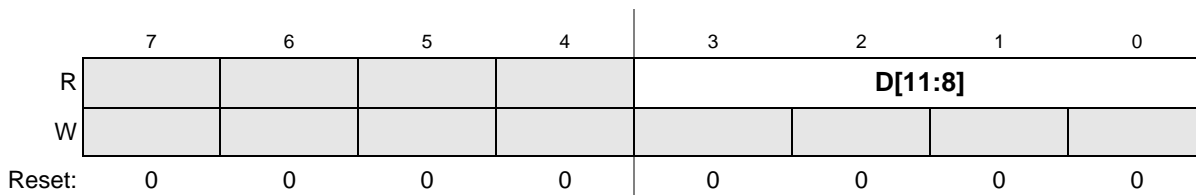
Figure 10-7. Data Result High Register (ADCRH)

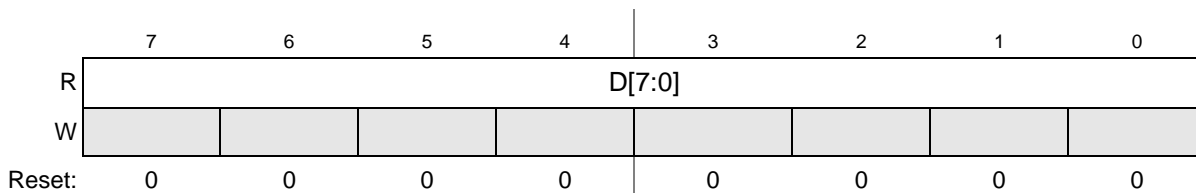|	7	6	5	4	3	2	1	0
R				D[7:0]				
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-8. Data Result Low Register (ADCRL)

ADCRH contains the upper bits of the result of a conversion based on the conversion mode. ADCRL contains the lower eight bits of the result of a conversion, or all eight bits of an 8-bit single-ended conversion. Unused bits in the ADCRH register are cleared. For example, when configured for 10-bit single-ended mode, D[11:10] are cleared.

Table 10-10 below describes the behavior of the data result registers in the different modes of operation.

Table 10-10. Data Result Register Description

Conversion Mode	DATA												Format
	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
12b single-ended	D	D	D	D	D	D	D	D	D	D	D	D	unsigned right justified
10b single-ended	0	0	D	D	D	D	D	D	D	D	D	D	unsigned right justified
8b single-ended	0	0	0	0	D	D	D	D	D	D	D	D	unsigned right justified

D: Data .

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. These bits are compared to the upper four bits of the result following a conversion in 12-bit mode when the compare function is enabled.

	7	6	5	4	3	2	1	0
R	0	0	0	0				
W						CV[11:8}		
Reset:	0	0	0	0	0	0	0	0
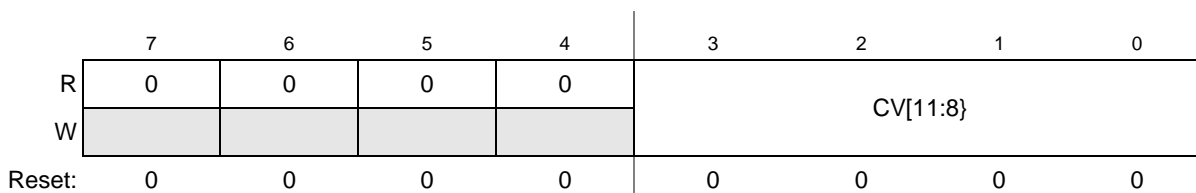
Figure 10-9. Compare Value High Register (ADCCVH)

In 10-bit mode, the ADCCVH register holds the upper two bits of the 10-bit compare value (ADCV9 – ADCV8). 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. Bits ADCV7:ADCV0 are compared to the lower 8 bits of the result following a conversion in 12-bit, 10-bit or 8-bit mode.
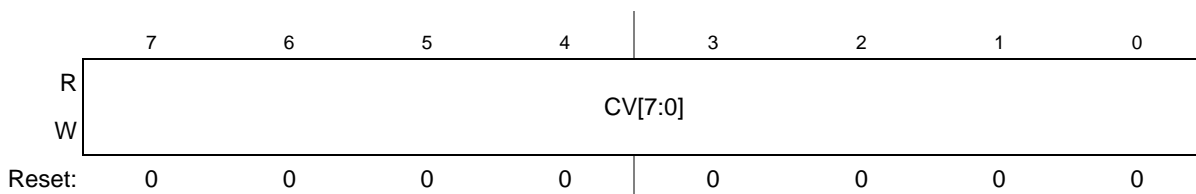
	7	6	5	4	3	2	1	0
R				CV[7:0]				
W								
Reset:	0	0	0	0	0	0	0	0

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

10.3.7   Status and Control Register 2 (ADCSC2)

The ADCSC2 register contains the conversion active, hardware/software trigger select, compare function and voltage reference select of the ADC module.
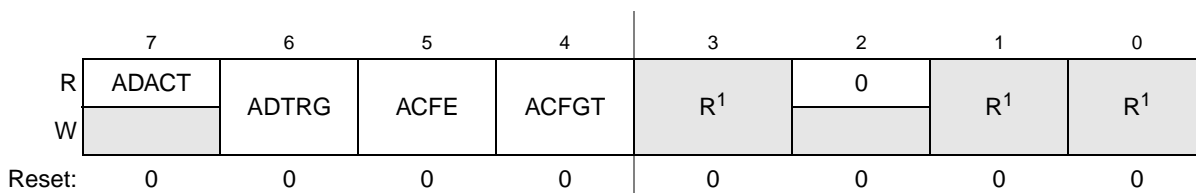
	7	6	5	4	3	2	1	0
R	ADACT	ADTRG	ACFE	ACFGT	R[1]	0	R[1]	R[1]
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-11. Status and Control Register 2 (ADCSC2)

[1] Bits 3, 1 and 0 are reserved bits that must always be written to 0.

Table 10-11. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	**Conversion Active** - ADACT indicates that a conversion or hardware averaging 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** - ADTRG selects the type of trigger used for initiating a conversion. Two types of trigger 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. Refer to Section 10.4.4.1 for more information on initiating conversions. 0 Software trigger selected 1 Hardware trigger selected
5 ACFE	**Compare Function Enable** - ACFE 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 level 1 Compare triggers when input is greater than or equal to compare level.

10.3.8 Status and Control Register 3 (ADCSC3)

The ADCSC3 register controls the calibration, continuous convert and hardware averaging functions of the ADC module.
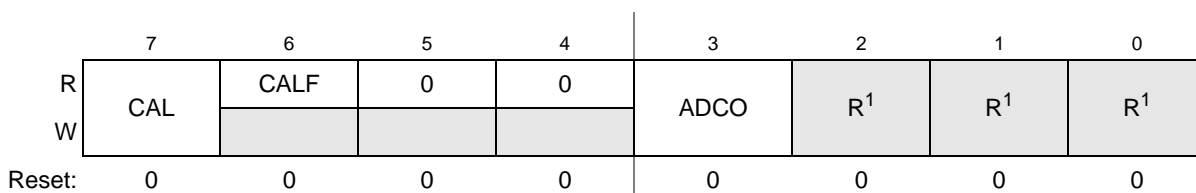
	7	6	5	4	3	2	1	0
R	CAL	CALF	0	0	ADCO	R^1	R^1	R^1
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-12. Status and Control Register 3 (ADCSC3)

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

Table 10-12. ADCSC3 Register Field Descriptions

Field	Description
7 CAL	**Calibration -** CAL begins the calibration sequence when set. This bit stays set while the calibration is in progress and is cleared when the calibration sequence is complete. The CALF bit must be checked to determine the result of the calibration sequence. Once started, the calibration routine cannot be interrupted by writes to the ADC registers or the results will be invalid and the CALF bit will set. Setting the CAL bit will abort any current conversion.
6 CALF	**Calibration Failed Flag -** CALF displays the result of the calibration sequence. The calibration sequence will fail if ADTRG = 1, any ADC register is written, or any stop mode is entered before the calibration sequence completes.The CALF bit is cleared by writing a 1 to this bit. 0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.
3 ADCO	**Continuous Conversion Enable** - ADCO enables continuous conversions.Refer to Section 10.4.4.1 for more information on initiating conversions. 0 One conversion or one set of conversions if the hardware average function is enabled (AVGE=1) after initiating a conversion. 1 Continuous conversions or sets of conversions if the hardware average function is enabled (AVGE=1) after initiating a conversion.

10.3.9 ADC Offset Correction Register (ADCOFSH:ADCOFSL)

The ADC Offset Correction Register (ADCOFSH:ADCOFSL) contains the user-selected or calibration-generated offset error correction value.

This register is a 2's complement, left justified, 16b value formed by the concatenation of:

- ADCOFSH
- ADCOFSL

The value in the offset correction registers (ADCOFSH:ADCOFSL) is subtracted from the conversion and the result is transferred into the result registers (ADCRH:ADCRL).

NOTE

If the result is above the maximum or below the minimum result value, it is forced to the appropriate limit for the current mode of operation.
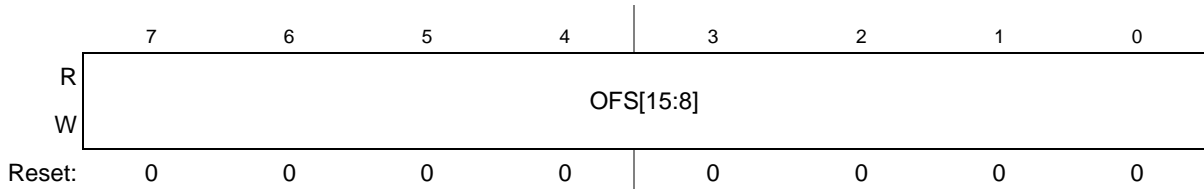
For additional information, please see Section 10.4.7.

	7	6	5	4	3	2	1	0
R				OFS[15:8]				
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-13. Offset Calibration High Register (ADCOFSH)

	7	6	5	4	3	2	1	0
R				OFS[7:0]				
W								
Reset:	0	0	0	0	0	1	0	0
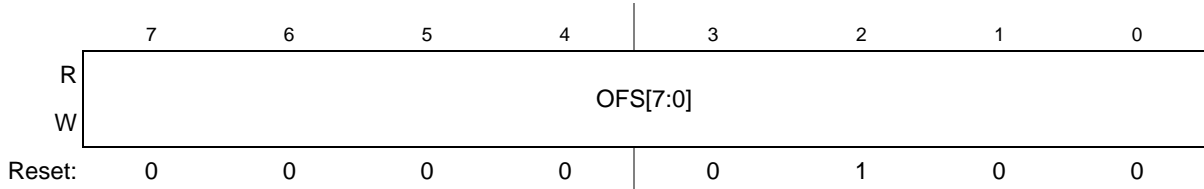
Figure 10-14. Offset Calibration Low Register (ADCOFSL)

10.3.10 ADC Plus-Side Gain Register (ADCPGH:ADCPGL)

The Plus-Side Gain Register (ADCPGH:ADCPGL) contains the gain error correction for the overall conversion in single-ended mode. ADCPGH:ADCPGL represent a 16 bit floating point number representation of the gain adjustment factor, with the decimal point fixed between ADPG15 and ADPG14. This register must be written by the user with the value described in the calibration procedure or the gain error specifications may not be met.

	7	6	5	4	3	2	1	0
R				PG[15:8]				
W								
Reset:	1	0	0	0	0	0	1	0
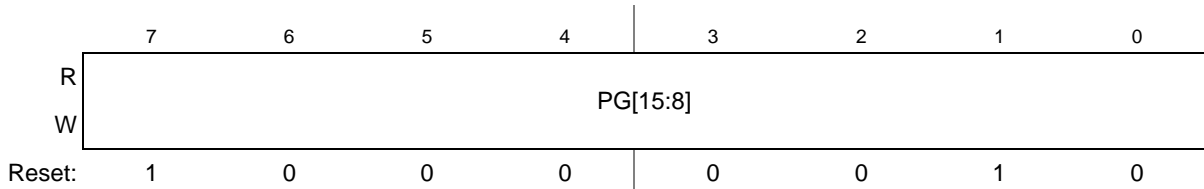
Figure 10-15. ADC Plus Gain High Register (ADCPGH)

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

Figure 10-16. ADC Plus Gain Low Register (ADCPGL)

10.3.11 ADC Minus-Side Gain Register (ADCMGH:ADCMGL)

The Minus-Side Gain Register (ADCMGH:ADCMGL) represent a 16-bit floating point number representation of the gain adjustment factor, with the decimal point fixed between ADPG15 and ADPG14.

This register must be written by the user with the value described in the calibration procedure or the gain error specifications may not be met.
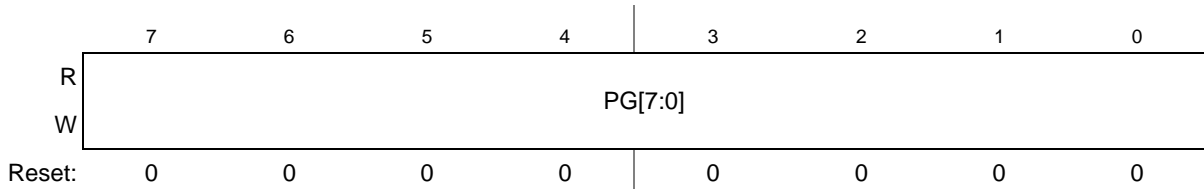
	7	6	5	4	3	2	1	0
R				MG[15:8]				
W								
Reset:	1	0	0	0	0	0	1	0
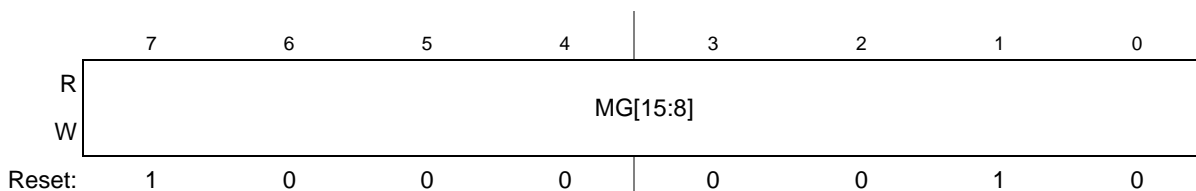
Figure 10-17. ADC Gain Register (ADCMGH)

	7	6	5	4	3	2	1	0
R				MG[7:0]				
W								
Reset:	0	0	0	0	0	0	0	0
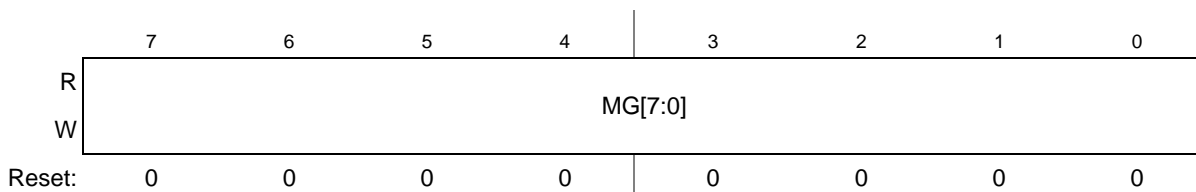
Figure 10-18. ADC Gain Register (ADCMGL)

10.3.12 ADC Plus-Side General Calibration Value Registers (ADCCLPx)

The Plus-Side General Calibration Value Registers (ADCCLPx) contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLP0[5:0], CLP1[6:0], CLP2[7:0], CLP3[8:0], CLP4[9:0], CLPS[5:0], and CLPD[5:0]. ADCCLPx are automatically set once the self calibration sequence is done (CAL is cleared). If these registers are written by the user after calibration, the linearity error specifications may not be met.

	7	6	5	4	3	2	1	0
R	0	0			CLPD[5:0]			
W								
Reset:	0	0	0	0	1	0	1	0
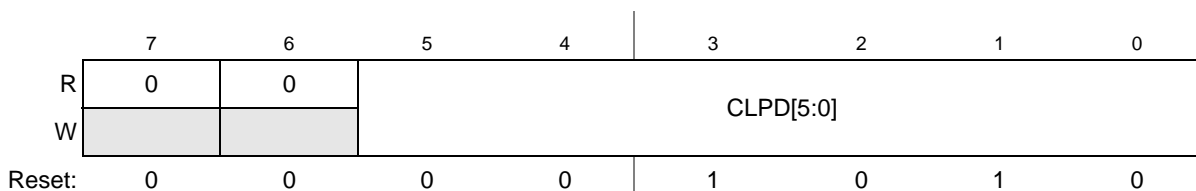
Figure 10-19. Plus-Side General Calibration Register (ADCCLPD)

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

Figure 10-20. Plus-Side General Calibration Register (ADCCLPS)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	CLP4[9:8]	
W								
Reset:	0	0	0	0	0	0	1	0
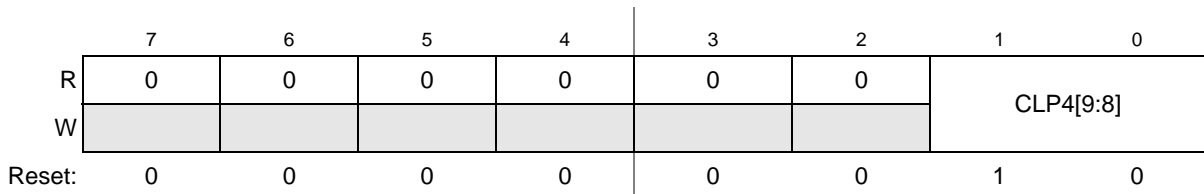
Figure 10-21. Plus-Side General Calibration Register (ADCCLP4H)

	7	6	5	4	3	2	1	0
R	CLP4[7:0]							
W								
Reset:	0	0	0	0	0	0	0	0
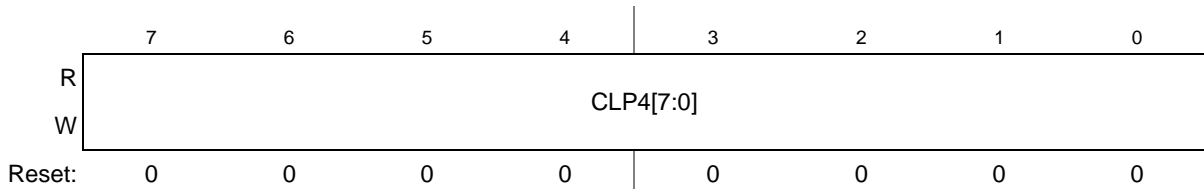
Figure 10-22. Plus-Side General Calibration Register (ADCCLP4L)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	CLP3[8]
W								
Reset:	0	0	0	0	0	0	0	1
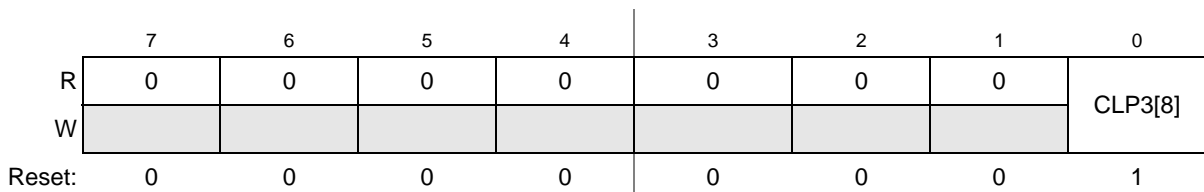
Figure 10-23. Plus-Side General Calibration Register (ADCCLP3H)

	7	6	5	4	3	2	1	0
R	CLP3[7:0]							
W								
Reset:	0	0	0	0	0	0	0	0
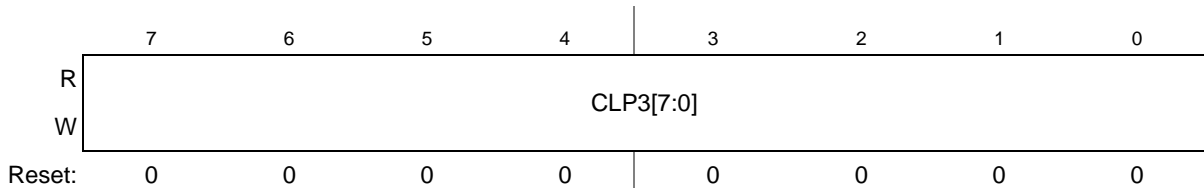
Figure 10-24. Plus-Side General Calibration Register (ADCCLP3L)

	7	6	5	4	3	2	1	0
R	CLP2[7:0]							
W								
Reset:	1	0	0	0	0	0	0	0

Figure 10-25. Plus-Side General Calibration Register (ADCCLP2)

	7	6	5	4	3	2	1	0
R	0	CLP1[6:0]						
W								
Reset:	0	1	0	0	0	0	0	0
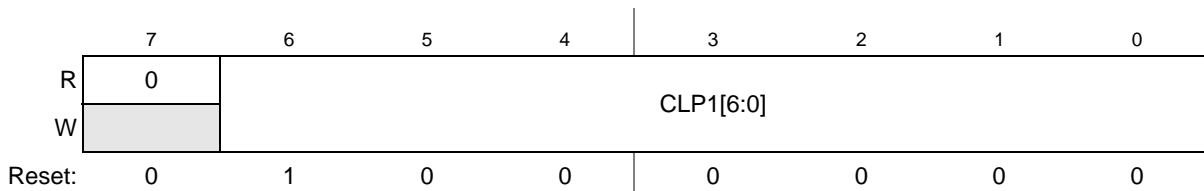
Figure 10-26. Plus-Side General Calibration Register (ADCCLP1)

	7	6	5	4	3	2	1	0
R	0	0			CLP0[5:0]			
W								
Reset:	0	0	1	0	0	0	0	0

Figure 10-27. Plus-Side General Calibration Register (ADCCLP0)
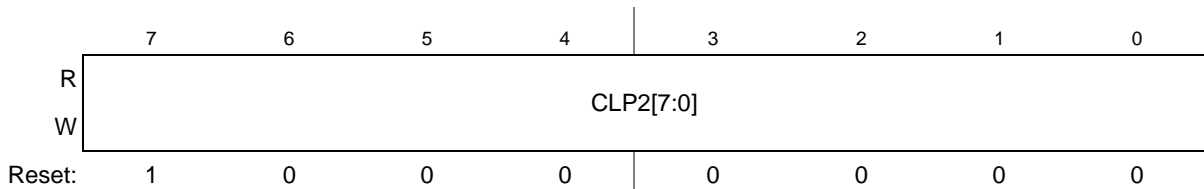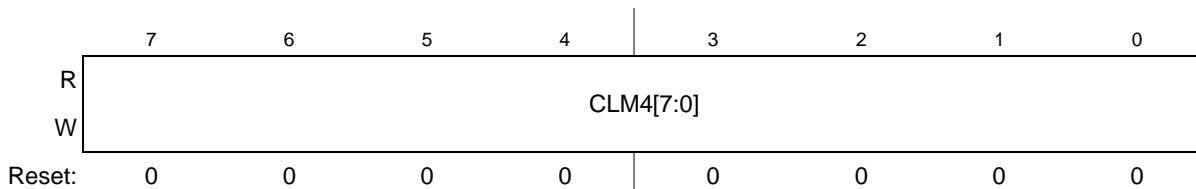
10.3.13 ADC Minus-Side General Calibration Value Registers (ADCCLMx)

ADCCLMx contain calibration information that is generated by the calibration function. These registers contain seven calibration values of varying widths: CLM0[5:0], CLM1[6:0], CLM2[7:0], CLM3[8:0], CLM4[9:0], CLMS[5:0], and CLMD[5:0]. ADCCLMx are automatically set once the self calibration sequence is done (CAL is cleared). If these registers are written by the user after calibration, the linearity error specifications may not be met.

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

Figure 10-28. Minus-Side General Calibration Register (ADCCLMD)

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

Figure 10-29. Minus-Side General Calibration Register (ADCCLMS)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	CLM4[9:8]	
W								
Reset:	0	0	0	0	0	0	1	0
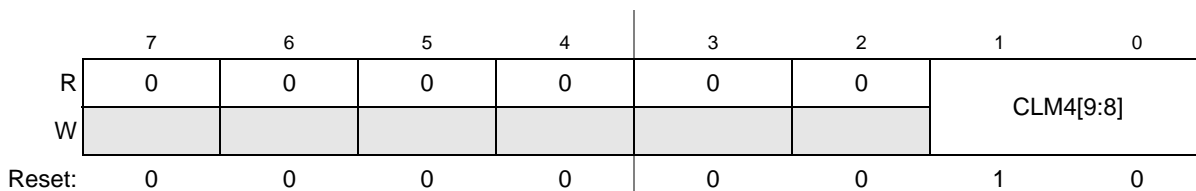
Figure 10-30. Minus-Side General Calibration Register (ADCCLM4H)

	7	6	5	4	3	2	1	0
R				CLM4[7:0]				
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-31. Minus-Side General Calibration Register (ADCCLM4L)

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	CLM3[8]
W								
Reset:	0	0	0	0	0	0	0	1
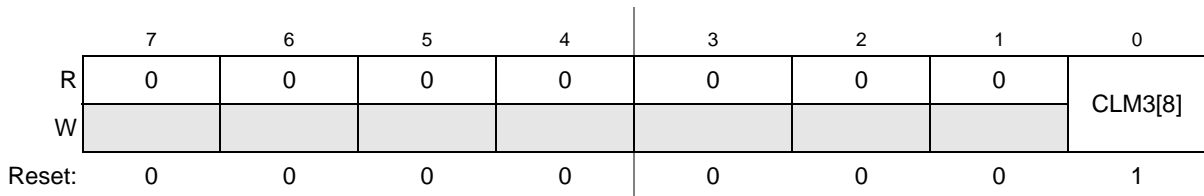
Figure 10-32. Minus-Side General Calibration Register (ADCCLM3H)

	7	6	5	4	3	2	1	0
R				CLM3[7:0]				
W								
Reset:	0	0	0	0	0	0	0	0
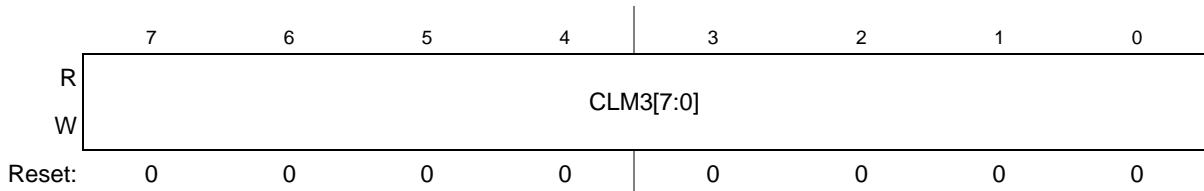
Figure 10-33. Minus-Side General Calibration Register (ADCCLM3L)

	7	6	5	4	3	2	1	0
R				CLM2[7:0]				
W								
Reset:	1	0	0	0	0	0	0	0
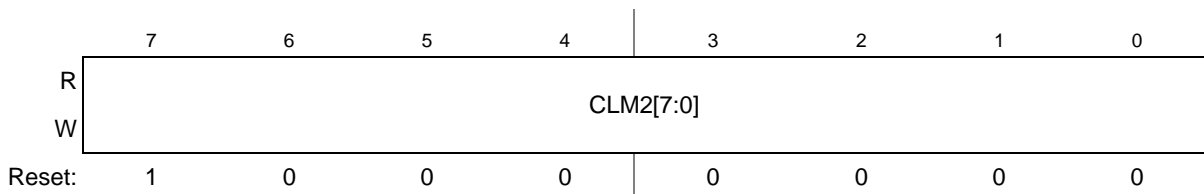
Figure 10-34. Minus-Side General Calibration Register (ADCCLM2)

	7	6	5	4	3	2	1	0
R	0				CLM1[6:0]			
W								
Reset:	0	1	0	0	0	0	0	0
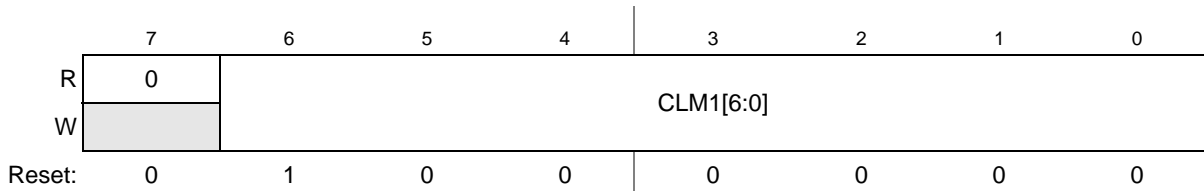
Figure 10-35. Minus-Side General Calibration Register (ADCCLM1)

	7	6	5	4	3	2	1	0
R	0	0			CLM0[5:0]			
W								
Reset:	0	0	1	0	0	0	0	0

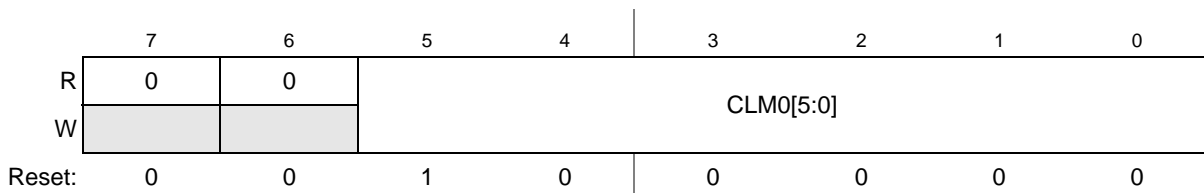**Figure 10-36. Minus-Side General Calibration Register (ADCCLM0)**

10.3.14 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.
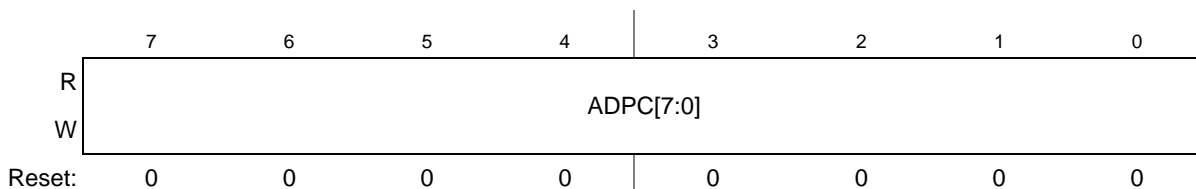
	7	6	5	4	3	2	1	0
R					ADPC[7:0]			
W								
Reset:	0	0	0	0	0	0	0	0

Figure 10-37. Pin Control 1 Register (APCTL1)

Table 10-13. 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

10.3.15 Pin Control 2 Register (APCTL2)

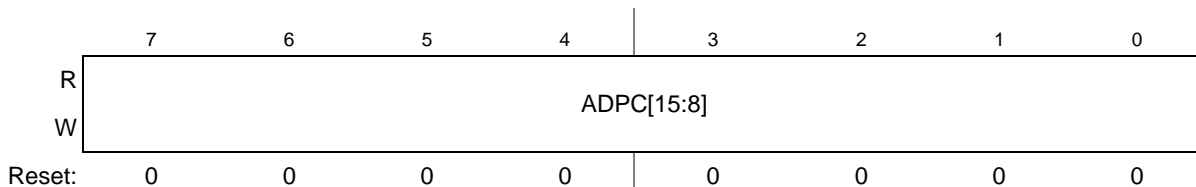APCTL2 controls channels 8–15 of the ADC module.

	7	6	5	4	3	2	1	0
R								
W				ADPC[15:8]				
Reset:	0	0	0	0	0	0	0	0

Figure 10-38. Pin Control 2 Register (APCTL2)

Table 10-14. 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

10.3.16 Pin Control 3 Register (APCTL3)

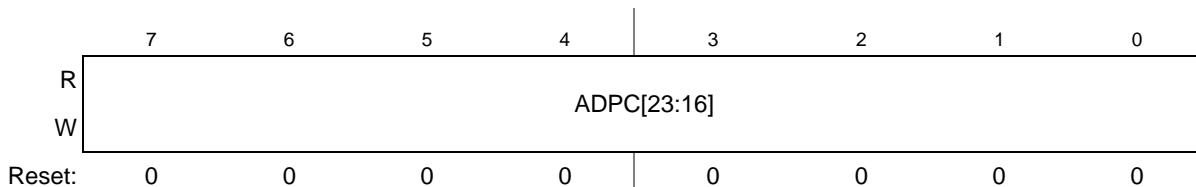APCTL3 controls channels 23–16 of the ADC module.

	7	6	5	4	3	2	1	0
R								
W				ADPC[23:16]				
Reset:	0	0	0	0	0	0	0	0

Figure 10-39. Pin Control 3 Register (APCTL3)

Table 10-15. 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, stop2 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 and the asynchronous clock output enable is disabled (ADACKEN=0), the module is in its lowest power state. The ADC can perform an analog-to-digital conversion on any of the software selectable channels. All modes perform conversion by a successive approximation algorithm.

To meet accuracy specifications, the ADC module must be calibrated using the on chip calibration function. Calibration is recommended to be done after any reset. See Section 10.4.6 for details on how to perform calibration.

When the conversion is completed, the result is placed in the data registers (ADCRH and ADCRL). The conversion complete flag (COCO) is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled (AIENn=1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of the compare value 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 with using the ADIV bits.
- 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. Conversions are possible using ADACK as the input clock source while the MCU is in stop3 mode. Refer to Section 10.4.4.4 for more information.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC may 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 (APCTL1, APCTL2, APCTL3, and APCTL4) 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 1's) if software triggered operation is selected (ADTRG=0).
- Following a hardware trigger (ADHWT) event if hardware triggered operation is selected (ADTRG=1)
- Following the transfer of the result to the data registers when continuous conversion is enabled (ADCO=1).

If continuous conversions are enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation (ADTRG=0), continuous conversions begin after ADCSC1 is written and continue until aborted. In hardware triggered operation (ADTRG=1), 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. If the compare functions are disabled, this is indicated by the setting of COCO. If the compare function is enabled, COCO sets and conversion result data is transferred only if the compare condition is true. An interrupt is generated if AIEN is high at the time that COCO is set.In all modes except 8-bit single-ended conversions, 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 (the ADCRH register has been read but the ADCRL register has not). When blocking is active, the conversion result 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 conversion result data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

NOTE

If continuous conversions are enabled, the blocking mechanism could result in the loss of data occurring at specific timepoints. To avoid this issue, the data must be read in fewer cycles than an ADC conversion time, accounting for interrupt or software polling loop latency.

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:

- Writing ADCSC1 while ADCSC1 is actively controlling a conversion aborts the current conversion. In software trigger mode (ADTRG=0), a write to ADCSC1 initiates a new conversion (if the ADCH bits are equal to a value other than all 1s).

- A write to any ADC register besides the ADCSC1 register occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.

- The MCU is reset or enters stop2 mode.

- The MCU enters stop3 mode with ADACK not enabled.

When a conversion is aborted, the contents of the data registers, ADCRH and ADCRL are not altered. The data registers continue to be the values transferred after the completion of the last successful conversion. If the conversion was aborted by a reset or stop2, ADCRH: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 but the asynchronous clock output is disabled (ADACKEN=0), the ADACK clock generator will also remain in its idle state (disabled) until a conversion is initiated. If the asynchronous clock output is enabled (ADACKEN=1), it will remain active regardless of the state of the ADC or the MCU power mode.

Power consumption when the ADC is 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 upon: the sample time (as determined by ADLSMP and ADLSTS bits), the MCU bus frequency, the conversion mode (as determined by MODE bits), the high speed configuration (ADHSC bit), and the frequency of the conversion clock (f_{ADCK}).

The ADHSC bit is used to configure a higher clock input frequency. This will allow faster overall conversion times. In order to meet internal A/D converter timing requirements the ADHSC bit adds additional ADCK cycles. Conversions with ADHSC = 1 take four more ADCK cycles. ADHSC should be used when the ADCLK exceeds the limit for ADHSC = 0.

After the module becomes active, sampling of the input begins. ADLSMP and ADLSTS select between sample times based on the conversion mode that is selected. 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). 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. The maximum total conversion time for all configurations is summarized in the equation that follows. Refer to Table 10-16 through Table 10-19 for the variables referenced in the equation.

Conversion Time Equation *Eqn. 10-2*

$$ConversionTime = SFCAdder + (BCT + LSTAdder + HSCAdder)$$

Table 10-16. Single or First Continuous Time Adder (SFCAdder)

ADLSMP	ADACKEN	ADICLK	Single or First Continuous Time Adder (SFCAdder)
1	x	0x, 10	3 ADCK cycles + 5 bus clock cycles
1	1	11	3 ADCK cycles + 5 bus clock cycles[1]
1	0	11	5μs + 3 ADCK cycles + 5 bus clock cycles
0	x	0x, 10	5 ADCK cycles + 5 bus clock cycles
0	1	11	5 ADCK cycles + 5 bus clock cycles[1]
0	0	11	5μs + 5 ADCK cycles + 5 bus clock cycles

[1] ADACKEN must be 1 for at least 5us prior to the conversion is initiated to achieve this time

Table 10-17. Base Conversion Time (BCT)

Mode	Base Conversion Time (BCT)
8b se	17 ADCK cycles

Table 10-17. Base Conversion Time (BCT)

Mode	Base Conversion Time (BCT)
10b s.e.	20 ADCK cycles
12b s.e.	20 ADCK cycles

Table 10-18. Long Sample Time Adder (LSTAdder)

ADLSMP	ADLSTS	Long Sample Time Adder (LSTAdder)
0	xx	0 ADCK cycles
1	00	20 ADCK cycles
1	01	12 ADCK cycles
1	10	6 ADCK cycles
1	11	2 ADCK cycles

Table 10-19. High-Speed Conversion time Adder (HSCAdder)

ADHSC	High Speed Conversion Time Adder (HSCAdder)
0	0 ADCK cycles
1	4 ADCK cycles

NOTE

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

10.4.4.6   Conversion Time Examples

The following examples use Equation 10-2 and the information provided in Table 10-16 through Table 10-19.

10.4.4.6.1   Typical conversion time configuration

A typical configuration for ADC conversion is: 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, long sample time disabled and high speed conversion disabled. The conversion time for a single conversion is calculated by

using Equation 10-2 and the information provided in Table 10-16 through Table 10-19. The following table lists the variables of Equation 10-2.

Table 10-20. Typical Conversion Time

Variable	Time
SFCAdder	5 ADCK cycles + 5 bus clock cycles
BCT	20 ADCK cycles
LSTAdder	0
HSCAdder	0

The resulting conversion time is generated using the parameters listed in Table 10-20. So for Bus clock equal to 8 MHz and ADCK equal to 8 MHz the resulting conversion time is 3.75 us.

10.4.5 Automatic Compare Function

The compare function can be configured to check if the result is less than or greater-than-or-equal-to a compare value. The compare mode is determined by ACFGT. After the input is sampled and converted, the compare value (ADCCVH:ADCCVL) is used as described in Table 10-21.

Table 10-21. Compare Modes

ACFGT	Function	Compare Mode Description
0	Less than threshold	Compare true if the result is less than the ADCCV registers.
1	Greater than or equal to threshold	Compare true if the result is greater than or equal to ADCCV registers.

If the condition selected evaluates true, COCO is set.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCO is not set and the conversion result data will not be transferred to the result register. An ADC interrupt is generated upon the setting of COCO if the respective 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 Calibration Function

The ADC contains a self-calibration function that is required to achieve the specified accuracy. Calibration must be run or valid calibration values written after any reset and before a conversion is initiated. The calibration function sets the offset calibration value and the plus-side and minus-side calibration values.

The offset calibration value is automatically stored in the ADC Offset Correction Registers (ADCOFSH and ADCOFSL) and the plus-side and minus-side calibration values are automatically stored in the ADC Plus-Side and Minus-Side Calibration registers (CLPD, CLPS, CLP4, CLP3, CLP2, CLP1, CLP0 and CLMD, CLMS, CLP4, CLM3, CLM2, CLM1, CLM0). The user must configure the ADC correctly prior to calibration, and must generate the plus-side and minus-side gain calibration results and store them in the ADC GAIN registers (ADCPGH and ADCPGL) after the calibration function completes.

Prior to calibration, the user must configure the ADC's clock source and frequency, low power configuration, voltage reference selection, sample time and the high speed configuration according to the application's clock source availability and needs. If the application uses the ADC in a wide variety of configurations, the configuration for which the highest accuracy is required should be selected, or multiple calibrations can be done for the different configurations. The input channel, conversion mode continuous function, compare function, and resolution mode are all ignored during the calibration function.

To initiate calibration, the user sets the CAL bit and the calibration will automatically begin if the ADTRG bit = 0. If ADTRG = 1, the CAL bit will not get set and the calibration fail flag (CALF) will be set. While calibration is active, no ADC register can be written and no stop mode may be entered or the calibration routine will be aborted causing the CAL bit to clear and the CALF bit to set. At the end of a calibration sequence the COCO bit of the ADSC1A register will be set. The AIEN1 bit can be used to allow an interrupt to occur at the end of a calibration sequence. If at the end of calibration routine the CALF bit is not set, the automatic calibration routine completed successfully.

To complete calibration, the user must generate the gain calibration values using the following procedure:

- Initialize (clear) a 16b variable in RAM.
- Add the following plus-side calibration results CLP0, CLP1, CLP2, CLP3, CLP4, and CLPS to the variable.
- Divide the variable by two.
- Set the MSB of the variable.
- The previous two steps can be achieved by setting the carry bit, rotating-right through the carry bit on the high byte and again on the low byte.
- Store the value in the plus-side gain calibration registers ADCPGH and ADCPGL.
- Repeat procedure for the minus-side gain calibration value.

When complete the user may reconfigure and use the ADC as desired. A second calibration may also be performed if desired by clearing and again setting the CAL bit.

Overall the calibration routine may take as many as 14000 ADCK cycles and 100 bus cycles, depending on the results and the clock source chosen. For an 8 MHz clock source this is about 1.7 msec. To reduce this latency, the calibration values (offset, plus- and minus-side gain, and plus- and minus-side calibration values) may be stored in flash after an initial calibration and recovered prior to the first ADC conversion.

This should reduce the calibration latency to 20 register store operations on all subsequent power, reset, or stop2 mode recoveries.

10.4.7 User-Defined Offset Function

The ADC Offset Correction Register (ADCOFSH:ADCOFSL) contains the user selected or calibration generated offset error correction value. This register is a 2's complement, left justified, 16b value formed by the concatenation of ADCOFSH and ADCOFSL. The value in the offset correction registers (ADCOFSH:ADCOFSL) is subtracted from the conversion and the result is transferred into the result registers (ADCRH:ADCRL). If the result is above the maximum or below the minimum result value, it is forced to the appropriate limit for the current mode of operation.

The formatting of the ADC Offset Correction Register is different from the Data Result Registers (ADCRH:ADCRL) to preserve the resolution of the calibration value regardless of the conversion mode selected. Lower order bits are ignored in lower resolution modes. For example, in 8b single-ended mode, the bits OFS[14:7] are subtracted from D[7:0]; bit OFS[15] indicates the sign (negative numbers are effectively added to the result) and bits OFS[6:0] are ignored. ADCOFSH is automatically set according to calibration requirements once the self calibration sequence is done (CAL is cleared). Write ADCOFSH:ADCOFSL to override the calibration result if desired. If the Offset Correction Register is written to a value that is different from the calibration value, the ADC error specifications may not be met. It is recommended that the value generated by the calibration function be stored in memory before overwriting with a specified value.

NOTE

There is an effective limit to the values of Offset that can be set. If the magnitude of the offset is too great, the results of the conversions cap off at the limits.

Use the offset calibration function to remove application offsets or DC bias values. The Offset Correction Registers ADCOFSH and ADCOFSL may be written with a number in 2's complement format and this offset will be subtracted from the result (or hardware averaged value). To add an offset, store the negative offset in 2's complement format and the effect will be an addition. An offset correction that results in an out-of-range value will be forced to the minimum or maximum value (the minimum value for single-ended conversions is 0x0000).

To preserve accuracy, the calibrated offset value initially stored in the ADCOFS registers must be added to the user defined offset. For applications which may change the offset repeatedly during operation, it is recommended to store the initial offset calibration value in flash so that it can be recovered and added to any user offset adjustment value -nd the sum stored in the ADCOFS registers.

10.4.8 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 10-3 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{TEMP} - V_{TEMP25}) \div m)$$

<div align="right">*Eqn. 10-3*</div>

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, you read the temperature sensor channel, calculate V_{TEMP}, and compare to V_{TEMP25}. If V_{TEMP} is greater than V_{TEMP25}, the cold slope value is applied in Equation 10-3. If V_{TEMP} is less than V_{TEMP25}, the hot slope value is applied in Equation 10-3.

For more information on using the temperature sensor, consult AN3031.

10.4.9   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.

If the compare and hardware averaging functions are disabled, a conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from wait mode if the respective ADC interrupt is enabled (AIEN=1). If the hardware averaging function is enabled the COCO will set (and generate an interrupt if enabled) when the selected number of conversions are complete. If the compare function is enabled the COCO will set (and generate an interrupt if enabled) only if the compare conditions are met. If a single conversion is selected and the compare trigger is not met, the ADC will return to its idle state and cannot wake the MCU from wait mode unless a new conversion is initiated by the hardware trigger.

10.4.10   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.

10.4.10.1   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 the ADC registers, including ADCRH and ADCRL, are unaffected by stop3 mode. After exiting from stop3 mode, a software or hardware trigger is required to resume conversions.

10.4.10.2 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.

If the compare and hardware averaging functions are disabled, a conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from stop3 mode if the respective ADC interrupt is enabled (AIEN = 1). The result register will contain the data from the first completed conversion that occurred during stop3 mode. If the hardware averaging function is enabled the COCO will set (and generate an interrupt if enabled) when the selected number of conversions are complete. If the compare function is enabled the COCO will set (and generate an interrupt if enabled) only if the compare conditions are met. If a single conversion is selected and the compare is not true, the ADC will return to its idle state and cannot wake the MCU from stop3 mode unless a new conversion is initiated by another hardware trigger.

> **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 conversion result data transfer blocking mechanism (discussed in Section 10.4.4.2, "Completing Conversions) is cleared when entering stop3 and continuing ADC conversions.

10.4.11 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 single-ended resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to Table 10-6, Table 10-7, and Table 10-8 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. Calibrate the ADC by following the calibration instructions in Section 10.4.6, "Calibration Function.

2. 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.

3. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.

4. Update status and control register 3 (ADSC3) to select whether conversions will be continuous or completed only once (ADCO) and to select whether to perform hardware averaging.

5. Update status and control register (ADCSC1) to select 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

> ```
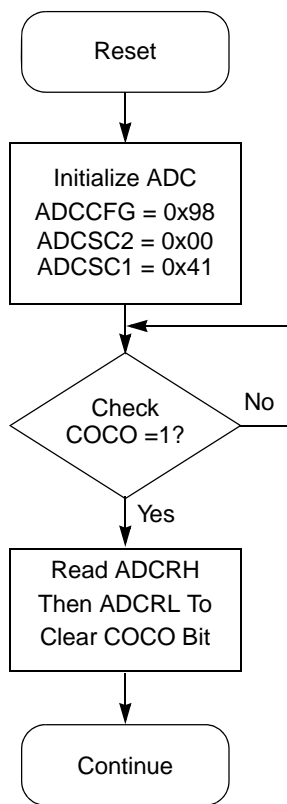> All other AD pins remain general purpose I/O pins.
> ```

Figure 10-40. 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 Voltage Reference 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.

10.6.1.3 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.

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 and changing the ADLSTS bits (to increase the sample window) 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:

$$1 \text{ lsb} = (V_{REFH} - V_{REFL}) / 2^N$$ *Eqn. 10-4*

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 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
Cyclic Redundancy Check (S08CRCV3)

11.1   Introduction

The CRC block provides hardware acceleration for CRC16-CCITT compliancy with $x^{16} + x^{12} + x^5 + 1$ polynomial. It is implemented on the 8-bit peripheral bus, and provides a very simple programming interface of only two 8-bit registers. This peripheral is available in both RUN and LPRUN modes.

NOTE

For details on low-power mode operation, refer to Table 3-4 in Chapter 3, "Modes of Operation".
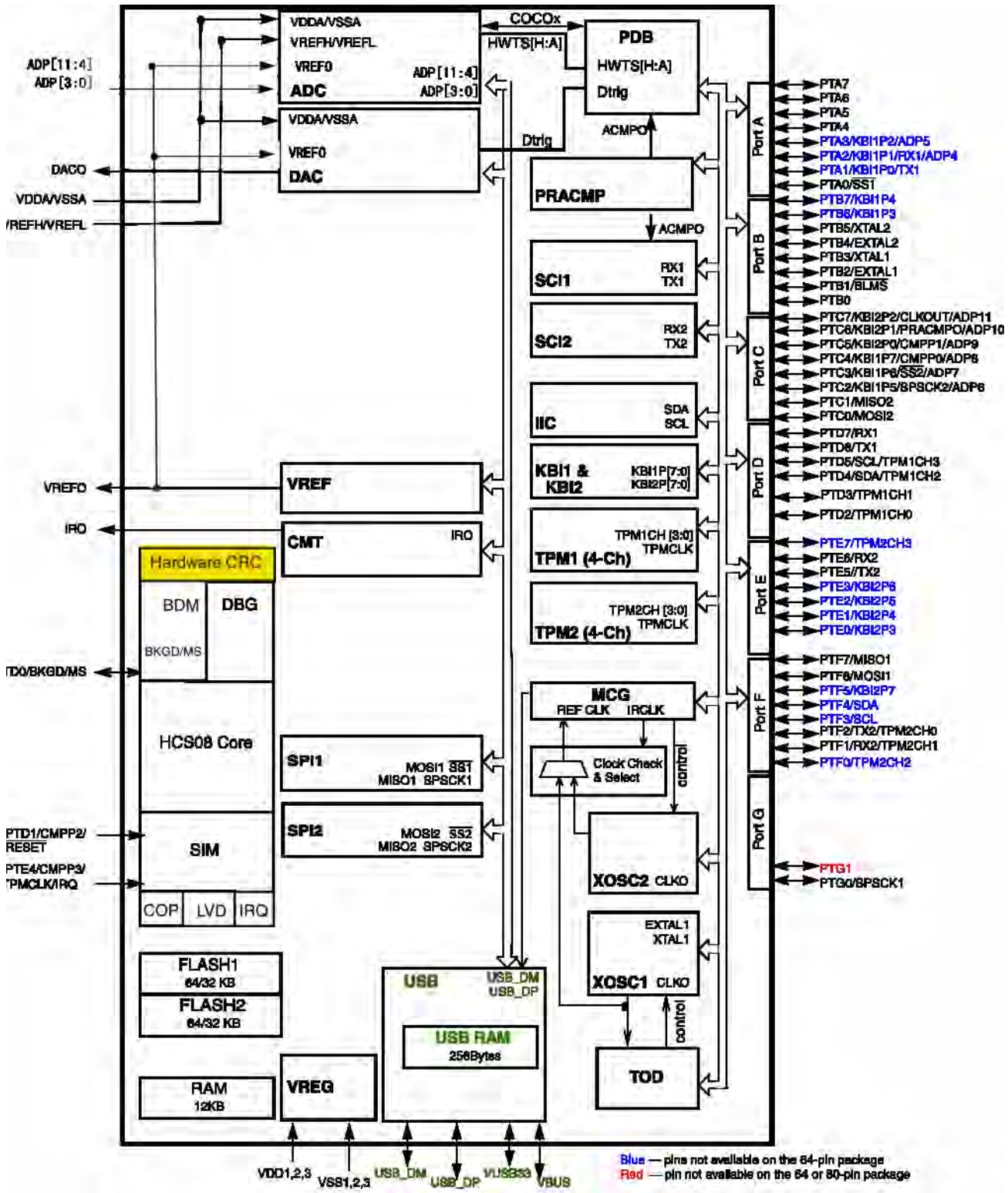
Figure 11-1. MC9S08JE128 series Block Diagram

11.1.1 Features

Features of the CRC module include:

- Hardware CRC generator circuit using 16-bit shift register
- CRC16-CCITT compliancy with $x^{16} + x^{12} + x^5 + 1$ polynomial
- Error detection for all single, double, odd, and most multi-bit errors
- Programmable initial seed value
- High-speed CRC calculation
- Optional feature to transpose input data and CRC result via transpose register, required on applications where bytes are in LSb (Least Significant bit) format.

11.1.2 Modes of Operation

This section defines the CRC operation in run, wait, and stop modes.

- Run Mode — This is the basic mode of operation.
- Wait Mode — The CRC module is operational.
- Stop 2 Modes —The CRC module is not functional in these modes and will be put into its reset state upon recovery from stop.
- Stop 3 Mode — In this mode, the CRC module will go into a low-power standby state. Any CRC calculations in progress will stop and resume after the CPU goes into run mode.

11.1.3   Block Diagram

Figure 11-2 provides a block diagram of the CRC module.
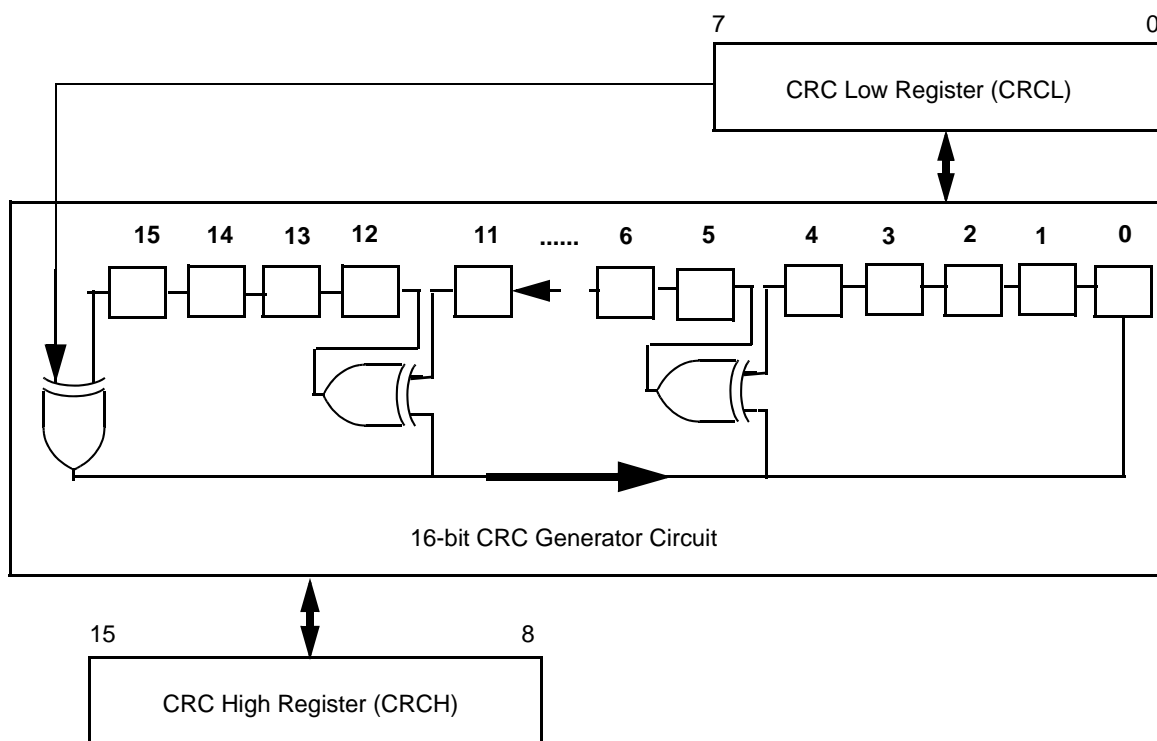
Figure 11-2. Cyclic Redundancy Check (CRC) Module Block Diagram

11.2   External Signal Description

There are no CRC signals that connect off chip.

11.3   Register Definition

11.3.1   Memory Map

Table 11-1. CRC Register Summary

Name		7	6	5	4	3	2	1	0
CRCH (offset=0)	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								

Table 11-1. CRC Register Summary

Name		7	6	5	4	3	2	1	0
CRCL (offset=1)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
TRANSPOSE (offset=2)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								

NOTE

The transpose feature (offset=2) is optional. If this feature is required by the MCU, parameter TRANSPOSE_FEATURE should be 1'b1. Otherwise, it should be 1'b0 (default).

NOTE

Offsets 4,5,6 and 7 are also mapped onto the CRCL register. This is an *alias* only used on CF1Core (version 1 of ColdFire core) and should be ignored for HCS08 cores.

11.3.2   Register Descriptions

The CRC module includes:

- A 16-bit CRC result and seed register (CRCH:CRCL)
- An 8-bit transpose register to convert from LSb to MSb format (or vice-versa) when required by the application

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

11.3.2.1   CRC High Register (CRCH)

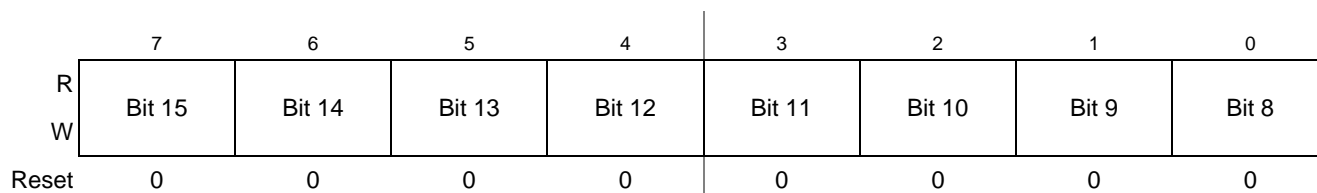|	7	6	5	4	3	2	1	0
R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
W								
Reset	0	0	0	0	0	0	0	0

Figure 11-3. CRC High Register (CRCH)

Table 11-2. Register Field Descriptions

Field	Description
7:0 CRCH	**CRCH** — This is the high byte of the 16-bit CRC register. A write to CRCH will load the high byte of the initial 16-bit seed value directly into bits 15-8 of the shift register in the CRC generator. The CRC generator will then expect the low byte of the seed value to be written to CRCL and loaded directly into bits 7-0 of the shift register. Once both seed bytes written to CRCH:CRCL have been loaded into the CRC generator, and a byte of data has been written to CRCL, the shift register will begin shifting. A read of CRCH will read bits 15-8 of the current CRC calculation result directly out of the shift register in the CRC generator.

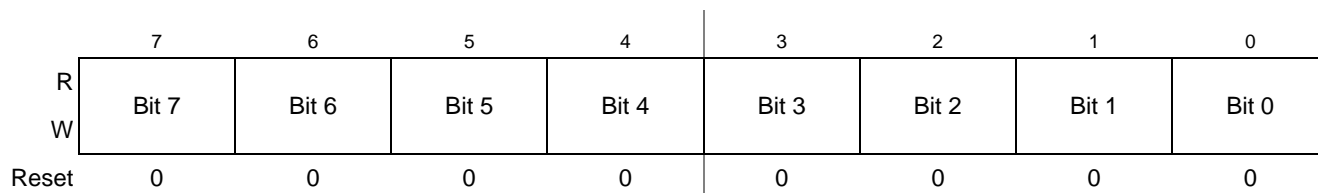## 11.3.2.2   CRC Low Register (CRCL)

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

Figure 11-4. CRC Low Register (CRCL)

Table 11-3. Register Field Descriptions

Field	Description
7:0 CRCL	**CRCL** — This is the low byte of the 16-bit CRC register. Normally, a write to CRCL will cause the CRC generator to begin clocking through the 16-bit CRC generator. As a special case, if a write to CRCH has occurred previously, a subsequent write to CRCL will load the value in the register as the low byte of a 16-bit seed value directly into bits 7-0 of the shift register in the CRC generator. A read of CRCL will read bits 7-0 of the current CRC calculation result directly out of the shift register in the CRC generator.

11.3.2.3   Transpose Register (TRANSPOSE)

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset	0	0	0	0	0	0	0	0
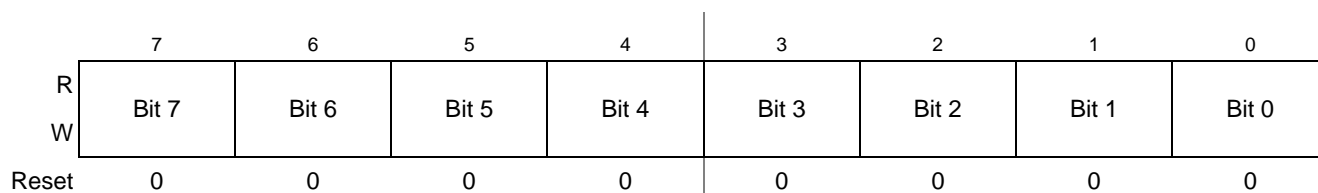
Figure 11-5. CRC High Register (CRCH)

Table 11-4. Register Field Descriptions

Field	Description
7:0 TRANSPOSE	**TRANSPOSE** -- This register is used to transpose data, converting from LSb to MSb (or vice-versa). The byte to be transposed should be first written to TRANSPOSE and then subsequent reads from TRANSPOSE will return the transposed value of the last written byte (bit 7 becomes bit 0, bit 6 becomes bit 1, and so forth).

11.4 Functional Description

To enable the CRC function, a write to the CRCH register will trigger the first half of the seed mechanism which will place the CRCH value directly into bits 15-8 of the CRC generator shift register. The CRC generator will then expect a write to CRCL to complete the seed mechanism.

As soon as the CRCL register is written to, its value will be loaded directly into bits 7-0 of the shift register, and the second half of the seed mechanism will be complete. This value in CRCH:CRCL will be the initial seed value in the CRC generator.

Now the first byte of the data on which the CRC calculation will be applied should be written to CRCL. This write after the completion of the seed mechanism will trigger the CRC module to begin the CRC checking process. The CRC generator will shift the bits in the CRCL register (MSB first) into the shift register of the generator. One Bus cycle after writing to CRCL all 8 bits have been shifted into the CRC generator, and then the result of the shifting, or the value currently in the shift register, can be read directly from CRCH:CRCL, and the next data byte to include in the CRC calculation can be written to the CRCL register.

This next byte will then also be shifted through the CRC generator's 16-bit shift register, and after the shifting has been completed, the result of this second calculation can be read directly from CRCH:CRCL.

After each byte has finished shifting, a new CRC result will appear in CRCH:CRCL, and an additional byte may be written to the CRCL register to be included within the CRC16-CCITT calculation. A new CRC result will appear in CRCH:CRCL each time 8-bits have been shifted into the shift register.

To start a new CRC calculation, write to CRCH, and the seed mechanism for a new CRC calculation will begin again.

11.4.1 ITU-T (CCITT) Recommendations and Expected CRC Results

The CRC polynomial 0x1021 ($x^{16} + x^{12} + x^5 + 1$) is popularly known as *CRC-CCITT* since it was initially proposed by the ITU-T (formerly CCITT) committee.

Although the ITU-T recommendations are very clear about the polynomial to be used, 0x1021, they accept variations in the way the polynomial is implemented:

- ITU-T V.41 implements the same circuit shown in Figure 11-2, but it recommends a SEED = 0x0000.
- ITU-T T.30 and ITU-T X.25 implement the same circuit shown in Figure 11-2, but they recommend the final CRC result to be negated (one-complement operation). Also, they recommend a SEED = 0xFFFF.

Moreover, it is common to find circuits in literature slightly different from the one suggested by the recommendations above, but also known as CRC-CCITT circuits (many variations require the message to be augmented with zeros).

The circuit implemented in the CRC module is exactly the one suggested by the ITU-T V.41 recommendation, with an added flexibility of a programmable SEED. As in ITU-T V.41, no augmentation is needed and the CRC result is not negated. Table 11-5 shows some expected results that can be used as

a reference. Notice that these are ASCII string messages. For example, message "123456789" is encoded with bytes 0x31 to 0x39 (see ASCII table).

Table 11-5. Expected CRC results

ASCII String Message	SEED (initial CRC value)	CRC result
"A"	0x0000	**0x58e5**
"A"	0xffff	**0xb915**
"A"	0x1d0f[1]	**0x9479**
"123456789"	0x0000	**0x31c3**
"123456789"	0xffff	**0x29b1**
"123456789"	0x1d0f[1]	**0xe5cc**
A string of 256 upper case "A" characters with no line breaks	0x0000	**0xabe3**
A string of 256 upper case "A" characters with no line breaks	0xffff	**0xea0b**
A string of 256 upper case "A" characters with no line breaks	0x1d0f[1]	**0xe938**

[1] **One common variation of CRC-CCITT require the message to be augmented with zeros and a SEED=0xFFFF. The CRC module will give the same results of this alternative implementation when SEED=0x1D0F and no message augmentation.**

## 11.4.2   Programming model extension for CF1Core

NOTE

This section is used on CF1Core (version 1 of ColdFire core) and should be ignored for HCS08 cores.

The CRC module extends its original programming model to allow faster CRC calculations on CF1 cores. Memory offsets 0x4, 0x5, 0x6 and 0x7 are mapped (aliased) onto the CRCL register, in a way that the CF1Core can execute 32-bit store instructions to write data to the CRC module. The code should use a single *mov.l* store instruction to send four bytes beginning at address 0x4, which will be decomposed by the platform into four sequential/consecutive byte writes to offsets 0x4-0x7 (all aliased to the CRCL position).

In addition, reads from 0x4-0x7 return the contents of the CRCL register. Reads from 0x2-0x3 (unused/reserved) return 0x00. Writes to 0x2-0x3 have no effect.

Due to internal CF1Core characteristics, this approach provides a greater data transfer rate than the original programming model used on HCS08 (single byte writes to CRCL position). Figure 11-6 illustrates a message calculation on CF1Core.
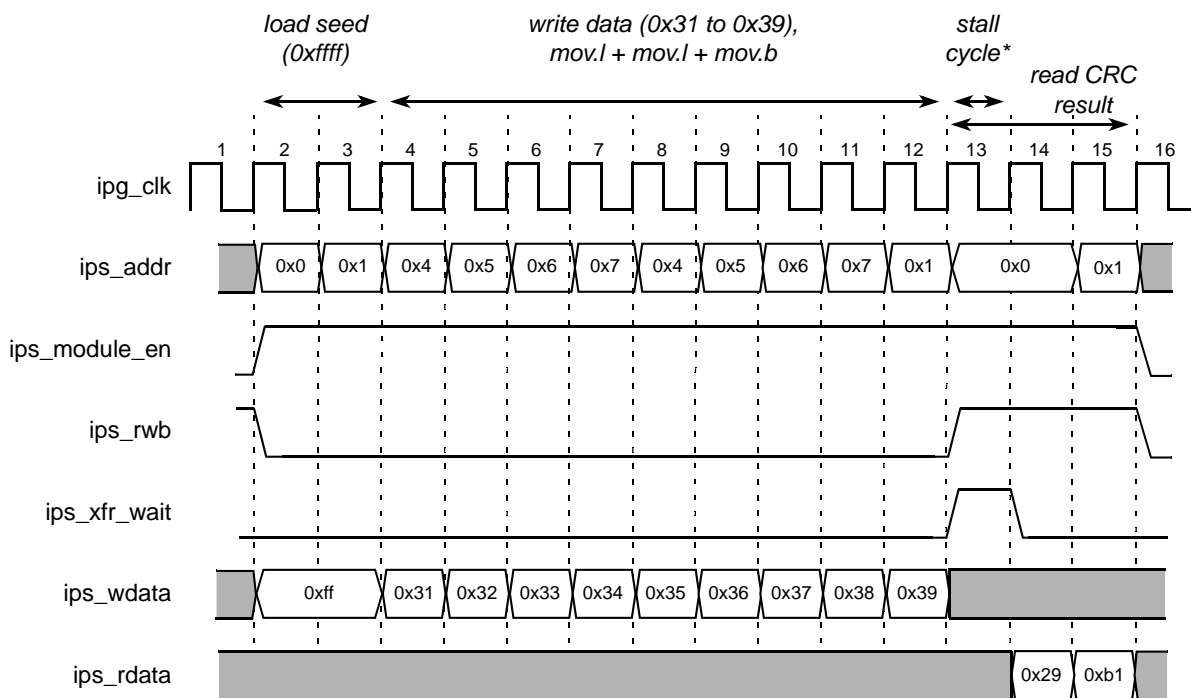
* On cycle 13, there is a read-after-write hazard, since calculation of 0x39 data is underway. *ips_xfr_wait* is asserted to signalize a stall cycle (the IPS master should wait until cycle 14 to read the CRC result).

Figure 11-6. CRC calculation of ASCII message "123456789" (0x31 to 0x39) on CF1Core

11.4.3   Transpose feature

The CRC module provides an optional feature to transpose data (invert bit order). This feature is specially useful on applications where the LSb format is used, since the CRC CCITT expects data in the MSb format. In that case, before writing new data bytes to CRCL, these bytes should be transposed as follows:

1. Write data byte to TRANSPOSE register
2. Read data from TRANSPOSE register (subsequent reads will result in the transposed value of the last written data)
3. Write transposed byte to CRCL.

After all data is fed into CRC, the CRCH:CRCL result is available in the MSb format. Then, these two bytes should also be transposed: the values read from CRCH:CRCL should be written/read to/from the TRANSPOSE register.

Although the transpose feature was initially designed to address LSb applications interfacing with the CRC module, it is important to notice that this feature is not necessarily tied to CRC applications. Since it was designed as an independent register, any application should be able to transpose data by writing/reading to/from the TRANSPOSE register (e.g. Big endian / Little endian conversion in USB).

11.5   Initialization Information

To initialize the CRC Module and initiate a CRC16-CCITT calculation, follow this procedure:

1. Write high byte of initial seed value to CRCH.

2. Write low byte of initial seed value to CRCL.

3. Write first byte of data on which CRC is to be calculated to CRCL (an alternative option is offered for CF1Cores. See Section 11.4.2, "Programming model extension for CF1Core).

4. In the next bus cycle after step 3, if desired, the CRC result from the first byte can be read from CRCH:CRCL.

5. Repeat steps 3-4 until the end of all data to be checked.

Chapter 12
Carrier Modulator Timer (S08CMTV1)

12.1 Introduction

The CMT consists of a carrier generator, modulator, and transmitter that drives the infrared out (IRO) pin.

12.2 Clock Selection

Unlike the Coldire V1 MCF51JE devices, the bus clock is the only clock source for the CMT in the S08 core.

12.3 IRO Pin Description

The IRO pin is the only pin associated with the CMT. The pin is driven by the transmitter output when the MCGEN bit in the CMTMSC register and the IROPEN bit in the CMTOC register are set. If the MCGEN bit is clear and the IROPEN bit is set, the pin is driven by the IROL bit in the CMTOC register. This enable user software to directly control the state of the IRO pin by writing to the IROL bit. If the IROPEN bit is clear, the pin is disabled and is not driven by the CMT module. This is so the CMT can be configured as a modulo timer for generating periodic interrupts without causing pin activity.
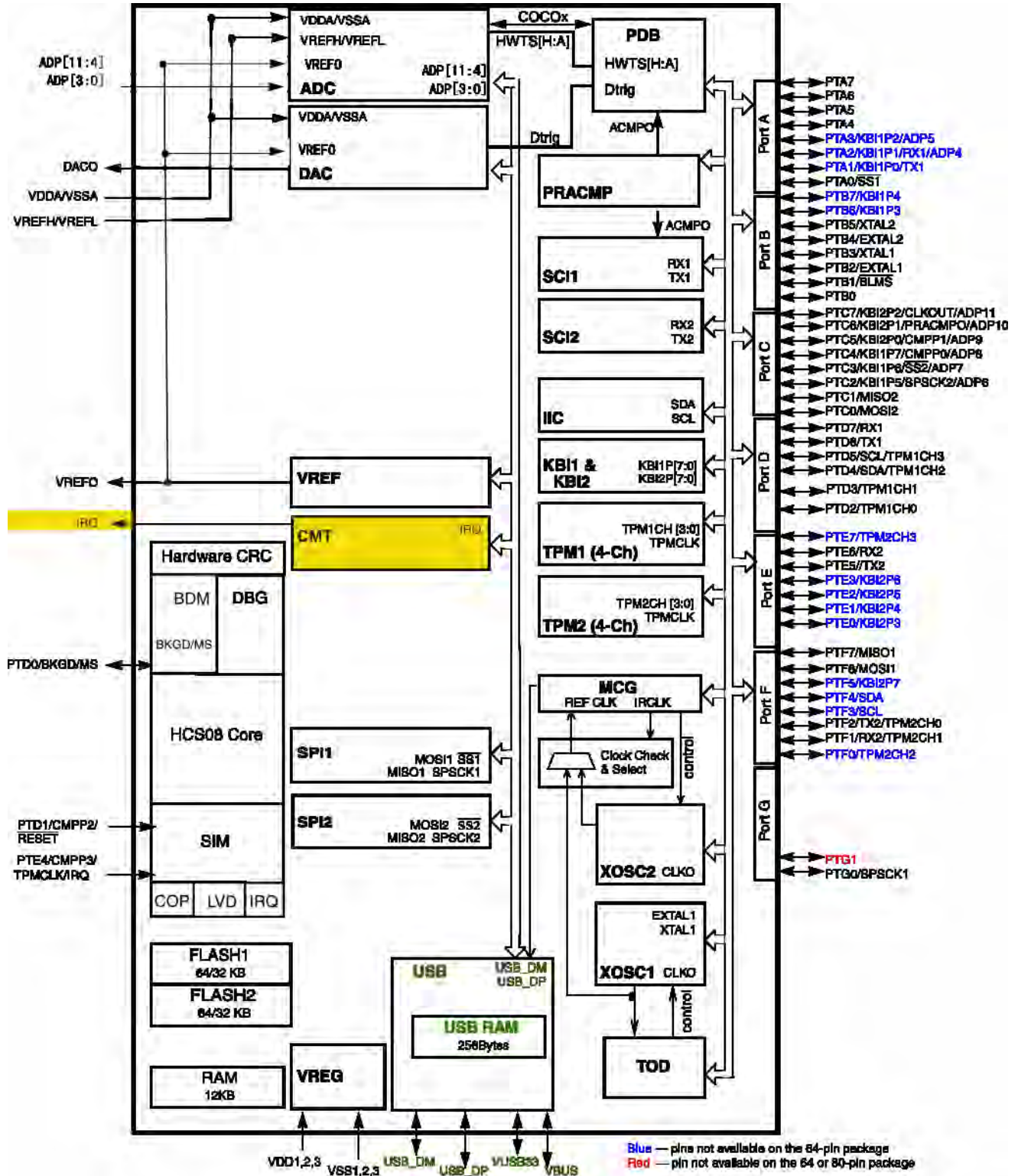
Figure 12-1. Block Diagram Highlighting the CMT Module

12.4 Features

The CMT consists of a carrier generator, modulator, and transmitter which drives the infrared out pin. The features of this module include:

- Four modes of operation
    - Time with independent control of high and low times
    - Baseband
    - Frequency shift key (FSK)
    - Direct software control of IRO pin
- Extended space operation in time, baseband, and FSK modes
- Selectable input clock divide: 1, 2, 4, or 8
- Interrupt on end of cycle
    - Ability to disable IRO pin and use as timer interrupt

12.5 CMT Block Diagram
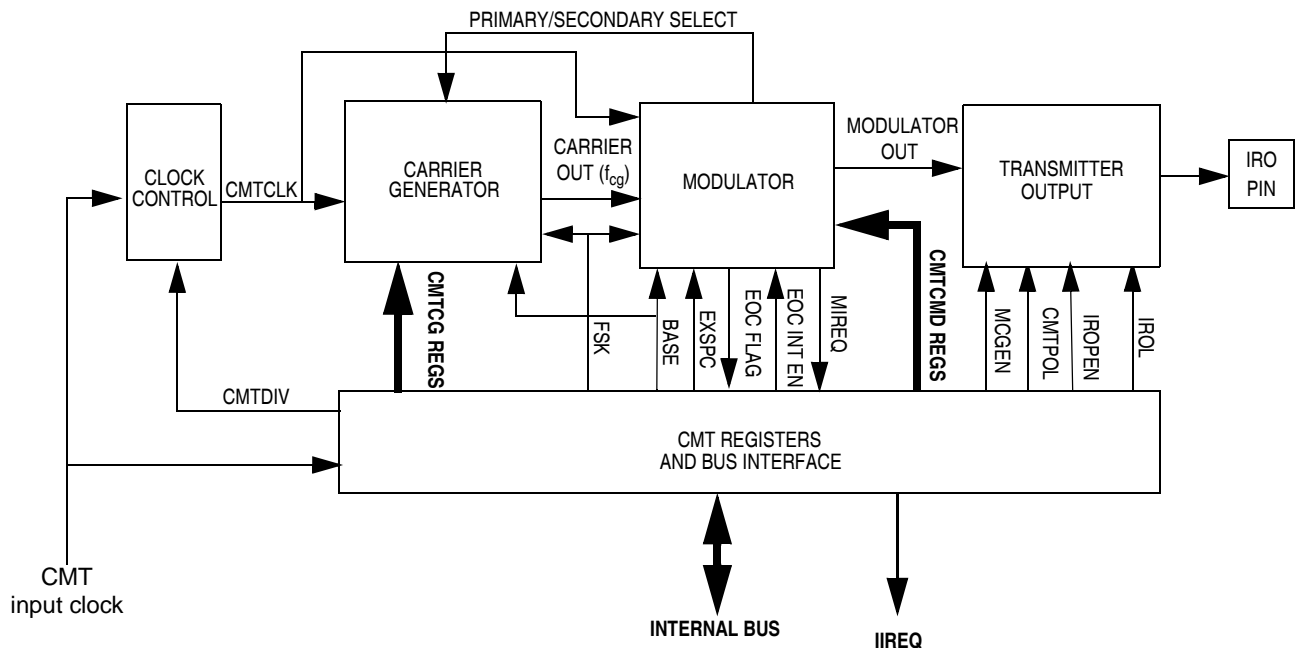
Figure 12-2. Carrier Modulator Transmitter Module Block Diagram

12.6 External Signal Descriptions

There is only one pin associated with the CMT, the IRO pin. The pin is driven by the transmitter output when the MCGEN bit in the CMTMSC register is set and the IROPEN bit in the CMTOC register is set. If the MCGEN bit is clear and the IROPEN bit is set, the pin is driven by the IROL bit in the CMTOC register. This enables user software to directly control the state of the IRO pin by writing to the IROL bit.

If the IROPEN bit is clear, the pin is disabled and is not driven by the CMT module. This is so the CMT can be configured as a modulo timer for generating periodic interrupts without causing pin activity.

12.7   Register Definition

The following registers control and monitor CMT operation:

- CMT carrier generator data registers (CMTCGH1, CMTCGL1, CMTCGH2, CMTCGL2)
- CMT output control register (CMTOC)
- CMT modulator status and control register (CMTMSC)
- CMT modulator period data registers (CMTCMD1, CMTCMD2, CMTCMD3, CMTCMD4)

12.7.1   Carrier Generator Data Registers (CMTCGH1, CMTCGL1, CMTCGH2, and CMTCGL2)

The carrier generator data registers contain the primary and secondary high and low values for generating the carrier output.

CMTCGH1	7	6	5	4	3	2	1	0
Read: Write:	PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0
Reset:	U	U	U	U	U	U	U	U

CMTCGL1	7	6	5	4	3	2	1	0
Read: Write:	PL7	PL6	PL5	PL4	PL3	PL2	PL1	PL0
Reset:	U	U	U	U	U	U	U	U

CMTCGH2	7	6	5	4	3	2	1	0
Read: Write:	SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
Reset:	U	U	U	U	U	U	U	U

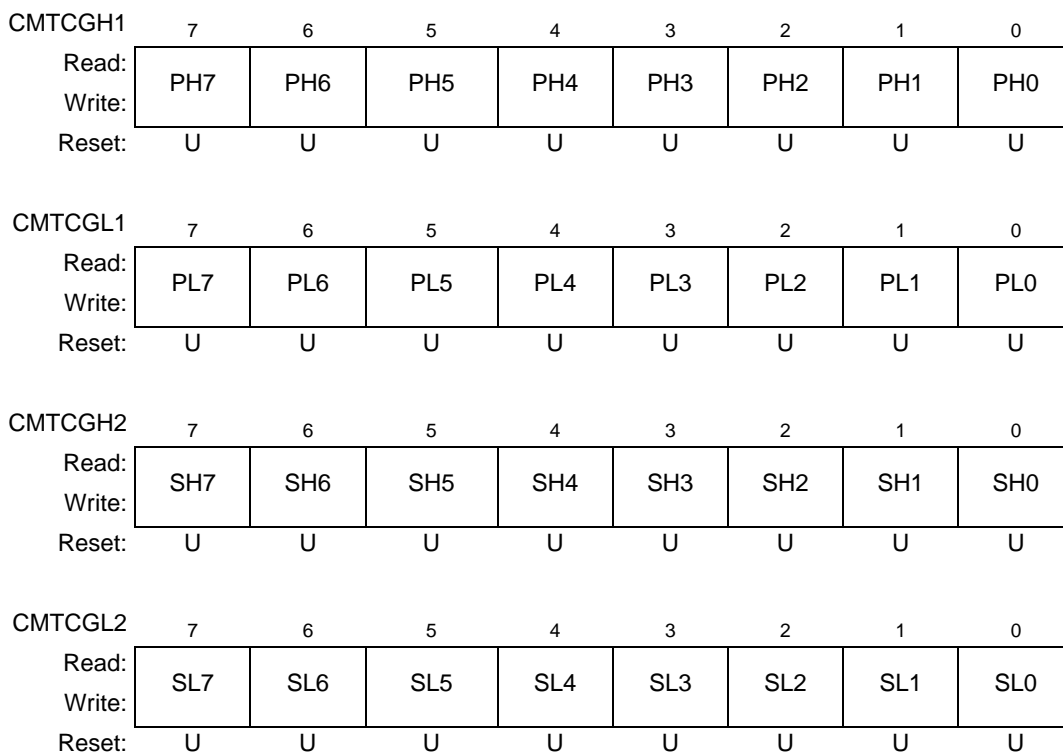| CMTCGL2	7	6	5	4	3	2	1	0
Read: Write:	SL7	SL6	SL5	SL4	SL3	SL2	SL1	SL0
Reset:	U	U	U	U	U	U	U	U

Figure 12-3. CMT Carrier Generator Data Registers (CMTCGH1, CMTCGL1, CMTCGH2, CMTCGL2)

Table 12-1. Carrier High and Low Time Data Values

Carrier	Data Values	Description
Primary Carrier High and Low Time Data Values	PH0–PH7 and PL0–PL7	When selected, these bits contain the number of input clocks required to generate the carrier high and low time periods. When operating in time mode (see Section 12.8.2.1, "Time Mode") this register pair is always selected. When operating in FSK mode (see Section 12.8.2.3, "FSK Mode") this register pair and the secondary register pair are alternately selected under control of the modulator. The primary carrier high and low time values are undefined out of reset. These bits must be written to nonzero values before the carrier generator is enabled to avoid spurious results.
Secondary Carrier High and Low Time Data Values	SH0–SH7 and SL0–SL7	When selected, these bits contain the number of input clocks required to generate the carrier high and low time periods. When operating in time mode (see Section 12.8.2.1, "Time Mode") this register pair is never selected. When operating in FSK mode (see Section 12.8.2.3, "FSK Mode") this register pair and the primary register pair are alternately selected under control of the modulator. The secondary carrier high and low time values are undefined out of reset. These bits must be written to nonzero values before the carrier generator is enabled when operating in FSK mode.

12.7.2   CMT Output Control Register (CMTOC)

This register is used to control the IRO output of the CMT module.

	7	6	5	4	3	2	1	0
R	IROL	CMTPOL	IROPEN	0	0	0	0	0
W								
Reset:	0	0	0	0	0	0	0	0
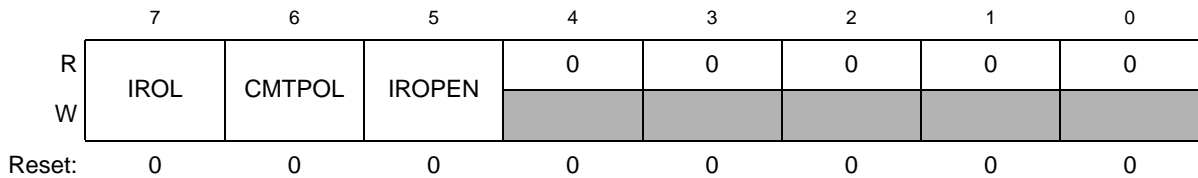
Figure 12-4. CMT Output Control Register (CMTOC)
Table 12-2. PRACMPCS Field Descriptions

Field	Description
7 IROL	**IRO Latch Control** Reading IROL reads the state of the IRO latch. Writing IROL changes the state of the IRO pin when the MCGEN bit in the CMTMSC register is clear and the IROPEN bit is set.

Table 12-2. PRACMPCS Field Descriptions (Continued)

Field	Description
6 CMTPOL	**CMT Output Polarity** **The CMTPOL bit controls the polarity of the IRO pin output of the CMT.** 1 IRO pin is active high 0 IRO pin is active low
5 IROPEN	**IRO Pin Enable** The IROPEN bit is used to enable and disable the IRO pin. When the pin is enabled, it is an output which drives out either the CMT transmitter output or the state of the IROL bit depending on whether the MCGEN bit in the CMTMSC register is set or not. Also, the state of the output is either inverted or not depending on the state of the CMTPOL bit. When the pin is disabled, it is in a high impedance state so as not to draw any current. The pin is disabled during reset. 0 IRO pin disabled 1 IRO pin enabled as output

12.7.3   CMT Modulator Status and Control Register

The CMT modulator status and control register (CMTMSC) contains the modulator and carrier generator enable (MCGEN), end of cycle interrupt enable (EOCIE), FSK mode select (FSK), baseband enable (BASE), extended space (EXSPC), prescaler (CMTDIV) bits, and the end of cycle (EOCF) status bit.

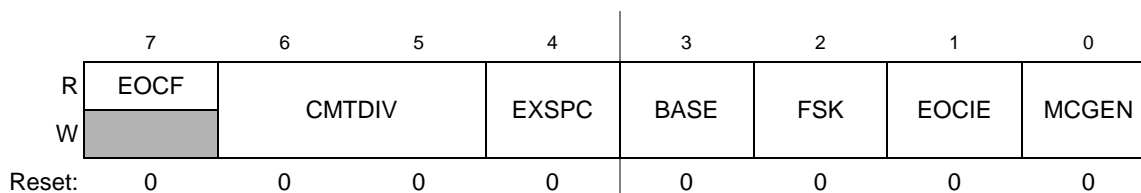|	7	6	5	4	3	2	1	0
R	EOCF	CMTDIV		EXSPC	BASE	FSK	EOCIE	MCGEN
W								
Reset:	0	0	0	0	0	0	0	0

Figure 12-5. CMT Modulator Status and Control Register (CMTMSC)

Table 12-3. PRACMPCS Field Descriptions

Field	Description
7 EOCF	**End Of Cycle Status Flag** The EOCF bit is set when: • The modulator is not currently active and the MCGEN bit is set to begin the initial CMT transmission. • At the end of each modulation cycle while the MCGEN bit is set. This is recognized when a match occurs between the contents of the space period register and the down counter. At this time, the counter is initialized with the (possibly new) contents of the mark period buffer, CMTCMD1 and CMTCMD2, and the space period register is loaded with the (possibly new) contents of the space period buffer, CMTCMD3 and CMTCMD4. This flag is cleared by a read of the CMTMSC register followed by an access of CMTCMD2 or CMTCMD4. In the case where the MCGEN bit is cleared and then set before the end of the modulation cycle, EOCF will not be set when MCGEN is set, but will be set at the end of the current modulation cycle. 0 No end of modulation cycle occurrence since flag last cleared 1 End of modulator cycle has occurred
6–5 CMTDIV	**CMT Clock Divide Prescaler** The CMT clock divide prescaler causes the CMT to be clocked at the CMT input clock frequency, or the CMT input clock frequency divided by 1, 2 ,4, or 8. Since these bits are not double buffered, they should not be changed during a transmission. 00 CMT input clock ÷ 1 01 CMT input clock ÷ 2 10 CMT input clock ÷ 4 11 CMT input clock ÷ 8
4 EXSPC	**Extended Space Enable** The EXSPC bit enables extended space operation 1 Extended space enabled 0 Extended space disabled
3 BASE	**Baseband Enable** When set, the BASE bit disables the carrier generator and forces the carrier output high for generation of baseband protocols. When BASE is clear, the carrier generator is enabled and the carrier output toggles at the frequency determined by values stored in the carrier data registers. See Section 12.8.2.2, "Baseband Mode". This bit is cleared by reset. This bit is not double buffered and should not be written to during a transmission. 0 Baseband mode disabled 1 Baseband mode enabled
2 FSK	**FSK Mode Select** The FSK bit enables FSK operation 0 CMT operates in Time or Baseband mode 1 CMT operates in FSK mode
1 EOCIE	**End of Cycle Interrupt Enable** A CPU interrupt will be requested when EOCF is set if EOCIE is high.0CPU interrupt disabled 1 CPU interrupt enabled
0 MCGEN	**Modulator and Carrier Generator Enable** Setting MCGEN will initialize the carrier generator and modulator and will enable all clocks. Once enabled, the carrier generator and modulator will function continuously. When MCGEN is cleared, the current modulator cycle will be allowed to expire before all carrier and modulator clocks are disabled (to save power) and the modulator output is forced low. To prevent spurious operation, the user should initialize all data and control registers before enabling the system. 0 Modulator and carrier generator disabled 1 Modulator and carrier generator enabled

12.7.4 CMT Modulator Data Registers (CMTCMD1, CMTCMD2, CMTCMD3 and CMTCMD4)

The modulator data registers control the mark and space periods of the modulator for all modes. The contents of these registers are transferred to the modulator down counter and space period register upon the completion of a modulation period.
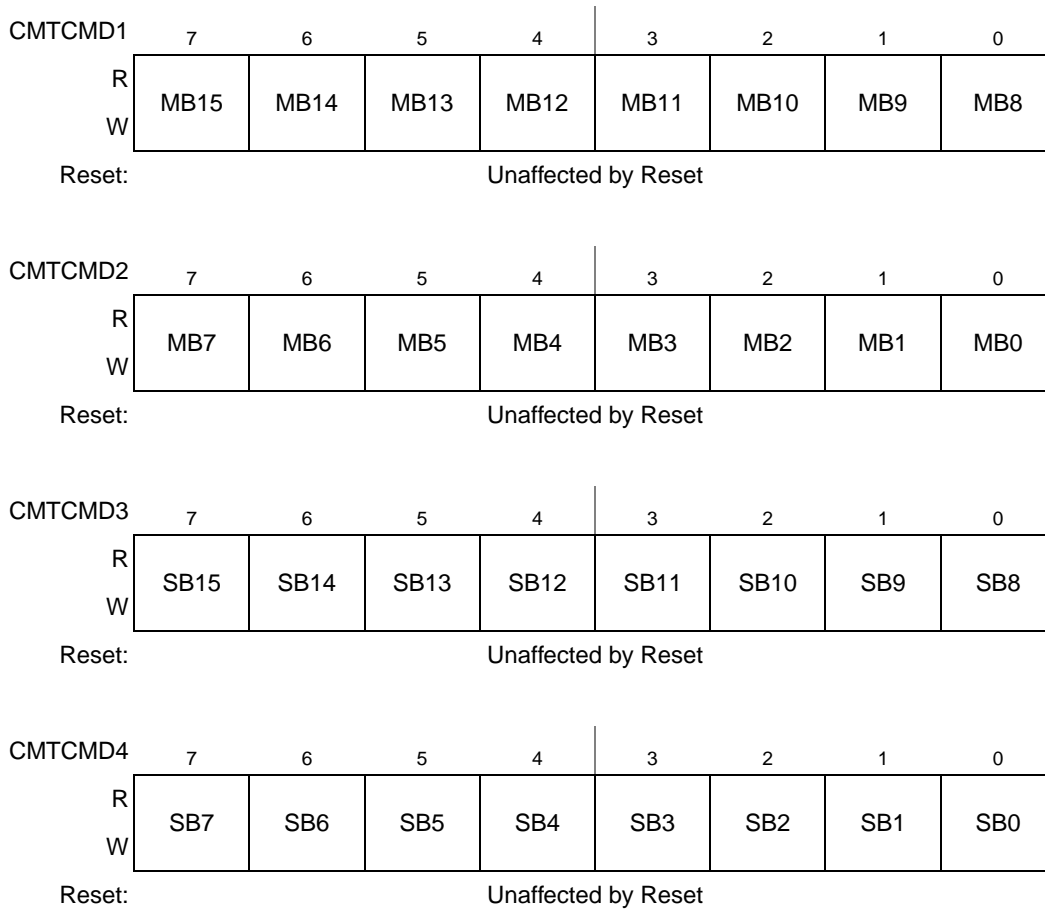

Figure 12-6. CMT Modulator Data Registers (CMTCMD1, CMTCMD2, CMTCMD3 and CMTCMD4)

12.8 Functional Description

The CMT module consists of a carrier generator, a modulator, a transmitter output and control registers. The block diagram is shown in Figure 12-2. The module has three main modes of operation:

- Time
- Baseband
- Frequency shift key (FSK).

When operating in time mode, the user independently defines the high and low times of the carrier signal to determine both period and duty cycle. The carrier generator resolution is 125 ns when operating with an 8 MHz internal bus frequency and the CMTMSC[CMTDIV] equal 00. The carrier generator can generate

signals with periods between 250 ns (4 MHz) and 127.5 μs (7.84 kHz) in steps of 125 ns. The table below shows the relationship between the clock divide bits and the carrier generator resolution, minimum carrier generator period, and minimum modulator period.

Table 12-4. Clock Divide

CMT Input Clock (MHz)	CMTDIV	Carrier Generator Resolution (μs)	Min Carrier Generator Period (μs)	Min Modulator Period (μs)
8	00	0.125	0.25	1.0
8	01	0.25	0.5	2.0
8	10	0.5	1.0	4.0
8	11	1.0	2.0	8.0

The possible duty cycle options will depend upon the number of counts required to complete the carrier period. For example, a 1.6 MHz signal has a period of 625 ns and will therefore require 5 x 125 ns counts to generate. These counts may be split between high and low times, so the duty cycles available will be 20% (one high, four low), 40% (two high, three low), 60% (three high, two low) and 80% (four high, one low).

For lower frequency signals with larger periods, higher resolution (as a percentage of the total period) duty cycles are possible.

When the BASE bit in the CMT modulator status and control register (CMTMSC) is set, the carrier output (f_{cg}) to the modulator is held to IRO pin active level continuously to allow for the generation of baseband protocols.

A third mode allows the carrier generator to alternate between two sets of high and low times. When operating in FSK mode, the generator will toggle between the two sets when instructed by the modulator, allowing the user to dynamically switch between two carrier frequencies without CPU intervention.

The modulator provides a simple method to control protocol timing. The modulator has a minimum resolution of 1.0 μs with an 8 MHz internal CMT input clock. It can count bus clocks (to provide real-time control) or it can count carrier clocks (for self-clocked protocols). See Section 12.8.2, "Modulator" for more details.

The transmitter output block controls the state of the infrared out pin (IRO). The modulator output is gated on to the IRO pin when the modulator/carrier generator is enabled.

A summary of the possible modes is shown in Table 12-5.

Table 12-5. CMT Modes of Operation

Mode	MCGEN Bit[1]	BASE Bit[2]	FSK Bit[(2)]	EXSPC Bit	Comment
Time	1	0	0	0	f_{cg} controlled by primary high and low registers. f_{cg} transmitted to IRO pin when modulator gate is open.

Table 12-5. CMT Modes of Operation (Continued)

Mode	MCGEN Bit[1]	BASE Bit[2]	FSK Bit[(2)]	EXSPC Bit	Comment
Baseband	1	1	x	0	f_{cg} is always active level. IRO pin active level when modulator gate is open.
FSK	1	0	1	0	f_{cg} control alternates between primary high/low registers and secondary high/low registers. f_{cg} transmitted to IRO pin when modulator gate is open.
Extended Space	1	x	x	1	Setting the EXSPC bit causes subsequent modulator cycles to be spaces (modulator out not asserted) for the duration of the modulator period (mark and space times).
IRO Latch	0	x	x	x	IROL bit controls state of IRO pin.

[1] To prevent spurious operation, initialize all data and control registers before beginning a transmission (MCGEN=1).
[2] These bits are not double buffered and should not be changed during a transmission (while MCGEN=1).

12.8.1 Carrier Generator

The carrier signal is generated by counting a register-selected number of input clocks (125 ns for an 8 MHz bus) for both the carrier high time and the carrier low time. The period is determined by the total number of clocks counted. The duty cycle is determined by the ratio of high time clocks to total clocks counted. The high and low time values are user programmable and are held in two registers.

An alternate set of high/low count values is held in another set of registers to allow the generation of dual frequency FSK (frequency shift keying) protocols without CPU intervention.

NOTE

Only non-zero data values are allowed. The carrier generator will not work if any of the count values are equal to zero.

The MCGEN bit in the CMTMSC register must be set and the BASE bit must be cleared to enable carrier generator clocks. When the BASE bit is set, the carrier output to the modulator is held to IRO pin active level continuously. The block diagram is shown in .
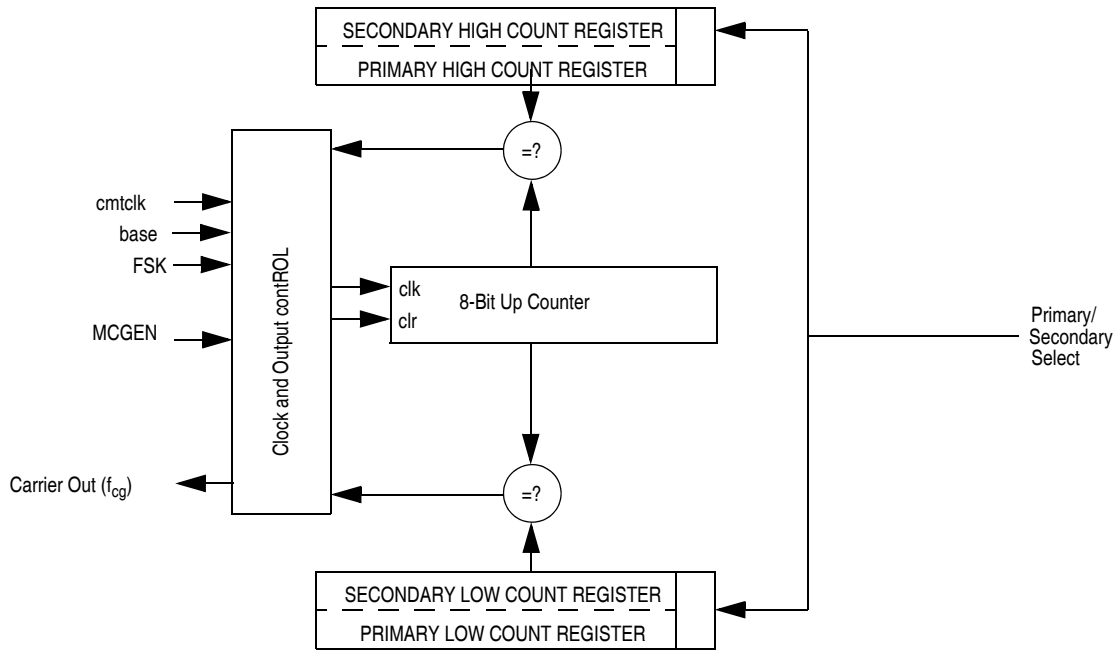
Figure 12-7. Carrier Generator Block Diagram

The high/low time counter is an 8-bit up counter. After each increment, the contents of the counter are compared with the appropriate high or low count value register. When the compare value is reached, the counter is reset to a value of 0x01, and the compare is redirected to the other count value register.

Assuming that the high time count compare register is currently active, a valid compare will cause the carrier output to be driven IRO pin active level. The counter will continue to increment (starting at reset value of 0x01). When the value stored in the selected low count value register is reached, the counter will again be reset and the carrier output will be driven IRO pin inactive level.

The cycle repeats, automatically generating a periodic signal which is directed to the modulator. The lowest frequency (maximum period) and highest frequency (minimum period) which can be generated are defined as:

$$f_{max} = f_{CMTCLK} \div (2 \times 1) \text{ Hz} \qquad \qquad \textbf{\textit{Eqn. 12-1}}$$

$$f_{min} = f_{CMTCLK} \div (2 \times (2^8 - 1)) \text{ Hz} \qquad \qquad \textbf{\textit{Eqn. 12-2}}$$

In the general case, the carrier generator output frequency is:

$$f_{cg} = f_{CMTCLK} \div (\text{Highcount} + \text{Lowcount}) \text{ Hz} \qquad \qquad \textbf{\textit{Eqn. 12-3}}$$

Where:

> $0 < Highcount < 256$ and
> $0 < Lowcount < 256$

The duty cycle of the carrier signal is controlled by varying the ratio of high time to low + high time. As the input clock period is fixed, the duty cycle resolution will be proportional to the number of counts required to generate the desired carrier period.

$$\text{Duty Cycle} = \frac{\text{Highcount}}{\text{Highcount} + \text{Lowcount}}$$

Eqn. 12-4

12.8.2 Modulator

The modulator has three main modes of operation:

- The modulator can gate the carrier onto the modulator output (Time mode)
- The modulator can control the logic level of the modulator output (Baseband mode)
- The modulator can count carrier periods and instruct the carrier generator to alternate between two carrier frequencies whenever a modulation period (mark + space counts) expires (FSK mode)

The modulator includes a 17-bit down counter with underflow detection. The counter is loaded from the 16-bit modulation mark period buffer registers, CMTCMD1 and CMTCMD2. The most significant bit is loaded with a logic zero and serves as a sign bit. When the counter holds a positive value, the modulator gate is open and the carrier signal is driven to the transmitter block.

When the counter underflows, the modulator gate is closed and a 16-bit comparator is enabled which compares the logical complement of the value of the down counter with the contents of the modulation space period register which has been loaded from the registers, CMTCMD3 and CMTCMD4.

When a match is obtained the cycle repeats by opening the modulator gate, reloading the counter with the contents of CMTCMD1 and CMTCMD2, and reloading the modulation space period register with the contents of CMTCMD3 and CMTCMD4.

Should the contents of the modulation space period register be all zeroes, the match will be immediate and no space period will be generated (for instance, for FSK protocols which require successive bursts of different frequencies).

The MCGEN bit in the CMTMSC register must be set to enable the modulator timer.
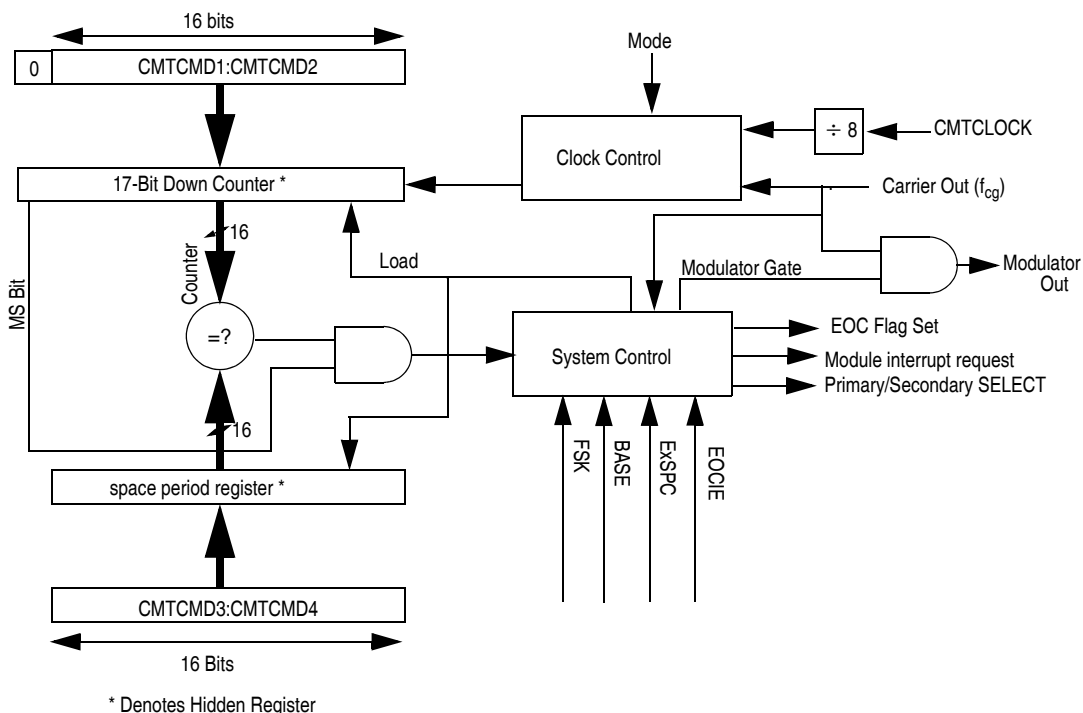
Figure 12-8. Modulator Block Diagram

12.8.2.1 Time Mode

When the modulator operates in time mode (MCGEN bit is set, BASE bit is clear, and FSK bit is clear), the modulation mark period consists of an integer number of CMTCLK ÷ 8 clock periods. The modulation space period consists of zero or an integer number of CMTCLK ÷ 8 clock periods. With an 8 MHz bus and CMTDIV = 00, the modulator resolution is 1 µs and has a maximum mark and space period of about 65.535 ms each. See Figure 12-9 for an example of the time mode and baseband mode outputs.

The mark and space time equations for time and baseband mode are:

$$t_{mark} = (CMTCMD1{:}CMTCMD2 + 1) \div (f_{CMTCLK} \div 8)$$ *Eqn. 12-5*

$$t_{space} = CMTCMD3{:}CMTCMD4 \div (f_{CMTCLK} \div 8)$$ *Eqn. 12-6*

where CMTCMD1:CMTCMD2 and CMTCMD3:CMTCMD4 are the decimal values of the concatenated registers.
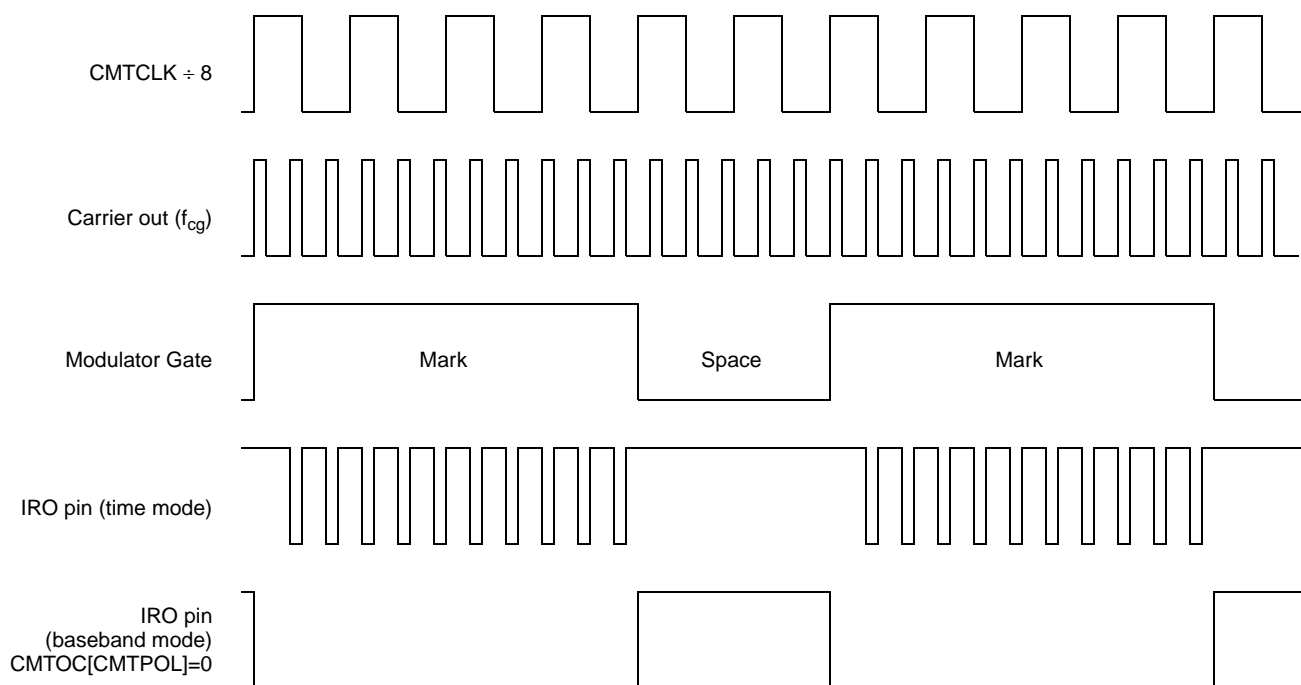
Figure 12-9. Example CMT Output in Time and Baseband Modes

12.8.2.2   Baseband Mode

Baseband mode (MCGEN bit is set and BASE bit is set) is a derivative of time mode, where the mark and space period is based on (CMTCLK ÷ 8) counts. The mark and space calculations are the same as in time mode. In this mode the modulator output will be at active level for the duration of the mark period and at an inactive level for the duration of a space period. See Figure 12-9 for an example of the output for both baseband and time modes. In the example, the carrier out frequency (f_{cg}) is generated with a high count of 0x01 and a low count of 0x02 which results in a divide of 3 of CMTCLK with a 33% duty cycle. The modulator down counter was loaded with the value 0x0003 and the space period register with 0x0002.

NOTE

The waveforms in Figure 12-9 and Figure 12-10 are for the purpose of conceptual illustration and are not meant to represent precise timing relationships between the signals shown.

12.8.2.3   FSK Mode

When the modulator operates in FSK mode (MCGEN bit is set, FSK bit is set, and BASE bit is clear), the modulation mark and space periods consist of an integer number of carrier clocks (space period can be zero). When the mark period expires, the space period is transparently started (as in time mode). The carrier generator toggles between primary and secondary data register values whenever the modulator space period expires.

The space period provides an interpulse gap (no carrier). If CMTCMD3:CMTCMD4 = 0x0000, then the modulator and carrier generator will switch between carrier frequencies without a gap or any carrier glitches (zero space).

Using timing data for carrier burst and interpulse gap length calculated by the CPU, FSK mode can automatically generate a phase-coherent, dual-frequency FSK signal with programmable burst and interburst gaps.

The mark and space time equations for FSK mode are:

$$t_{mark} = (CMTCMD1:CMTCMD2 + 1) \div f_{cg}$$

Eqn. 12-7

$$t_{space} = CMTCMD3:CMTCMD4 \div f_{cg}$$

Eqn. 12-8

Where f_{cg} is the frequency output from the carrier generator. The example in figure below shows what the IRO pin output looks like in FSK mode with the following values: CMTCMD1:CMTCMD2 = 0x0003, CMTCMD3:CMTCMD4 = 0x0002, primary carrier high count = 0x01, primary carrier low count = 0x02, secondary carrier high count = 0x03, and secondary carrier low count = 0x01.
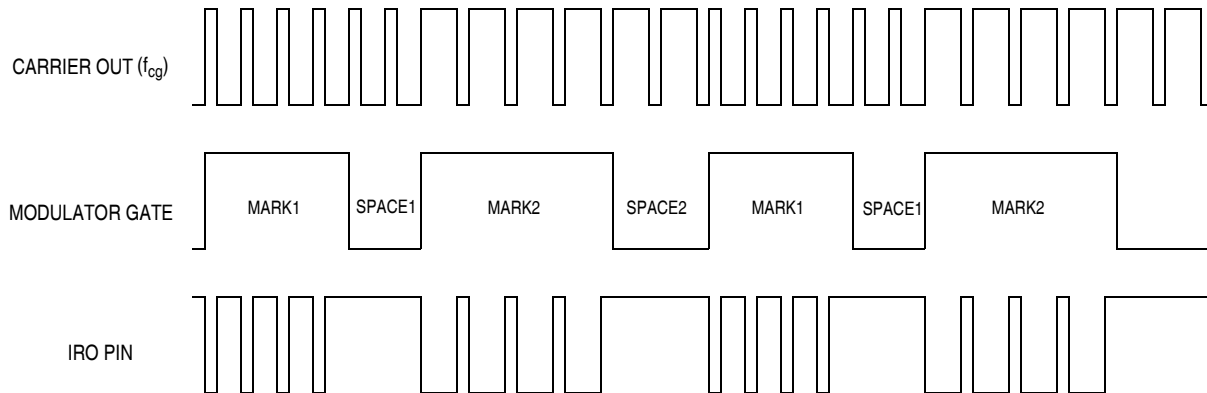
Figure 12-10. Example CMT Output in FSK Mode

12.8.2.4   Extended Space Operation

In either time, baseband or FSK mode, the space period can be made longer than the maximum possible value of the space period register. Setting the EXSPC bit in the CMTMSC register will force the modulator to treat the next modulation period (beginning with the next load of the counter and space period register) as a space period equal in length to the mark and space counts combined. Subsequent modulation periods will consist entirely of these extended space periods with no mark periods. Clearing EXSPC will return the modulator to standard operation at the beginning of the next modulation period.

12.8.2.4.1   EXSPC Operation in Time Mode

To calculate the length of an extended space in time or baseband modes, add the mark and space times and multiply by the number of modulation periods that EXSPC is set.

$$t_{exspace} = (t_{mark} + t_{space}) \times \text{(number of modulation periods)} \qquad \text{\textit{Eqn. 12-9}}$$

For an example of extended space operation, see Figure 12-11.

NOTE

The EXSPC feature can be used to emulate a zero mark event.

Figure 12-11. Extended Space Operation

12.8.2.4.2 EXSPC Operation in FSK Mode

In FSK mode, the modulator continues to count carrier out clocks, alternating between the primary and secondary registers at the end of each modulation period.
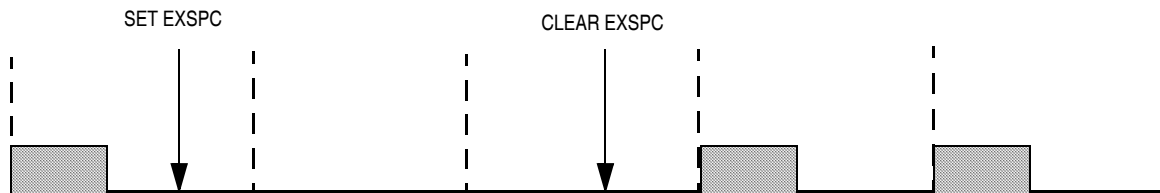
To calculate the length of an extended space in FSK mode, one needs to know whether the EXSPC bit was set on a primary or secondary modulation period, as well as the total number of both primary and secondary modulation periods completed while the EXSPC bit is high. A status bit for the current modulation is not accessible to the CPU. If necessary, software should maintain tracking of the current modulation cycle (primary or secondary). The extended space period ends at the completion of the space period time of the modulation period during which the EXSPC bit is cleared.

If the EXSPC bit was set during a primary modulation cycle, use the equation:

$$t_{exspace} = (t_{space})_p + (t_{mark} + t_{space})_s + (t_{mark} + t_{space})_p + ... \qquad \text{\textit{Eqn. 12-10}}$$

Where the subscripts p and s refer to mark and space times for the primary and secondary modulation cycles.

If the EXSPC bit was set during a secondary modulation cycle, use the equation:

$$t_{exspace} = (t_{space})_s + (t_{mark} + t_{space})_p + (t_{mark} + t_{space})_s + ... \qquad \text{\textit{Eqn. 12-11}}$$

12.8.3 Transmitter

The transmitter output block controls the state of the infrared out pin (IRO). The modulator output is gated on to the IRO pin when the modulator/carrier generator is enabled. When the modulator/carrier generator is disabled, the IRO pin is controlled by the state of the IRO latch.

A polarity bit in the CMTOC register enables the IRO pin to be high true or low true.

12.8.4 CMT Interrupts

The end of cycle flag (EOCF) is set when:

- The modulator is not currently active and the MCGEN bit is set to begin the initial CMT transmission
- At the end of each modulation cycle (when the counter is reloaded from CMTCMD1:CMTCMD2) while the MCGEN bit is set

In the case where the MCGEN bit is cleared and then set before the end of the modulation cycle, the EOCF bit will not be set when the MCGEN is set, but will become set at the end of the current modulation cycle.

When the MCGEN becomes disabled, the CMT module does not set the EOC flag at the end of the last modulation cycle.

The EOCF bit is cleared by reading the CMT modulator status and control register (CMTMSC) followed by an access of CMTCMD2 or CMTCMD4.

If the EOC interrupt enable (EOCIE) bit is high when the EOCF bit is set, the CMT module will generate an interrupt request. The EOCF bit must be cleared within the interrupt service routine to prevent another interrupt from being generated after exiting the interrupt service routine.

The EOC interrupt is coincident with loading the down-counter with the contents of CMTCMD1:CMTCMD2 and loading the space period register with the contents of CMTCMD3:CMTCMD4. The EOC interrupt provides a means for the user to reload new mark/space values into the modulator data registers. Modulator data register updates will take effect at the end of the current modulation cycle. Note that the down-counter and space period register are updated at the end of every modulation cycle, irrespective of interrupt handling and the state of the EOCF flag.

12.8.5   Low-Power Mode Operation

12.8.5.1   Wait Mode Operation

During wait mode the CMT, if enabled, will continue to operate normally. However, there will be no new codes or changes of pattern mode while in wait mode, because the CPU is not operating.

12.8.5.2   Stop3 Mode Operation

During Stop3 mode, clocks to the CMT module are halted. No registers are affected.

Note that because the clocks are halted, the CMT will resume upon exit from Stop3. Software should ensure that the Stop3 mode is not entered while the modulator is still in operation to prevent the IRO pin from being asserted while in Stop3 mode. This may require a time-out period from the time that the MCGEN bit is cleared to allow the last modulator cycle to complete.

12.8.5.3   Stop2 Mode Operation

During Stop2 mode, the CMT module is completely powered off internally and the IRO pin state at the time that Stop2 mode is entered is latched and held. To prevent the IRO pin from being asserted while in Stop2 mode, software should assure that the pin is not active when entering Stop2 mode. Upon wake-up from Stop2 mode, the CMT module will be in the reset state.

12.8.5.4 Background Mode Operation

When the microcontroller is in active background mode, the CMT temporarily suspends all counting until the microcontroller returns to normal user mode.

Chapter 13
12-Bit Digital-to-Analog Converter (DAC12LVLPV1)

13.1 Introduction

The 12-bit Digital-to-Analog Converter (DAC) is a low-power, general purpose DAC. The 12-bit Digital-to-Analog converter (DAC) can provide a voltage output on the DACO pin.

Figure 13-1 shows the MC9S08JE128 series block diagram with the DAC highlighted.

13.1.1 DAC Clock Gating

The bus clock to the DAC can be gated on and off using the DAC bit in SCGC1. These bits are set after any reset, which enables the bus clock to this module. To conserve power, these bits can be cleared to disable the clock to this module when not in use. See Section 5.7.8, "System Clock Gating Control 1 Register (SCGC1)," for details.

13.1.2 DAC V_{ext} and V_{int} Configuration

On this device, V_{ext} is connected to VDDA and V_{int} is connected to the bandgap voltage, V_{refo}.

13.1.3 DAC Hardware Trigger Configuration

On this device, the hardware trigger is connected to the PDB output.
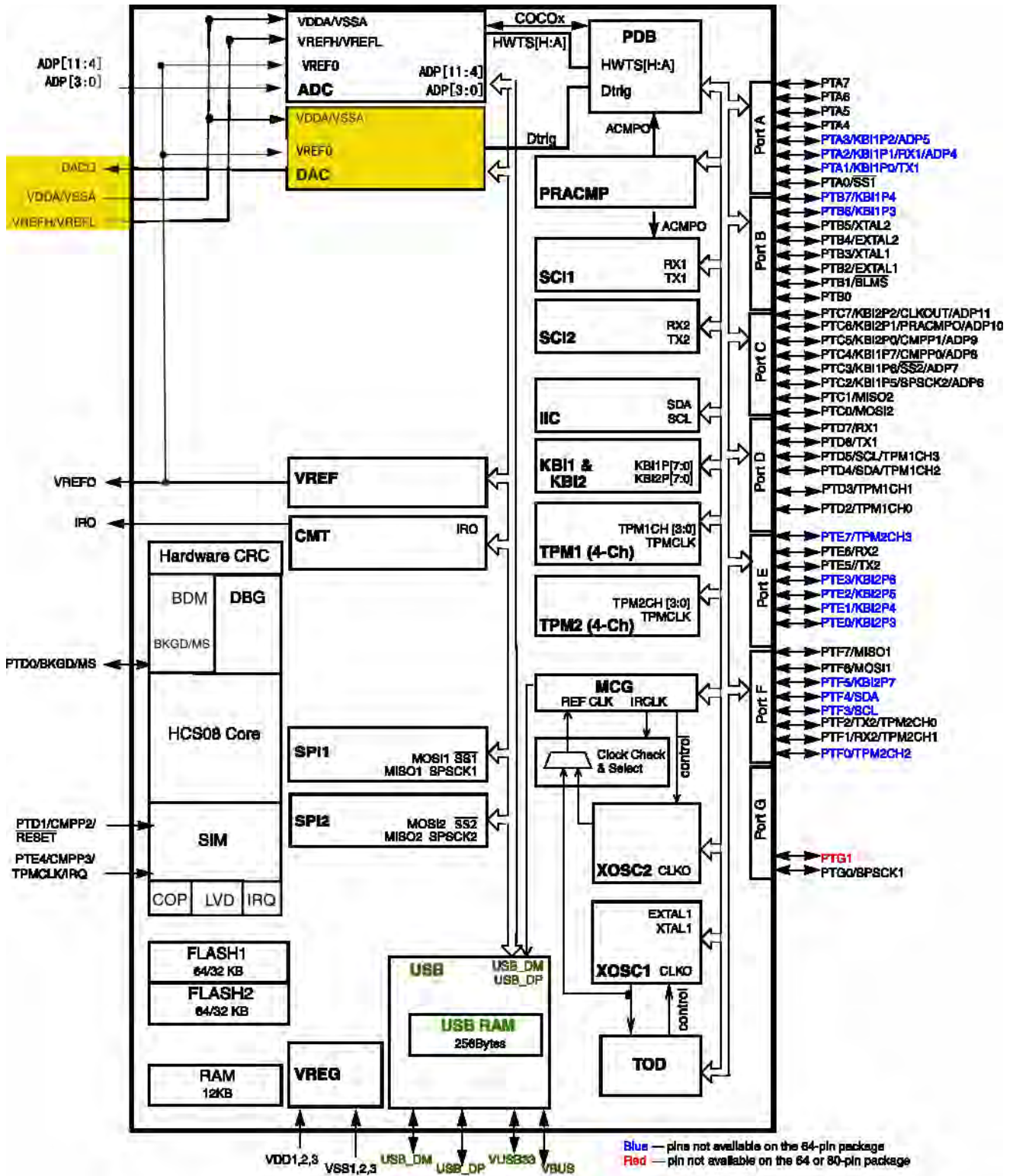
Figure 13-1. MC9S08JE128 series Block Diagram

MC9S08JE128 Series Reference Manual, Rev. 3

Freescale Semiconductor

13.1.4 Features

The DAC module features include:

- 1.8 V — 3.6 V operation.
- On-chip programmable reference generator output (1/4096 Vin to Vin, step is 1/4096 Vin)
- Vin can be selected from two reference sources:
  - While VDDA is 1.8 V ~ 3.6 V, VREFH_EXT (VDDA) can be used and guaranteed.
  - While VDDA is 2.4 V ~ 3.6 V, both VREFH_INT (output of VREF) and VREFH_EXT (VDDA) can be used and guaranteed.
- Optional operation in STOP3 mode
- 16-word data buffer supported with configurable watermark and multiple operation modes

13.1.5 Block Diagram

Figure 13-2 is the block diagram of the DAC module.
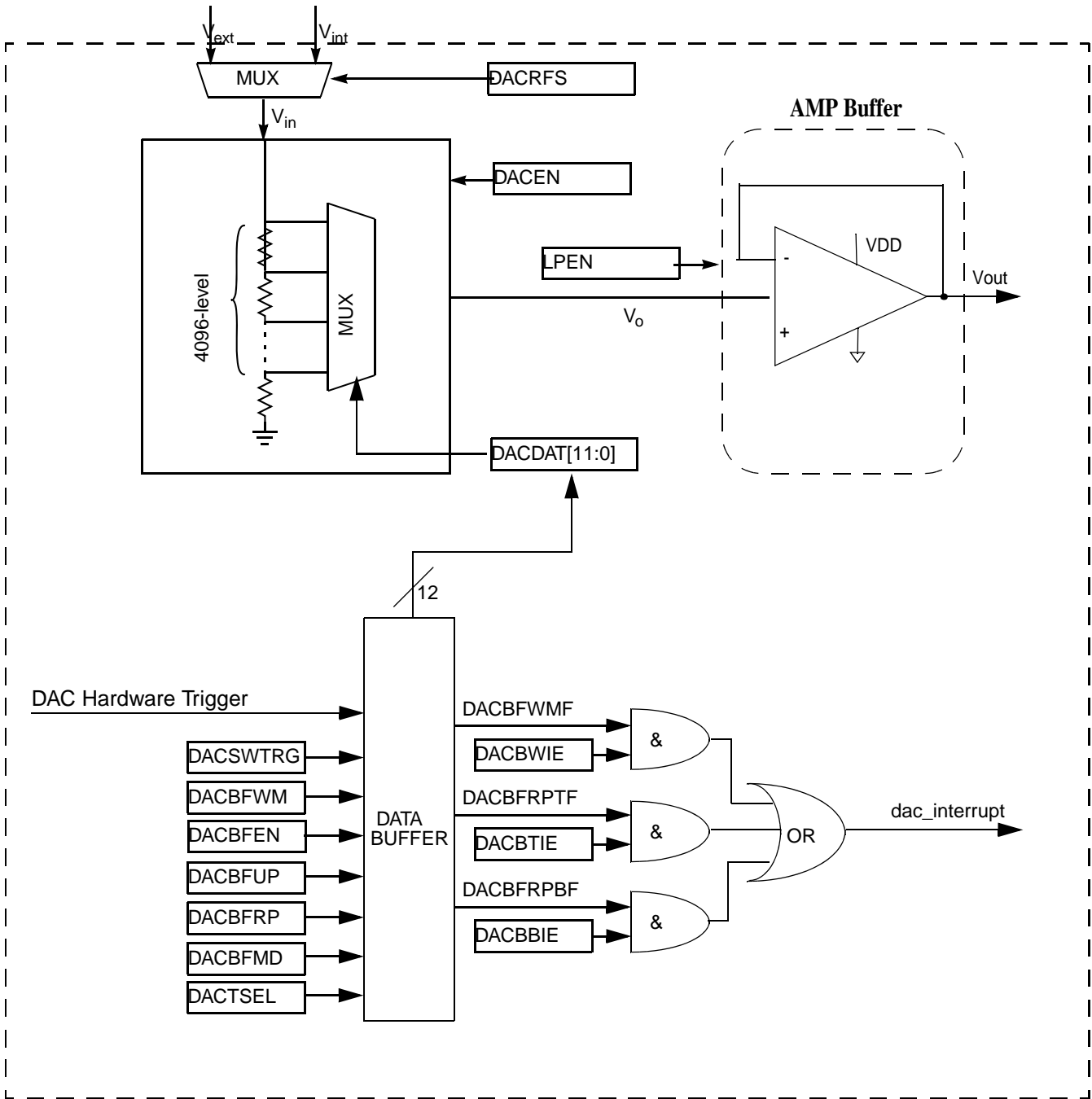
Figure 13-2. DAC Block Diagram

13.2   Register Definition

The DAC has 20 8-bit registers to control analog comparator and programmable voltage divider to perform the Digital to Analog functions.

13.2.1 DAC Data Register x (DACDATxH:DACDATxL)

	7	6	5	4	3	2	1	0
R								
W				DACDATx[7:0]				
Reset	0	0	0	0	0	0	0	0
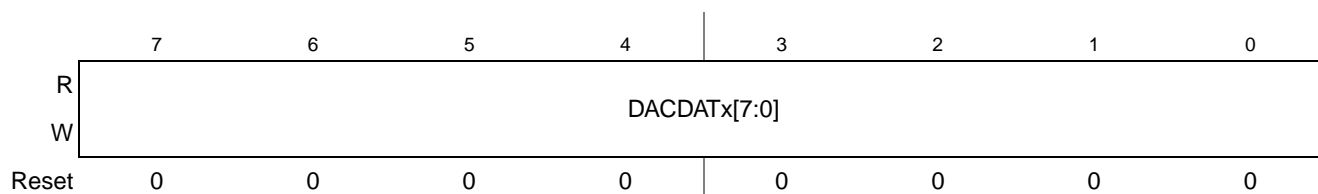
Figure 13-3. DAC Data Register x Low (DACDATxL)

	7	6	5	4	3	2	1	0
R								
W					DACDATx[11:8]			
Reset	0	0	0	0	0	0	0	0
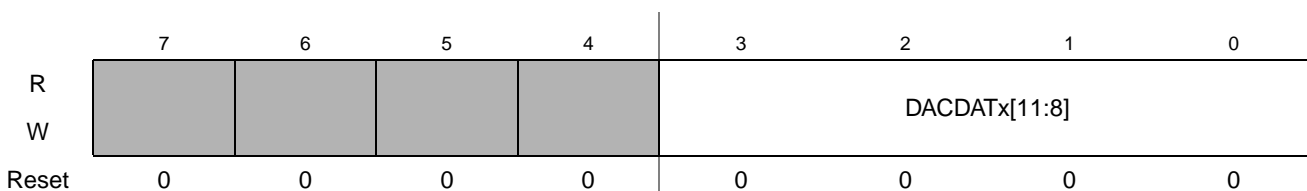
Figure 13-4. DAC Data Register x High (DACDATxH)

When the DAC Buffer is not enabled, DACDAT0[11:0] controls the output voltage based on the following formula.

$$Vout = Vin * (1+DACDAT0[11:0])/4096 \qquad \textit{Eqn. 13-1}$$

When the DAC Buffer is enabled, DACDATx[11:0] is mapped to the 16-word buffer. Refer to 3.1 "DAC Buffer Operation" for details.

When writing a new value to DAC Data Register x (DACDATxH:DACDATxL), the write order should be DACDATxH first, then DACDATxL. Otherwise the value will not be properly loaded to the effective DAC hard block and a mismatch between the DAC output may be observed. Reading DACDATx as the write-value does not mean the value is properly loaded to the DAC hard block. Therefore, the write sequence of high byte first then low byte must be followed.

13.2.2 DAC Status Register (DACS)

	7	6	5	4	3	2	1	0
R						DACWM	DACRPT	DACRPB
W						0	0	0
Reset	0	0	0	0	0	0	1	0

Figure 13-5. DAC Status Register

NOTE

If one of the flag is set, it must be cleared by software, or it will keep it's value. Write zero value to the status register, the relevant bit (s) will be cleared. It's no effect to write 1 to this register. After reset, DACRPT is set, it can be cleared by software if needed. The flag registers can be set only when the data buffer status is changed.

Table 13-1. DAC Status Register

Field	Description
7:3	Reserved
2 DACWM	DAC buffer watermark flag. 0 The DAC buffer read pointer doesn't reach the watermark level 1 The DAC buffer read pointer reaches the watermark level
1 DACRPT	DAC buffer read pointer top position flag. 0 The DAC buffer read pointer is not zero 1 The DAC buffer read pointer is zero
0 DACRPB	DAC buffer read pointer bottom position flag. 0 The DAC buffer read pointer isn't equal to the DACBFUP 1 The DAC buffer read pointer is equal to the DACBFUP.

13.2.3   DAC Control Register (DACC0)

	7	6	5	4	3	2	1	0
R W	DACEN	DACRFS	DACTSEL	DACSTRG	LPEN	DACWIE	DACTIE	DACBIE
Reset	0	0	0	0	0	0	0	0
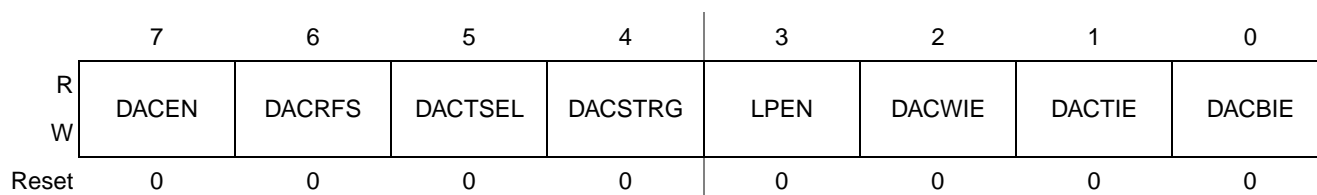
Figure 13-6. DAC Control Register 0 (DACC0)

Table 13-2. DACC0 Field Descriptions

Field	Description
7 DACEN	DAC enable — The DACEN bit starts the Programmable Reference Generator operation. 0 The DAC system is disabled. 1 The DAC system is enabled.
6 DACRFS	DAC Reference Select 0 The DAC selects V_{int} as the reference voltage. 1 The DAC selects V_{ext} as the reference voltage.
5 DACTSEL	DAC trigger select 0 The DAC hardware trigger is selected. 1 The DAC software trigger is selected.
4 DACSTRG	DAC software trigger — active high. This is a write-only bit; it is always "0" when read. If the DAC software trigger is selected and buffer enabled, writing a 1 to this bit advances the buffer read pointer once. 0 The DAC soft trigger is not valid. 1 The DAC soft trigger is valid.
3 LPEN	DAC low power control 0 High power mode. 1 Low power mode.
2 DACWIE	DAC buffer watermark interrupt enable. 0 The DAC buffer watermark interrupt is disabled. 1 The DAC buffer watermark interrupt is enabled.

Table 13-2. DACC0 Field Descriptions (Continued)

Field	Description
1 DACTIE	DAC buffer read pointer top flag interrupt enable. 0 The DAC buffer read pointer top flag interrupt is disabled. 1 The DAC buffer read pointer top flag interrupt is enabled.
0 DACBIE	DAC buffer read pointer bottom flag interrupt enable. 0 The DAC buffer read pointer bottom flag interrupt is disabled. 1 The DAC buffer read pointer bottom flag interrupt is enabled.

13.2.4 DAC Control Register1 (DACC1)

	7	6	5	4	3	2	1	0
R				DACBFWM		DACBFMD		DACBFE
W								
Reset	0	0	0	0	0	0	0	0
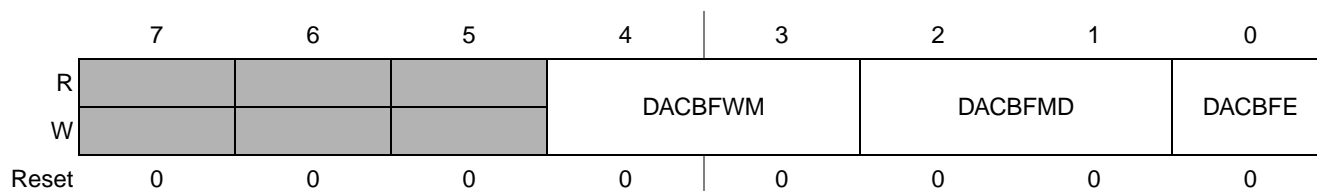
Figure 13-7. DAC Control Register 1 (DACC1)

Table 13-3. DACC1 Field Descriptions

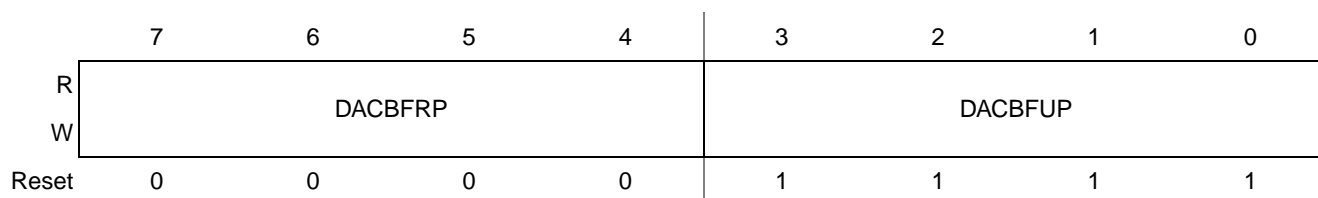| Field	Description
7:3	Reserved
4:3 DACBFWM	**DAC Buffer Watermark Select** — When the word number between the read pointer and the upper address is equal to DACBFWM, the DACWM bit in status register will be set. DACWM can be used to inform the software need to refresh the DAC buffer. 00 1 word 01 2 words 10 3 words 11 4 words
2:1 DACBFMD	**DAC Buffer Work Mode Selec**t 00 Normal mode 01 Swing mode 10 One-time scan mode 11 Reserved
0 DACBFE	**DAC Buffer Enable** 0 Buffer read pointer disabled. The converted data is always the first word of the buffer. 1 Buffer read pointer enabled. The converted data is the word that the read pointer points to. It means converted data can be from any word of the buffer.

13.2.5   DAC Control Register 2 (DACC2)

	7	6	5	4	3	2	1	0
R		DACBFRP				DACBFUP		
W								
Reset	0	0	0	0	1	1	1	1

Figure 13-8. DAC Control Register 2 (DACC2)

Table 13-4. DACC2 Field Descriptions

Field	Description
7:4 DACBFRP	**DAC Buffer Read Pointer** — These 4 bits keep the current value of the buffer read pointer
3:0 DACBFUP	**DAC Buffer Upper Limit** — These 4 bits select the buffer's upper limit. The buffer read pointer cannot exceed it.

13.3   Functional Description

The 12-bit Digital-to-Analog Convertor (DAC12LVLP) module can select one of two reference inputs V_{in1} and V_{in2} as the DAC reference voltage (V_{in}) by DACRFS bit of DACC0 register. Refer to the module introduction for information on the source for V_{ext} and V_{int}. When the DAC12LVLP is enabled, it converts the data in DACDAT0[11:0] or the data from the DAC data buffer to a stepped analog output voltage. The output voltage range is from $V_{in}/4096$ to V_{in}, and the step is $V_{in}/4096$.

13.3.1   DAC Data Buffer Operation

When the DAC is enabled and the buffer isn't enabled, the DAC12LVLP module always converts the data in DACDAT0 to analog output voltage.

When both the DAC and the buffer are enabled, the DAC12LVLP converts the data in the data buffer to analog output voltage. The data buffer read pointer advances to the next word in the event the hardware trigger or the software trigger occurs. Refer to the DAC12LVLP Introduction section for the hardware trigger connection. The data buffer can be configured to operate in normal mode, swing mode, or one-time scan mode. When the buffer operation is switched from one mode to another, the read pointer doesn't change. The read pointer can be set to any value between 0 and DACBFUP by writing DACBFRP in DACC2.

13.3.1.1   Buffer Normal Mode

This is the default mode. The buffer works as a circular buffer. The read pointer increases by one every time when the trigger occurs. When the read pointer reaches the upper limit, it goes to the zero directly in the next trigger event.

13.3.1.2   Buffer Swing Mode

This mode is similar to the Normal mode. But when the read pointer reaches the upper limit, it doesn't go to the zero. It will descend by one in the next trigger events until zero is reached.

13.3.1.3   Buffer One-time Scan Mode

The read pointer increases by one every time the trigger occurs. When it reaches the upper limit, it stops. If the read pointer is reset to an address other than the upper limit, it will increase to the upper address and then stop.

NOTE

If the software set the read pointer to the upper limit, the read pointer will not advance in this mode.

13.3.2   Resets

During reset, the DAC12LVLP is configured in the default mode. DAC12LVLP is disabled.

13.3.3   Low Power Mode Operation

13.3.3.1   Wait Mode Operation

In wait mode, the DAC12LVLP operates normally if enabled.

13.3.3.2   Stop Mode Operation

The DAC12LVLP continues to operate in stop3 mode if enabled, the output voltage will hold the value before STOP.

In stop2 mode, the DAC12LVLP is fully shut down.

13.3.4   Background Mode Operation

When the MCU is in active background mode, the DAC12LVLP operates normally.

Chapter 14
Inter-Integrated Circuit (S08IICV3)

14.1 Introduction

The inter-integrated circuit (IIC) provides a method of communication between a number of devices. The interface is designed to operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

NOTE
The SDA and SCL should not be driven above V_{DD}. These pins are pseudo open-drain and contain a protection diode to V_{DD}.

14.1.1 Module Configuration

The IIC module pins, SDA and SCL can be repositioned under software control using IICPS in SOPT3 as as shown in Table 14-1. IICPS in SOPT3 selects which general-purpose I/O ports are associated with IIC operation.

Table 14-1. IIC Position Options

IICPS in SOPT3	Port Pin for SDA	Port Pin for SCL
0 (default)	PTD4	PTD5
1	PTF4	PTF3

14.1.2 IIC Clock Gating

The bus clock to the IIC can be gated on and off using the IIC bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the IIC bit can be cleared to disable the clock to this module when not in use. See Section 5.7.8, "System Clock Gating Control 1 Register (SCGC1)," for details.
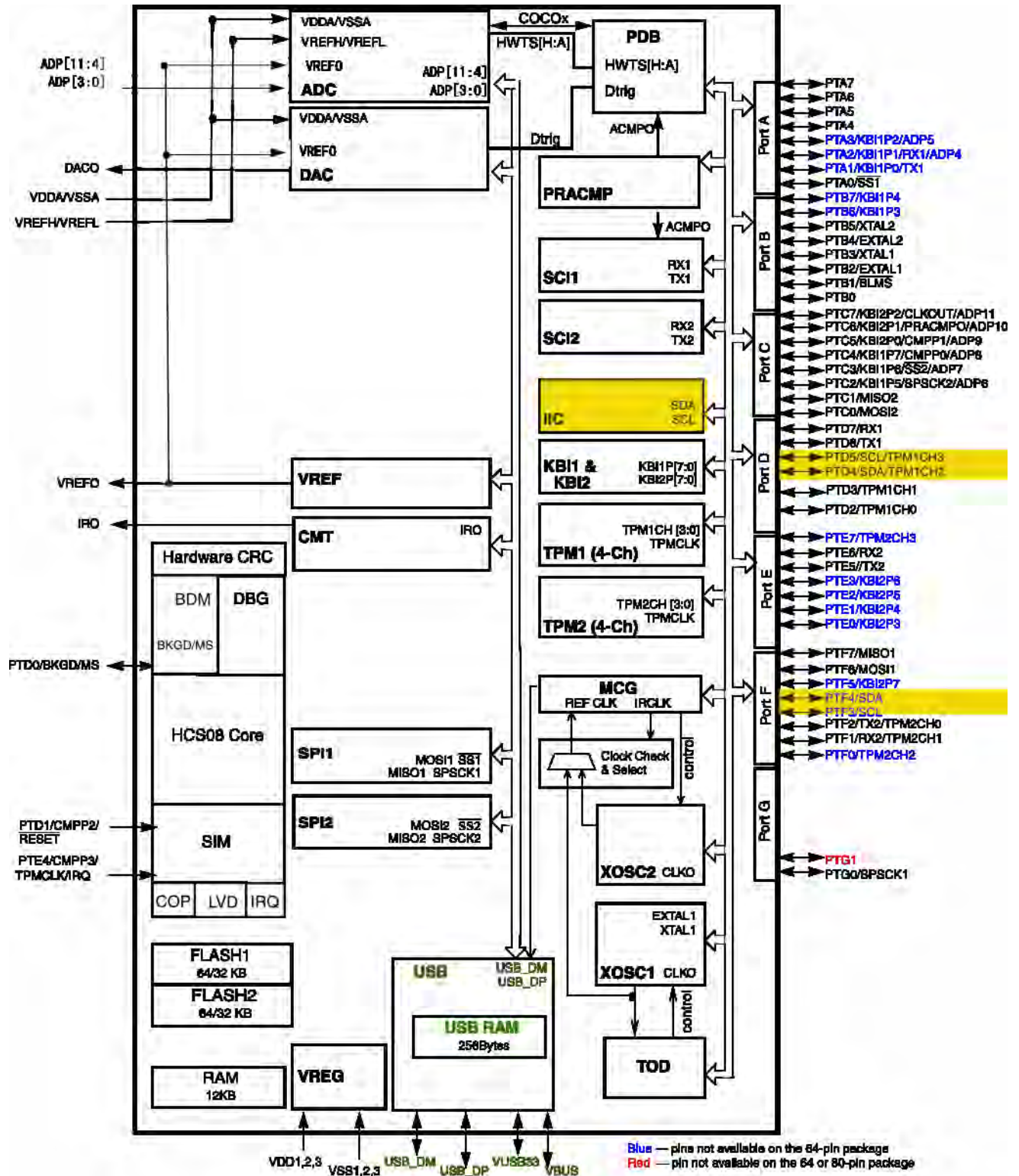
Figure 14-1. Block Diagram Highlighting the IIC Module

Supports System Management Bus Specification (SMBus), version 2.

14.1.3   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/detection
- Acknowledge bit generation/detection
- Bus busy detection
- General call recognition
- 10-bit address extension
- Support System Management Bus Specification(SMBus), version2
- Programmable glitch input filter

14.1.4   Modes of Operation

A brief description of the IIC in the various MCU modes follows:

- **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 will reset the register contents.

14.1.5 Block Diagram

Figure 14-2 is a block diagram of the IIC.

Figure 14-2. IIC Functional Block Diagram

14.2 External Signal Description

This section describes each user-accessible pin signal.

14.2.1 SCL — Serial Clock Line

The bidirectional SCL is the serial clock line of the IIC system.
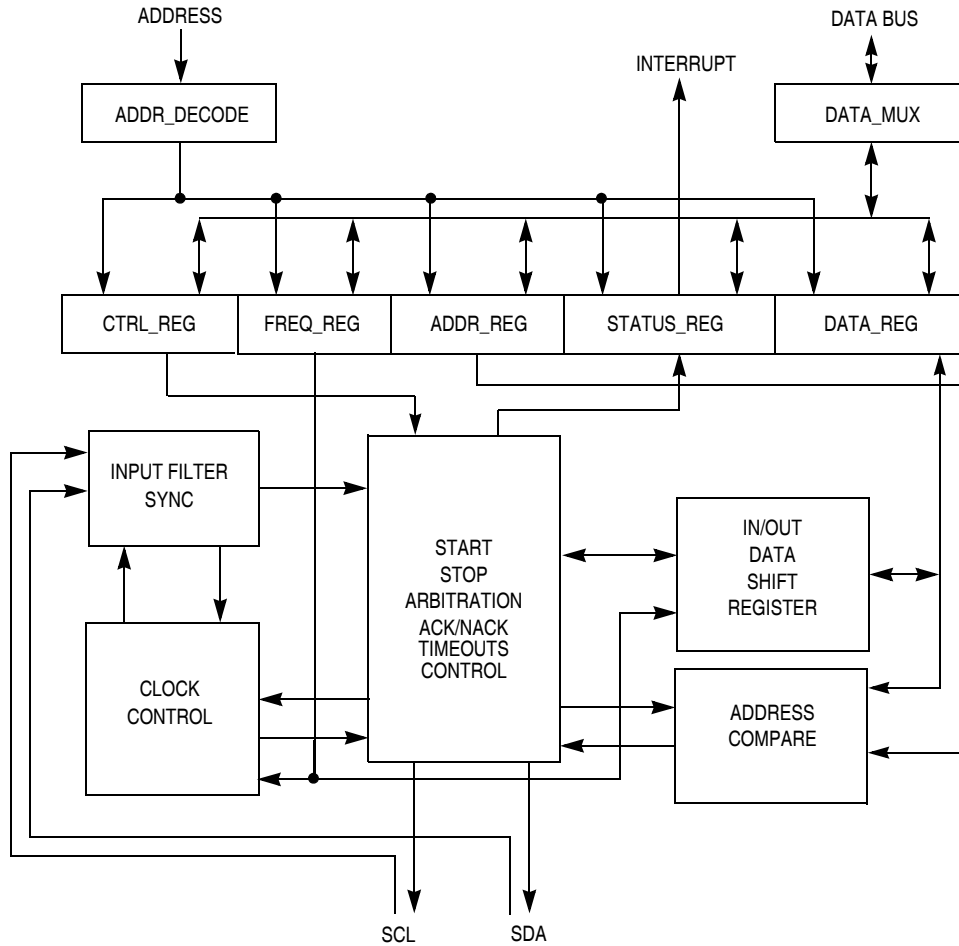
14.2.2 SDA — Serial Data Line

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

14.3 Register Definition

14.3.1 Module Memory Map

The IIC has ten 8-bit registers. The base address of the module is hardware programmable. The IIC register map is fixed and begins at the module's base address. Table 14-2 summarizes the IIC module's address space. The following section describes the bit-level arrangement and functionality of each register.

Table 14-2. Module Memory Map

Address	Use	Access
Base + $0000	IIC Address Register 1 (IICA1)	read/write
Base + $0001	IIC Frequency Divider Register (IICF)	read/write
Base + $0002	IIC Control Register 1 (IICC1)	read/write
Base + $0003	IIC Status Register (IICS)	read
Base + $0004	IIC Data IO Register (IICD)	read/write
Base + $0005	IIC Control Register 2 (IICC2)	read/write
Base + $0006	IIC input programmable filter (IICFLT)	read/write
Base + $0006	SMBUS IIC Control and Status Register (IICSMB)	read/write
Base + $0007	IIC Address Register 2 (IICA2)	read/write
Base + $0008	IIC SCL Low Time Out Register High (IICSLTH)	read/write
Base + $0009	IIC SCL Low Time Out Register Low (IICSLTL)	read/write
Base + $000A	IIC input programmable filter (IICFLT)	read/write

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.

NOTE

If SMBus is selected, then the IICFLT's address is $000A, or it should be $0006.

14.3.2 IIC Address Register 1 (IICA1)

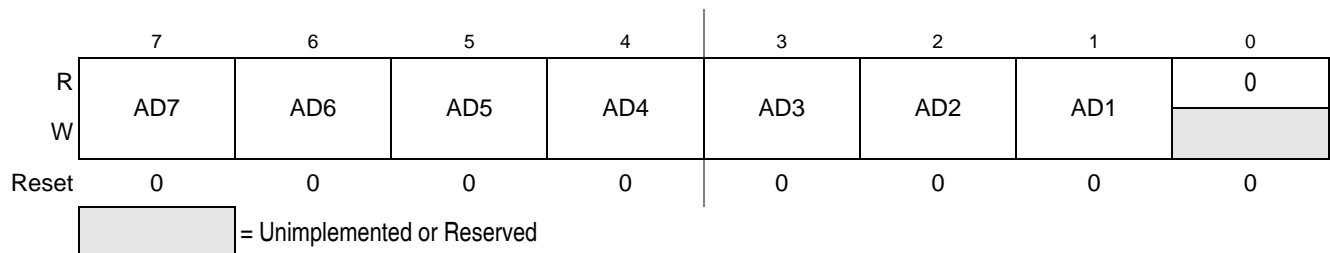|	7	6	5	4	3	2	1	0
R	AD7	AD6	AD5	AD4	AD3	AD2	AD1	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 14-3. IIC Address Register 1 (IICA1)

Table 14-3. IICA1 Field Descriptions

Field	Description
7:1 AD[7:1]	**Slave Address 1**— 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.

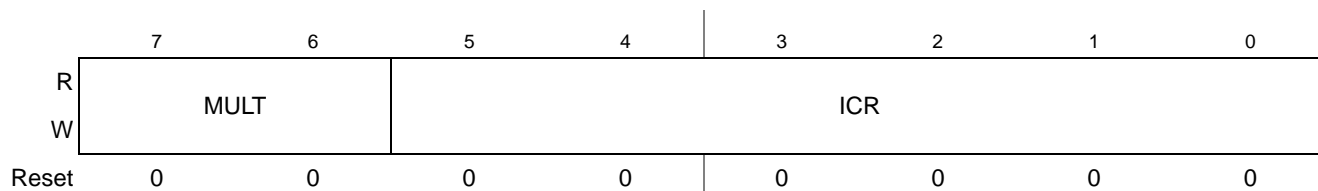## 14.3.3 IIC Frequency Divider Register (IICF)

	7	6	5	4	3	2	1	0
R		MULT				ICR		
W								
Reset	0	0	0	0	0	0	0	0

Figure 14-4. IIC Frequency Divider Register (IICF)

Table 14-4. IICF Field Descriptions

Field	Description
7:6 MULT	**IIC Multiplier Factor** — The MULT bits define the multiplier factor mul. This factor is used along with the SCL divider to generate 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 are used to determine the IIC baud rate, the SDA hold time, the SCL Start hold time and the SCL Stop hold time. Table 14-5 provides the SCL divider and hold values for corresponding values of the ICR. The SCL divider multiplied by multiplier factor mul is used to generate IIC baud rate. **IIC baud rate = bus speed (Hz)/(mul * SCL divider)** *Eqn. 14-1* 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. 14-2* 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. 14-3* 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. 14-4*

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

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 14-5. IIC Divider and Hold Values

ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SCL Hold (Stop) Value	ICR (hex)	SCL Divider	SDA Hold Value	SCL Hold (Start) Value	SCL Hold (Stop) Value
00	20	7	6	11	20	160	17	78	81
01	22	7	7	12	21	192	17	94	97
02	24	8	8	13	22	224	33	110	113
03	26	8	9	14	23	256	33	126	129
04	28	9	10	15	24	288	49	142	145
05	30	9	11	16	25	320	49	158	161
06	34	10	13	18	26	384	65	190	193
07	40	10	16	21	27	480	65	238	241
08	28	7	10	15	28	320	33	158	161
09	32	7	12	17	29	384	33	190	193
0A	36	9	14	19	2A	448	65	222	225
0B	40	9	16	21	2B	512	65	254	257
0C	44	11	18	23	2C	576	97	286	289
0D	48	11	20	25	2D	640	97	318	321
0E	56	13	24	29	2E	768	129	382	385
0F	68	13	30	35	2F	960	129	478	481
10	48	9	18	25	30	640	65	318	321
11	56	9	22	29	31	768	65	382	385
12	64	13	26	33	32	896	129	446	449
13	72	13	30	37	33	1024	129	510	513
14	80	17	34	41	34	1152	193	574	577
15	88	17	38	45	35	1280	193	638	641
16	104	21	46	53	36	1536	257	766	769
17	128	21	58	65	37	1920	257	958	961
18	80	9	38	41	38	1280	129	638	641
19	96	9	46	49	39	1536	129	766	769
1A	112	17	54	57	3A	1792	257	894	897
1B	128	17	62	65	3B	2048	257	1022	1025
1C	144	25	70	73	3C	2304	385	1150	1153
1D	160	25	78	81	3D	2560	385	1278	1281
1E	192	33	94	97	3E	3072	513	1534	1537
1F	240	33	118	121	3F	3840	513	1918	1921

14.3.4 IIC Control Register (IICC1)

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

= Unimplemented or Reserved
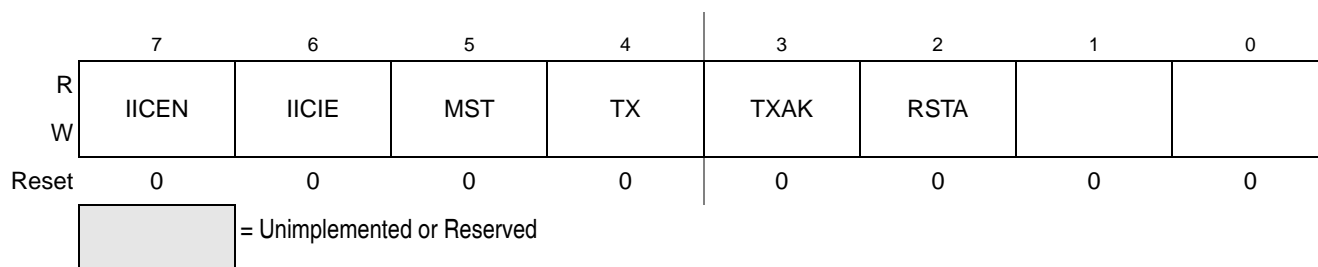
Figure 14-5. IIC Control Register (IICC1)

Table 14-6. IICC1 Field Descriptions

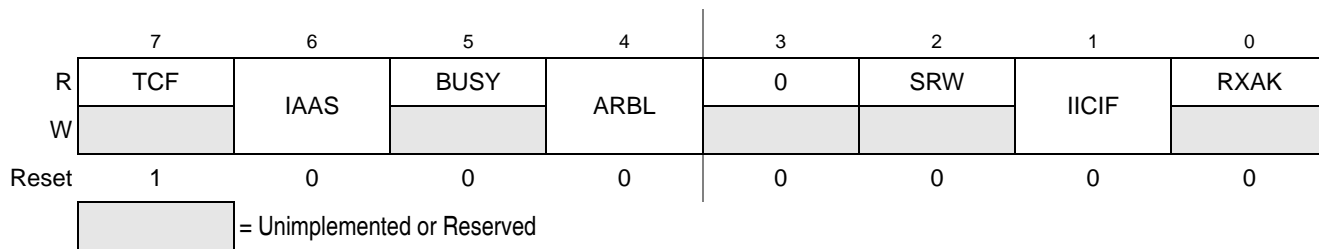| Field	Description
7 IICEN	**IIC Enable** — The IICEN bit determines whether the IIC module is enabled. 0 IIC is not enabled. 1 IIC is enabled.
6 IICIE	**IIC Interrupt Enable** — The IICIE bit determines whether an IIC interrupt is requested. 0 IIC interrupt request not enabled. 1 IIC interrupt request enabled.
5 MST	**Master Mode Select** — When the MST bit is changed from a 0 to a 1, 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 will always be 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 both master and slave receivers. There are two conditions will effect NAK/ACK generation. If FACK (fast NACK/ACK) is cleared, 0 An acknowledge signal will be sent out to the bus on the following receiving data byte. 1 No acknowledge signal response is sent to the bus on the following receiving data byte. If FASK bit is set. no ACK or NACK is sent out after receiving one data byte until this TXAK bit is written 0 An acknowledge signal will be sent out to the bus on the current receiving data byte 1 No acknowledge signal response is sent to the bus on the current receiving data byte Note: SCL is held to low until TXAK is written.
2 RSTA (Write Only read always 0)	**Repeat START** — Writing a 1 to this bit will generate a repeated START condition provided it is the current master. Attempting a repeat at the wrong time will result in loss of arbitration. 0 No repeat start detected in bus operation. 1 Repeat start generated.

14.3.5   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 14-6. IIC Status Register (IICS)

Table 14-7. IICS Field Descriptions

Field	Description
7 TCF	**Transfer Complete Flag** — This bit is set on the completion of a byte and acknowledge bit transfer. Note that 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 one of the following conditions is met 1) When the calling address matches the programmed slave address, 2) If the GCAEN bit is set and a general call is received. 3) If SIICAEN bit is set, when the calling address matches the 2nd programmed slave address 4) If ALERTEN bit is set and SMBus alert response address is received This bit is set before ACK bit. The CPU needs to check the SRW bit and set TX/RX bit accordingly. Writing the IICC1 register with any value 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/$\overline{W}$ command bit of the calling address sent to the master. 0 Slave receive, master writing to slave. 1 Slave transmit, master reading from slave.

Table 14-7. IICS Field Descriptions (Continued)

Field	Description
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 including ACK/NACK bit completes if FACK = 0 • One byte transfer including ACK/NACK bit completes if FACK = 1 and this byte is an address byte • One byte transfer excluding ACK/NCAK bit completes if FACK = 1 and this byte is a data byte. an ACK or NACK is sent out on the bus by writing 0 or 1 to TXAK after this bit is set. • Match of slave addresses to calling address (Primary Slave address, General Call address, Alert Response address, and Second Slave address) (Received address is stored in data register) • Arbitration lost • Timeouts in SMBus mode except high timeout 0 No interrupt pending. 1 Interrupt pending. NOTE: The IICIF will be cleared right after 1 bus cycle delay after writing a logic 1 to it.
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.

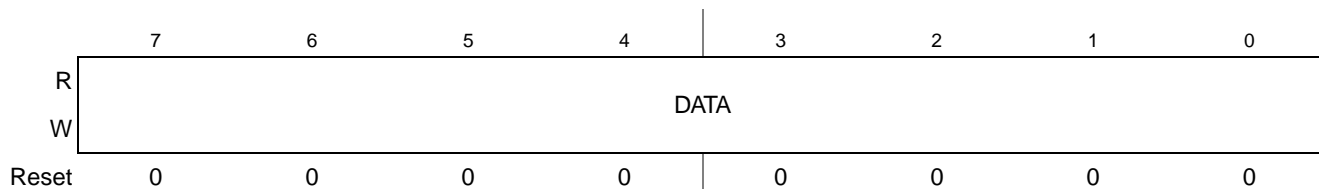## 14.3.6   IIC Data I/O Register (IICD)

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

Figure 14-7. IIC Data I/O Register (IICD)

Table 14-8. 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.

Note that 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, then reading the IICD will not initiate the receive.

Reading the IICD will return the last byte received while the IIC is configured in either master receive or slave receive modes. The IICD does not reflect every byte that is 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 (Start bit) or assertion of RSTA bit (repeated Start) is used for the address transfer and should comprise of the calling address (in bit 7 to bit 1) concatenated with the required R/$\overline{\text{W}}$ bit (in position bit 0).
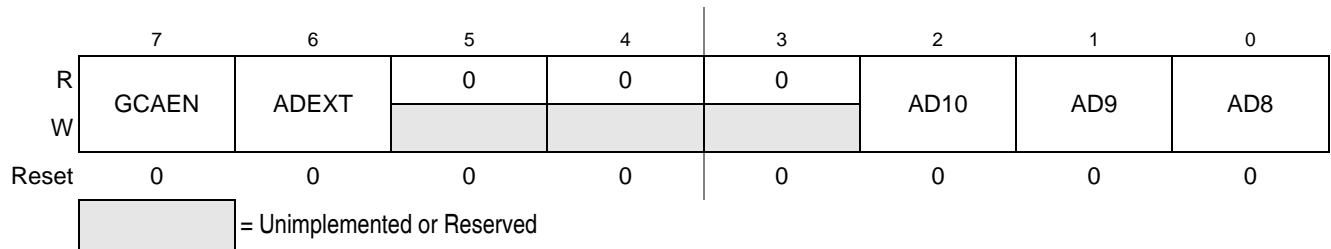
14.3.7 IIC Control Register 2 (IICC2)

	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 14-8. IIC Control Register (IICC2)

Table 14-9. 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.

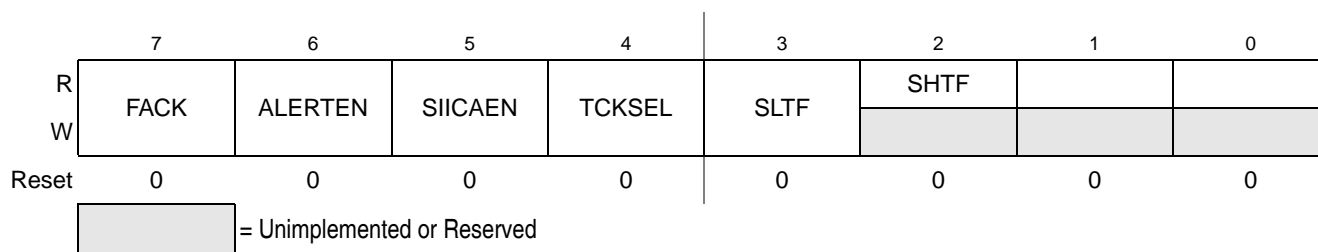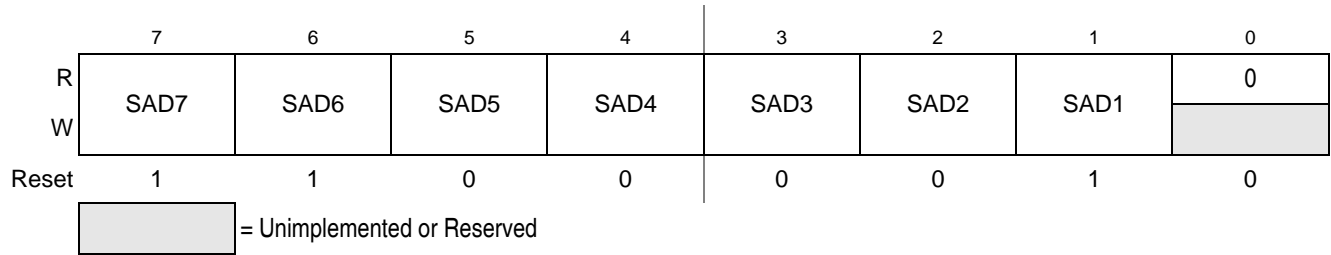## 14.3.8 IIC SMBus Control and Status Register (IICSMB)

	7	6	5	4	3	2	1	0
R	FACK	ALERTEN	SIICAEN	TCKSEL	SLTF	SHTF		
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Table 14-10. IICSMB Field Descriptions

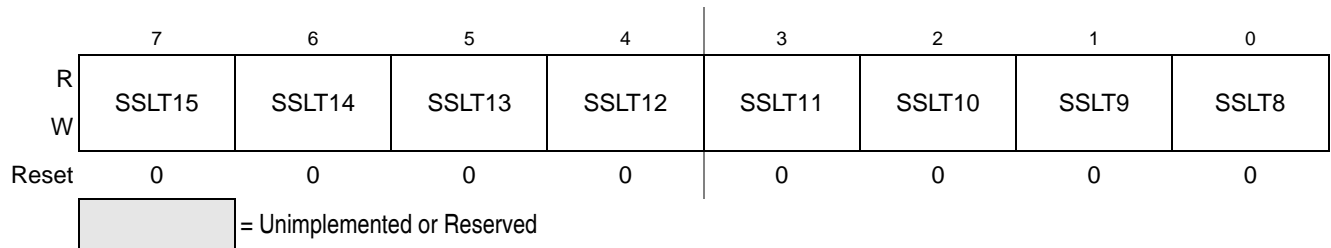| Field	Description
7 FACK	**Fast NACK/ACK enable** — For SMBus Packet Error Checking, CPU should be able to issue an ACK or NACK according to the result of receiving data byte. 0 ACK or NACK will be sent out on the following receiving data byte. 1 Writing an 0 to TXAK after receiving data byte will generate an ACK; Writing an 1 to TXAK after receiving data byte will generate a NACK
6 ALERTEN	**SMBus Alert Response Address Enable** — The ALERTEN bit enables or disable SMBus alert response address. 0 SMBus alert response address matching is disabled 1 SMBus alert response address matching is enabled.
5 SIICAEN	**Second IIC Address Enable** — The SIICAEN bit enables or disable SMBus device default address. 0 IIC Address Register 2 matching is disabled. 1 IIC Address Register 2 matching is enabled.
4 TCKSEL	Time Out Counter Clock Select— This bit selects the clock sources of Time Out Counter 0 Time Out Counter counts at bus/64 frequency 1 Time Out Counter counts at the bus frequency
3 SLTF	SCL Low Timeout Flag — This read-only bit is set to logic 1 when IICSLT loaded non zero value (LoValue) and a SCL Low Time Out occurs. This bit is cleared by software, by writting a logic 1 to it or IICIF. 0 No LOW TIME OUT occurs. 1 A LOW TIME OUT occurs. Note: LOW TIME OUT function is disabled when IIC SCL LOW TIMER OUT register is set to zero
2 SHTF	SCL High Timeout Flag — This read-only bit is set to logic 1 when SCL and SDA are held high more than clock * LoValue/512, which indicates the Bus Free. This bit is cleared automatically. 0 No HIGH TIMEOUT occurs. 1 An HIGH TIMEOUT occurs.

NOTE

1. A master can assume that the bus is free if it detects that the clock and data signals have been high for greater than THIGH,MAX, however, the SHTF will rise in bus transmission process but bus idle state.

2. When TCKSEL=1 there is no meaning to monitor SHTF since the bus speed is too high to match the protocol of SMBus.
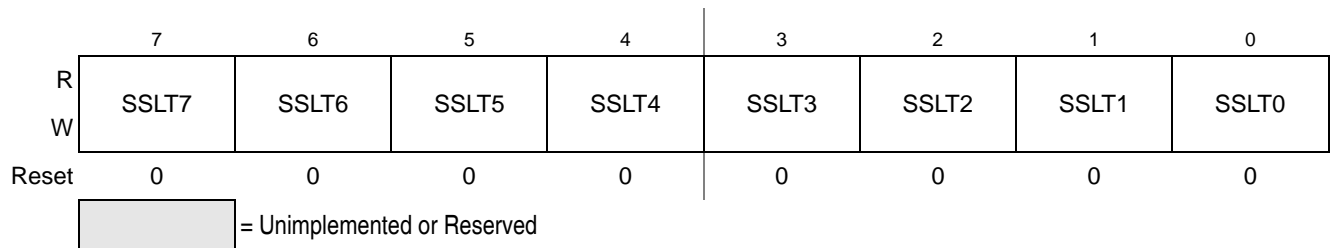
14.3.9 IIC Address Register 2 (IICA2)

	7	6	5	4	3	2	1	0
R	SAD7	SAD6	SAD5	SAD4	SAD3	SAD2	SAD1	0
W								
Reset	1	1	0	0	0	0	1	0

= Unimplemented or Reserved

Field	Description
7:1 SAD[7:1]	**SMBUs Address** — The AD[7:1] field contains the slave address to be used by the SMBus. This field is used on the device default address or other related address

14.3.10 IIC SCL Low Time Out Register High (IICSLTH)

	7	6	5	4	3	2	1	0
R	SSLT15	SSLT14	SSLT13	SSLT12	SSLT11	SSLT10	SSLT9	SSLT8
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Field	Description
7:0 SSLT[15:8]	The value in this register is the most significant byte of SCL low time out value that determines the time-out period of SCL low.

14.3.11 IIC SCL LowTime Out register Low (IICSLTL)

	7	6	5	4	3	2	1	0
R	SSLT7	SSLT6	SSLT5	SSLT4	SSLT3	SSLT2	SSLT1	SSLT0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

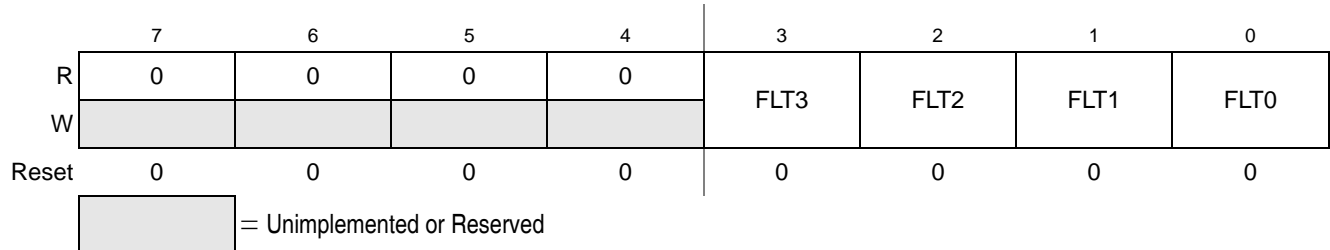| Field	Description
7:0 SSLT[7:0]	The value in this register is the least significant byte of SCL low time out value that determines the time-out period of SCL low..

14.3.12 IIC Programmable Input Glitch Filter (IICFLT)

	7	6	5	4	3	2	1	0
R	0	0	0	0	FLT3	FLT2	FLT1	FLT0
W					FLT3	FLT2	FLT1	FLT0
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Table 14-11. IICFLT Field Descriptions

Field	Description
3:0 FLT	IIC Programmable Filter Factor contain the programming controls for the width of glitch (in terms of bus clock cycles) the filter should absorb; in other words, the filter does not let glitches less than or equal to this width setting pass. FLT[3:0] 0000 No Filter / Bypass 0001 Filter glitches up to width of 1 (half) IPBUS clock cycle 0010 Filter glitches up to width of 2 (half) IPBUS clock cycles 0011 Filter glitches up to width of 3 (half) IPBUS clock cycles 0100 Filter glitches up to width of 4 (half) IPBUS clock cycles 0101 Filter glitches up to width of 5 (half) IPBUS clock cycles 0110 Filter glitches up to width of 6 (half) IPBUS clock cycles 0111 Filter glitches up to width of 7 (half) IPBUS clock cycles 1000 Filter glitches up to width of 8 (half) IPBUS clock cycle 1001 Filter glitches up to width of 9 (half) IPBUS clock cycles 1010 Filter glitches up to width of 10 (half) IPBUS clock cycles 1011 Filter glitches up to width of 11 (half) IPBUS clock cycles 1100 Filter glitches up to width of 12 (half) IPBUS clock cycles 1101 Filter glitches up to width of 13 (half) IPBUS clock cycles 1110 Filter glitches up to width of 14 (half) IPBUS clock cycles 1111 Filter glitches up to width of 15 (half) IPBUS clock cycles NOTE: The width of the FLT is an integration option which can be changed in different SoCs And Also the clock source used is an integration configurative option - It could be the 2X IPBus clock or the IPbus clock - which one needs to be identified at architectural definition. For the 4-bit definitions above, hard descriptions of "half" IPBUS clock cycles will not be the case when the IPBUS clock is used for filtering logic.

14.4 Functional Description

This section provides a complete functional description of the IIC module.

14.4.1 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 14-9.
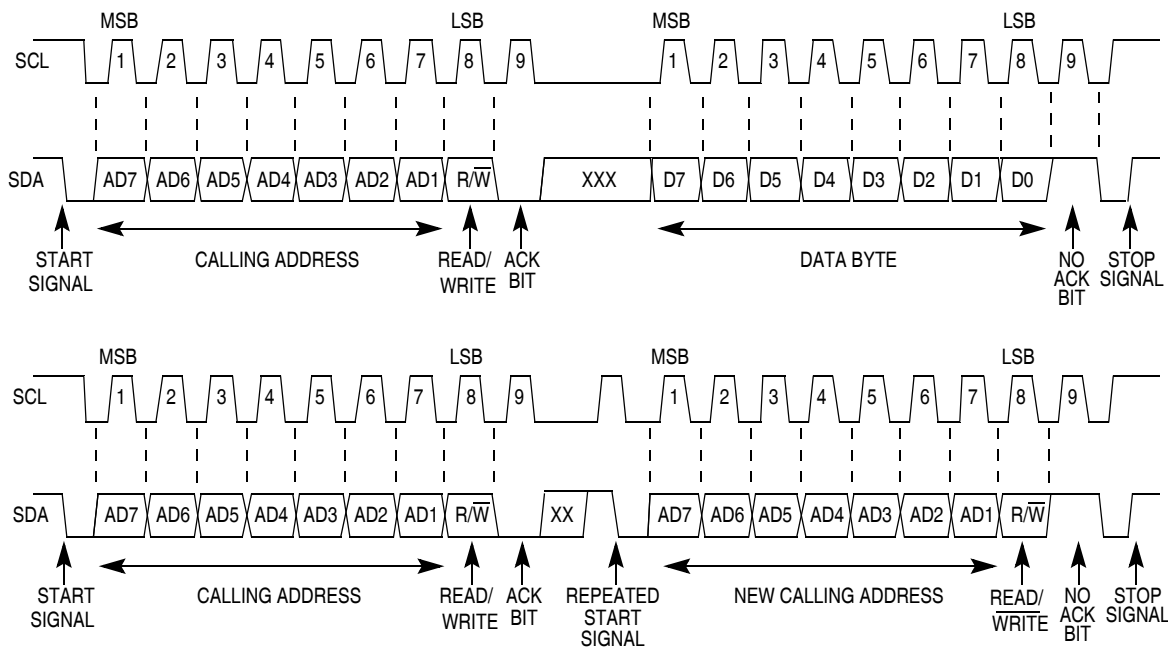

Figure 14-9. IIC Bus Transmission Signals

14.4.1.1 START Signal

When the bus is free; in other words, no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in Figure 14-9, 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.

14.4.1.2 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/$\overline{\text{W}}$ bit. The R/$\overline{\text{W}}$ bit tells the slave the desired direction of data transfer.

  1 = Read transfer, the slave transmits data to the master.
  0 = Write transfer, the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 14-9).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address that is 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 will revert to slave mode and operate correctly even if it is being addressed by another master.

14.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/$\overline{\text{W}}$ 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 14-9. 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 9th 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.

14.4.1.4 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 14-9).

The master can generate a STOP even if the slave has generated an acknowledge at which point the slave must release the bus.

14.4.1.5 Repeated START Signal

As shown in Figure 14-9, 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.

14.4.1.6 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.

14.4.1.7 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 14-10). 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.
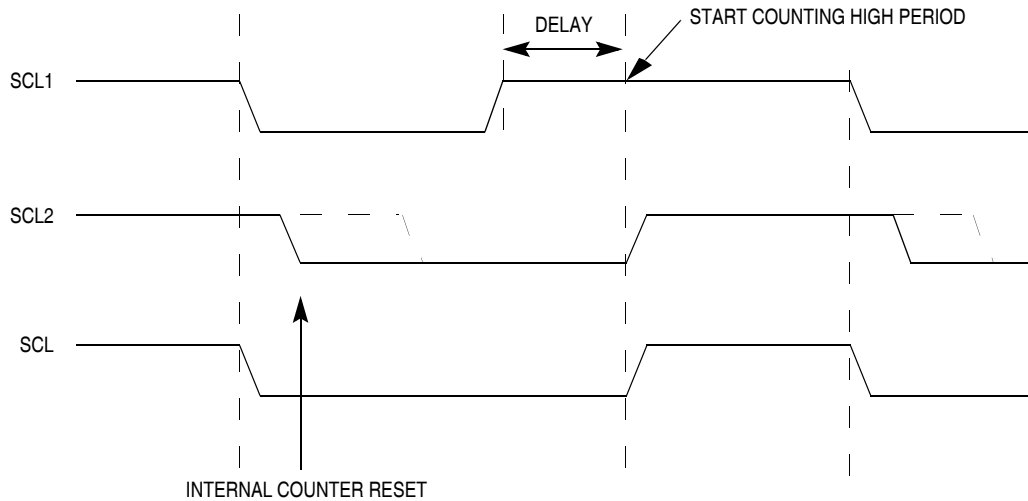
Figure 14-10. IIC Clock Synchronization

14.4.1.8 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 case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

14.4.1.9 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.

14.4.2   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.

14.4.2.1   Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 14-12). 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/$\overline{\text{W}}$ direction bit) is 0. It is possible that more than one device will find a match and generate an acknowledge (A1). Each slave that finds a match will compare the eight bits of the second byte of the slave address with its own address, but only one slave will find a match and generate an acknowledge (A2). The matching slave will remain 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 14-12. 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 will see an IIC interrupt. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as valid data.

14.4.2.2   Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/$\overline{\text{W}}$ bit (see Table 14-13). 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/$\overline{\text{W}}$) 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 will also compare the first seven bits of the first byte of the slave address with their own addresses and test the eighth (R/$\overline{\text{W}}$) bit. However, none of them will be addressed because R/$\overline{\text{W}}$ = 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 14-13. 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 will see an IIC interrupt. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as valid data.

14.4.3　Address Matching

All received Addresses can be requested in 7-bit or 10-bit address. IIC Address Register 1, which contains IIC primary slave address, always participates the address matching process. If the GCAEN bit is set, general call will participate the address matching process. If the ALERTEN bit is set, alert response will participate the address matching process. If SIICAEN bit is set, the IIC Address Register 2 will participate the address matching process.

When the IIC responds to one of above mentioned address, it acts as a slave-receiver and the IAAS bit is set after the address cycle. Software need to read the IICD register after the first byte transfer to determine which the address is matched.

14.4.4　System Management Bus Specification

SMBus provides a control bus for system and power management related tasks. A system may use SMBus to pass messages to and from devices instead of tripping individual control lines. Removing the individual control lines reduces pin count. Accepting messages ensures future expandability.With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status.

14.4.4.1　Timeouts

The $T_{TIMEOUT,MIN}$ parameter allows a master or slave to conclude that a defective device is holding the clock low indefinitely or a master is intentionally trying to drive devices off the bus. It is highly recommended that a slave device release the bus (stop driving the bus and let SCL and SDA float high) when it detects any single clock held low longer than $T_{TIMEOUT,MIN}$. Devices that have detected this condition should reset their communication and be able to receive a new START condition in no later than $T_{TIMEOUT,MAX}$.

SMBus defines a clock low time-out, $T_{TIMEOUT}$ of 35 ms and specifies $T_{LOW: SEXT}$ as the cumulative clock low extend time for a slave device and specifies $T_{LOW: MEXT}$ as the cumulative clock low extend time for a master device.

14.4.4.1.1　SCL Low Timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than a "timeout" value condition. Devices that have detected the timeout condition must reset the communication. When active master, if the IIC detects that SMBCLK low has exceeded the value of $T_{TIMEOUT,MIN}$ it must generate a stop condition within or after the current data byte in the transfer process. When slave, upon detection of the $T_{TIMEOUT,MIN}$ condition, the IIC shall reset its communication and be able to receive a new START condition.

14.4.4.1.2 SCL High (SMBus Free) Timeout

The IIC shall assume that the bus is idle, when it has determined that the SMBCLK and SMBDAT signals have been high for at least $T_{HIGH:MAX}$. HIGH timeout can occur in two ways: 1) HIGH timeout detected after a STOP condition appears on the bus; 2) HIGH timeout detected after a START condition, but before a STOP condition appears on the bus. Any master detecting either scenario can assume the bus is free then SHTF rises. HIGH timeout occurred in scenario 2 if it ever detects that both the following is true: BUSY bit is high and SHTF is high.

14.4.4.1.3 CSMBCLK TIMEOUT MEXT

Figure1-10: Timeout measurement intervals illustrates the definition of the timeout intervals, $T_{LOW:SEXT}$ and $T_{LOW:MEXT}$. When master mode, the I2C must not cumulatively extend its clock cycles for a period greater than $T_{LOW:MEXT}$ within a byte, where each byte is defined as START-to-ACK, ACK-to-ACK, or ACK-to-STOP. When CSMBCLK TIMEOUT MEXT occurs SMBus MEXT will rise and also trigger the SLTF.

14.4.4.1.4 CSMBCLK TIMEOUT SEXT

A Master is allowed to abort the transaction in progress to any slave that violates the $T_{LOW:SEXT}$ or $T_{TIMEOUT,MIN}$ specifications. This can be accomplished by the Master issuing a STOP condition at the conclusion of the byte transfer in progress. When slave, the I2C must not cumulatively extend its clock cycles for a period greater than $T_{LOW:SEXT}$ during any message from the initial START to the STOP. When CSMBCLK TIMEOUT SEXT occurs SEXT will rise and also trigger SLTF.
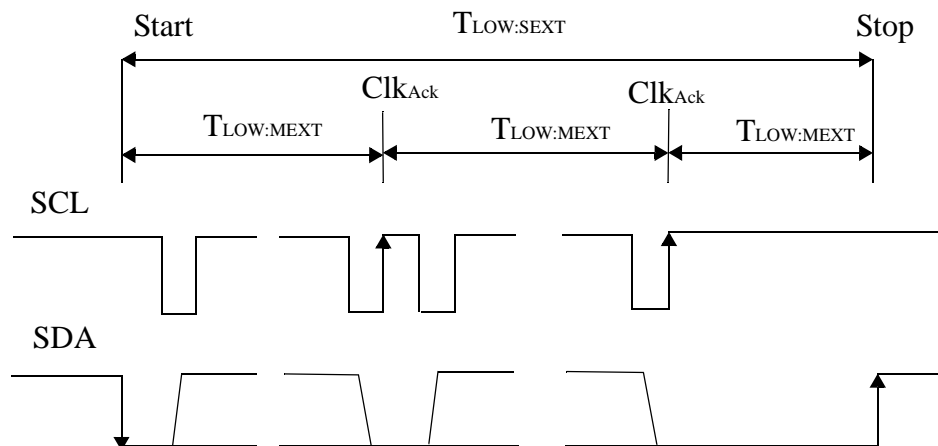

Figure 14-11. Timeout measurement intervals

NOTE

CSMBCLK TIMEOUT SEXT and MEXT are optional functions which will be implemented in second step.

14.4.4.2   FAST ACK and NACK

To improve reliability and communication robustness, implementation of Packet Error Checking (PEC) by SMBus devices is optional for SMBus devices but required for devices participating in and only during the Address Resolution Protocol (ARP) process. The PEC is a CRC-8 error checking byte, calculated on all the message bytes. The PEC is appended to the message by the device that supplied the last data byte. If the PEC is present but not correct, a NACK is issued by receiver. Otherwise an ACK will be issued. In order to calculate the CRC-8 by software, this module can hold SCL line to low after receiving eighth SCL (bit 8th) if this byte is a data byte. So software can determine whether an ACK or NACK should be sent out to the bus by setting or clearing TXAK bit if FASK (fast ACK/NACK enable bit) is enabled.

SMBus requires devices to acknowledge their own address always, as a mechanism to detect a removable devices presence on the bus (battery, docking station, etc.) Besides to indicate a slave device busy condition, SMBus is using the NACK mechanism also to indicate the reception of an invalid command or data. Since such a condition may occur on the last byte of the transfer, it is required that SMBus devices have the ability to generate the not acknowledge after the transfer of each byte and before the completion of the transaction. This is important because SMBus does not provide any other resend signaling. This difference in the use of the NACK signaling has implications on the specific implementation of the SMBus port, especially in devices that handle critical system data such as the SMBus host and the SBS components.

NOTE

In the last byte of master receive slave transmit mode, the master should send NACK to bus so FACK should be switched off before the last byte transmit.

14.5   Resets

The IIC is disabled after reset. The IIC cannot cause an MCU reset.

14.6   Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in Table 14-14 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. The user can determine the interrupt type by reading the status register. For SMBus timeouts interrupt, the interrupt is driven by SLTF and masked with bit IICIE. The SLTF bit must be cleared by software by

writing a 1 to it in the interrupt routine. The user can determine the interrupt type by reading the status register.

NOTE

In Master receive mode, the FACK should be set zero before the last byte transfer.

Table 14-14. 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
SMBus Timeout Interrupt Flag	SLTF	IICIF	IICIE

14.6.1 Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the 9th clock to indicate the completion of byte transfer.

14.6.2 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.

14.6.3 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 by writing a 1 to it.

14.6.4 Timeouts Interrupt in SMbus

When IICIE is set, the IIC asserts a timeout interrupt output SLTF upon detection of any of the mentioned timeout conditions, with one exception. The HIGH TIMEOUT mechanism shall not be used to influence

the timeout interrupt output, because the HIGH TIMEOUT indicates an idle condition on the bus.And SLTF will rise when it matches the HIGH TIMEOUT and fall automatically to just indicate the bus status.

14.6.5   Programmable input glitch filter

An IIC glitch filter has been added outside the IIC legacy modules, but within the IIC package. This filter can absorb glitches on the IIC clock and data lines for I2C module. The width of the glitch to absorb can be specified in terms of number of half bus clock cycles. A single glitch filter control register is provided as IICFLT. Effectively, any down-up-down or up-down-up transition on the data line that occurs within the number of clock cycles programmed here is ignored by the IIC.the programmer only needs to specify the size of glitch (in terms of bus clock cycles) for the filter to absorb and not pass.

14.7 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: IICA1
   — 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 14-12

**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 14-12
5. Write: IICC1
   — to enable TX
6. Write: IICC1
   — to enable MST (master mode)
7. Write: IICD
   — with the address of the target slave. (The LSB of this byte will determine whether the communication is master receive or transmit.)

Module Use

The routine shown in Figure 14-12 can handle both master and slave IIC operations. For slave operation, an incoming IIC message that contains the proper address will begin IIC communication. For master operation, communication must be initiated by writing to the IICD register.

Register Model

IICA1	AD[7:1]							0

Address to which the module will respond when addressed as a slave (in slave mode)

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

IICFLT	0	0	0	0	FLT3	FLT2	FLT1	FLT0

IIC Programmable Input Glitch Filter

Figure 14-12. Typical IIC Interrupt Routine
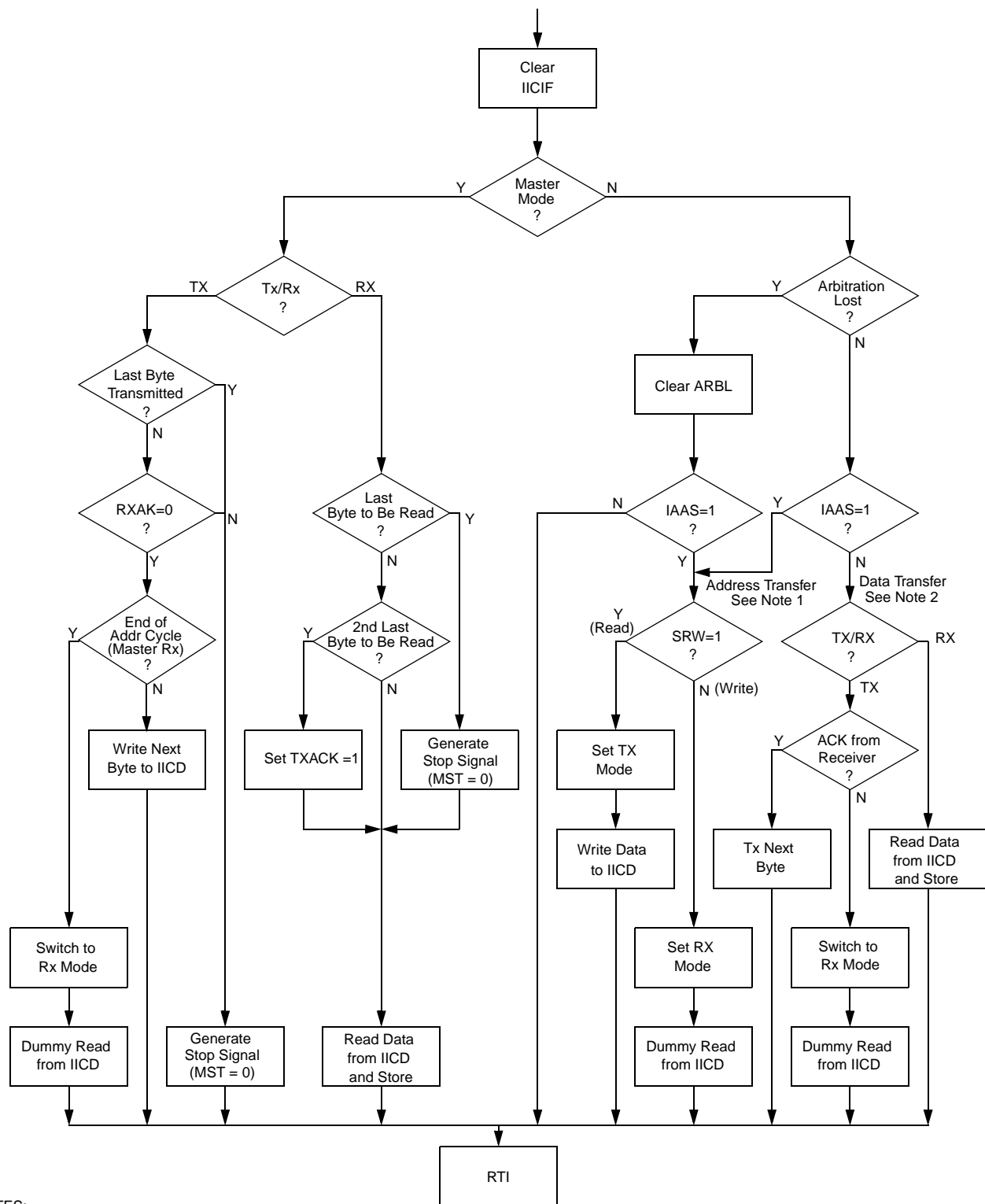
NOTES:
1. If general call is enabled, a check must be done to determine whether the received address was a general call address (0x00). If the received address was a general call address, then the general call must be handled by user software.
2. When 10-bit addressing is used to address a slave, the slave will see an interrupt following the first byte of the extended address. User software must ensure that for this interrupt, the contents of IICD are ignored and not treated as a valid data transfer.

Figure 14-14. Typical IIC SMBus Interrupt Routine

NOTES:
1. If general call siicaen is enabled, a check must be done to determine whether the received address was a general call address (0x00) or SMbus device default address. If the received address was one of them, then it must be handled by user software.
2. Flow chart1 means figure 1-12 Typical IIC Interrupt Routine.
3. Delay about 1-2bit scl cycle waiting data register updated then clear IICIF

14.8   SMBALERT#

Another optional signal is an interrupt line for devices that want to trade their ability to master for a pin. SMBALERT# is a wired-AND signal just as the SMBCLK and SMBDAT signals are. SMBALERT# is used in conjunction with the SMBus General Call Address. Messages invoked with the SMBus are 2 bytes long. (There is no ALERT# port in the current block).

A slave-only device can signal the host that it wants to talk through SMBALERT#. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the Alert Response Address (ARA). Only the device(s) which pulled SMBALERT# low will acknowledge the Alert Response Address.

The host performs a modified Receive Byte operation. The 7 bit device address provided by the slave transmit device is placed in the 7 most significant bits of the byte. The eighth bit can be a zero or one.

If more than one device pulls SMBALERT# low, the highest priority (lowest address) device will win communication rights via standard arbitration during the slave address transfer. After acknowledging the slave address, the device must disengage its SMBALERT# pulldown. If the host still sees SMBALERT# low when the message transfer is complete, it knows to read the ARA again.

A host which does not implement the SMBALERT# signal may periodically access the ARA.

S	Alert Response Address	Rd	A	Device Address	A	P

Figure 14-15. A 7-bit-Addressable Device Responds to an ARA

NOTE

The user should put Device Address on bus by software after response to the Alert response address in current block.

Chapter 15
Multipurpose Clock Generator (S08MCGV3)

15.1   Introduction

The multipurpose clock generator (MCG) module provides several clock source choices for this device. The module contains a frequency-locked loop (FLL) and a phase-locked loop (PLL) that are controllable by an internal or an external reference clock. 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. The selected clock source is passed through a reduced bus divider that allows a lower output clock frequency to be derived.

For USB operation on the MC9S08JE128 series series, the MCG must be configured for PLL engaged external (PEE) mode to achieve a MCGOUT frequency of 48 MHz.

NOTE

MCGPLLSCLK is not used on the MC9S08JE128 series.

TheMC9S08JE128 series devices are unique in that they support a Time Of Day module which includes a dedicated oscillator. The TOD oscillator can also be used as the reference clock into the MCG.
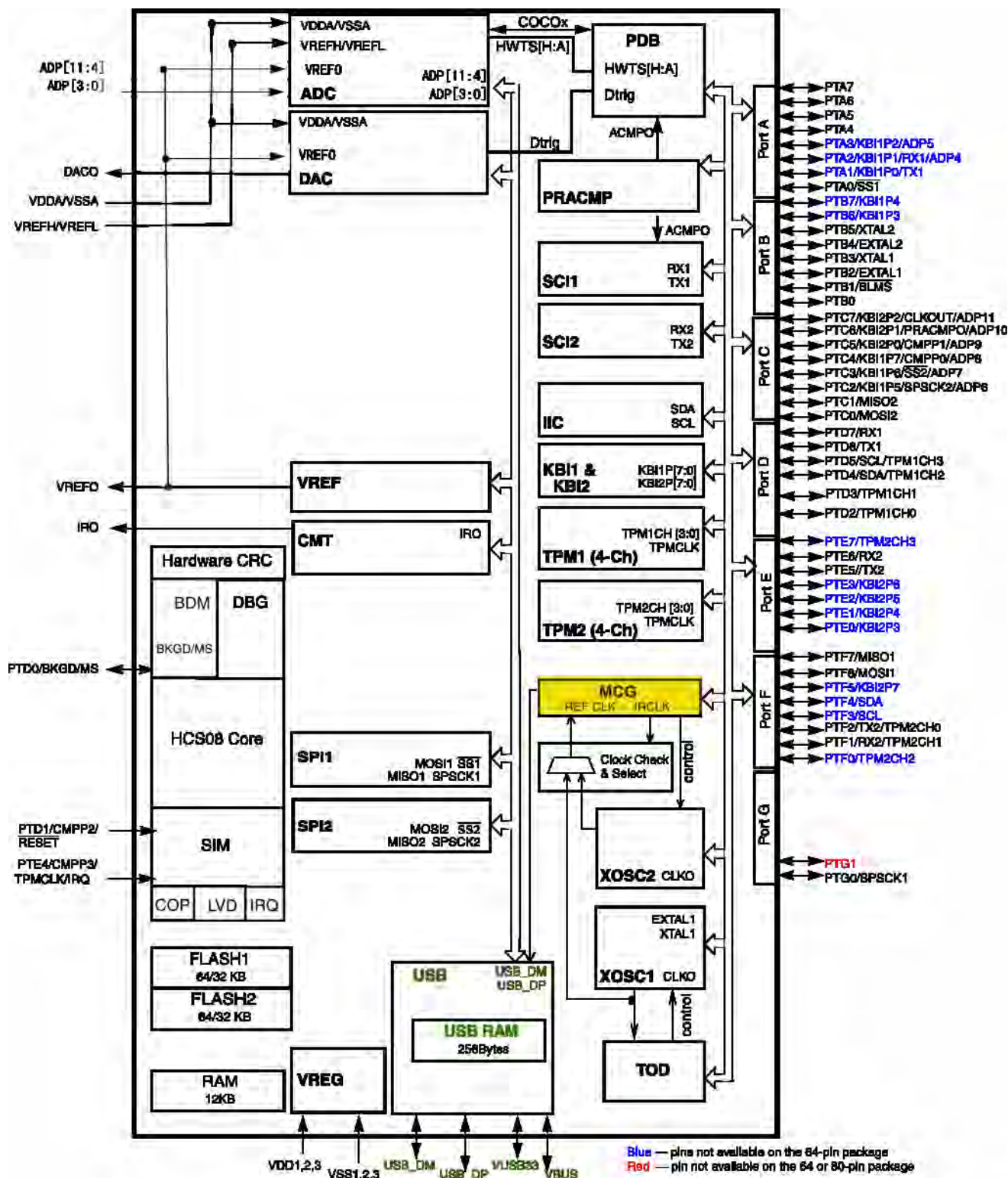
Figure 15-1. Block Diagram Highlighting the MCG Module

15.1.1 Clock Check & Select Function

The "Clock Check & Select" feature of Figure 1-2 and Figure 15-2 is a very simple block used to check the oscillator clocks for activity and control the mux which selects the external clock for the MCG. This function is implemented external to the MCG module itself, and is specific to the MC9S08JE128 series device.

Four registers make up this operation:

Table 15-1. Clock Check & Select Registers

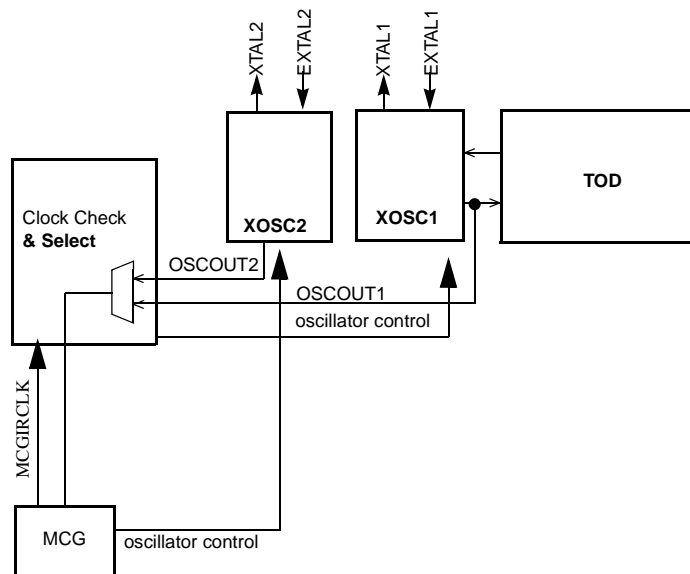| Register	Function
CCSCTRL	Clock Check & Select Control Register
CCSTMR1	8-bit counter incremented by XOSC1 timebase
CCSTMR2	8-bit counter incremented by XOSC2 timebase
CCSTMRIR	8-bit counter incremented via IR clock timebase

Figure 15-2. A subset of Figure 1-2
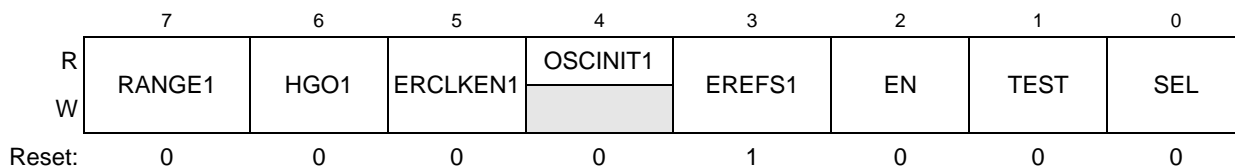
15.1.2 Clock Check & Select Control (CCSCTRL)

	7	6	5	4	3	2	1	0
R	RANGE1	HGO1	ERCLKEN1	OSCINIT1	EREFS1	EN	TEST	SEL
W								
Reset:	0	0	0	0	1	0	0	0

Figure 15-3. Clock Check & Select Control Register (CCSCTRL)

Table 15-2. CSS Control Register Field Descriptions

Field	Description
7 RANGE1	**RANGE1** — selects the frequency range for the external oscillator 1 (XOSC1). 0 Low frequency range selected for the external oscillator. 1 High frequency range selected for the external oscillator.
6 HGO1	**HGO1** — controls the external oscillator 1 (XOSC1) mode of operation. 0 Configures external oscillator for low-power operation. 1 Configures external oscillator for high gain operation.
5 ERCLKEN1	**ERCLKEN1** — 0 MCGERCLK inactive. 1 MCGERCLK active.
4 OSCINIT1	**OSCINIT1** — If ERESF1 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 EREFS1 or ERCLKEN1 is cleared. 0 XOSC1 Initialization is not complete or not enabled as an oscillator. 1 XOSC1 Initialization is complete and its output is stable.
3 EREFS1	**EREFS1** — selects the source for the external reference clock of the MCG when using XOSC1. 0 External Clock Source requested. 1 Oscillator requested.
2 EN	**EN** — 0 The OSCOUT1, OSCOUT2 and MCGIRCLK inputs to the clock check counters are disabled for power saving. 1 The clock inputs are enabled.
1 TEST	**TEST—** Writing a "1" to this bit will clear CSSTMR1, CSSTMR2 and CSSTMRIR. The three counters will then begin incrementing until any one of the three hits 0xFF, at which point the test completes, and TEST is cleared.
0 SEL	**External Clock Select** — 0 XOSC2 is selected as the external clock input to the MCG (default) 1 XOSC1 is selected as the external clock input to the MCG This bit should only be changed when the MCG is NOT utilizing the external clock input.

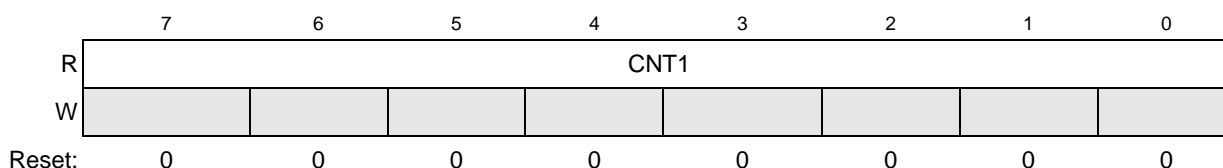## 15.1.3 CSS XOSC1 Timer Register (CCSTMR1)

	7	6	5	4	3	2	1	0
R				CNT1				
W								
Reset:	0	0	0	0	0	0	0	0

Figure 15-4. CSS XOSC1 Timer Register (CSSTMR1)

Table 15-3. CSSTMR1 Register Field Descriptions

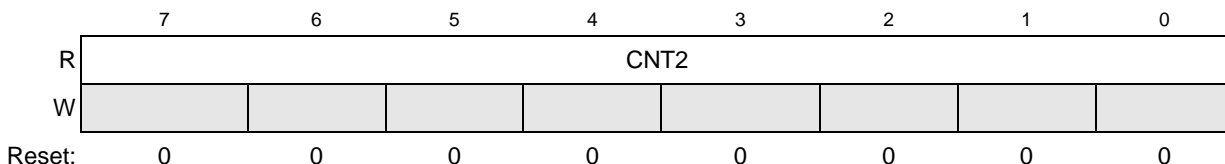| Field	Description
7:0 CNT1	**CNT1** — This register is one of three used to compare XOSC1, XOSC2 and internal relaxation oscillator frequencies. It is initialized to zero upon a write of "1" to CSSCTRL[TEST]. It contains a valid value once CSSCTRL[TEST] resets itself to "0". By comparing the values of the three registers, application code can determine the crude health of the various clock sources.

15.1.4 CSS XOSC2 Timer Register (CCSTMR2)

	7	6	5	4	3	2	1	0
R				CNT2				
W								
Reset:	0	0	0	0	0	0	0	0

Figure 15-5. CSS XOSC2 Timer Register (CSSTMR2)

Table 15-4. CSSTMR2 Register Field Descriptions

Field	Description
7:0 CNT2	**CNT2** — This register is one of three used to compare XOSC1, XOSC2 and internal relaxation oscillator frequencies. It is initialized to zero upon a write of "1" to CSSCTRL[TEST]. It contains a valid value once CSSCTRL[TEST] resets itself to "0". By comparing the values of the three registers, application code can determine the crude health of the various clock sources.

15.1.5 CSS Internal Reference Clock Timer Register (CCSTMRIR)

	7	6	5	4	3	2	1	0
R				CNTIR				
W								
Reset:	0	0	0	0	0	0	0	0
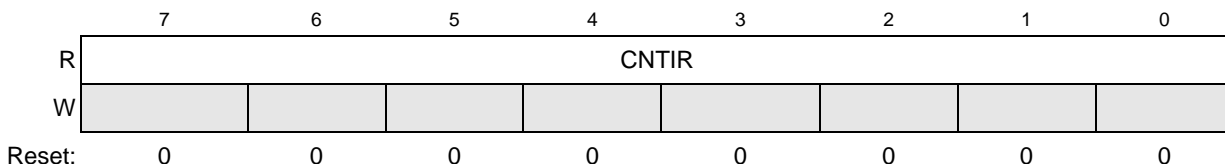
Figure 15-6. CSS Internal Reference Clock Timer Register (CSSTMRIR)

Table 15-5. CSSTMRIR Register Field Descriptions

Field	Description
7:0 CNTIR	**CNTIR** — This register is one of three used to compare XOSC1, XOSC2 and internal relaxation oscillator frequencies. It is initialized to zero upon a write of "1" to CSSCTRL[TEST]. It contains a valid value once CSSCTRL[TEST] resets itself to "0". By comparing the values of the three registers, application code can determine the crude health of the various clock sources.

15.1.6 Operation

The clock check and select (CCS) feature is a basic clock monitor of the internal reference clock and two XOSC clocks using the following 8-bit counters:

- CCSTMR1
- CCSTMR2
- CCSTMRIR

The major purpose of this feature is to monitor the stability and availability of the XOSC clocks and provide a rough frequency measurement of the three clock sources. The registers are 8-bit so the maximum count is 255.

Basically, whichever counter reaches 255 first sends a stop signal to the other two counters. It takes approximately 3 clock cycles for the stop signal to sync with the other two counters. Software can then compare the three registers to obtain a crude (3/256 is ~1.2%) measurement of how well the three frequencies correlate.

NOTE

The clock check feature should only be used to check the stability and availability of the XOSC clocks. CCS is not intended to be used as an accurate frequency measurement of the clocks.

15.1.7 Features

Key features of the MCG module are:

- Frequency-locked loop (FLL)
    — Internal or external reference clock 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 a separate crystal 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.Three selectable digitally controlled oscillators (DCOs) optimized for different frequency ranges.
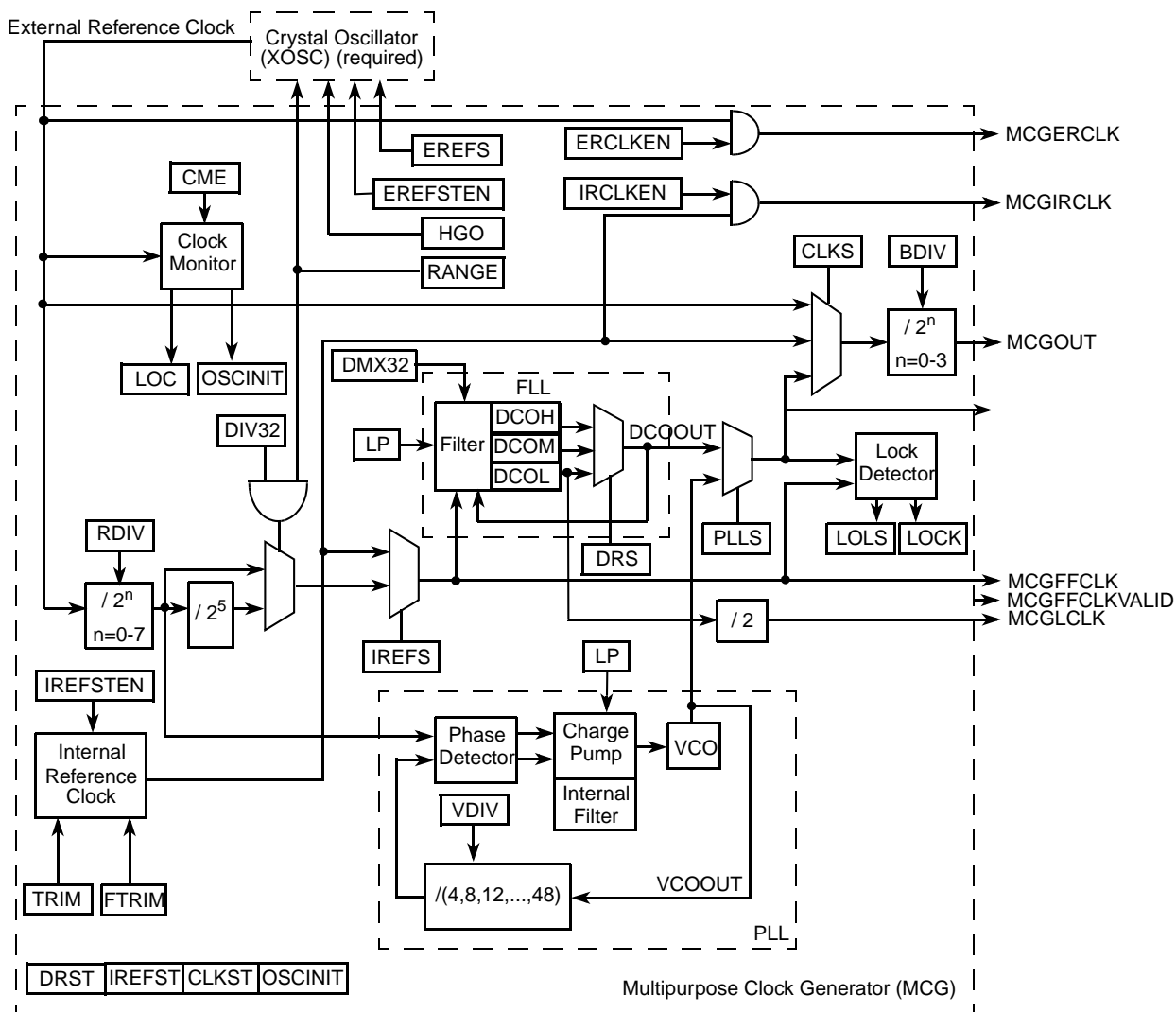- Option to maximize DCO output frequency for a 32,768 Hz external reference clock source.

Figure 15-7. Multipurpose Clock Generator (MCG) Block Diagram

NOTE

The MCG requires the attachment of a crystal oscillator (XOSC) module, which provides an external reference clock.

15.1.8 Modes of Operation

There are several modes of operation for the MCG. For details, see Section 15.4.1, "MCG Modes of Operation."

15.2 External Signal Description

There are no MCG signals that connect off chip.

15.3 Register Definition

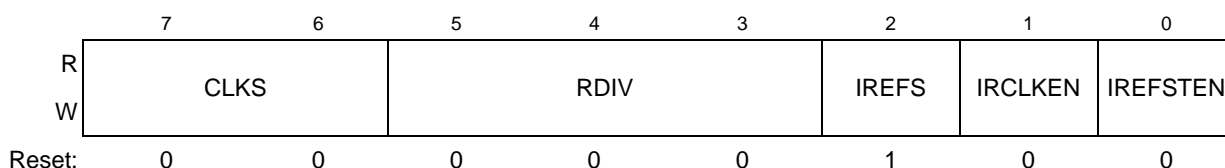### 15.3.1 MCG Control Register 1 (MCGC1)

	7	6	5	4	3	2	1	0
R	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
W								
Reset:	0	0	0	0	0	1	0	0

Figure 15-8. MCG Control Register 1 (MCGC1)

Table 15-6. 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	**External Reference Divider** — Selects the amount to divide down the external reference clock. 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. See Table 15-7 and Table 15-8 for the divide-by factors.
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

Table 15-7. FLL External Reference Divide Factor

RDIV	Divide Factor		
	RANGE:DIV32 0:X	RANGE:DIV32 1:0	RANGE:DIV32 1:1
0	1	1	32
1	2	2	64
2	4	4	128
3	8	8	256
4	16	16	512
5	32	32	1024
6	64	64	Reserved
7	128	128	Reserved

Table 15-8. PLL External Reference Divide Factor

RDIV	Divide Factor
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

15.3.2 MCG Control Register 2 (MCGC2)

	7	6	5	4	3	2	1	0
R	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
W								
Reset:	0	1	0	0	0	0	0	0
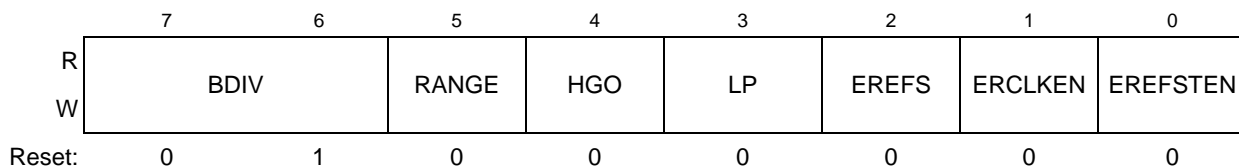
Figure 15-9. MCG Control Register 2 (MCGC2)

Table 15-9. MCG Control Register 2 Field Descriptions

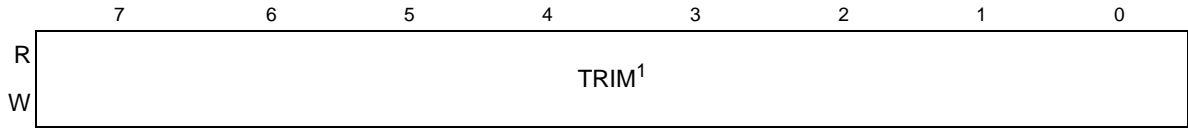| Field	Description
7:6 BDIV	**Bus Frequency Divider** — Selects the amount to divide down the clock source selected by the CLKS bits in the MCGC1 register. This controls the bus frequency. 00 Encoding 0 — Divides selected clock by 1 01 Encoding 1 — Divides selected clock by 2 (reset default) 10 Encoding 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 crystal oscillator or external clock source. 1 High frequency range selected for the crystal oscillator of 1 MHz to 16 MHz (1 MHz to 40 MHz for external clock source) 0 Low frequency range selected for the crystal oscillator of 32 kHz to 100 kHz (32 kHz to 1 MHz for external clock source)
4 HGO	**High Gain Oscillator Select** — Controls the crystal oscillator mode of operation. 1 Configure crystal oscillator for high gain operation 0 Configure crystal 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

15.3.3 MCG Trim Register (MCGTRM)

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

Figure 15-10. MCG Trim Register (MCGTRM)

[1] A value for TRIM is loaded during reset from a factory programmed location when not in any BDM mode. If in a BDM mode, a default value of 0x80 is loaded.

Table 15-10. 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.

15.3.4   MCG Status and Control Register (MCGSC)

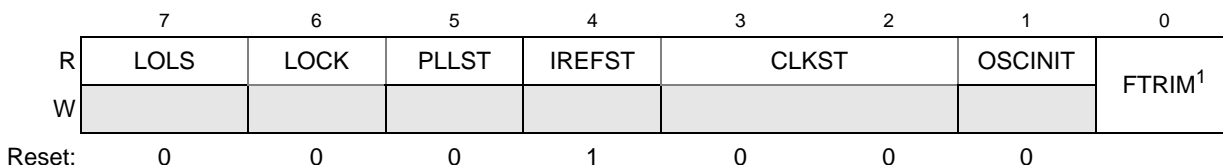|	7	6	5	4	3	2	1	0
R	LOLS	LOCK	PLLST	IREFST	CLKST		OSCINIT	FTRIM[1]
W								
Reset:	0	0	0	1	0	0	0	

Figure 15-11. MCG Status and Control Register (MCGSC)

[1] A value for FTRIM is loaded during reset from a factory programmed location when not in any BDM mode. If in a BDM mode, a default value of 0x0 is loaded.

Table 15-11. MCG Status and Control Register Field Description

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, changing the value of DMX32, DRS[1:0] and IREFS bits in FBE, FBI, FEE and FEI modes; DIV32 bit in FBE and FEE modes; TRIM[7:0] bits in FBI and FEI modes; RDIV[2:0] bits in FBE, FEE, PBE and PEE modes; VDIV[3:0] bits in PBE and PEE modes; and PLLS bit, causes the lock status bit to clear and stay clear until the FLL or PLL has reacquired lock. Entry into BLPI, BLPE or stop mode also causes the lock status bit to clear and stay cleared until the exit of 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 15-11. MCG Status and Control Register Field Description (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 crystal 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.

15.3.5   MCG Control Register 3 (MCGC3)

	7	6	5	4	3	2	1	0
R	LOLIE	PLLS	CME	DIV32	\	VDIV		
W								
Reset:	0	0	0	0	0	0	0	1
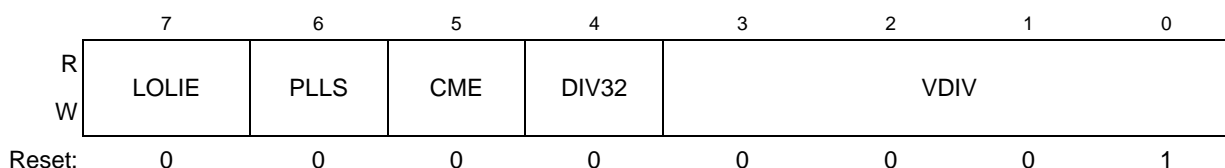
Figure 15-12. MCG PLL Register (MCGPLL)

Table 15-12. 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
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. If the external reference clock is set to be disabled when the MCG enters STOP mode (EREFSTEN=0), then the CME bit should be set to a logic 0 before the MCG enters STOP mode. Otherwise a reset request may occur while in STOP mode. 0 Clock monitor is disabled. 1 Generate a reset request on loss of external clock.

Table 15-12. MCG PLL Register Field Descriptions (Continued)

Field	Description
4 DIV32	**Divide-by-32 Enable** — Controls an additional divide-by-32 factor to the external reference clock for the FLL when RANGE bit is set. When the RANGE bit is 0, this bit has no effect. Writes to this bit are ignored if PLLS bit is set. 0 Divide-by-32 is disabled. 1 Divide-by-32 is enabled when RANGE=1.
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 — Multiply by 44. 1100 Encoding 12 — Multiply by 48. 1101 Encoding 13 — Reserved (default to M=48). 111x Encoding 14-15 — Reserved (default to M=48).

15.3.6 MCG Control Register 4 (MCGC4)
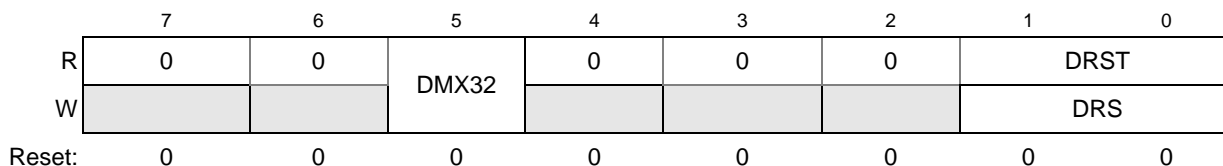
	7	6	5	4	3	2	1	0
R	0	0	DMX32	0	0	0	DRST	
W							DRS	
Reset:	0	0	0	0	0	0	0	0

Figure 15-13. MCG Control Register 4 (MCGC4)

Table 15-13. MCG Test and Control Register Field Descriptions

Field	Description
7:6	Reserved for test, user code should not write 1's to these bits.
5 DMX32	**DCO Maximum frequency with 32.768 kHz reference** — The DMX32 bit controls whether or not the DCO frequency range is narrowed to its maximum frequency with a 32.768 kHz reference. See Table 15-14. 0 DCO has default range of 25%. 1 DCO is fined tuned for maximum frequency with 32.768 kHz reference.
4:2	Reserved for test, user code should not write 1's to these bits.
1:0 DRST DRS	**DCO Range Status** — The DRST read bits indicate the current frequency range for the FLL output, DCOOUT. See Table 15-14. The DRST bits do not update immediately after a write to the DRS field due to internal synchronization between clock domains. The DRST bits are not valid in BLPI, BLPE, PBE or PEE mode and it reads zero regardless of the DCO range selected by the DRS bits. **DCO Range Select** — The DRS bits select the frequency range for the FLL output, DCOOUT. Writes to the DRS bits while either the LP or PLLS bit is set are ignored. 00 Low range. 01 Mid range. 10 High range. 11 Reserved

Table 15-14. DCO frequency range[1]

DRS	DMX32	Reference range	FLL factor	DCO range
00	0	31.25 - 39.0625 kHz	512	16 - 20 MHz
	1	32.768 kHz	608	19.92 MHz
01	0	31.25 - 39.0625 kHz	1024	32 - 40 MHz
	1	32.768 kHz	1216	39.85 MHz
10	0	31.25 - 39.0625 kHz	1536	48-60 MHz
	1	32.768 kHz	1824	59.77 MHz
11	Reserved			

[1] The resulting bus clock frequency should not exceed the maximum specified bus clock frequency of the device.

15.3.7   MCG Test Register (MCGT)

Table 15-15. MCG Test Register Field Descriptions

Field	Description
7:6	Reserved for test, user code should not write 1's to these bits.
5	Reserved, user code should not write 1's to these bits
4:1	Reserved for test, user code should not write 1's to these bits.
0	Reserved, user code should not write 1's to these bits

15.4 Functional Description

15.4.1 MCG Modes of Operation

The MCG operates in one of the modes described in Table 15-16.

NOTE

The MCG restricts transitions between modes. For the permitted transitions, see Section 15.4.2, "MCG Mode State Diagram."

Table 15-16. MCG Modes of Operation

Mode	Related field values	Description
FLL Engaged Internal (FEI)	• MCGC1[IREFS] = 1 • MCGC1[CLKS] = 00 • MCGC3[PLLS] = 0	Default. MCGOUT is derived from the FLL clock, which is controlled by the internal reference clock. The FLL clock frequency locks to a multiplication factor, as selected by the DRS[1:0] and DMX32 bits, times the internal reference frequency. MCGLCLK is derived from the FLL, and the PLL is disabled in a low-power state.
FLL Engaged External (FEE)	• MCGC1[IREFS] = 0 • MCGC1[CLKS] = 00 • MCGC1[RDIV] is programmed to divide the reference clock to be within the range of 31.2500 to 39.0625 kHz. • MCGC3[PLLS] = 0	MCGOUT is derived from the FLL clock, which is controlled by the external reference clock. The external reference clock that is enabled can be produced by an external crystal, ceramic resonator, or another external clock source connected to the required crystal oscillator (XOSC). The FLL clock frequency locks to a multiplication factor, as selected by the DRS[1:0] and DMX32 bits, times the external reference frequency, as specified by MCGC1[RDIV], MCGC2[RANGE], and MCGC3[DIV32]. MCGLCLK is derived from the FLL, and the PLL is disabled in a low-power state.
FLL Bypassed Internal (FBI)	• MCGC1[IREFS] = 1 • MCGC1[CLKS] = 01 • MCGC2[LP] = 0 (or the BDM is enabled) • MCGC3[PLLS] = 0	MCGOUT is derived from the internal reference clock; 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. MCGOUT is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL clock frequency locks to a multiplication factor, as selected by the DRS[1:0] and DMX32 bits, times the internal reference frequency. MCGLCLK is derived from the FLL, and the PLL is disabled in a low-power state.
FLL Bypassed External (FBE)	• MCGC1[IREFS] = 0 • MCGC1[CLKS] = 10 • MCGC1[RDIV] is programmed to divide the reference clock to be within the range of 31.2500 to 39.0625 kHz • MCGC2[LP] = 0 (or the BDM is enabled) • MCGC3[PLLS] = 0	MCGOUT is derived from the external reference clock; 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 MCGOUT is driven from the external reference clock. MCGOUT is derived from the external reference clock. The external reference clock that is enabled can be produced by an external crystal, ceramic resonator, or another external clock source connected to the required crystal oscillator (XOSC).The FLL clock is controlled by the external reference clock, and the FLL clock frequency locks to a multiplication factor, as selected by the DRS[1:0] and DMX32 bits, times the external reference frequency, as selected by MCGC1[RDIV], MCGC2[RANGE], and MCGC3[DIV32]. MCGLCLK is derived from the FLL, and the PLL is disabled in a low-power state.

Table 15-16. MCG Modes of Operation (Continued)

Mode	Related field values	Description
PLL Engaged External (PEE)	• MCGC1[IREFS] = 0 • MCGC1[CLKS] = 00 • MCGC1[RDIV] is programmed to divide the reference clock to be within the range of 1 to 2 MHz. • PLLS = 1	MCGOUT is derived from the PLL clock, which is controlled by the external reference clock. The external reference clock that is enabled can be produced by an external crystal, ceramic resonator, or another external clock source connected to the required crystal oscillator (XOSC). The PLL clock frequency locks to a multiplication factor, as specified by MCGC3[VDIV], times the external reference frequency, as specified by MCGC1[RDIV], MCGC2[RANGE], and MCGC3[DIV32]. If the BDM is enabled, MCGLCLK is derived from the DCO (open-loop mode) divided by two. If the BDM is not enabled, the FLL is disabled in a low-power state. In this mode, MCGT[DRST] is read as a 0 regardless of the value of MCGT[DRS].
PLL Bypassed External (PBE)	• MCGC1[IREFS] = 0 • MCGC1[CLKS] = 10 • MCGC1[RDIV] is programmed to divide the reference clock to be within the range of 1 to 2 MHz. • MCGC2[LP] = 0 • MCGC3[PLLS] = 1	MCGOUT is derived from the external reference clock; 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 MCGOUT is driven from the external reference clock. MCGOUT is derived from the external reference clock. The external reference clock that is enabled can be produced by an external crystal, ceramic resonator, or another external clock source connected to the required crystal oscillator (XOSC). The PLL clock frequency locks to a multiplication factor, as specified by MCGC3[VDIV], times the external reference frequency, as specified by MCGC1[RDIV], MCGC2[RANGE], and MCGC3[DIV32]. If the BDM is enabled, MCGLCLK is derived from the DCO (open-loop mode) divided by two. If the BDM is not enabled, the FLL is disabled in a low-power state. In this mode, MCGT[DRST] is read as a 0 regardless of the value of MCGT[DRS].
Bypassed Low Power Internal (BLPI)	• MCGC1[IREFS] = 1 • MCGC1[CLKS] = 01MCGC3[PLLS] = 0 • MCGC2[LP] = 1 (and the BDM is disabled)	MCGOUT is derived from the internal reference clock. The PLL and FLL are disabled, and MCGLCLK is not available for BDC communications. If the BDM becomes enabled, the mode switches to FLL bypassed internal (FBI) mode. In this mode, MCGT[DRST] is read as a 0 regardless of the value of MCGT[DRS].
Bypassed Low Power External (BLPE)	• MCGC1[IREFS] = 0 • MCGC1[CLKS] = 10 • MCGC2[LP] = 1 (and the BDM is disabled)	MCGOUT is derived from the external reference clock. The external reference clock that is enabled can be produced by an external crystal, ceramic resonator, or another external clock source connected to the required crystal oscillator (XOSC). The PLL and FLL are disabled, and MCGLCLK is not available for BDC communications. If the BDM becomes enabled, the mode switches to one of the bypassed external modes as determined by the state of MCGC3[PLLS]. In this mode, MCGT[DRST] is read as a 0 regardless of the value of MCGT[DRS].

Table 15-16. MCG Modes of Operation (Continued)

Mode	Related field values	Description
Stop	—	Entered whenever the MCU enters a Stop state. The FLL and PLL are disabled, and all MCG clock signals are static except in the following cases: MCGIRCLK is active in Stop mode when all the following conditions become true: • MCGC1[IRCLKEN] = 1 • MCGC1[IREFSTEN] = 1 MCGERCLK is active in Stop mode when all the following conditions become true: • MCGC2[ERCLKEN] = 1 • MCGC2[EREFSTEN] = 1

15.4.2 MCG Mode State Diagram

Figure 15-15 shows the MCG's mode state diagram. The arrows indicate the permitted mode transitions.
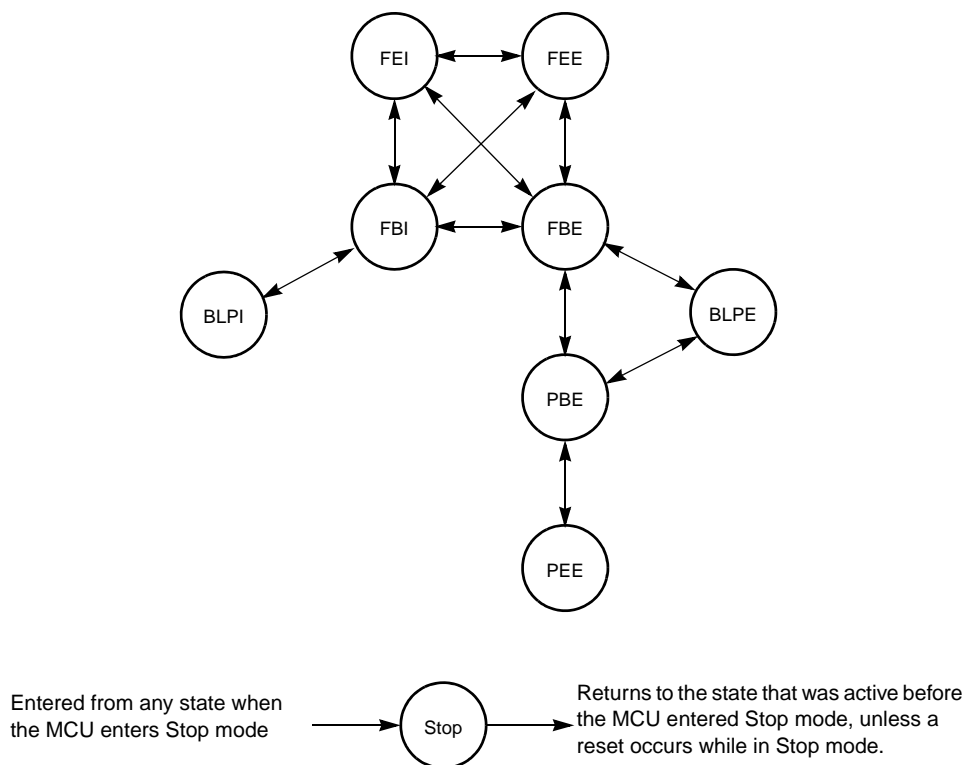

Figure 15-15. MCG Mode State Diagram

15.4.3 Mode Switching

The IREFS bit can be changed at anytime, but the actual switch to the newly selected clock is shown by the IREFST bit. When switching between engaged internal and engaged external modes, the FLL or PLL will begin locking again after the switch is completed.

The CLKS bits can also be changed at anytime, but the actual switch to the newly selected clock is shown by the CLKST bits. If the newly selected clock is not available, the previous clock will remain selected.

The DRS bits can be changed at anytime except when LP bit is 1. If the DRS bits are changed while in FLL engaged internal (FEI) or FLL engaged external (FEE), the bus clock remains at the previous DCO range until the new DCO starts. When the new DCO starts the bus clock switches to it. After switching to the new DCO the FLL remains unlocked for several reference cycles. Once the selected DCO startup time is over, the FLL is locked. The completion of the switch is shown by the DRST bits.

For details see Figure 15-15.

15.4.4   Bus Frequency Divider

The BDIV bits can be changed at anytime and the actual switch to the new frequency will occur immediately.

15.4.5   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. The DRS bit can not be written while LP bit is 1.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.

15.4.6   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.

15.4.7   External Reference Clock

The MCG module can support an external reference clock with frequencies between 31.25 kHz to 40 MHz in all modes. 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.

15.4.8 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.

15.5   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.

15.5.1   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 $t_{fll\_acquire}$ milliseconds.

NOTE

If the internal reference is not already trimmed, 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.

15.5.1.1   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 15-15). 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. If the RANGE bit (bit 5) in MCGC2 is set, set DIV32 in MCGC3 to allow access to the proper RDIV values.
3. Write to MCGC1 to select the clock mode.
    — If entering FEE mode, 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.
4. Once 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.

5. Write to the MCGC4 register to determine the DCO output (MCGOUT) frequency range. Make sure that the resulting bus clock frequency does not exceed the maximum specified bus clock frequency of the device.

— By default, with DMX32 cleared to 0, the FLL multiplier for the DCO output is 512. For greater flexibility, if a mid-range FLL multiplier of 1024 is desired instead, set the DRS[1:0] bits to %01 for a DCO output frequency of 33.55 MHz. If a high-range FLL multiplier of 1536 is desired instead, set the DRS[1:0] bits to %10 for a DCO output frequency of 50.33 MHz.

— When using a 32.768 kHz external reference, if the maximum low-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set the DRS[1:0] bits to %00 and set the DMX32 bit to 1. The resulting DCO output (MCGOUT) frequency with the new multiplier of 608 will be 19.92 MHz.

— When using a 32.768 kHz external reference, if the maximum mid-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set the DRS[1:0] bits to %01 and set the DMX32 bit to 1. The resulting DCO output (MCGOUT) frequency with the new multiplier of 1216 will be 39.85 MHz.

— When using a 32.768 kHz external reference, if the maximum high-range DCO frequency that can be achieved with a 32.768 kHz reference is desired, set the DRS[1:0] bits to %10 and set the DMX32 bit to 1. The resulting DCO output (MCGOUT) frequency with the new multiplier of 1824 will be 59.77 MHz.

6. Wait for the LOCK bit in MCGSC to become set, indicating that the FLL has locked to the new multiplier value designated by the DRS and DMX32 bits.

NOTE

Setting DIV32 (bit 4) in MCGC3 is strongly recommended for FLL external modes when using a high frequency range (RANGE = 1) external reference clock. The DIV32 bit is ignored in all other modes.

To change from FEI clock mode to FBI clock mode, follow this procedure:

1. Change the CLKS bits in MCGC1 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.

15.5.2   Using a 32.768 kHz Reference

In FEE and FBE modes, if using a 32.768 kHz external reference, at the default FLL multiplication factor of 512, the DCO output (MCGOUT) frequency is 16.78 MHz at high-range. If the DRS[1:0] bits are set to %01, the multiplication factor is doubled to 1024, and the resulting DCO output frequency is 33.55 Mhz at mid-range. If the DRS[1:0] bits are set to %10, the multiplication factor is set to 1536, and the resulting DCO output frequency is 50.33 Mhz at high-range. Make sure that the resulting bus clock frequency does not exceed the maximum specified bus clock frequency of the device.

Setting the DMX32 bit in MCGC4 to 1 increases the FLL multiplication factor to allow the 32.768 kHz reference to achieve its maximum DCO output frequency. When the DRS[1:0] bits are set to %00, the 32.768 kHz reference can achieve a high-range maximum DCO output of 19.92 MHz with a multiplier of 608. When the DRS[1:0] bits are set to %01, the 32.768 kHz reference can achieve a mid-range maximum DCO output of 39.85 MHz with a multiplier of 1216. When the DRS[1:0] bits are set to %10, the 32.768 kHz reference can achieve a high-range maximum DCO output of 59.77 MHz with a multiplier of 1824. Make sure that the resulting bus clock frequency does not exceed the maximum specified bus clock frequency of the device.

In FBI and FEI modes, setting the DMX32 bit is not recommended. If the internal reference is trimmed to a frequency above 32.768 kHz, the greater FLL multiplication factor could potentially push the microcontroller system clock out of specification and damage the part.

15.5.3   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.

If switching to FBE or FEE mode, first setting the DIV32 bit will ensure a proper reference frequency is sent to the FLL clock at all times.

In FBE, FEE, FBI, and FEI modes, at any time, the application can switch the FLL multiplication factor between 512, 1024, and 1536 with the DRS[1:0] bits in MCGC4. Writes to the DRS[1:0] bits will be ignored if LP=1 or PLLS=1.

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 15-17. MCGOUT Frequency Calculation Options

Clock Mode	f_{MCGOUT}[1]	Note
FEI (FLL engaged internal)	$(f_{int} * F) / B$	Typical f_{MCGOUT} = 16 MHz immediately after reset.
FEE (FLL engaged external)	$(f_{ext} / R * F) / B$	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, F is the FLL factor selected by the DRS[1:0] and DMX32 bits, and M is the multiplier selected by the VDIV bits.

This section will include 3 mode switching examples using an 8 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).

15.5.3.1   Example 1: Moving from FEI to PEE Mode: External Crystal = 8 MHz, Bus Frequency = 16 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 16 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 the crystal 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) Because RANGE = 1, set DIV32 (bit 4) in MCGC3 to allow access to the proper RDIV bits while in an FLL external mode.

d) MCGC1 = 0x98 (%10011000)

   – 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 %011, or divide-by-256 because 8MHz / 256 = 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

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 is %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: MCGC3 = 0x58 (%01011000)

   – PLLS (bit 6) set to 1, selects the PLL. At this time, with an RDIV value of %011, the FLL reference divider of 256 is switched to the PLL reference divider of 8 (see Table 15-8), resulting in a reference frequency of 8 MHz/ 8 = 1 MHz. In BLPE mode,changing the PLLS bit only prepares the MCG for PLL usage in PBE mode

   – DIV32 (bit 4) still set at 1. Because the MCG is in a PLL mode, the DIV32 bit is ignored. Keeping it set at 1 makes transitions back into an FLL external mode easier.

   – VDIV (bits 3-0) set to %1000, or multiply-by-32 because 1 MHz reference * 32= 32MHz. 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

c) BLPE: If transitioning through BLPE mode, clear LP (bit 3) in MCGC2 to 0 here to switch to PBE mode

d) PBE: Loop until PLLST (bit 5) in MCGSC is set, indicating that the current source for the PLLS clock is the PLL

e) PBE: Then loop until LOCK (bit 6) in MCGSC is set, indicating that the PLL has acquired lock

3. Lastly, 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

b) 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

– Now, With an RDIV of divide-by-8, a BDIV of divide-by-1, and a VDIV of multiply-by-32, MCGOUT = [(8 MHz / 8) * 32] / 1 = 32 MHz, and the bus frequency is MCGOUT / 2, or 16 MHz
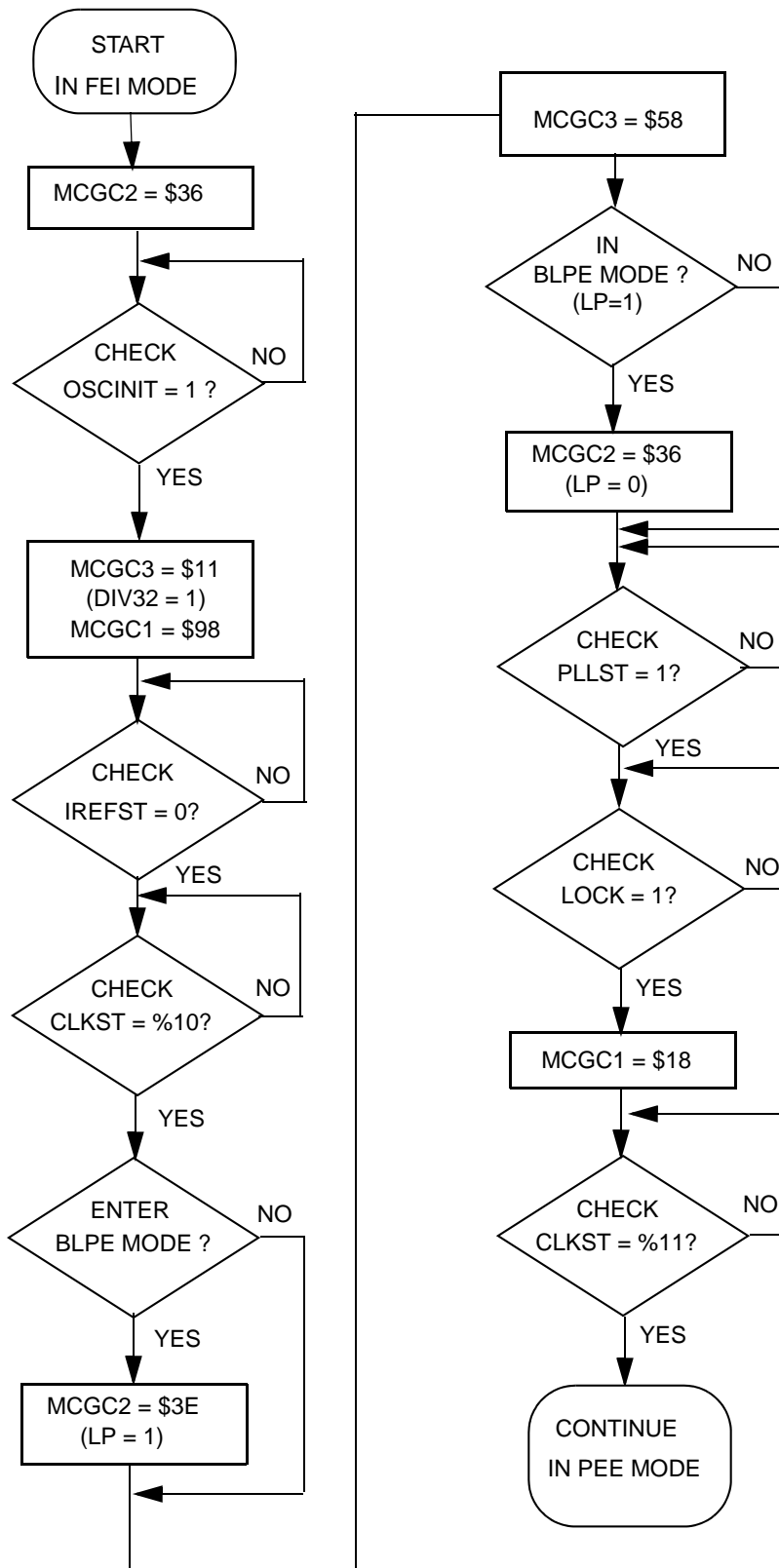
Figure 15-16. Flowchart of FEI to PEE Mode Transition using an 8 MHz crystal

15.5.3.2 Example 2: Moving from PEE to BLPI Mode: Bus Frequency =16 kHz

In this example, the MCG will move through the proper operational modes from PEE mode with an 8MHz crystal configured for an 16 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 = 0x98 (%10011000)
        – 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: MCGC3 = 0x18(%00011000)
        – PLLS (bit 6) clear to 0 to select the FLL. At this time, with an RDIV value of %011, the PLL reference divider of 8 is switched to an FLL divider of 256 (see Table 15-7), resulting in a reference frequency of 8 MHz / 256 = 31.25 kHz. If RDIV was not previously set to %011 (necessary to achieve required 31.25-39.06 kHz FLL reference frequency with an 8 MHz external source frequency), it must be changed prior to clearing the PLLS bit. 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.
        – DIV32 (bit 4) set to 1 (if previously cleared), automatically switches RDIV bits to the proper reference divider for the FLL clock (divide-by-256)
    c) BLPE: If transitioning through BLPE mode, clear LP (bit 3) in MCGC2 to 0 here to switch to FBE mode
    d) FBE: Loop until PLLST (bit 5) in MCGSC is clear, indicating that the current source for the PLLS clock is the FLL
    e) 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 = 0x5C (%01011100)
        – 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) remain unchanged because the reference divider does not affect the internal reference.

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 BLPI mode.

a) MCGC2 = 0x08 (%00001000)

- LP (bit 3) in MCGSC is 1
- RANGE, HGO, EREFS, ERCLKEN, and EREFSTEN bits are ignored when the IREFS bit (bit2) in MCGC is set. They can remain set, or be cleared at this point.
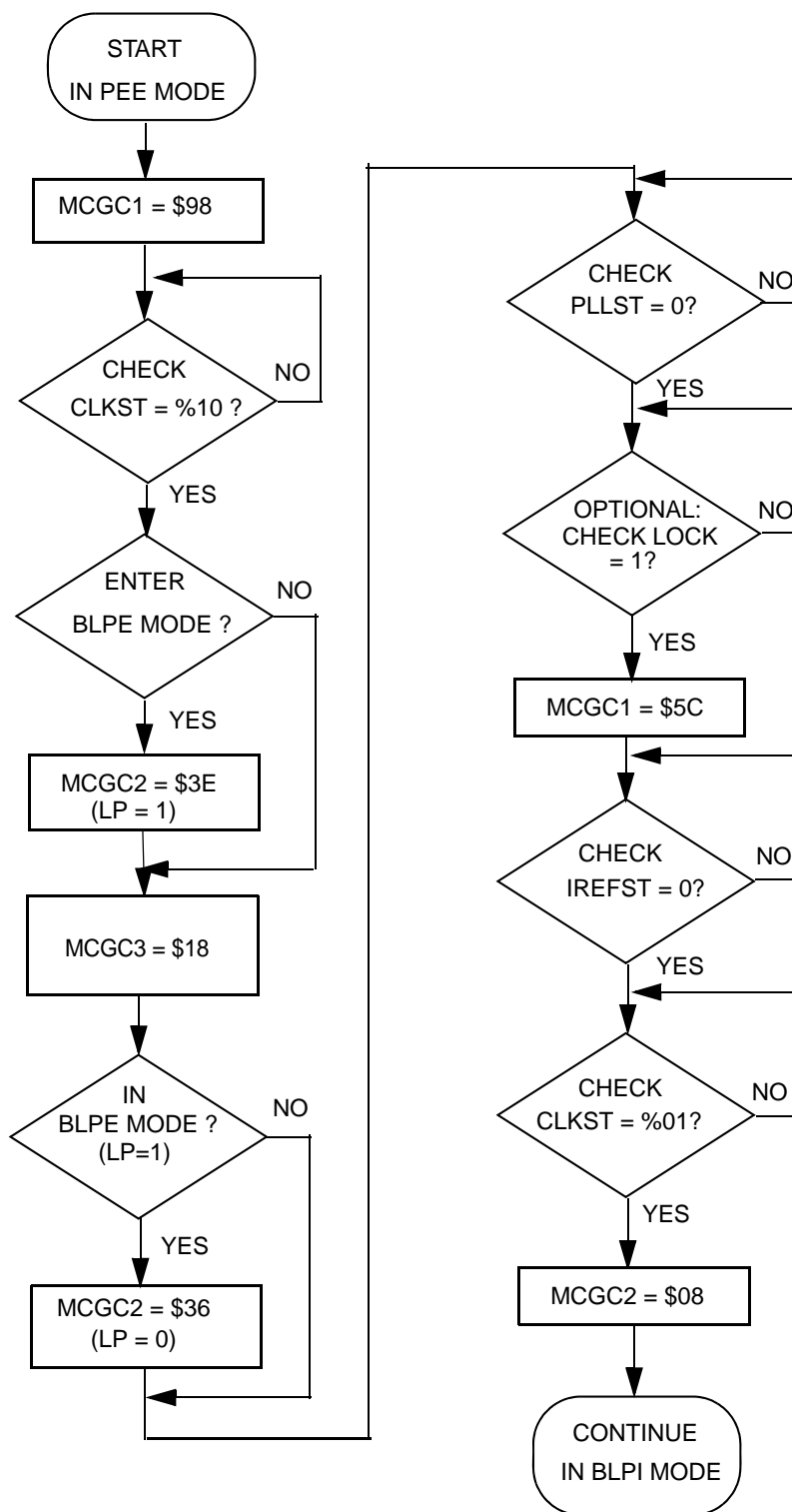
Figure 15-17. Flowchart of PEE to BLPI Mode Transition using an 8 MHz crystal

15.5.3.3 Example 3: Moving from BLPI to FEE Mode: External Crystal = 8 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 an 8MHz 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 8 MHz is within the high frequency range

      – HGO (bit 4) set to 1 to configure the crystal 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 = 0x18 (%00011000)

      – 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) remain at %011, or divide-by-256 for a reference of 8 MHz / 256 = 31.25 kHz.

      – 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

   g) Now, with a 31.25 kHz reference frequency, a fixed DCO multiplier of 512, and a bus divider of 1, MCGOUT = 31.25 kHz * 512 / 1 = 16 MHz. Therefore, the bus frequency is 8 MHz.

   h) At this point, by default, the DRS[1:0] bits in MCGC4 are set to %00 and DMX32 in MCGC4 is cleared to 0. If a bus frequency of 16MHz is desired instead, set the DRS[1:0] bits to $01 to switch the FLL multiplication factor from 512 to 1024 and loop until LOCK (bit 6) in MCGSC is set, indicating that the FLL has reacquired LOCK. To return the bus frequency to 8 MHz, set the DRS[1:0] bits to %00 again, and the FLL multiplication factor will switch back to 512. Then loop again until the LOCK bit is set.
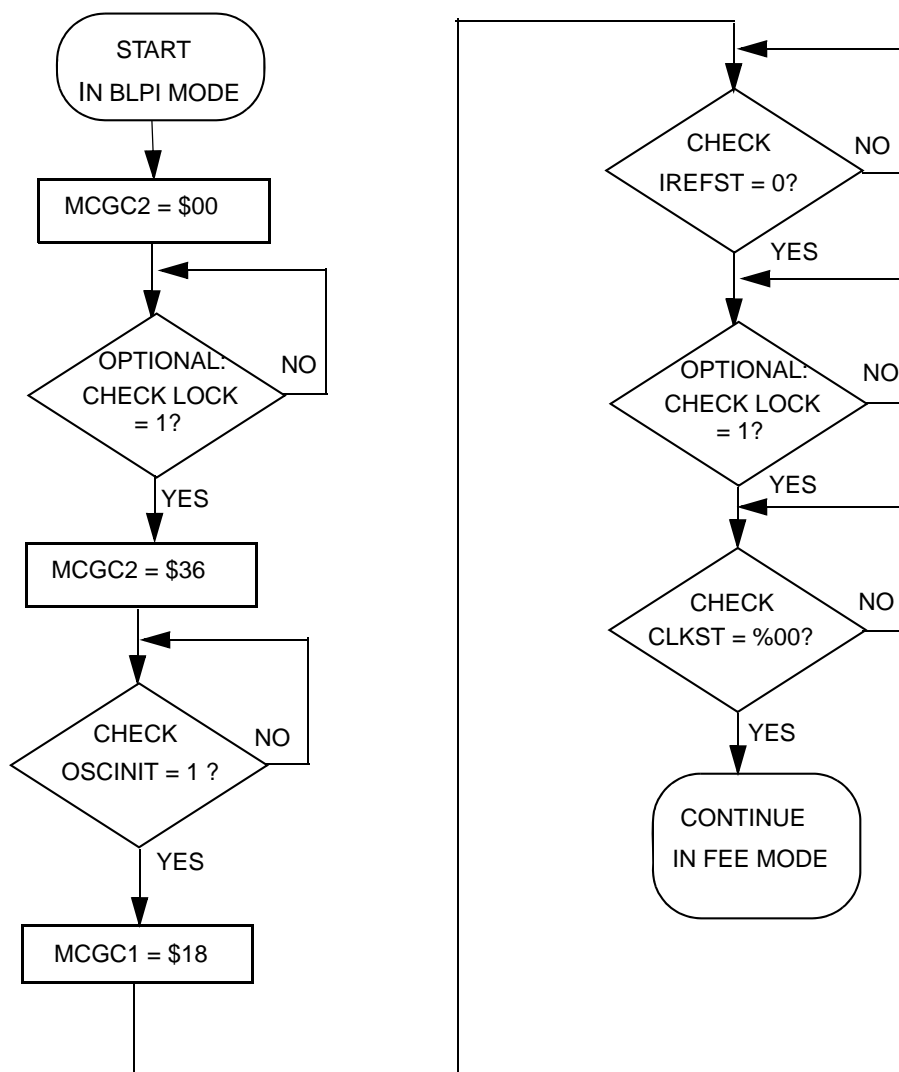
Figure 15-18. Flowchart of BLPI to FEE Mode Transition using an 8 MHz crystal

15.5.4 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 reset is the factory trim value unless the device resets into any BDM mode in which case it is 0x800. 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: Internal Reference Clock Trim later in this section, this approach will be demonstrated.

If a user specified trim value has been found for a device (to replace the factory trim value), 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 to the user specified value 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.

15.5.4.1   Example 4: Internal Reference Clock Trim

For applications that require a user specified 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.

Initial conditions:
1) Clock supplied from ATE has 500 μsec duty period
2) MCG configured for internal reference with 8MHz bus
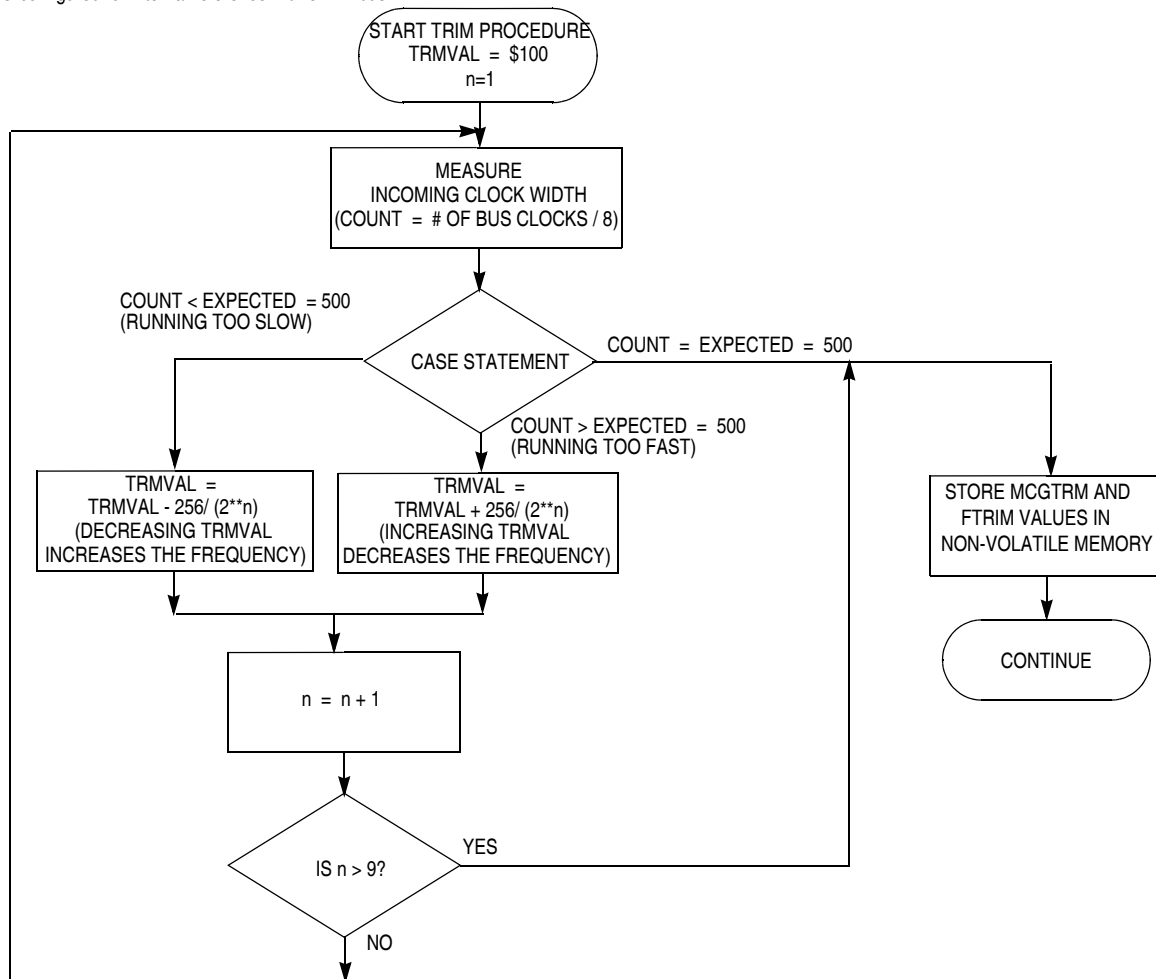
Figure 15-19. Trim Procedure

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 15-19 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 16
Programmable Delay Block (S08PDBV1)

16.1 Introduction

This simple timer is used to control the hardware triggering of the ADC and the DAC so that precise timing between DAC updates and ADC conversions can be achieved.

16.1.1 Overview

Many applications need to synchronize the time at which multiple ADC samples are taken with respect to an external trigger or event. The Programmable Delay Block provides controllable delays from either an external trigger or a programmable interval tick to the sample trigger input of one or more ADCs. The PDB also can generate a hardware trigger to the DAC. This signal can be used to advance the DAC Buffer pointer in order to change sensor biasing while ADC conversions are being triggered.

16.1.2 PDB Trigger inputs

The PDB on these devices has three input trigger sources that initiate the PDB operation. The three sources are:

- ACMP output (ACMPO)
- When ADC Conversion Complete Flag (ADCSC1_COCO) is set.
- Software Trigger

To select the input trigger source, set the TRIGSEL bits in the PDBC1 register as described in the following table.

Table 16-1. Selecting the PDB Input Trigger Source

Input Trigger Source	PDBC1 Register Value for TRIGSEL bits	Connected to
ACMPO	0b001	TRIGGERIN1
ADCSC1_COCO	0b000	TRIGGERIN0
Software Trigger	0b111	—

16.2   ADC Hardware Triggers and Selects

When the ADC conversion is completed and the PDB is used as an ADC hardware trigger source, the ADCSC1_COCO bit can be used as the input trigger source for the PDB (if configured that way). In other words, the PDB and ADC can trigger each other cyclically and indefinitely until otherwise aborted.
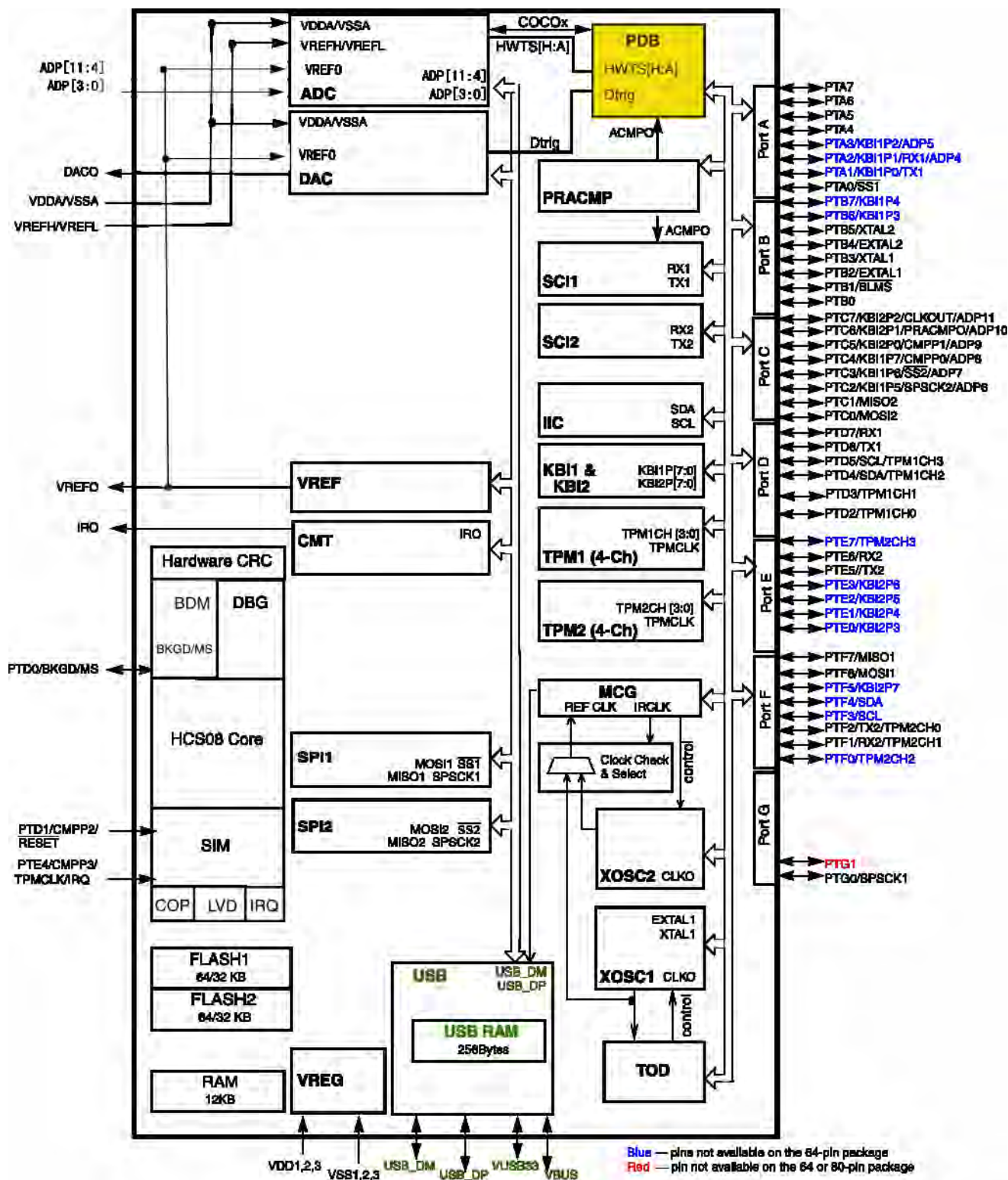
Figure 16-1. Block Diagram Highlighting the PDB

16.2.1   Features

- Eight channels
  - Each channel supplies a single trigger output event
  - Each trigger output is independently enabled and individually controlled
  - Each channel can be triggered from a programmed delay or from previous channel acknowledgment
- All channel trigger outputs can be ORed together to schedule multiple conversions from one input trigger event
- Prescaler options to support divided-by-1 up to divided-by-2560
- DAC Trigger Interval Register (16-bit) that is used to place the interval count between DAC hardware trigger output signals
- Multiple sources for PDB input triggering
  - Single software trigger input
  - Up to 3 trigger inputs from either on-chip or off-chip sources
- Positive transition of trigger_in will initiate the PDB primary counter
- Continuous trigger or single shot mode supported
- Bypass mode supported
- One programmable interrupt

16.2.2   Modes of Operation

Modes of operation include:

Disabled   Counter is off and all trigger outputs are low.

Enabled OneShot   Counter is enabled and restarted at count zero upon receiving a positive edge on the trigger input. Each Trigger output asserts once per input trigger.

Enabled Continuous Counter is enabled and restarted at count zero. The counter will be rolled over to zero again when the count reaches the value specified in the MOD internal buffer, and counting restarts. This enables a continuous stream of triggers out as a result of a single trigger input.**Enabled Bypassed**The input trigger bypasses the PDB logic entirely. It is possible to bypass all or only one of the trigger outputs; Therefore this mode can be used in conjunction with any of the above.

<div align="center">NOTE</div>

- For all of the Enabled modes, if the DAC hardware trigger output is enabled each time the count reaches the DAC interval count value, a trigger signal is sent to the DAC.

16.2.3   Block Diagram
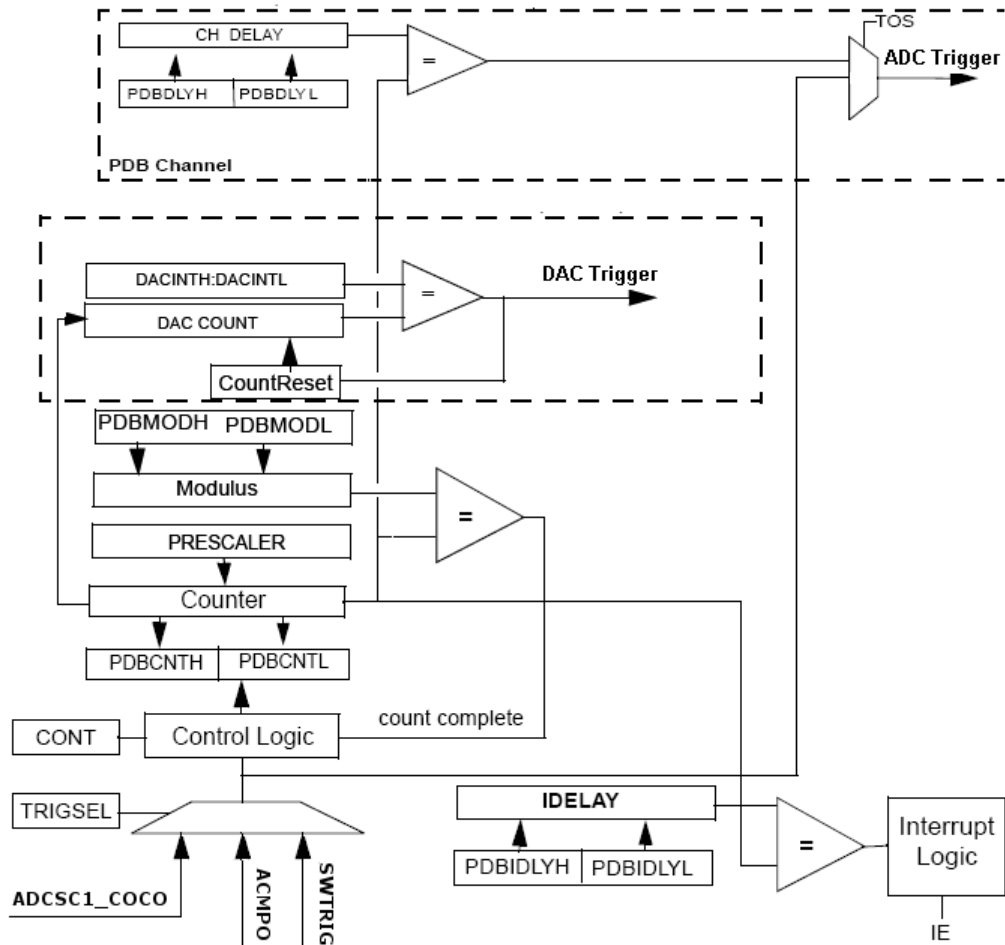
Figure 16-2. PDB Block Diagram

Figure 16-2. illustrates the basic structure of the PDB block. It contains a primary counter whose output is compared against several different digital values.

The time between assertion of the trigger input (software or hardware trigger) to the point at which the ADC output trigger is asserted depends upon the following factors:

- Trigger input to Pre-Trigger = (PRESCALER x PDBDLYH:L) + 1 peripheral bus clock cycle
- Add one additional bus clock cycle to determine the time at which the trigger output changes.

Figure 16-3. is an example of OneShot mode. The trigger delay for the ADC is set by the value placed in the PDBDLYH:L register. The PDB trigger input signal is from a separate module or software trigger.
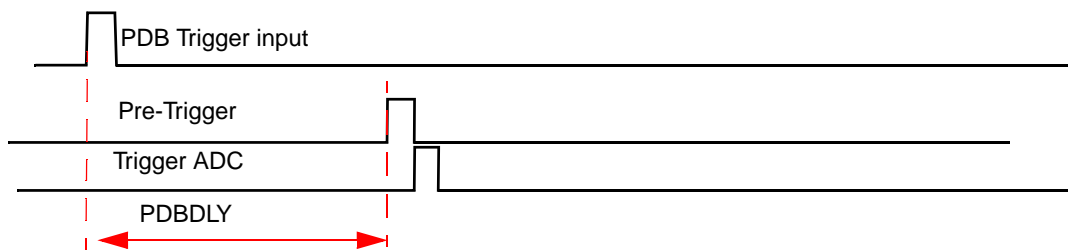
Figure 16-3. Delay between PDB trigger input to output ADC trigger

The third digital value, modulus, resets the counter to zero at the end of the count. If the CONT bit in the PDBC1 is set, the counter resumes a new count. Otherwise, the timer operation will cease until the next trigger input event occurs.

The DAC Interval Register (DACINTH:L) determines the time between DAC hardware trigger pulses. If the DAC trigger output is enabled (DACTOE is set) each time the PDB counter register increments the number of counts placed in the DACINTH:L registers, the PDB will output a DAC trigger pulse.

Together the DAC trigger pulse and the ADC trigger pulses allow precise timing of DAC updates and ADC measurements.

The following 16-bit registers impact PDB operation:

- Channel Delay register (PDBDLYH:L)
- Modulus register (PDBMODH:L)
- Interrupt Delay register (PDBIDLYH:L)
- DAC Trigger Interval register (DACINTH:L)

Like all 16-bit registers, these internal registers are buffered and any values written to them are written first to their buffer registers. The circumstances that cause these registers to be updated with the values from their buffer registers are summarized in the following table according to the current operation mode.

Table 16-2. Updating the registers that impact PDB operation

Operation Mode	PDBSC Register		Value Updates
	LDOK bit	LDMOD bit	
Any mode	Write 1	0	Immediately after write 1 to LDOK.

Table 16-2. Updating the registers that impact PDB operation (Continued)

Operation Mode	PDBSC Register		Value Updates
	LDOK bit	LDMOD bit	
Continuous	Write 1	1	Immediately after counter rolls over.
OneShot	Write 1	1	Immediately after trigger signal received.

Values written to any of these registers after a logic 1 is written to the LDOK bit, are ignored and the associated buffer register is not updated until the existing values in the buffer registers are loaded into the internal registers. Read the LDOK bit to determine if the values in buffer registers have been loaded into internal registers and have taken effect.

The PDB 16-bit counter operates on up count mode. The two read-only counter registers contain the high and low bytes of the value in the PDB counter. Reading either byte latches the contents of both bytes into a buffer where they remain latched until another read of either bytes is performed. Odd-numbered reads return new data from the counter. Even-numbered reads return latched data.

The following figures are examples of the LDMOD bit effects. The PDB Counter is configured in continuous mode where it repeats counting up and rolling over to 0 after matching a fixed PDB Modulus (PDBMOD) register value. The black arrows represent Pre-Triggers based on previous PDB delay registers value. The red arrows represent Pre-Triggers based on new effective PDB delay registers value.
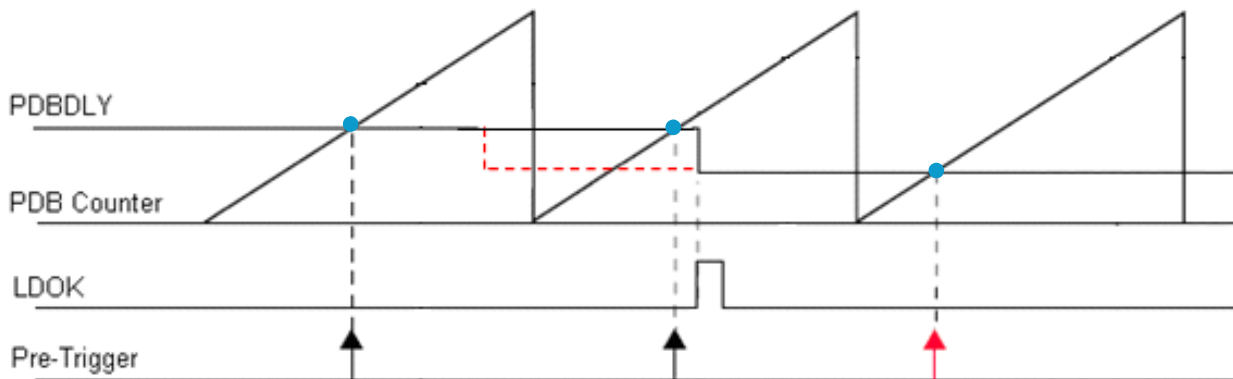
Symbol	Description
− − −	Attempts to write a new value to PDB delay (PDBDLY) registers.
↑	Pre-Triggers based on previous PDB delay registers value.
↑	Pre-Triggers based on new effective PDB delay registers value.
•	Pre-Triggers are asserted when the PDB Counter reaches the corresponding effective PDBDLY value.

Figure 16-4. Registers Update with LDMOD bit = 0 in Continuous Mode

In Figure 16-4, new values for PDBDLY registers do not become effective until after the LDOK is written to 1. The new values become effective immediately after LDOK is written to 1.
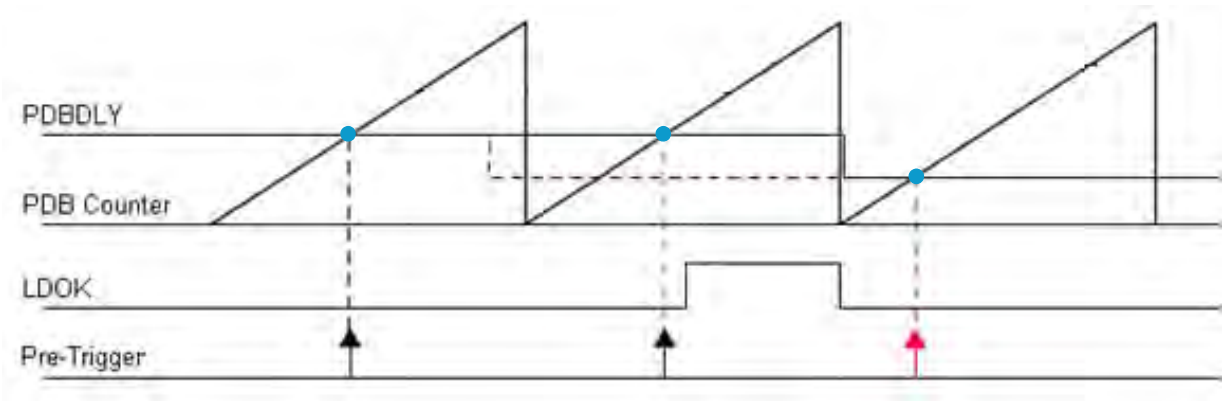
Symbol	Description
– – –	Attempts to write a new value to PDB delay (PDBDLY) registers.
↑	Pre-Triggers based on previous PDB delay registers value.
↑	Pre-Triggers based on new effective PDB delay registers value.
●	Pre-Triggers are asserted when the PDB Counter reaches the corresponding effective PDBDLY value.

Figure 16-5. Registers Update with LDMODE Bit = 1 in Continuous Mode

In Figure 16-5, new values for PDBDLY registers do not become effective until after the LDOK is written to 1 and PDB Counter rolls over to 0. The LDOK bit is held at high until PDB Counter rolls over to 0. This bit can be used to indicate when the new delay registers values become effective.

16.3   Memory Map and Registers

16.3.1   Memory Map

Offset	Register	Description
0x00	PDBSC	PDB Status and Control Register
0x01	PDBC1	PDB Control Register 1
0x02	PDBC2	PDB Control Register 2
0x03	PDBCHEN	PDB Channel Enable
0x04	PDBMODH	PDB Modulus Register High
0x05	PDBMODL	PDB Modulus Register Low
0x06	PDBCNTH	PDB Counter Register High
0x07	PDBCNTL	PDB Counter Register Low
0x08	PDBIDLYH	PDB Interrupt Delay Register High
0x09	PDBIDLYL	PDB Interrupt Delay Register Low
0x0A	DACINTH	DAC Trigger Interval Register High
0x0B	DACINTL	DAC Trigger Interval Register Low
0x0C	PDBDLYH	PDB Delay n Register High
0x0D	PDBDLYL	PDB Delay n Register Low

16.3.2   Registers Descriptions

16.3.2.1   PDB Status and Control Register (PDBSC)

This register contains status and control bits for the Programmable Delay Block.

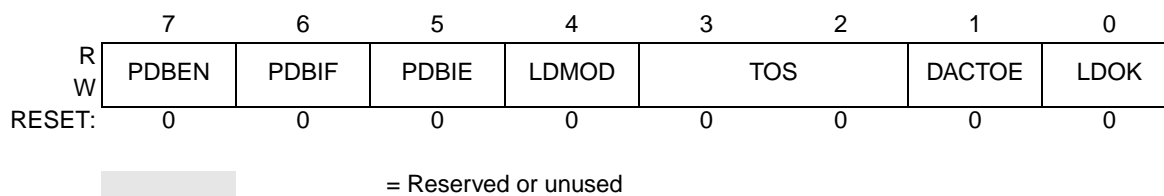|	7	6	5	4	3	2	1	0
R W	PDBEN	PDBIF	PDBIE	LDMOD	TOS		DACTOE	LDOK
RESET:	0	0	0	0	0	0	0	0

= Reserved or unused

Figure 16-6. PDB Status and Control Register (PDBSC)

Table 16-3. PDBSC Register Field Descriptions

Field	Description
7 PDBEN	**PDB module Enable** 0 Counter is off and all trigger signals are disabled. 1 Counter is enabled.
6 PDBIF	**PDB Interrupt Flag** This bit is set when a successful compare of value of counter and the IDELAY internal buffer occurs. Clear this bit by writing logic one to it.
5 PDBIE	**PDB Interrupt Enable** 0 Interrupt requests disabled 1 Interrupt requests enabled
4 LDMOD	**Load Mode Select** 0 Internal registers of PDBDLYnH:L, PDBMODH:L, PDBIDLYH:L and DACINTH:L are updated with values of their buffer registers and take effect immediately after logic 1 is written into LDOK bit 1 Internal registers of PDBDLYnH:L, PDBMODH:L, PDBIDLYH:L, and DACINTH:L are updated with values of their buffer registers and take effect when the counter rolls over in continuous mode or trigger signal is received in one shot mode after logic 1 is written into LDOK bit
3:2 TOS	**Trigger Output Select** 00 Counter delay is bypassed 01 Trigger A is function of Channel A Delay only 10 RESERVED NOTE: Do not write 10 to the TOS bits. Writing 10 to the TOS bits can have unexpected trigger output. 11 Reserved
1 DACTOE	**DAC Trigger Output Enable** 0 No DAC trigger output 1 DAC interval register defines the number of PDB counts between DAC trigger outputs.
0 LDOK	**Load OK** — Writing logic 1 to this bit loads values in buffer registers of PDBDLYnH:L, PDBMODH:L, PDBIDLYH:L and DACINTH:L into their internal registers. The internal delay registers, modulus value, interrupt delay and DAC trigger interval value will take effect immediately if LDMOD = 0, or when the counter rolls over in continuous mode or trigger signal is received in one shot mode if LDMOD = 1. Any value written to any one of the above registers, after a logic 1 being written to LDOK bit, will be ignored and buffer register will not be updated until the values in buffer registers are loaded into the internal registers. This bit is cleared when the values in buffer registers are loaded into internal registers. Writing logic "0" to this bit has no effect. Reading this bit can determine if the values in buffer registers are loaded into internal registers and take effect.

16.3.2.2 PDB Control Register 1 (PDBC1)

This register contains control bits for the Programmable Delay Block.

	7	6	5	4	3	2	1	0
R W		PRESCALER			TRIGSEL		CONT	MULT
RESET:	0	0	0	0	0	0	0	0

= Reserved or unused
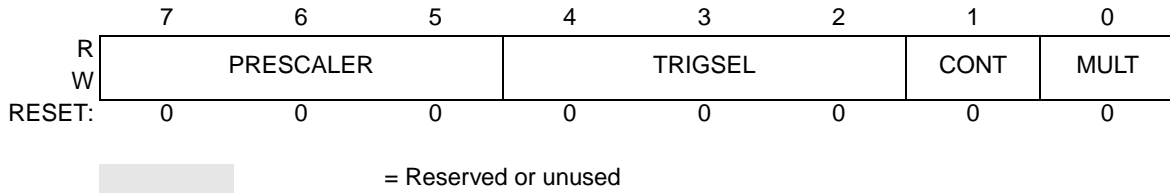
Figure 16-7. PDB Control Register 1 (PDBC1)

Table 16-4. PDBC1 Register Field Descriptions

Field	Description
7:5 PRESCALER	**Clock Prescaler Select** 000 timer uses peripheral clock 001 timer uses peripheral clock / 2 010 timer uses peripheral clock / 4 011 timer uses peripheral clock / 8 100 timer uses peripheral clock / 16 101 timer uses peripheral clock / 32 110 timer uses peripheral clock / 64 111 timer uses peripheral clock / 128
4:2 TRIGSEL	**Input Trigger Select** 000 TriggerIn0 is ADCSC1_COCO. 001 TriggerIn1 is ACMPO. 111 SWTRIG is selected.
1 CONT	**Continuous Mode Enable** 0 Module is in OneShot mode. 1 Module is in continuous mode.
0 MULT	**Multiply Prescaler bit** 0 Prescale factor is defined by the prescaler select bits. 1 Prescale factor is defined by the value selected by the prescaler bits multiplied by 20.

16.3.2.3   PDB Control Register 2 (PDBC2)

This register is used to enable each of the acknowledgment detect input for each of the channels.

	7	6	5	4	3	2	1	0
R	–	–	–	–	–	–	–	
W								SWTRIG
RESET:	0	0	0	0	0	0	0	0

= Reserved or unused
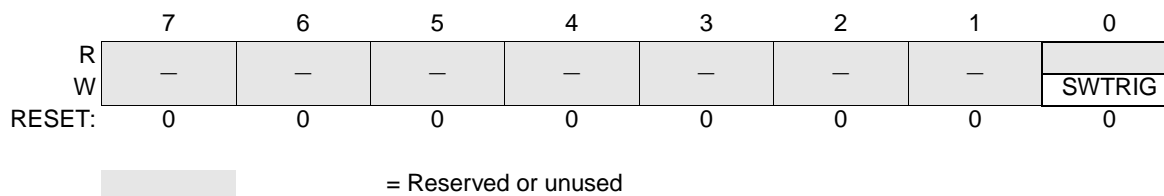
Figure 16-8. PDB Control Register (PDBC2)

Table 16-5. PDBC2 Register Field Descriptions

Field	Description
7:1 Reserved	**RESERVED —** NOTE: Do not set Bit 1 through Bit 7 to 1. Writing 1 to any of Bits 1 - 7 can cause unforeseen ADC behavior.
0 SWTRIG	**Software Trigger—** When TRIGSEL = 3'b111 and the module is enabled, writing a one to this field triggers a reset and restart of the counter.

16.3.2.4 PDB Channel Enable Register (PDBCHEN)

This register is used to enable or disable each of the trigger outputs associated with channel 0 to 7.

	7	6	5	4	3	2	1	0
R	–	–	–	–	–	–	–	CHEN0
W								
RESET:	1	1	1	1	1	1	1	1

[shaded] = Reserved or unused
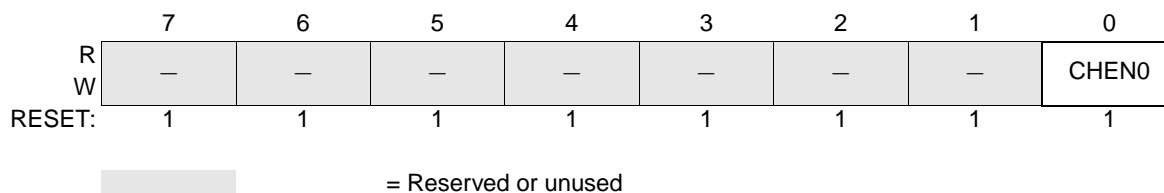
Figure 16-9. PDB Channel Enable Register (PDBCHEN)

Table 16-6. PDBCHEN Register Field Descriptions

Field	Description
7:1 Reserved	**RESERVED —** NOTE: Bit 1 through Bit 7 are reset to 1s. Make sure you write them to all zero when using the PDB. Only CHEN0 should be set to 1 when the PDB is used to trigger an ADC conversion. Do not set Bit 1 through Bit 7 to one.
0 CHEN0	**PDB channel enable bits** 0 Channel n trigger outputs are disabled and held low. 1 Channel n trigger outputs are enabled.

NOTE

Bit 1 to bit 7 are reset to 1s. Make sure you clear them when using the PDB. Only CHEN0 should be set to 1 when the PDB is used to trigger an ADC conversion. Do not set any bit from bit 1 to bit 7 to one.

16.3.2.5 PDB Modulus Registers (PDBMODH:PDBMODL)

These registers specify the period of the counter in terms of peripheral bus clock cycles. When the counter reaches this value, it will be reset back to all zeroes. If the PDBCS_CONT is set, the count will begin anew. Reads of these registers will return the value of internal registers that is taking affect for the current period of the PDB.

	7	6	5	4	3	2	1	0
R				MOD[15:8]				
W								
RESET:	1	1	1	1	1	1	1	1

[shaded] = Reserved or unused

Figure 16-10. PDB Modulus Register High (PDBMODH)

	7	6	5	4	3	2	1	0
R				MOD[7:0]				
W								
RESET:	1	1	1	1	1	1	1	1

[shaded] = Reserved or unused

Figure 16-11. PDB Modulus Register Low (PDBMODL)

16.3.2.6 PDB Counter Registers (PDBCNTH:PDBCNTL)

This is a 16-bit counter which operates on up-count mode. The two read-only counter registers contain the high and low bytes of the value in the PDB counter. Reading either byte latches the contents of both bytes into a buffer where they remain latched until PDBCNTH or PDBCNTL is read.

	7	6	5	4	3	2	1	0
R				COUNT[15:8]				
W								
RESET:	0	0	0	0	0	0	0	0
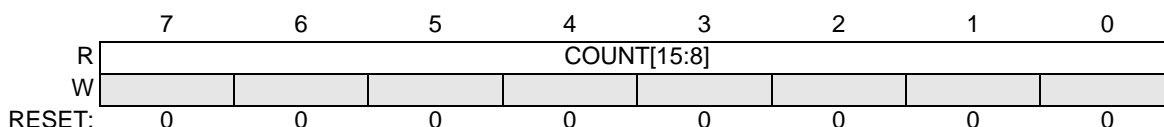
= Reserved or unused

Figure 16-12. PDB Counter Register High (PDBCNTH)

	7	6	5	4	3	2	1	0
R				COUNT[7:0]				
W								
RESET:	0	0	0	0	0	0	0	0

= Reserved or unused

Figure 16-13. PDB Counter Register Low (PDBCNTL)

16.3.2.7 PDB Interrupt Delay Register (PDBIDLYH:PDBIDLYL)

These registers can be used to read and write the value to schedule the PDB interrupt. This feature can be used to schedule an independent interrupt at some point in the PDB cycle. Reads of these registers return the value of internal registers that is taking affect for the current period of the PDB.

	7	6	5	4	3	2	1	0
R				IDELAY[15:8]				
W								
RESET:	1	1	1	1	1	1	1	1
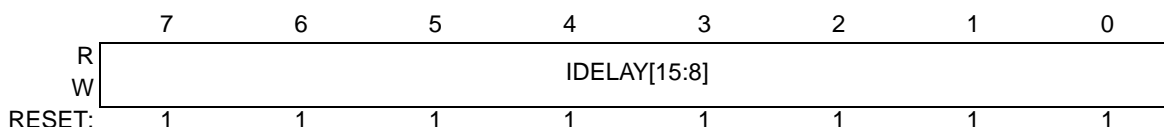
= Reserved or unused

Figure 16-14. PDB Interrupt Delay Register High (PDBIDLYH)

	7	6	5	4	3	2	1	0
R				IDELAY[7:0]				
W								
RESET:	1	1	1	1	1	1	1	1
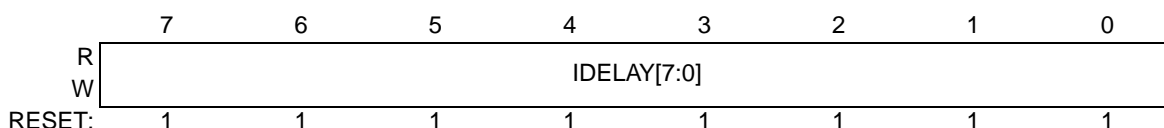
= Reserved or unused

Figure 16-15. PDB Interrupt Delay Register Low (PDBIDLYL)

PDBSC_IE must be set in order for an interrupt to be issued as a result of the count value equaling IDELAY. However, PDBSC_IF is set whenever the count value equals IDELAY.

16.3.2.8 DAC Interval Registers (DACINTH:DACINTL)

These registers specify the time between DAC hardware trigger pulses. If the DAC trigger output is enabled (DACTOE) each time the PDB counter register increments the number of counts placed in the

DACINTH:L registers, the PDB outputs a DAC trigger pulse. Reads of these registers return the value of internal registers that is taking affect for the current period of the PDB.
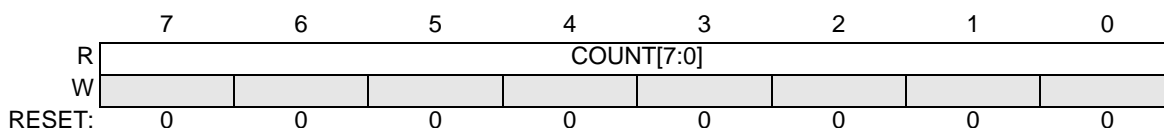
	7	6	5	4	3	2	1	0
R								
W				DACINT[15:8]				
RESET:	0	0	0	0	0	0	0	0
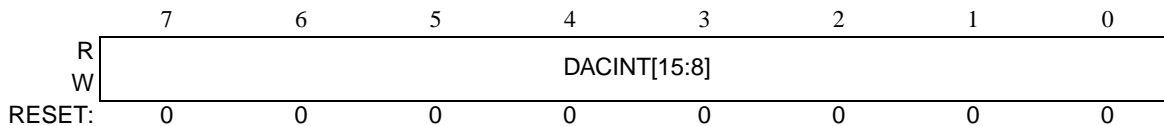
= Reserved or unused

Figure 16-16. DAC Interval Register High (DACINTH)

	7	6	5	4	3	2	1	0
R								
W				DACINT[7:0]				
RESET:	0	0	0	0	0	0	0	0

= Reserved or unused

Figure 16-17. DAC Interval Register Low (DACINTL)

16.3.2.9 PDB Delay Registers (PDBDLYH:PDBDLYL)

This register is used to specify the delay from assertion of TriggerIn to assertion of the Trigger outputs. The delay is only applicable if the module is enabled and the associated output trigger has not been bypassed. The delay is in terms of peripheral clock cycles. Reads of this register will return the value of internal registers that is taking affect for the current period of the PDB.

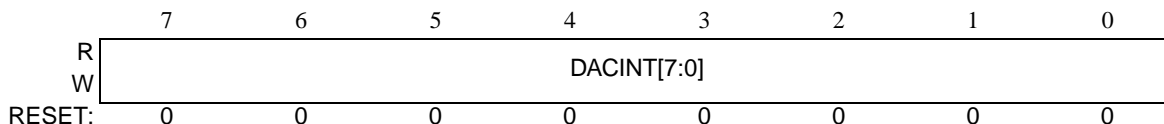|	7	6	5	4	3	2	1	0
R								
W				DELAY[15:8]				
RESET:	0	0	0	0	0	0	0	0
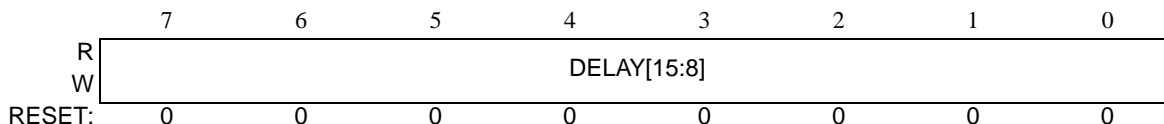
= Reserved or unused

Figure 16-18. PDB Delay n Register High (PDBDLYH)

	7	6	5	4	3	2	1	0
R								
W				DELAY[7:0]				
RESET:	0	0	0	0	0	0	0	0
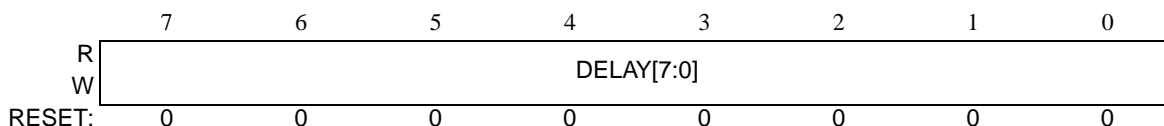
= Reserved or unused

Figure 16-19. PDB Delay n Register Low (PDBDLYL)

16.3.3 Functional Description

16.3.3.1 Impact of Using the Prescaler on Timing Resolution

Use of prescalers greater than 1 limit the count/delay accuracy in terms of peripheral clock cycles (to the modulus of the prescaler value). If the prescaler is set to div 2, then the only values of total peripheral clocks that can be detected are even values; if div is set to 4, then the only values of total peripheral clocks

that can be decoded as detected are mod (4) and so forth. If the users want to set a really long delay value and used div 128, then they would be limited to an resolution of 128 bus clocks.

Therefore, use the lowest possible prescaler for a given application.

16.4 Resets

This module has a single reset input, corresponding to the chip-wide peripheral reset. After reset, all registers are set to their reset values.

16.5 Clocks

This module has a single clock input, the peripheral bus clock.

16.6 Interrupts

This module has one interrupt source. When a successful comparison of the counter value and the IDELAY internal buffer occurs, the PDBIF is set. If the PDBIE is set, an interrupt is generated. Write a 1 to the PDBIF bit to clear the interrupt.

Chapter 17
Serial Communications Interface (S08SCIV4)

17.1 Introduction

The SCI allows asynchronous serial communications with peripheral devices and other CPUs.

NOTE

Ignore any references to stop1 low-power mode in this chapter, because this device does not support it.

For details on low-power mode operation, refer to Table 3-4 in Chapter 3, "Modes of Operation".

17.1.1 SCIx Clock Gating

The bus clock to the SCIx can be gated on and off using the SCIx bit in SCGC1. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the SCIx bit can be cleared to disable the clock to this module when not in use. See Section 5.7.8, "System Clock Gating Control 1 Register (SCGC1)," for details.

17.1.2 Module Configuration

The SCI1 and SCI2 module pins can be repositioned via software-control using SCIxPS in SOPT3 as shown in Table 17-1 and Table 17-2. SCIxPS in SOPT3 selects which general-purpose I/O ports are associated with SCI operation.

Table 17-1. SCI1 Position Options

SCI1PS in SOPT3	Port Pin for TX1	Port Pin for RX1
0 (default)	PTA1	PTA2
1	PTD6	PTD7

Table 17-2. SCI2 Position Options

SCI2PS in SOPT3	Port Pin for TX2	Port Pin for RX2
0 (default)	PTE5	PTE6
1	PTF2	PTF1

17.1.3   Module Block Diagram

Figure 17-1 shows the MC9S08JE128 series block diagram with the SCI module highlighted.
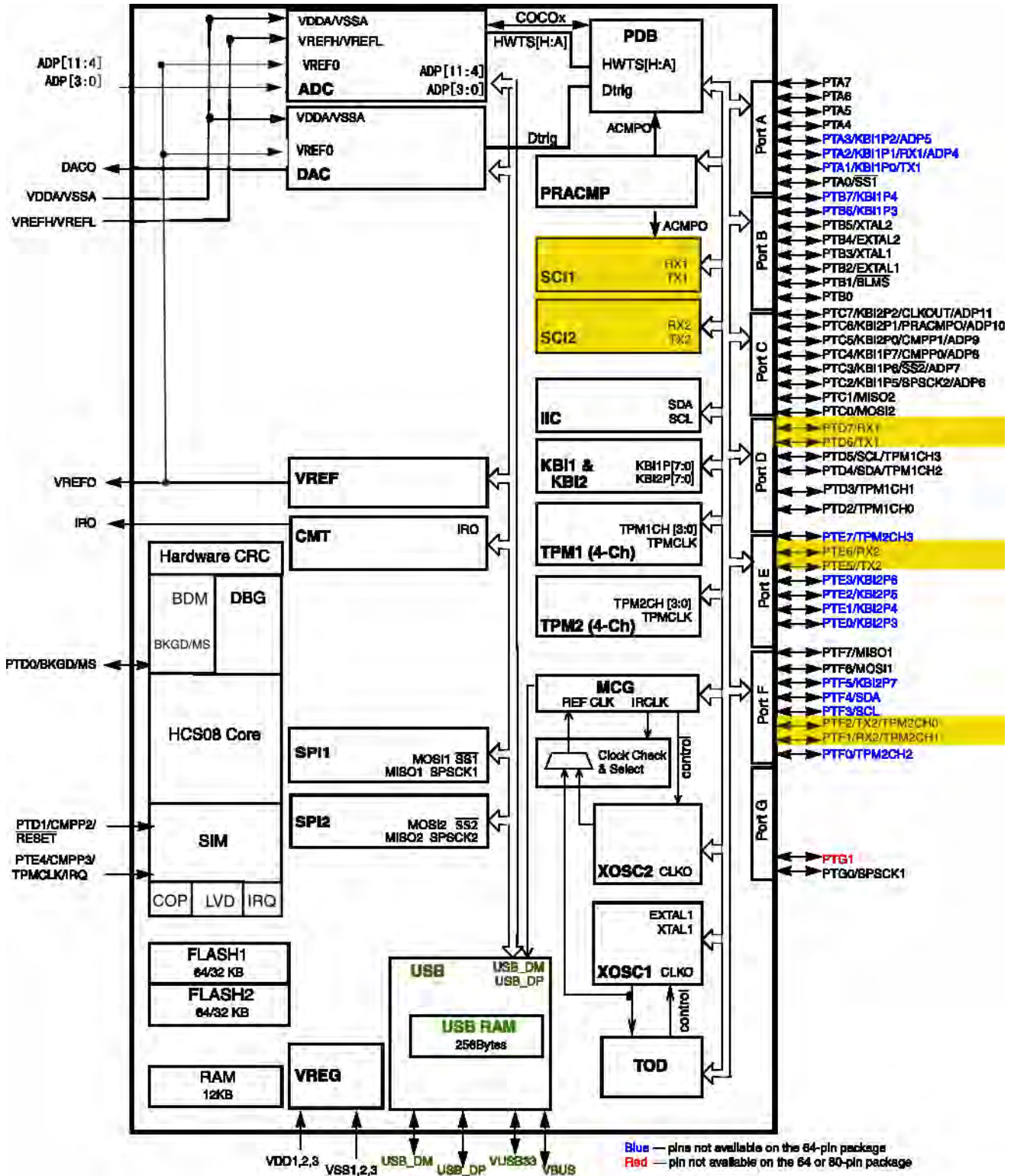
Figure 17-1. MC9S08JE128 series Block Diagram Highlighting SCI Block and Pins

17.1.4 Interfacing the SCIs to Off-Chip Opto-Isolators

SCI1 is designed with twice the normal I/O drive capability on the TX1 pin. The RX pin can either be fed directly from the digital I/O buffer, or those signals can be pre-conditioned using the comparators as shown in Figure 17-2 Similarly, the TX output can be modulated with the output of one of the timers before being passed off chip.
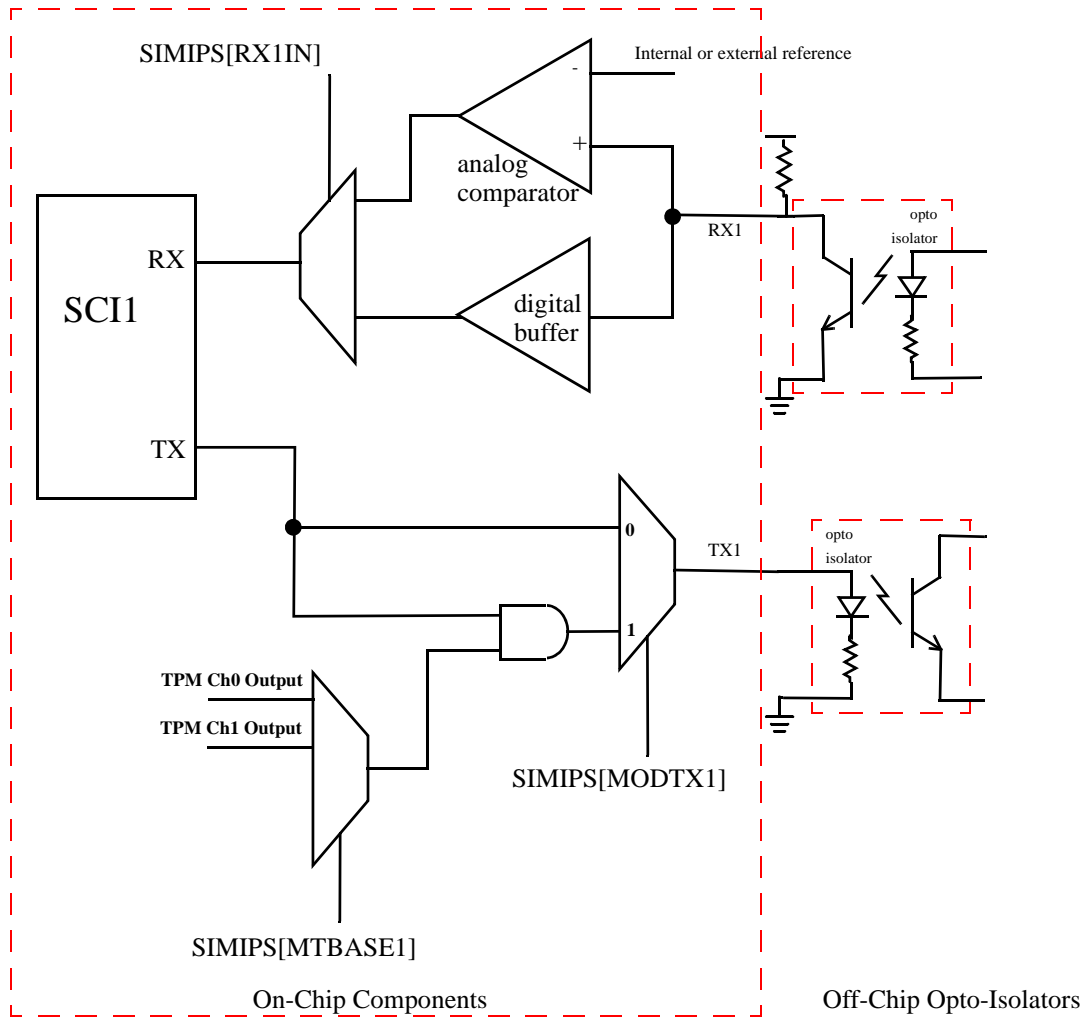
Figure 17-2. On-Chip Signal Conditioning Associated with SCI11 RX and TX Pins

Controls for the circuitry shown in Figure 17-2 are discussed in Section 5.7.14, "SIM Internal Peripheral Select Register (SIMIPS)."

17.1.5 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

17.1.6   Modes of Operation

See Section 17.3, "Functional Description," for details concerning SCI operation in the following modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

17.1.7   Block Diagram

Figure 17-3 shows the transmitter portion of the SCI.
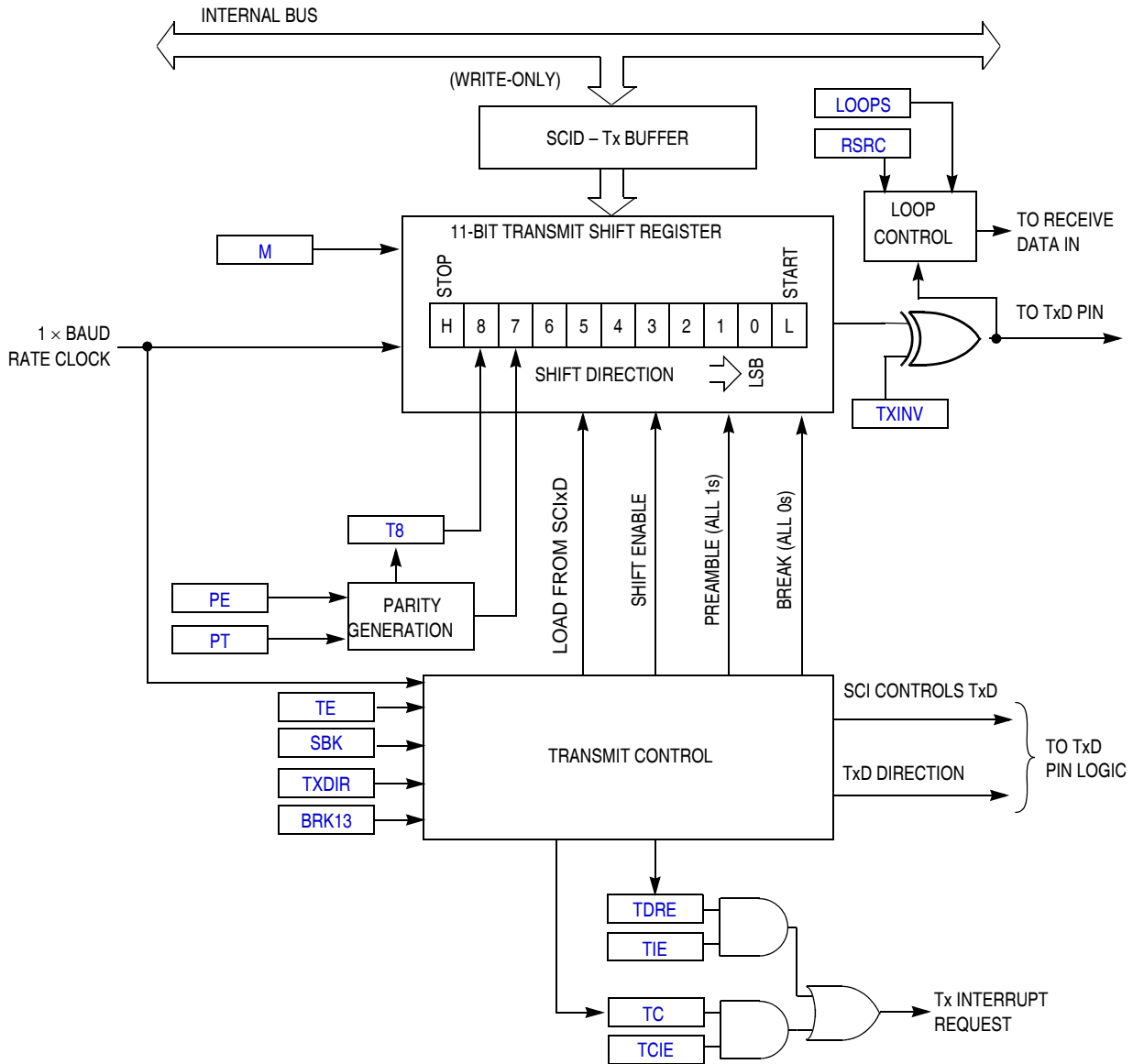
Figure 17-3. SCI Transmitter Block Diagram

Figure 17-4 shows the receiver portion of the SCI.
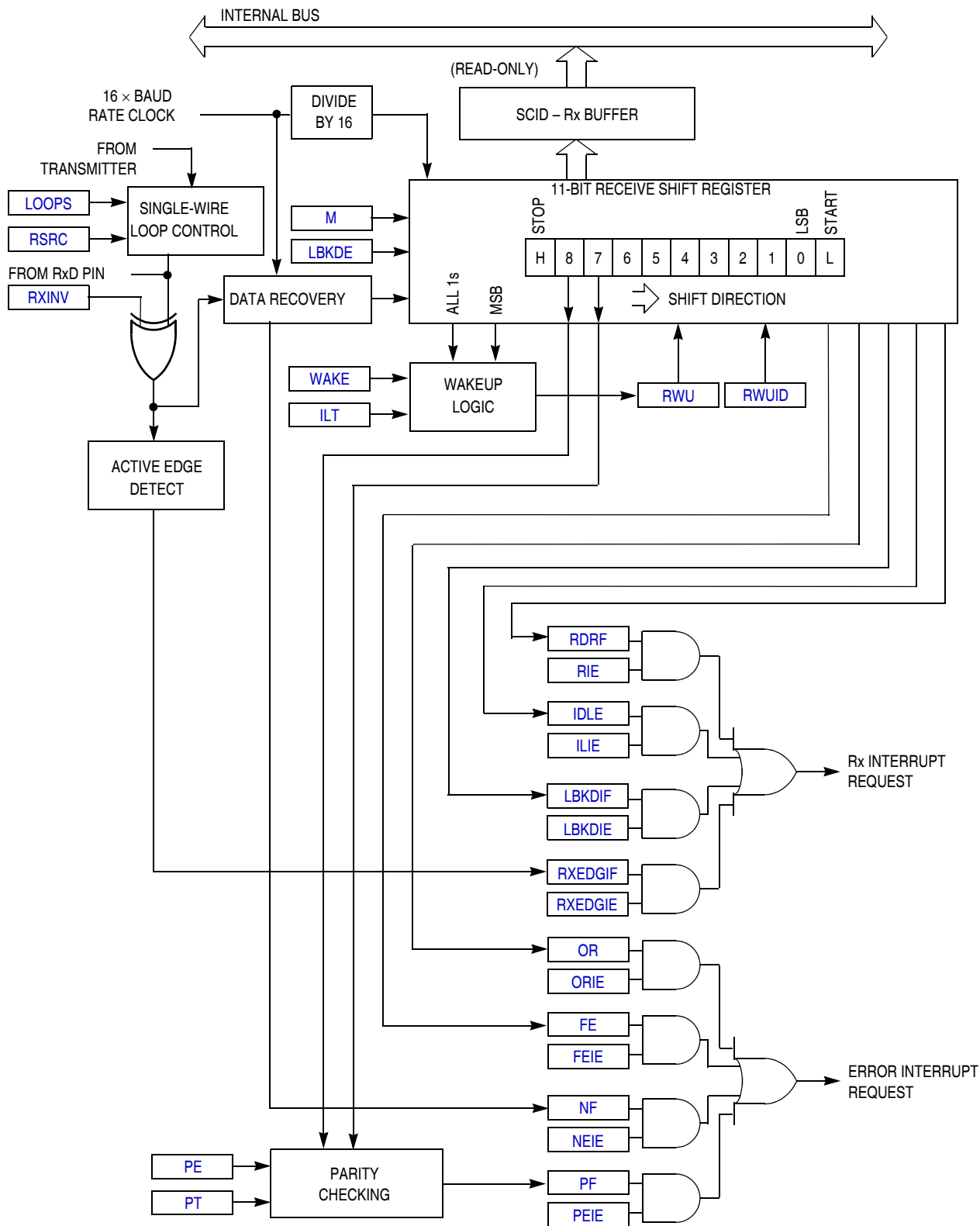
Figure 17-4. SCI Receiver Block Diagram

17.2 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 the *MC9S08JE128 series Reference Manual* 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.

17.2.1 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
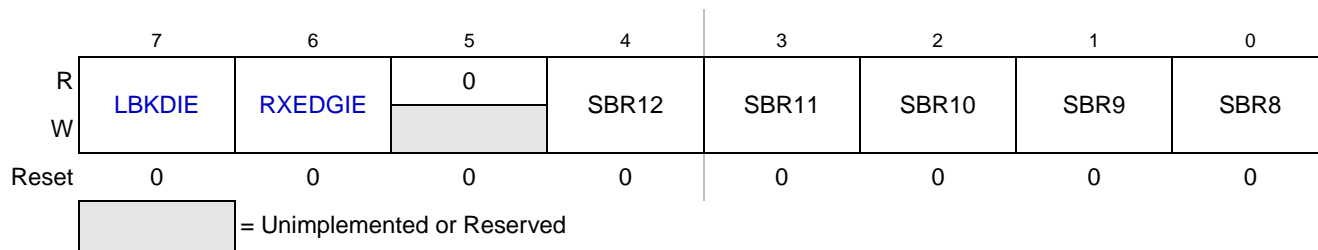
Figure 17-5. SCI Baud Rate Register (SCIxBDH)

Table 17-3. 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 17-4.

	7	6	5	4	3	2	1	0
R	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
W								
Reset	0	0	0	0	0	1	0	0

Figure 17-6. SCI Baud Rate Register (SCIxBDL)

Table 17-4. SCIxBDL Field Descriptions

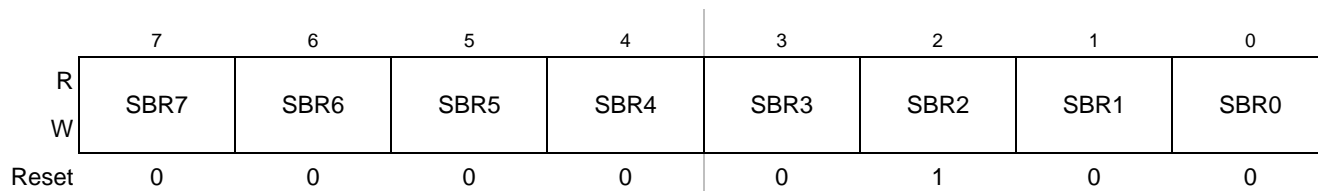| Field	Description
7:0 SBR[7:0]	**Baud Rate Modulo Divisor** — These 13 bits in SBR[12:0] are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 17-3.

17.2.2   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	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
W								
Reset	0	0	0	0	0	0	0	0
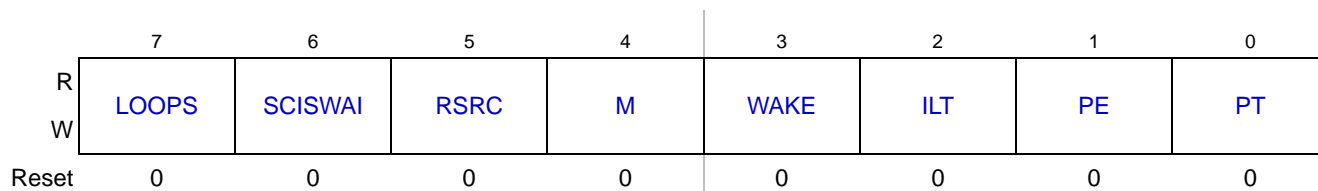
Figure 17-7. SCI Control Register 1 (SCIxC1)

Table 17-5. 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 that follows.) 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 17-5. SCIxC1 Field Descriptions (Continued)

Field	Description
3 WAKE	**Receiver Wakeup Method Select** — Refer to Section 17.3.3.2, "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 17.3.3.2.1, "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.

17.2.3 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
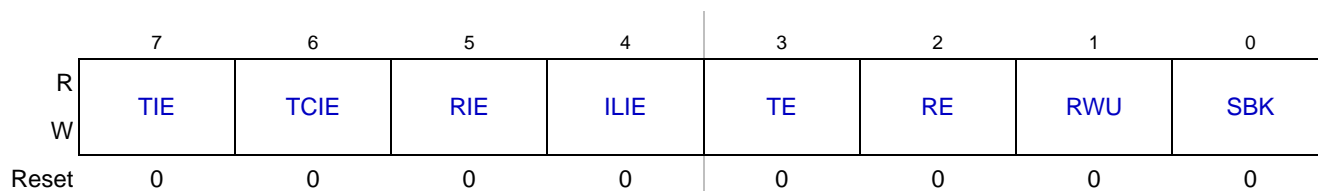
Figure 17-8. SCI Control Register 2 (SCIxC2)

Table 17-6. 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 17-6. 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 17.3.2.1, "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 17.3.3.2, "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 17.3.2.1, "Send Break and Queued Idle" for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

17.2.4   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) clear these status flags.

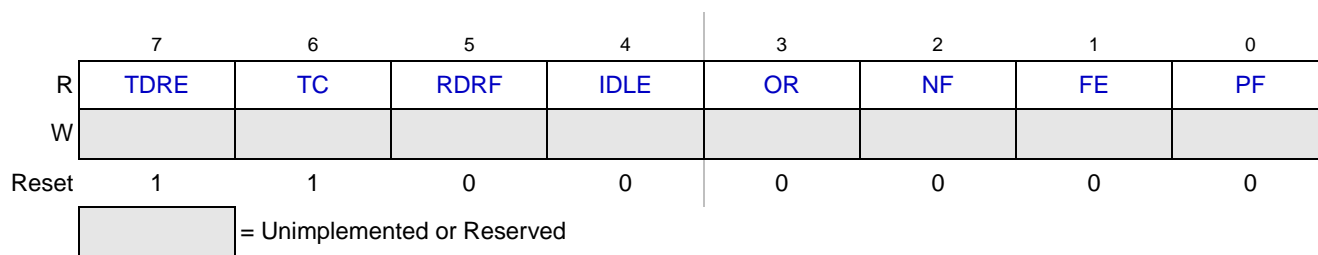|	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W								
Reset	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 17-9. SCI Status Register 1 (SCIxS1)

Table 17-7. SCIxS1 Field Descriptions

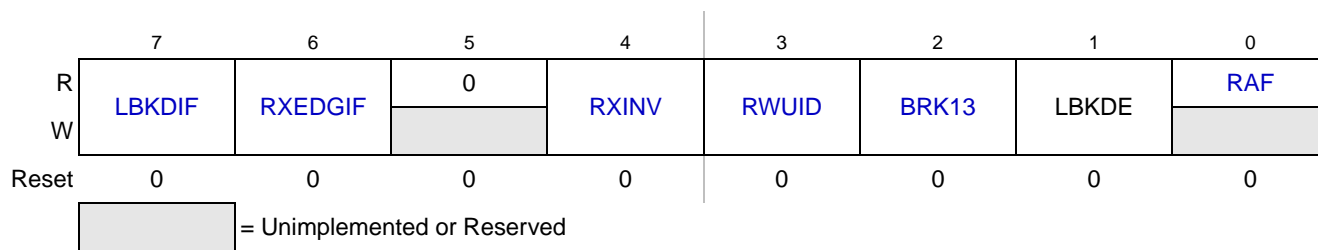| Field	Description
7 TDRE	**Transmit Data Register Empty Flag** — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIxS1 with TDRE = 1 and then write to the SCI 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 is set at the same time as the RDRF flag is 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 17-7. 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.

17.2.5   SCI Status Register 2 (SCIxS2)

This register contains 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 17-10. SCI Status Register 2 (SCIxS2)

Table 17-8. 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 selects a longer transmitted break character length. The state of this bit does not affect the detection of a framing error. 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 17-8. SCIxS2 Field Descriptions (Continued)

Field	Description
1 LBKDE	**LIN Break Detection Enable**— LBKDE selects 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 running 14% faster than the master. This would trigger normal break detection circuitry 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.

17.2.6 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
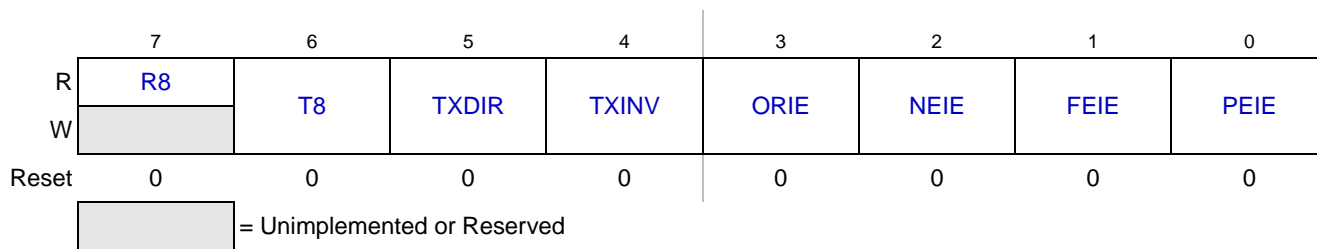
Figure 17-11. SCI Control Register 3 (SCIxC3)

Table 17-9. 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 17-9. 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.

17.2.7 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 17-12. SCI Data Register (SCIxD)

17.3 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.

17.3.1 Baud Rate Generation

As shown in Figure 17-13, the clock source for the SCI baud rate generator is the bus-rate clock.
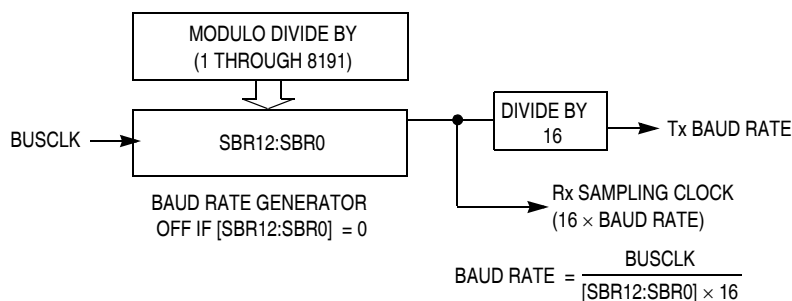
Figure 17-13. SCI Baud Rate Generation

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. 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 SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about ± 4.5 % for 8-bit data format and about ± 4 % 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.

17.3.2 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 17-3.

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 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, 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 can 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 in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

17.3.2.1   Send Break and Queued Idle

The SBK control bit in SCIxC2 sends break characters 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 remains 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 are 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 shared with TxD is an output driving a logic 1. This ensures that the TxD line looks 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 17-10. 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

17.3.3   Receiver Functional Description

In this section, the receiver block diagram (Figure 17-4) is 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.

Set RXINV = 1 to invert the receiver input. Set the RE bit in SCIxC2 to enable the receiver. 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 •, "8- and 9-bit data modes." For the remainder of this discussion, 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 reads SCIxD to get the data from the receive data register. The RDRF flag is cleared automatically when the program that handles receive data issues a 2-step sequence. Refer to Section 17.3.4, "Interrupts and Status Flags" for more details about flag clearing.

17.3.3.1 Data Sampling Technique

The SCI receiver uses a 16× baud rate clock for sampling. To search for a falling edge on the RxD serial data input pin, the receiver starts taking logic level samples at 16 times the baud rate. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The 16× baud rate clock divides 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) is set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges. 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 remains set.

17.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message 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. This eliminates the need for software overhead to handle 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.

17.3.3.2.1 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 sets 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.

17.3.3.2.2 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 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.

17.3.4 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. A third vector is used for OR, NF, FE, and PF error conditions. Local interrupt enable masks can separately mask each of these ten interrupt sources. Software can still poll the flags when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates available 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 is requested when 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) — are 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 is set instead of 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).

17.3.5   Additional SCI Functions

The following sections describe additional SCI functions.

17.3.5.1   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.

Typically, use the 9-bit data mode in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or use it 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.

17.3.5.2   Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In 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 remains 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 resumes 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.

17.3.5.3   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.

17.3.5.4   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 18
16-bit Serial Peripheral Interface 1 (S08SPI16V2)

18.1 Introduction

The SPI1 in MC9S08JE128 series MCUs is a 16-bit serial peripheral interface (SPI) module with FIFO.

NOTE

There are two SPI modules on this device. Replace SPIx with the appropriate peripheral designation (SPI1 or SPI2), depending upon your use:

- SPI1 — for the 16-bit SPI with FIFO module.
- SPI2 — for the 8-bit SPI module.

18.1.1 SPI1 Clock Gating

The bus clock to the SPI1 can be gated on and off using the SPI1 bit in SCGC2. These bits are set after any reset, which enables the bus clock to this module. To conserve power, these bits can be cleared to disable the clock to this module when not in use. See Section 5.7.9, "System Clock Gating Control 2 Register (SCGC2)," for details.
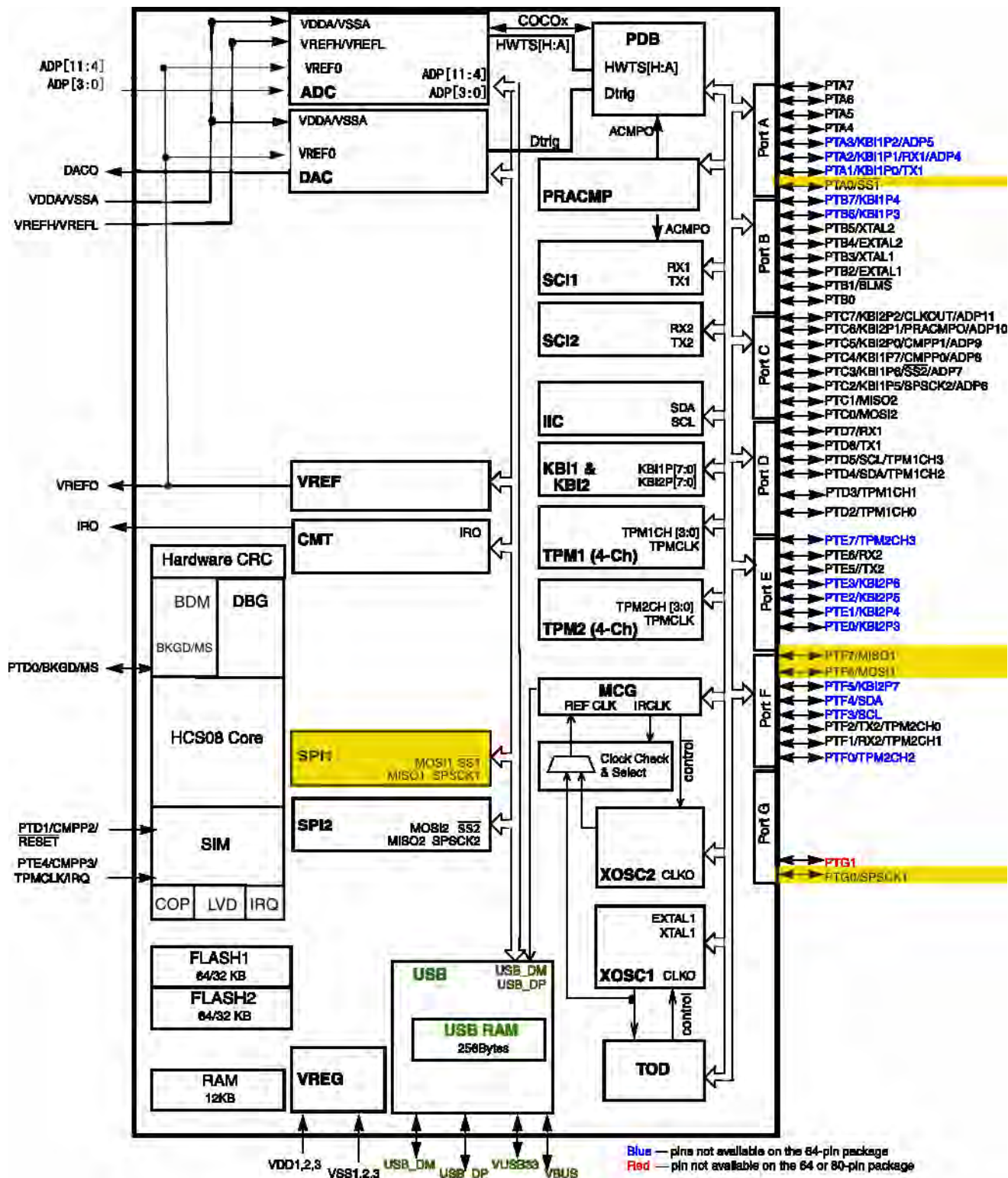
Figure 18-1. Block Diagram Highlighting SPI1 Block

18.1.2 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
- 64-bit FIFO mode for high speed/large amounts of data transfers.

18.1.3 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/stop4 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 18.4.10, "Low-power Mode Options."

18.1.4 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.

18.1.4.1 SPI System Block Diagram

Figure 18-2 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.
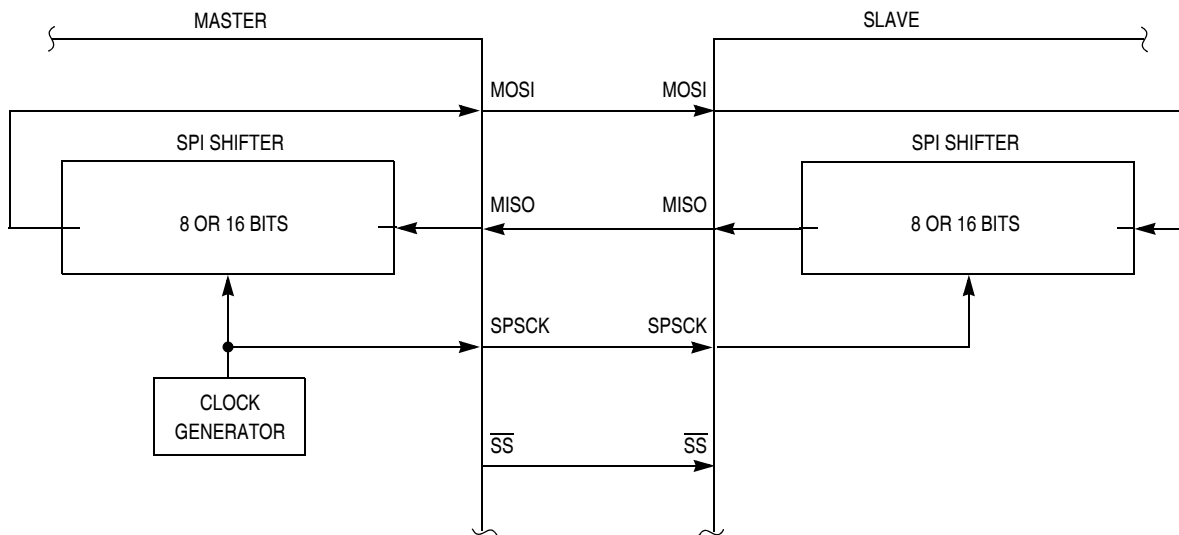
Figure 18-2. SPI System Connections

18.1.4.2 SPI Module Block Diagram

Figure 18-3 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.

Additionally there is an 8-byte receive FIFO and an 8-byte transmit FIFO that once enabled provide features to allow less CPU interrupts to occur when transmitting/receiving high volume/high speed data. When FIFO mode is enabled, the SPI can still function in either 8-bit or 16-bit mode (as per **SPIMODE** bit) and 3 additional flags help monitor the FIFO status and two of these flags can provide CPU interrupts.

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.
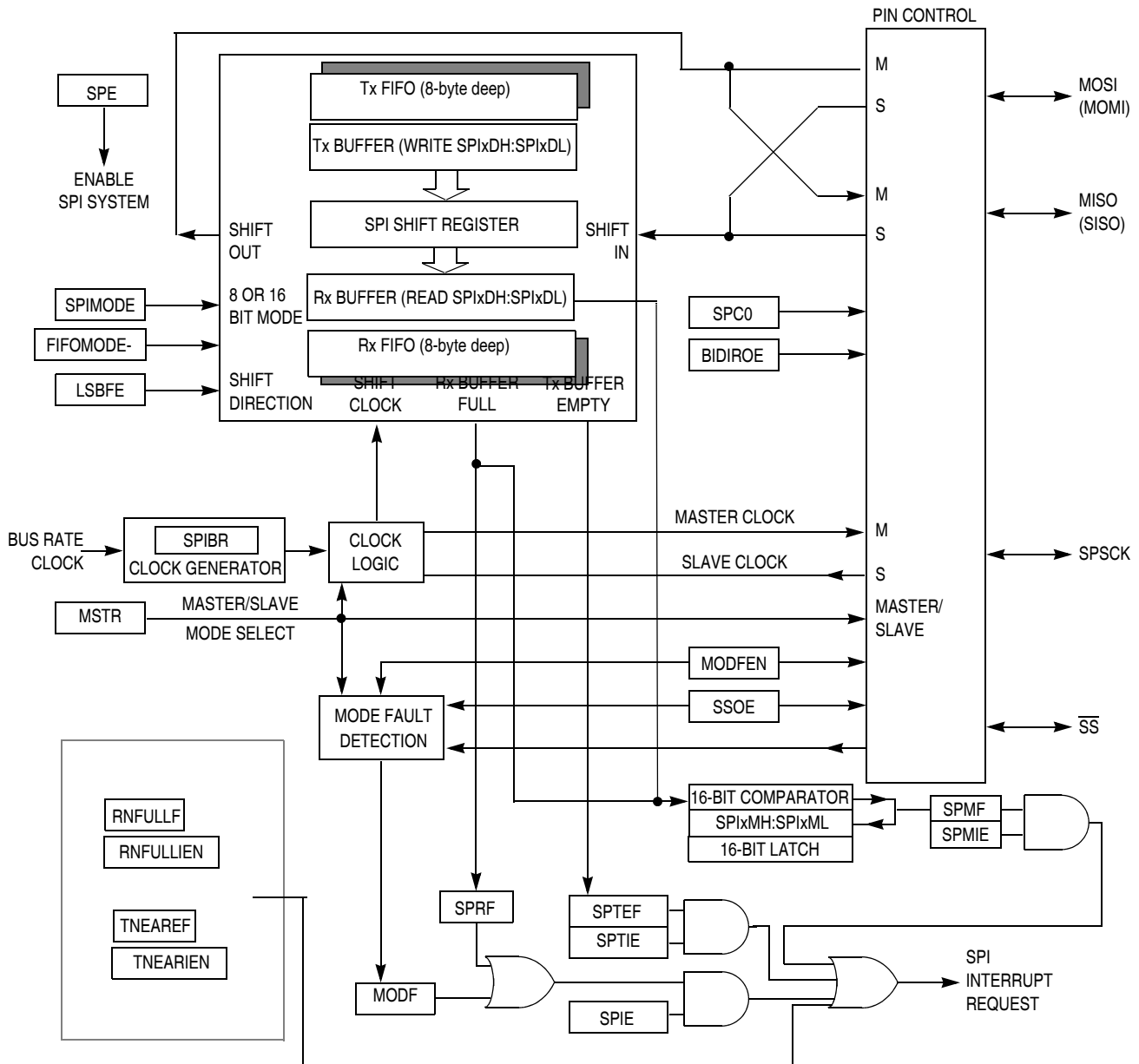
Figure 18-3. SPI Module Block Diagram

18.2 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.

18.2.1 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.

18.2.2 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.

18.2.3 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.

18.2.4 $\overline{SS}$ — 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).

18.3 Register Definition

The SPI has above 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.

18.3.1 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
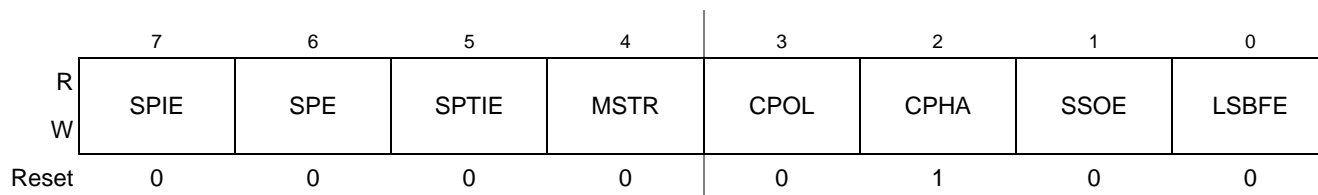
Figure 18-4. SPI Control Register 1 (SPIxC1)

Table 18-1. SPIxC1 Field Descriptions

Field	Description
7 SPIE	**FIFOMODE=0** **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 **FIFOMODE=1** **SPI Read FIFO Full Interrupt Enable** — This bit when set enables the SPI to interrupt the CPU when the Receive FIFO is full. An interrupt will occur when SPRF flag is set or MODF is set. 0 Read FIFO Full Interrupts are disabled 1 Read FIFO Full Interrupts are enabled
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** — **FIFOMODE=0** This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). An interrupt occurs when the SPI transmit buffer is empty (SPTEF is set) **FIFOMODE=1** This is the interrupt enable bit for SPI transmit FIFO empty (SPTEF). An interrupt occurs when the SPI transmit FIFO is empty (SPTEF is set) 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 18.4.6, "SPI Clock Formats" for more details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)

Table 18-1. SPIxC1 Field Descriptions (Continued)

Field	Description
2 CPHA	**Clock Phase** — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to Section 18.4.6, "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 18-2.
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 18-2. $\overline{SS}$ 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

18.3.2   SPI Control Register 2 (SPIxC2)

This read/write register is used to control optional features of the SPI system. Bits 5 and 2 are not implemented and always read 0.

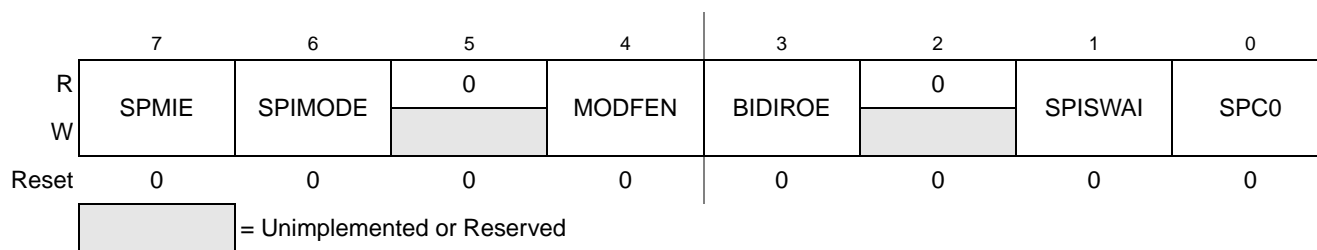|	7	6	5	4	3	2	1	0
R	SPMIE	SPIMODE	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 18-5. SPI Control Register 2 (SPIxC2)

Table 18-3. 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 18.4.5, "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 18-2 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 18-4. 0 SPI uses separate pins for data input and data output. 1 SPI configured for single-wire bidirectional operation.

Table 18-4. 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	

18.3.3   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	SPR3	SPR2	SPR1	SPR0
W		SPPR2	SPPR1	SPPR0	SPR3	SPR2	SPR1	SPR0
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved
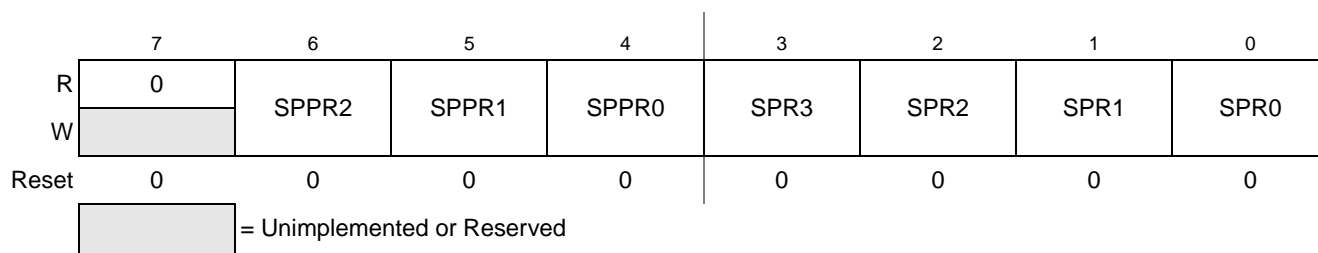
Figure 18-6. SPI Baud Rate Register (SPIxBR)

Table 18-5. 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 18-6. 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 18-19). See Section 18.4.7, "SPI Baud Rate Generation," for details.
3:0 SPR[3:0]	**SPI Baud Rate Divisor** — This 4-bit field selects one of nine divisors for the SPI baud rate divider as shown in Table 18-7. The input to this divider comes from the SPI baud rate prescaler (see Figure 18-19). See Section 18.4.7, "SPI Baud Rate Generation," for details.

Table 18-6. 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 18-7. SPI Baud Rate Divisor

SPR3:SPR2:SPR1:SPR0	Rate Divisor
0:0:0:0	2
0:0:0:1	4
0:0:1:0	8
0:0:1:1	16
0:1:0:0	32
0:1:0:1	64
0:1:1:0	128
0:1:1:1	256
1:0:0:0	512
All other combinations	Reserved

18.3.4   SPI Status Register (SPIxS)

This register has eight read-only status bits. Writes have no meaning or effect.

	7	6	5	4	3	2	1	0
R	SPRF	SPMF	SPTEF	MODF	RNFULLF	TNEAREF	TXFULLF	RFIFOEF
W								
Reset	0	0	1	0	0	0[1]	0	0[1]

= Unimplemented or Reserved

Figure 18-7. SPI Status Register (SPIxS)

[1] Note: PoR values of TNEAREF and RFIFOEF is 0. If status register is reset due to change of SPIMODE, FIFOMODE or SPE than, if FIFOMODE = 1, TNEAREF and RFIFOEF resets to 1 else if FIFOMODE = 0, TNEAREF and RFIFOEF resets to 0

This register has 4 additional flags RNFULLF, TNEARF, TXFULLF and RFIFOEF which provide mechanisms to support an 8-byte FIFO mode. When in 8-byte FIFO mode, the function of SPRF and SPTEF differ slightly from the normal buffered modes, mainly in how these flags are cleared by the amount available in the transmit and receive FIFOs.

Table 18-8. 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. **FIFOMODE=1** **SPI Read FIFO FULL Flag** — This bit indicates the status of the Read FIFO when FIFOMODE enabled. The SPRF is set when the read FIFO has received 64bits (4 words or 8 bytes) of data from the shifter and there has been no CPU reads of SPIxDH:SPIxDL. SPRF is cleared by reading the SPI Data Register, which empties the FIFO, assuming another SPI message is not received. 0 Read FIFO is not Full 1 Read FIFO is Full.
6 SPMF	**SPI Match Flag** — SPMF is set after SPRF = 1 when the value in the receive data buffer matches the value in SPIxMH:SPIxML. 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 **FIFOMODE=1** **SPI Transmit FIFO Empty Flag** — *This bit when in FIFOMODE now changed to provide status of the FIFO rather than an 8or16-bit buffer.* This bit is set when the Transmit FIFO is empty. It is cleared by writing a data value to the transmit FIFO at SPIxDH:SPIxDL. SPTEF is automatically set when all data from transmit FIFO 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, a second write of data to the SPIxDH:SPIxDL will clear this SPTEF flag. After completion of the transfer of the data in the shift register, the queued data from the transmit FIFO will automatically move to the shifter and SPTEF will be set only when all data written to the transmit FIFO has been transferred to the shifter. If no new data is waiting in the transmit FIFO, SPTEF simply remains set and no data moves from the buffer to the shifter. 0 SPI FIFO not empty 1 SPI FIFO 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

Table 18-8. SPIxS Register Field Descriptions

Field	Description
3 RNFULLF	**Receive FIFO Nearly Full Flag** — This flag is set when more than three16bit words or six 8bit bytes of data remain in the receive FIFO provided SPIxC3[4] = 0 or when more than two 16bit words or four 8bit bytes of data remain in the receive FIFO provided SPIxC3[4] = 1. It has no function if FIFOMODE=0. 0 Receive FIFO has received less than 48bits/32bits (See SPIxC3[4]). 1 Receive FIFO has received 48bits/32bits(See SPIxC3[4]) or more.
2 TNEAREF	**Transmit FIFO Nearly Empty Flag** — This flag is set when only one 16bit word or 2 8bit bytes of data remain in the transmit FIFO provided SPIxC3[5] = 0 or when only two 16bit words or 4 8bit bytes of data remain in the transmit FIFO provided SPIxC3[5] =1. If FIFOMODE is not enabled this bit should be ignored. 0 Transmit FIFO has more than 16bits/32bits (See SPIxC3[5]) left to transmit. 1 Transmit FIFO has 16bits/32 bits (See SPIxC3[5])or less left to transmit
1 TXFULLF	**Transmit FIFO Full Flag** - This bit indicates status of transmit FIFO when FIFO mode is enabled. This flag is set when there are 8 bytes in transmit FIFO. If FIFOMODE is not enabled this bit should be ignored. 0 Transmit FIFO has less than 8 bytes. 1 Transmit FIFO has 8 bytes of data.
0 RFIFOEF	**SPI Read FIFO Empty Flag** — This bit indicates the status of the Read FIFO when FIFOMODE enabled. If FIFOMODE is not enabled this bit should be ignored. 0 Read FIFO has data. Reads of the SPIxDH:SPIxDL registers in 16-bit mode or SPIxDL register in 8-bit mode will empty the Read FIFO. 1 Read FIFO is empty.

For FIFO management there are two other important flags that are used to help make the operation more efficient when transferring large amounts of data. These are the Receive FIFO Nearly Full Flag (**RNFULLF**) and the Transmit FIFO Nearly Empty Flag (**TNEAREF**). Both these flags provide a "watermark" feature of the FIFOs to allow continuous transmissions of data when running at high speed.

The **RNFULLF** flag can generate an interrupt if the **RNFULLIEN** bit in the **SPIxC3** Register is set which allows the CPU to start emptying the Receive FIFO without delaying the reception of subsequent bytes. The user can also determine if all data in Receive FIFO has been read by monitoring the **RFIFOEF** flag.

The **TNEAREF** flag can generate an interrupt if the **TNEARIEN** bit n the **SPIxC3** Register is set which allows the CPU to start filling the Transmit FIFO before it is empty and thus provide a mechanism to have no breaks in SPI transmission.

NOTE

SPIxS and both TX and RX FIFOs gets reset due to change in SPIMODE, FIFOMODE or SPE. PoR values of SPIxS are show in Figure 18-7 and Figure 18-8.

Figure 18-7 and Figure 18-8 shows the reset values due to change of modes after PoR.

	7	6	5	4	3	2	1	0
R	SPRF	SPMF	SPTEF	MODF	RNFULLF	TNEAREF	TXFULLF	RFIFOEF
W								
Reset	0	0	1	0	0	0	0	0

Figure 18-8. Reset values of SPIxS after PoR with FIFOMODE = 0

	7	6	5	4	3	2	1	0
R	SPRF	SPMF	SPTEF	MODF	RNFULLF	TNEAREF	TXFULLF	RFIFOEF
W								
Reset	0	0	1	0	0	1	0	1

Figure 18-9. Reset values of SPIxS after PoR with FIFOMODE = 1

18.3.5   SPI Data Registers (SPIxDH:SPIxDL)

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

Figure 18-10. SPI Data Register High (SPIxDH)

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

Figure 18-11. 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.

MC9S08JE128 Series Reference Manual, Rev. 3   Freescale Semiconductor

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.

18.3.6   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	Bit 15	14	13	12	11	10	9	Bit 8
W								
Reset	0	0	0	0	0	0	0	0
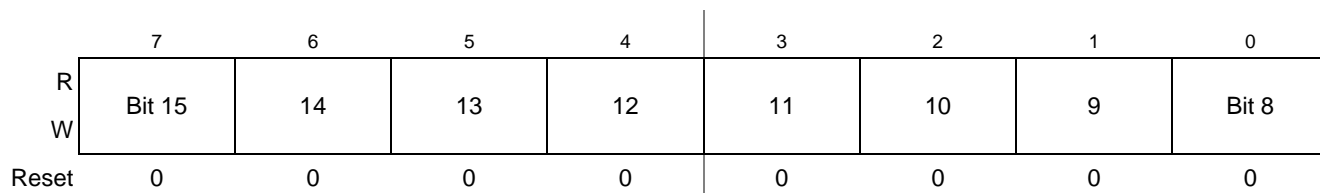
Figure 18-12. SPI Match Register High (SPIxMH)

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

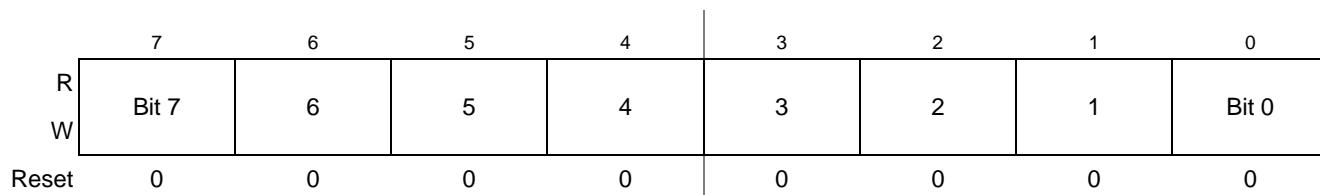**Figure 18-13. SPI Match Register Low (SPIxML)**

18.3.7   SPI Control Register 3 (SPIxC3) — enable FIFO feature

The SPI Control Register 3 introduces a 64-bit FIFO function on both transmit and receive buffers to be utilized on the SPI. Utilizing this FIFO feature allows the SPI to provide high speed transfers of large amounts of data without consuming large amounts of the CPU bandwidth.

Enabling this FIFO function will effect the behavior of some of the Read/Write Buffer flags in the SPIxS register namely:

- The **SPRF** of the **SPIxS** register will be set when the Receive FIFO is filled and will interrupt the CPU if the **SPIE** in the **SPIxC1** register is set.

and

- The **SPTEF** of the **SPIxS** register will be set when the Transmit FIFO is empty, and will interrupt the CPU if the **SPTIE** bit is set in the **SPIxC1** register. See **SPIxC1** and **SPIxS** registers.

FIFO mode is enabled by setting the **FIFOMODE** bit, and provides the SPI with an 8-byte receive FIFO and an 8-byte transmit FIFO to reduce the amount of CPU interrupts for high speed/high volume data transfers.

Two interrupt enable bits **TNEARIEN** and **RNFULLIEN** provide CPU interrupts based on the "watermark" feature of the **TNEARF** and **RNFULLF** flags of the SPIxS register.

Note: This register has six read/write control bits. Bits 7 through 6 are not implemented and always read 0. Writes have no meaning or effect. Write to this register happens only when FIFOMODE bit is 1.
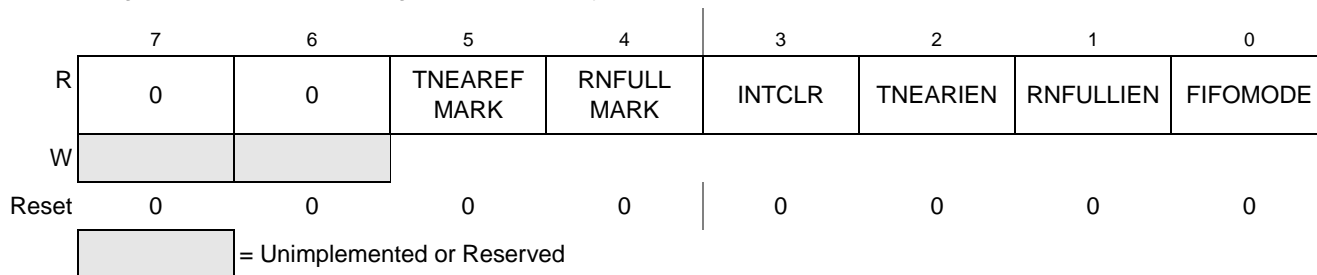
	7	6	5	4	3	2	1	0
R	0	0	TNEAREF MARK	RNFULL MARK	INTCLR	TNEARIEN	RNFULLIEN	FIFOMODE
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 18-14. SPI Control Register (SPIxC3S)

Table 18-9. SPIxC3S Register Field Descriptions

Field	Description
5 TNEAREF MARK	**Transmit FIFO Nearly Empty Water Mark** — This bit selects the mark after which TNEAREF flag is asserted. 0 TNEAREF is set when Transmit FIFO has16bits or less. 1 TNEAREF is set when Transmit FIFO has 32bits or less.
4 RNFULLF MARK	**Receive FIFO Nearly Full Water Mark** — This bit selects the mark for which RNFULLF flag is asserted 0 RNFULLF is set when Receive FIFO has 48bits or more 1 RNFULLF is set when Receive FIFO has 32bits or more.
3 INTCLR	**Interrupt Clearing Mechanism Select** — This bit selects the mechanism by which SPRF, SPTEF, TNEAREF, RNFULLF interrupts gets cleared. 0 Interrupts gets cleared when respective flags gets cleared depending on the state of FIFOs 1 Interrupts gets cleared by writing to the SPIxCI respective bits.
2 TNEARIEN	**Transmit FIFO Nearly Empty Interrupt Enable** — Writing to this bit enables the SPI to interrupt the CPU when the TNEAREF flag is set. This is an additional interrupt on the SPI and will only interrupt the CPU if SPTIE in the SPIxC1 register is also set. This bit is ignored and has no function if FIFOMODE=0. 0 No interrupt on Transmit FIFO Nearly Empty Flag being set. 1 Enable interrupts on Transmit FIFO Nearly Empty Flag being set.

Table 18-9. SPIxC3S Register Field Descriptions

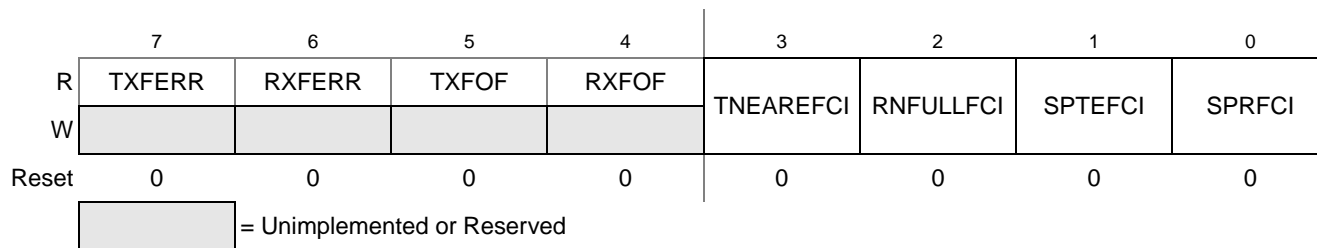| Field	Description
1 RNFULLIEN	**Receive FIFO Nearly Full Interrupt Enable** — Writing to this bit enables the SPI to interrupt the CPU when the RNEARFF flag is set. This is an additional interrupt on the SPI and will only interrupt the CPU if SPIE in the SPIxC1 register is also set. This bit is ignored and has no function if FIFOMODE=0. 0 No interrupt on RNEARFF being set. 1 Enable interrupts on RNEARFF being set.
0 FIFOMODE	**SPI FIFO Mode Enable** — This bit enables the SPI to utilize a 64bit FIFO (8 bytes 4 16-bit words) for both transmit and receive buffers. 0 Buffer mode disabled. 1 Data available in the receive data buffer.

18.3.8 SPI Clear Interrupt Register (SPIxCI)

The SPI Clear Interrupt register has 4 bits dedicated for clearing the interrupts. Writing 1 to these bits clears the respective interrupts if INTCLR bit in SPIxCR3 is set.

It also have 2 bits to indicate the transmit FIFO and receive FIFO overrun conditions. When receive FIFO is full and a data is received RXFOF flag is set. Similarly, when transmit FIFO is full and write happens to SPIDR TXFOF is set. These flags get cleared when a read happens to this register with the flags set.

There are two more bits to indicate the error flags. These flags gets set when due to some spurious reasons entries in FIFO becomes greater than 8. At this point all the flags in status register gets reset and entries in FIFO are flushed with respective error flags set. These flags are cleared when a read happen at SPIxCI with the error flags set.

Note: Bits [7:4] are read-only bits. These bits gets cleared when a read happens to this register with the flags set. Bits [3:0] are clear interrupts bits which clears the interrupts by writing 1 to respective bits. Reading these bits always return 0.

	7	6	5	4	3	2	1	0
R	TXFERR	RXFERR	TXFOF	RXFOF	TNEAREFCI	RNFULLFCI	SPTEFCI	SPRFCI
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Table 18-10. SPIxCI Register Field Descriptions

Field	Description
7 TXFERR	**Transmit FIFO Error Flag**- This flag indicates that TX FIFO error occurred because entries in FIFO goes above 8. 0 No TX FIFO error occurred. 1 TX FIFO error occurred.
6 RXFERR	**Receive FIFO Error Flag**- This flag indicates that RX FIFO error occurred because entries in FIFO goes above 8. 0 No RX FIFO error occurred. 1 RX FIFO error occurred.

Table 18-10. SPIxCI Register Field Descriptions

Field	Description
5 TXFOF	**TX FIFO Overflow Flag**- This Flag indicates that TX FIFO overflow condition has occurred. 0 TX FIFO overflow condition has not occurred. 1 TX FIFO overflow condition occurred.
4 RXFOF	**RX FIFO Overflow Flag** - This Flag indicates that RX FIFO overflow condition has occurred. 0 RX FIFO overflow condition has not occurred. 1 RX FIFO overflow condition occurred.
3 TNEAREFCI	**Transmit FIFO Nearly Empty Flag Clear Interrupt Register** — Write of 1 clears the TNEAREF interrupt provided SPIxC3[3] is set.
2 RNFULLFCI	**Receive FIFO Nearly Full Flag Clear Interrupt Register** — Write of 1 clears the RNFULLF interrupt provided SPIxC3[3] is set.
1 SPTEFCI	**Transmit FIFO Empty Flag Clear Interrupt Register** — Write of 1 clears the SPTEF interrupt provided SPIxC3[3] is set.
0 SPRFCI	**Receive FIFO Full Flag Clear Interrupt Register** — Write of 1 clears the TNEAREF interrupt provided SPIxC3[3] is set.

18.4 Functional Description

18.4.1 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.

18.4.2 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 SPR3, 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 18.4.6, "SPI Clock Formats).

NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, BIDIROE with SPC0 set, SPIMODE, FIFOMODE, SPPR2-SPPR0 and SPR3-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.

18.4.3   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 FIFOMODE and SPIMODE in slave mode will corrupt a transmission in progress and has to be avoided.
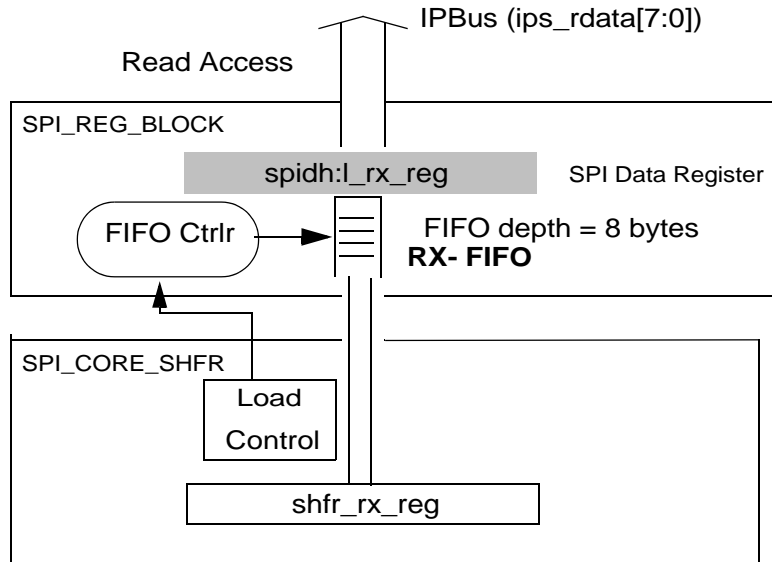
18.4.4 SPI FIFO MODE

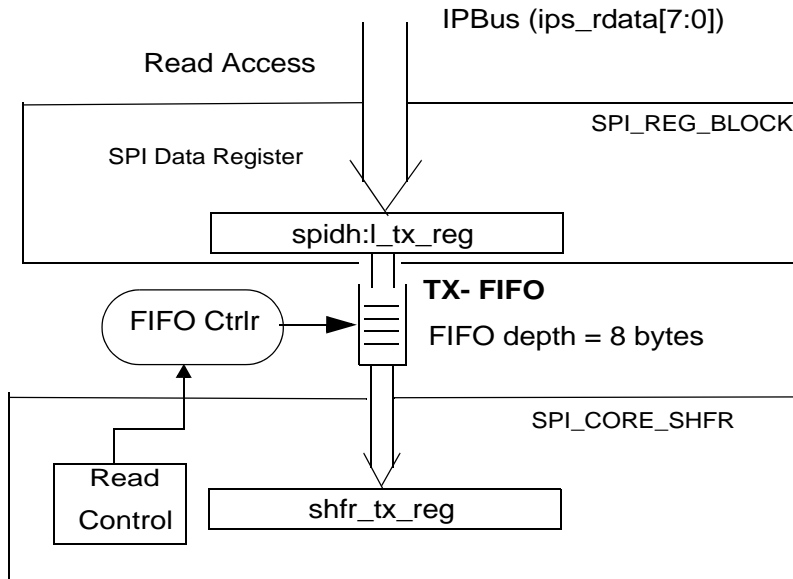**Figure 18-15. SPIH:L read side structural overview in FIFO mode**

Figure 18-16. SPIH:L write side structural overview in FIFO mode

SPI works in FIFO mode when SPIxC3[0] bit is set. When in FIFO mode SPI RX buffer and SPI TX buffer is replaced by a 8 byte deep FIFO as shown in figure above.

18.4.5   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. There is no buffer mechanism for the reading of SPIxMH and SPIxML since they can only be changed by writing at CPU side. 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.

18.4.6   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 18-17 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 eighth 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 18-17. 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 18-18 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
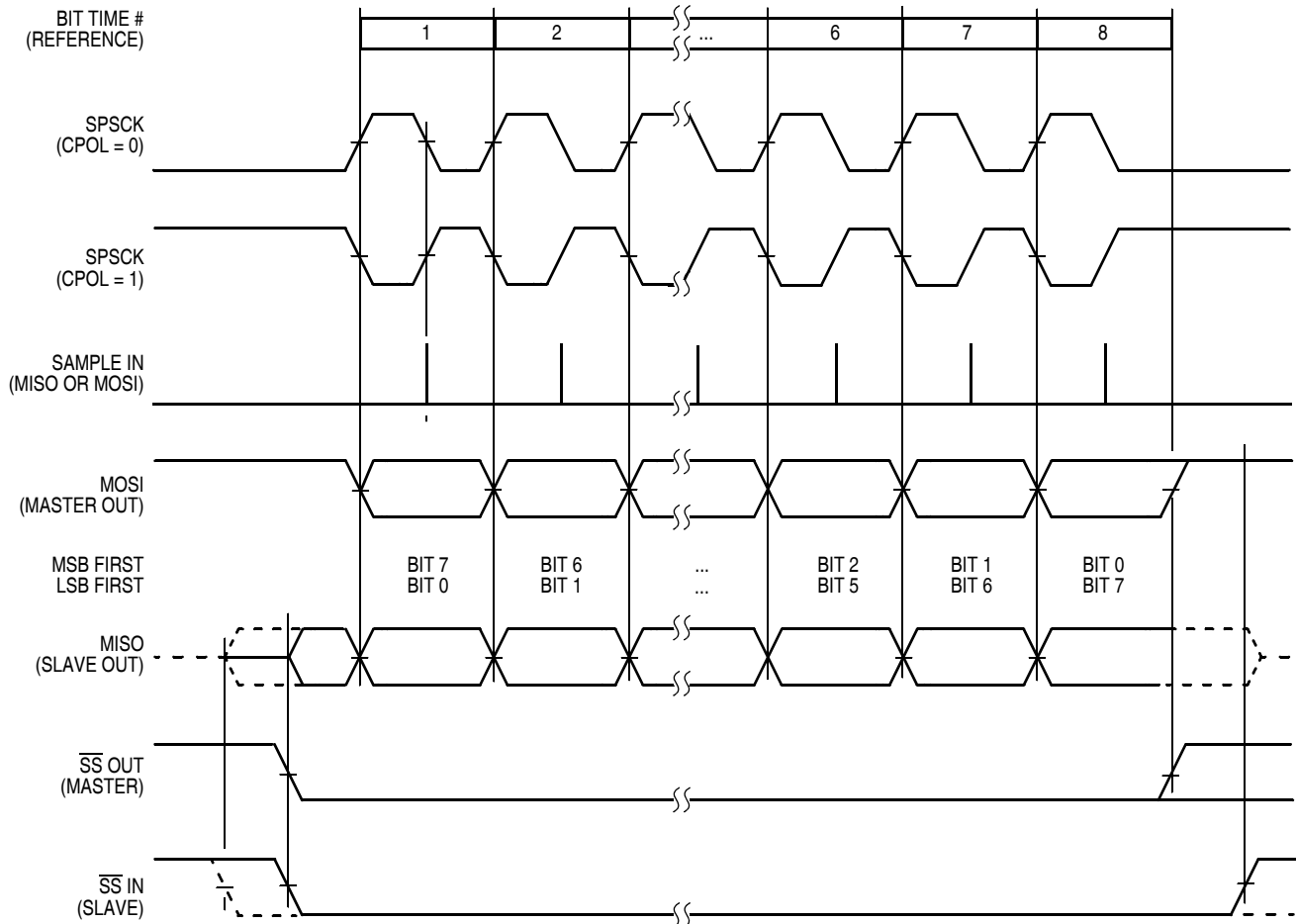select input of a slave.

Figure 18-18. 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.

18.4.7   SPI Baud Rate Generation

As shown in Figure 18-19, 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 (SPR3:SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128,
256 or 512 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 except those reserved combinations in Table 18-7:

$$BaudRateDivisor = (SPPR + 1) \bullet 2^{(SPR + 1)}$$

The baud rate can be calculated with the following equation:

$$Baud\ Rate = BusClock / BaudRateDivisor$$
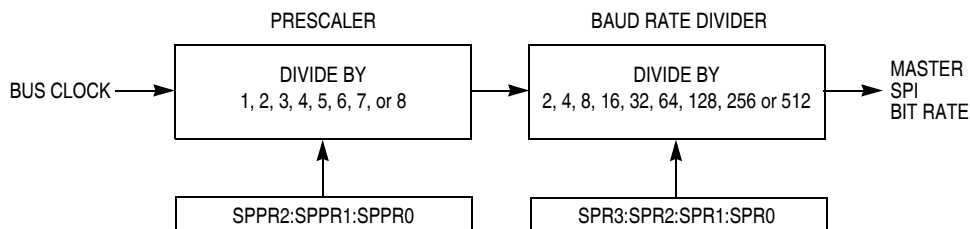
Figure 18-19. SPI Baud Rate Generation

18.4.8 Special Features

18.4.8.1 $\overline{SS}$ 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 18-2.

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.

18.4.8.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI Control Register 2 (see Section Table 18-11., "Normal Mode and Bidirectional Mode.") 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 18-11. 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.

18.4.9   Error Conditions

The SPI has one error condition:

- Mode fault error

18.4.9.1   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.

18.4.10 Low-power Mode Options

18.4.10.1 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.

18.4.10.2 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.

18.4.10.3 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.

18.4.10.4 Reset

The reset values of registers and signals are described in Section 18.3, "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.

18.4.10.5 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.

18.4.11 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).

18.4.11.1 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 18-2). 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 18.3.4, "SPI Status Register (SPIxS)."

18.4.11.2 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 18.3.4, "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.

18.4.11.3 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 18.3.4, "SPI Status Register (SPIxS).

18.4.11.4 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.

18.4.11.5 TNEAREF

TNEAREF flag is set when only one 16bit words or 2 8bit bytes of data remain in the transmit FIFO provided SPIxC3[5] = 0 or when only two 16bit words or 4 8bit bytes of data remain in the transmit FIFO provided SPIxC3[5] =1. If FIFOMODE is not enabled this bit should be ignored.

Clearing of this interrupts depends on state of SPIxC3[3] and the status of TNEAREF as described Section 18.3.4, "SPI Status Register (SPIxS)

18.4.11.6 RNFULLF

RNFULLF is set when more than three16bit words or six 8bit bytes of data remain in the receive FIFO provided SPIxC3[4] = 0 or when more than two 16bit words or four 8bit bytes of data remain in the receive FIFO provided SPIxC3[4] = 1.

Clearing of this interrupts depends on state of SPIxC3[3] and the status of RNFULLF as described Section 18.3.4, "SPI Status Register (SPIxS)

18.5   Initialization/Application Information

18.5.1   SPI Module Initialization Example

18.5.1.1   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.

18.5.1.2   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	= 0000	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	FIFOMODE is not enabled

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.

```
                        ┌──────────────┐
                        │    RESET     │
                        └──────┬───────┘
                               │
                    ┌──────────▼──────────┐
                    │   INITIALIZE SPI    │
                    │   SPIxC1 = 0x54     │
                    │   SPIxC2 = 0xC0     │
                    │   SPIxBR = 0x00     │
                    │   SPIxMH = 0xXX     │
                    └──────────┬──────────┘
                               │◄──────────────────┐
                    YES        ▼                    │
                    ◄─────────╱ ╲─────── NO         │
                              ╲ ╱  SPTEF = 1        │
                               ?                    │
                    YES        ▼                    │
                    ┌──────────▼──────────┐         │
                    │     WRITE TO        │         │
                    │  SPIxDH:SPIxDL      │         │
                    └──────────┬──────────┘         │
                               │◄──────┐            │
                              ╱ ╲──── NO            │
                              ╲ ╱  SPRF = 1         │
                               ?                    │
                    YES        ▼                    │
                    ┌──────────▼──────────┐         │
                    │       READ          │         │
                    │   SPIxDH:SPIxDL     │         │
                    └──────────┬──────────┘         │
                               │                    │
                              ╱ ╲──── NO ───────────┘
                              ╲ ╱  SPMF = 1
                               ?
                    YES        ▼
                ┌──────────────▼──────────────┐
                │  READ SPMF WHILE SET        │
                │    TO CLEAR FLAG,           │
                │  THEN WRITE A 1 TO IT       │
                └──────────────┬──────────────┘
                        ┌──────▼───────┐
                        │   CONTINUE   │
                        └──────────────┘
```
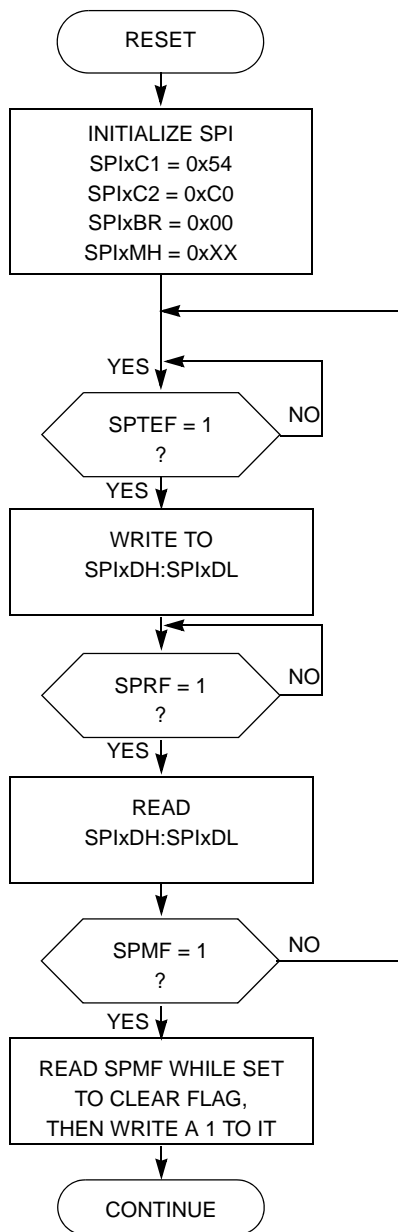
Figure 18-20. Initialization Flowchart Example for SPI Master Device in 16-bit Mode for FIFOMODE = 0
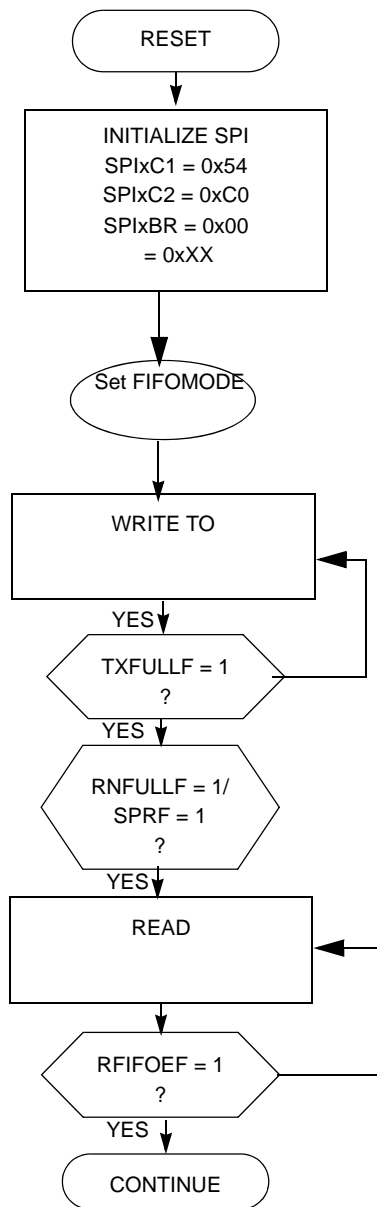
Figure 18-21. Initialization Flowchart Example for SPI Master Device in 16-bit Mode for FIFOMODE=1

Chapter 19
8-bit Serial Peripheral Interface 2 (S08SPI8V5)

19.1   Introduction

Figure 21-1 shows the MC9S08JE128 series block diagram with the SPI2 highlighted.

NOTE

There are two SPI modules on this device. Replace SPIx with the appropriate peripheral designation (SPI1 or SPI2), depending upon your use:

- SPI1 — for the 16-bit SPI with FIFO module.
- SPI2 — for the 8-bit SPI module.

19.1.1   SPI2 Clock Gating

The bus clock to the SPI2 can be gated on and off using the SPI2 bit in SCGC2. These bits are set after any reset, which enables the bus clock to this module. To conserve power, these bits can be cleared to disable the clock to this module when not in use. See Section 5.7.9, "System Clock Gating Control 2 Register (SCGC2)," for details.
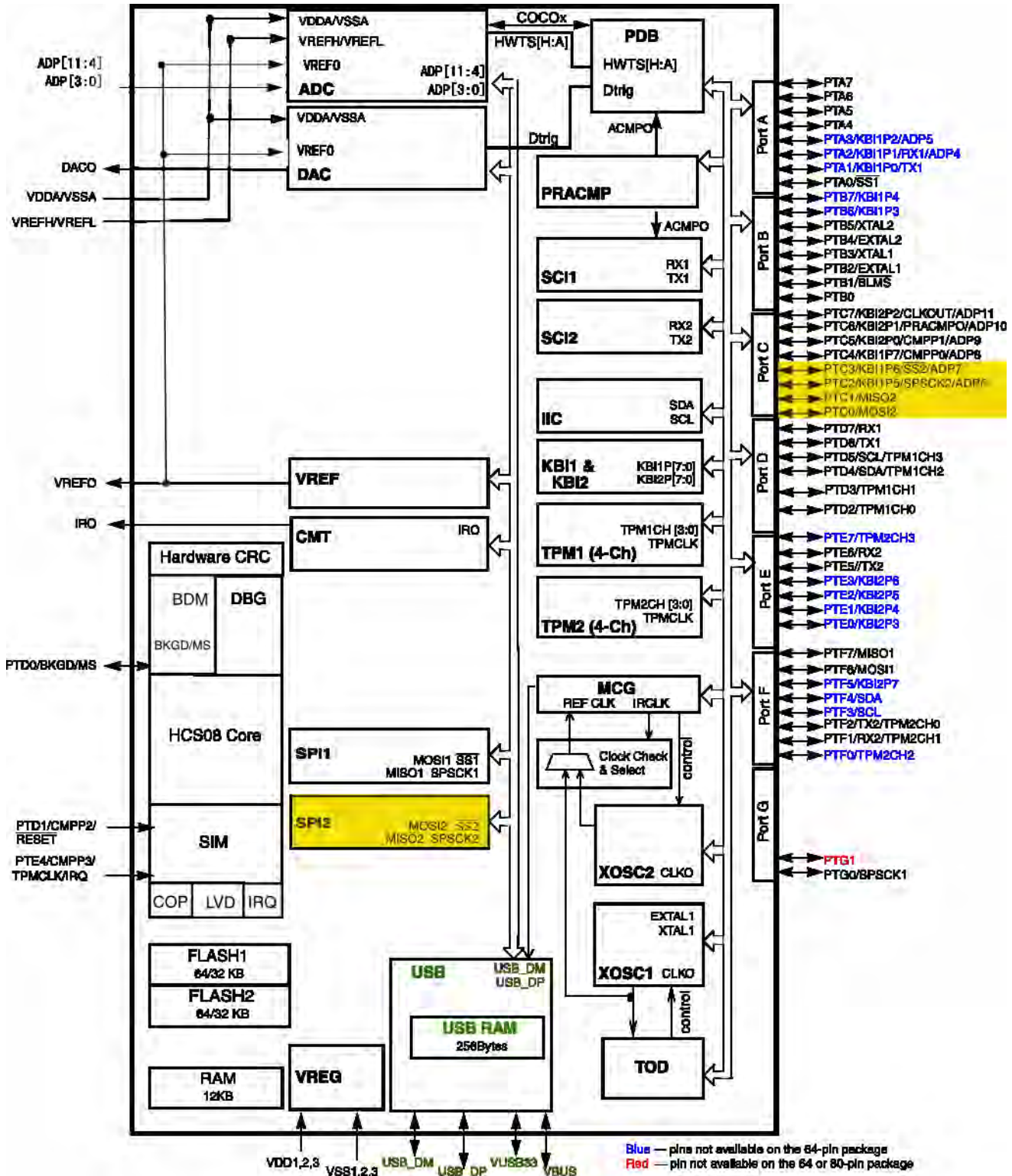
Figure 19-1. Block Diagram Highlighting SPI2 Block and Pins

19.1.2   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
- Receive data buffer hardware match feature

19.1.3   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/stop4 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 19.4.8, "Low-power Mode Options."

19.1.4   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.

19.1.4.1 SPI System Block Diagram

Figure 19-2 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.
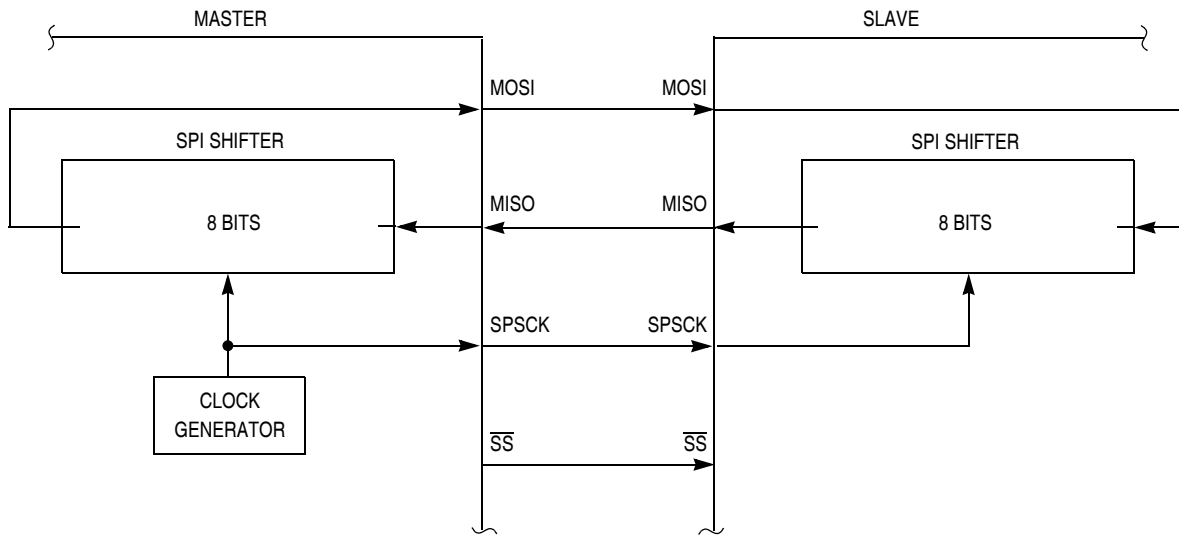

Figure 19-2. SPI System Connections

19.1.4.2 SPI Module Block Diagram

Figure 19-3 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 SPIxD) 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 SPIxD). 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.
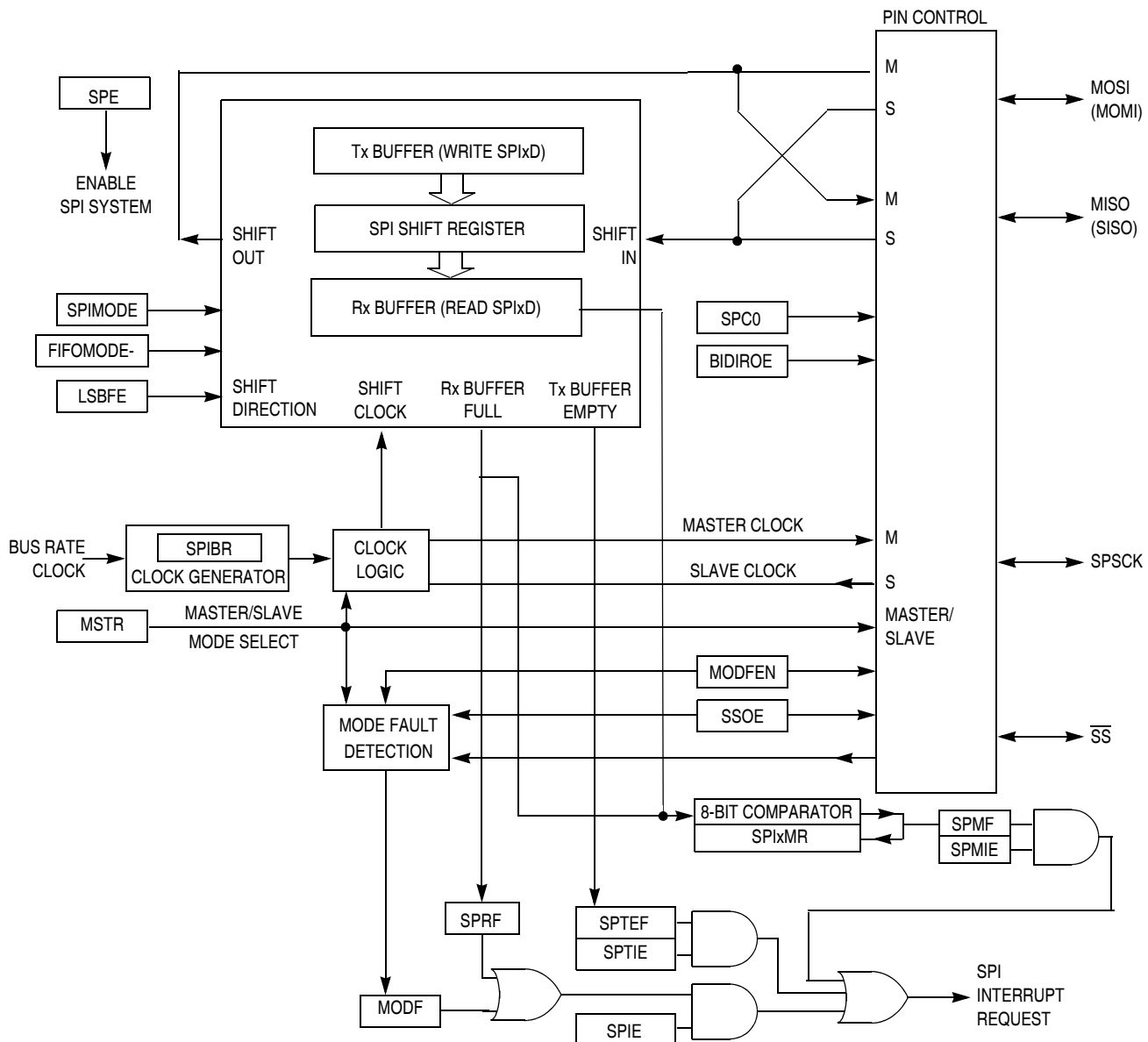
Figure 19-3. SPI Module Block Diagram

19.2   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.

19.2.1   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.

19.2.2 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.

19.2.3 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.

19.2.4 $\overline{\text{SS}}$ — 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).

19.3 Register Definition

The SPI has above 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.

19.3.1 SPI Control Register 1 (SPIxC1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.
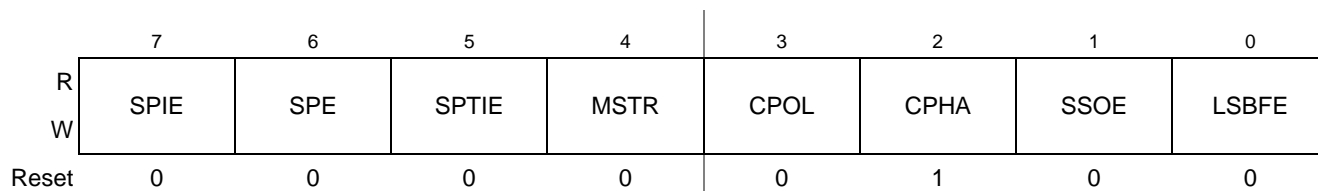
	7	6	5	4	3	2	1	0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset	0	0	0	0	0	1	0	0

Figure 19-4. SPI Control Register 1 (SPIxC1)

Table 19-1. SPIxC1 Field Descriptions

Field	Description
7 SPIE	**FIFOMODE=0** **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 **FIFOMODE=1** **SPI Read FIFO Full Interrupt Enable** — This bit when set enables the SPI to interrupt the CPU when the Receive FIFO is full. An interrupt will occur when SPRF flag is set or MODF is set. 0 Read FIFO Full Interrupts are disabled 1 Read FIFO Full Interrupts are enabled
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** — **FIFOMODE=0** This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). An interrupt occurs when the SPI transmit buffer is empty (SPTEF is set) **FIFOMODE=1** This is the interrupt enable bit for SPI transmit FIFO empty (SPTEF). An interrupt occurs when the SPI transmit FIFO is empty (SPTEF is set) 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 19.4.4, "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 19.4.4, "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 19-2.
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 19-2. $\overline{SS}$ 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

19.3.2   SPI Control Register 2 (SPIxC2)

This read/write register is used to control optional features of the SPI system. Bits 6, 5 and 2 are not implemented and always read 0.
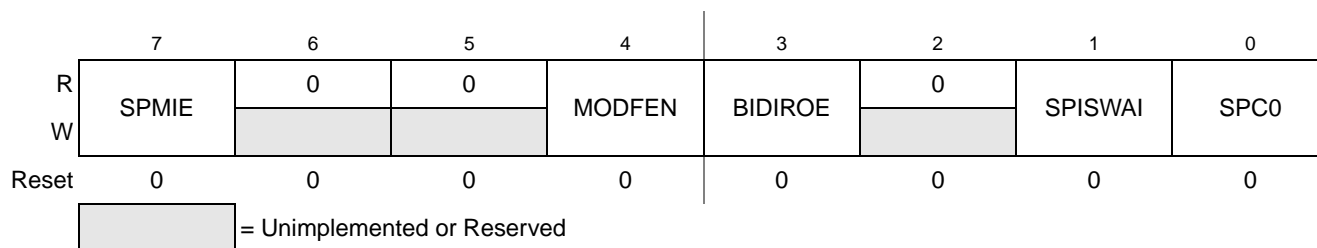
	7	6	5	4	3	2	1	0
R	SPMIE	0	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
W								
Reset	0	0	0	0	0	0	0	0

☐ = Unimplemented or Reserved

Figure 19-5. SPI Control Register 2 (SPIxC2)

Table 19-3. SPIxC2 Register Field Descriptions

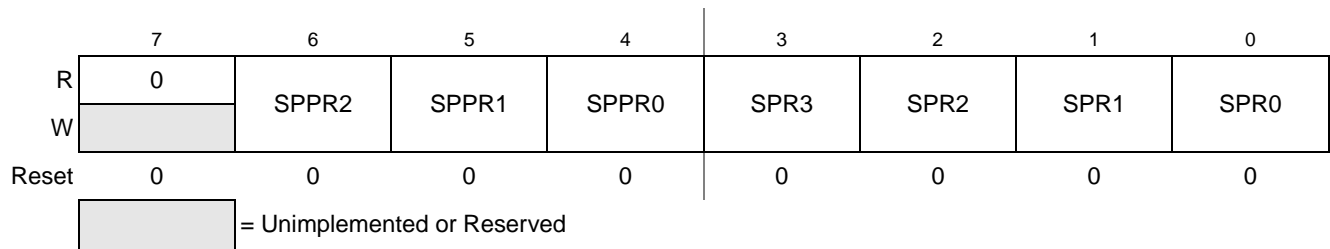| Field	Description
7 SPMIE	**SPI Match Interrupt Enable** — This is the interrupt enable for the SPI receive data buffer hardware match (SPMF) function. 0 Interrupts from SPMF inhibited (use polling). 1 When SPMF = 1, requests a hardware interrupt.
4 MODFEN	**Master Mode-Fault Function Enable** — When 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 19-2 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 19-4. 0 SPI uses separate pins for data input and data output. 1 SPI configured for single-wire bidirectional operation.

Table 19-4. 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	

19.3.3   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	SPR3	SPR2	SPR1	SPR0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 19-6. SPI Baud Rate Register (SPIxBR)

Table 19-5. 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 19-6. 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 19-12). See Section 19.4.5, "SPI Baud Rate Generation," for details.
3:0 SPR[3:0]	**SPI Baud Rate Divisor** — This 4-bit field selects one of nine divisors for the SPI baud rate divider as shown in Table 19-7. The input to this divider comes from the SPI baud rate prescaler (see Figure 19-12). See Section 19.4.5, "SPI Baud Rate Generation," for details.

Table 19-6. 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 19-7. SPI Baud Rate Divisor

SPR3:SPR2:SPR1:SPR0	Rate Divisor
0:0:0:0	2
0:0:0:1	4
0:0:1:0	8
0:0:1:1	16
0:1:0:0	32
0:1:0:1	64
0:1:1:0	128
0:1:1:1	256
1:0:0:0	512
All other combinations	Reserved

19.3.4   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.
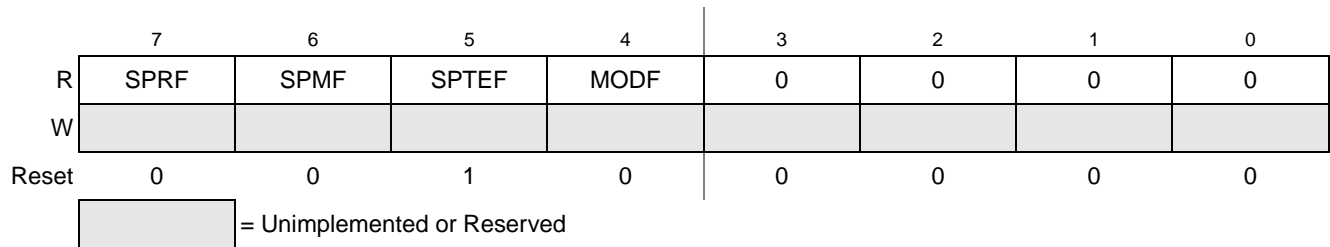
|	7	6	5	4	3	2	1	0
R	SPRF	SPMF	SPTEF	MODF	0	0	0	0
W								
Reset	0	0	1	0	0	0	0	0

= Unimplemented or Reserved

Figure 19-7. SPI Status Register (SPIxS)

Table 19-8. 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 (SPIxD). 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 SPIxMR. 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 SPIxMR registers. 1 Value in the receive data buffer matches the value in SPIxMR 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 SPIxD. SPIxS must be read with SPTEF = 1 before writing data to SPIxD or the SPIxD 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 SPIxD 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

19.3.5 SPI Data Register
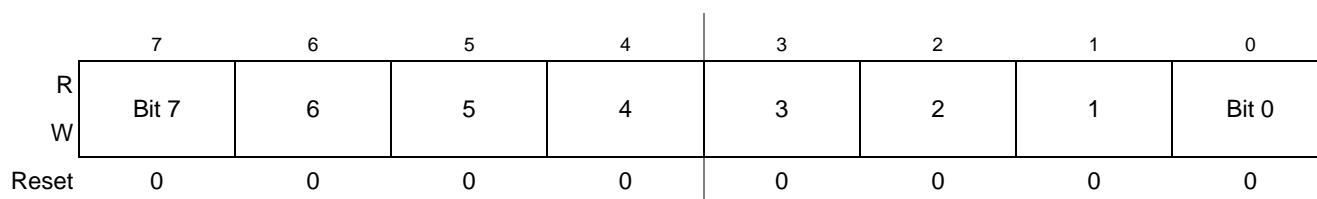
Figure 19-8. SPI Data Register(SPIxD)

The SPI data register is both the input and output register for SPI data. A write to this register 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 register, or the write will be ignored.

Data may be read from SPIxD 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.

19.3.6   SPI Match Register

This register contains 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 SPIxMR register.
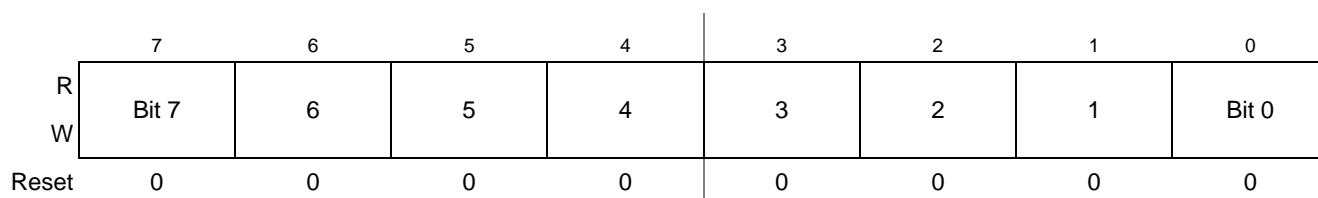
	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W								
Reset	0	0	0	0	0	0	0	0

Figure 19-9. SPI Match Register(SPIxMR)

19.4   Functional Description

19.4.1   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 SPIxD). When a transfer is complete, received data is moved into the receive data buffer. The SPIxD register acts 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.

19.4.2   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 SPR3, 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 19.4.4, "SPI Clock Formats).

NOTE

> A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, BIDIROE with SPC0 set, SPPR2-SPPR0 and SPR3-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.

19.4.3   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 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 setin slave mode will corrupt a transmission in progress and has to be avoided.

19.4.4　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 19-10 shows the clock formats when 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 eighth 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 SS OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master 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 SS IN waveform applies to the slave select input of a slave.

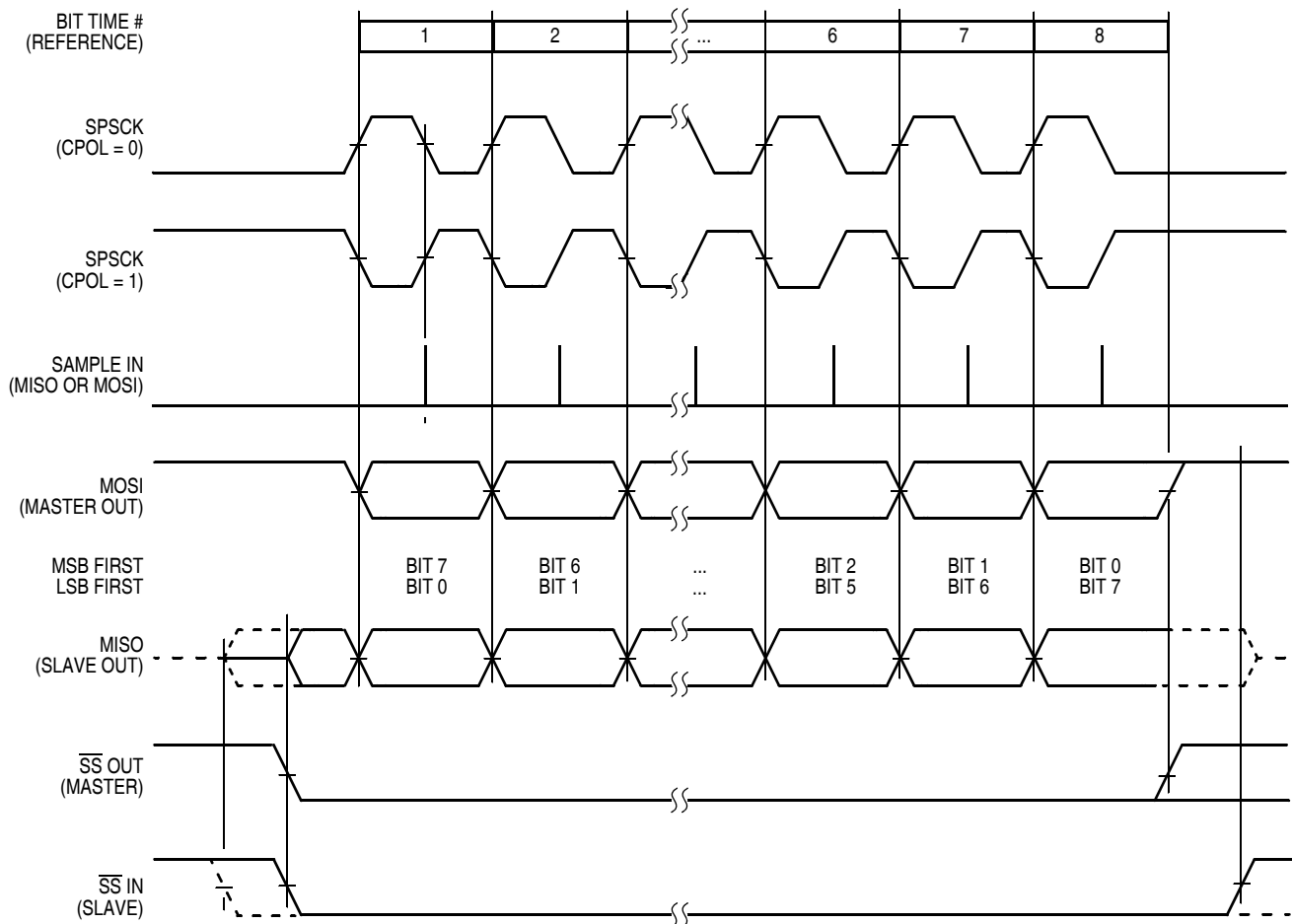**Figure 19-10. SPI Clock Formats (CPHA = 1)**

When CPHA = 1, the slave begins to drive its MISO output when 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 19-11 shows the clock formats when 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 19-11. 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 $\overline{\text{SS}}$ input must go to its inactive high level between transfers.

19.4.5 SPI Baud Rate Generation

As shown in Figure 19-12, 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 (SPR3:SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, 256 or 512 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 except those reserved combinations in Table 19-7:

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \bullet 2^{(\text{SPR} + 1)}$$

The baud rate can be calculated with the following equation:

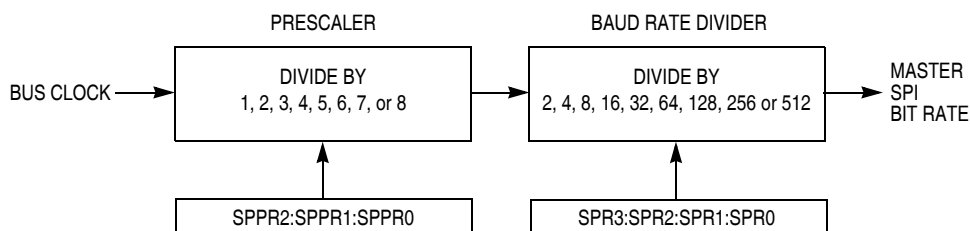$$\text{Baud Rate} = \text{BusClock} / \text{BaudRateDivisor}$$

Figure 19-12. SPI Baud Rate Generation

19.4.6 Special Features

19.4.6.1 $\overline{\text{SS}}$ Output

The $\overline{\text{SS}}$ output feature automatically drives the $\overline{\text{SS}}$ pin low during transmission to select external devices and drives it high during idle to deselect external devices. When $\overline{\text{SS}}$ output is selected, the $\overline{\text{SS}}$ output pin is connected to the $\overline{\text{SS}}$ input pin of the external device.

The $\overline{\text{SS}}$ output is available only in master mode during normal SPI operation by asserting the SSOE and MODFEN bits as shown in Table 19-2.

The mode fault feature is disabled while $\overline{\text{SS}}$ output is enabled.

NOTE

Care must be taken when using the $\overline{\text{SS}}$ output feature in a multi-master system since the mode fault feature is not available for detecting system errors between masters.

19.4.6.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI Control Register 2 (see Section Table 19-9., "Normal Mode and Bidirectional Mode.") 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 19-9. Normal Mode and Bidirectional Mode

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.

19.4.7 Error Conditions

The SPI has one error condition:

- Mode fault error

19.4.7.1 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.

19.4.8 Low-power Mode Options

19.4.8.1 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.

19.4.8.2 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 SPIxD 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 SPIxD registers until after the slave SPI has exited wait or stop mode. A SPRF flag and SPIxD 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 SPIxD copy will occur.

19.4.8.3   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.

19.4.8.4   Reset

The reset values of registers and signals are described in Section 19.3, "Register Definition." which details the registers and their bit-fields.

*   If a data transmission occurs in slave mode after reset without a write to SPIxD, it will transmit garbage, or the data last received from the master before the reset.
*   Reading from the SPIxD after reset will always read zeros.

19.4.8.5   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.

19.4.9 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).

19.4.9.1 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 19-2). 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 19.3.4, "SPI Status Register (SPIxS)."

19.4.9.2 SPRF

SPRF occurs when new data has been received and copied to the SPI receive data buffer. Once SPRF is set, it does not clear until it is serviced. SPRF has an automatic clearing process which is described in Section 19.3.4, "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 SPIxD.

19.4.9.3 SPTEF

SPTEF occurs when the SPI transmit buffer is ready to accept new data. Once SPTEF is set, it does not clear until it is serviced. SPTEF has an automatic clearing process which is described in Section 19.3.4, "SPI Status Register (SPIxS).

19.4.9.4 SPMF

SPMF occurs when the data in the receive data buffer is equal to the data in the SPI match register.

19.5   Initialization/Application Information

19.5.1   SPI Module Initialization Example

19.5.1.1   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 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 (SPIxMR) 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 (SPIxD) to begin transfer.

19.5.1.2   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 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 = 0x80(%10000000)

Bit 7	SPMIE	= 1	SPI hardware match interrupt enabled
Bit 6		= 0	Unimplemented
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		= 0000	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 SPIMR = 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

SPIxMR = 0xXX

Holds bits 0–7 of the hardware match buffer.

SPIxD = 0xxx

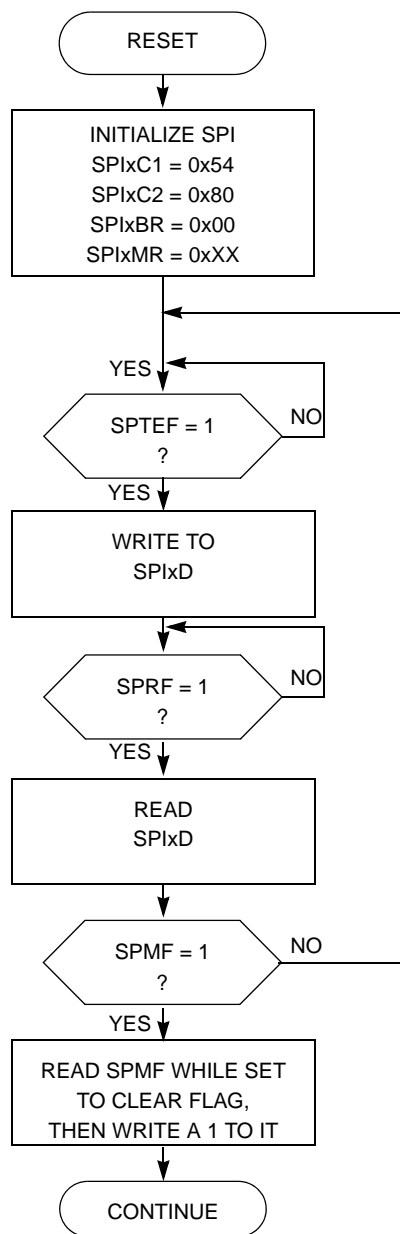Holds bits 0–7 of the data to be transmitted by the transmit buffer and received by the receive buffer.

Figure 19-13. Initialization Flowchart Example for SPI Master Device

Chapter 20
Time Of Day Module (S08TODV1)

20.1 Introduction

The time of day module (TOD) consists of one 8-bit counter, one 6-bit match register, several binary-based and decimal-based prescaler dividers, three clock source options, and one interrupt that can be used for quarter second, one second and match conditions. 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.

20.1.1 TOD Clock Sources

The TOD module on MC9S08JE128 series can be clocked from the MCGIRCLK, XOSC1 or the LPO.

20.1.2 TOD Modes of Operation

All clock sources are available in all modes except stop2. The XOSC1 and LPO can be enabled as the clock source of the TOD in stop2.

20.1.3 TOD Status after Stop2 Wakeup

The registers associated with the TOD will be unaffected after a stop2 wakeup.

20.1.4 TOD Clock Gating

The bus clock to the TOD can be gated on and off using the TOD bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the TOD bit can be cleared to disable the clock to this module when not in use. See Section 5.7.9, "System Clock Gating Control 2 Register (SCGC2)," for details.
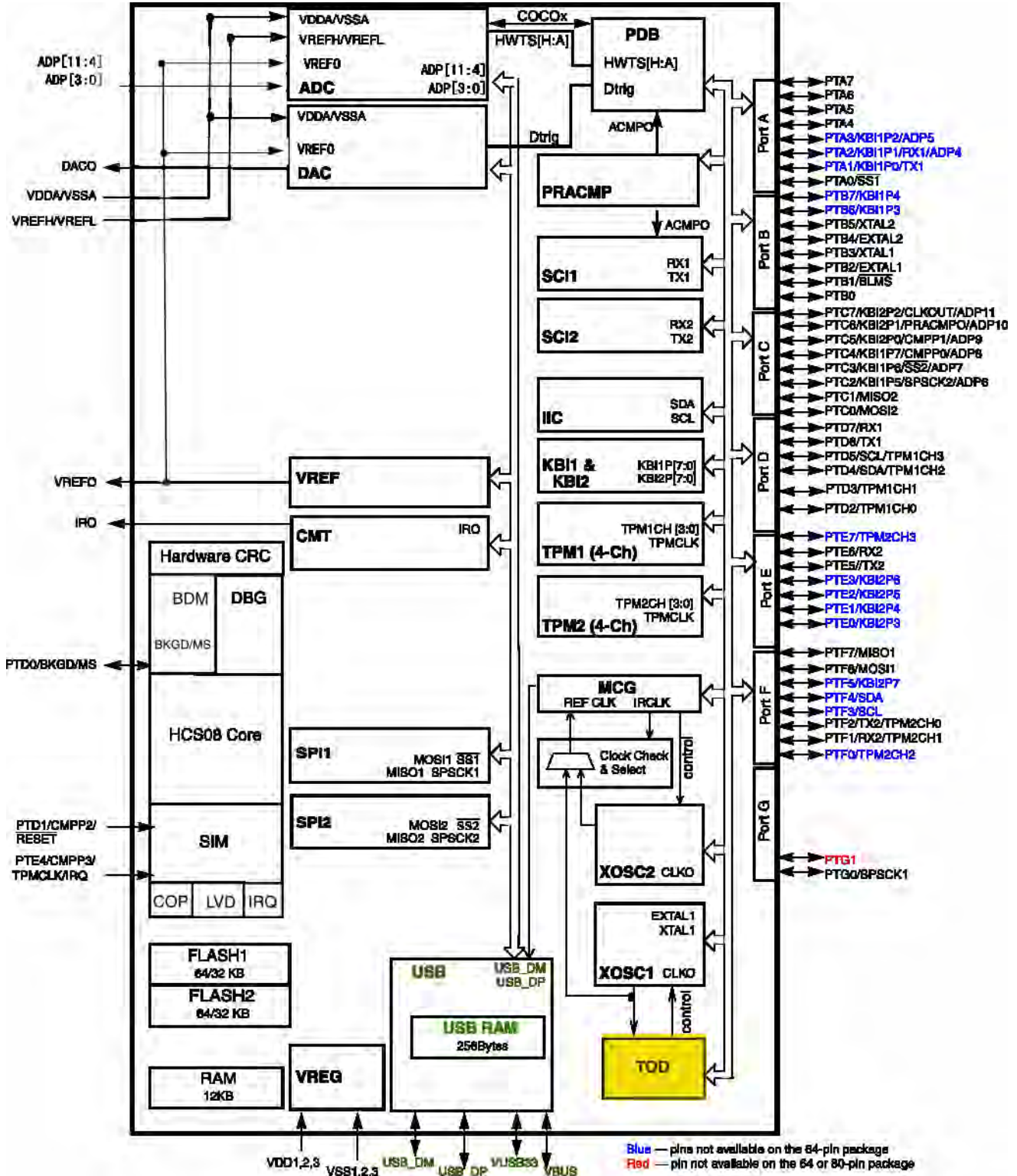
Figure 20-1. MC9S08JE128 Block Diagram Highlighting TOD Block

20.1.5   Features

Time of day module features include:

- 8-bit counter register that increments every 0.25 seconds
- Prescaler that handles standard input frequencies
- Configurable interrupts
  — Quarter second
  — One second
  — Match
- Match functionality enabled by a control bit
- Counter reset on match to facilitate time-keeping software
- Operation in low-power modes: LPrun, wait, LPwait, stop3, and stop2
- Option to use internal 1 kHz low-power oscillator (LPO)
- TOD clock output (TODCLK)

20.1.6   Modes of Operation

The TOD module supports five low-power operation modes:

Table 20-1. TOD Operation Modes

Mode	Operation
Stop2	In stop2 mode the TOD module can use an external clock source or the internal 1 KHz LPO as an input. The TOD clock input MCGIRCLK is not available in stop2 mode. In stop2 mode, all interrupt sources (quarter second, 1 second, and match) are capable of waking the MCU from stop2 mode. • If selected, the external clock source must be enabled to operate in stop mode. (EREFSTEN) • If enabled, the TODCLK signal is generated.
Stop3	In stop3 mode the TOD module can use an external clock source, the internal 1 kHz LPO or MCGIRCLK as an input. In stop3 mode all interrupt sources (quarter second, 1 second, and match) are capable of waking the MCU from stop3 mode. • If selected, the external clock source must be enabled to operate in stop mode. (EREFSTEN) • If selected, the internal clock source must be enabled to operate in stop mode. (IREFSTEN) • If enabled, the TODCLK signal is generated.
LPWait	In low-power wait mode the TOD reference clock continues to run. All interrupt sources (quarter second, 1 second, and match) are capable of waking the MCU from low-power wait mode. If enabled, the TODCLK signal is generated.
Wait	In wait mode the TOD reference clock continues to run. All interrupt sources (quarter second, 1 second, and match) are capable of waking the MCU from wait mode. If enabled, the TODCLK signal is generated.
LPRun	In low-power run mode the TOD reference clock continues to run. All interrupt sources (quarter second, 1 second, and match) are capable of waking the MCU from low-power run mode. If enabled, the TODCLK signal is generated.

20.1.7 Block Diagram
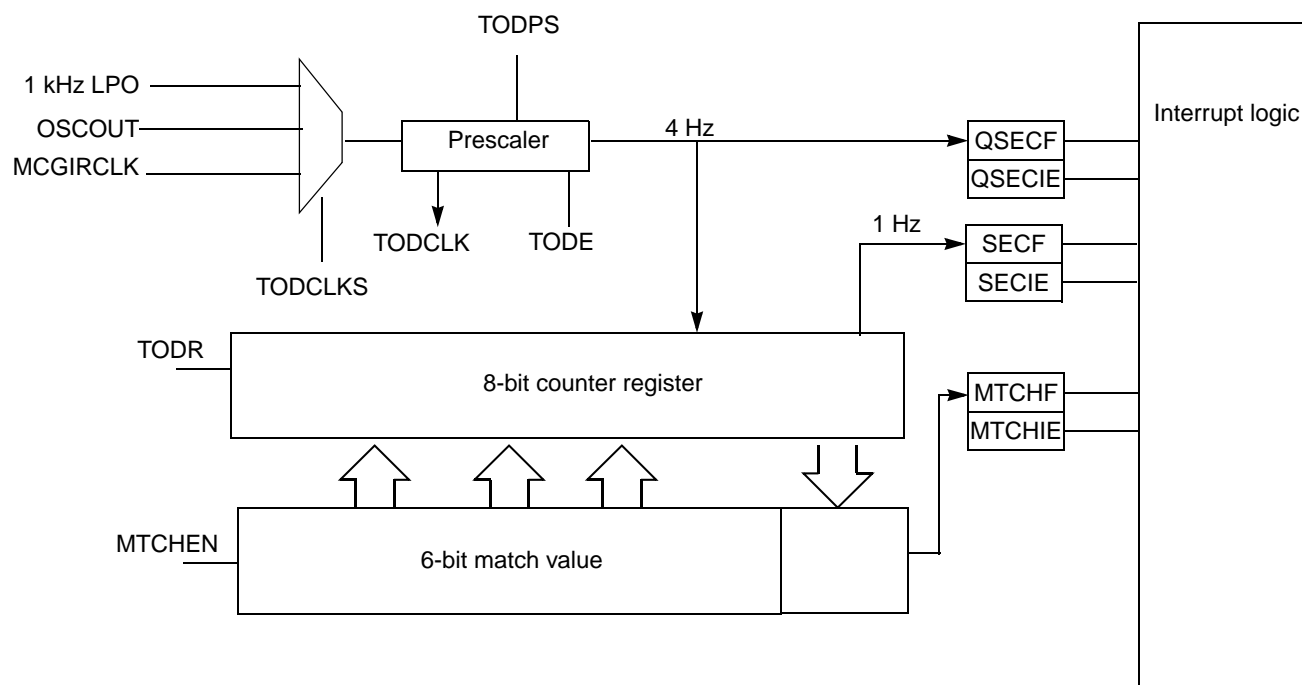
Figure 20-2 is a block diagram of the TOD module.

Figure 20-2. TOD Block Diagram

20.2 External Signal Description

The TOD module can output TODCLK. This clock can be used by other modules.

Table 20-2. Signal Properties

Name	Function
TODCLK	Clock output to be used by other modules.
TODMTCHS	TOD match signal

20.2.1 TOD Clock (TODCLK)

A clock can be provided for other modules. The TODPS bits are used to create the TOD clock. The TODCLKEN bit in the TODC register can enable or disable the TOD clock. If enabled, the TODCLK signal will be generated in low power modes.

20.2.2 TOD Match Signal (TODMTCHS)

The TOD module has output (TODMTCHS) that is set when the TOD match condition occurs. This output does not depend the value on the MTCHIE bit. For example, this output can be used to start an ADC conversion (ADC hardware trigger). This output is cleared automatically.

20.3 Register Descriptions

This section consists of register descriptions. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order.

20.3.1 TOD Control Register (TODC)

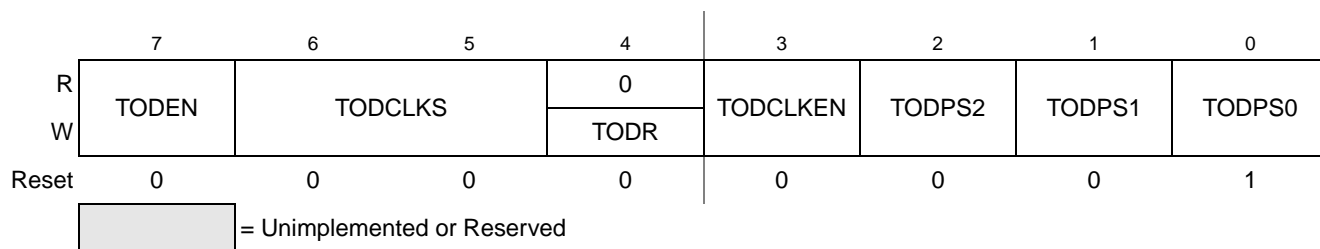|	7	6	5	4	3	2	1	0
R	TODEN	TODCLKS		0	TODCLKEN	TODPS2	TODPS1	TODPS0
W				TODR				
Reset	0	0	0	0	0	0	0	1

☐ = Unimplemented or Reserved

Figure 20-3. TOD Control Register (TODC)

Read: anytime

Write:TODEN, TODR anytime.

Table 20-3. TODC Field Descriptions

Field	Description
7 TODEN	**Time of Day Enable—** The TODEN bit enables the TOD module. 0 TOD module disabled and the TODCNT is reset to $00 and the 4 Hz generator is cleared. 1 TOD module enabled.
6:5 TODCLKS	**TOD Clock Source—** The TOD module has three possible clock sources. This bit field is used to select which clock source is the basis for the TOD. Do not change TODCLKS if TODEN = 1. 00 Selects the OSCOUT clock as the TOD clock source 01 1 kHz LPO as the TOD clock source 10 Selects the internal reference clock as the TOD clock source 11 Reserved
4 TODR	**TOD Reset—** Reset the TOD counter to $00 and the 4 Hz generator is cleared. When this bit is set, all TOD counts are cleared, allowing counting to begin at exactly $00. This bit always reads as 0. 0 No operation. 1 TOD counter register reset to $00.
3 TODCLKEN	**TOD Clock Enable—** The TODCLKEN bit enables the TOD clock, which can be used by other MCU peripherals. 0 TOD clock output disabled. 1 TOD clock output enabled.
2:0 TODPS[2:0]	**TOD Prescaler Bits—** The TOD prescaler bits are used to divide TOD reference clocks to create a 4 Hz timebase. Do not change TODPS if TODEN = 1. 000 1kHz prescaler (Use for 1 kHz LPO) 001 Use for 32.768 kHz (TODCLK = 32.768kHz) 010 Use for 32 kHz (TODCLK = 32kHz) 011 Use for 38.4 kHz (TODCLK = 38.4kHz) 100 Use for 4.9152 MHz (TODCLK = 38.4 kHz) 101 Use for 4 MHz (TODCLK = 32kHz) 110 Use for 8 MHz (TODCLK = 32kHz) 111 Use for 16 MHz (TODCLK = 32kHz)

20.3.2 TOD Status and Control Register (TODSC)

	7	6	5	4	3	2	1	0
R	QSECF	SECF	MTCHF	QSECIE	SECIE	MTCHIE	MTCHEN	MTCHWC
W								
Reset	0	0	0	0	0	0	0	0

	= Unimplemented or Reserved

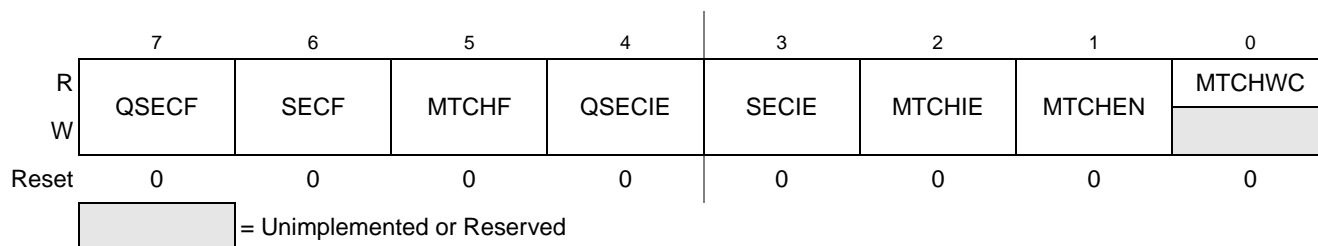**Figure 20-4. TOD Status Register (TODSC)**

Read: anytime

Write: anytime

Table 20-4. TODSC Field Descriptions

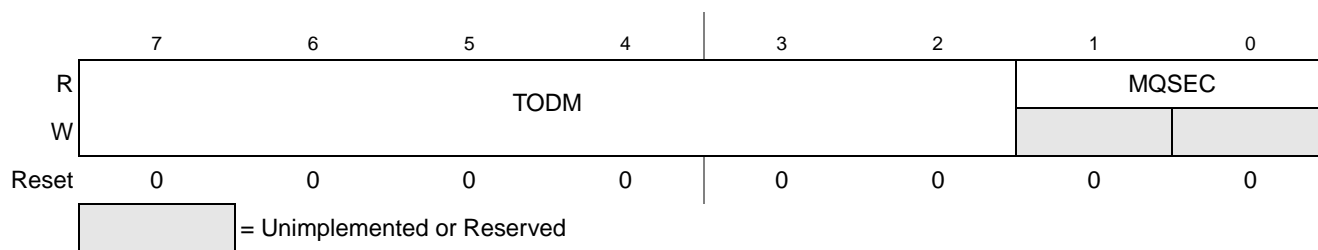| Field	Description
7 QSECF	**Quarter-Second Interrupt Flag—**QSECF indicates a quarter-second condition occurred. Write a 1 to QSECF to clear the interrupt flag . 0 Quarter -second interrupt condition has not occurred. 1 Quarter-second interrupt condition has occurred.
6 SECF	**Second Interrupt Flag—** SECF indicates a one-second condition occurred. Write a 1 to SECF to clear the interrupt flag . 0 One-second interrupt condition has not occurred. 1 One-second interrupt condition has occurred.
5 MTCHF	**Match Interrupt Flag—** MTCHF indicates the match condition occurred. Write a 1 to the MTCHF to clear the interrupt flag. 0 Match interrupt condition has not occurred. 1 Match interrupt condition has occurred.
4 QSECIE	**Quarter-Second Interrupt Enable—** Enables or disables quarter-second interrupt. The quarter-second interrupt occurs every time the TOD counter register increments. 0 Quarter-second interrupt disabled. 1 Quarter-second interrupt enabled.
3 SECIE	**Second Interrupt Enable—** Enables or disables one-second interrupt. The one-second interrupt occurs every fourth count of the TOD counter register. 0 One-second interrupt disabled. 1 One-second interrupt enabled.
2 MTCHIE	**Match Interrupt Enable—** Enables/disables the match interrupt. The match interrupt condition occurs when the match functionality is enabled (MTCHEN = 1) and the TOD counter register reaches the value in the match register 0 Match interrupt disabled. 1 Match interrupt enabled.
1 MTCHEN	**Match Function Enable—** Enables/disables the match functionality. This bit must be set for the match interrupt to occur when the TOD counter register reaches the value in the match register. When a match occurs, the upper 6 bits of the counter register are reset to 0. 0 Match function disabled. 1 Match function enabled.
0 MTCHWC	**Match Write Complete—** This bit indicates when writes to the match register have been completed. When the match register is written this bit is set, when the match register is completed this bit is cleared automatically. 0 Last write to the TOD match register is comlete. 1 Write to the TOD match register is not complete.

20.3.3 TOD Match Register (TODM)

	7	6	5	4	3	2	1	0
R				TODM			MQSEC	
W								
Reset	0	0	0	0	0	0	0	0

☐ = Unimplemented or Reserved

Figure 20-5. TOD Match Register (TODM)

Read: anytime

Table 20-5. TODM Field Descriptions

Field	Description
7:2 TODM	**TOD Match Value—** The 6 bit match value is compared against the upper 6 bits of the TOD counter register and a match occurs when the TOD counter register reaches the value in the TOD match register. The lower 2 bits of the TOD match register (MQSEC) are preloaded by hardware with the lower 2 bits of TODCNT when the match value is written. When a match occurs and MTCHEN = 1, the match Interrupt flag is asserted and the upper 6 bits of the TOD counter register are reset to 0.
1:0 MQSEC	**Match Quarter-Second Bits—**These bits are preloaded with the lower 2 bits of TODCNT when the match value is written. These bits are read only.

20.3.4 TOD Counter Register (TODCNT)

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

☐ = Unimplemented or Reserved
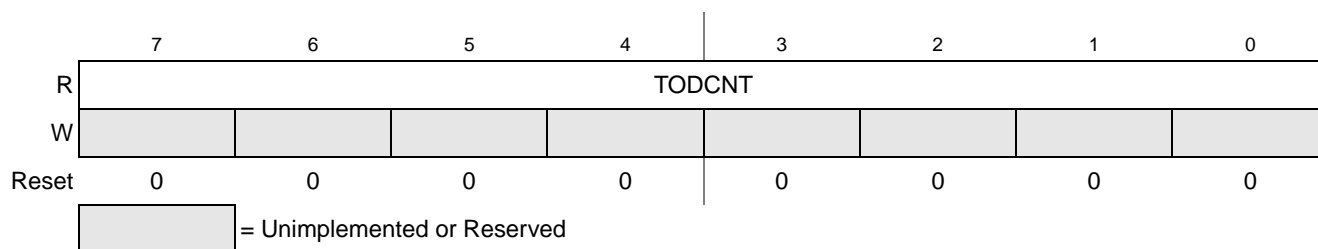
Figure 20-6. TOD Counter Register (TODCNT)

Read: anytime

Table 20-6. TODCNT Field Descriptions

Field	Description
7:0 TODCNT	**TOD Counter Register—** The TOD count register contains the current state of the TOD quarter-second counter.

20.4 Functional Description

This section provides a complete functional description of the TOD block, detailing the operation of the design from the end-user perspective.

Before enabling the TOD module by asserting the TODEN bit in the TODC register, the proper clock source and prescaler must be selected. Out of reset, the TOD module is configured with default settings, but these settings are not optimal for every application. The default settings configure TOD operation for an external 32.768 KHz oscillator. The TOD module provides a prescaler that allows several versatile configuration settings for the input clock source Refer to the TODPS bit field description in Table 20-3 to see the different input clock sources that can be used.

The TOD clock source and the TOD prescaler must be modified only if the TOD is disabled. (TODEN = 0)

Selection of the proper clock source and prescaler settings is critical because these settings are used to create the 4 Hz internal time base that is the basis for the TOD counter register. This ensures that quarter-second and one-second interrupts will occur at the appropriate times. The match interrupt can be used to generate interrupts at intervals that are multiple seconds.

Time of day can be maintained using software and the TOD module.

20.4.1   TOD Counter Register

The TOD counter is an 8-bit counter that starts at $00 and increments until $FF. After $FF is reached, the counter rolls over to $00. This creates up to 256 counts.

A 4 Hz signal is the reference clock to the TOD counter. Each tick in the TOD counter is 0.25 of a second and 4 ticks are a second. The second and quarter-second interrupts depend on proper initialization of the TOD prescaler and clock source bits.

The TODR bit in the TODC register can be used to reset the TOD counter to $00. The TODR bit also resets the 4 Hz generator ensuring that all TOD counting is starting at absolute zero time.

If the match function is enabled (MTCHEN = 1), the match flag is set when the TOD counter register reaches the value in the TOD match register and the upper 6 bits of the TOD counter register are reset to 0.

20.4.2   TOD Match Value

The TOD match value is used to generate interrupts at multiple second intervals. The TOD match value is a 6-bit value that can be set to any value from $00 to $3F. To enable match functionality, the MTCHEN bit must be set. Always configure the TODM value before setting the MTCHEN bit, this ensures that the next Match condition will occur at the desired setting.

When TODM is written,the lower 2 bits of the TODCNT are preloaded into MQSEC (the lower 2 bits of the TODM register). When TODCNT reaches the value in the TOD match register, a match condition is generated, the MTCHF is set, and the upper 6 bits of the TOD counter register are reset to 0.

For example, if the match value is set to $3C and the lower 2 bits of the TODCNT are 00, a match condition occurs when the counter transitions from $EF to $F0.

TOD Counter Register = $F0							
1	1	1	1	0	0	0	0

TOD Match Value = $3C						MQSEC	
1	1	1	1	0	0	0	0

When the match condition occurs, the upper six bits of the TOD counter register are reset to 0. The lower two bits are not affected, ensureing that one-second and quarter-second interrupt functions are not affected by a match. No time counts are lost during this time, therefore, the time interval between match conditions is exactly the same. For this case, as long as the match value is not changed, the next match condition occurs after 240 TOD counts or 1 minute. This operation facilitates timekeeping by generating a match condition after every minute.

The match value can also be used to facilitate alarm timeouts. For example, to generate an interrupt in 1 minute past the current time.

So if the TOD counter register is $50, the TOD match value must be set to $10 to generate an interrupt 1 minute past the time when the counter was $50. This can be calculated by the following algorithm.

Match value = (TOD counter register/4 + The TOD match value for 1 minute) and mask overvflow *Eqn. 20-1*

$$\text{Match value} = (\$50/4 + \$3C) \text{ and } \$3F = \$10 \qquad \textit{Eqn. 20-2}$$

When the match condition occurs, the upper six bits of the TOD counter register reset to 0. If the match register is not changed, the next match condition occurs when the upper six bits of the TOD counter matches the match value ($10). This occurs when the TOD counter reaches $40 or 64 TOD quarter-second counts. The next match condition occurs in 16 seconds, not 1 minute.

If the TOD match value is written to a value that is below the current value of the upper six bits of the TOD counter register, the match condition does not occur until the TOD counter has rolled over and matched the TOD match value. For example, if the TOD counter is $FF and the TOD match value is written to $01 then the match interrupt occurs after eight TOD counter ticks (2 seconds). Subsequent match interrupts occur every second.

If the TOD match value is written from a value that is close to the current value of the TODCNT register to a new value, the match condition may still occur at the old TOD match value. The TOD match value will be valid if it is changed when the TODCNT is 2 TODCNT values from the current TOD match value.

If the TOD match value is written to $00 and the TODCNT is reset by the TODR bit or by disabling and enabling the TOD module, the TOD match condition will not be valid. Care should be taken when writing the TOD match value to $00 because the TOD match interrupt may not occur at the expected time interval if the TOD match value is written to $00 and the TODCNT is reset.

The MTCHIE bit is used to enable or disable an interrupt when a match condition occurs.

20.4.3 Match Write Complete

When writing the TOD Match Value it may be necessary to monitor the MTCHWC bit for the case of back-to-back writes of the TOD match value. If the second write is done while MTCHWC = 1, the write

does not occur. The MTCHWC can be used to verify that the last TOD match register write is complete. Normal TOD use does not require back-to-back writes of the TOD match register.

20.4.4 TOD Clock Select and Prescaler

The TOD module defaults to operate using a 32.768 kHz clock input. Three possible clock sources are available to the TOD module:

- OSCOUT (TODCLKS = 00)
- Internal 1kHz LPO (TODCLKS =01)
- MCGIRCLK (TODCLKS =10)

NOTE

Select the appropriate clock source for the desired accuracy. OSCOUT is as accurate as the external source, MCGIRCLK is as accurate as the MCG specifications, and the 1 kHz LPO is the least accurate clock source.

Table 20-7 shows the TOD clock tree. The clock tree shows the three possible clock sources and the TOD clock output.
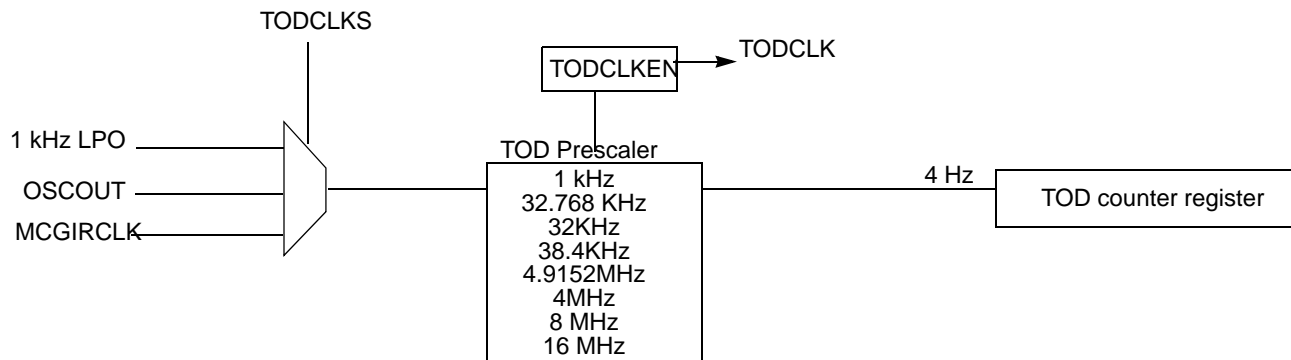
Figure 20-7. TOD Clock Tree

The TOD prescaler has dividers to generate the 4 Hz reference input to the TOD counter register. Setting the TODPS bits correctly configures internal dividers to generate the 4 Hz signal. The TOD prescaler also uses dividers to generate the TOD clock output. The TODPS must not be changed if TODEN = 1.

20.4.4.1 TOD Clock Output

The TOD module can be configured to generate a TOD clock. The TOD clock is available for use by other peripherals of the microcontroller as determined by the SOC guide. The TOD clock output is enabled with the TODCLKEN bit in the TODC register.

The TOD prescaler contains dividers for low and high frequency inputs as described in the TODPS bit description. The following table shows the value of TOD clock for various inputs.

Input	TODPS[2:0]	TOD Clock
1 kHz LPO	000	-
32.768 KHz	001	32.768 KHz
32 KHz	010	32 KHz
38.4 KHz	011	38.4 KHz
4.9152 MHz	100	38.4 kHz
4 MHz	101	32 kHz
8 MHz	110	32kHz
16 MHz	111	32 kHz

20.4.5   Quarter-Second, One-Second, and Match Interrupts

The TOD module contains quarter-second, one-second, and match interrupts to facilitate time of day applications. These interrupts have corresponding flags (QSECF, SECF, and MTCHF) that are set when the conditions are met and for the case of MTCHF the match functionality must be enabled (MTCHEN =1). These flags are set regardless if the interrupt is enabled using the coresponding interrupt enable bits. These flags must be cleared to allow the next interrupt to occur.

20.4.5.1   Quarter-Second Interrupt

The TOD counter register is a quarter-second counter. Each tick in this counter represents a quarter second. If the quarter-second interrupt is enabled (QSECIE = 1), each tick in this register results in a TOD interrupt. For example, when the TOD counter register changes from $01 to $02, the QSECF is set and the interrupt is generated. To clear the QSECF, an interrupt service routine must write a 1 to it so that the next quarter-second interrupt can occur.

20.4.5.2   One-Second Interrupt

The one-second interrupt is enabled by the SECIE bit in the TODSC register. When the one-second interrupt is enabled, an interrupt is generated every four counts of the TOD counter register. For example, when the TOD counter register changes from $03 to $04, the SECF is set and an interrupt is generated. To clear the SECF, an interrupt service routine must write a 1 to it so that the next one-second interrupt can occur.

If both the quarter-second and one-second interrupts are enabled, only one interrupt is generated when the TOD counter register changes from $03 to $04. The QSECF and the SECF are set and both must be cleared by the TOD interrupt service routine.

20.4.5.3   Match Interrupt

The TOD match functionality can be used to generate interrupts at multiple-second time intervals. This functionality must be enabled using the MTCHEN bit. If MTCHEN = 1, the TOD match interrupt occurs when the TOD counter register reaches the value that is placed in the TODM and the MTCHIE bit is set.

NOTE

The TOD match value (TODM) should always be written before enabling the match functionality (MTCHEN = 1).

To generate an interrupt after 4 seconds, the TOD match value must be written to a value that is four counts past the upper six bits of the TOD counter register. See Section 20.3.3, "TOD Match Register (TODM)," for an example. When the TOD counter register reaches the value placed in TODM, the MTCHF is set and an interrupt is generated. The upper six bits of the TOD counter are reset. An interrupt service routine must clear the MTCHF by writing a 1 to it so that the next match interrupt can occur. All interrupt flags that are set must be cleared by the TOD interrupt service routine so that the interrupt may occur again.

If the MTCHIE bit is not set when MTCHEN =1, then match conditions will occur, the MTCHF will be set, and the upper six bits of the TOD counter register will reset to 0 after the match condition has occured.

20.4.6   Resets

During a reset, the TOD module is configured in the default mode. The default mode includes the following settings:

- TODEN is cleared, TOD is disabled
- TODCLKS is cleared and TODPS = 001, defaults to 32.768 kHz external oscillator clock source
- All TOD interrupts are disabled
- MTCHEN is cleared, match condition is disabled
- TOD counter register and TOD match register are cleared

20.4.7   Interrupts

See Section 20.4.5, "Quarter-Second, One-Second, and Match Interrupts."

20.5   Initialization

This section provides a recommended initialization sequence for the TOD module and also includes initialization examples for several possible TOD application scenarios.

20.5.1   Initialization Sequence

This list provides a recommended initialization sequence for the TOD module.

1. Configure the TODC register
   a) Configure TOD clock source (TODCLKS bit)
   b) Configure the proper TOD prescaler (TODPS bits)
2. Write the TOD match register (optional)
   a) Write the TOD match value to the desired value (TODM register)
3. Enable the TODSC register(optional)
   a) Enable match functionality

b) Enable desired TOD interrupts (QSECIE, SECIE, MTCHIE bits)

4. Enable TODC register

a) (Optional) Enable TOD clock output (TODCLKEN bit)

b) Enable the TOD module (TODEN bit)

20.5.2 Initialization Examples

This section provides initialization information for configuring the TOD. Each example details the register and bit field values required to achieve the appropriate TOD configuration for a given TOD application scenario. Table 20-7 lists each example and the setup requirements.

Table 20-7. TOD Application Scenarios

Example	TOD Clock Source	Required Interrupt Timeout	TOD Clock Output
1	External 32.768 kHz	0.25 sec.	no
2	Internal 38.4 kHz	1 sec.	no
3	Internal 1 kHz LPO	approx. 1 min.	no

These examples illustrate the flexibility of the TOD module to be configured to meet a range of application requirements including:

- Clock inputs/sources
- Required interrupt timeout
- TOD clock output

20.5.2.1 Initialization Example 1

TOD setup requirements for example 1 are reiterated in Table 20-8:

Table 20-8. TOD Setup Requirements (Example 1)

Example	TOD Clock Source	Required Interrupt Timeout	TOD clock Output
1	External 32.768 kHz	0.25 sec.	no

Table 20-9 lists the required setup values required to initialize the TOD as specified in example 1:

Table 20-9. Initialization Register Values for Example 1

Register	Bit or Bit Field	Binary Value	Comment
TODC 100X1001	TODEN	1	Enable the TOD module, last step of initialization
	TODCLKS	00	Use OSCOUT for external 32.768 KHz TOD clock source
	TODR	X	TOD reset optional
	TODCLKEN	0	TOD clock output is disabled
	TODPS	001	For 32.768 KHz, TODPS is set to 001
TODSC 1XX1000x	QSECF	1	Clear the quarter-second interrupt QSECF
	SECF	X	One-second interrupt is not used
	MTCHF	X	Match interrupt is not used
	QSECIE	1	Enable quarter-second interrupts
	SECIE	0	One-second interrupt is not used
	MTCHIE	0	Match interrupt is not used
	MTCHEN	0	Match functionality is disabled
	MTCHWC	X	Match complete flag not used
TODM $XX	TODM	$XX	Match value is not used

20.5.2.2 Initialization Example 2

TOD setup requirements for example 2 are reiterated in Table 20-10:

Table 20-10. TOD Setup Requirements (Example 2)

Example	TOD Clock Source	Required Interrupt Timeout	TOD Clock Output
2	Internal 38.4 kHz	1 sec.	no

Table 20-11 lists the required setup values required to initialize the TOD as specified in example 2:

Table 20-11. Initialization Register Values for Example 2

Register	Bit or Bit Field	Binary Value	Comment
TODC 110X1010	TODEN	1	Enable the TOD module, last step of initialization
	TODCLKS	10	Use MCGIRCLK for internal 38.4 kHz TOD clock source
	TODR	X	TOD Reset optional
	TODCLKEN	0	TODCLK output is disabled
	TODPS	011	For 38.4 KHz, TODPS is set to 011
TODSC X1X0100x	QSECF	X	Quarter-second interrupt is not used
	SECF	1	Clear one-second interrupt flag
	MTCHF	X	Match interrupt is not used
	QSECIE	0	Quarter-second interrupt is not used
	SECIE	1	Enable one-second interrupt
	MTCHIE	0	Match interrupt is not used
	MTCHEN	0	Match functionality is disabled
	MTCHWC	X	Match complete flag not used
TODM $XX	TODM	$XX	Match value is not used

20.5.2.3 Initialization Example 3

TOD setup requirements for example 3 are reiterated in Table 20-12:

Table 20-12. TOD Setup Requirements (Example 3)

Example	TOD Clock Source	Required Interrupt Timeout	TOD Clock Output
3	Internal 1 kHz LPO	approx. 1 min.	no

Table 20-13 lists the required setup values required to initialize the TOD as specified in Example 3:

Table 20-13. Initialization Register Values for Example 3

Register	Bit or Bit field	Binary Value	Comment
TODC 10110000	TODEN	1	Enable the TOD module, last step of initialization
	TODCLKS	01	Use for internal 1 kHz low-power oscillator TOD clock source
	TODR	1	TOD reset to begin TOD counts at $00
	TODCLKEN	0	TODCLK output disabled
	TODPS	000	For 1 KHz low-power oscillator,TODPS is set to 000
TODSC XX10011x	QSECF	X	Quarter-second interrupt is not used
	SECF	X	One-second interrupt is not used
	MTCHF	1	Clear match interrupt flag
	QSECIE	0	Quarter-second interrupt is not used
	SECIE	0	One-second interrupt is not used
	MTCHIE	1	Match interrupt is enabled
	MTCHEN	1	Match functionality is enabled
	MTCHWC	X	Match complete flag not used
TODM 111100XX	TODM	$3C	Match value is set to $3C to generate an interrupt after $F0 or 240 counts of the TOD counter register. Always write the match value before setting the MTCHEN bit.

20.6 Application Information

The most common application for the TOD module is keeping the time of day. The TOD module can be used to keep track of the second, minute, hour, day, month, and year. To accomplish these tasks, software is used to count quarter seconds or seconds. Seconds are added up in RAM until they reach minutes, minutes are added up until they reach hours, and so forth.

Chapter 21
Timer/Pulse-Width Modulator (S08TPMV3)

21.1   Introduction

The MC9S08JE128 series of devices contains two 4-channel TPM peripherals, TPM1 and TPM2.

Figure 21-1 shows the MC9S08JE128 series block diagram with the TPMx highlighted.

21.1.1   ACMP/TPM Configuration Information

The ACMP module can be configured to connect the output of the analog comparator to the TPM1 input capture channel 0 by setting the corresponding ACIC bit in SOPT2. With ACIC set, the TPM1CH0 pin is not available externally regardless of the configuration of the TPM1 module.

21.1.2   TPM External Clock

Both the TPM modules on the MC9S08JE128 series use the TCLK pin.

21.1.3   TPM Clock Gating

The bus clock to the TPMs can be gated on and off using the TPM2 bit and TPM1 bit in SCGC1. These bits are set after any reset, which enables the bus clock to this module. To conserve power, these bits can be cleared to disable the clock to this module when not in use. See Section 5.7.8, "System Clock Gating Control 1 Register (SCGC1)," for details.
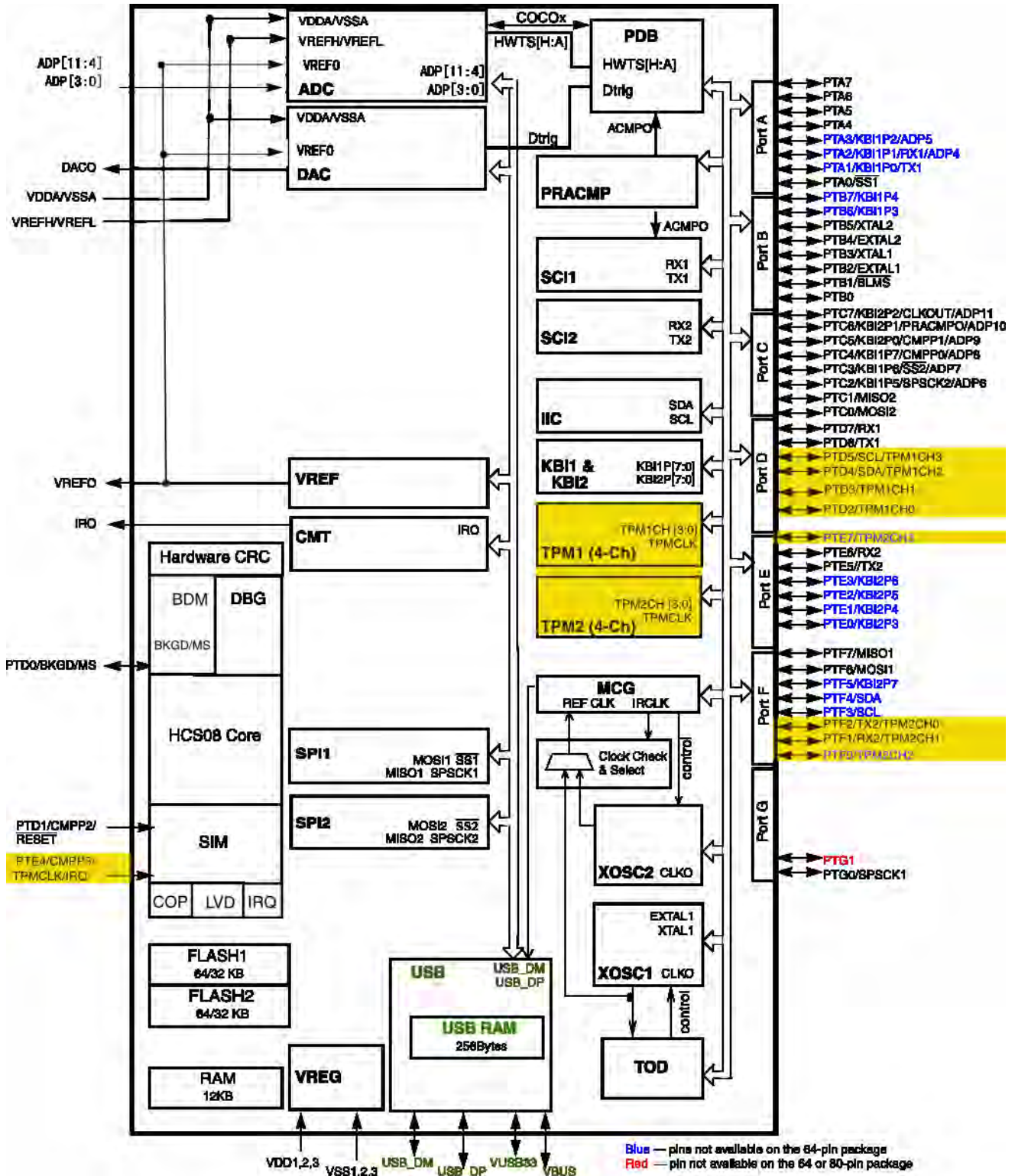
Figure 21-1. Block Diagram Highlighting TPMx Blocks

21.1.4   Features

The TPM includes these distinctive features:

- One to eight channels:
  - Each channel is 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 is configured for buffered, center-aligned pulse-width-modulation (CPWM) on all channels
- Timer clock source selectable as bus clock, fixed frequency clock, or an external clock
  - Prescale taps for divide-by 1, 2, 4, 8, 16, 32, 64, or 128 used for any clock input selection
  - Fixed frequency clock is an additional clock input to allow the selection of an on chip clock source other than bus clock
  - Selecting external clock connects TPM clock to a chip level input pin therefore allowing to synchronize the TPM counter with an off chip clock source
- 16-bit free-running or modulus count with up/down selection
- One interrupt per channel and one interrupt for TPM counter overflow

21.1.5   Modes of Operation

In general, TPM channels are 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 MCU is in active BDM background or BDM foreground mode, the TPM temporarily suspends all counting until the MCU returns to normal user operating mode. During stop mode, all TPM input clocks are stopped, so the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally. If the TPM does not need to produce a real time reference or provide the interrupt sources needed to wake the MCU from wait mode, the power can then be saved 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) are selected as the active edge that 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 is 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. You can 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 that is 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.

21.1.6   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 21-2 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.
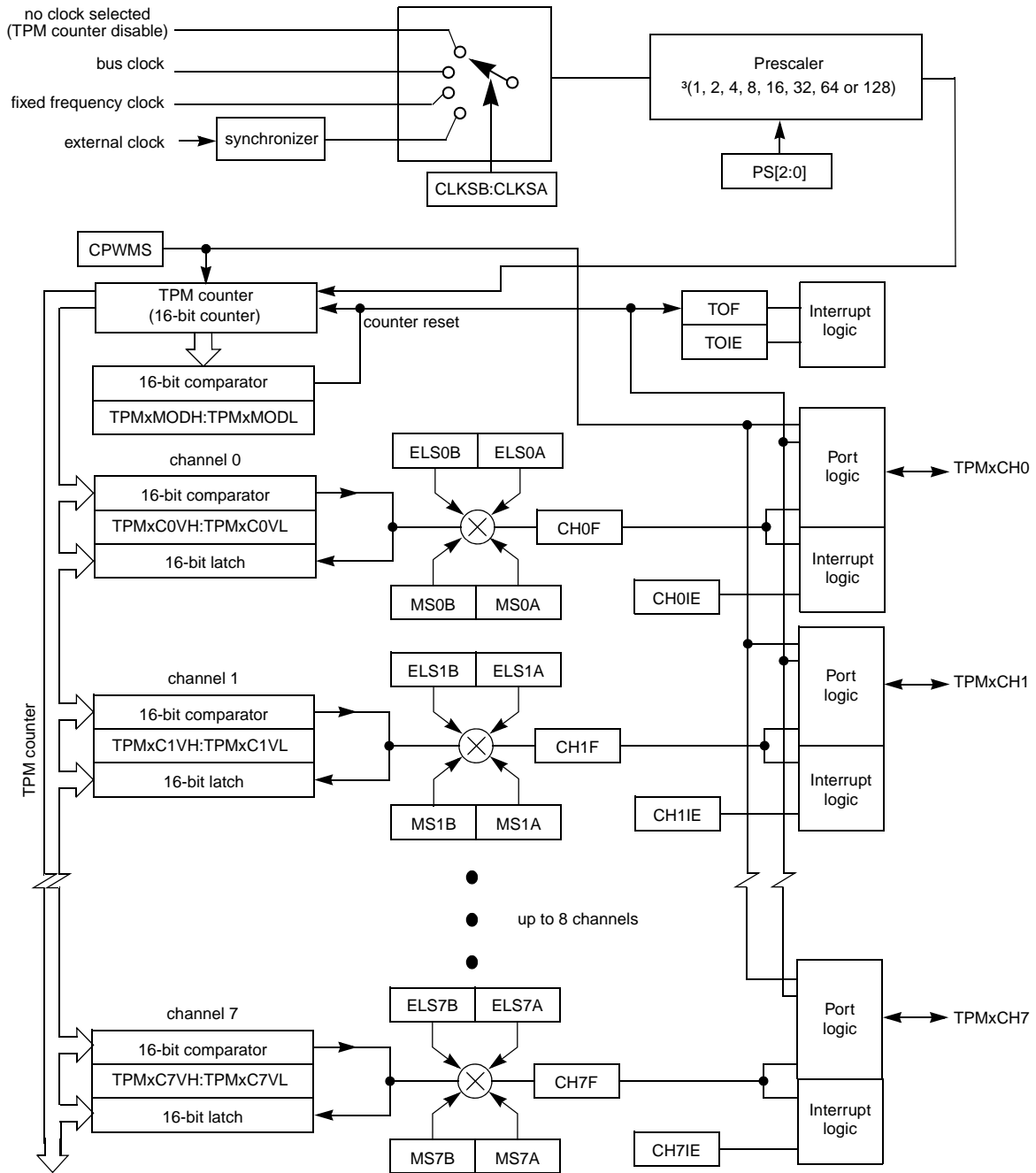
Figure 21-2. TPM Block Diagram

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.

21.2 Signal Description

Table 21-1 shows the user-accessible signals for the TPM. The number of channels are 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 21-1. Signal Properties

Name	Function
EXTCLK[1]	External clock source that is selected to drive the TPM counter.
TPMxCHn[2]	I/O pin associated with TPM channel n.

[1] The external clock pin can be shared with any channel pin. However, depending upon full-chip implementation, this signal could be connected to a separate external pin.

[2] n = channel number (1–8)

21.2.1 Detailed Signal Descriptions

21.2.1.1 EXTCLK — External Clock Source

The external clock signal can share the same pin as a channel pin, however the channel pin can not be used for channel I/O function when external clock is selected. If this pin is used as an external clock (CLKSB:CLKSA = 1:1), the channel can still be configured to output compare mode therefore allowing its use as a timer (ELSnB:ELSnA = 0:0).

For proper TPM operation, the external clock frequency must not exceed one-fourth of the bus clock frequency.

21.2.1.2 TPMxCHn — TPM Channel n I/O Pins

The TPM channel does not control the I/O pin when ELSnB:ELSnA or CLKSB:CLKSA are cleared so it normally reverts to general purpose I/O control. When CPWMS is set and ELSnB:ELSnA are not cleared, all TPM channels are configured for center-aligned PWM and the TPMxCHn pins are all controlled by TPM. When CPWMS is cleared, 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 ≠ 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 trigger input capture events. The channel input signal is synchronized on 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).

When a channel is configured for output compare (CPWMS = 0, MSnB:MSnA = 0:1, and ELSnB:ELSnA ≠ 0:0), the TPMxCHn pin is an output controlled by the TPM. The ELSnB:ELSnA bits determine whether the TPMxCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the TPM 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, the pin is then toggled.

When a channel is configured for edge-aligned PWM (CPWMS = 0, MSnB = 1, and ELSnB:ELSnA ≠ 0:0), the TPMxCHn pin is an output controlled by the TPM, and ELSnB:ELSnA bits control the polarity of the PWM output signal. When ELSnB is set and ELSnA is cleared, the TPMxCHn pin is forced high at the start of each new period (TPMxCNT=0x0000), and it is forced low when the channel value register matches the TPM counter. When ELSnA is set, the TPMxCHn pin is forced low at the start of each new period (TPMxCNT=0x0000), and it is forced high when the channel value register matches the TPM counter.

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005

Figure 21-3. High-true pulse of an edge-aligned PWM

TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005
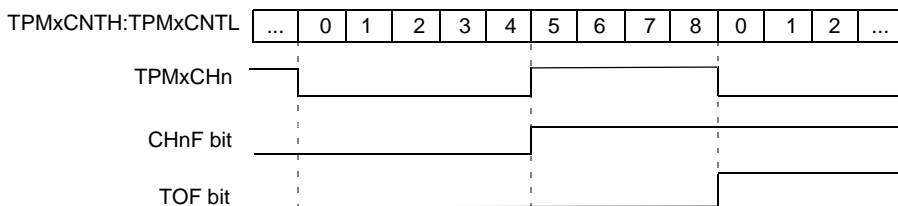
Figure 21-4. Low-true pulse of an edge-aligned PWM

When the TPM is configured for center-aligned PWM (CPWMS = 1 and ELSnB:ELSnA ≠ 0:0), the TPMxCHn pins are outputs controlled by the TPM, and ELSnB:ELSnA bits control the polarity of the PWM output signal. If ELSnB is set and ELSnA is cleared, the corresponding TPMxCHn pin is cleared when the TPM counter is counting up, and the channel value register matches the TPM counter; and it is

set when the TPM counter is counting down, and the channel value register matches the TPM counter. If ELSnA is set, the corresponding TPMxCHn pin is set when the TPM counter is counting up and the channel value register matches the TPM counter; and it is cleared when the TPM counter is counting down and the channel value register matches the TPM counter.

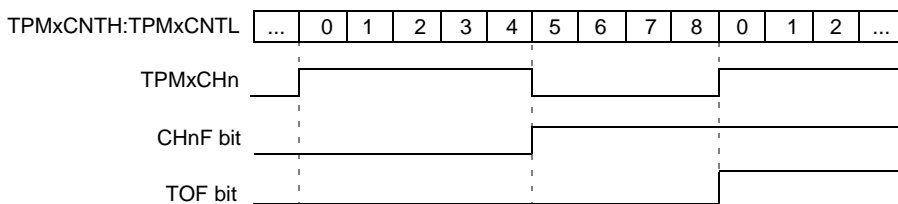TPMxMODH:TPMxMODL = 0x0008
TPMxCnVH:TPMxCnVL = 0x0005

Figure 21-5. High-true pulse of a center-aligned PWM

TPMxMODH:TPMxMODL = 0x0008
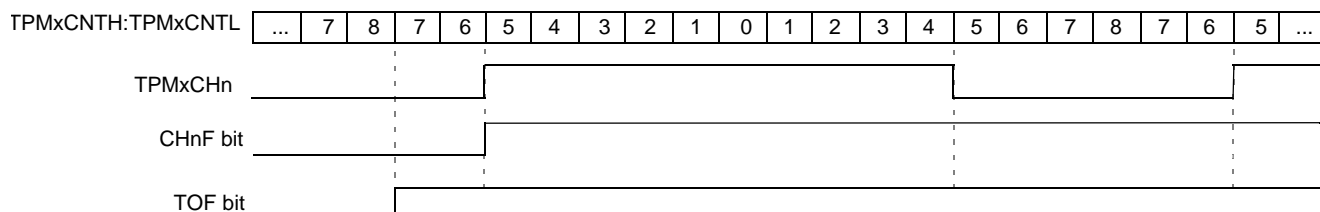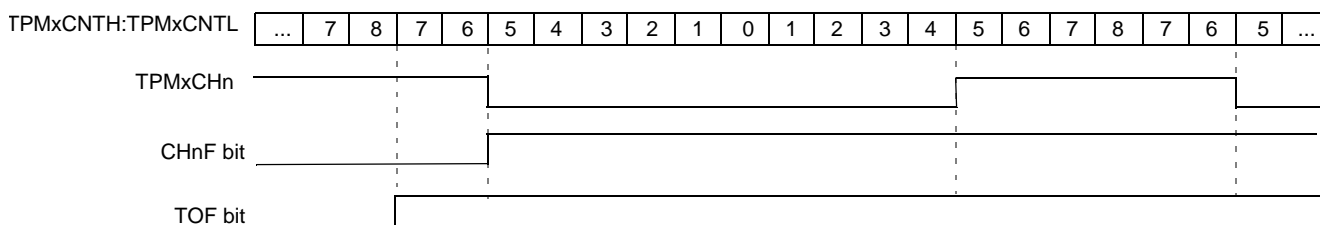TPMxCnVH:TPMxCnVL = 0x0005

Figure 21-6. Low-true pulse of a center-aligned PWM

21.3   Register Definition

21.3.1   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.
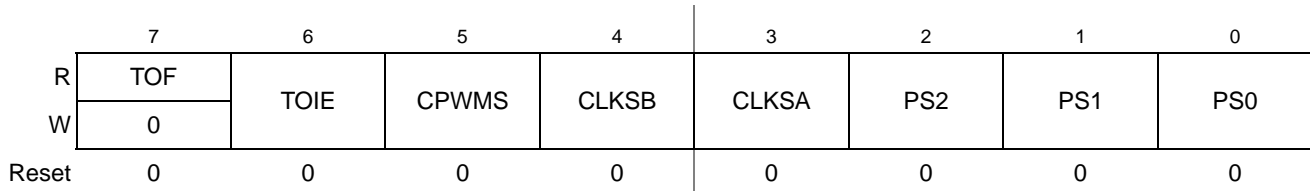
	7	6	5	4	3	2	1	0
R	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
W	0							
Reset	0	0	0	0	0	0	0	0

Figure 21-7. TPM Status and Control Register (TPMxSC)

Table 21-2. 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 completed, the sequence is reset so TOF remains 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. 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.
4–3 CLKS[B:A]	Clock source selection bits. As shown in Table 21-3, this 2-bit field is used to disable the TPM counter or select one of three clock sources to TPM counter and counter prescaler.
2–0 PS[2:0]	Prescale factor select. This 3-bit field selects one of eight division factors for the TPM clock as shown in Table 21-4. This prescaler is located after any clock synchronization or clock selection so it affects the clock selected to drive the TPM counter. The new prescale factor affects the selected clock on the next bus clock cycle after the new value is updated into the register bits.

Table 21-3. TPM Clock Selection

CLKSB:CLKSA	TPM Clock to Prescaler Input
00	No clock selected (TPM counter disable)

Table 21-3. TPM Clock Selection

CLKSB:CLKSA	TPM Clock to Prescaler Input
01	Bus clock
10	Fixed frequency clock
11	External clock

Table 21-4. Prescale Factor Selection

PS[2:0]	TPM Clock Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

21.3.2 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 big-endian or little-endian order that 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	TPMxCNT[15:8]							
W	Any write to TPMxCNTH clears the 16-bit counter							
Reset	0	0	0	0	0	0	0	0
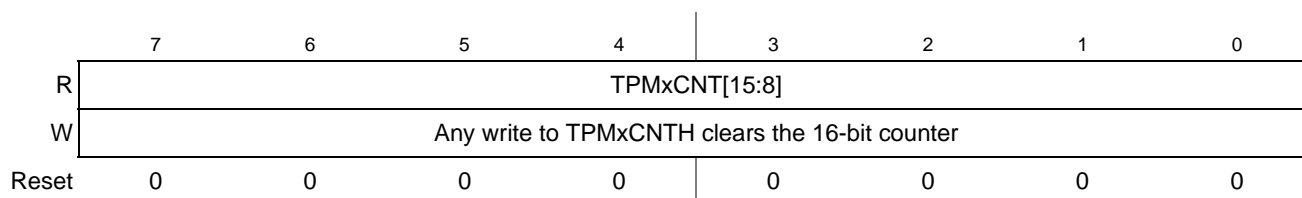
Figure 21-8. TPM Counter Register High (TPMxCNTH)

	7	6	5	4	3	2	1	0
R				TPMxCNT[7:0]				
W			Any write to TPMxCNTL clears the 16-bit counter					
Reset	0	0	0	0	0	0	0	0
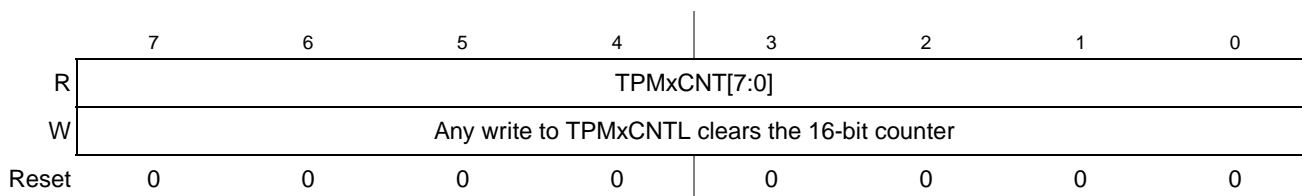
Figure 21-9. TPM Counter Register Low (TPMxCNTL)

When BDM is active, the timer counter is frozen (this is the value you read). The coherency mechanism is frozen so 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 you were in the middle of reading a 16-bit register when BDM became active, it reads 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:TPMxCNTL registers, regardless of the data involved in the write.

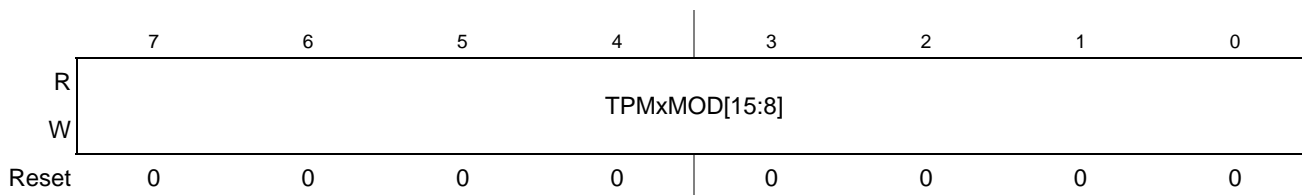## 21.3.3 TPM Counter Modulo Registers (TPMxMODH:TPMxMODL)

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 that results in a free running timer counter (modulo disabled).

Writes to any of the registers TPMxMODH and TPMxMODL actually writes to buffer registers and the registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB and CLKSA are not cleared, 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 is 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) so 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				TPMxMOD[15:8]				
Reset	0	0	0	0	0	0	0	0

Figure 21-10. TPM Counter Modulo Register High (TPMxMODH)

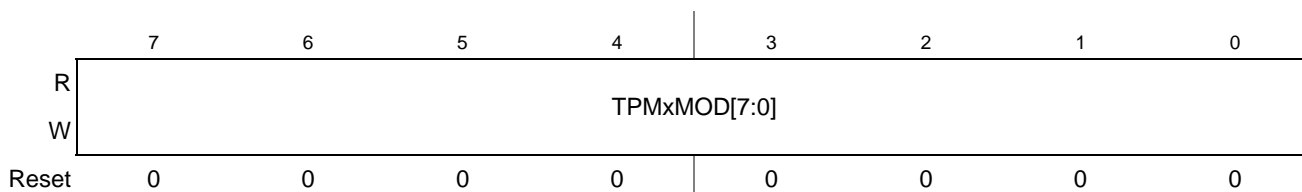|	7	6	5	4	3	2	1	0
R								
W				TPMxMOD[7:0]				
Reset	0	0	0	0	0	0	0	0

Figure 21-11. 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 occurs.

21.3.4 TPM Channel n Status and Control Register (TPMxCnSC)

TPMxCnSC contains the channel-interrupt-status flag and control bits that configure the interrupt enable, channel configuration, and pin function.
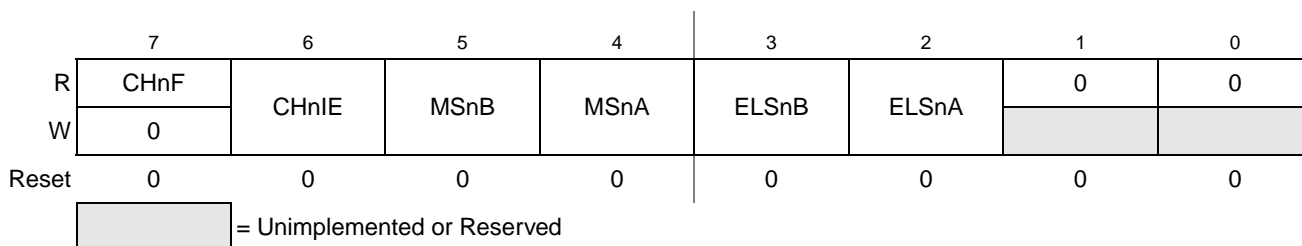
	7	6	5	4	3	2	1	0
R	CHnF	CHnIE	MSnB	MSnA	ELSnB	ELSnA	0	0
W	0							
Reset	0	0	0	0	0	0	0	0

    = Unimplemented or Reserved

Figure 21-12. TPM Channel n Status and Control Register (TPMxCnSC)

Table 21-5. 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 input. 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 is not 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 this bit is set and channel n interrupt is enabled (CHnIE = 1). Clear CHnF by reading TPMxCnSC while this bit is set and then writing a logic 0 to it. If another interrupt request occurs before the clearing sequence is completed CHnF remains set. This is done so a CHnF interrupt request is not lost due to clearing a previous CHnF. Reset clears this 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 this bit. 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 is cleared, setting the MSnB bit configures TPM channel n for edge-aligned PWM mode. Refer to the summary of channel mode and setup controls in Table 21-6.

Table 21-5. TPMxCnSC Field Descriptions (Continued)

Field	Description
4 MSnA	Mode select A for TPM channel n. When CPWMS and MSnB are cleared, the MSnA bit configures TPM channel n for input capture mode or output compare mode. Refer to Table 21-6 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 21-6, these bits select the polarity of the input edge that triggers an input capture event, select the level that is driven in response to an output compare match, or select the polarity of the PWM output. If ELSnB and ELSnA bits are cleared, the channel pin is not controlled by TPM. This configuration can be used by software compare only, because it does not require the use of a pin for the channel.

Table 21-6. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin is not controlled by TPM. It is reverted 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	00	Output compare	Software compare only
		01		Toggle output on channel match
		10		Clear output on channel match
		11		Set output on channel match
	1X	10	Edge-aligned PWM	High-true pulses (clear output on channel match)
		X1		Low-true pulses (set output on channel match)
1	XX	10	Center-aligned PWM	High-true pulses (clear output on channel match when TPM counter is counting up)
		X1		Low-true pulses (set output on channel match when TPM counter is counting up)

21.3.5   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.
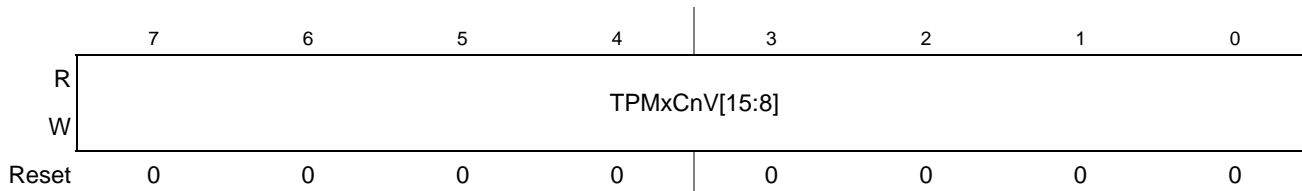
	7	6	5	4	3	2	1	0
R W				TPMxCnV[15:8]				
Reset	0	0	0	0	0	0	0	0

Figure 21-13. TPM Channel Value Register High (TPMxCnVH)

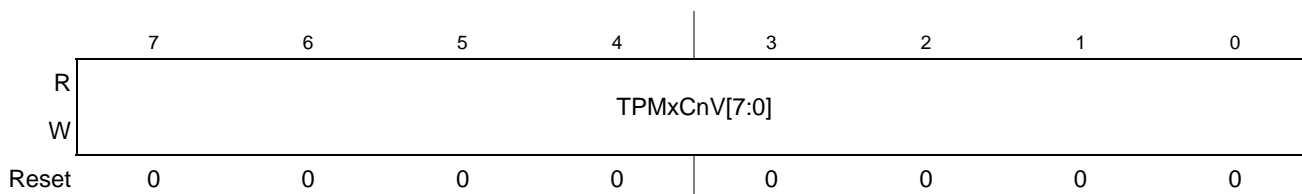|	7	6	5	4	3	2	1	0
R								
W				TPMxCnV[7:0]				
Reset	0	0	0	0	0	0	0	0

Figure 21-14. 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 is ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxCnSC register) so 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 you were in the middle of reading a 16-bit register when BDM became active, it reads 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 were 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:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written.
- If CLKSB and CLKSA are not cleared and in output compare mode, 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 and CLKSA are not cleared and in EPWM or CPWM modes, 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 is 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 that is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen so 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 and output compare operation after 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 is fully exercised, the channel registers are updated using the buffered values (while BDM was not active).

21.4   Functional Description

All TPM functions are associated with a central 16-bit counter that allows flexible selection of the clock and prescale factor. There is also a 16-bit modulo register associated with this 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 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 TPM 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 are covered in the associated mode explanation sections.

21.4.1   Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

21.4.1.1   Counter Clock Source

The 2-bit field, CLKSB:CLKSA, in the timer status and control register (TPMxSC) disables the TPM counter or selects one of three clock sources to TPM counter (Table 21-3). After any MCU reset, CLKSB and CLKSA are cleared so no clock is selected and the TPM counter is disabled (TPM is in a very low power state). You can read or write these control bits at any time. Disabling the TPM counter by writing 00 to CLKSB:CLKSA bits, does not affect the values in the TPM counter or other registers.

The fixed frequency clock is an alternative clock source for the TPM counter that allows the selection of a clock other than the bus clock or external clock. This clock input is defined by chip integration. You can refer chip specific documentation for further information. Due to TPM hardware implementation limitations, the frequency of the fixed frequency clock must not exceed the bus clock frequency. The fixed frequency clock has no limitations for low frequency operation.

The external clock passes through a synchronizer clocked by the bus clock to assure that counter transitions are properly aligned to bus clock transitions.Therefore, in order to meet Nyquist criteria considering also jitter, the frequency of the external clock source must not exceed 1/4 of the bus clock frequency.

When the external clock source is shared with a TPM channel pin, this pin must not be used in input capture mode. However, this channel can be used in output compare mode with ELSnB:ELSnA = 0:0 for software timing functions. In this case, the channel output is disabled, but the channel match events continue to set the appropriate flag.

21.4.1.2   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 interrupt is generated, or interrupt-driven operation (TOIE = 1) where the interrupt is generated whenever the TOF is set.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS = 1). If CPWMS is cleared and there is no modulus limit, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF is set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF is 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 is set as the counter changes direction at the end of the count value set in the modulus register (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).

21.4.1.3   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 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. The terminal count value and 0x0000 are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) is set at the end of the terminal-count period (as the count changes to the next lower count value).

21.4.1.4   Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to 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.

21.4.2   Channel Mode Selection

If CPWMS is cleared, MSnB and MSnA bits determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

21.4.2.1   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 is 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 TPMxCnSC.

An input capture event sets a flag bit (CHnF) that optionally generates a CPU interrupt request.

While in BDM, the input capture function works as configured. When an external event occurs, the TPM latches the contents of the TPM counter (frozen because of the BDM mode) into the channel value registers and sets the flag bit.

21.4.2.2   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 TPMxCnVH:TPMxCnVL registers of an output compare channel, the TPM can set, clear, or toggle the channel pin.

Writes to any of TPMxCnVH and TPMxCnVL registers actually write to buffer registers. In output compare mode, the TPMxCnVH:TPMxCnVL registers are updated with the value of their write buffer only after both bytes were written and according to the value of CLKSB:CLKSA bits:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB and CLKSA are not cleared, 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) that optionally generates a CPU interrupt request.

21.4.2.3   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 value of the timer channel register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by ELSnA bit. 0% and 100% duty cycle cases are possible.

The time between the modulus overflow and the channel match value (TPMxCnVH:TPMxCnVL) is the pulse width or duty cycle (Figure 21-15). If ELSnA is cleared, the counter overflow forces the PWM signal high, and the channel match forces the PWM signal low. If ELSnA is set, the counter overflow forces the PWM signal low, and the channel match forces the PWM signal high.
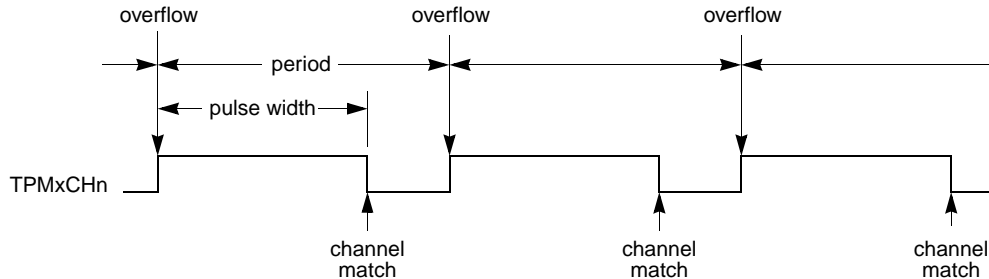
Figure 21-15. EPWM period and pulse width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. A 100% duty cycle is 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.

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, the TPMxCnVH:TPMxCnVL registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB and CLKSA are not cleared, 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.

21.4.2.4 Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The channel match 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 must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA determines the polarity of the CPWM signal.

pulse width = 2 × (TPMxCnVH:TPMxCnVL)

period = 2 × (TPMxMODH:TPMxMODL); TPMxMODH:TPMxMODL = 0x0001–0x7FFF

If TPMxCnVH:TPMxCnVL is zero or negative (bit 15 set), the duty cycle is 0%. If TPMxCnVH:TPMxCnVL is a positive value (bit 15 clear) and is greater than the non-zero modulus setting, the duty cycle is 100% because the channel match never occurs. 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 is much longer than required for normal applications.

All zeros in TPMxMODH:TPMxMODL is a special case that must not be used with center-aligned PWM mode. When CPWMS is cleared, this case corresponds to the counter running free from 0x0000 through 0xFFFF. When CPWMS is set, 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 channel match value in the TPM channel registers (times two) determines the pulse width (duty cycle) of the CPWM signal (Figure 21-16). If ELSnA is cleared, a channel match occurring while counting up clears the CPWM output signal and a channel match occurring while counting down sets the output. 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.
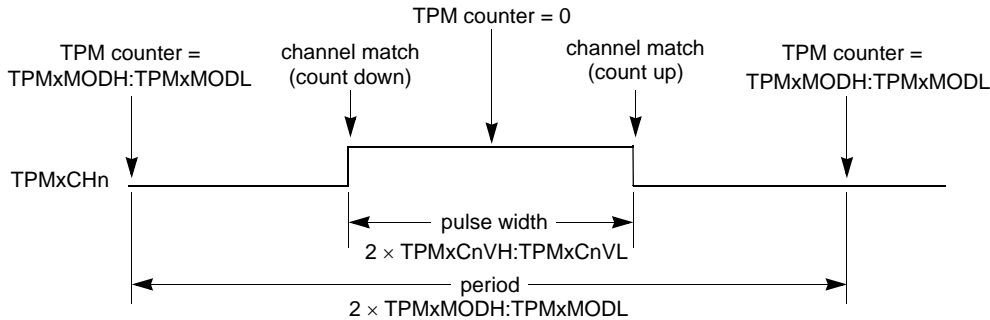
Figure 21-16. CPWM period and pulse width (ELSnA=0)

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 is set.

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 center-aligned PWM mode, the TPMxCnVH:TPMxCnVL registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits:

- If CLKSB and CLKSA are cleared, the registers are updated when the second byte is written
- If CLKSB and CLKSA are not cleared, 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.

When TPMxCNTH:TPMxCNTL equals TPMxMODH:TPMxMODL, the TPM can optionally generate a TOF interrupt (at the end of this count).

21.5 Reset Overview

21.5.1 General

The TPM is reset whenever any MCU reset occurs.

21.5.2 Description of Reset Operation

Reset clears TPMxSC that disables TPM counter clock and overflow interrupt (TOIE=0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared. This configures all TPM channels for input capture operation and the associated pins are not controlled by TPM.

21.6 Interrupts

21.6.1 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 21-7.

Table 21-7. Interrupt Summary

Interrupt	Local Enable	Source	Description
TOF	TOIE	Counter overflow	Set each time the TPM counter reaches its terminal count (at transition to its next count value)
CHnF	CHnIE	Channel event	An input capture event or channel match took place on channel n

The TPM module provides high-true interrupt signals.

21.6.2 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 is read (polled) by software to determine that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable the interrupt generation. While the interrupt enable bit is set, the interrupt is generated whenever the associated interrupt flag is set. 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 followed by a write of zero 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.

21.6.2.1 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.

21.6.2.1.1 Normal Case

When CPWMS is cleared, TOF is set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFF to 0x0000.

21.6.2.1.2 Center-Aligned PWM Case

When CPWMS is set, TOF is 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).

21.6.2.2 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).

21.6.2.2.1 Input Capture Events

When a channel is configured as an input capture channel, the ELSnB:ELSnA bits select if channel pin is not controlled by TPM, rising edges, falling edges, or any edge as the edge that 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 21.6.2, "Description of Interrupt Operation."

21.6.2.2.2 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 in Section 21.6.2, "Description of Interrupt Operation."

21.6.2.2.3 PWM End-of-Duty-Cycle Events

When the channel is configured for edge-aligned PWM, the channel flag is set when the timer counter matches the channel value register that 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 when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described in Section 21.6.2, "Description of Interrupt Operation."

Chapter 22
Universal Serial Bus (S08USBV1)

22.1 Introduction

This section describes the USB device controller.

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 regulator, Endpoint RAM and other control logics.

22.1.1 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. These requirements must be met in order to use the USB.

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

22.1.2 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.

22.1.3 USB Clock Gating

The bus clock to the USB can be gated on and off using the USB bit in SCGC2. This bit is set after any reset, which enables the bus clock to this module. To conserve power, the USB bit can be cleared to disable the clock to this module when not in use. See Section 5.7.9, "System Clock Gating Control 2 Register (SCGC2)," for details.

Figure 22-1. Block Diagram Highlighting the USB Module
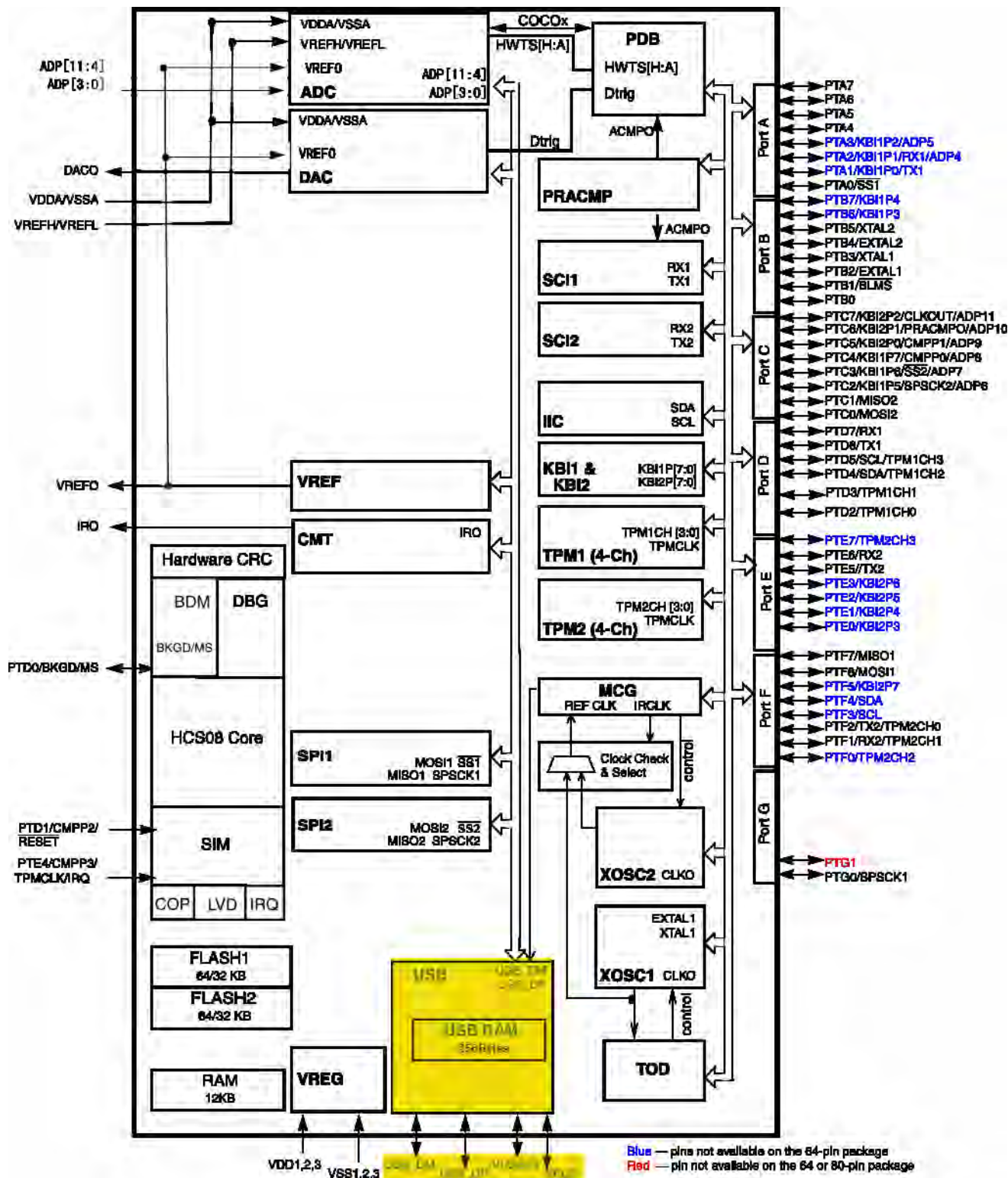
22.2   Features

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

22.2.1 Modes of Operation

Table 22-1. Operating Modes

Mode	Description
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 tranceiver must be disabled by firmware.
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 such that the USB suspend current consumption targets are achieved . The USB module is notified about entering suspend mode when the SLEEPF flag is set; this occurs after the USB bus is idle for 3ms. 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 become set. This will trigger an asynchronous interrupt that will wake the MCU from stop3 mode and enable clocks to the USB module. The USBRESMEN bit should then be cleared immediately after Stop3 recovery to clear the LPRESF flag bit.
Wait	USB module is operational.

22.2.2 Block Diagram

Figure 22-2 is a block diagram of the USB module.
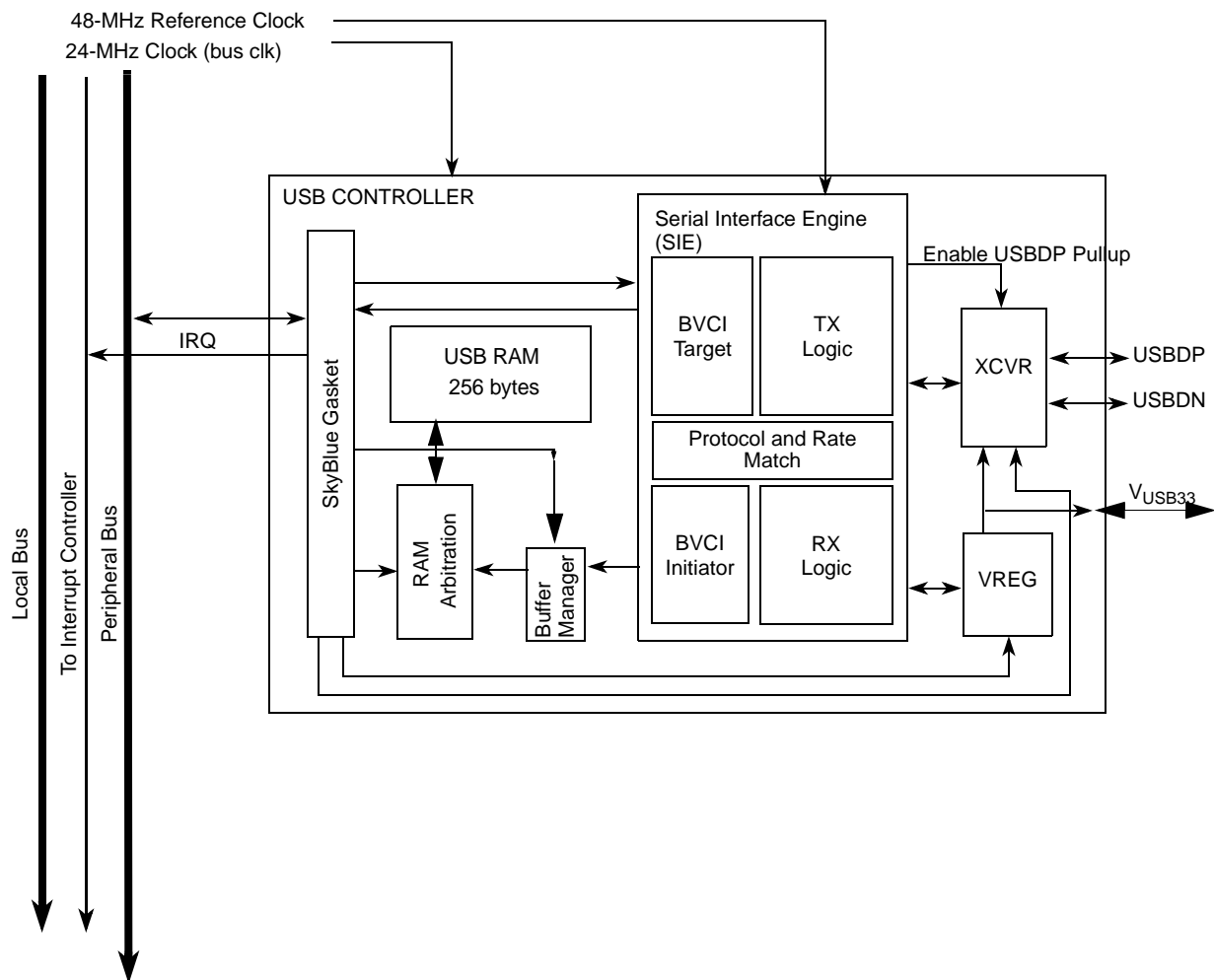
Figure 22-2. USB Module Block Diagram

22.3 External Signal Description

The USB module requires both data and power pins. Table 22-2 describes each of the USB external pins.

Table 22-2. 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	3.3V USB voltage regulator output or 3.3V USB transceiver/resistor supply input.	—

22.3.1 USBDP

USBDP is the positive USB differential signal. In a USB peripheral application, connect an external 33 ohms ±1% resistor in series with this signal in order to meet the USB Specification Rev. 2.0 impedance requirement.

22.3.2 USBDN

USBDN is the negative USB differential signal. In a USB peripheral application, connect an external 33 ohms ±1% resistor in series with this signal in order to meet the USB Specification, Rev. 2.0 impedance requirement.

22.3.3 V_{USB33}

V_{USB33} is connected to the on-chip 3.3V 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 pull-up resistor. If the VREG is disabled by software, the application must input an external 3.3V power supply to the USB module via V_{USB33}.

22.4 Register Definition

This section describes the memory map and control/status registers for the USB module.

22.4.1 USB Control Register 0 (USBCTL0)

	7	6	5	4	3	2	1	0
R	0	USBPU	USBRES MEN	LPRESF	0	USBVREN	0	USBPHYEN
W	USBRESET							
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 22-3. USB Transceiver and Regulator Control Register 0 (USBCTL0)
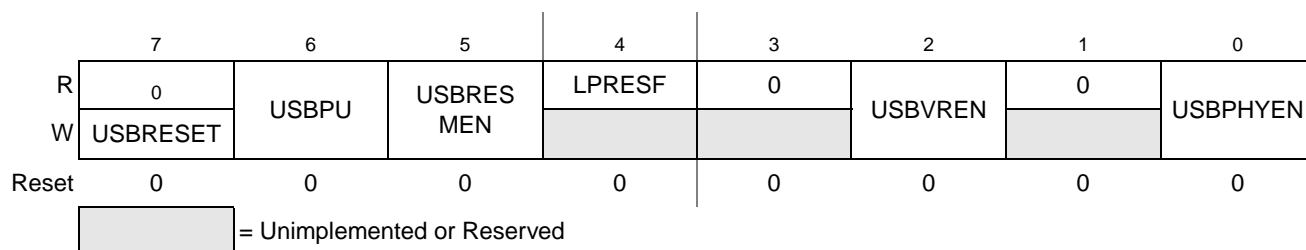
Table 22-3. USBCTL0 Field Descriptions

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 USBPHY and USBVREG). 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 pull-up resistor on the USBDP line. 0 Internal USBDP pull-up resistor is disabled; The application can use an external pull-up resistor 1 Internal USBDP pull-up 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 become set, indicating a K-state on the USB bus. This bit should be set before entering low-power stop3 mode only after SLEEPF=1 (USB is entering suspend mode). It should 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 should 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 PHY Transceiver Enable** — When the USB PHY is disabled, USBDP and USBDN are hi-Z. It is recommended that the USB PHY be enabled before setting the USBEN bit in the CTL register. The firmware must ensure that the USB PHY remains enabled when entering USB SUSPEND mode. 0 On-chip USB PHY is disabled 1 On-chip USB PHY is enabled

22.4.2 Peripheral ID Register (PERID)

The Peripheral ID Register will read 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 22-4. Peripheral ID Register (PERID)

Table 22-4. 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 core.

22.4.3 Peripheral ID Complement Register (IDCOMP)

The Peripheral ID Complement Register 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
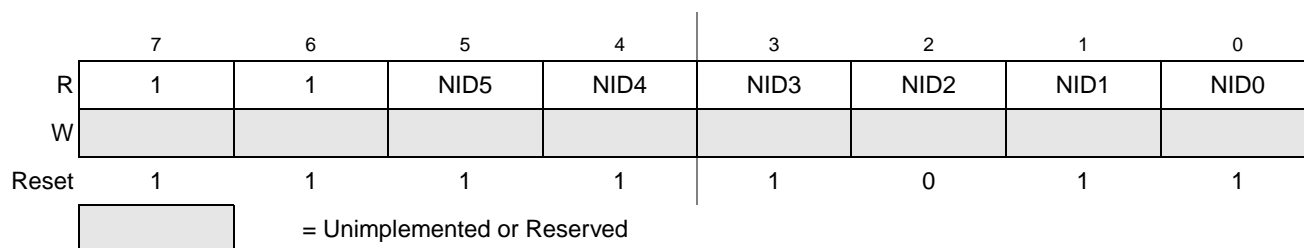
Figure 22-5. Peripheral ID Complement Register (IDCOMP)

Table 22-5. IDCOMP Field Descriptions

Field	Description
5:0 NID[5:0]	**Compliment ID Number** — One's complement version of ID[5:0].

22.4.4 Peripheral Revision Register (REV)

The Revision Register will read back the value of the USB Peripheral Revision.
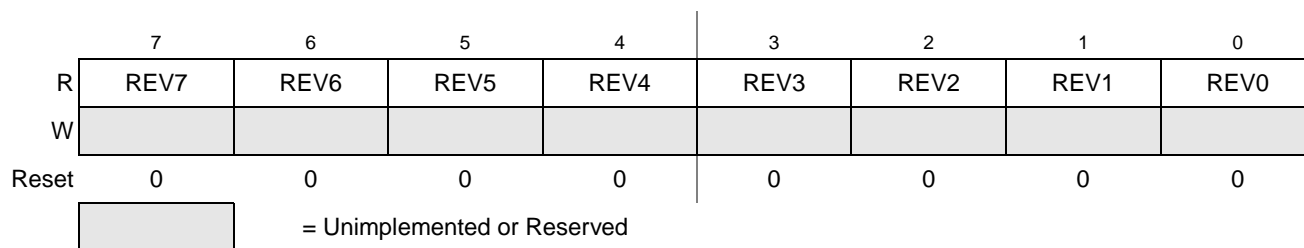
	7	6	5	4	3	2	1	0
R	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
W								
Reset	0	0	0	0	0	0	0	0

☐ = Unimplemented or Reserved

Figure 22-6. Peripheral Revision Register (REV)

Table 22-6. REV Field Descriptions

Field	Description
8–0 REV[7:0]	**Revision** — Revision number of the USB module USB 2.0 Core.

22.4.5 Interrupt Status Register (INTSTAT)

The Interrupt Status Register (INTSTAT) contains bits for each of the interrupt sources within the USB module. Each of these bits are qualified with their 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 one 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
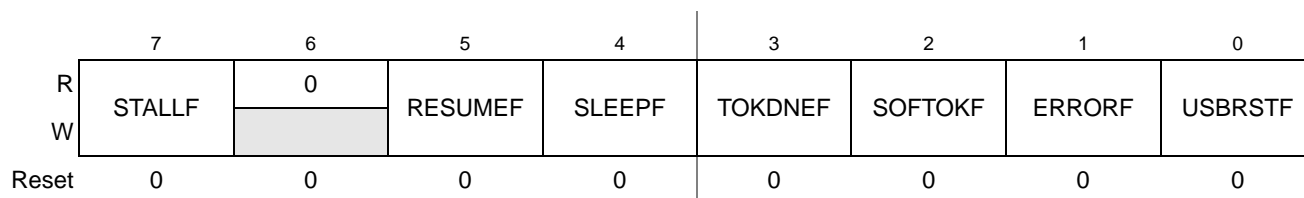
Figure 22-8. Interrupt Status Register (INTSTAT)

Table 22-8. 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 should be 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

Table 22-8. INTSTAT Field Descriptions (Continued)

Field	Description
3 TOKDNEF	**Token Complete Flag** — This bit is set when the current transaction is complete. The firmware should 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 a Start Of Frame (SOF) token 1 The USB module has received a Start Of Frame (SOF) token
1 ERRORF	**Error Flag** — This bit is set when any of the error conditions within the ERRSTAT register have 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

22.4.6 Interrupt Enable Register (INTENB)

The Interrupt Enable Register (INTENB) contains enable 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	STALL		RESUME	SLEEP	TOKDNE	SOFTOK	ERROR	USBRST
Reset	0	0	0	0	0	0	0	0
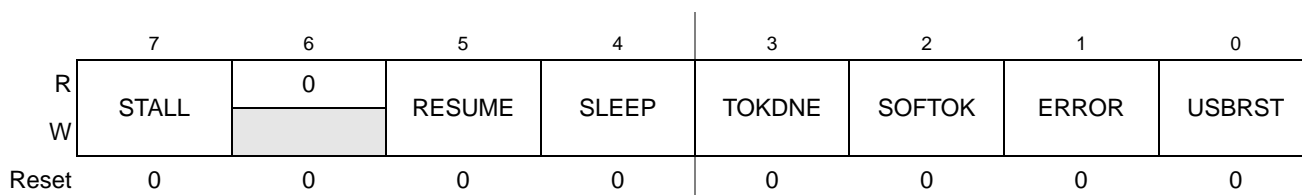
Figure 22-9. Interrupt Enable Register (INTENB)

Table 22-9. 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

Table 22-9. INTENB Field Descriptions (Continued)

Field	Description
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
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

22.4.7   Error Interrupt Status Register (ERRSTAT)

The Error Interrupt Status Register (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 22.4.8, "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 one 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
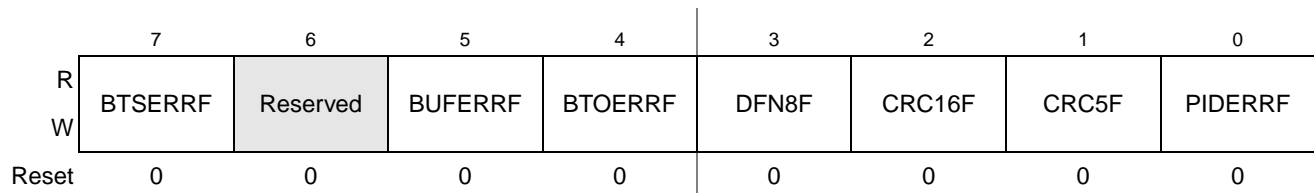
Figure 22-10. Error Interrupt Status Register (ERRSTAT)

Table 22-10. 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

Table 22-10. ERRSTAT Field Descriptions (Continued)

Field	Description
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
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

22.4.8 Error Interrupt Enable Register (ERRENB)

	7	6	5	4	3	2	1	0
R W	BTSERR	0	BUFERR	BTOERR	DFN8	CRC16	CRC5	PIDERR
Reset	0	0	0	0	0	0	0	0
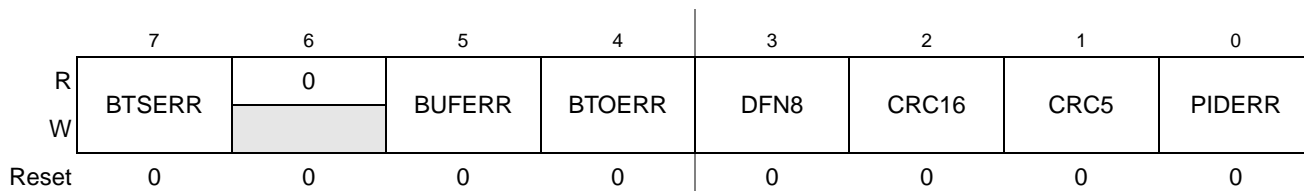
Figure 22-11. Error Interrupt Enable Register (ERRENB)

Table 22-11. 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

Table 22-11. ERRSTAT Field Descriptions (Continued)

Field	Description
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
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

22.4.9 Status Register (STAT)

The Status Register reports the transaction status within the USB module. When the MCU receives a TOKDNE interrupt, the Status Register should be 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.

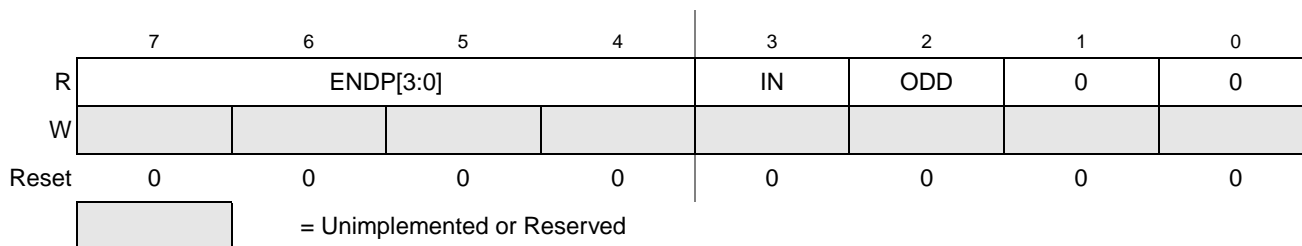|	7	6	5	4	3	2	1	0
R		ENDP[3:0]			IN	ODD	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 22-12. Status Register (STAT)

Table 22-12. 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
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 is 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

22.4.10 Control Register (CTL)

The control register provides various control and configuration information for the USB module.
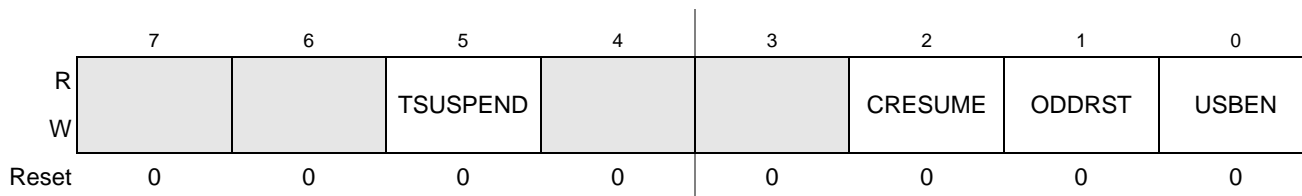
	7	6	5	4	3	2	1	0
R			TSUSPEND			CRESUME	ODDRST	USBEN
W								
Reset	0	0	0	0	0	0	0	0

Figure 22-13. Control Register (CTL)

Table 22-13. CTL Field Descriptions

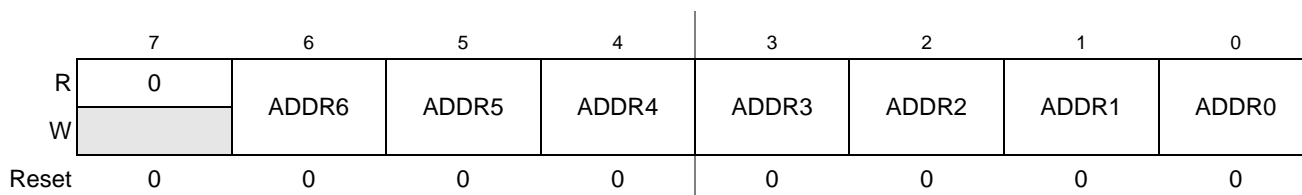| Field	Description
5 TSUSPEND	**Transaction Suspend** — This bit is set by the serial interface engine (SIE) when a Setup Token is received, allowing software to dequeue any pending packet transactions in the BDT before resuming token processing. The TSUSPEND bit informs the processor that the SIE has 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 (Section 7.1.7.7) and then clear it to 0. 0 Do not execute remote wakeup 1 Execute resume signaling - remote wakeup

Table 22-13. CTL Field Descriptions (Continued)

Field	Description
1 ODDRST	**Odd Reset** — Setting this bit resets 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 enables 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

22.4.11 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 22-14. Address Register (ADDR)

Table 22-14. 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

22.4.12 Frame Number Register (FRMNUML, FRMNUMH)

The Frame Number Registers contains the 11-bit frame number. The Frame Number Register requires 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
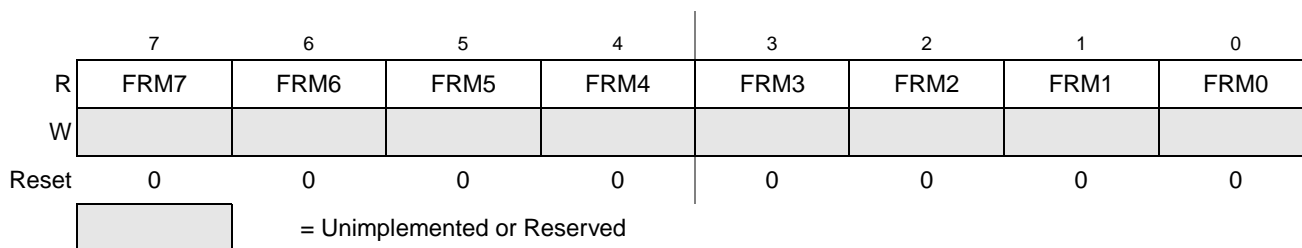
Figure 22-15. Frame Number Register Low (FRMNUML)

Table 22-15. 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
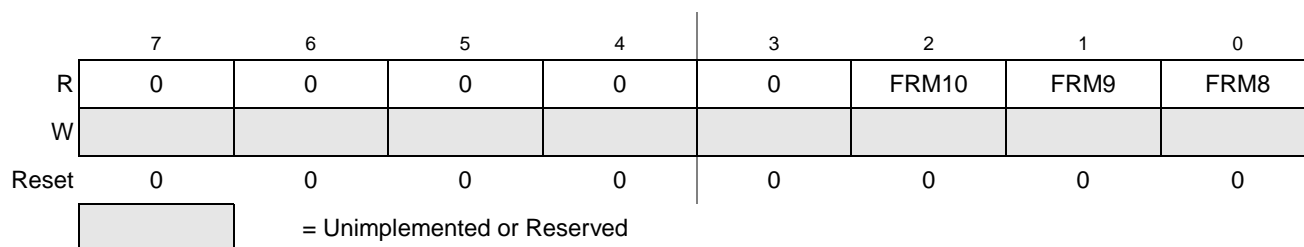
Figure 22-16. Frame Number Register High (FRMNUMH)

Table 22-16. 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.

22.4.13 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 should set EPCTL0 to contain 0x0D.

.

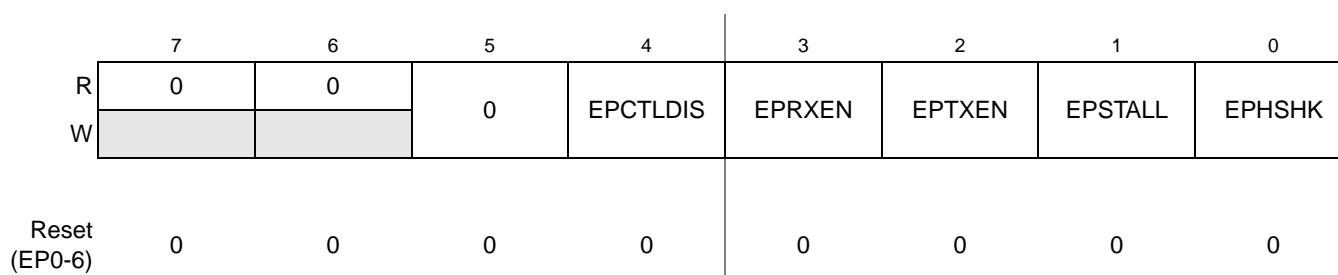|	7	6	5	4	3	2	1	0
R	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
W								
Reset (EP0-6)	0	0	0	0	0	0	0	0

Figure 22-17. Endpoint Control Register (EPCTLn)

Table 22-17. 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 22-18.
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 22-18.
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 22-18.
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 22-18. 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	RESERVE

22.5 Functional Description

This section describes the functional behavior of the USB module. It documents data packet processing both for endpoint 0 and data endpoints, USB suspend and resume states, SOF token processing, reset conditions and interrupts.

22.5.1 Block Descriptions

The block diagram is found in Figure 22-2. 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 3.3V regulator (VREG), endpoint buffer manager, shared RAM arbitration, USB RAM and the SkyBlue gasket.

22.5.1.1 USB Serial Interface Engine (SIE)

The USB serial interface engine (SIE) is comprised of two major functions: (1) TX Logic and (2) 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.

22.5.1.1.1 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.

22.5.1.1.2 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 NRZ 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.

22.5.1.2 MCU/Memory Interfaces

22.5.1.2.1 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.

22.5.1.2.2 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.

22.5.1.2.3 RAM Arbitration

The arbitration block allows access to the USB RAM block from the SkyBlue gasket block and from the SIE.

22.5.1.3 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 both 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 system memory.

22.5.1.4 USB Transceiver (XCVR)

The USB transceiver is electrically compliant to the *Universal Serial Bus Specification 2.0.* This physical layer 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.

22.5.1.5 USB On-Chip Voltage Regulator (VREG)

The on-chip 3.3V regulator provides a stable power source to power the USB internal transceiver and provide for the termination of an internal or external pull-up resistor. When the on-chip regulator is enabled, it requires a voltage supply input in the range from 3.9V to 5.5V, and the voltage regulator output will be in the range of 3.0V to 3.6V.

With a dedicated on-chip USB 3.3V 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 3.3V regulator is disabled, a 3.3V 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 3.3V 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 pull-up 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 3.0V to 3.6V, with a 900 ohm (worst-case) pull-up.
- Power off — This mode is entered anytime when stop2 is entered or when the voltage regulator is disabled.

22.5.1.6   USB On-Chip USBDP Pullup Resistor

The pull-up resistor on the USBDP line required for full-speed operation by the USB Specification Rev. 2.0 can be either internal or external to the MCU, depending on the application requirements. An on-chip pull-up resistor, implemented as specified in the USB 2.0 resistor ECN, is optionally available via firmware configuration. Alternatively, this on-chip pull-up resistor can be disabled, and the USB module can be configured to use an external pull-up resistor for the USBDP line instead. If using an external pull-up 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 pull-up resistor is internal or external to the MCU. If USBPU is clear, the internal pull-up resistor on USBDP is disabled, and an external USBDP pull-up can be used. When using an external USBDP pull-up, if the voltage regulator is enabled, the V_{USB33} voltage output can be used with the USBDP pull-up. While the use of the internal USBDP pull-up resistor is generally recommended, the figure below shows the USBDP pull-up resistor configuration for a USB device using an external resistor tied to V_{USB33}.
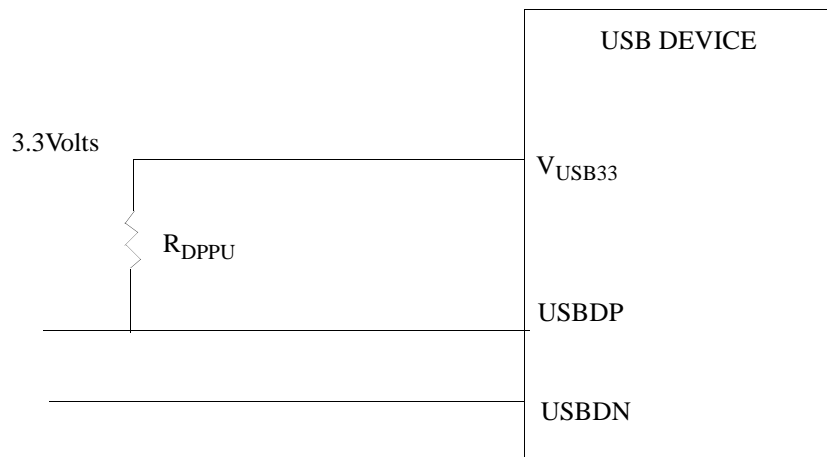

Figure 22-18. USBDP/USBDN Pull-up Resistor Configuration for USB module

22.5.1.7   USB Powering and USBDP Pull-up Enable Options

The USB module provides a single-chip solution for USB device applications that are either 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 pull-up resistor on the USBDP line. For the USB module on this device, the pull-up on USBDP should only be applied when a valid VBUS connection is sensed, as required by the USB specification.

In bus-powered applications, system power should 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 pull-up can be enabled accordingly.

With self-powered applications, determining when a valid USB connection is made is different than with bus-powered applications. For 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 pull-up resistor can be enabled using the USBPU bit. If an external pull-up resistor is used instead of the internal one, the VBUS sensing mechanism must be included in the system design.

Table 22-19 below summarizes the differences in enabling the USBDP pull-up for the different USB power modes.

Table 22-19. USBDP Pull-up Enable for Different USB Power Modes

Power	USBDP Pull-up	Pull-up 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

22.5.2   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 will need to intelligently manage 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 should not modify the BD or its corresponding data buffer.

22.5.2.1   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. The first BD entry is the EVEN BD and second BD entry is the ODD BD.

22.5.2.2 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 22-20. 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 ... 0xFF	USB RAM available for endpoint buffers	

When the USB module receives a USB token on an enabled endpoint, it interrogates the BDT. The USB module must read the corresponding endpoint BD entry and determine if it owns the BD and corresponding data buffer.

22.5.2.3 Buffer Descriptor Formats

The buffer descriptors (BD's) are groups of registers which provide endpoint buffer control information for the USB module and the MCU. The BD's have different meanings based on who is reading the BD in memory.

The USB module uses the data stored in the BD's 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 is 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 Table 22-21 for the BD example of Enndpoint 0 IN start from USB RAM offset 0x00.

The format for the buffer descriptor is shown in Table 22-22.

Table 22-21. Buffer Descriptor Example

Offset	Register							
	7	6	5	4	3	2	1	0
0x00	OWN	DATA0/1	0 BDTKPID[3]	0 BDTKPID[2]	DTS BDTKPID[1]	BDTSTALL BDTKPID[0]	0	0
0x01	BC[7:0]							
0x02	EPADR[9:2]							

Table 22-22. 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 should not change it in any way. This byte of the BD should 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
DTS BDTKPID[1]	**Data Toggle Synchronization / BD Token PID [1]** — This bit enables data toggle synchrnoization. 0 No data toggle synchronization is performed. 1 Data toggle synchronization is performed.
BDTSTALL BDTKPID[0]	**BDT Stall / BD Token PID [0]** — 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
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[9:2]	**Endpoint Address**— The endpoint address bits represent the upper 8 bits of the 10-bit buffer address within the module's local USB RAM. Bits [1:0] of EPADR are always zero, therefore the address of the buffer must always start on a four-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.
BDTKPID	**BD Token PID** — 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.

22.5.3   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 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 22-19 shows a timeline for how a typical USB token would be processed.
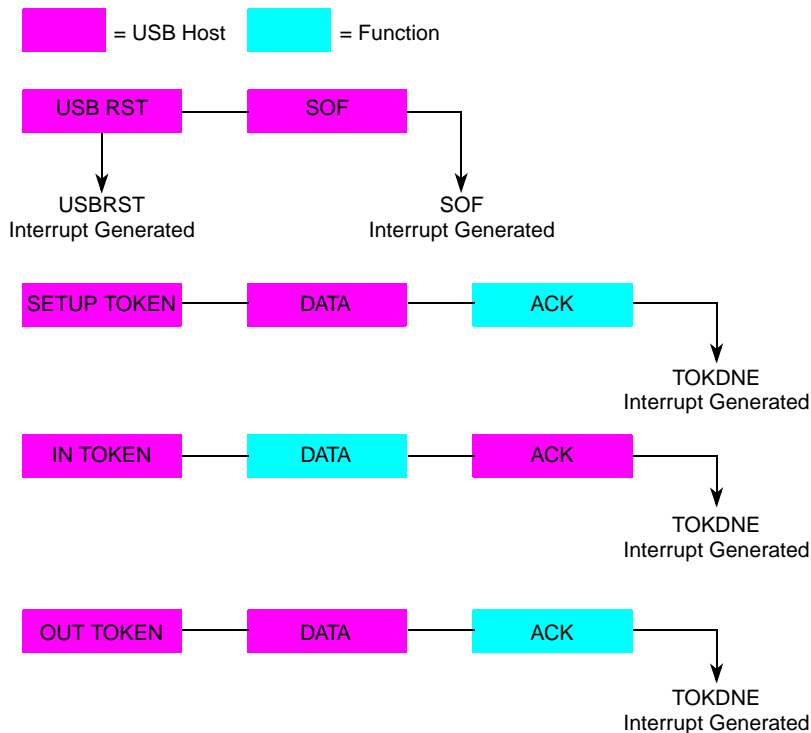

Figure 22-19. USB Packet Flow

The USB has two sources of data overrun error:

1. 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.
2. The packet received may be larger than the negotiated MAXPACKET size. This is 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 of the core. Depending on the values of the INTENB and ERRENB register, the core may assert an interrupt to notify the processor 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 a 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 core 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 core will assert the BUFERRF bit of the ERRSTAT register (which could trigger an interrupt, as above) and a TOKDNE interrupt will fire. 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.

22.5.4 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.

22.5.4.1 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.

22.5.4.2 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 64K bytes 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 should 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 zero 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.

22.5.4.3   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.

22.5.5   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.

22.5.6 Suspend/Resume

The USB supports a single low-power mode called Suspend. Getting into and out of the suspended state is described in the following sections.

22.5.6.1 Suspend

The USB host can put a single device or the entire bus into the suspended 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 3ms. 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 500µA when in the suspended 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 3ms idle, the USB device shall 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 becomes set enables this asynchronous notification feature. The USB resume signaling will then cause the LPRESF bit to become 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.

22.5.6.2 Resume

There are three ways to get out of the suspended 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).

22.5.6.2.1 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 become 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 becomes set, firmware should 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.

22.5.6.2.2   USB Reset Signaling

Reset can wake a device from the suspended mode.

22.5.6.2.3   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.

22.5.7   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.

22.5.7.1   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.

22.5.7.2   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 enabled
- USB transceiver enabled
- USBDP pullup disabled
- Endpoints disabled except for EP0
- USB address register set to zero

22.5.8 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 look at the STAT register and determine which BDT entry was affected by the transaction.

Chapter 23
Voltage Reference Module (S08VREFV1)

23.1 Introduction

The Voltage Reference is intended to supply an accurate voltage output that is trimmable by an 8bit register in 0.5mV steps. A nominal trim value to achieve the specified VREF voltage will be automatically loaded upon a reset into the VREF Trim Register. The voltage reference can be used to provide a reference voltage to external peripherals or as a reference to analog peripherals such as the ADC or ACMP.

Figure 25-1 shows the MC9S08JE128 series block diagram with the VREF highlighted.

23.1.1 VREF Configuration Information

The MC9S08JE128 series uses the VREF module to provide the bandgap signal. The bandgap signal is provided to the ADC, DAC, and ACMP peripherals

23.1.2 VREF Clock Gating

The bus clock to the VREF can be gated on and off using the VREF bit in SCGC1. These bits are set after any reset, which enables the bus clock to this module. To conserve power, these bits can be cleared to disable the clock to this module when not in use. See Section 5.7.8, "System Clock Gating Control 1 Register (SCGC1)," for details.
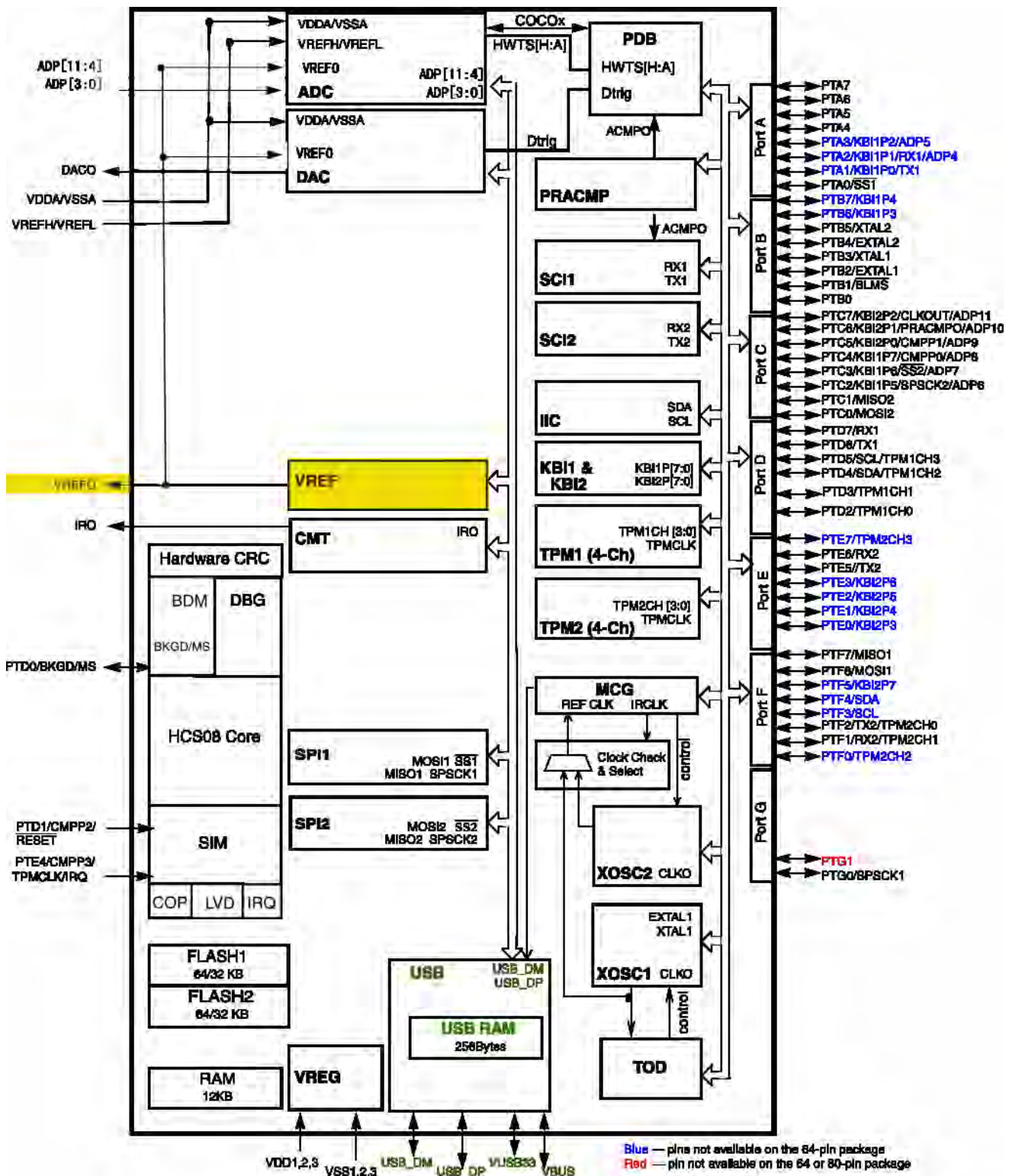
Figure 23-1. Block Diagram Highlighting VREF Module

23.1.3 Overview

The Voltage Reference optimizes the existing bandgap and bandgap buffer system. Unity gain amplifiers ease the design and keep power consumption low. The 8-bit trim register is user accessible so that the voltage reference can be trimmed in application.
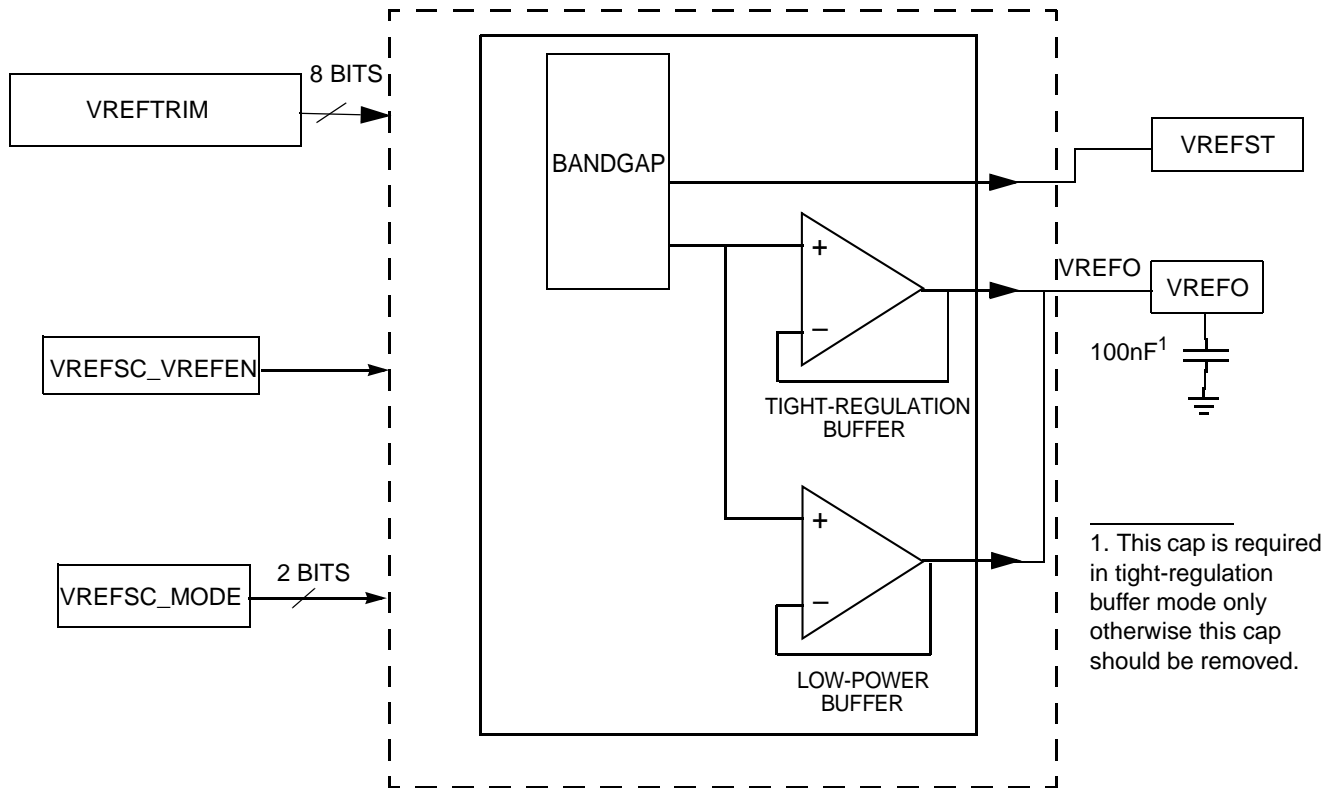
Figure 23-2. Voltage Reference Block Diagram

The voltage reference contains trim resolution of 0.5mV per step. The buffer has modes with improved high-current swing output. In addition, a low-power buffer mode is available for use with ADCs or DACs. The voltage reference can be output on a dedicated output pin[1].

23.1.4 Features

The Voltage Reference module has the following features:

- Programmable trim register with 0.5mV steps, automatically loaded with factory trimmed value upon reset
- Programmable mode selection:
  — Off
  — Bandgap out (or stabilization delay)
  — Low-power buffer mode

1. In Tight-Regulation buffer mode, a 100nF capacitor is required to be connected between the VREFO pad and the ground.

    — Tight-regulation buffer mode
- 1.15 V output at room temperature[1]
- Dedicated output pin VREFO

23.1.5   Modes of Operation

The Voltage Reference continues normal operation in Run, Wait, LPRun, and LPWait modes. The Voltage Reference module can be enabled to operate in STOP3 mode.

23.1.6   External Signal Description

Table 23-1 shows the Voltage Reference signals properties.

Table 23-1. Signal Properties

Name	Function	I/O	Reset	Pull Up
VREFO	Internally generated voltage reference output	O	—	—

23.2   Memory Map and Register Definition

23.2.1   VREF Trim Register (VREFTRM)

The VREFTRM register contains eight bits that contain the trim data for the Voltage Reference as described in Table 23-2.

Register address: Base + 0x00

	7	6	5	4	3	2	1	0
R W				TRM[7:0]				
Reset:	X	X	X	X	X	X	X	X
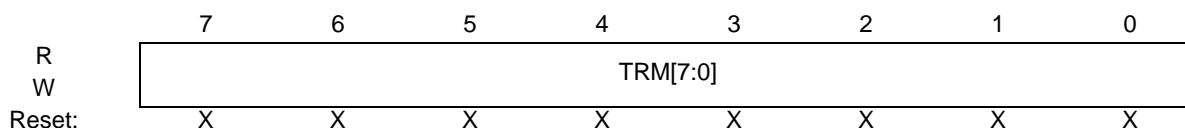
Figure 23-3. VREF Trim Register (VREFTRM)

Table 23-2. VREFTRM Register Field Descriptions

Field	Description
7:0 TRM[7:0]	Trim Bits 7:0 These bits change the resulting VREF output by ±0.5mV for each step.

Table 23-3. VREFTRM Register Settings

TRM7	TRM6	TRM5	TRM4	TRM3	TRM2	TRM1	TRM0	VREF (mV)
1	0	0	0	0	0	0	0	max
1	0	0	0	0	0	0	1	max-0.5

---

1. Please refer to datasheet for details.

Table 23-3. VREFTRM Register Settings (Continued)

TRM7	TRM6	TRM5	TRM4	TRM3	TRM2	TRM1	TRM0	VREF (mV)
1	0	0	0	0	0	1	0	max-1.0
1	0	0	0	0	0	1	1	max-1.5
1	. . . . . .							. . . . . .
1	1	1	1	1	1	0	0	mid+1.5
1	1	1	1	1	1	0	1	mid+1.0
1	1	1	1	1	1	1	0	mid+0.5
1	1	1	1	1	1	1	1	mid
0	0	0	0	0	0	0	0	mid-0.5
0	0	0	0	0	0	0	1	mid-1.0
0	0	0	0	0	0	1	0	mid-1.5
0	0	0	0	0	0	1	1	mid-2.0
0	. . . . . .							. . . . . .
0	1	1	1	1	1	0	0	min+1.5
0	1	1	1	1	1	0	1	min+1.0
0	1	1	1	1	1	1	0	min+0.5
0	1	1	1	1	1	1	1	min

23.2.2   VREF Status and Control Register (VREFSC)

The VREF status and control register contains the control bits used to enable the internal voltage reference and select the buffer mode to be used.

Register address: Base + 0x01

	7	6	5	4	3	2	1	0
R	VREFEN	0	0	0	0	VREFST	MODE	
W								
Reset:	0	0	0	0	0	0	0	0
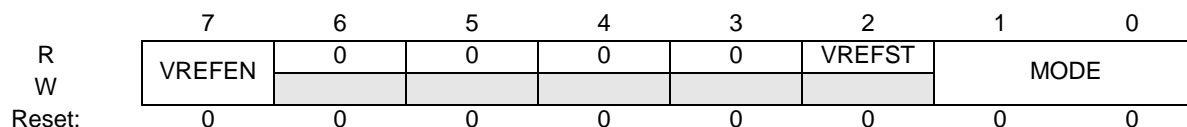
Figure 23-4. Voltage Reference Control Register (VREFSC)

Table 23-4. VREFSC Register Field Descriptions

Field	Description
7 VREFEN	**Internal Voltage Reference Enable** — This bit is used to enable the Voltage Reference module. 0 the Voltage Reference module is disabled 1 the Voltage Reference module is enabled
2 VREFST	**Internal Voltage Reference Stable** — This bit indicates that the Voltage Reference module has completed its startup and stabilization. 0 Voltage Reference module is disabled or not stable 1 Voltage Reference module is stable
1:0 MODE[1:0]	**Mode selection** — These bits are used to select the modes of operation for the Voltage Reference module. 00 Bandgap on only, for stabilization and startup 01 Low-power buffer enabled 10 Tight-regulation buffer enabled 11 RESERVED

23.3 Functional Description

The Voltage Reference is a bandgap buffer system. Unity gain amplifiers are used.

The Voltage Reference can be used in two main cases. It can be used as a reference to analog peripherals such as an ADC channel or analog comparator input. For this case, the low-power buffer can be used. When the tight-regulation buffer is enabled, VREFO can be used both internally and externally.

Table 23-5 shows all possible functional configurations of the Voltage Reference.

Table 23-5. Voltage Reference Functional Configurations

VREFEN	MODE[1:0]	Configuration	Functionality
0	X	Voltage Reference Disabled	Off
1	00	Voltage Reference Enabled, Bandgap on only	Startup and Standby
1	01	Voltage Reference Enabled, Low-Power buffer on	Can be used for internal peripherals only and VREFO pin should not be loaded
1	10	Voltage Reference Enabled, Tight-Regulation buffer on	Can be used both internally and externally. A 100nF capacitor is required on VREFO and 10mA max drive strength is allowed.
1	11	Voltage Reference Disabled	RESERVED

23.3.1   Voltage Reference Disabled, VREFEN=0

When VREFEN=0, the Voltage Reference is disabled, all bandgap and buffers are disabled. The Voltage Reference is in off mode.

23.3.2   Voltage Reference Enabled, VREFEN=1

When VREFEN=1, the Voltage Reference is enabled, and different modes should be set by the Mode[1:0] bits.

23.3.2.1   Mode[1:0]=00

The internal bandgap is on to generate an accurate voltage output that can be trimmed by the bits TRM[7:0] in 0.5 mV steps. The bandgap requires some time for startup and stabilization. The VREFST bit can be monitored to determine if the stabilization and startup is complete.

Both low-power buffer and tight-regulation buffer are disabled in this mode, and there is no buffered voltage output. The Voltage Reference is in standby mode.

23.3.2.2   Mode[1:0]=01

The internal bandgap is on.

The low-power buffer is enabled to generate a buffered internal voltage. It can be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

23.3.2.3   Mode[1:0]=10

The internal bandgap is on.

The tight-regulation buffer is enabled to generate a buffered voltage to VREFO with load regulation less than 100 uV/mA. A 100nF capacitor is required on VREFO pin and it is allowed to drive 10 mA maximum current. VREFO can be used internally and/or externally.

23.3.2.4   Mode[1:0]=11

RESERVED

23.4   Initialization Information

The Voltage Reference requires some time for startup and stabilization. Once the VREFEN bit is set, the VREFST bit can be monitored to determine if the stabilization and startup is complete.

When the Voltage Reference is already enabled and stabilized, changing the Mode selection bits (MODE[1:0]) will not clear the VREFST bit, but there will be some startup time before the output voltage is stabilized when the low-power buffer or tight-regulation buffer is enabled, and there will be some setting time when a step change of the load current occurs.

Chapter 24
Development Support

24.1 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. 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.

24.1.1 Forcing Active Background

The method for forcing active background mode depends on the specific HCS08 derivative. For the MC9S08JE128 series, you can force active background after a power-on reset by holding the BKGD pin low as the device exits the reset condition. You can also force active background by driving BKGD low immediately after a serial background command that writes a one to the BDFR bit in the SBDFR register. Other causes of reset, including an external pin reset or an internally generated error reset ignore the state of the BKGD pin and reset into normal user mode. If no debug pod is connected to the BKGD pin, the MCU will always reset into normal operating mode.

24.1.2 Module Configuration

The alternate BDC clock source is the MCGLCLK. This clock source is selected by clearing the CLKSW bit in the BDCSCR register. For details on MCGLCLK, see the "Functional Description" section of the MCG chapter.

24.1.3 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

24.2   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.
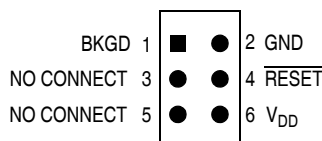

Figure 24-1. BDM Tool Connector

24.2.1   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 24.2.2, "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 24.2.2, "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.

24.2.2   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 24-2 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 24-2. BDC Host-to-Target Serial Bit Timing

Figure 24-3 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.
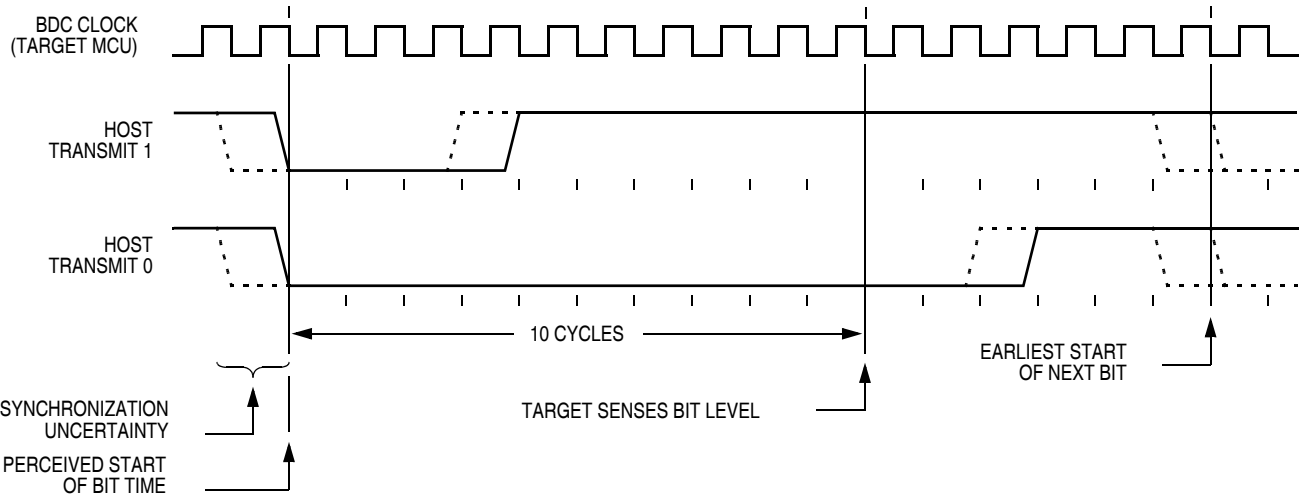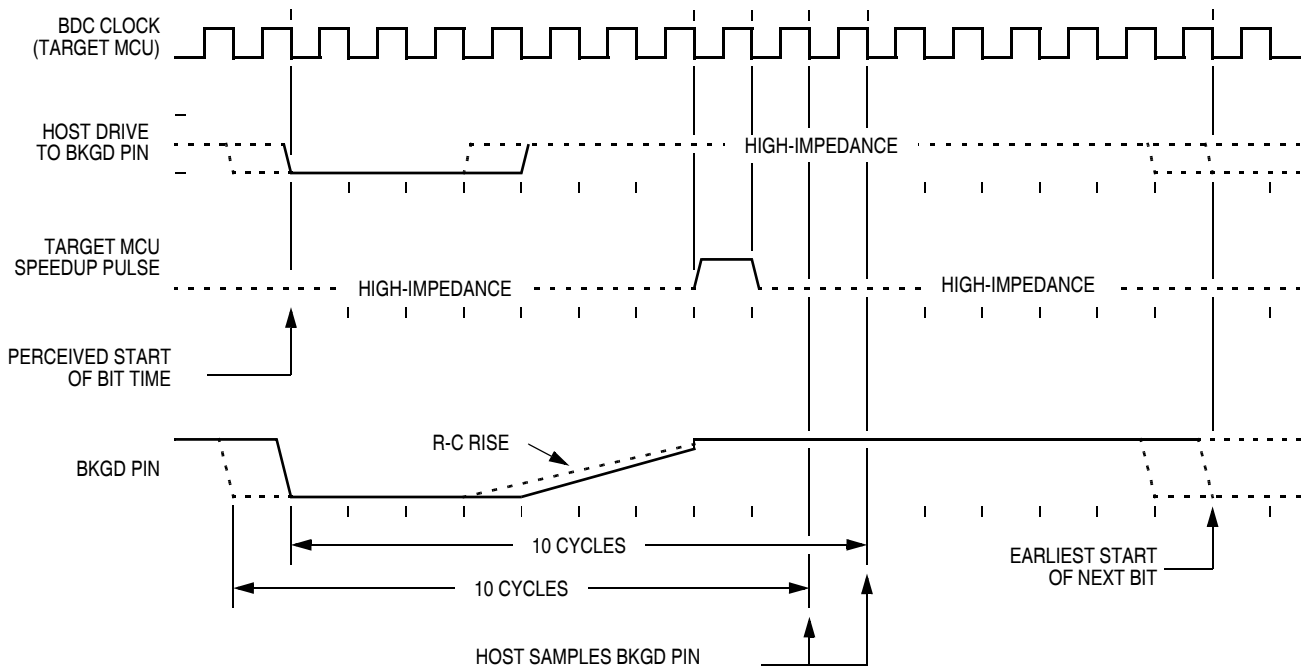

Figure 24-3. BDC Target-to-Host Serial Bit Timing (Logic 1)

Figure 24-4 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.
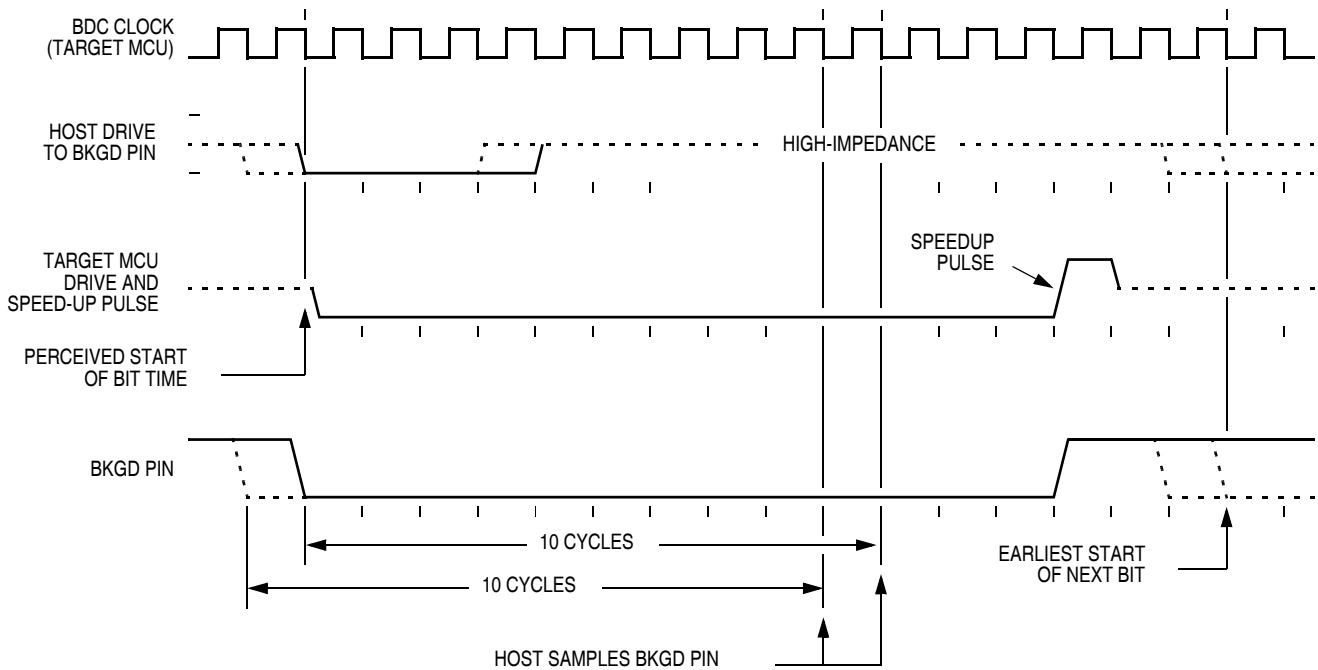

Figure 24-4. BDM Target-to-Host Serial Bit Timing (Logic 0)

24.2.3   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 24-1 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 24-1 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 24-1. 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.

24.2.4   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.

24.3   Register Definition

This section contains the descriptions of the BDC registers and control bits.

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.

24.3.1   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.

24.3.1.1 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 24-5. BDC Status and Control Register (BDCSCR)

Table 24-2. 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

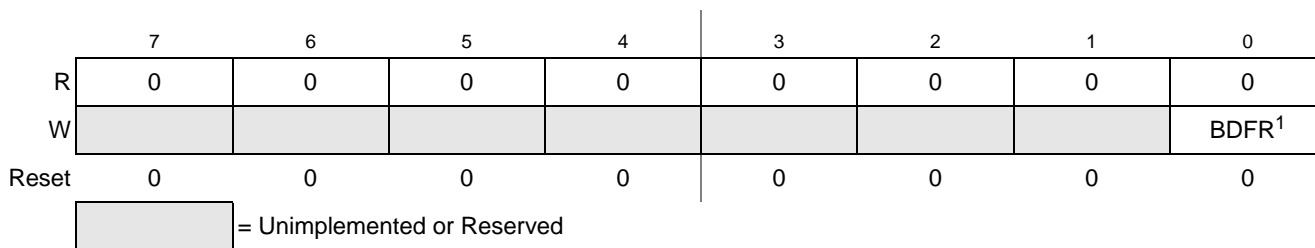**Table 24-2. BDCSCR Register Field Descriptions (Continued)**

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 MC9S08JE128 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

24.3.1.2 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 24.2.4, "BDC Hardware Breakpoint."

24.3.2 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 24-6. System Background Debug Force Reset Register (SBDFR)

Table 24-3. 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.

24.3.3 Debug Module

The DBG module implements an on-chip ICE (in-circuit emulation) system and allows non-intrusive debug of application software by providing an on-chip trace buffer with flexible triggering capability. The trigger also can provide extended breakpoint capacity. The on-chip ICE system is optimized for the HCS08 8-bit architecture and supports 64K bytes or 128K bytes of memory space.

24.3.4 Features

The on-chip ICE system includes these distinctive features:

- Three comparators (A, B, and C) with ability to match addresses in 128K space
  — Dual mode, Comparators A and B used to compare addresses
  — Full mode, Comparator A compares address and Comparator B compares data
  — Can be used as triggers and/or breakpoints
  — Comparator C can be used as a normal hardware breakpoint
  — Loop1 capture mode, Comparator C is used to track most recent COF event captured into FIFO
- Tag and Force type breakpoints
- Nine trigger modes
  — A
  — A Or B
  — A Then B
  — A And B, where B is data (Full mode)
  — A And Not B, where B is data (Full mode)
  — Event Only B, store data
  — A Then Event Only B, store data
  — Inside Range, A $\leq$ Address $\leq$ B
  — Outside Range, Address < A or Address > B
- FIFO for storing change of flow information and event only data
  — Source address of conditional branches taken
  — Destination address of indirect JMP and JSR instruction
 — Destination address of interrupts, RTI, RTC, and RTS instruction

— Data associated with Event B trigger modes

• Ability to End-trace until reset and Begin-trace from reset

24.3.5 Modes of Operation

The on-chip ICE system can be enabled in all MCU functional modes. The DBG module is disabled if the MCU is secure. The DBG module comparators are disabled when executing a Background Debug Mode (BDM) command.

24.3.6 Block Diagram

Figure 24-7 shows the structure of the DBG module.
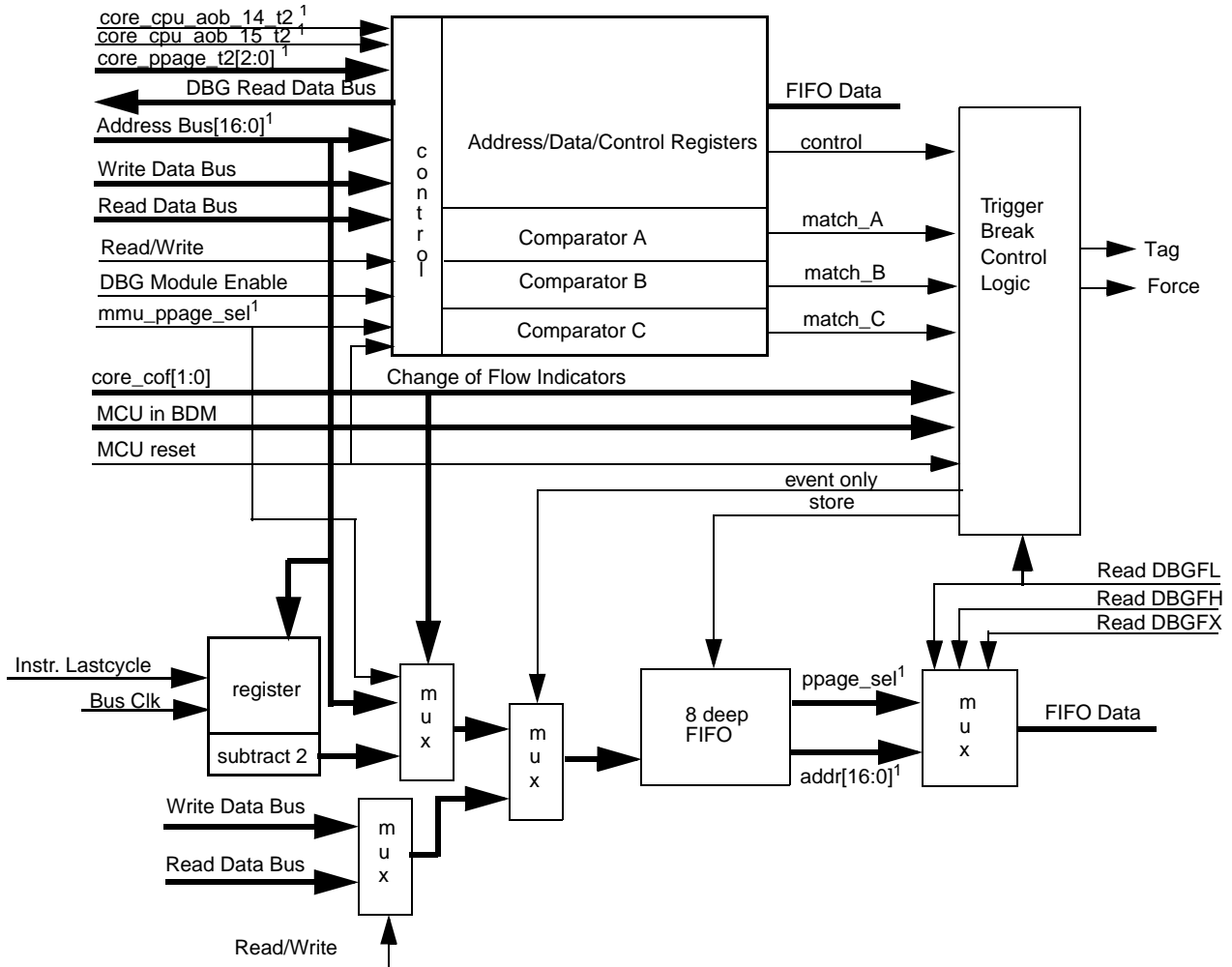
1. In 64K versions of this module there are only 16 address lines [15:0], there are no core_cpu_aob_14_t2, core_cpu_aob_15_t2, core_ppage_t2[2:0], and ppage_sel signals.

Figure 24-7. DBG Block Diagram

24.4   Signal Description

The DBG module contains no external signals.

24.5   Memory Map and Registers

This section provides a detailed description of all DBG registers accessible to the end user.

24.5.1   Module Memory Map

Table 24-4 shows the registers contained in the DBG module.

Table 24-4. Module Memory Map

Address	Use	Access
Base + $0000	Debug Comparator A High Register (DBGCAH)	Read/write
Base + $0001	Debug Comparator A Low Register (DBGCAL)	Read/write
Base + $0002	Debug Comparator B High Register (DBGCBH)	Read/write
Base + $0003	Debug Comparator B Low Register (DBGCBL)	Read/write
Base + $0004	Debug Comparator C High Register (DBGCCH)	Read/write
Base + $0005	Debug Comparator C Low Register (DBGCCL)	Read/write
Base + $0006	Debug FIFO High Register (DBGFH)	Read only
Base + $0007	Debug FIFO Low Register (DBGFL)	Read only
Base + $0008	Debug Comparator A Extension Register (DBGCAX)	Read/write
Base + $0009	Debug Comparator B Extension Register (DBGCBX)	Read/write
Base + $000A	Debug Comparator C Extension Register (DBGCCX)	Read/write
Base + $000B	Debug FIFO Extended Information Register (DBGFX)	Read only
Base + $000C	Debug Control Register (DBGC)	Read/write
Base + $000D	Debug Trigger Register (DBGT)	Read/write
Base + $000E	Debug Status Register (DBGS)	Read only
Base + $000F	Debug FIFO Count Register (DBGCNT)	Read only

Table 24-5. Register Bit Summary

	7	6	5	4	3	2	1	0
DBGCAH	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
DBGCAL	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBGCBH	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
DBGCBL	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBGCCH	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
DBGCCL	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Table 24-5. Register Bit Summary (Continued)

	7	6	5	4	3	2	1	0
DBGFH	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
DBGFL	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DBGCAX	RWAEN	RWA	PAGSEL	0	0	0	0	bit-16
DBGCBX	RWBEN	RWB	PAGSEL	0	0	0	0	bit-16
DBGCCX	RWCEN	RWC	PAGSEL	0	0	0	0	bit-16
DBGFX	PPACC	0	0	0	0	0	0	bit-16
DBGC	DBGEN	ARM	TAG	BRKEN	-	-	-	LOOP1
DBGT	TRGSEL	BEGIN	0	0	TRG[3:0]			
DBGS	AF	BF	CF	0	0	0	0	ARMF
DBGCNT	0	0	0	0	CNT[3:0]			

24.5.2   Register Descriptions

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

NOTE

For all registers below, consider: U = Unchanged, bit maintain its value after reset.

24.5.2.1   Debug Comparator A High Register (DBGCAH)

Module Base + 0x0000

	7	6	5	4	3	2	1	0
R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
W								
POR or non-end-run	1	1	1	1	1	1	1	1
Reset end-run[1]	U	U	U	U	U	U	U	U

Figure 24-8. Debug Comparator A High Register (DBGCAH)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-6. DBGCAH Field Descriptions

Field	Description
Bits 15–8	**Comparator A High Compare Bits** — The Comparator A High compare bits control whether Comparator A will compare the address bus bits [15:8] to a logic 1 or logic 0. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

24.5.2.2   Debug Comparator A Low Register (DBGCAL)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W								
POR or non-end-run	1	1	1	1	1	1	1	0
Reset end-run[1]	U	U	U	U	U	U	U	U

Figure 24-9. Debug Comparator A Low Register (DBGCAL)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-7. DBGCAL Field Descriptions

Field	Description
Bits 7–0	**Comparator A Low Compare Bits** — The Comparator A Low compare bits control whether Comparator A will compare the address bus bits [7:0] to a logic 1 or logic 0. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

24.5.2.3 Debug Comparator B High Register (DBGCBH)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R W	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run[1]	U	U	U	U	U	U	U	U

Figure 24-10. Debug Comparator B High Register (DBGCBH)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-8. DBGCBH Field Descriptions

Field	Description
Bits 15–8	**Comparator B High Compare Bits** — The Comparator B High compare bits control whether Comparator B will compare the address bus bits [15:8] to a logic 1 or logic 0. Not used in Full mode. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

24.5.2.4 Debug Comparator B Low Register (DBGCBL)

Module Base + 0x0003

	7	6	5	4	3	2	1	0
R W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run[1]	U	U	U	U	U	U	U	U

Figure 24-11. Debug Comparator B Low Register (DBGCBL)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-9. DBGCBL Field Descriptions

Field	Description
Bits 7–0	**Comparator B Low Compare Bits** — The Comparator B Low compare bits control whether Comparator B will compare the address bus or data bus bits [7:0] to a logic 1 or logic 0. 0　Compare corresponding address bit to a logic 0, compares to data if in Full mode 1　Compare corresponding address bit to a logic 1, compares to data if in Full mode

24.5.2.5　Debug Comparator C High Register (DBGCCH)

Module Base + 0x0004

	7	6	5	4	3	2	1	0
R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run[1]	U	U	U	U	U	U	U	U

Figure 24-12. Debug Comparator C High Register (DBGCCH)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-10. DBGCCH Field Descriptions

Field	Description
Bits 15–8	**Comparator C High Compare Bits** — The Comparator C High compare bits control whether Comparator C will compare the address bus bits [15:8] to a logic 1 or logic 0. 0　Compare corresponding address bit to a logic 0 1　Compare corresponding address bit to a logic 1

24.5.2.6　Debug Comparator C Low Register (DBGCCL)

Module Base + 0x0005

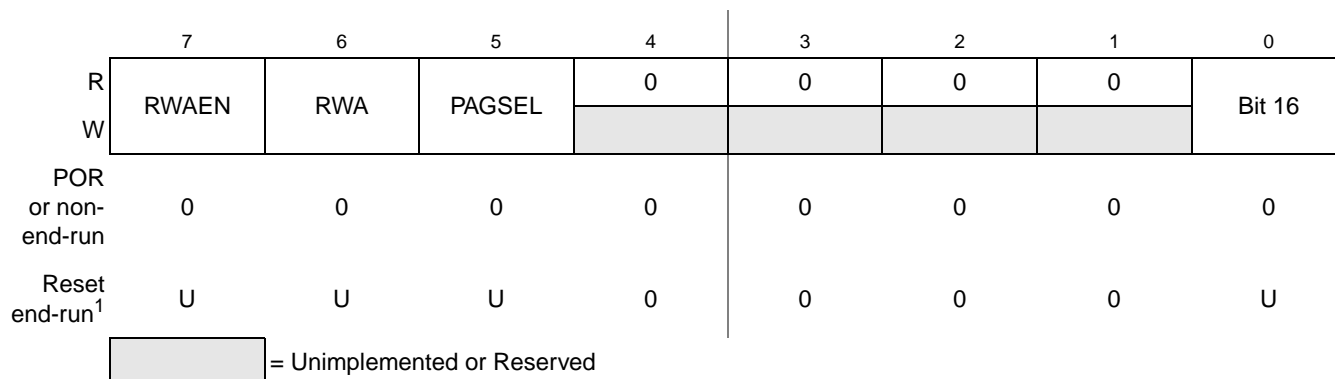|	7	6	5	4	3	2	1	0
R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run[1]	U	U	U	U	U	U	U	U

Figure 24-13. Debug Comparator C Low Register (DBGCCL)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-11. DBGCCL Field Descriptions

Field	Description
Bits 7–0	**Comparator C Low Compare Bits** — The Comparator C Low compare bits control whether Comparator C will compare the address bus bits [7:0] to a logic 1 or logic 0. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

24.5.2.7 Debug FIFO High Register (DBGFH)

Module Base + 0x0006

	7	6	5	4	3	2	1	0
R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run[1]	U	U	U	U	U	U	U	U

= Unimplemented or Reserved

Figure 24-14. Debug FIFO High Register (DBGFH)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-12. DBGFH Field Descriptions

Field	Description
Bits 15–8	**FIFO High Data Bits** — The FIFO High data bits provide access to bits [15:8] of data in the FIFO. This register is not used in event only modes and will read a $00 for valid FIFO words.

24.5.2.8 Debug FIFO Low Register (DBGFL)

Module Base + 0x0007

	7	6	5	4	3	2	1	0
R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run[1]	U	U	U	U	U	U	U	U

= Unimplemented or Reserved

Figure 24-15. Debug FIFO Low Register (DBGFL)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-13. DBGFL Field Descriptions

Field	Description
Bits 7–0	**FIFO Low Data Bits** — The FIFO Low data bits contain the least significant byte of data in the FIFO. When reading FIFO words, read DBGFX and DBGFH before reading DBGFL because reading DBGFL causes the FIFO pointers to advance to the next FIFO location. In event-only modes, there is no useful information in DBGFX and DBGFH so it is not necessary to read them before reading DBGFL.

24.5.2.9 Debug Comparator A Extension Register (DBGCAX)

Module Base + 0x0008

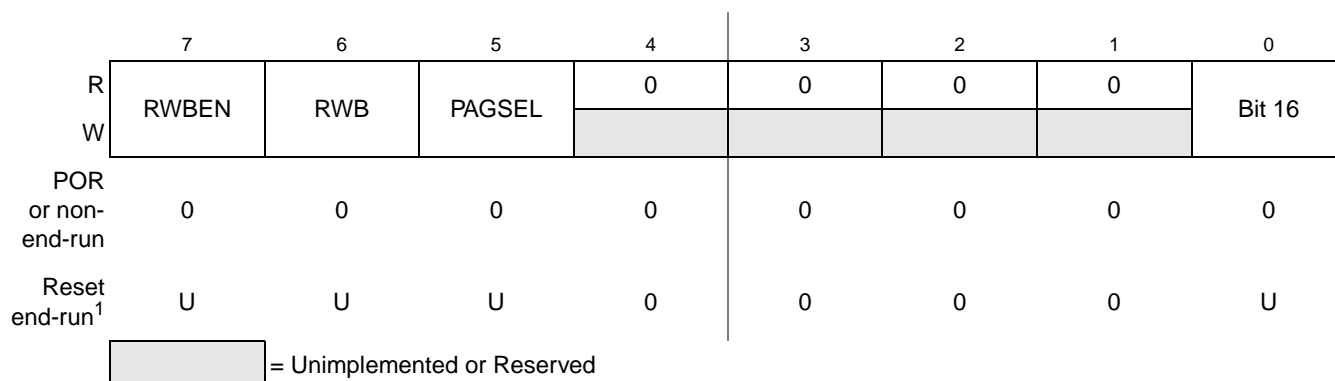|	7	6	5	4	3	2	1	0
R	RWAEN	RWA	PAGSEL	0	0	0	0	Bit 16
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run[1]	U	U	U	0	0	0	0	U

= Unimplemented or Reserved

Figure 24-16. Debug Comparator A Extension Register (DBGCAX)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-14. DBGCAX Field Descriptions

Field	Description
7 RWAEN	**Read/Write Comparator A Enable Bit** — The RWAEN bit controls whether read or write comparison is enabled for Comparator A. 0 Read/Write is not used in comparison 1 Read/Write is used in comparison
6 RWA	**Read/Write Comparator A Value Bit** — The RWA bit controls whether read or write is used in compare for Comparator A. The RWA bit is not used if RWAEN = 0. 0 Write cycle will be matched 1 Read cycle will be matched

Table 24-14. DBGCAX Field Descriptions (Continued)

Field	Description
5 PAGSEL	**Comparator A Page Select Bit** — This PAGSEL bit controls whether Comparator A will be qualified with the internal signal (mmu_ppage_sel) that indicates an extended access through the PPAGE mechanism. When mmu_ppage_sel = 1, the 17-bit core address is a paged program access, and the 17-bit core address is made up of PPAGE[2:0]:addr[13:0]. When mmu_ppage_sel = 0, the 17-bit core address is either a 16-bit CPU address with a leading 0 in bit 16, or a 17-bit linear address pointer value. 0 Match qualified by mmu_ppage_sel = 0 so address bits [16:0] correspond to a 17-bit CPU address with a leading zero at bit 16, or a 17-bit linear address pointer address 1 Match qualified by mmu_ppage_sel = 1 so address bits [16:0] compare to flash memory address made up of PPAGE[2:0]:addr[13:0]
0 Bit 16	**Comparator A Extended Address Bit 16 Compare Bit** — The Comparator A bit 16 compare bit controls whether Comparator A will compare the core address bus bit 16 to a logic 1 or logic 0. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

24.5.2.10 Debug Comparator B Extension Register (DBGCBX)

Module Base + 0x0009

	7	6	5	4	3	2	1	0
R	RWBEN	RWB	PAGSEL	0	0	0	0	Bit 16
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run[1]	U	U	U	0	0	0	0	U

	= Unimplemented or Reserved

Figure 24-17. Debug Comparator B Extension Register (DBGCBX)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-15. DBGCBX Field Descriptions

Field	Description
7 RWBEN	**Read/Write Comparator B Enable Bit** — The RWBEN bit controls whether read or write comparison is enabled for Comparator B. In full modes, RWAEN and RWA are used to control comparison of R/W and RWBEN is ignored. 0 Read/Write is not used in comparison 1 Read/Write is used in comparison
6 RWB	**Read/Write Comparator B Value Bit** — The RWB bit controls whether read or write is used in compare for Comparator B. The RWB bit is not used if RWBEN = 0. In full modes, RWAEN and RWA are used to control comparison of R/W and RWB is ignored. 0 Write cycle will be matched 1 Read cycle will be matched

Table 24-15. DBGCBX Field Descriptions (Continued)

Field	Description
5 PAGSEL	**Comparator B Page Select Bit** — This PAGSEL bit controls whether Comparator B will be qualified with the internal signal (mmu_papge_sel) that indicates an extended access through the PPAGE mechanism. When mmu_papge_sel = 1, the 17-bit core address is a paged program access, and the 17-bit core address is made up of PPAGE[2:0]:addr[13:0]. When mmu_papge_sel = 0, the 17-bit core address is either a 16-bit CPU address with a leading 0 in bit 16, or a 17-bit linear address pointer value. This bit is not used in full modes where comparator B is used to match the data value. 0 Match qualified by mmu_ppage_sel = 0 so address bits [16:0] correspond to a 17-bit CPU address with a leading zero at bit 16, or a 17-bit linear address pointer address 1 Match qualified by mmu_ppage_sel = 1 so address bits [16:0] compare to flash memory address made up of PPAGE[2:0]:addr[13:0]
0 Bit 16	**Comparator B Extended Address Bit 16 Compare Bit** — The Comparator B bit 16 compare bit controls whether Comparator B will compare the core address bus bit 16 to a logic 1 or logic 0. This bit is not used in full modes where comparator B is used to match the data value. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

24.5.2.11 Debug Comparator C Extension Register (DBGCCX)

Module Base + 0x000A

Figure 24-18. Debug Comparator C Extension Register (DBGCCX)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-16. DBGCCX Field Descriptions

Field	Description
7 RWCEN	**Read/Write Comparator C Enable Bit** — The RWCEN bit controls whether read or write comparison is enabled for Comparator C. 0 Read/Write is not used in comparison 1 Read/Write is used in comparison
6 RWC	**Read/Write Comparator C Value Bit** — The RWC bit controls whether read or write is used in compare for Comparator C. The RWC bit is not used if RWCEN = 0. 0 Write cycle will be matched 1 Read cycle will be matched

Table 24-16. DBGCCX Field Descriptions (Continued)

Field	Description
5 PAGSEL	**Comparator C Page Select Bit** — This PAGSEL bit controls whether Comparator C will be qualified with the internal signal (mmu_papge_sel) that indicates an extended access through the PPAGE mechanism. When mmu_ppage_sel = 1, the 17-bit core address is a paged program access, and the 17-bit core address is made up of PPAGE[2:0]:addr[13:0]. When mmu_papge_sel = 0, the 17-bit core address is either a 16-bit CPU address with a leading 0 in bit 16, or a 17-bit linear address pointer value. 0 Match qualified by mmu_ppage_sel = 0 so address bits [16:0] correspond to a 17-bit CPU address with a leading zero at bit 16, or a 17-bit linear address pointer address 1 Match qualified by mmu_ppage_sel = 1 so address bits [16:0] compare to flash memory address made up of PPAGE[2:0]:addr[13:0]
0 Bit 16	**Comparator C Extended Address Bit 16 Compare Bit** — The Comparator C bit 16 compare bit controls whether Comparator C will compare the core address bus bit 16 to a logic 1 or logic 0. 0 Compare corresponding address bit to a logic 0 1 Compare corresponding address bit to a logic 1

24.5.2.12 Debug FIFO Extended Information Register (DBGFX)

Module Base + 0x000B

	7	6	5	4	3	2	1	0
R	PPACC	0	0	0	0	0	0	Bit 16
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run[1]	U	0	0	0	0	0	0	U

= Unimplemented or Reserved

Figure 24-19. Debug FIFO Extended Information Register (DBGFX)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the bits in this register do not change after reset.

Table 24-17. DBGFX Field Descriptions

Field	Description
7 PPACC	**PPAGE Access Indicator Bit** — This bit indicates whether the captured information in the current FIFO word is associated with an extended access through the PPAGE mechanism or not. This is indicated by the internal signal mmu_ppage_sel which is 1 when the access is through the PPAGE mechanism. 0 The information in the corresponding FIFO word is event-only data or an unpaged 17-bit CPU address with bit-16 = 0 1 The information in the corresponding FIFO word is a 17-bit flash address with PPAGE[2:0] in the three most significant bits and CPU address[13:0] in the 14 least significant bits
0 Bit 16	**Extended Address Bit 16** — This bit is the most significant bit of the 17-bit core address.

24.5.2.13 Debug Control Register (DBGC)

Module Base + 0x000C

	7	6	5	4	3	2	1	0
R	DBGEN	ARM	TAG	BRKEN	0	0	0	LOOP1
W								
POR or non-end-run	1	1	0	0	0	0	0	0
Reset end-run[1]	U	0	U	0	0	0	0	U

= Unimplemented or Reserved

Figure 24-20. Debug Control Register (DBGC)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the ARM and BRKEN bits are cleared but the remaining control bits in this register do not change after reset.

Table 24-18. DBGC Field Descriptions

Field	Description
7 DBGEN	**DBG Module Enable Bit** — The DBGEN bit enables the DBG module. The DBGEN bit is forced to zero and cannot be set if the MCU is secure. 0 DBG not enabled 1 DBG enabled
6 ARM	**Arm Bit** — The ARM bit controls whether the debugger is comparing and storing data in FIFO. See Section 24.6.4.2, "Arming the DBG Module" for more information. 0 Debugger not armed 1 Debugger armed
5 TAG	**Tag or Force Bit** — The TAG bit controls whether a debugger or comparator C breakpoint will be requested as a tag or force breakpoint to the CPU. The TAG bit is not used if BRKEN = 0. 0 Force request selected 1 Tag request selected
4 BRKEN	**Break Enable Bit** — The BRKEN bit controls whether the debugger will request a breakpoint to the CPU at the end of a trace run, and whether comparator C will request a breakpoint to the CPU. 0 CPU break request not enabled 1 CPU break request enabled
0 LOOP1	**Select LOOP1 Capture Mode** — This bit selects either normal capture mode or LOOP1 capture mode. LOOP1 is not used in event-only modes. 0 Normal operation - capture COF events into the capture buffer FIFO 1 LOOP1 capture mode enabled. When the conditions are met to store a COF value into the FIFO, compare the current COF address with the address in comparator C. If these addresses match, override the FIFO capture and do not increment the FIFO count. If the address does not match comparator C, capture the COF address, including the PPACC indicator, into the FIFO and into comparator C.

24.5.2.14 Debug Trigger Register (DBGT)

Module Base + 0x000D

	7	6	5	4	3	2	1	0
R	TRGSEL	BEGIN	0	0			TRG	
W[2]								
POR or non-end-run	0	1	0	0	0	0	0	0
Reset end-run[1]	U	U	0	0	U	U	U	U

= Unimplemented or Reserved

Figure 24-21. Debug Trigger Register (DBGT)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the control bits in this register do not change after reset.

[2] The DBG trigger register (DBGT) can not be changed unless ARM=0.

Table 24-19. DBGT Field Descriptions

Field	Description
7 TRGSEL	**Trigger Selection Bit** — The TRGSEL bit controls the triggering condition for the comparators. See Section 24.6.4, "Trigger Break Control (TBC)" for more information. 0 Trigger on any compare address access 1 Trigger if opcode at compare address is executed
6 BEGIN	**Begin/End Trigger Bit** — The BEGIN bit controls whether the trigger begins or ends storing of data in FIFO. 0 Trigger at end of stored data 1 Trigger before storing data
3–0 TRG	**Trigger Mode Bits** — The TRG bits select the trigger mode of the DBG module as shown in Table 24-20.

Table 24-20. Trigger Mode Encoding

TRG Value	Meaning
0000	A Only
0001	A Or B
0010	A Then B
0011	Event Only B
0100	A Then Event Only B
0101	A And B (Full Mode)
0110	A And Not B (Full mode)
0111	Inside Range
1000	Outside Range

Table 24-20. Trigger Mode Encoding (Continued)

TRG Value	Meaning
1001 ↓ 1111	No Trigger

NOTE

The DBG trigger register (DBGT) can not be changed unless ARM=0.

24.5.2.15 Debug Status Register (DBGS)

Module Base + 0x000E

	7	6	5	4	3	2	1	0
R	AF	BF	CF	0	0	0	0	ARMF
W								
POR or non-end-run	0	0	0	0	0	0	0	1
Reset end-run[1]	U	U	U	0	0	0	0	0

= Unimplemented or Reserved

Figure 24-22. Debug Status Register (DBGS)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, ARMF gets cleared by reset but AF, BF, and CF do not change after reset.

Table 24-21. DBGS Field Descriptions

Field	Description
7 AF	**Trigger A Match Bit** — The AF bit indicates if Trigger A match condition was met since arming. 0 Comparator A did not match 1 Comparator A match
6 BF	**Trigger B Match Bit** — The BF bit indicates if Trigger B match condition was met since arming. 0 Comparator B did not match 1 Comparator B match
5 CF	**Trigger C Match Bit** — The CF bit indicates if Trigger C match condition was met since arming. 0 Comparator C did not match 1 Comparator C match
0 ARMF	**Arm Flag Bit** — The ARMF bit indicates whether the debugger is waiting for trigger or waiting for the FIFO to fill. While DBGEN = 1, this status bit is a read-only image of the ARM bit in DBGC. See Section 24.6.4.2, "Arming the DBG Module" for more information. 0 Debugger not armed 1 Debugger armed

24.5.2.16 Debug Count Status Register (DBGCNT)

Module Base + 0x000F

	7	6	5	4	3	2	1	0
R	0	0	0	0		CNT		
W								
POR or non-end-run	0	0	0	0	0	0	0	0
Reset end-run[1]	0	0	0	0	U	U	U	U

= Unimplemented or Reserved

Figure 24-23. Debug Count Status Register (DBGCNT)

[1] In the case of an end-trace to reset where DBGEN=1 and BEGIN=0, the CNT[3:0] bits do not change after reset.

Table 24-22. DBGS Field Descriptions

Field	Description
3–0 CNT	**FIFO Valid Count Bits** — The CNT bits indicate the amount of valid data stored in the FIFO. Table 24-23 shows the correlation between the CNT bits and the amount of valid data in FIFO. The CNT will stop after a count to eight even if more data is being stored in the FIFO. The CNT bits are cleared when the DBG module is armed, and the count is incremented each time a new word is captured into the FIFO. The host development system is responsible for checking the value in CNT[3:0] and reading the correct number of words from the FIFO because the count does not decrement as data is read out of the FIFO at the end of a trace run.

Table 24-23. CNT Bits

CNT Value	Meaning
0000	No data valid
0001	1 word valid
0010	2 words valid
0011	3 words valid
0100	4 words valid
0101	5 words valid
0110	6 words valid
0111	7 words valid
1000	8 words valid

24.6 Functional Description

This section provides a complete functional description of the on-chip ICE system. The DBG module is enabled by setting the DBGEN bit in the DBGC register. Enabling the module allows the arming, triggering and storing of data in the FIFO. The DBG module is made up of three main blocks, the Comparators, Trigger Break Control logic and the FIFO.

24.6.1 Comparator

The DBG module contains three comparators, A, B, and C. Comparator A compares the core address bus with the address stored in the DBGCAX, DBGCAH, and DBGCAL registers. Comparator B compares the core address bus with the address stored in the DBGCBX, DBGCBH, and DBGCBL registers except in full mode, where it compares the data buses to the data stored in the DBGCBL register. Comparator C compares the core address bus with the address stored in the DBGCCX, DBGCCH, and DBGCCL registers. Matches on Comparators A, B, and C are signaled to the Trigger Break Control (TBC) block.

24.6.1.1 RWA and RWAEN in Full Modes

In full modes ("A And B" and "A And Not B") RWAEN and RWA are used to select read or write comparisons for both comparators A and B. To select write comparisons and the write data bus in Full Modes set RWAEN=1 and RWA=0, otherwise read comparisons and the read data bus will be selected. The RWBEN and RWB bits are not used and will be ignored in Full Modes.

24.6.1.2 Comparator C in LOOP1 Capture Mode

Normally comparator C is used as a third hardware breakpoint and is not involved in the trigger logic for the on-chip ICE system. In this mode, it compares the core address bus with the address stored in the DBGCCX, DBGCCH, and DBGCCL registers. However, in LOOP1 capture mode, comparator C is managed by logic in the DBG module to track the address of the most recent change-of-flow event that was captured into the FIFO buffer. In LOOP1 capture mode, comparator C is not available for use as a normal hardware breakpoint.

When the ARM and DBGEN bits are set to one in LOOP1 capture mode, comparator C value registers are cleared to prevent the previous contents of these registers from interfering with the LOOP1 capture mode operation. When a COF event is detected, the address of the event is compared to the contents of the DBGCCX, DBGCCH, and DBGCCL registers to determine whether it is the same as the previous COF entry in the capture FIFO. If the values match, the capture is inhibited to prevent the FIFO from filling up with duplicate entries. If the values do not match, the COF event is captured into the FIFO and the DBGCCX, DBGCCH, and DBGCCL registers are updated to reflect the address of the captured COF event. When comparator C is updated, the PAGSEL bit (bit-7 of DBGCCX) is updated with the PPACC value that is captured into the FIFO. This bit indicates whether the COF address was a paged 17-bit program address using the PPAGE mechanism (PPACC=1) or a 17-bit CPU address that resulted from an unpaged CPU access.

24.6.2   Breakpoints

A breakpoint request to the CPU at the end of a trace run can be created if the BRKEN bit in the DBGC register is set. The value of the BEGIN bit in DBGT register determines when the breakpoint request to the CPU will occur. If the BEGIN bit is set, begin-trigger is selected and the breakpoint request will not occur until the FIFO is filled with 8 words. If the BEGIN bit is cleared, end-trigger is selected and the breakpoint request will occur immediately at the trigger cycle.

When traditional hardware breakpoints from comparators A or B are desired, set BEGIN=0 to select an end-trace run and set the trigger mode to either 0x0 (A-only) or 0x1 (A OR B) mode.

There are two types of breakpoint requests supported by the DBG module, tag-type and force-type. Tagged breakpoints are associated with opcode addresses and allow breaking just before a specific instruction executes. Force breakpoints are not associated with opcode addresses and allow breaking at the next instruction boundary. The TAG bit in the DBGC register determines whether CPU breakpoint requests will be a tag-type or force-type breakpoints. When TAG=0, a force-type breakpoint is requested and it will take effect at the next instruction boundary after the request. When TAG=1, a tag-type breakpoint is registered into the instruction queue and the CPU will break if/when this tag reaches the head of the instruction queue and the tagged instruction is about to be executed.

24.6.2.1   Hardware Breakpoints

Comparators A, B, and C can be used as three traditional hardware breakpoints whether the on-chip ICE real-time capture function is required or not. To use any breakpoint or trace run capture functions set DBGEN=1. BRKEN and TAG affect all three comparators. When BRKEN=0, no CPU breakpoints are enabled. When BRKEN=1, CPU breakpoints are enabled and the TAG bit determines whether the breakpoints will be tag-type or force-type breakpoints. To use comparators A and B as hardware breakpoints, set DBGT=0x81 for tag-type breakpoints and 0x01 for force-type breakpoints. This sets up an end-type trace with trigger mode "A OR B".

Comparator C is not involved in the trigger logic for the on-chip ICE system.

24.6.3   Trigger Selection

The TRGSEL bit in the DBGT register is used to determine the triggering condition of the on-chip ICE system. TRGSEL applies to both trigger A and B except in the event only trigger modes. By setting the TRGSEL bit, the comparators will qualify a match with the output of opcode tracking logic. The opcode tracking logic is internal to each comparator and determines whether the CPU executed the opcode at the compare address. With the TRGSEL bit cleared a comparator match is all that is necessary for a trigger condition to be met.

NOTE

If the TRGSEL is set, the address stored in the comparator match address registers must be an opcode address for the trigger to occur.

24.6.4   Trigger Break Control (TBC)

The TBC is the main controller for the DBG module. Its function is to decide whether data should be stored in the FIFO based on the trigger mode and the match signals from the comparator. The TBC also determines whether a request to break the CPU should occur.

The TAG bit in DBGC controls whether CPU breakpoints are treated as tag-type or force-type breakpoints. The TRGSEL bit in DBGT controls whether a comparator A or B match is further qualified by opcode tracking logic. Each comparator has a separate circuit to track opcodes because the comparators could correspond to separate instructions that could be propagating through the instruction queue at the same time.

In end-type trace runs (BEGIN=0), when the comparator registers match, including the optional R/W match, this signal goes to the CPU break logic where BRKEN determines whether a CPU break is requested and the TAG control bit determines whether the CPU break will be a tag-type or force-type breakpoint. When TRGSEL is set, the R/W qualified comparator match signal also passes through the opcode tracking logic. If/when it propagates through this logic, it will cause a trigger to the ICE logic to begin or end capturing information into the FIFO. In the case of an end-type (BEGIN=0) trace run, the qualified comparator signal stops the FIFO from capturing any more information.

If a CPU breakpoint is also enabled, you would want TAG and TRGSEL to agree so that the CPU break occurs at the same place in the application program as the FIFO stopped capturing information. If TRGSEL was 0 and TAG was 1 in an end-type trace run, the FIFO would stop capturing as soon as the comparator address matched, but the CPU would continue running until a TAG signal could propagate through the CPUs instruction queue which could take a long time in the case where changes of flow caused the instruction queue to be flushed. If TRGSEL was one and TAG was zero in an end-type trace run, the CPU would break before the comparator match signal could propagate through the opcode tracking logic to end the trace run.

In begin-type trace runs (BEGIN=1), the start of FIFO capturing is triggered by the qualified comparator signals, and the CPU breakpoint (if enabled by BRKEN=1) is triggered when the FIFO becomes full. Since this FIFO full condition does not correspond to the execution of a tagged instruction, it would not make sense to use TAG=1 for a begin-type trace run.

24.6.4.1   Begin- and End-Trigger

The definition of begin- and end-trigger as used in the DBG module are as follows:

- Begin-trigger: Storage in FIFO occurs after the trigger and continues until 8 locations are filled.
- End-trigger: Storage in FIFO occurs until the trigger with the least recent data falling out of the FIFO if more than 8 words are collected.

24.6.4.2   Arming the DBG Module

Arming occurs by enabling the DBG module by setting the DBGEN bit and by setting the ARM bit in the DBGC register. The ARM bit in the DBGC register and the ARMF bit in the DBGS register are cleared when the trigger condition is met in end-trigger mode or when the FIFO is filled in begin-trigger mode. In the case of an end-trace where DBGEN=1 and BEGIN=0, ARM and ARMF are cleared by any reset to

end the trace run that was in progress. The ARMF bit is also cleared if ARM is written to zero or when the DBGEN bit is low. The TBC logic determines whether a trigger condition has been met based on the trigger mode and the trigger selection.

24.6.4.3 Trigger Modes

The on-chip ICE system supports nine trigger modes. The trigger modes are encoded as shown in Table 24-20. The trigger mode is used as a qualifier for either starting or ending the storing of data in the FIFO. When the match condition is met, the appropriate flag AF or BF is set in DBGS register. Arming the DBG module clears the AF, BF, and CF flags in the DBGS register. In all trigger modes except for the event only modes change of flow addresses are stored in the FIFO. In the event only modes only the value on the data bus at the trigger event B comparator match address will be stored.

24.6.4.3.1 A Only

In the A Only trigger mode, if the match condition for A is met, the AF flag in the DBGS register is set.

24.6.4.3.2 A Or B

In the A Or B trigger mode, if the match condition for A or B is met, the corresponding flag(s) in the DBGS register are set.

24.6.4.3.3 A Then B

In the A Then B trigger mode, the match condition for A must be met before the match condition for B is compared. When the match condition for A or B is met, the corresponding flag in the DBGS register is set.

24.6.4.3.4 Event Only B

In the Event Only B trigger mode, if the match condition for B is met, the BF flag in the DBGS register is set. The Event Only B trigger mode is considered a begin-trigger type and the BEGIN bit in the DBGT register is ignored.

24.6.4.3.5 A Then Event Only B

In the A Then Event Only B trigger mode, the match condition for A must be met before the match condition for B is compared. When the match condition for A or B is met, the corresponding flag in the DBGS register is set. The A Then Event Only B trigger mode is considered a begin-trigger type and the BEGIN bit in the DBGT register is ignored.

24.6.4.3.6 A And B (Full Mode)

In the A And B trigger mode, Comparator A compares to the address bus and Comparator B compares to the data bus. In the A and B trigger mode, if the match condition for A and B happen on the same bus cycle, both the AF and BF flags in the DBGS register are set. If a match condition on only A or only B happens, no flags are set.

For Breakpoint tagging operation with an end-trigger type trace, only matches from Comparator A will be used to determine if the Breakpoint conditions are met and Comparator B matches will be ignored.

24.6.4.3.7 A And Not B (Full Mode)

In the A And Not B trigger mode, comparator A compares to the address bus and comparator B compares to the data bus. In the A And Not B trigger mode, if the match condition for A and Not B happen on the same bus cycle, both the AF and BF flags in the DBGS register are set. If a match condition on only A or only Not B occur no flags are set.

For Breakpoint tagging operation with an end-trigger type trace, only matches from Comparator A will be used to determine if the Breakpoint conditions are met and Comparator B matches will be ignored.

24.6.4.3.8 Inside Range, A $\leq$ address $\leq$ B

In the Inside Range trigger mode, if the match condition for A and B happen on the same bus cycle, both the AF and BF flags in the DBGS register are set. If a match condition on only A or only B occur no flags are set.

24.6.4.3.9 Outside Range, address < A or address > B

In the Outside Range trigger mode, if the match condition for A or B is met, the corresponding flag in the DBGS register is set.

The four control bits BEGIN and TRGSEL in DBGT, and BRKEN and TAG in DBGC, determine the basic type of debug run as shown in Table 1.21. Some of the 16 possible combinations are not used (refer to the notes at the end of the table).

Table 24-24. Basic Types of Debug Runs

BEGIN	TRGSEL	BRKEN	TAG	Type of Debug Run
0	0	0	x[1]	Fill FIFO until trigger address (No CPU breakpoint - keep running)
0	0	1	0	Fill FIFO until trigger address, then force CPU breakpoint
0	0	1	1	Do not use[2]
0	1	0	x[1]	Fill FIFO until trigger opcode about to execute (No CPU breakpoint - keep running)
0	1	1	0	Do not use[3]
0	1	1	1	Fill FIFO until trigger opcode about to execute (trigger causes CPU breakpoint)
1	0	0	x[1]	Start FIFO at trigger address (No CPU breakpoint - keep running)
1	0	1	0	Start FIFO at trigger address, force CPU breakpoint when FIFO full
1	0	1	1	Do not use[4]
1	1	0	x[1]	Start FIFO at trigger opcode (No CPU breakpoint - keep running)
1	1	1	0	Start FIFO at trigger opcode, force CPU breakpoint when FIFO full
1	1	1	1	Do not use[4]

[1] When BRKEN = 0, TAG is do not care (x in the table).

[2] In end trace configurations (BEGIN = 0) where a CPU breakpoint is enabled (BRKEN = 1), TRGSEL should agree with TAG. In this case, where TRGSEL = 0 to select no opcode tracking qualification and TAG = 1 to specify a tag-type CPU breakpoint, the CPU breakpoint would not take effect until sometime after the FIFO stopped storing values. Depending on program loops or interrupts, the delay could be very long.

[3] In end trace configurations (BEGIN = 0) where a CPU breakpoint is enabled (BRKEN = 1), TRGSEL should agree with TAG. In this case, where TRGSEL = 1 to select opcode tracking qualification and TAG = 0 to specify a force-type CPU breakpoint, the CPU breakpoint would erroneously take effect before the FIFO stopped storing values and the debug run would not complete normally.

4 In begin trace configurations (BEGIN = 1) where a CPU breakpoint is enabled (BRKEN = 1), TAG should not be set to 1. In begin trace debug runs, the CPU breakpoint corresponds to the FIFO full condition which does not correspond to a taggable instruction fetch.

24.6.5   FIFO

The FIFO is an eight word deep FIFO. In all trigger modes except for event only, the data stored in the FIFO will be change of flow addresses. In the event only trigger modes only the data bus value corresponding to the event is stored. In event only trigger modes, the high byte of the valid data from the FIFO will always read a 0x00 and the extended information byte in DBGFX will always read 0x00.

24.6.5.1   Storing Data in FIFO

In all trigger modes except for the event only modes, the address stored in the FIFO will be determined by the change of flow indicators from the core. The signal core_cof[1] indicates the current core address is the destination address of an indirect JSR or JMP instruction, or a RTS, RTC, or RTI instruction or interrupt vector and the destination address should be stored. The signal core_cof[0] indicates that a conditional branch was taken and that the source address of the conditional branch should be stored.

24.6.5.2   Storing with Begin-Trigger

Storing with Begin-Trigger can be used in all trigger modes. Once the DBG module is enabled and armed in the begin-trigger mode, data is not stored in the FIFO until the trigger condition is met. Once the trigger condition is met the DBG module will remain armed until 8 words are stored in the FIFO. If the core_cof[1] signal becomes asserted, the current address is stored in the FIFO. If the core_cof[0] signal becomes asserted, the address registered during the previous last cycle is decremented by two and stored in the FIFO.

24.6.5.3   Storing with End-Trigger

Storing with End-Trigger cannot be used in event-only trigger modes. Once the DBG module is enabled and armed in the end-trigger mode, data is stored in the FIFO until the trigger condition is met. If the core_cof[1] signal becomes asserted, the current address is stored in the FIFO. If the core_cof[0] signal becomes asserted, the address registered during the previous last cycle is decremented by two and stored in the FIFO. When the trigger condition is met, the ARM and ARMF will be cleared and no more data will be stored. In non-event only end-trigger modes, if the trigger is at a change of flow address the trigger event will be stored in the FIFO.

24.6.5.4   Reading Data from FIFO

The data stored in the FIFO can be read using BDM commands provided the DBG module is enabled and not armed (DBGEN=1 and ARM=0). The FIFO data is read out first-in-first-out. By reading the CNT bits

in the DBGCNT register at the end of a trace run, the number of valid words can be determined. The FIFO data is read by optionally reading the DBGFX and DBGFH registers followed by the DBGFL register. Each time the DBGFL register is read the FIFO is shifted to allow reading of the next word however the count does not decrement. In event-only trigger modes where the FIFO will contain only the data bus values stored, to read the FIFO only DBGFL needs to be accessed.

The FIFO is normally only read while ARM and ARMF=0, however reading the FIFO while the DBG module is armed will return the data value in the oldest location of the FIFO and the TBC will not allow the FIFO to shift. This action could cause a valid entry to be lost because the unexpected read blocked the FIFO advance.

If the DBG module is not armed and the DBGFL register is read, the TBC will store the current opcode address. Through periodic reads of the DBGFX, DBGFH, and DBGFL registers while the DBG module is not armed, host software can provide a histogram of program execution. This is called profile mode. Since the full 17-bit address and the signal that indicates whether an address is in paged extended memory are captured on each FIFO store, profile mode works correctly over the entire extended memory map.

24.6.6 Interrupt Priority

When TRGSEL is set and the DBG module is armed to trigger on begin- or end-trigger types, a trigger is not detected in the condition where a pending interrupt occurs at the same time that a target address reaches the top of the instruction pipe. In these conditions, the pending interrupt has higher priority and code execution switches to the interrupt service routine.

When TRGSEL is clear and the DBG module is armed to trigger on end-trigger types, the trigger event is detected on a program fetch of the target address, even when an interrupt becomes pending on the same cycle. In these conditions, the pending interrupt has higher priority, the exception is processed by the core and the interrupt vector is fetched. Code execution is halted before the first instruction of the interrupt service routine is executed. In this scenario, the DBG module will have cleared ARM without having recorded the change-of-flow that occurred as part of the interrupt exception. Note that the stack will hold the return addresses and can be used to reconstruct execution flow in this scenario.

When TRGSEL is clear and the DBG module is armed to trigger on begin-trigger types, the trigger event is detected on a program fetch of the target address, even when an interrupt becomes pending on the same cycle. In this scenario, the FIFO captures the change of flow event. Because the system is configured for begin-trigger, the DBG remains armed and does not break until the FIFO has been filled by subsequent change of flow events.

24.7 Resets

The DBG module cannot cause an MCU reset.

There are two different ways this module will respond to reset depending upon the conditions before the reset event. If the DBG module was setup for an end trace run with DBGEN=1 and BEGIN=0, ARM, ARMF, and BRKEN are cleared but the reset function on most DBG control and status bits is overridden so a host development system can read out the results of the trace run after the MCU has been reset. In all other cases including POR, the DBG module controls are initialized to start a begin trace run starting from when the reset vector is fetched. The conditions for the default begin trace run are:

- DBGCAX=0x00, DBGCAH=0xFF, DBGCAL=0xFE so comparator A is set to match when the 16-bit CPU address 0xFFFE appears during the reset vector fetch
- DBGC=0xC0 to enable and arm the DBG module
- DBGT=0x40 to select a force-type trigger, a BEGIN trigger, and A-only trigger mode

24.8 Interrupts

The DBG contains no interrupt source.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
http://compass.freescale.net/go/168063291
1-800-441-2447 or 1-303-675-2140
Fax: 1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MC9S08JE128RM
Rev. 3, 12/2010