

MC9S08JS16

MC9S08JS8

MC9S08JS16L

MC9S08JS8L

Reference Manual

HCS08 *Microcontrollers*

Related Documentation:

- **MC9S08JS16 (Data Sheet)**
Contains pin assignments and diagrams, all electrical specifications, and mechanical drawing outlines.

Find the most current versions of all documents at:
<http://www.freescale.com>

MC9S08JS16RM
Rev. 4
4/2009

[freescale.com](http://www.freescale.com)

MC9S08JS16 Features

8-Bit HCS08 Central Processor Unit (CPU)

- 48 MHz HCS08 CPU (central processor unit)
- 24 MHz internal bus frequency
- HC08 instruction set with added BGND instruction
- Support for up to 32 interrupt/reset sources

Memory Options

- Up to 16 KB of on-chip in-circuit programmable flash memory with block protection and security options
- Up to 512 bytes of on-chip RAM
- 256 bytes of USB RAM

Clock Source Options

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

System Protection

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

Power-Saving Modes

- Wait plus two stops

USB Bootload

- Mass erase entire flash array
- Partial erase flash array — erase all flash blocks except for the first 1 KB of flash
- Program flash

Peripherals

- **USB** — USB 2.0 full-speed (12 Mbps) with dedicated on-chip 3.3 V regulator and

transceiver; supports endpoint 0 and up to 6 additional endpoints

- **SPI** — One 8- or 16-bit selectable serial peripheral interface module with a receive data buffer hardware match function
- **SCI** — One serial communication interface module with optional 13-bit break. Full duplex non-return to zero (NRZ); LIN master extended break generation; LIN slave extended break detection; wakeup on active edge
- **MTIM** — One 8-bit modulo counter with 8-bit prescaler and overflow interrupt
- **TPM** — One 2-channel 16-bit timer/pulse-width modulator (TPM) module: selectable input capture, output compare, and edge-aligned PWM capability on each channel. Timer module may be configured for buffered, centered PWM (CPWM) on all channels
- **KBI** — 8-pin keyboard interrupt module
- **RTC** — Real-time counter with binary- or decimal-based prescaler
- **CRC** — Hardware CRC generator circuit using 16-bit shift register; CRC16-CCITT compliancy with $x^{16}+x^{12}+x^5+1$ polynomial

Input/Output

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

Package Options

- 24-pin quad flat no-lead (QFN)
- 20-pin small outline IC package (SOIC)

MC9S08JS16 MCU Series Reference Manual

Covers: MC9S08JS16
MC9S08JS8
MC9S08JS16L
MC9S08JS8L

MC9S08JS16RM
Rev. 4
4/2009

Revision History

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

<http://freescale.com>

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

Revision Number	Revision Date	Description of Changes
1	8/27/2008	Initial public release.
2	12/17/2008	Changed the content of register at address 0xFFAE and 0xFFAF in Table 4-4 and added the description of factory trim value before this table. Deleted duplicated information in KBI Features section. Changed the default of PTASE/PTBSE registers after reset to 0.
3	3/6/2009	Updated Figure 4-4 and Figure 4-5 .
4	4/24/2009	Added MC9S08JS16L and MC9S08JS8L information.

This product incorporates SuperFlash[®] technology licensed from SST.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc.
© Freescale Semiconductor, Inc., 2008-2009. All rights reserved.

List of Chapters

Chapter Number	Title	Page
Chapter 1	Device Overview	17
Chapter 2	Pins and Connections	21
Chapter 3	Modes of Operation	29
Chapter 4	Memory	35
Chapter 5	Resets, Interrupts, and System Configuration	63
Chapter 6	Parallel Input/Output	79
Chapter 7	Central Processor Unit (S08CPUV2)	87
Chapter 8	Keyboard Interrupt (S08KBIV2)	107
Chapter 9	Multi-Purpose Clock Generator (S08MCGV1)	115
Chapter 10	Modulo Timer (S08MTIMV1)	147
Chapter 11	Real-Time Counter (S08RTCV1)	157
Chapter 12	Serial Communications Interface (S08SCIV4)	167
Chapter 13	16-Bit Serial Peripheral Interface (S08SPI16V1)	187
Chapter 14	Timer/Pulse-Width Modulator (S08TPMV3)	215
Chapter 15	Universal Serial Bus Device Controller (S08USBV1)	243
Chapter 16	Cyclic Redundancy Check Generator (S08CRCV2)	275
Chapter 17	Development Support	283

Contents

Section Number	Title	Page
Chapter 1		
Device Overview		
1.1	Introduction	17
1.2	MCU Block Diagram	18
1.3	System Clock Distribution	19
Chapter 2		
Pins and Connections		
2.1	Introduction	21
2.2	Device Pin Assignment	21
2.3	Recommended System Connections	22
2.3.1	Power (V_{DD} , V_{SS} , V_{SSOSC} , V_{USB33})	24
2.3.2	Oscillator (XTAL, EXTAL)	24
2.3.3	RESET Pin	24
2.3.4	Background/Mode Select (BKGD/MS)	25
2.3.5	Bootloader Mode Select (BLMS)	26
2.3.6	USB Data Pins (USBDP, USBDN)	26
2.3.7	General-Purpose I/O and Peripheral Ports	26
Chapter 3		
Modes of Operation		
3.1	Introduction	29
3.2	Features	29
3.3	Run Mode	29
3.4	Active Background Mode	29
3.5	Wait Mode	30
3.6	Stop Modes	31
3.6.1	Stop3 Mode	31
3.6.2	Stop2 Mode	32
3.6.3	On-Chip Peripheral Modules in Stop Modes	33
Chapter 4		
Memory		
4.1	MC9S08JS16 Series Memory Map	35
4.1.1	Reset and Interrupt Vector Assignments	36
4.2	Register Addresses and Bit Assignments	37
4.3	RAM (System RAM)	43
4.4	USB RAM	44
4.5	Bootloader ROM	44

4.5.1	External Signal Description	44
4.5.2	Modes of Operation	45
4.5.3	Flash Memory Map	46
4.5.4	Bootloader Operation	47
4.6	Flash Memory	50
4.6.1	Features	50
4.6.2	Program and Erase Time	50
4.6.3	Program and Erase Command Execution	51
4.6.4	Burst Program Execution	52
4.6.5	Access Errors	54
4.6.6	Flash Block Protection	54
4.6.7	Flash Block Protection Disabled	55
4.6.8	Vector Redirection	55
4.7	Security	55
4.8	Flash Registers and Control Bits	57
4.8.1	Flash Clock Divider Register (FCDIV)	57
4.8.2	Flash Options Register (FOPT and NVOPT)	58
4.8.3	Flash Configuration Register (FCNFG)	59
4.8.4	Flash Protection Register (FPROT and NVPROT)	59
4.8.5	Flash Status Register (FSTAT)	60
4.8.6	Flash Command Register (FCMD)	61

Chapter 5 Resets, Interrupts, and System Configuration

5.1	Introduction	63
5.2	Features	63
5.3	MCU Reset	63
5.4	Computer Operating Properly (COP) Watchdog	64
5.5	Interrupts	65
5.5.1	Interrupt Stack Frame	66
5.5.2	External Interrupt Request (IRQ) Pin	66
5.5.3	Interrupt Vectors, Sources, and Local Masks	67
5.6	Low-Voltage Detect (LVD) System	69
5.6.1	Power-On Reset Operation	69
5.6.2	Low-Voltage Detection (LVD) Reset Operation	69
5.6.3	Low-Voltage Warning (LVW) Interrupt Operation	69
5.7	Reset, Interrupt, and System Control Registers and Control Bits	70
5.7.1	Interrupt Pin Request Status and Control Register (IRQSC)	70
5.7.2	System Reset Status Register (SRS)	71
5.7.3	System Background Debug Force Reset Register (SBDFR)	72
5.7.4	System Options Register 1 (SOPT1)	73
5.7.5	System Options Register 2 (SOPT2)	74
5.7.6	Flash Protection Defeat Register (FPROTD)	75
5.7.7	SIGNATURE Register (SIGNATURE)	75
5.7.8	System Device Identification Register (SDIDH, SDIDL)	76

5.7.9	System Power Management Status and Control 1 Register (SPMSC1)	77
5.7.10	System Power Management Status and Control 2 Register (SPMSC2)	78

Chapter 6 Parallel Input/Output

6.1	Introduction	79
6.2	Port Data and Data Direction	79
6.3	Pin Control	80
6.3.1	Internal Pullup Enable	81
6.3.2	Output Slew Rate Control Enable	81
6.3.3	Output Drive Strength Select	81
6.4	Pin Behavior in Stop Modes	81
6.5	Parallel I/O and Pin Control Registers	81
6.5.1	Port A I/O Registers (PTAD and PTADD)	82
6.5.2	Port A Pin Control Registers (PTAPE, PTASE, PTADS)	82
6.5.3	Port B I/O Registers (PTBD and PTBDD)	84
6.5.4	Port B Pin Control Registers (PTBPE, PTBSE, PTBDS)	84

Chapter 7 Central Processor Unit (S08CPUV2)

7.1	Introduction	87
7.1.1	Features	87
7.2	Programmer's Model and CPU Registers	88
7.2.1	Accumulator (A)	88
7.2.2	Index Register (H:X)	88
7.2.3	Stack Pointer (SP)	89
7.2.4	Program Counter (PC)	89
7.2.5	Condition Code Register (CCR)	89
7.3	Addressing Modes	91
7.3.1	Inherent Addressing Mode (INH)	91
7.3.2	Relative Addressing Mode (REL)	91
7.3.3	Immediate Addressing Mode (IMM)	91
7.3.4	Direct Addressing Mode (DIR)	91
7.3.5	Extended Addressing Mode (EXT)	92
7.3.6	Indexed Addressing Mode	92
7.4	Special Operations	93
7.4.1	Reset Sequence	93
7.4.2	Interrupt Sequence	93
7.4.3	Wait Mode Operation	94
7.4.4	Stop Mode Operation	94
7.4.5	BGND Instruction	95
7.5	HCS08 Instruction Set Summary	96

Chapter 8 Keyboard Interrupt (S08KBIV2)

8.1	Introduction	107
8.1.1	Features	109
8.1.2	Modes of Operation	109
8.1.3	Block Diagram	109
8.2	External Signal Description	110
8.3	Register Definition	110
8.3.1	KBI Status and Control Register (KBISC)	110
8.3.2	KBI Pin Enable Register (KBIPE)	111
8.3.3	KBI Edge Select Register (KBIES)	111
8.4	Functional Description	112
8.4.1	Edge Only Sensitivity	112
8.4.2	Edge and Level Sensitivity	112
8.4.3	KBI Pullup/Pulldown Resistors	113
8.4.4	KBI Initialization	113

Chapter 9 Multi-Purpose Clock Generator (S08MCGV1)

9.1	Introduction	115
9.1.1	Features	117
9.1.2	Modes of Operation	119
9.2	External Signal Description	119
9.3	Register Definition	120
9.3.1	MCG Control Register 1 (MCGC1)	120
9.3.2	MCG Control Register 2 (MCGC2)	121
9.3.3	MCG Trim Register (MCGTRM)	122
9.3.4	MCG Status and Control Register (MCGSC)	123
9.3.5	MCG Control Register 3 (MCGC3)	124
9.4	Functional Description	126
9.4.1	Operational Modes	126
9.4.2	Mode Switching	130
9.4.3	Bus Frequency Divider	131
9.4.4	Low Power Bit Usage	131
9.4.5	Internal Reference Clock	131
9.4.6	External Reference Clock	131
9.4.7	Fixed Frequency Clock	132
9.5	Initialization / Application Information	132
9.5.1	MCG Module Initialization Sequence	132
9.5.2	MCG Mode Switching	133
9.5.3	Calibrating the Internal Reference Clock (IRC)	144

Chapter 10 Modulo Timer (S08MTIMV1)

10.1	Introduction	147
------	--------------------	-----

10.1.1	MTIM Configuration Information	147
10.1.2	Features	149
10.1.3	Modes of Operation	149
10.1.4	Block Diagram	150
10.2	External Signal Description	150
10.3	Register Definition	150
10.3.1	MTIM Status and Control Register (MTIMSC)	152
10.3.2	MTIM Clock Configuration Register (MTIMCLK)	153
10.3.3	MTIM Counter Register (MTIMCNT)	154
10.3.4	MTIM Modulo Register (MTIMMOD)	154
10.4	Functional Description	155
10.4.1	MTIM Operation Example	156

Chapter 11 Real-Time Counter (S08RTCV1)

11.1	Introduction	157
11.1.1	Features	159
11.1.2	Modes of Operation	159
11.1.3	Block Diagram	160
11.2	External Signal Description	160
11.3	Register Definition	160
11.3.1	RTC Status and Control Register (RTCSC)	161
11.3.2	RTC Counter Register (RTCCNT)	162
11.3.3	RTC Modulo Register (RTCMOD)	162
11.4	Functional Description	162
11.4.1	RTC Operation Example	163
11.5	Initialization/Application Information	164

Chapter 12 Serial Communications Interface (S08SCIV4)

12.1	Introduction	167
12.1.1	Features	169
12.1.2	Modes of Operation	169
12.1.3	Block Diagram	169
12.2	Register Definition	172
12.2.1	SCI Baud Rate Registers (SCIBDH, SCIBDL)	172
12.2.2	SCI Control Register 1 (SCIC1)	173
12.2.3	SCI Control Register 2 (SCIC2)	174
12.2.4	SCI Status Register 1 (SCIS1)	175
12.2.5	SCI Status Register 2 (SCIS2)	177
12.2.6	SCI Control Register 3 (SCIC3)	178
12.2.7	SCI Data Register (SCID)	179
12.3	Functional Description	179
12.3.1	Baud Rate Generation	179
12.3.2	Transmitter Functional Description	180

12.3.3 Receiver Functional Description	181
12.3.4 Interrupts and Status Flags	183
12.3.5 Additional SCI Functions	184

Chapter 13 16-Bit Serial Peripheral Interface (S08SPI16V1)

13.1 Introduction	187
13.1.1 SPI Port Configuration Information	187
13.1.2 Features	190
13.1.3 Modes of Operation	190
13.1.4 Block Diagrams	190
13.2 External Signal Description	192
13.2.1 SPCK — SPI Serial Clock	193
13.2.2 MOSI — Master Data Out, Slave Data In	193
13.2.3 MISO — Master Data In, Slave Data Out	193
13.2.4 \overline{SS} — Slave Select	193
13.3 Register Definition	193
13.3.1 SPI Control Register 1 (SPIC1)	193
13.3.2 SPI Control Register 2 (SPIC2)	195
13.3.3 SPI Baud Rate Register (SPIBR)	196
13.3.4 SPI Status Register (SPIS)	197
13.3.5 SPI Data Registers (SPIDH:SPIDL)	198
13.3.6 SPI Match Registers (SPIMH:SPIML)	199
13.4 Functional Description	199
13.4.1 General	199
13.4.2 Master Mode	200
13.4.3 Slave Mode	201
13.4.4 Data Transmission Length	202
13.4.5 SPI Clock Formats	203
13.4.6 SPI Baud Rate Generation	205
13.4.7 Special Features	205
13.4.8 Error Conditions	207
13.4.9 Low Power Mode Options	207
13.4.10 SPI Interrupts	209
13.5 Initialization/Application Information	210
13.5.1 SPI Module Initialization Example	210

Chapter 14 Timer/Pulse-Width Modulator (S08TPMV3)

14.1 Introduction	215
14.2 Features	215
14.3 TPMV3 Differences from Previous Versions	217
14.3.1 Migrating from TPMV1	219
14.3.2 Features	220
14.3.3 Modes of Operation	220

14.3.4	Block Diagram	221
14.4	Signal Description	223
14.4.1	Detailed Signal Descriptions	223
14.5	Register Definition	227
14.5.1	TPM Status and Control Register (TPMSC)	227
14.5.2	TPM-Counter Registers (TPMCNTH:TPMCNTL)	228
14.5.3	TPM Counter Modulo Registers (TPMMODH:TPMMODL)	229
14.5.4	TPM Channel n Status and Control Register (TPMCnSC)	230
14.5.5	TPM Channel Value Registers (TPMCnVH:TPMCnVL)	231
14.6	Functional Description	233
14.6.1	Counter	233
14.6.2	Channel Mode Selection	235
14.7	Reset Overview	238
14.7.1	General	238
14.7.2	Description of Reset Operation	238
14.8	Interrupts	238
14.8.1	General	238
14.8.2	Description of Interrupt Operation	239

Chapter 15

Universal Serial Bus Device Controller (S08USBV1)

15.1	Introduction	243
15.1.1	Clocking Requirements	243
15.1.2	Current Consumption in USB Suspend	243
15.1.3	3.3 V Regulator	243
15.1.4	Features	246
15.1.5	Modes of Operation	246
15.1.6	Block Diagram	247
15.2	External Signal Description	248
15.2.1	USBDP	248
15.2.2	USBDN	248
15.2.3	V _{USB33}	248
15.3	Register Definition	248
15.3.1	USB Control Register 0 (USBCTL0)	249
15.3.2	Peripheral ID Register (PERID)	249
15.3.3	Peripheral ID Complement Register (IDCOMP)	250
15.3.4	Peripheral Revision Register (REV)	250
15.3.5	Interrupt Status Register (INTSTAT)	251
15.3.6	Interrupt Enable Register (INTENB)	252
15.3.7	Error Interrupt Status Register (ERRSTAT)	253
15.3.8	Error Interrupt Enable Register (ERRENB)	254
15.3.9	Status Register (STAT)	255
15.3.10	Control Register (CTL)	256
15.3.11	Address Register (ADDR)	257
15.3.12	Frame Number Register (FRMNUML, FRMNUMH)	257

15.3.13	Endpoint Control Register (EPCTLn, n=0-6)	258
15.4	Functional Description	259
15.4.1	Block Descriptions	259
15.4.2	Buffer Descriptor Table (BDT)	264
15.4.3	USB Transactions	267
15.4.4	USB Packet Processing	269
15.4.5	Start of Frame Processing	270
15.4.6	Suspend/Resume	271
15.4.7	Resets	272
15.4.8	Interrupts	273

Chapter 16 Cyclic Redundancy Check Generator (S08CRCV2)

16.1	Introduction	275
16.1.1	Features	277
16.1.2	Modes of Operation	277
16.1.3	Block Diagram	278
16.2	External Signal Description	278
16.3	Register Definition	278
16.3.1	Memory Map	278
16.3.2	Register Descriptions	279
16.4	Functional Description	280
16.4.1	ITU-T (CCITT) Recommendations & Expected CRC Results	280
16.5	Initialization Information	281

Chapter 17 Development Support

17.1	Introduction	283
17.1.1	Features	284
17.2	Background Debug Controller (BDC)	284
17.2.1	BKGD Pin Description	285
17.2.2	Communication Details	286
17.2.3	BDC Commands	289
17.2.4	BDC Hardware Breakpoint	292
17.3	On-Chip Debug System (DBG)	293
17.3.1	Comparators A and B	293
17.3.2	Bus Capture Information and FIFO Operation	293
17.3.3	Change-of-Flow Information	294
17.3.4	Tag vs. Force Breakpoints and Triggers	294
17.3.5	Trigger Modes	295
17.3.6	Hardware Breakpoints	297
17.4	Register Definition	297
17.4.1	BDC Registers and Control Bits	297
17.4.2	System Background Debug Force Reset Register (SBDFR)	299
17.4.3	DBG Registers and Control Bits	300

Chapter 1

Device Overview

1.1 Introduction

MC9S08JS16 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.

NOTE

The only difference between MC9S08JS16/MC9S08JS8 and MC9S08JS16L/MC9S08JS8L is that MC9S08JS16 and MC9S08JS8 support USB bootloader function with voltage above 3.9 V while MC9S08JS16L and MC9S08JS8L support USB bootloader function at 3.3 V.

Disable internal USB voltage regulator and apply 3.3 V to the V_{USB33} pin when using MC9S08JS16L and MC9S08JS8L for the USB bootloader function.

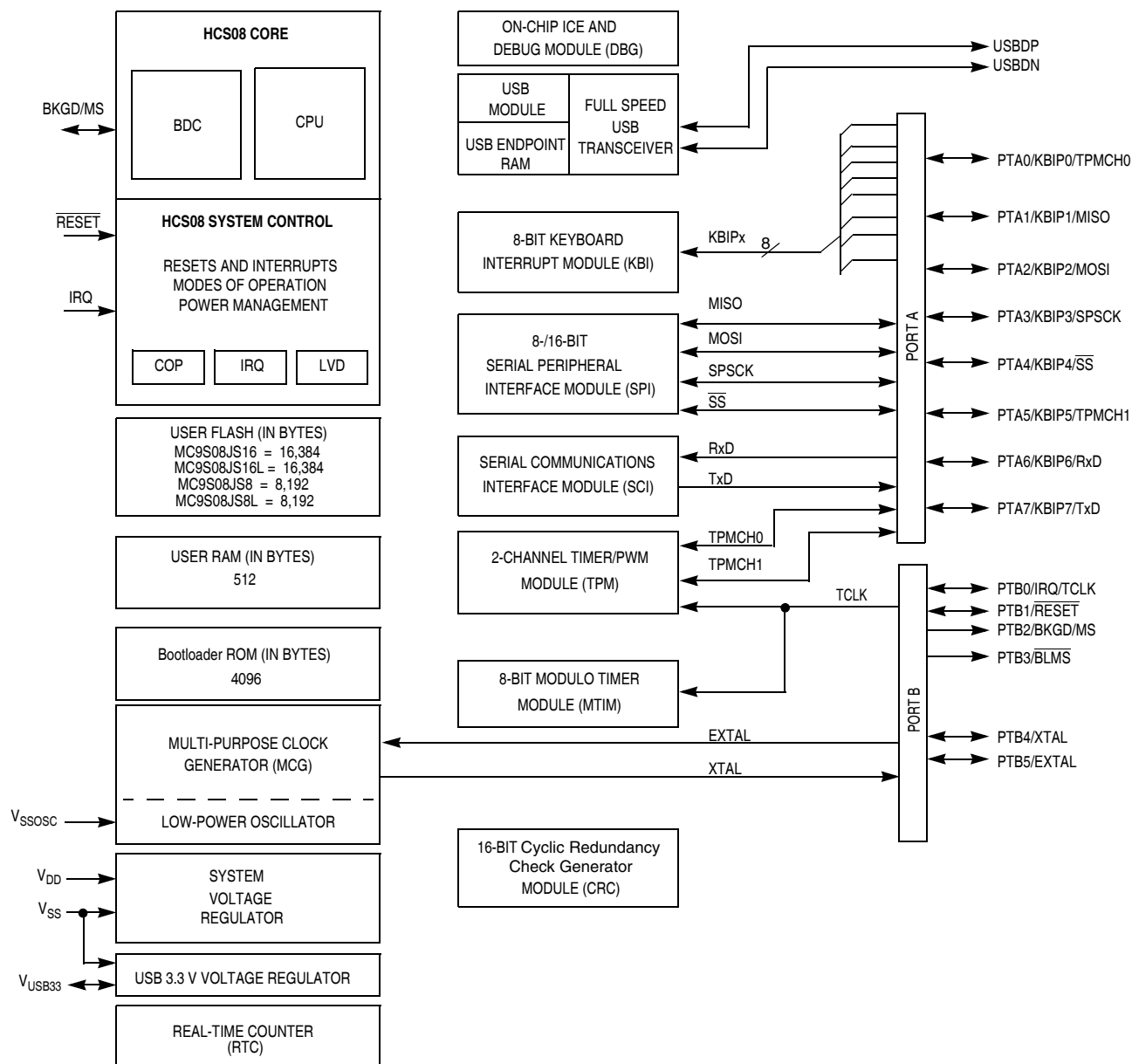
Table 1-1 summarizes the peripheral availability per package type for the devices available in the MC9S08JS16 series.

Table 1-1. MC9S08JS16 Series Features by MCU and Package

Feature	MC9S08JS8/MC9S08JS8L		MC9S08JS16/MC9S08JS16L	
	24-pin QFN	20-pin SOIC	24-pin QFN	20-pin SOIC
Flash size (bytes)	8,192		16,384	
RAM size (bytes)	512		512	
USB RAM (bytes)	256		256	
IRQ	yes		yes	
KBI	8		8	
SCI	yes		yes	
SPI	yes		yes	
MTIM	yes		yes	
TPM channels	2		2	
USB	yes		yes	
CRC	yes		yes	
I/O pins	14 (2 output only)		14 (2 output only)	

1.2 MCU Block Diagram

The block diagram in Figure 1-1 shows the structure of the MC9S08JS16 series MCU.



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1). Pulldown is enabled if rising edge detect is selected (IRQEDG = 1).
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. RESET contains integrated pullup device if PTB1 enabled as reset pin function (RSTPE = 1).
5. Pin contains integrated pullup device.
6. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 1-1. MC9S08JS16 Series Block Diagram

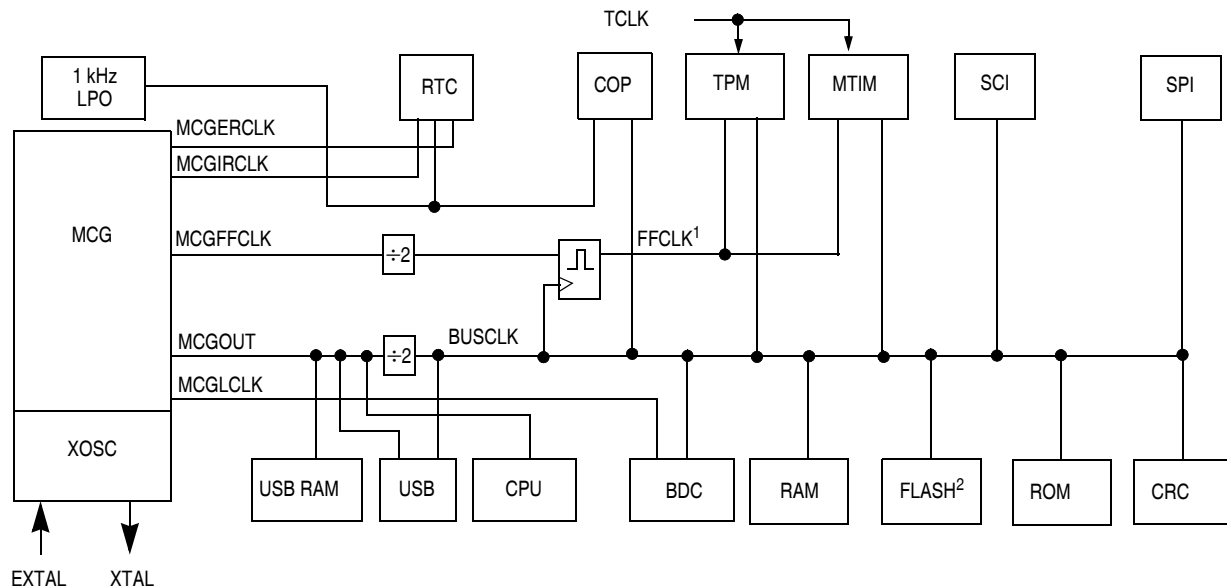
Table 1-2 lists the functional versions of the on-chip modules.

Table 1-2. Versions of On-Chip Modules

Module	Version
Central Processing Unit (CPU)	2
Keyboard Interrupt (KBI)	2
Multi-Purpose Clock Generator (MCG)	1
Real-Time Counter (RTC)	1
Serial Communications Interface (SCI)	4
Serial Peripheral Interface (SPI16)	1
Modulo Timer (MTIM)	1
Timer Pulse-Width Modulator (TPM)	3
Universal Serial Bus (USB)	1
Cyclic Redundancy Check Generator (CRC)	2
Debug Module (DBG)	2

1.3 System Clock Distribution

TCLK — External input clock source for TPM and MTIM and is referenced as TPMCLK in [Chapter 14](#), “[Timer/Pulse-Width Modulator \(S08TPMV3\)](#).”



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

² Flash and EEPROM have frequency requirements for program and erase operation. See MC9S08JS16 Series *Data Sheet* for details.

Figure 1-2. System Clock Distribution Diagram

The MCG supplies the following clock sources:

- MCGOUT — This clock source is used as the CPU, USB RAM and USB module clock, and is divided by two to generate the peripheral bus clock (BUSCLK). Control bits in the MCG control registers determine which of the three clock sources is connected:
 - Internal reference clock
 - External reference clock
 - Frequency-locked loop (FLL) or phase-locked loop (PLL) output
 See [Chapter 9, “Multi-Purpose Clock Generator \(S08MCGV1\)”](#), for details on configuring the MCGOUT clock.
- MCGLCLK — This clock source is derived from the digitally controlled oscillator (DCO) of the MCG. Development tools can select this internal self-clocked source to speed up BDC communications in systems where the bus clock is slow.
- MCGIRCLK — This is the internal reference clock and can be selected as the real-time counter (RTC) clock source. [Chapter 9, “Multi-Purpose Clock Generator \(S08MCGV1\)”](#), explains the MCGIRCLK in more detail. See [Chapter 11, “Real-Time Counter \(S08RTCv1\)”](#), for more information regarding the use of MCGIRCLK.
- MCGERCLK — This is the external reference clock and can be selected as the clock source of RTC module. [Section 9.4.6, “External Reference Clock,”](#) explains the MCGERCLK in more detail. See [Chapter 11, “Real-Time Counter \(S08RTCv1\)”](#), for more information regarding the use of MCGERCLK with this module.
- MCGFFCLK — This clock source is divided by two to generate FFCLK after being synchronized to the BUSCLK. It can be selected as clock source for the TPM or MTIM modules. The frequency of the MCGFFCLK is determined by the settings of the MCG. See [Section 9.4.7, “Fixed Frequency Clock,”](#) for details.
- LPO clock — This clock is generated from an internal low power oscillator that is completely independent of the MCG module. The LPO clock can be selected as the clock source to the RTC or COP modules. See [Chapter 11, “Real-Time Counter \(S08RTCv1\)”](#), and [Section 5.4, “Computer Operating Properly \(COP\) Watchdog,”](#) for details on using the LPO clock with these modules.
- TCLK — TCLK is the optional external clock source for the TPM or MTIM modules. The TCLK must be limited to 1/4th the frequency of the BUSCLK for synchronization. See [Chapter 14, “Timer/Pulse-Width Modulator \(S08TPMV3\)”](#), for more details.

Chapter 2 Pins and Connections

2.1 Introduction

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

2.2 Device Pin Assignment

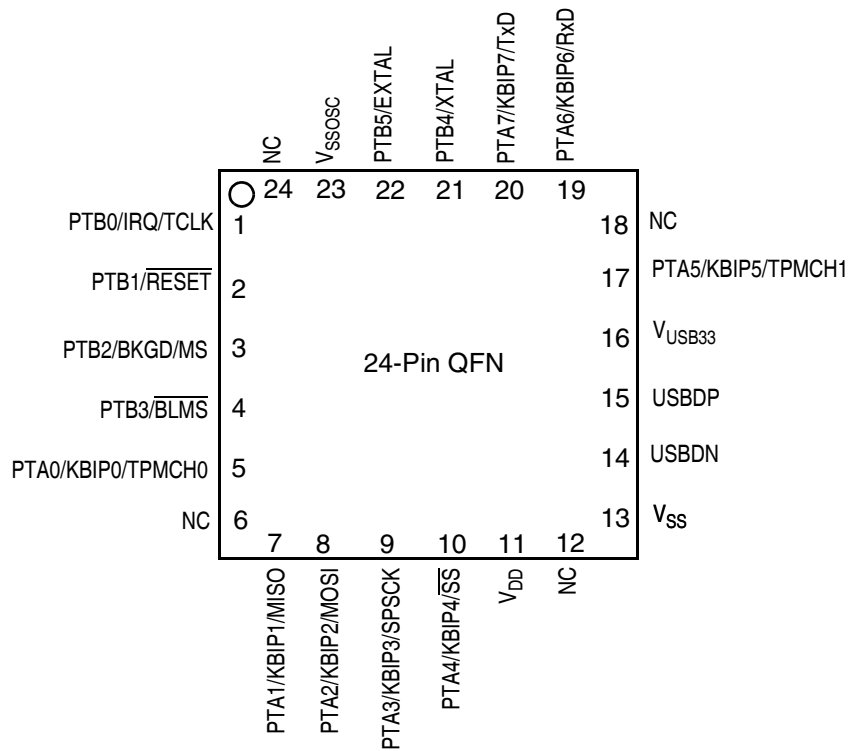


Figure 2-1. MC9S08JS16 Series in 24-pin QFN Package

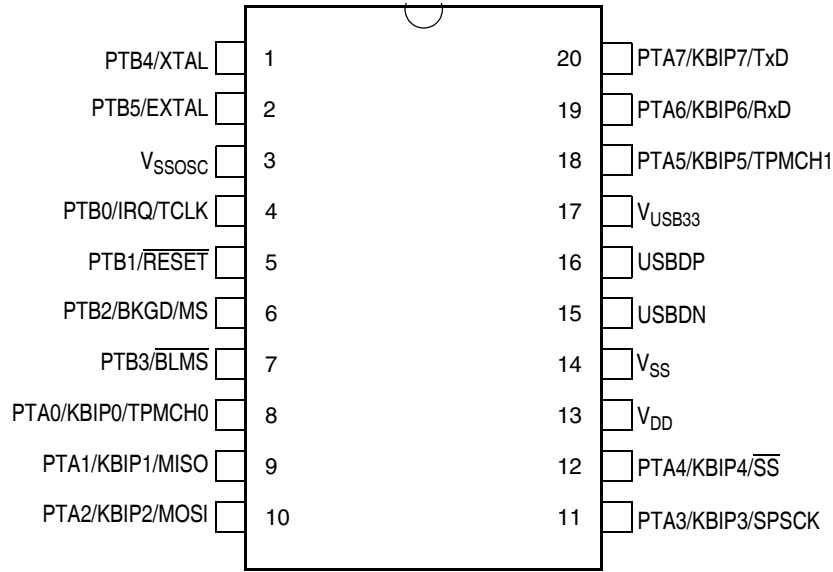
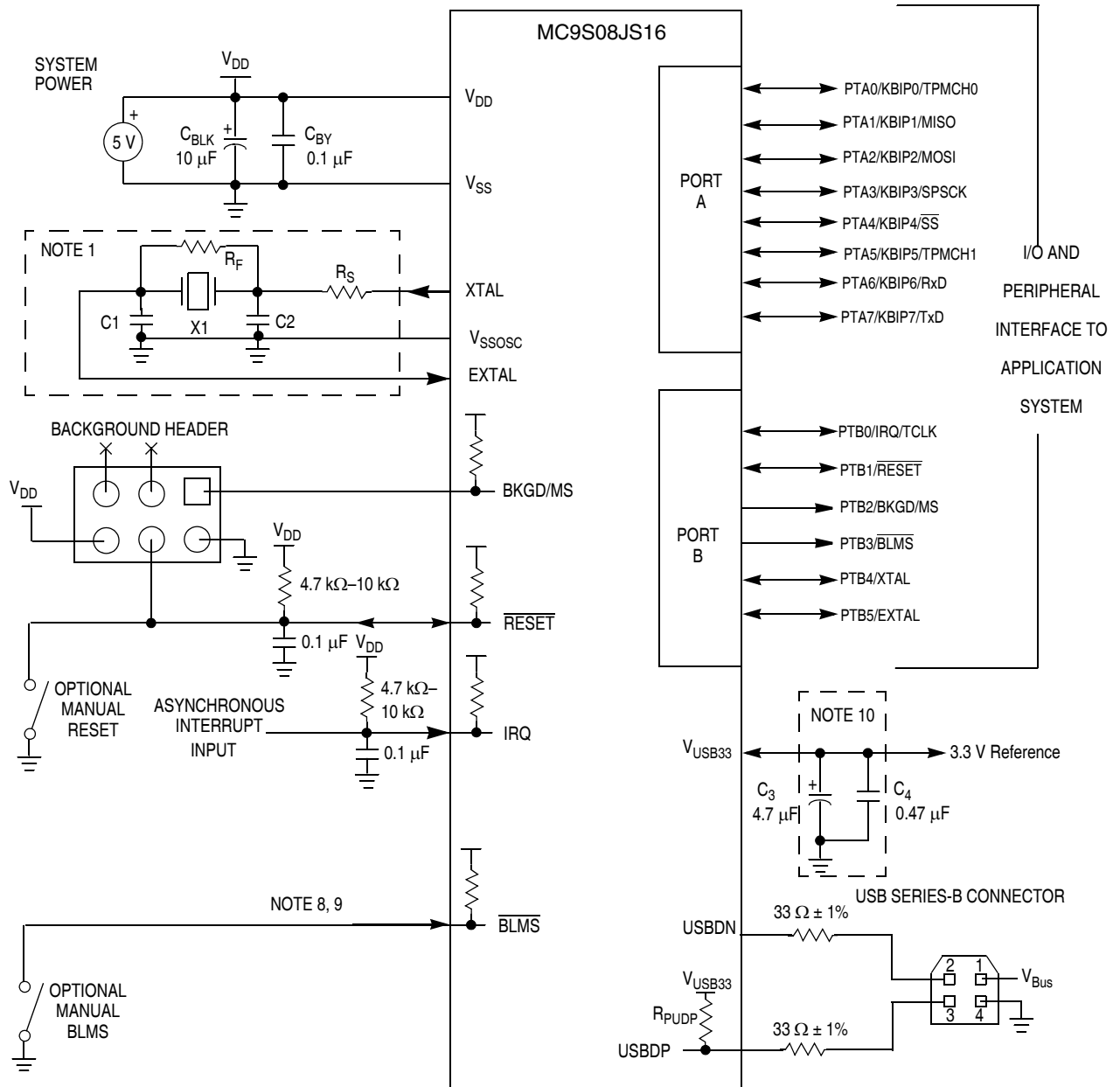


Figure 2-2. MC9S08JS16 Series in 20-Pin SOIC Package

2.3 Recommended System Connections

Figure 2-3 shows pin connections that are common to almost all MC9S08JS16 series application systems.



NOTES:

1. External crystal circuitry is not required if using the MCG internal clock. For USB operation, an external crystal is required.
2. XTAL and EXTAL use the same pins as PTB4 and PTB5, respectively.
3. RC filters on RESET and IRQ are recommended for EMC-sensitive applications.
4. R_{PUDP} is shown for full-speed USB only. The diagram shows a configuration where the on-chip regulator and R_{PUDP} are enabled. The voltage regulator output is used for R_{PUDP}. R_{PUDP} can optionally be disabled if using an external pullup resistor on USBDP.
5. V_{BUS} is a 5.0 V supply from upstream port that can be used for USB operation.
6. USBDP and USBDN are powered by the 3.3 V regulator.
7. For USB operation, an external crystal with a value of 2 MHz or 4 MHz is recommended.
8. BLMS pin has loading limitation, do not connect any cap with this pin.
9. If there is an external pullup resistor on PTB2/PTB3, PTB2/PTB3 must be configured as an output pin. Otherwise there will be current consumption on the pin (about 0.5 mA for a 10 kΩ resistor). The load on PTB2/PTB3 must be smaller than 50 pF.
10. When using internal V_{USB33} as supply, there needs to be an external cap.

Figure 2-3. Basic System Connections

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

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

Typically, application systems have two separate capacitors across the power pins. In this case, there must be a bulk electrolytic capacitor, such as a 10 μF tantalum capacitor, to provide bulk charge storage for the overall system and a 0.1 μF ceramic bypass capacitor located as near to the paired V_{DD} and V_{SS} power pins as practical to suppress high-frequency noise. The MC9S08JS16 has a V_{SSOSC} pin. This pin must be connected to the system ground plane or to the primary V_{SS} pin through a low-impedance connection.

V_{USB33} is connected to the internal USB 3.3 V regulator. See [Chapter 15, “Universal Serial Bus Device Controller \(S08USBV1\)”](#) for a complete description of the V_{USB33} power pin.

2.3.2 Oscillator (XTAL, EXTAL)

Immediately after reset, the MCU uses an internally generated clock provided by the multi-purpose clock generator (MCG) module. For more information on the MCG, see [Chapter 9, “Multi-Purpose Clock Generator \(S08MCGV1\)”](#).

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

R_S (when used) and R_F must be low-inductance resistors such as carbon composition resistors. Wire-wound resistors, and some metal film resistors, have too much inductance. C1 and C2 normally are 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 M Ω to 10 M Ω . Higher values are sensitive to humidity and lower values reduce gain and (in extreme cases) could prevent startup.

C1 and C2 are typically in the 5 pF to 25 pF range and are chosen to match the requirements of a specific crystal or resonator. Be sure to take into account printed circuit board (PCB) capacitance and MCU pin capacitance when selecting C1 and C2. The crystal manufacturer typically specifies a load capacitance 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).

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

After a power-on reset (POR), the PTB1/ $\overline{\text{RESET}}$ pin defaults to a general-purpose input port pin, PTB1. Setting RSTPE in SOPT1 configures the pin to be the $\overline{\text{RESET}}$ pin containing an internal pullup device. After configured as $\overline{\text{RESET}}$, the pin remains $\overline{\text{RESET}}$ until the next LVD or POR. The $\overline{\text{RESET}}$ pin when enabled 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. If desired, a manual external reset can be added by supplying a simple switch to ground (pull reset pin low to force a reset).

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

NOTE

The voltage on the internally pulled up $\overline{\text{RESET}}$ pin when measured is below V_{DD} . The internal gates connected to this pin are pulled to V_{DD} . If the $\overline{\text{RESET}}$ pin is required to drive to a V_{DD} level, an external pullup must be used.

NOTE

In EMC-sensitive applications, an external RC filter is recommended on the $\overline{\text{RESET}}$ pin, if enabled (Figure 2-3).

2.3.4 Background/Mode Select (BKGD/MS)

During a power-on-reset (POR) or background debug force reset (see Section 5.7.3, “System Background Debug Force Reset Register (SBD FR),” for details), the PTB2/BKGD/MS pin functions as a mode select pin. Immediately after reset rises the pin functions as the background pin and can be used for background debug communication. When enabled as the BKGD/MS pin ($\text{BKGDPE} = 1$), an internal pullup device is automatically enabled.

The background debug communication function is enabled when BKGDPE in SOPT1 is set. BKGDPE is set following any reset of the MCU and must be cleared to use the PTB2/BKGD/MS pin’s alternative pin functions.

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

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

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

NOTE

Before the IRQ or $\overline{\text{RESET}}$ function is enabled, be sure to enable the GPIO pullup in that function's pin and wait about 2 μs . Otherwise the IRQ or $\overline{\text{RESET}}$ configuration may fail.

2.3.5 Bootloader Mode Select ($\overline{\text{BLMS}}$)

During a power-on-reset (POR), the CPU detects the state of the PTB3/ $\overline{\text{BLMS}}$ pin that functions as a mode select pin. When the logic is low and BKGD/MS is not pulled low, the CPU enters the bootloader mode. During a power-on-reset (POR), an internal pullup device is automatically enabled in PTB3/ $\overline{\text{BLMS}}$ pin. Immediately after reset rises the pin functions as a general-purpose output only pin and an internal pullup device is automatically disabled.

2.3.6 USB Data Pins (USBDP, USBDN)

The USBDP (D+) and USBDN (D-) pins are the analog input/output lines for full-speed data communication in the USB physical layer (PHY) module. An optional internal pullup resistor for the USBDP pin, R_{PUDP} , is available.

2.3.7 General-Purpose I/O and Peripheral Ports

The MC9S08JS16 series of MCUs supports up to 14 general-purpose I/O pins, including two output-only pins, which are shared with on-chip peripheral functions (timers, serial I/O, keyboard interrupts, etc.).

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

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

Immediately after reset, all pins except the output-only pin (PTB2/BKGD/MS, PTB3/ $\overline{\text{BLMS}}$) are configured as high-impedance general-purpose inputs with internal pullup devices disabled.

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

Pin Number (Package)		<-- Lowest Priority --> Highest		
24 (QFN)	20 (SOIC)	Port Pin	Alt 1	Alt 2
1	4	PTB0	IRQ	TCLK
2	5	PTB1		$\overline{\text{RESET}}$
3	6	PTB2	BKGD	MS
4	7	PTB3		$\overline{\text{BLMS}}$

Table 2-1. Pin Availability by Package Pin-Count (continued)

Pin Number (Package)		<-- Lowest Priority --> Highest		
24 (QFN)	20 (SOIC)	Port Pin	Alt 1	Alt 2
5	8	PTA0	KBIP0	TPMCH0
6	—	NC		
7	9	PTA1	KBIP1	MISO
8	10	PTA2	KBIP2	MOSI
9	11	PTA3	KBIP3	SPSCK
10	12	PTA4	KBIP4	\overline{SS}
11	13			V_{DD}
12	—	NC		
13	14			V_{SS}
14	15			USBDN
15	16			USBDP
16	17			V_{USB33}
17	18	PTA5	KBIP5	TPMCH1
18	—	NC		
19	19	PTA6	KBIP6	RxD
20	20	PTA7	KBIP7	TxD
21	1	PTB4	XTAL	
22	2	PTB5	EXTAL	
23	3			V_{SSOSC}
24	—	NC		

NOTE

When an alternative function is first enabled, it is possible to get a spurious edge to the module. User software must clear 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. All modules that share a pin must be disabled before another module is enabled.

Chapter 3

Modes of Operation

3.1 Introduction

The operating modes of the MC9S08JS16 series are described in this section. Entry into each mode, exit from each mode, and functionality while in each mode are described.

3.2 Features

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

3.3 Run Mode

Run is the normal operating mode for the MC9S08JS16 series. This mode is selected upon the MCU exiting reset if the BKGD/MS pin is high. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFFE:0xFFFF after reset.

3.4 Active Background Mode

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

Active background mode is entered in any of the following ways:

- When the BKGD/MS pin is low during POR or immediately after issuing a background debug force reset (see [Section 5.7.3, “System Background Debug Force Reset Register \(SBDFR\)”](#))
- 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 MC9S08JS16 series devices are 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 17, “Development Support.”](#)

3.5 Wait Mode

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

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

3.6 Stop Modes

One of two stop modes is entered upon execution of a STOP instruction when STOPE in SOPT1 is set. In any stop mode, the bus and CPU clocks are halted. The MCG module can be configured to leave the reference clocks running. See [Chapter 9, “Multi-Purpose Clock Generator \(S08MCGV1\)”](#), for more information.

Some HCS08 devices that are designed for low-voltage operation (1.8 to 3.6 V) also include stop1 mode. The MC9S08JS16 series of MCUs do not include stop1 mode.

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

¹ ENBDM is located in the BDCSCR which is only accessible through BDC commands, see [Section 14.4.1.1, “BDC Status and Control Register \(BDCSCR\)”](#).

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

3.6.1 Stop3 Mode

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

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

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

3.6.1.1 LVD Enabled in Stop Mode

The LVD system is capable of generating 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 XOSC to operate with an external reference when RANGE in MCGC2 is set, the LVD must be left enabled when entering stop3.

NOTE

To get low USB suspend current, before entering USB suspend mode, we must set ERCLKEN and EREFSTEN in MCGC2, but leave LVD disabled in stop3.

3.6.1.2 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if ENBDM in BDCSCR is set. This register is described in [Chapter 17, “Development Support.”](#) If ENBDM is set when the CPU executes a STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation. If the user attempts to enter stop2 with ENBDM set, the MCU will enter stop3 instead.

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

3.6.2 Stop2 Mode

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

Exit from stop2 is performed by asserting either wakeup pin: $\overline{\text{RESET}}$ or IRQ. When an application utilizes the stop2 state, $\overline{\text{RESET}}$ or IRQ pin must be pre-configured as an input prior to entering stop2. There is a direct analog connection from $\overline{\text{RESET}}$ or IRQ pad to the power management controller wakeup pin — if configured as a GPIO output, it could prevent operation of stop2.

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

Upon wakeup 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 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 are 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 are configured as peripheral I/O, the user must reconfigure the peripheral module that interfaces to the pin before writing to the PPDACK bit. If the peripheral module is not enabled before writing to PPDACK, the pins will be controlled by their associated port control registers when the I/O latches are opened.

3.6.3 On-Chip Peripheral Modules in Stop Modes

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

Table 3-2. Stop Mode Behavior

Peripheral	Mode	
	Stop2	Stop3
CPU	Off	Standby
RAM	Standby	Standby
Flash	Off	Standby
Parallel Port Registers	Off	Standby
MCG	Off	Optionally On ¹
RTC	Optionally on ²	Optionally on ²
SCI	Off	Standby
SPI	Off	Standby
MTIM	Off	Standby
TPM	Off	Standby
System Voltage Regulator	Off	Standby
XOSC	Off	Optionally On ³
I/O Pins	States Held	States Held
USB (SIE and PHY)	Off	Optionally On ⁴
USB 3.3 V Regulator	Off	Standby
USB RAM	Standby	Standby
CRC	Off	Standby

¹ IREFSTEN set in MCGC1, else in standby.

² RTCPS[3:0] in RTCSC does not equal 0 before entering stop, else off.

³ EREFSTEN set in MCGC2, else in standby. For high frequency range (RANGE in MCGC2 set), the LVD must also be enabled in stop3.

⁴ USBEN in CTL is set and USBPHYEN in USBCTL0 is set, else off.

Chapter 4 Memory

4.1 MC9S08JS16 Series Memory Map

Figure 4-1 shows the memory map for the MC9S08JS16 series. On-chip memory in the MC9S08JS16 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 0x007F)
- High-page registers (0x1800 through 0x185F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

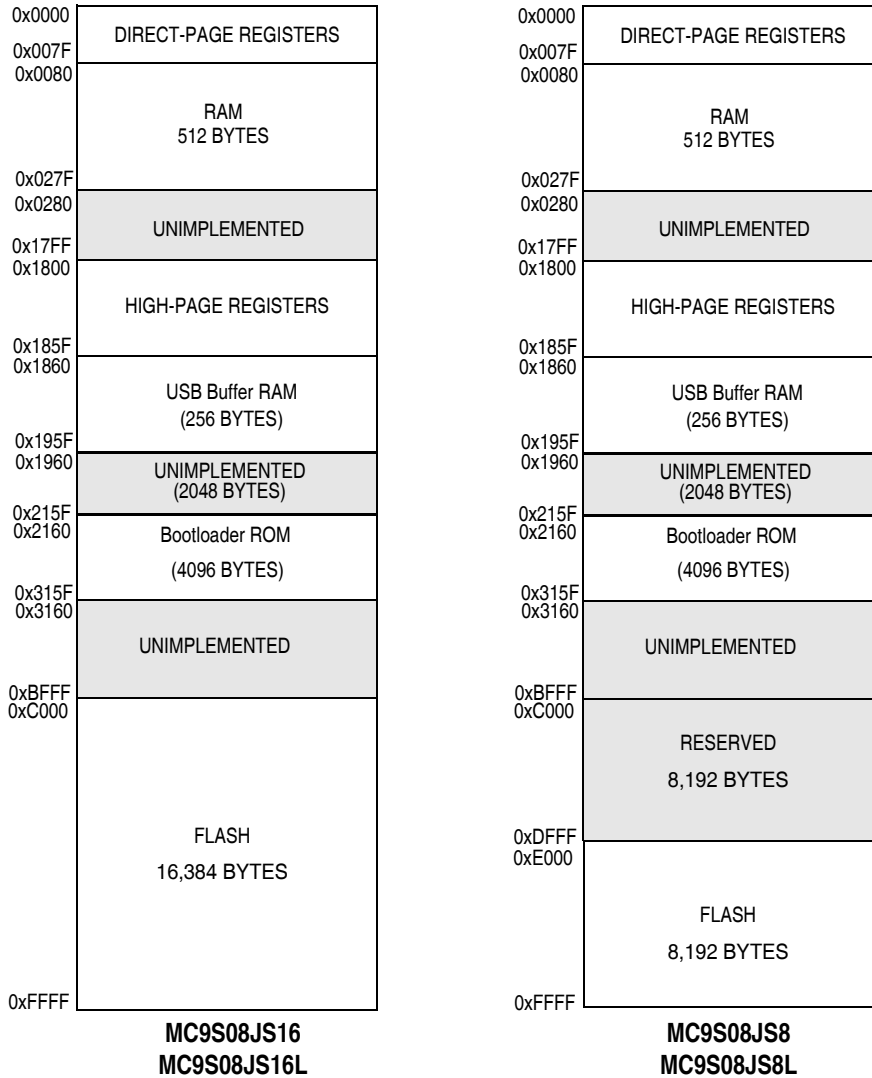


Figure 4-1. MC9S08JS16 Series Memory Map

4.1.1 Reset and Interrupt Vector Assignments

Table 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 MC9S08JS16 series. For more details about resets, interrupts, interrupt priority, and local interrupt mask controls, refer to Chapter 5, “Resets, Interrupts, and System Configuration.”

Table 4-1. Reset and Interrupt Vectors

Address (High/Low)	Vector	Vector Name
0xFFC0:0xFFC1 to 0xFFC2:FFC3	Unused vector space ¹	—
0xFFC4:FFC5	RTC	Vrtc
0xFFC6:FFC7	Unused vector space	—
0xFFC8:FFC9	Unused vector space	—
0xFFCA:FFCB	MTIM	Vmtim
0xFFCC:FFCD	KBI	Vkeyboard
0xFFCE:FFCF to 0xFFD2:FFD3	Unused vector space	—
0xFFD4:FFD5	SCI Transmit	Vscitx
0xFFD6:FFD7	SCI Receive	Vscirx
0xFFD8:FFD9	SCI Error	Vscierr
0xFFDA:FFDB to 0xFFE6:FFE7	Reserved	—
0xFFE8:FFE9	TPM Overflow	Vtpmovf
0xFFEA:FFEB	TPM Channel 1	Vtpmch1
0xFFEC:FFED	TPM Channel 0	Vtpmch0
0xFFEE:FFEF	Unused vector space	—
0xFFF0:FFF1	USB Status	Vusb
0xFFF2:FFF3	Unused vector space	—
0xFFF4:FFF5	SPI	Vspi
0xFFF6:FFF7	MCG Loss of Lock	Vlol
0xFFF8:FFF9	Low Voltage Detect	Vlvd
0xFFFA:FFFB	IRQ	Virq
0xFFFC:FFFD	SWI	Vswi
0xFFFE:FFFF	Reset	Vreset

¹ Unused vector space is available for use as general flash memory. However, other devices in the S08 family of MCUs may use these locations as interrupt vectors. Therefore, care must be taken when using these locations if the code will be ported to other MCUs.

4.2 Register Addresses and Bit Assignments

The registers in the MC9S08JS16 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:

- Three values that are loaded into working registers at reset
- An 8-byte backdoor comparison key that 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 that requires only 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 3)

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	0	0	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x0003	PTBDD	0	0	PTBDD5	PTBDD4	R	R	PTBDD1	PTBDD0
0x0004– 0x0007	Reserved	—	—	—	—	—	—	—	—
0x0008	MTIMSC	TOF	TOIE	TRST	TSTP	0	0	0	0
0x0009	MTIMCLK	0	0	CLKS		PS			
0x000A	MTIMCNT	COUNT							
0x000B	MTIMMOD	MOD							
0x000C	CRCH	Bit 15	14	13	12	11	10	9	Bit 8
0x000D	CRCL	Bit 7	6	5	4	3	2	1	Bit 0
0x000E	Reserved	—	—	—	—	—	—	—	—
0x000F	Reserved	—	—	—	—	—	—	—	—
0x0010	TPMSC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0011	TPMCNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0012	TPMCNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0013	TPMMODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0014	TPMMODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0015	TPMC0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0016	TPMC0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0017	TPMC0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0018	TPMC1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0019	TPMC1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x001A	TPMC1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x001B	IRQSC	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x001C	KBISC	0	0	0	0	KBF	KBACK	KBIE	KBMOD
0x001D	KBIPE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x001E	KBIES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
0x001F	Reserved	—	—	—	—	—	—	—	—
0x0020	SCIBDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0021	SCIBDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x0022	SCIC1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x0023	SCIC2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x0024	SCIS1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x0025	SCIS2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x0026	SCIC3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x0027	SCID	Bit 7	6	5	4	3	2	1	Bit 0
0x0028– 0x002F	Reserved	—	—	—	—	—	—	—	—
0x0030	SPIC1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0031	SPIC2	SPMIE	SPIMODE	0	MODFEN	BIDIROE	0	SPISWAI	SPC0

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0032	SPIBR	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
0x0033	SPIS	SPRF	SPMF	SPTEF	MODF	0	0	0	0
0x0034	SPIDH	Bit 15	14	13	12	11	10	9	Bit 8
0x0035	SPIDL	Bit 7	6	5	4	3	2	1	Bit 0
0x0036	SPIMH	Bit 15	14	13	12	11	10	9	Bit 8
0x0037	SPIML	Bit 7	6	5	4	3	2	1	Bit 0
0x0038– 0x003F	Reserved	—	—	—	—	—	—	—	—
0x0040	MCGC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
0x0041	MCGC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
0x0042	MCGTRM	TRIM							
0x0043	MCGSC	LOLS	LOCK	PLLST	IREFST	CLKST		OSCINIT	FTRIM
0x0044	MCGC3	LOLIE	PLLS	CME	0	VDIV			
0x0045– 0x0047	Reserved	—	—	—	—	—	—	—	—
0x0048	RTCSC	RTIF	RTCLKS		RTIE	RTCPS			
0x0049	RTCCNT	RTCCNT							
0x004A	RTCMOD	RTCMOD							
0x004B– 0x004F	Reserved	—	—	—	—	—	—	—	—
0x0050	USBCTLO	USBRESET	USBPU	USBRESMEN	LPRESF	0	USBVREN	0	USBPHYEN
0x0051– 0x0057	Reserved	—	—	—	—	—	—	—	—
0x0058	PERID	0	0	ID5	ID4	ID3	ID2	ID1	ID0
0x0059	IDCOMP	1	1	NID5	NID4	NID3	NID2	NID1	NID0
0x005A	REV	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
0x005B– 0x005E	Reserved	—	—	—	—	—	—	—	—
0x005F	Reserved	—	—	—	—	—	—	—	—
0x0060	INTSTAT	STALLF	0	RESUMEF	SLEEPF	TOKDNEF	SOFTOKF	ERRORF	USBRSTF
0x0061	INTENB	STALL	0	RESUME	SLEEP	TOKDNE	SOFTOK	ERROR	USBRST
0x0062	ERRSTAT	BTSERRF	—	BUFERRF	BTOERRF	DFN8F	CRC16F	CRC5F	PIDERRF
0x0063	ERRENB	BTSERR	0	BUFERR	BTOERR	DFN8	CRC16	CRC5	PIDERR
0x0064	STAT	ENDP				IN	ODD	0	0
0x0065	CTL	—	—	TSUSPEND	—	—	CRESUME	ODDRST	USBEN
0x0066	ADDR	0	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
0x0067	FRMNUML	FRM7	FRM6	FRM5	FRM4	FRM3	FRM2	FRM1	FRM0
0x0068	FRMNUMH	0	0	0	0	0	FRM10	FRM9	FRM8
0x0069– 0x006C	Reserved	—	—	—	—	—	—	—	—
0x006D	EPCTL0	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSK
0x006E	EPCTL1	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSK
0x006F	EPCTL2	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSK

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0070	EPCTL3	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x0071	EPCTL4	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x0072	EPCTL5	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x0073	EPCTL6	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x0074– 0x007F	Reserved	—	—	—	—	—	—	—	—

High-page registers, as 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.

NOTE

The MCG factory trim values will automatically be loaded into the 0x180C and 0x180D registers after any reset.

Table 4-3. High-Page Register Summary (Sheet 1 of 2)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	LOC	LVD	—
0x1801	SBD FR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT1	COPT		STOPE	1	0	BLMSS	BKGDPE	RSTPE
0x1803	SOPT2	COPCLKS	COPW	0	0	0	SPIFE	0	0
0x1804	Reserved	—	—	—	—	—	—	—	—
0x1805	Reserved	—	—	—	—	—	—	—	—
0x1806	SDIDH	—	—	—	—	ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	Reserved	—	—	—	—	—	—	—	—
0x1809	SPMSC1	LVWF	LVWACK	LVWIE	LVDRE	LVDSE	LVDE	0	0
0x180A	SPMSC2	0	0	LVDV	LVWV	PPDF	PPDACK	0	PPDC
0x180B	Reserved	—	—	—	—	—	—	—	—
0x180C	Reserved for storage of FTRIM	0	0	0	0	0	0	0	FTRIM
0x180D	Reserved for storage of MCGTRIM	TRIM							
0x180E	FPROTD	—	—	—	—	—	—	—	Bit 0
0x180F	SIGNATURE	SIGNATURE semaphore							
0x1810	DBGCAH	Bit 15	14	13	12	11	10	9	Bit 8
0x1811	DBGCAL	Bit 7	6	5	4	3	2	1	Bit 0
0x1812	DBGCBH	Bit 15	14	13	12	11	10	9	Bit 8
0x1813	DBG CBL	Bit 7	6	5	4	3	2	1	Bit 0
0x1814	DBGFH	Bit 15	14	13	12	11	10	9	Bit 8
0x1815	DBGFL	Bit 7	6	5	4	3	2	1	Bit 0
0x1816	DBG C	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
0x1817	DBG T	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
0x1818	DBG S	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1819–	Reserved	—	—	—	—	—	—	—	—
0x181F		—	—	—	—	—	—	—	—
0x1820	FCDIV	DIVLD	PRDIV8	DIV5	DIV4	DIV3	DIV2	DIV1	DIV0
0x1821	FOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
0x1822	Reserved	—	—	—	—	—	—	—	—
0x1823	FCNFG	0	0	KEYACC	0	0	0	0	0
0x1824	FPROT	FPS7	FPS6	FPS5	FPS4	FPS3	FPS2	FPS1	FPDIS
0x1825	FSTAT	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x1826	FCMD	FCMD7	FCMD6	FCMD5	FCMD4	FCMD3	FCMD2	FCMD1	FCMD0
0x1827–	Reserved	—	—	—	—	—	—	—	—
0x183F		—	—	—	—	—	—	—	—
0x1840	PTAPE	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
0x1841	PTASE	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x1842	PTADS	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x1843	Reserved	—	—	—	—	—	—	—	—
0x1844	PTBPE	0	0	PTBPE5	PTBPE4	R	R	PTBPE1	PTBPE0
0x1845	PTBSE	0	0	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	0
0x1846	PTBDS	0	0	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	0
0x1847–	Reserved	—	—	—	—	—	—	—	—
0x185F		—	—	—	—	—	—	—	—

Nonvolatile flash registers, as shown in [Table 4-4](#), are located in the flash memory. These registers include an 8-byte backdoor key that 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.

The factory MCG trim value is stored in the flash information row (IFR¹) and will be loaded into the MCGTRM and MCGSC registers after any reset. The internal reference trim values stored in flash, TRIM and FTRIM, can be programmed by third party programmers and must be copied into the corresponding MCG registers by user code to override the factory trim.

NOTE

When the MCU is in active BDM, the trim value in the IFR will not be loaded, the MCGTRM register will reset to 0x80 and the FTRIM bit in the MCGSC register will be reset to 0.

1. IFR — Nonvolatile information memory that can be only accessed during production test. During production test, system initialization, configuration and test information is stored in the IFR. This information cannot be read or modified in normal user or background debug modes.

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 MCGTRIM	TRIM							
0xFFB0– 0xFFB7	NVBACKKEY	8-Byte Comparison Key							
0xFFB8		Flash Block Checksum High Byte							
0xFFB9		Flash Block Checksum Low Byte							
0xFFBA		Checksum Bypass							
0xFFBB	Reserved for user's storage of data	User Data							
0xFFBC	Reserved for user's storage of data	User Data							
0xFFBD	NVPROT	FPS7	FPS6	FPS5	FPS4	FPS3	FPS2	FPS1	FPDIS
0xFFBE		Flash Partial Erase semaphore							
0xFFBF	NVOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00

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

0xFFBB and 0xFFBC are used to store user data, such as the user's MCG trimmed value.

4.3 RAM (System RAM)

The MC9S08JS16 series devices include 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 MC9S08JS16 series, 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
```

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 4.7, “Security,”](#) for a detailed description of the security feature.

4.4 USB RAM

USB RAM is discussed in detail in [Chapter 15, “Universal Serial Bus Device Controller \(S08USBV1\).”](#)

4.5 Bootloader ROM

The Bootloader ROM holds code that is used to erase and program flash memory. The Bootloader ROM provides quick and reliable flash erase and program procedures from a USB host such as a personal computer (PC).

Freescall provides a PC GUI (Graphic User Interface) that can communicate with bootloader in ROM via USB interface (PC as a host USB device).

Working with the PC GUI, the user can:

- Mass erase entire flash array
- Partial erase flash array— erase all flash blocks except for the first 1 KB of flash
- Program flash
- Reset MCU

NOTE

If bootloader function is used, the MCU supply voltage must be above 4 V. The internal USB 3.3 V voltage regulator will be enabled when entering bootloader mode.

NOTE

USB bootloader requires an external oscillator which must be at 2 MHz, 4 MHz, 6 MHz, 8 MHz, 12 MHz, or 16 MHz. Bootloader code can identify the external oscillator automatically. If the bootloader is not used, there are no such restrictions.

NOTE

The USB descriptors for bootloader are fixed: VID is 0x15A2, PID is 0x0038. The user’s application can use its own descriptors.

4.5.1 External Signal Description

The $\overline{\text{BLMS}}$ pin of bootloader ROM decides whether the MCU will enter bootloader mode directly during power-on reset. This pin is only examined during Power-On Reset (POR).

The signal properties of bootrom are shown in [Table 4-5](#).

Table 4-5. Signal Properties

Signal	Function	I/O
BLMS	Bootloader mode selection	I

4.5.2 Modes of Operation

After any reset, the MCU jumps to bootloader ROM address, where several qualification factors are examined to determine whether to jump to the bootloader code or to the user code. The bootloader ROM can be accessed in bootloader mode or user mode. This section describes the valid operations and protection mechanism in these two 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

4.5.2.1 Bootloader Mode

Bootloader mode can be entered in the following 4 conditions:

1. 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.
2. 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.
3. 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.
4. 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.

4.5.2.2 User mode

User mode can be entered in the following 3 conditions:

1. 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.
2. 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.
3. When a reset occurs (other than a power-on reset), if the SIGNATURE semaphore is not equal to 0xC3, the user mode is entered.

4.5.2.3 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.

4.5.2.4 Disable Flash Protection

The protected flash section can be disabled by back-to-back writes of 0x55 and 0xAA to the address of FPROTD (flash protection defeat register), then setting bit FPDIS = 1 in FPROT.

4.5.3 Flash Memory Map

The general flash memory map of bootloader is shown in [Figure 4-2](#).

- Flash block checksum stores in 0xFFB8 (checksum high byte) and 0xFFB9 (checksum low byte)
- Checksum bypass information stores in 0xFFBA
- Flash partial erase semaphore stores in 0xFFBE

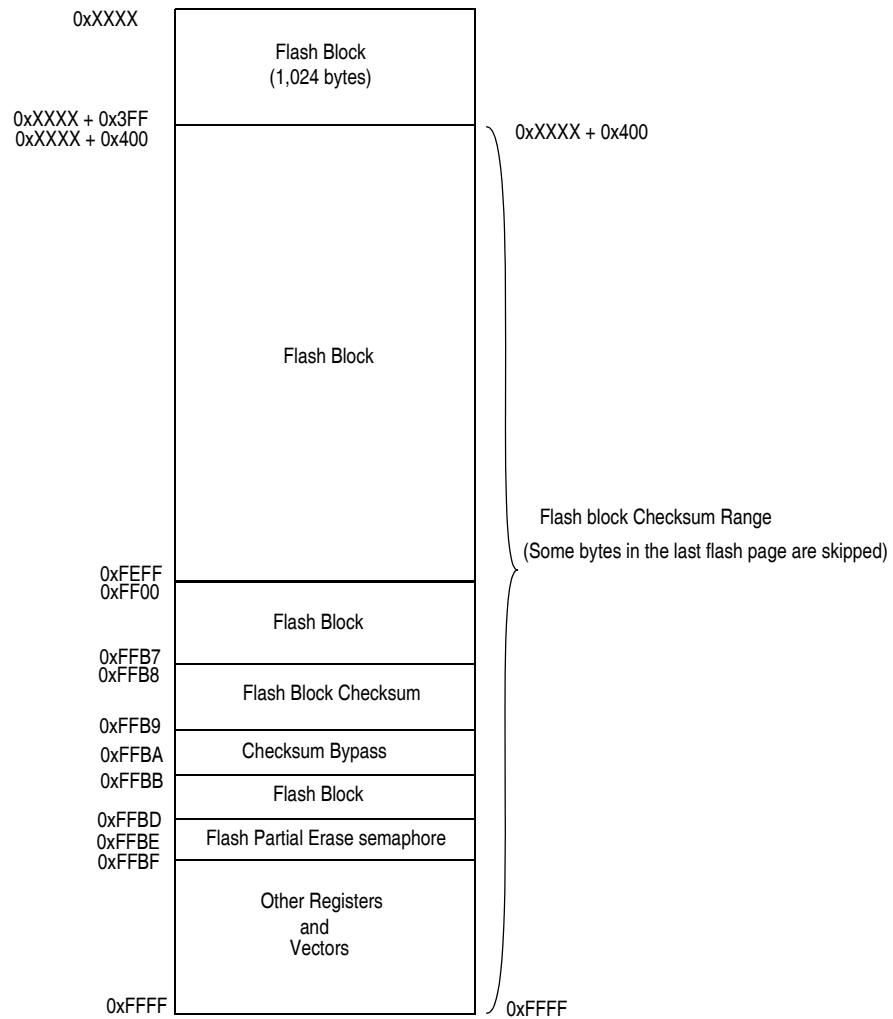


Figure 4-2. General Flash Memory Map

The value of the checksum bypass is programmed by the user. This value is checked by bootloader software only after power-on reset. The checksum bypass value in this byte indicates if the flash block checksum will be calculated or not, and if the flash sector whose address is behind the first 1 KB of flash is used as EEPROM. When the value is other than 0x00 and 0xFF, the calculation of flash block checksum is bypassed and the code jumps to bootloader entry. When the value is 0xFF, the flash block checksum will be calculated after power-on reset. When the value is 0x00, the calculation of flash block checksum is bypassed, the MCU jumps to user code entry and the flash sectors whose addresses are behind the first 1 KB of flash will be used as EEPROM.

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

NOTE

The first 1 KB segment of flash array is not in the CRC checksum calculation so that users can deploy this area as user EEPROM. If users need to use other flash locations out of the first 1 KB segment, they need to re-calculate the CRC and re-program the CRC checksum to ensure reliable entry after a POR.

4.5.4 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

4.5.4.1 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 from the end of first 1 KB of the flash to 0xFFFF (some bytes in last flash page are skipped). The first 1 KB of flash is not included in the checksum to allow users to use it as pseudo EEPROM in this area. 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.

JS16 flash block checksum range is 0xC400–0xFFAD and 0xFFC0–0xFFFF, JS8 flash block checksum range is 0xE400–0xFFAD and 0xFFC0–0xFFFF.

4.5.4.2 SIGNATURE Semaphore Register

Upon regular 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.

4.5.4.3 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.

4.5.4.4 Flow Chart

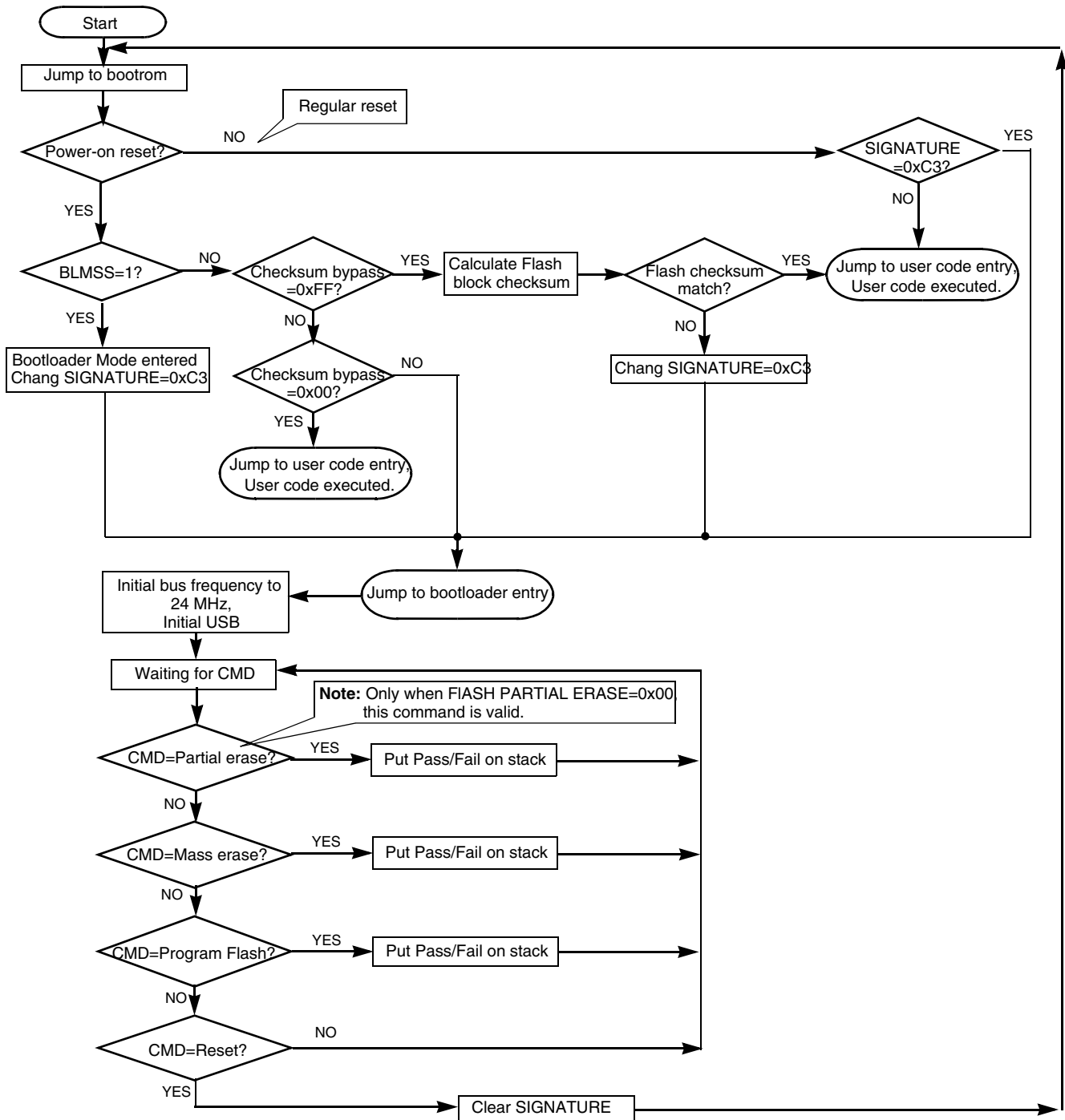


Figure 4-3. Bootloader Flow Chart

4.6 Flash Memory

The flash memory is 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.

4.6.1 Features

Features of the flash memory include:

- Flash size
 - MC9S08JS16 — 16,384 bytes (32 pages of 512 bytes each)
 - MC9S08JS8 — 8,192 bytes (16 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for flash and RAM
- Auto power-down for low-frequency read accesses

4.6.2 Program and Erase Time

Before any program or erase command can be accepted, the flash clock divider register (FCDIV) must be written to set the internal clock for the flash module to a frequency (f_{FCLK}) between 150 kHz and 200 kHz (see [Section 4.8.1, “Flash Clock Divider Register \(FCDIV\)”](#)). This register can be written only once, so normally this write is done during reset initialization. FCDIV cannot be written if the access error flag, FACCERR in FSTAT, is set. The user must ensure that FACCERR is not set before writing to the FCDIV register. One period of the resulting clock ($1/f_{\text{FCLK}}$) is used by the command processor to time program and erase pulses. An integer number of these timing pulses is used by the command processor to complete a program or erase command.

[Table 4-6](#) shows program and erase time. The bus clock frequency and FCDIV determine the frequency of FCLK (f_{FCLK}). The time for one cycle of FCLK is $t_{\text{FCLK}} = 1/f_{\text{FCLK}}$. The times are shown as a number of cycles of FCLK and as an absolute time for the case where $t_{\text{FCLK}} = 5 \mu\text{s}$. Program and erase time shown include overhead for the command state machine and enabling and disabling of program and erase voltages.

Table 4-6. Program and Erase Time

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

¹ Excluding start/end overhead

4.6.3 Program and Erase Command Execution

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

1. Write a data value to an address in the flash array. The address and data information from this write is latched into the flash interface. This write is a required first step in any command sequence. For erase and blank check commands, the value of the data is not important. For page erase commands, the address may be any address in the 512-byte page of flash to be erased. For mass erase and blank check commands, the address can be any address in the flash memory. Whole pages of 512 bytes are the smallest block of flash that can be erased.

NOTE

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

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

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

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

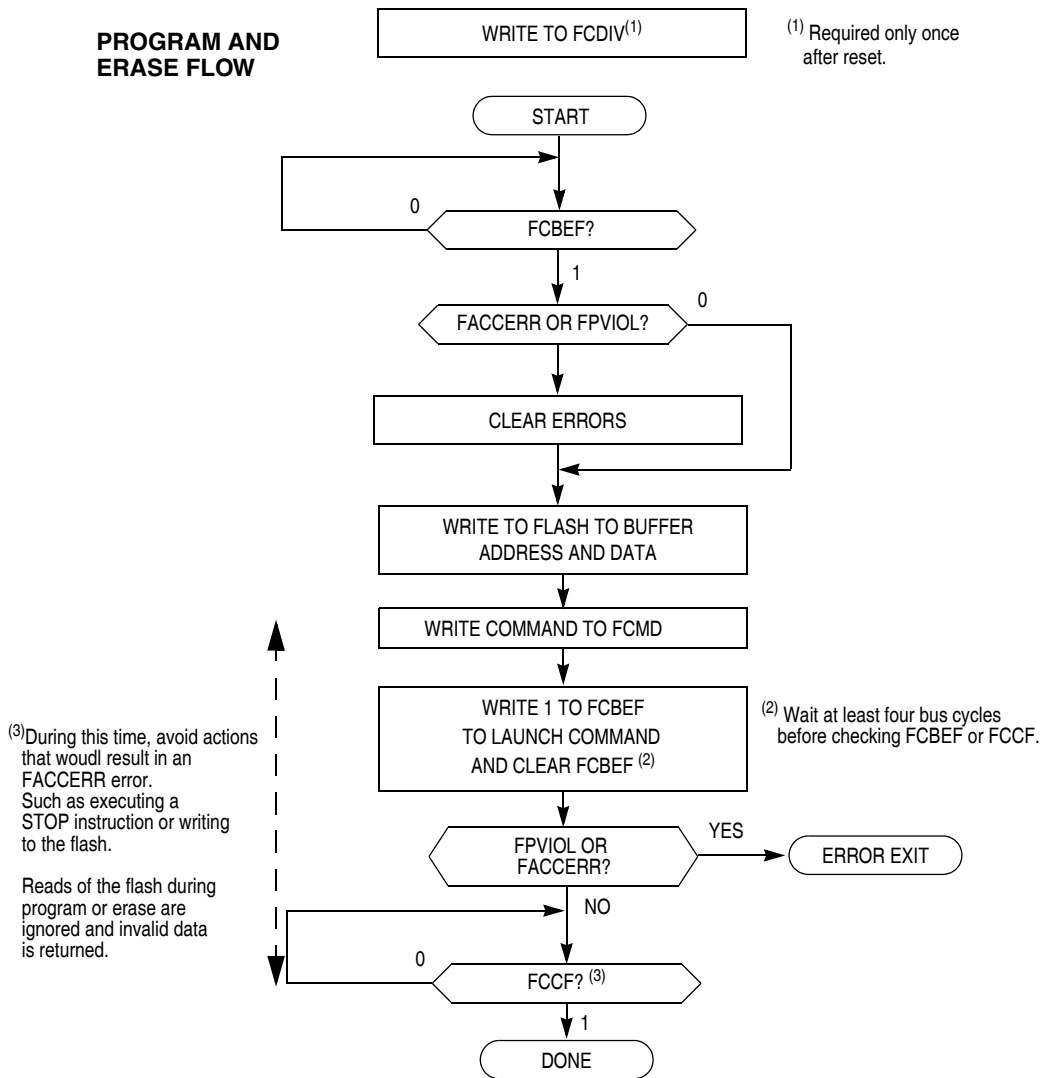


Figure 4-4. Flash Program and Erase Flowchart

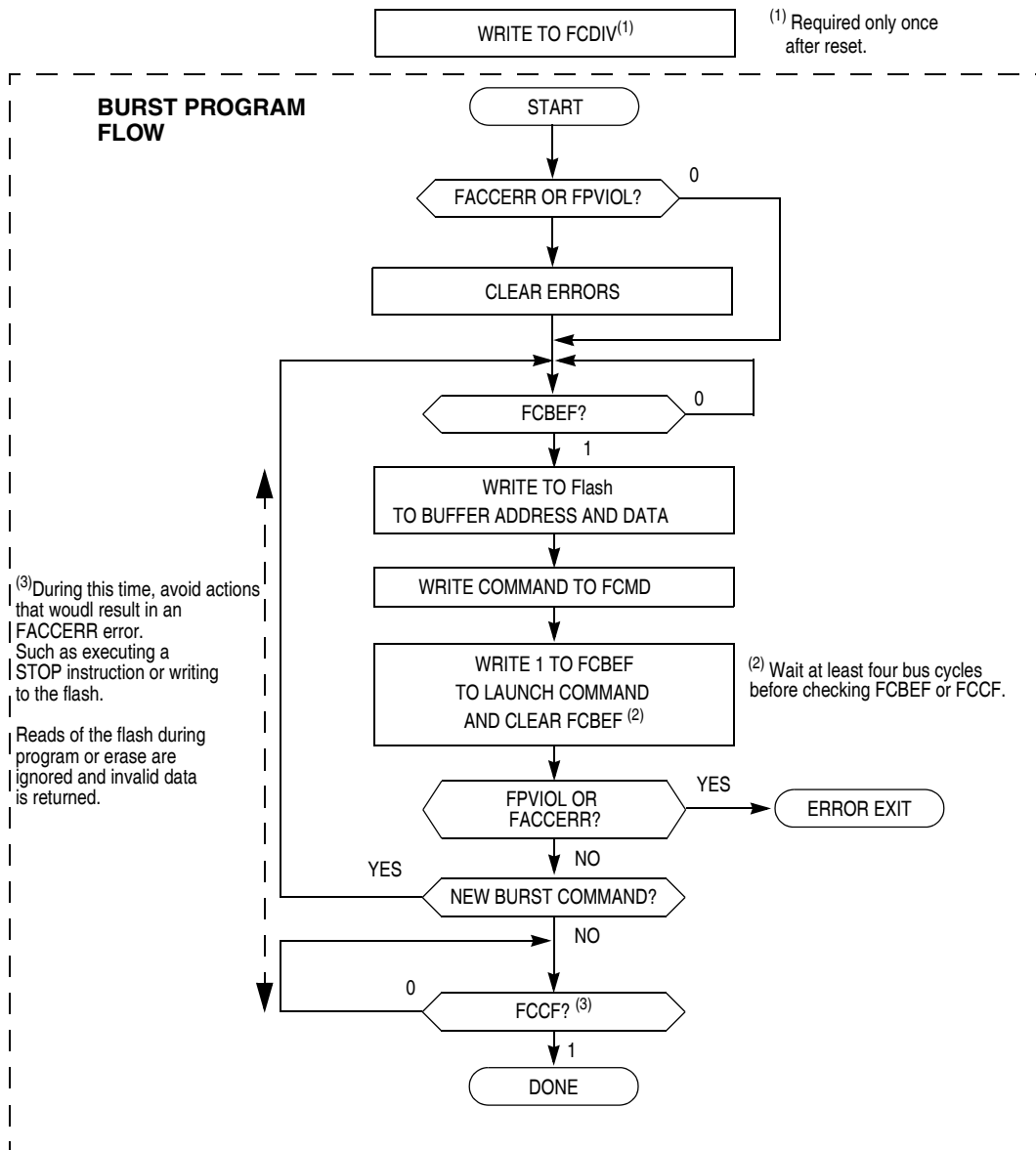
4.6.4 Burst Program Execution

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

- The next burst program command has been queued before the current program operation is completed.

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

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



4.6.5 Access Errors

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

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

- Writing to a flash address before the internal flash clock frequency has been set by writing to the FCDIV register
- Writing to a flash address while FCBEF is not set (A new command cannot be started until the command buffer is empty.)
- Writing a second time to a flash address before launching the previous command (There is only one write to flash for every command.)
- Writing a second time to FCMD before launching the previous command (There is only one write to FCMD for every command.)
- Writing to any flash control register other than FCMD after writing to a flash address
- Writing any command code other than the five allowed codes (0x05, 0x20, 0x25, 0x40, and 0x41) to FCMD
- Accessing (read or write) any flash control register other than the write to FSTAT (to clear FCBEF and launch the command) after writing the command to FCMD
- The MCU enters stop mode while a program or erase command is in progress (The command is aborted.)
- Writing the byte program, burst program, or page erase command code (0x20, 0x25, or 0x40) with a background debug command while the MCU is secured (The background debug controller can only do blank check and mass erase commands when the MCU is secure.)
- Writing 0 to FCBEF to cancel a partial command

4.6.6 Flash Block Protection

The block protection feature prevents the protected region of flash from program or erase changes. Block protection is controlled through the flash protection register (FPROT). When enabled, block protection begins at any 512 byte boundary below the last address of flash, 0xFFFF. (See [Section 4.8.4, “Flash Protection Register \(FPROT and NVPROT\).”](#))

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

The block protection mechanism is illustrated below. The FPS bits are used as the upper bits of the last address of unprotected memory. This address is formed by concatenating FPS7:FPS1 with logic 1 bit as shown. For example, to protect the last 1536 bytes of memory (addresses 0xFA00 through 0xFFFF), the FPS bits must be set to 1111 100, which results in the value 0xF9FF as the last address of unprotected memory. In addition to programming the FPS bits to the appropriate value, FPDIS (bit 0 of NVPROT)

must be programmed to logic 0 to enable block protection. Therefore the value 0xF8 must be programmed into NVPROT to protect addresses 0xFA00 through 0xFFFF.

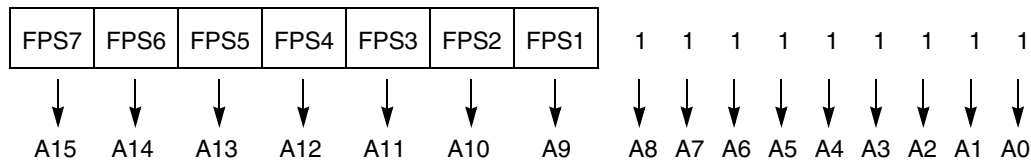


Figure 4-6. Block Protection Mechanism

One use for block protection is to block-protect an area of flash memory for a bootloader program. This bootloader program then can be used to erase the rest of the flash memory and reprogram it. Because the bootloader is protected, it remains intact even if MCU power is lost in the middle of an erase and reprogram operation.

4.6.7 Flash Block Protection Disabled

Protected flash section can be disabled by back to back write 0x55 and 0xAA to the address of FPROTD (flash protection defeat register) first, then setting bit FPDIS = 1 in FPROT.

4.6.8 Vector Redirection

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

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

4.7 Security

The MC9S08JS16 series include circuitry to prevent unauthorized access to the contents of flash and RAM memory. When security is engaged, flash and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

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

The on-chip debug module cannot be enabled while the MCU is secure. The separate background debug controller can still be used for background memory access commands of unsecured resources.

A user can choose to allow or disallow a security unlocking mechanism through an 8-byte backdoor security key. If the nonvolatile KEYEN bit in NVOPT/FOPT is 0, the backdoor key is disabled and there is no way to disengage security without completely erasing all flash locations. If KEYEN is 1, a secure user program can temporarily disengage security by:

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

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

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

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

1. Disable any block protections by writing FPROT. In MC9S08JS16 series, FPROT can be changed when FPROTD is set.
2. Mass erase flash if necessary.
3. Blank check flash. Provided flash is completely erased, security is disengaged until the next reset. To avoid returning to secure mode after the next reset, program NVOPT so SEC01:SEC00 = 1:0.

4.8 Flash Registers and Control Bits

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

4.8.1 Flash Clock Divider Register (FCDIV)

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

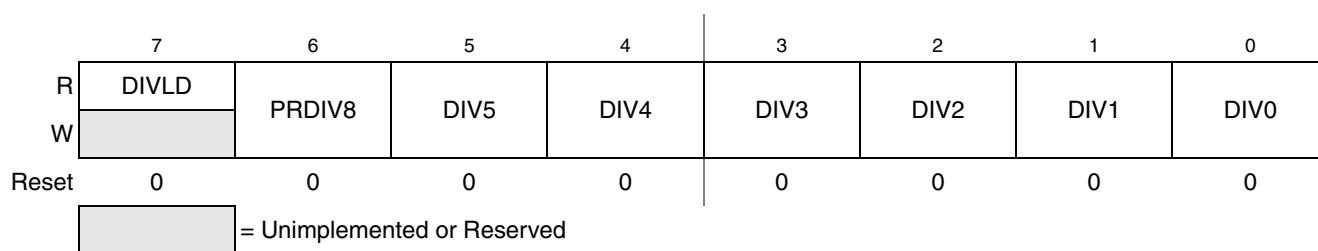


Figure 4-7. Flash Clock Divider Register (FCDIV)

Table 4-7. FCDIV Register Field Descriptions

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

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

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

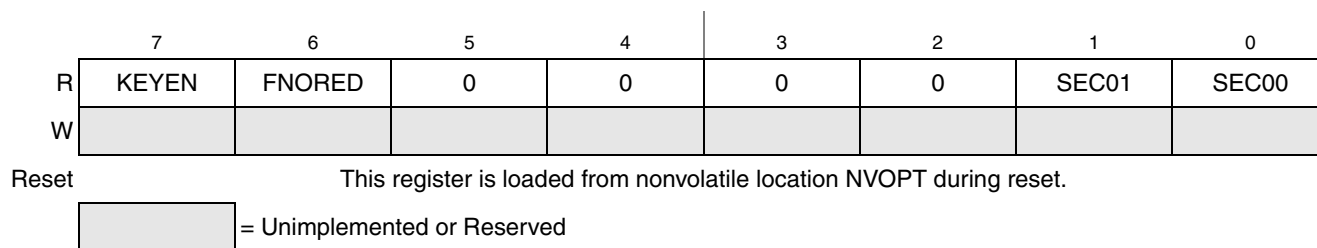
[Table 4-8](#) shows the appropriate values for PRDIV8 and DIV5:DIV0 for selected bus frequencies.

Table 4-8. Flash Clock Divider Settings

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

4.8.2 Flash Options Register (FOPT and NVOPT)

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


Figure 4-8. Flash Options Register (FOPT)
Table 4-9. FOPT Register Field Descriptions

Field	Description
7 KEYEN	Backdoor Key Mechanism Enable — When this bit is 0, the backdoor key mechanism cannot be used to disengage security. The backdoor key mechanism is accessible only from user (secured) firmware. BDM commands cannot be used to write key comparison values that would unlock the backdoor key. For more detailed information about the backdoor key mechanism, refer to Section 4.7, “Security.” 0 No backdoor key access allowed. 1 If user firmware writes an 8-byte value that matches the nonvolatile backdoor key (NVBACKKEY through NVBACKKEY+7 in that order), security is temporarily disengaged until the next MCU reset.
6 FNORED	Vector Redirection Disable — When this bit is 1, then vector redirection is disabled. 0 Vector redirection enabled. 1 Vector redirection disabled.
1:0 SEC0[1:0]	Security State Code — This 2-bit field determines the security state of the MCU as shown in Table 4-10 . When the MCU is secure, the contents of RAM and flash memory cannot be accessed by instructions from any unsecured source including the background debug interface. For more detailed information about security, refer to Section 4.7, “Security.”

Table 4-10. Security States

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

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

4.8.3 Flash Configuration Register (FCNFG)

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

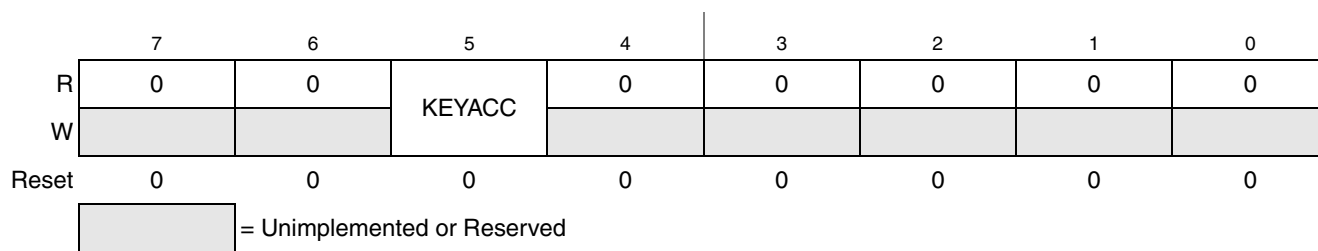


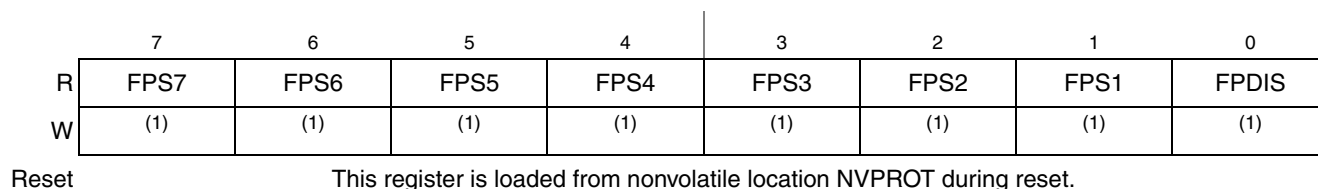
Figure 4-9. Flash Configuration Register (FCNFG)

Table 4-11. FCNFG Register Field Descriptions

Field	Description
5 KEYACC	Enable Writing of Access Key — This bit enables writing of the backdoor comparison key. For more detailed information about the backdoor key mechanism, refer to Section 4.7, “Security.” 0 Writes to 0xFFB0–0xFFB7 are interpreted as the start of a flash programming or erase command. 1 Writes to NVBACKKEY (0xFFB0–0xFFB7) are interpreted as comparison key writes.

4.8.4 Flash Protection Register (FPROT and NVPROT)

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



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

Figure 4-10. Flash Protection Register (FPROT)

Table 4-12. FPROT Register Field Descriptions

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

4.8.5 Flash Status Register (FSTAT)

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

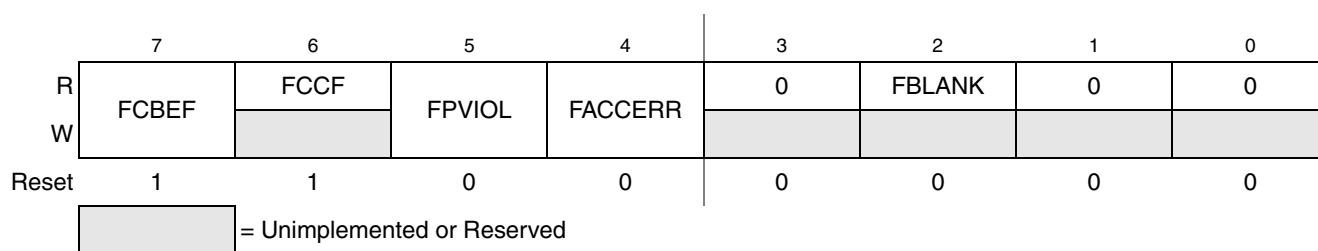


Figure 4-11. Flash Status Register (FSTAT)

Table 4-13. FSTAT Register Field Descriptions

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

Table 4-13. FSTAT Register Field Descriptions (continued)

Field	Description
4 FACCERR	Access Error Flag — FACCERR is set automatically when the proper command sequence is not obeyed exactly (the erroneous command is ignored), if a program or erase operation is attempted before the FCDIV register has been initialized, or if the MCU enters stop while a command was in progress. For a more detailed discussion of the exact actions that are considered access errors, see Section 4.6.5, “Access Errors.” FACCERR is cleared by writing a 1 to FACCERR. Writing a 0 to FACCERR has no meaning or effect. 0 No access error. 1 An access error has occurred.
2 FBLANK	Flash Verified as All Blank (erased) Flag — FBLANK is set automatically at the conclusion of a blank check command if the entire flash array was verified to be erased. FBLANK is cleared by clearing FCBEF to write a new valid command. Writing to FBLANK has no meaning or effect. 0 After a blank check command is completed and FCCF = 1, FBLANK = 0 indicates the flash array is not completely erased. 1 After a blank check command is completed and FCCF = 1, FBLANK = 1 indicates the flash array is completely erased (all 0xFF).

4.8.6 Flash Command Register (FCMD)

Only five command codes are recognized in normal user modes as shown in [Table 4-15](#). Refer to [Section 4.6.3, “Program and Erase Command Execution”](#) for a detailed discussion of flash programming and erase operations.

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

Figure 4-12. Flash Command Register (FCMD)
Table 4-14. FCMD Register Field Descriptions

Field	Description
FCMD[7:0]	Flash Command Bits — See Table 4-15

Table 4-15. Flash Commands

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

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

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

Chapter 5

Resets, Interrupts, and System Configuration

5.1 Introduction

This chapter discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the MC9S08JS16 series. Some interrupt sources from peripheral modules are discussed in greater detail in other chapters of this reference 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 (0xFFFFE:0xFFFF). On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pullup devices disabled. The I bit in the condition code register (CCR) is set to block maskable interrupts so the user program has a chance to initialize the stack pointer (SP) and system control settings. SP is forced to 0x00FF at reset.

The MC9S08JS16 series have following sources for reset:

- Power-on reset (POR)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- 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 used to force a system reset when the application software fails to execute as expected. To prevent a system reset from the COP timer (when it is enabled), application software must reset the COP counter periodically. If the application program gets lost and fails to reset the COP counter before it times out, a system reset is generated to force the system back to a known starting point.

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

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

The COPCLKS bit in SOPT2 (see [Section 5.7.5, “System Options Register 2 \(SOPT2\),”](#) for additional information) selects the clock source used for the COP timer. The clock source options are either the bus clock or an internal 1 kHz clock source. With each clock source, there are three associated time-outs controlled by the COPT bits in SOPT1. [Table 5-6](#) summarizes the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1 kHz 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 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 clock source is selected, the COP counter is re-initialized to zero upon entry to 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 becomes 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 generates the interrupt) so that other interrupts can be serviced without waiting for the first service routine to finish. This practice is not recommended for anyone other than the most experienced programmers because it can lead to subtle program errors that are difficult to debug.

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

NOTE

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

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

5.5.1 Interrupt Stack Frame

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

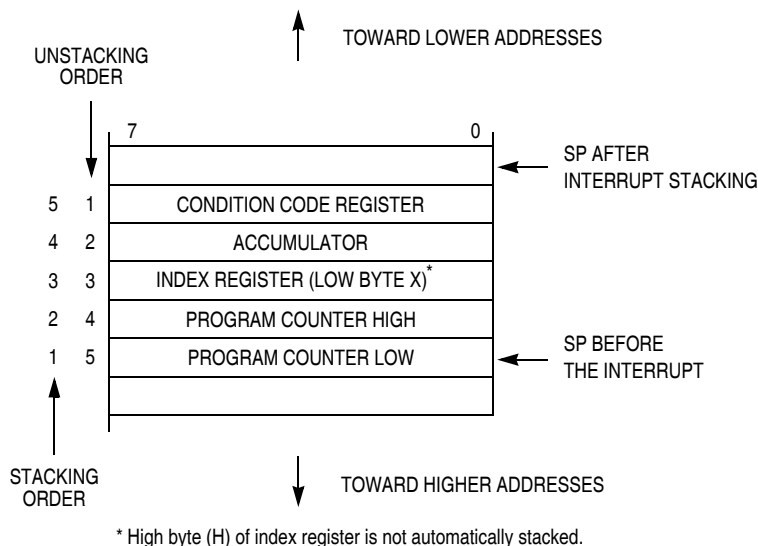


Figure 5-1. Interrupt Stack Frame

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

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

5.5.2 External Interrupt Request (IRQ) Pin

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

5.5.2.1 Pin Configuration Options

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

The IRQ pin, when enabled, defaults to use an internal pull device ($IRQPDD = 0$), the device is a pullup or pulldown depending on the polarity chosen. If the user desires to use an external pullup or pulldown, 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 to V_{DD} . If the IRQ pin is required to drive to a V_{DD} level an external pullup must be used

NOTE

When enabling the IRQ pin for use, the $IRQF$ will be set, and should be cleared prior to enabling the interrupt. When configuring the pin for falling edge and level sensitivity in a 5V system, it is necessary to wait at least 6 cycles between clearing the flag and enabling the interrupt.

5.5.2.2 Edge and Level Sensitivity

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

5.5.3 Interrupt Vectors, Sources, and Local Masks

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

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

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

Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
31 to 30	0xFFC0:FFC1 0xFFC2:FFC3	Unused vector space (available for user program) ¹				
29	0xFFC4:FFC5	Vrtc	System control	RTIF	RTIE	RTC real-time interrupt
28 to 27	0xFFC6:FFC7 0xFFC8:FFC9	Unused vector space (available for user program)				
26	0xFFCA:FFCB	Vmtim	MTIM	TOF	TOIE	MTIM overflow
25	0xFFCC:FFCD	Vkeyboard	KBI	KBF	KBIE	Keyboard pins
24 to 22	0xFFCE:FFCF to 0xFFD2:FFD3	Unused vector space (available for user program)				
21	0xFFD4:FFD5	Vscitx	SCI	TDRE TC	TIE TCIE	SCI transmit
20	0xFFD6:FFD7	Vscirx	SCI	IDLE RDRF	ILIE RIE	SCI receive
19	0xFFD8:FFD9	Vscierr	SCI	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI error
18 to 12	0xFFDA:FFDB to 0xFFE6:FFE7	Unused vector space (available for user program)				
11	0xFFE8:FFE9	Vtpmovf	TPM	TOF	TOIE	TPM overflow
10	0xFFEA:FFEB	Vtpmch1	TPM	CH1F	CH1IE	TPM channel 1
9	0xFFEC:FFED	Vtpmch0	TPM	CH0F	CH0IE	TPM channel 0
8	0xFFEE:FFEF	Unused vector space (available for user program)				
7	0xFFF0:FFF1	Vusb	USB	STALL RESUMEF SLEEPF TOKDNEF SOFTOKF ERRORF USBRSTF	STALL RESUME SLEEP TOKDNE SOFTOK ERROR USBRST	USB Status
6	0xFFF2:FFF3	Unused vector space (available for user program)				
5	0xFFF4:FFF5	Vspi	SPI	SPRF MODF SPTEF SPMF	SPIE SPTIE SPMIE	SPI
4	0xFFF6:FFF7	Vlol	MCG	LOLS	LOLIE	MCG loss of lock
3	0xFFF8:FFF9	Vlvd	System control	LVDF	LVDIE	Low-voltage detect

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

Vector Number	Address (High/Low)	Vector Name	Module	Source	Enable	Description
2	0xFFFFA:FFFB	Virq	IRQ	IRQF	IRQIE	IRQ pin
1	0xFFFFC:FFFD	Vswi	Core	SWI Instruction	—	Software interrupt
0	0xFFFFE:FFFF	Vreset	System control	COP LVD RESET pin Illegal opcode Illegal address LOC POR BDFR	COPE LVDRE — ILOP ILAD CME POR	Watchdog timer Low-voltage detect External pin Illegal opcode Illegal address Loss of clock Power-on-reset BDM-forced reset

¹ Unused vector space is available for use as general flash memory. However, other devices in the S08 family of MCUs may use this location as interrupt vectors. Therefore, care must be taken when using this location if the code will be ported to other MCUs.

5.6 Low-Voltage Detect (LVD) System

The MC9S08JS16 series include a system to protect against low-voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system is composed 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 re-arm voltage level, V_{POR} , the POR circuit will cause a reset condition. As the supply voltage rises, the LVD circuit will hold the MCU in reset until the supply has risen above the low voltage detection low threshold, V_{LVDL} . Both the POR bit and the LVD bit in SRS are set following a POR.

5.6.2 Low-Voltage Detection (LVD) Reset Operation

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

5.6.3 Low-Voltage Warning (LVW) Interrupt Operation

The LVD system has a low voltage warning flag to indicate to the user that the supply voltage is approaching the low voltage condition. When a low voltage warning condition is detected and is

configured for interrupt operation (LVWIE set to 1), LVWF in SPMSC1 will be set and an LVW interrupt request will occur.

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

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

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

Some control bits in the SOPT1 and SPMSC2 registers are related to modes of operation. Although brief descriptions of these bits are provided here, the related functions are discussed in greater detail in [Chapter 3, “Modes of Operation.”](#)

5.7.1 Interrupt Pin Request Status and Control Register (IRQSC)

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

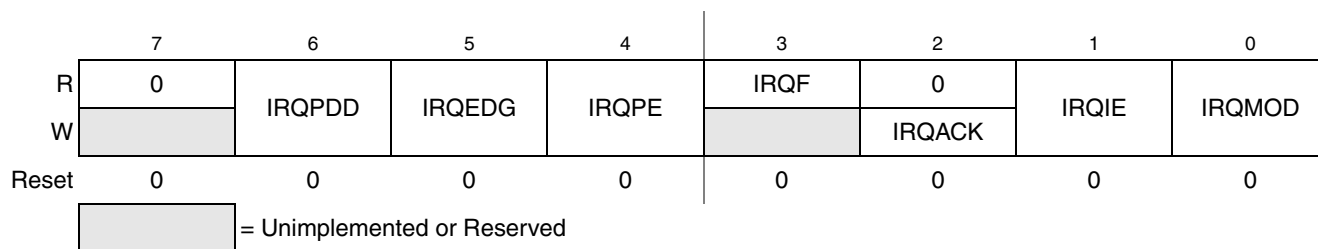


Figure 5-2. Interrupt Request Status and Control Register (IRQSC)

Table 5-2. IRQSC Register Field Descriptions

Field	Description
6 IRQPDD	Interrupt Request (IRQ) Pull Device Disable — This read/write control bit is used to disable the internal pullup device when the IRQ pin is enabled (IRQPE = 1) allowing for an external device to be used. 0 IRQ pull device enabled if IRQPE = 1. 1 IRQ pull device disabled if IRQPE = 1.
5 IRQEDG	Interrupt Request (IRQ) Edge Select — This read/write control bit is used to select the polarity of edges or levels on the IRQ pin that cause IRQF to be set. The IRQMOD control bit determines whether the IRQ pin is sensitive to both edges and levels or only edges. When the IRQ pin is enabled as the IRQ input and is configured to detect rising edges, the optional pullup resistor is re-configured as an optional pulldown resistor. 0 IRQ is falling edge or falling edge/low-level sensitive. 1 IRQ is rising edge or rising edge/high-level sensitive.
4 IRQPE	IRQ Pin Enable — This read/write control bit enables the IRQ pin function. When this bit is set the IRQ pin can be used as an interrupt request. 0 IRQ pin function is disabled. 1 IRQ pin function is enabled.

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

Field	Description
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 will be set. Writing any value to this register address causes a COP reset when the COP is enabled except for the values 0x55 and 0xAA. Writing a 0x55–0xAA sequence to this address clears the COP watchdog timer without affecting the contents of this register. The reset state of these bits depends on what caused the MCU to reset.

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

U = Unaffected by reset

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

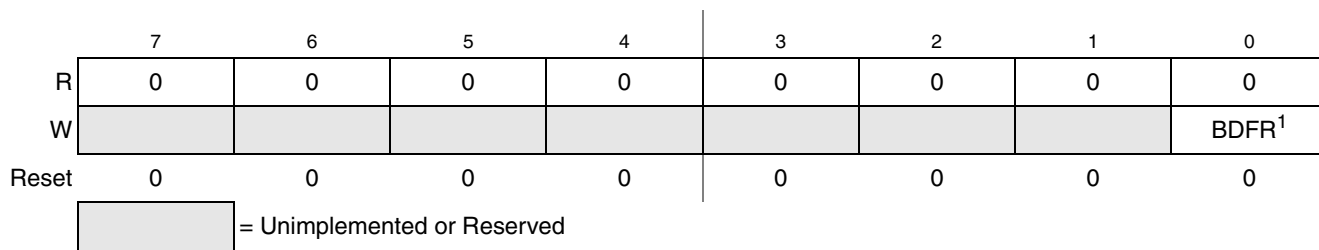
Figure 5-3. System Reset Status (SRS)

Table 5-3. SRS Register Field Descriptions

Field	Description
7 POR	Power-On Reset — Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVR) status bit is also set to indicate that the reset occurred while the internal supply was below the LVR threshold. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	External Reset Pin — Reset was caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 Reset came from external reset pin.
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	Illegal Address — Reset was caused by an attempt to access either data or an instruction at an unimplemented memory address. 0 Reset not caused by an illegal address 1 Reset caused by an illegal address
2 LOC	Loss-of-Clock Reset — Reset was caused by a loss of external clock. 0 Reset not caused by a loss of external clock. 1 Reset caused by a loss of external clock.
1 LVD	Low Voltage Detect — If the LVD is enable with the LVDRE or LVDSE bit is set, and the supply drops below the LVD trip voltage, an LVD reset will occur. This bit is also set by POR. 0 Reset not caused by LVD trip or POR. 1 Reset caused by LVD trip or POR.

5.7.3 System Background Debug Force Reset Register (SBDFR)

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



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

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

Table 5-4. SBDFR Register Field Descriptions

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

5.7.4 System Options Register 1 (SOPT1)

This register may be read at any time. Bits 4 and 3 are unimplemented and always read 0. This is a write-once register so only the first write after reset is honored. Any subsequent attempt to write to SOPT (intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. SOPT must be written during the user's reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

	COPT		STOPE	1	0	BLMSS	BKGDPE	RSTPE
R								
W								
Reset:	1	1	0	1	0	0	1	u ⁽¹⁾
POR:	1	1	0	1	0	0 or 1 ⁽²⁾	1	0
LVR:	1	1	0	1	0	0	1	u

¹ u = unaffected

² Depends on PTB3/ $\overline{\text{BLMS}}$ pin state.

Figure 5-5. System Options Register (SOPT1)
Table 5-5. SOPT1 Register Field Descriptions

Field	Description
7:6 COPT[1:0]	COP Watchdog Timeout — These write-once bits select the timeout period of the COP. COPT along with COPCLKS in SOPT2 defines the COP timeout period. See Table 5-6 .
5 STOPE	Stop Mode Enable — This write-once bit defaults to 0 after reset, which disables stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced. 0 Stop mode disabled. 1 Stop mode enabled.
2 BLMSS	BLMS Pin State — This read only bit indicates PTB3/ $\overline{\text{BLMS}}$ pin state during power-on reset (POR). 0 $\overline{\text{BLMS}}$ pin is high during POR. 1 $\overline{\text{BLMS}}$ pin is low and MS pin is high during POR.

Table 5-5. SOPT1 Register Field Descriptions (continued)

Field	Description
1 BKGDPE	Background Debug Mode Pin Enable — This write-once bit when set enables the PTB2/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 PTB2/BKGD/MS pin functions as PTB2. 1 PTB2/BKGD/MS pin functions as BKGD/MS.
0 RSTPE	RESET Pin Enable — This write-once bit when set enables the PTB1/RESET pin to function as RESET. When clear, the pin functions as one of its alternative functions. This pin defaults to a general-purpose input port function following a POR reset. When configured as RESET, the pin will be unaffected by LVR or other internal resets. When RSTPE is set, an internal pullup device is enabled on RESET. 0 PTB1/RESET pin functions as PTB1. 1 PTB1/RESET pin functions as RESET.

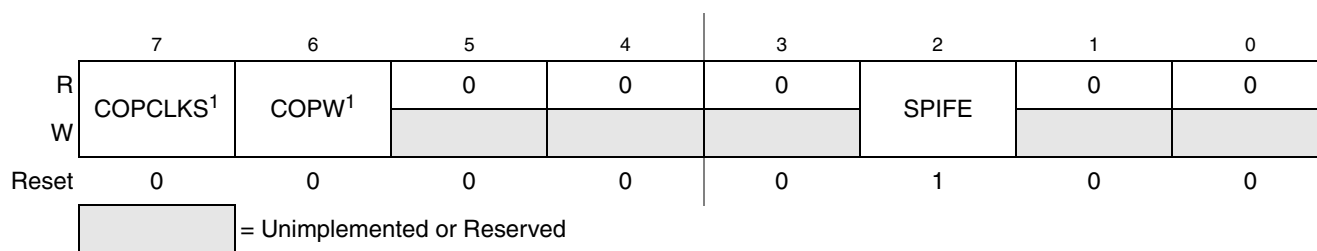
Table 5-6. COP Configuration Options

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

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

² Values shown in milliseconds based on $t_{LPO} = 1$ ms. See t_{LPO} in the *MC9S08JS16 Data Sheet* for the tolerance of this value.

5.7.5 System Options Register 2 (SOPT2)



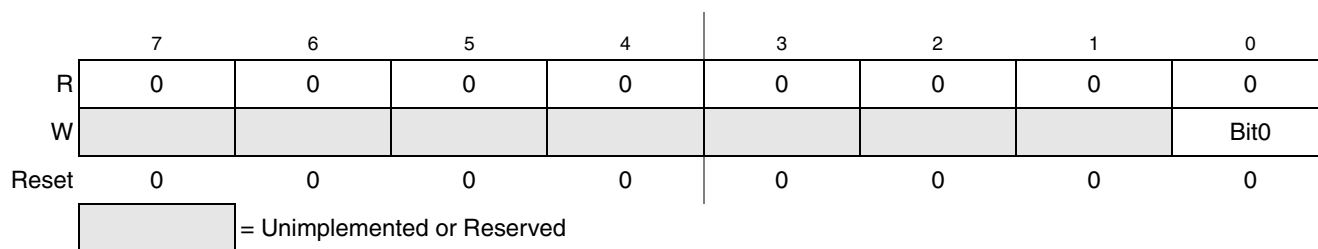
¹ This bit can be written only one time after reset. Additional writes are ignored.

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

Table 5-7. SOPT2 Register Field Descriptions

Field	Description
7 COPCLKS	COP Watchdog Clock Select — This write-once bit selects the clock source of the COP watchdog. 0 Internal 1 kHz clock is source to COP. 1 Bus clock is source to COP.
6 COPW	COP Window — This write-once bit selects the COP operation mode. When set, the 0x55–0xAA write sequence to the SRS register must occur in the last 25% of the selected period. Any write to the SRS register during the first 75% of the selected period will reset the MCU. 0 Normal COP operation. 1 Window COP operation.
2 SPIFE	SPI1 Ports Input Filter Enable 0 Disable input filter on SPI port pins to allow for higher maximum SPI baud rate. 1 Enable input filter on SPI port pins to eliminate noise and restrict maximum SPI baud rate.

5.7.6 Flash Protection Defeat Register (FPROTD)


Figure 5-7. Flash Protection Defeat Register (FPROTD)
Table 5-8. FPROTD Register Field Descriptions

Field	Description
0	Protected flash section can be disabled by write 0x55 and 0xAA back to back to the address of FPROTD first, then set bit FPDIS = 1 in FPROT register. This bit set to 1 only after write 0x55 and 0xAA back to back successfully that means FPDIS can be set to 1. Any write operation, will clear this bit except 0x55–AA write squnce. 0 The bit FPDIS in FPROT can not be set to 1. 1 The bit FPDIS in FPROT can be set to 1.

5.7.7 SIGNATURE Register (SIGNATURE)

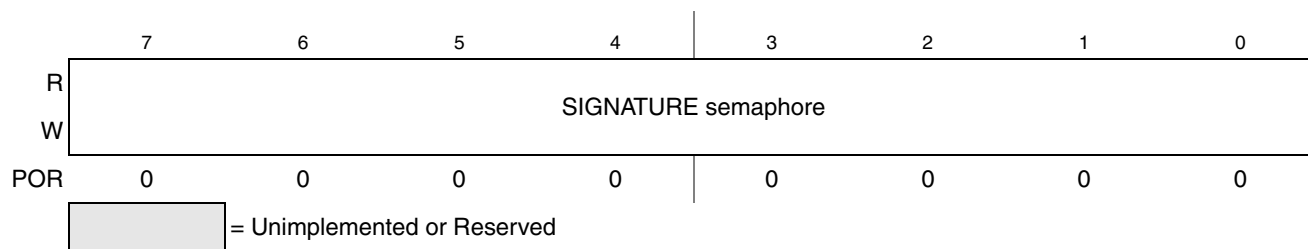
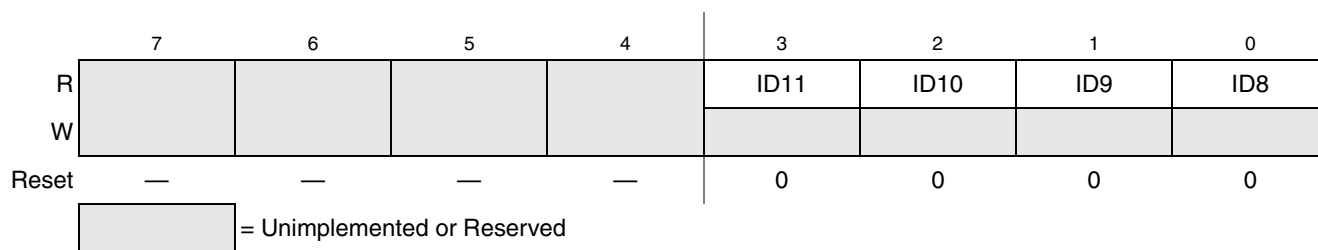

Figure 5-8. SIGNATURE Register (SIGNATURE)

Table 5-9. SIGNATURE Register Field Descriptions

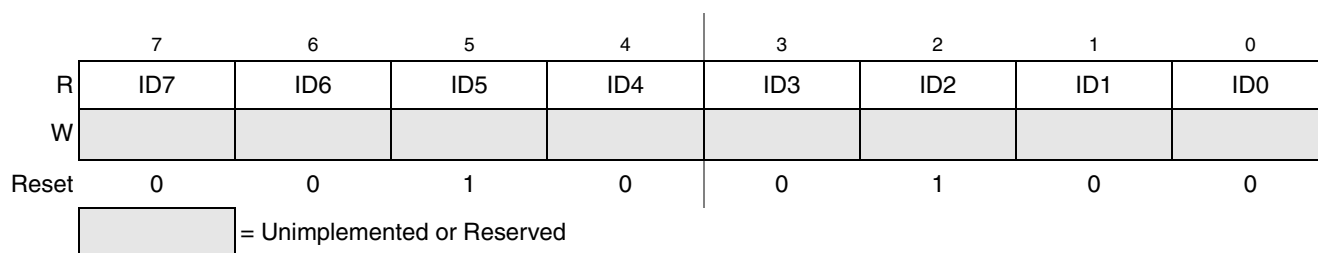
Field	Description
7:0	The SIGNATURE semaphore is used as the semaphore to jump to bootloader mode or not after regular reset. This byte only can be cleared by power-on reset (POR), content retains after regular reset.

5.7.8 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.


Figure 5-9. System Device Identification Register — High (SDIDH)
Table 5-10. 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 MC9S08JS16 series are hard coded to the value 0x024. See also ID bits in Table 5-11 .


Figure 5-10. System Device Identification Register — Low (SDIDL)
Table 5-11. 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 MC9S08JS16 series are hard coded to the value 0x024. See also ID bits in Table 5-10 .

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

R	LVWF ¹	0	LVWIE	LVDRE ²	LVDSE	LVDE ²	0	0
W		LVWACK						
Reset:	0	0	0	1	1	1	0	0

= Unimplemented or Reserved

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

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

Figure 5-11. System Power Management Status and Control 1 Register (SPMSC1)

Table 5-12. SPMSC1 Register Field Descriptions

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

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

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

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

= Unimplemented or Reserved
 u = Unaffected by reset

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

Figure 5-12. System Power Management Status and Control 2 Register (SPMSC2)

Table 5-13. SPMSC2 Register Field Descriptions

Field	Description
5 LVDV	Low-Voltage Detect Voltage Select — This bit selects the low voltage detect (LVD) trip point setting. It also selects the warning voltage range. See Table 5-14 .
4 LVWV	Low-Voltage Warning Voltage Select — This bit selects the low voltage warning (LVW) trip point voltage. See Table 5-14 .
3 PPDF	Partial Power Down Flag — This read-only status bit indicates that the MCU has recovered from stop2 mode. 0 MCU has not recovered from stop2 mode. 1 MCU recovered from stop2 mode.
2 PPDACK	Partial Power Down Acknowledge — Writing a 1 to PPDACK clears the PPDF bit
0 PPDC	Partial Power Down Control — This write-once bit controls whether stop2 or stop3 mode is selected. 0 Stop3 mode enabled. 1 Stop2, partial power down, mode enabled.

Table 5-14. LVD and LVW trip point typical values¹

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

¹ See MC9S08JS16 Series *Data Sheet* for minimum and maximum values.

Chapter 6

Parallel Input/Output

6.1 Introduction

This chapter explains software controls related to parallel input/output (I/O). The MC9S08JS16 has two I/O ports which include a total of 14 general-purpose I/O pins. See [Chapter 2, “Pins and Connections,”](#) for more information about the logic and hardware aspects of these pins.

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 pins 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

To avoid extra current drain from floating input pins, the user’s reset initialization routine in the application program must either enable on-chip pullup devices or change the direction of unconnected pins to outputs so the pins do not float.

NOTE

When PTB0 is configured as output pin, it’s open-drain output.

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.

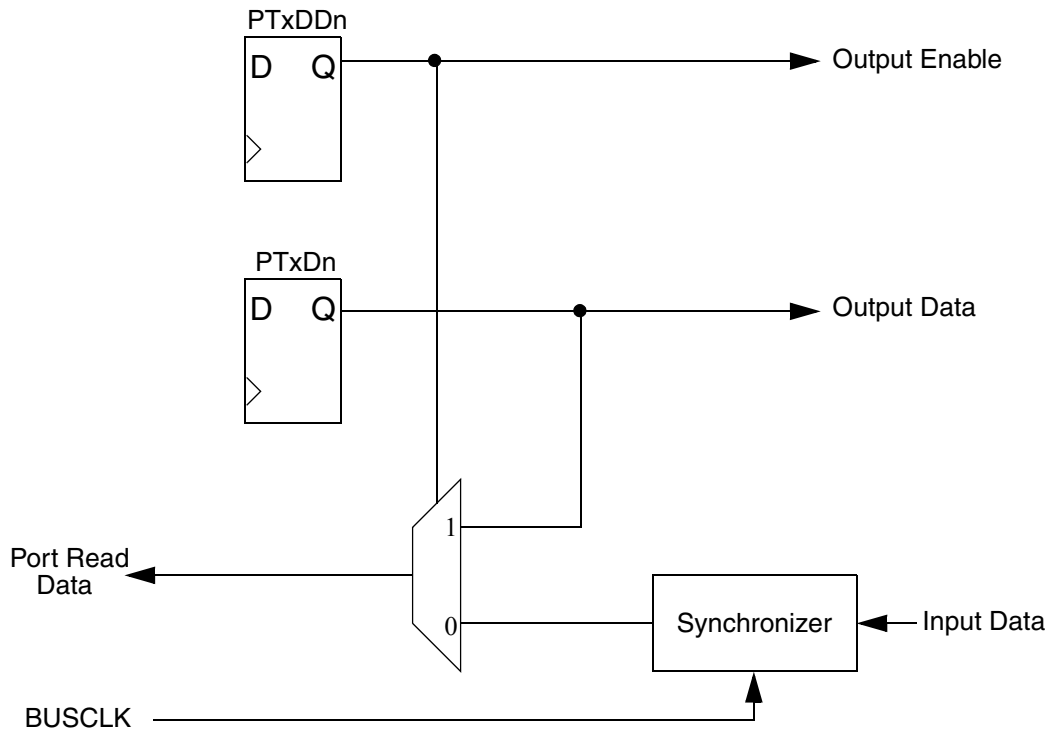


Figure 6-1. Parallel I/O Block Diagram

The data direction control bits determine whether the pin output driver is enabled. 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 bit which has 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.

Write to the port data register before changing the direction of a port pin to output. This ensures that the pin will not be driven momentarily with an old data value that happened to be in the port data register.

6.3 Pin Control

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

6.3.1 Internal Pullup Enable

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

6.3.2 Output Slew Rate Control Enable

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

6.3.3 Output Drive Strength Select

An output pin can be selected to have high output drive strength by setting the corresponding bit in one of the drive strength select registers (PTxDSn). When high drive is selected, the 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 register states must be restored from the values saved in RAM before the STOP instruction was executed. Peripherals may require initialization or restoration 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 circuitry 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 of 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.

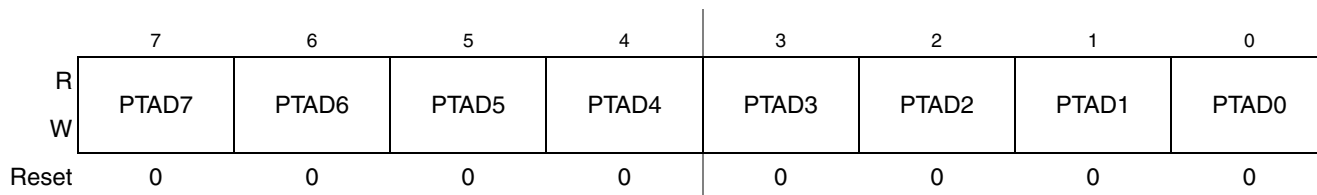


Figure 6-2. Port A Data Register (PTAD)

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.

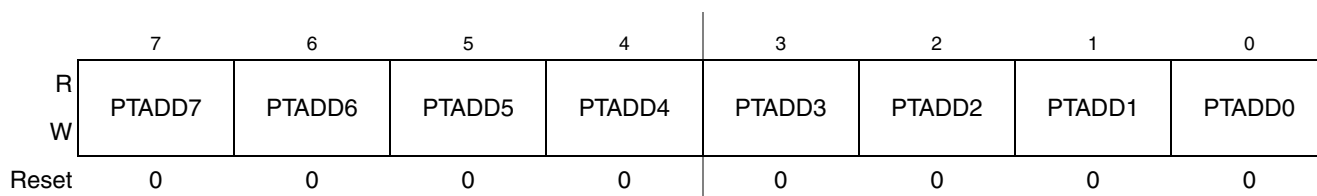


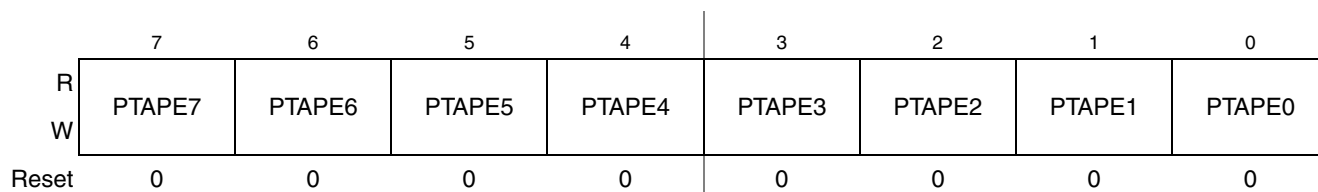
Figure 6-3. Data Direction for Port A Register (PTADD)

Table 6-2. PTADD Register Field Descriptions

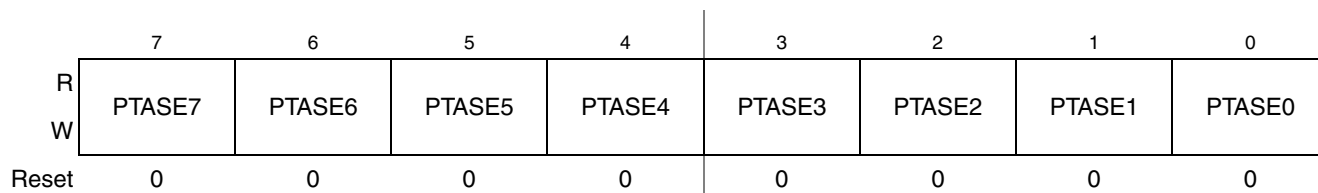
Field	Description
7:0 PTADD[7:0]	Data Direction for Port A Bits — These read/write bits control the direction of port A pins and what is read for PTAD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port A bit n and PTAD reads return the contents of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

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

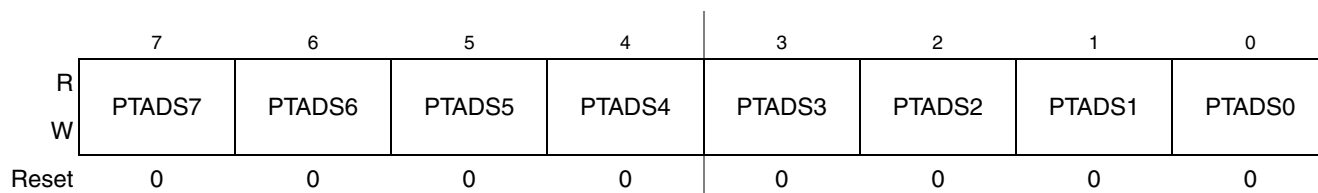
In addition to the I/O control, port A pins are controlled by the registers listed below.


Figure 6-4. Internal Pullup Enable for Port A (PTAPE)
Table 6-3. PTAPE Register Field Descriptions

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


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.


Figure 6-6. Output Drive Strength Selection for Port A (PTADS)
Table 6-5. PTADS 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.

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

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

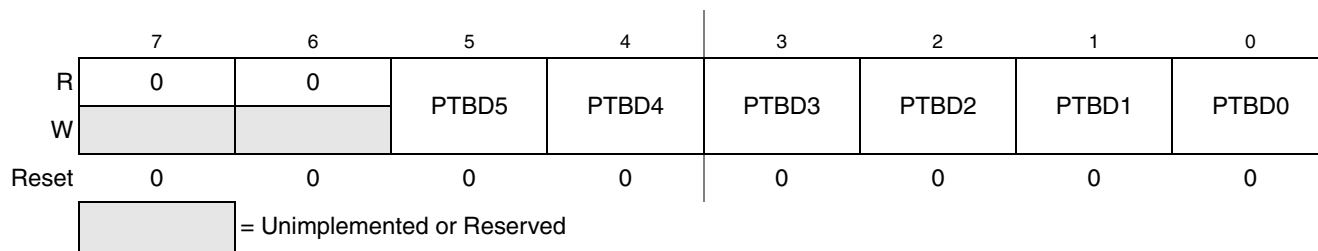


Figure 6-7. Port B Data Register (PTBD)

Table 6-6. PTBD Register Field Descriptions

Field	Description
5:0 PTBD[5:0]	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.

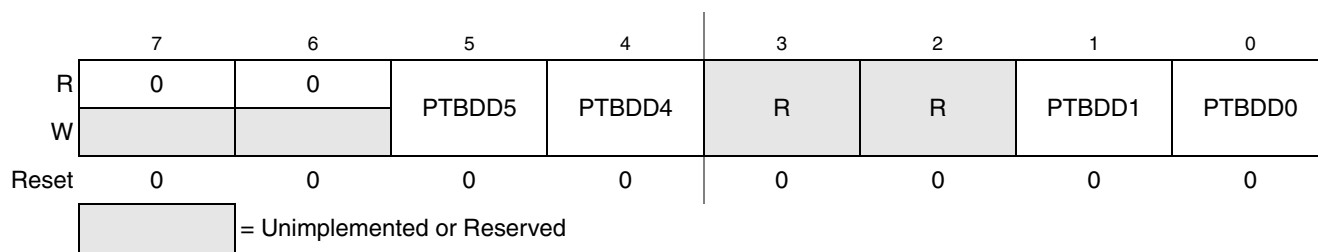


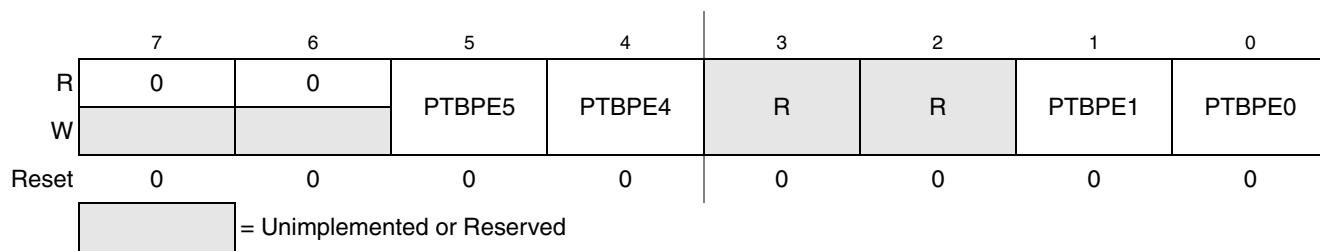
Figure 6-8. Data Direction for Port B (PTBDD)

Table 6-7. PTBDD Register Field Descriptions

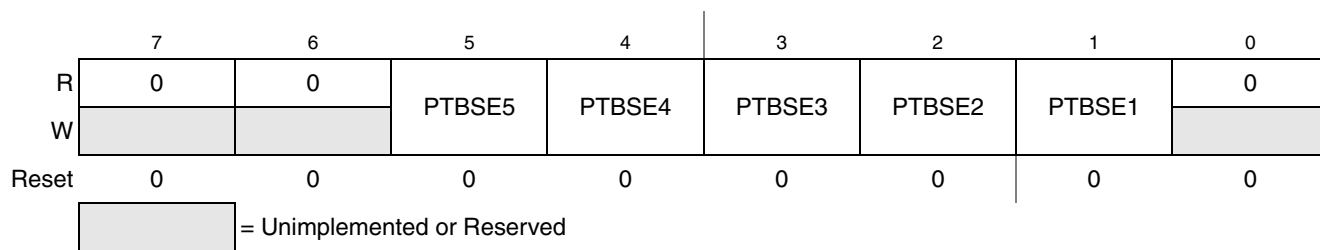
Field	Description
5:4, 1:0 PTBDD[5:4, 1:0]	Data Direction for Port B Bits — These read/write bits control the direction of port B pins and what is read for PTBD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port B bit n and PTBD reads return the contents of PTBDn.

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

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


Figure 6-9. Internal Pullup Enable for Port B (PTBPE)
Table 6-8. PTBPE Register Field Descriptions

Field	Description
5:4, 1:0 PTBPE[5:4, 1:0]	Internal Pullup Enable for Port B Bits — Each of these control bits determines if the internal pullup device is enabled for the associated PTB pin. For port B pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port B bit n. 1 Internal pullup device enabled for port B bit n.


Figure 6-10. Output Slew Rate Control Enable (PTBSE)
Table 6-9. PTBSE Register Field Descriptions

Field	Description
5:0 PTBSE[5:1]	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.

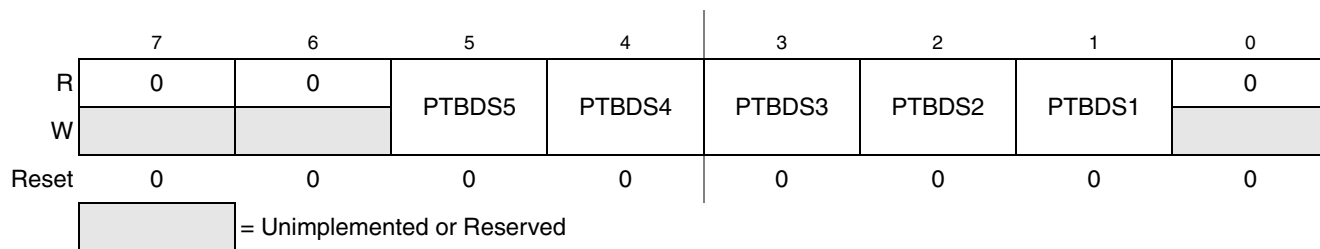

Figure 6-11. Output Drive Strength Selection for Port B (PTBDS)

Table 6-10. PTBDS Register Field Descriptions

Field	Description
5:0 PTBDS[5:1]	<p>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.</p> <p>0 Low output drive enabled for port B bit n.</p> <p>1 High output drive enabled for port B bit n.</p>

Chapter 7

Central Processor Unit (S08CPUV2)

7.1 Introduction

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

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

7.1.1 Features

Features of the HCS08 CPU include:

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

7.2 Programmer's Model and CPU Registers

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

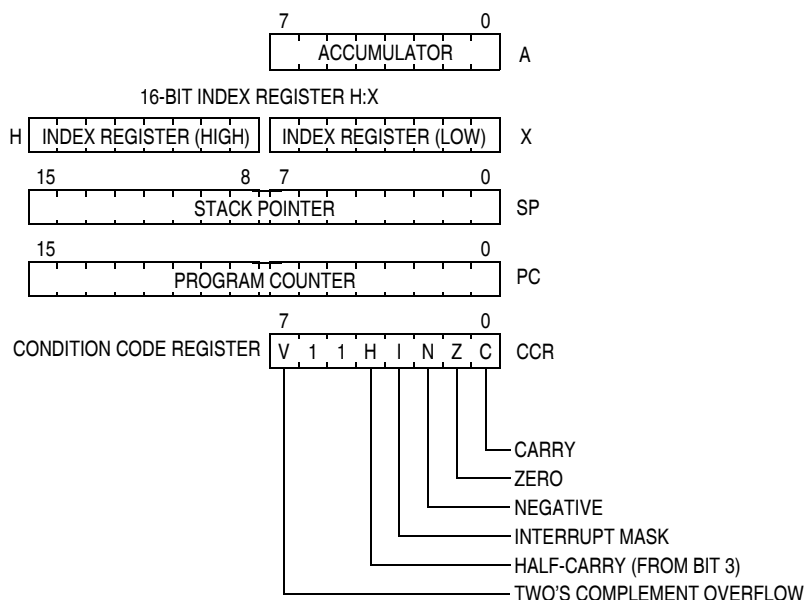


Figure 7-1. CPU Registers

7.2.1 Accumulator (A)

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

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

7.2.2 Index Register (H:X)

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

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

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

7.2.3 Stack Pointer (SP)

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

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

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

7.2.4 Program Counter (PC)

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

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

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

7.2.5 Condition Code Register (CCR)

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

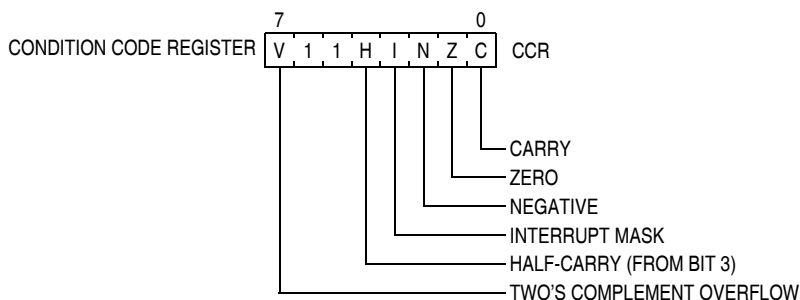


Figure 7-2. Condition Code Register

Table 7-1. CCR Register Field Descriptions

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

7.3 Addressing Modes

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

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

7.3.1 Inherent Addressing Mode (INH)

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

7.3.2 Relative Addressing Mode (REL)

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

7.3.3 Immediate Addressing Mode (IMM)

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

7.3.4 Direct Addressing Mode (DIR)

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

7.3.5 Extended Addressing Mode (EXT)

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

7.3.6 Indexed Addressing Mode

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

7.3.6.1 Indexed, No Offset (IX)

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

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

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

7.3.6.3 Indexed, 8-Bit Offset (IX1)

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

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

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

7.3.6.5 Indexed, 16-Bit Offset (IX2)

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

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

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

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

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

7.4 Special Operations

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

7.4.1 Reset Sequence

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

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

7.4.2 Interrupt Sequence

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

The CPU sequence for an interrupt is:

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

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

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

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

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

7.4.3 Wait Mode Operation

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

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

7.4.4 Stop Mode Operation

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

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

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

7.4.5 BGND Instruction

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

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

7.5 HCS08 Instruction Set Summary

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

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

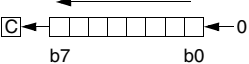
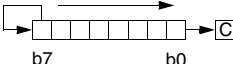
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR				
						VH	I	N	Z	C
ADC #opr8i ADC opr8a ADC opr16a ADC oprx16,X ADC oprx8,X ADC ,X ADC oprx16,SP ADC oprx8,SP	Add with Carry $A \leftarrow (A) + (M) + (C)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 ii B9 dd C9 hh ll D9 ee ff E9 ff F9 9E D9 ee ff 9E E9 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↑↑	-	↑↑	↑	↑
ADD #opr8i ADD opr8a ADD opr16a ADD oprx16,X ADD oprx8,X ADD ,X ADD oprx16,SP ADD oprx8,SP	Add without Carry $A \leftarrow (A) + (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	AB ii BB dd CB hh ll DB ee ff EB ff FB 9E DB ee ff 9E EB ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↑↑	-	↑↑	↑	↑
AIS #opr8i	Add Immediate Value (Signed) to Stack Pointer $SP \leftarrow (SP) + (M)$	IMM	A7 ii	2	pp	--	-	-	-	-
AIX #opr8i	Add Immediate Value (Signed) to Index Register (H:X) $H:X \leftarrow (H:X) + (M)$	IMM	AF ii	2	pp	--	-	-	-	-
AND #opr8i AND opr8a AND opr16a AND oprx16,X AND oprx8,X AND ,X AND oprx16,SP AND oprx8,SP	Logical AND $A \leftarrow (A) \& (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 ii B4 dd C4 hh ll D4 ee ff E4 ff F4 9E D4 ee ff 9E E4 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0-	-	↑	↑	-
ASL opr8a ASLA ASLX ASL oprx8,X ASL ,X ASL oprx8,SP	Arithmetic Shift Left  (Same as LSL)	DIR INH INH IX1 IX SP1	38 dd 48 58 68 ff 78 9E 68 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↑-	-	↑	↑	↑
ASR opr8a ASRA ASRX ASR oprx8,X ASR ,X ASR oprx8,SP	Arithmetic Shift Right 	DIR INH INH IX1 IX SP1	37 dd 47 57 67 ff 77 9E 67 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↑-	-	↑	↑	↑
BCC rel	Branch if Carry Bit Clear (if C = 0)	REL	24 rr	3	ppp	--	-	-	-	-

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR				
						VH	I	N	Z	C
BCLR <i>n,opr8a</i>	Clear Bit <i>n</i> in Memory ($M_n \leftarrow 0$)	DIR (b0)	11 dd	5	rfwpp	--	-	-	-	-
		DIR (b1)	13 dd	5	rfwpp					
		DIR (b2)	15 dd	5	rfwpp					
		DIR (b3)	17 dd	5	rfwpp					
		DIR (b4)	19 dd	5	rfwpp					
		DIR (b5)	1B dd	5	rfwpp					
		DIR (b6)	1D dd	5	rfwpp					
		DIR (b7)	1F dd	5	rfwpp					
BCS <i>rel</i>	Branch if Carry Bit Set (if $C = 1$) (Same as BLO)	REL	25 rr	3	ppp	--	-	-	-	-
BEQ <i>rel</i>	Branch if Equal (if $Z = 1$)	REL	27 rr	3	ppp	--	-	-	-	-
BGE <i>rel</i>	Branch if Greater Than or Equal To (if $N \oplus V = 0$) (Signed)	REL	90 rr	3	ppp	--	-	-	-	-
BGND	Enter active background if ENBDM=1 Waits for and processes BDM commands until GO, TRACE1, or TAGGO	INH	82	5+	fp...ppp	--	-	-	-	-
BGT <i>rel</i>	Branch if Greater Than (if $Z \mid (N \oplus V) = 0$) (Signed)	REL	92 rr	3	ppp	--	-	-	-	-
BHCC <i>rel</i>	Branch if Half Carry Bit Clear (if $H = 0$)	REL	28 rr	3	ppp	--	-	-	-	-
BHCS <i>rel</i>	Branch if Half Carry Bit Set (if $H = 1$)	REL	29 rr	3	ppp	--	-	-	-	-
BHI <i>rel</i>	Branch if Higher (if $C \mid Z = 0$)	REL	22 rr	3	ppp	--	-	-	-	-
BHS <i>rel</i>	Branch if Higher or Same (if $C = 0$) (Same as BCC)	REL	24 rr	3	ppp	--	-	-	-	-
BIH <i>rel</i>	Branch if IRQ Pin High (if IRQ pin = 1)	REL	2F rr	3	ppp	--	-	-	-	-
BIL <i>rel</i>	Branch if IRQ Pin Low (if IRQ pin = 0)	REL	2E rr	3	ppp	--	-	-	-	-
BIT # <i>opr8i</i> BIT <i>opr8a</i> BIT <i>opr16a</i> BIT <i>opr16,X</i> BIT <i>opr8,X</i> BIT <i>,X</i> BIT <i>opr16,SP</i> BIT <i>opr8,SP</i>	Bit Test (A) & (M) (CCR Updated but Operands Not Changed)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 ii B5 dd C5 hh ll D5 ee ff E5 ff F5 9E D5 ee ff 9E E5 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0-	-	↑	↓	-
BLE <i>rel</i>	Branch if Less Than or Equal To (if $Z \mid (N \oplus V) = 1$) (Signed)	REL	93 rr	3	ppp	--	-	-	-	-
BLO <i>rel</i>	Branch if Lower (if $C = 1$) (Same as BCS)	REL	25 rr	3	ppp	--	-	-	-	-
BLS <i>rel</i>	Branch if Lower or Same (if $C \mid Z = 1$)	REL	23 rr	3	ppp	--	-	-	-	-
BLT <i>rel</i>	Branch if Less Than (if $N \oplus V = 1$) (Signed)	REL	91 rr	3	ppp	--	-	-	-	-
BMC <i>rel</i>	Branch if Interrupt Mask Clear (if $I = 0$)	REL	2C rr	3	ppp	--	-	-	-	-
BMI <i>rel</i>	Branch if Minus (if $N = 1$)	REL	2B rr	3	ppp	--	-	-	-	-
BMS <i>rel</i>	Branch if Interrupt Mask Set (if $I = 1$)	REL	2D rr	3	ppp	--	-	-	-	-
BNE <i>rel</i>	Branch if Not Equal (if $Z = 0$)	REL	26 rr	3	ppp	--	-	-	-	-
BPL <i>rel</i>	Branch if Plus (if $N = 0$)	REL	2A rr	3	ppp	--	-	-	-	-

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR				
						VH	I	N	Z	C
BRA <i>rel</i>	Branch Always (if I = 1)	REL	20 rr	3	ppp	--	---	---	---	---
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Clear (if (Mn) = 0)	DIR (b0)	01 dd rr	5	rpppp	--	---	---	---	↑
		DIR (b1)	03 dd rr	5	rpppp					
		DIR (b2)	05 dd rr	5	rpppp					
		DIR (b3)	07 dd rr	5	rpppp					
		DIR (b4)	09 dd rr	5	rpppp					
		DIR (b5)	0B dd rr	5	rpppp					
		DIR (b6)	0D dd rr	5	rpppp					
		DIR (b7)	0F dd rr	5	rpppp					
BRN <i>rel</i>	Branch Never (if I = 0)	REL	21 rr	3	ppp	--	---	---	---	---
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Set (if (Mn) = 1)	DIR (b0)	00 dd rr	5	rpppp	--	---	---	---	↑
		DIR (b1)	02 dd rr	5	rpppp					
		DIR (b2)	04 dd rr	5	rpppp					
		DIR (b3)	06 dd rr	5	rpppp					
		DIR (b4)	08 dd rr	5	rpppp					
		DIR (b5)	0A dd rr	5	rpppp					
		DIR (b6)	0C dd rr	5	rpppp					
		DIR (b7)	0E dd rr	5	rpppp					
BSET <i>n,opr8a</i>	Set Bit <i>n</i> in Memory (Mn ← 1)	DIR (b0)	10 dd	5	r fwpp	--	---	---	---	---
		DIR (b1)	12 dd	5	r fwpp					
		DIR (b2)	14 dd	5	r fwpp					
		DIR (b3)	16 dd	5	r fwpp					
		DIR (b4)	18 dd	5	r fwpp					
		DIR (b5)	1A dd	5	r fwpp					
		DIR (b6)	1C dd	5	r fwpp					
		DIR (b7)	1E dd	5	r fwpp					
BSR <i>rel</i>	Branch to Subroutine PC ← (PC) + 0x0002 push (PCL); SP ← (SP) – 0x0001 push (PCH); SP ← (SP) – 0x0001 PC ← (PC) + <i>rel</i>	REL	AD rr	5	ssppp	--	---	---	---	---
CBEQ <i>opr8a,rel</i>	Compare and... Branch if (A) = (M)	DIR	31 dd rr	5	rpppp	--	---	---	---	---
CBEQA # <i>opr8i,rel</i>		IMM	41 ii rr	4	pppp					
CBEQX # <i>opr8i,rel</i>		IMM	51 ii rr	4	pppp					
CBEQ <i>opr8,X+,rel</i>		IX1+	61 ff rr	5	rpppp					
CBEQ <i>,X+,rel</i>		IX+	71 rr	5	r fppp					
CBEQ <i>opr8,SP,rel</i>		SP1	9E 61 ff rr	6	prpppp					
CLC	Clear Carry Bit (C ← 0)	INH	98	1	p	--	---	---	---	0
CLI	Clear Interrupt Mask Bit (I ← 0)	INH	9A	1	p	--	0	---	---	---
CLR <i>opr8a</i>	Clear M ← 0x00 A ← 0x00 X ← 0x00 H ← 0x00 M ← 0x00 M ← 0x00 M ← 0x00	DIR	3F dd	5	r fwpp	0	-	-	0	1
CLRA		INH	4F	1	p					
CLR X		INH	5F	1	p					
CLR H		INH	8C	1	p					
CLR <i>opr8,X</i>		IX1	6F ff	5	r fwpp					
CLR <i>,X</i>		IX	7F	4	r fwp					
CLR <i>opr8,SP</i>		SP1	9E 6F ff	6	prfwpp					

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR					
						VH	I	N	Z	C	
CMP #opr8i CMP opr8a CMP opr16a CMP oprx16,X CMP oprx8,X CMP ,X CMP oprx16,SP CMP oprx8,SP	Compare Accumulator with Memory A – M (CCR Updated But Operands Not Changed)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A1 ii B1 dd C1 hh ll D1 ee ff E1 ff F1 9E D1 ee ff 9E E1 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↓- ↓- ↓- ↓- ↓- ↓- ↓- ↓-	-	↑	↑	↑	↑
COM opr8a COMA COMX COM oprx8,X COM ,X COM oprx8,SP	Complement (One's Complement) M ← (\overline{M}) = 0xFF – (M) A ← (\overline{A}) = 0xFF – (A) X ← (\overline{X}) = 0xFF – (X) M ← (\overline{M}) = 0xFF – (M) M ← (\overline{M}) = 0xFF – (M) M ← (\overline{M}) = 0xFF – (M)	DIR INH INH IX1 IX SP1	33 dd 43 53 63 ff 73 9E 63 ff	5 1 1 5 4 6	rffwpp p p rffwpp rfwp prffwpp	0- 0- 0- 0- 0- 0-	-	↑	↑	↑	1
CPHX opr16a CPHX #opr16i CPHX opr8a CPHX oprx8,SP	Compare Index Register (H:X) with Memory (H:X) – (M:M + 0x0001) (CCR Updated But Operands Not Changed)	EXT IMM DIR SP1	3E hh ll 65 jj kk 75 dd 9E F3 ff	6 3 5 6	prrfpp ppp rrfpp prrfpp	↓- ↓- ↓- ↓-	-	↑	↑	↑	↑
CPX #opr8i CPX opr8a CPX opr16a CPX oprx16,X CPX oprx8,X CPX ,X CPX oprx16,SP CPX oprx8,SP	Compare X (Index Register Low) with Memory X – M (CCR Updated But Operands Not Changed)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A3 ii B3 dd C3 hh ll D3 ee ff E3 ff F3 9E D3 ee ff 9E E3 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↓- ↓- ↓- ↓- ↓- ↓- ↓- ↓-	-	↑	↑	↑	↑
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	INH	72	1	p	U-	-	↑	↑	↑	↑
DBNZ opr8a,rel DBNZA rel DBNZX rel DBNZ oprx8,X,rel DBNZ ,X,rel DBNZ oprx8,SP,rel	Decrement A, X, or M and Branch if Not Zero (if (result) ≠ 0) DBNZX Affects X Not H	DIR INH INH IX1 IX SP1	3B dd rr 4B rr 5B rr 6B ff rr 7B rr 9E 6B ff rr	7 4 4 7 6 8	rffwpppp fppp fppp rffwpppp rffwppp prffwpppp	-- -- -- -- -- --	-	-	-	-	-
DEC opr8a DECA DECX DEC oprx8,X DEC ,X DEC oprx8,SP	Decrement M ← (M) – 0x01 A ← (A) – 0x01 X ← (X) – 0x01 M ← (M) – 0x01 M ← (M) – 0x01 M ← (M) – 0x01	DIR INH INH IX1 IX SP1	3A dd 4A 5A 6A ff 7A 9E 6A ff	5 1 1 5 4 6	rffwpp p p rffwpp rffw prffwpp	↓- ↓- ↓- ↓- ↓- ↓-	-	↑	↑	↑	-
DIV	Divide A ← (H:A) ÷ (X); H ← Remainder	INH	52	6	fffffp	--	-	-	↑	↑	↑
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- 0- 0- 0- 0- 0- 0- 0-	-	↑	↑	-	

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

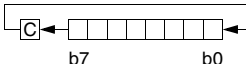
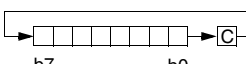
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR			
						VH	I	N	ZC
MOV <i>opr8a,opr8a</i> MOV <i>opr8a,X+</i> MOV <i>#opr8i,opr8a</i> MOV <i>,X+,opr8a</i>	Move $(M)_{\text{destination}} \leftarrow (M)_{\text{source}}$ In IX+/DIR and DIR/IX+ Modes, $H:X \leftarrow (H:X) + 0x0001$	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E dd dd 5E dd 6E ii dd 7E dd	5 5 4 5	rwppp rfwpp pwpp rfwpp	0-	-	↑	↓
MUL	Unsigned multiply $X:A \leftarrow (X) \times (A)$	INH	42	5	fffffp	-0	-	-	0
NEG <i>opr8a</i> NEGA NEGX NEG <i>opr8,X</i> NEG <i>,X</i> NEG <i>opr8,SP</i>	Negate $M \leftarrow (M) = 0x00 - (M)$ (Two's Complement) $A \leftarrow (A) = 0x00 - (A)$ $X \leftarrow (X) = 0x00 - (X)$ $M \leftarrow (M) = 0x00 - (M)$ $M \leftarrow (M) = 0x00 - (M)$ $M \leftarrow (M) = 0x00 - (M)$	DIR INH INH IX1 IX SP1	30 dd 40 50 60 ff 70 9E 60 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↑-	-	↑	↓
NOP	No Operation — Uses 1 Bus Cycle	INH	9D	1	p	--	-	-	-
NSA	Nibble Swap Accumulator $A \leftarrow (A[3:0]:A[7:4])$	INH	62	1	p	--	-	-	-
ORA <i>#opr8i</i> ORA <i>opr8a</i> ORA <i>opr16a</i> ORA <i>opr16,X</i> ORA <i>opr8,X</i> ORA <i>,X</i> ORA <i>opr16,SP</i> ORA <i>opr8,SP</i>	Inclusive OR Accumulator and Memory $A \leftarrow (A) (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA ii BA dd CA hh ll DA ee ff EA ff FA 9E DA ee ff 9E EA ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0-	-	↑	↓
PSHA	Push Accumulator onto Stack $\text{Push } (A); SP \leftarrow (SP) - 0x0001$	INH	87	2	sp	--	-	-	-
PSHH	Push H (Index Register High) onto Stack $\text{Push } (H); SP \leftarrow (SP) - 0x0001$	INH	8B	2	sp	--	-	-	-
PSHX	Push X (Index Register Low) onto Stack $\text{Push } (X); SP \leftarrow (SP) - 0x0001$	INH	89	2	sp	--	-	-	-
PULA	Pull Accumulator from Stack $SP \leftarrow (SP + 0x0001); \text{Pull } (A)$	INH	86	3	ufp	--	-	-	-
PULH	Pull H (Index Register High) from Stack $SP \leftarrow (SP + 0x0001); \text{Pull } (H)$	INH	8A	3	ufp	--	-	-	-
PULX	Pull X (Index Register Low) from Stack $SP \leftarrow (SP + 0x0001); \text{Pull } (X)$	INH	88	3	ufp	--	-	-	-
ROL <i>opr8a</i> ROLA ROLX ROL <i>opr8,X</i> ROL <i>,X</i> ROL <i>opr8,SP</i>	Rotate Left through Carry 	DIR INH INH IX1 IX SP1	39 dd 49 59 69 ff 79 9E 69 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↑-	-	↑	↓
ROR <i>opr8a</i> RORA RORX ROR <i>opr8,X</i> ROR <i>,X</i> ROR <i>opr8,SP</i>	Rotate Right through Carry 	DIR INH INH IX1 IX SP1	36 dd 46 56 66 ff 76 9E 66 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↑-	-	↑	↓

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR				
						VH	I	N	Z	C
RSP	Reset Stack Pointer (Low Byte) SPL ← 0xFF (High Byte Not Affected)	INH	9C	1	p	--	---	---	---	---
RTI	Return from Interrupt SP ← (SP) + 0x0001; Pull (CCR) SP ← (SP) + 0x0001; Pull (A) SP ← (SP) + 0x0001; Pull (X) SP ← (SP) + 0x0001; Pull (PCH) SP ← (SP) + 0x0001; Pull (PCL)	INH	80	9	uuuuufppp	↑↑		↑↑↑↑		
RTS	Return from Subroutine SP ← SP + 0x0001; Pull (PCH) SP ← SP + 0x0001; Pull (PCL)	INH	81	5	ufppp	--		---	---	---
SBC #opr8i SBC opr8a SBC opr16a SBC oprx16,X SBC oprx8,X SBC ,X SBC oprx16,SP SBC oprx8,SP	Subtract with Carry A ← (A) – (M) – (C)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A2 ii B2 dd C2 hh ll D2 ee ff E2 ff F2 9E D2 ee ff 9E E2 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rpp pprpp prpp			↑-	-↑↑↑	
SEC	Set Carry Bit (C ← 1)	INH	99	1	p	--		---	---	1
SEI	Set Interrupt Mask Bit (I ← 1)	INH	9B	1	p	--	1	---	---	---
STA opr8a STA opr16a STA oprx16,X STA oprx8,X STA ,X STA oprx16,SP STA oprx8,SP	Store Accumulator in Memory M ← (A)	DIR EXT IX2 IX1 IX SP2 SP1	B7 dd C7 hh ll D7 ee ff E7 ff F7 9E D7 ee ff 9E E7 ff	3 4 4 3 2 5 4	wpp pwpp pwpp wpp wp ppwpp pwpp		0-	-↑↑-		
STHX opr8a STHX opr16a STHX oprx8,SP	Store H:X (Index Reg.) (M:M + 0x0001) ← (H:X)	DIR EXT SP1	35 dd 96 hh ll 9E FF ff	4 5 5	wpp pwpp pwpp		0-	-↑↑-		
STOP	Enable Interrupts: Stop Processing Refer to MCU Documentation I bit ← 0; Stop Processing	INH	8E	2	fp...	--	0	---	---	---
STX opr8a STX opr16a STX oprx16,X STX oprx8,X STX ,X STX oprx16,SP STX oprx8,SP	Store X (Low 8 Bits of Index Register) in Memory M ← (X)	DIR EXT IX2 IX1 IX SP2 SP1	BF dd CF hh ll DF ee ff EF ff FF 9E DF ee ff 9E EF ff	3 4 4 3 2 5 4	wpp pwpp pwpp wpp wp ppwpp pwpp		0-	-↑↑-		

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR					
						VH	I	N	Z	C	
SUB #opr8i SUB opr8a SUB opr16a SUB oprx16,X SUB oprx8,X SUB ,X SUB oprx16,SP SUB oprx8,SP	Subtract $A \leftarrow (A) - (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A0 ii B0 dd C0 hh ll D0 ee ff E0 ff F0 9E D0 ee ff 9E E0 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp		↓-				-↑↑↑
SWI	Software Interrupt $PC \leftarrow (PC) + 0x0001$ Push (PCL); $SP \leftarrow (SP) - 0x0001$ Push (PCH); $SP \leftarrow (SP) - 0x0001$ Push (X); $SP \leftarrow (SP) - 0x0001$ Push (A); $SP \leftarrow (SP) - 0x0001$ Push (CCR); $SP \leftarrow (SP) - 0x0001$ $I \leftarrow 1$; PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	INH	83	11	sssssvvfppp	--	1	--	--	--	--
TAP	Transfer Accumulator to CCR $CCR \leftarrow (A)$	INH	84	1	p	↑↑				↑↑↑↑	
TAX	Transfer Accumulator to X (Index Register Low) $X \leftarrow (A)$	INH	97	1	p	--				--	--
TPA	Transfer CCR to Accumulator $A \leftarrow (CCR)$	INH	85	1	p	--				--	--
TST opr8a TSTA TSTX TST oprx8,X TST ,X TST oprx8,SP	Test for Negative or Zero (M) – 0x00 (A) – 0x00 (X) – 0x00 (M) – 0x00 (M) – 0x00 (M) – 0x00	DIR INH INH IX1 IX SP1	3D dd 4D 5D 6D ff 7D 9E 6D ff	4 1 1 4 3 5	rfpp p p rfpp rfp prfpp	0-				-↑↑-	
TSX	Transfer SP to Index Reg. $H:X \leftarrow (SP) + 0x0001$	INH	95	2	fp	--				--	--
TXA	Transfer X (Index Reg. Low) to Accumulator $A \leftarrow (X)$	INH	9F	1	p	--				--	--

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR				
						VH	I	N	Z	C
TXS	Transfer Index Reg. to SP SP ← (H:X) – 0x0001	INH	94	2	f _p	--	---	---	---	---
WAIT	Enable Interrupts; Wait for Interrupt I bit ← 0; Halt CPU	INH	8F	2+	f _p . . .	--	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 (0x00xx).
- opr16a* Any label or expression that evaluates to a 16-bit address.
- opr8* Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing.
- opr16* 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
- :

CCR Bits:

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

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 0xFFxx (high byte first)
- w Write 8-bit operand

CCR Effects:

- ↑ Set or cleared
- Not affected
- U Undefined

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

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

INH Inherent
 IMM Immediate
 DIR Direct
 EXT Extended
 DD DIR to DIR
 IX+D IX+ to DIR

REL Relative
 IX Indexed, No Offset
 IX1 Indexed, 8-Bit Offset
 IX2 Indexed, 16-Bit Offset
 IMM to DIR
 DIR to IX+

SP1 Stack Pointer, 8-Bit Offset
 SP2 Stack Pointer, 16-Bit Offset
 IX+ Indexed, No Offset with Post Increment
 IX1+ Indexed, 1-Byte Offset with Post Increment

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

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

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

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

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

Prebyte (9E) and Opcode in
 Hexadecimal

9E60	6
NEG	
3	SP1

 HCS08 Cycles
 Instruction Mnemonic
 Addressing Mode

Chapter 8

Keyboard Interrupt (S08KBIV2)

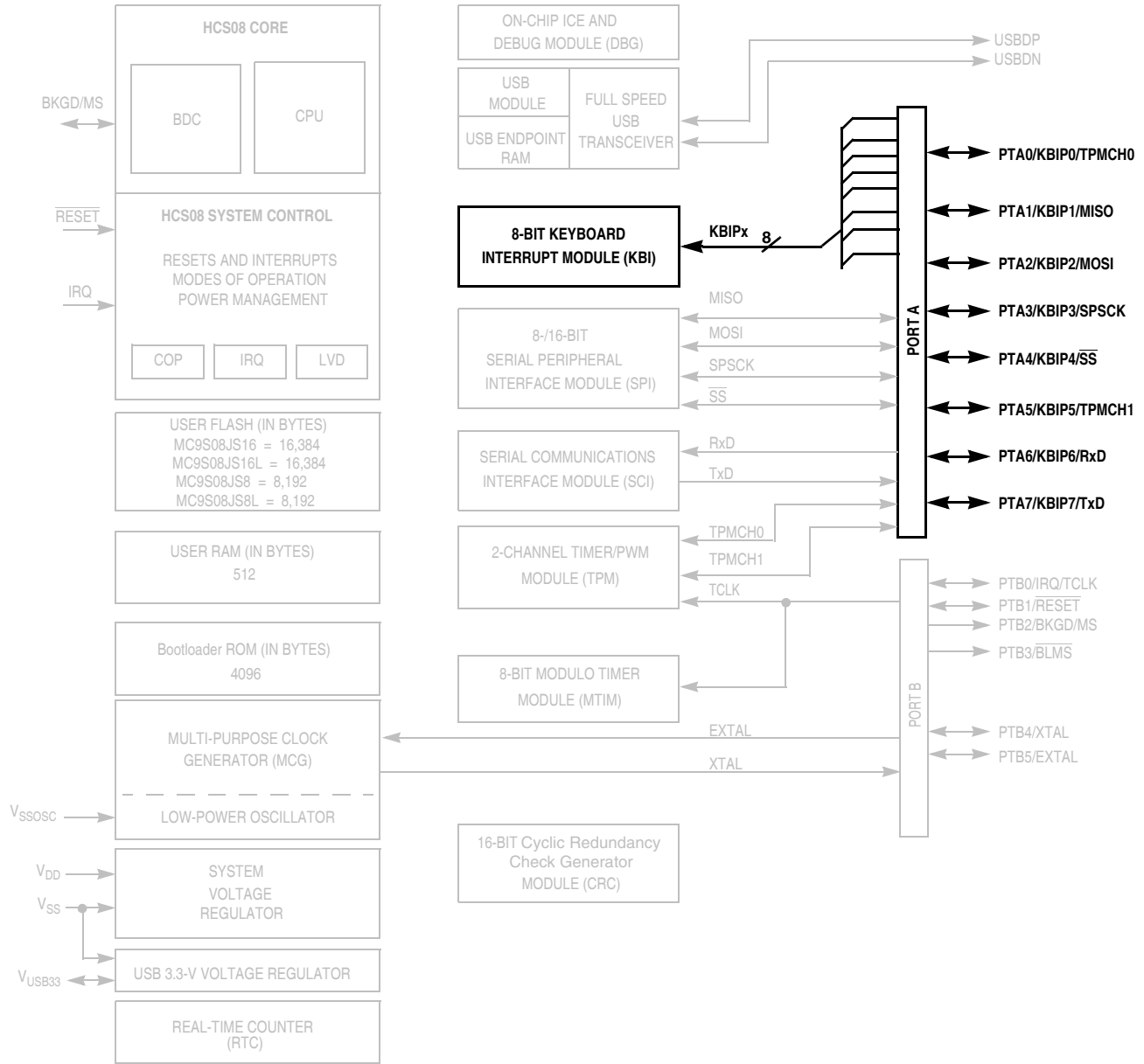
8.1 Introduction

The keyboard interrupt (KBI) module provides up to eight independently enabled external interrupt sources. MC9S08JS16 series devices contain one KBI module with up to eight interrupt sources.

NOTE

When enabling the KBI pin for use, the KBF will be set, and must be cleared prior to enabling the interrupt. When configuring the pin for falling edge and level sensitivity in a 5 V system, it is necessary to wait at least 1 cycle between clearing the flag and enabling the interrupt.

[Figure 8-1](#) shows the device-level block diagram with the KBI module highlighted.



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1). Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD} . IRQ must not be driven above V_{DD} .
4. \overline{RESET} contains integrated pullup device if PTB1 enabled as reset pin function (RSTPE = 1)
5. Pin contains integrated pullup device.
6. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 8-1. MC9S08JS16 Series Block Diagram Highlighting KBI Block and Pins

8.1.1 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.

8.1.2 Modes of Operation

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

8.1.2.1 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 ($KBIE = 1$).

8.1.2.2 KBI in Stop Modes

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

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

8.1.2.3 KBI in Active Background Mode

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

8.1.3 Block Diagram

The block diagram for the keyboard interrupt module is shown in [Figure 8-2](#).

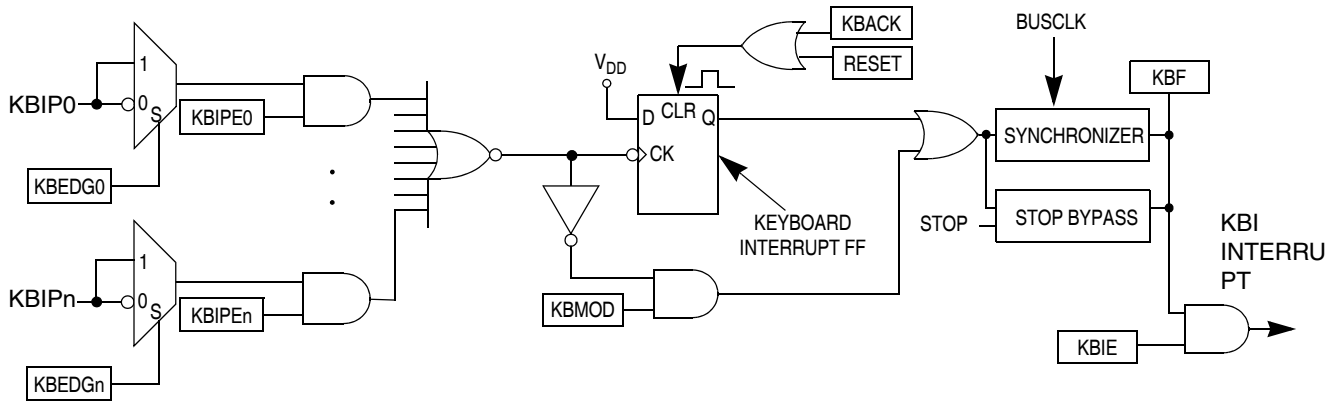


Figure 8-2. KBI Block Diagram

8.2 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 8-1](#).

Table 8-1. Signal Properties

Signal	Function	I/O
KBIPn	Keyboard interrupt pins	I

8.3 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 [Chapter 4, “Memory,”](#) for the absolute address assignments for all KBI registers. This section refers to registers and control bits only by their names.

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

8.3.1 KBI Status and Control Register (KBISC)

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

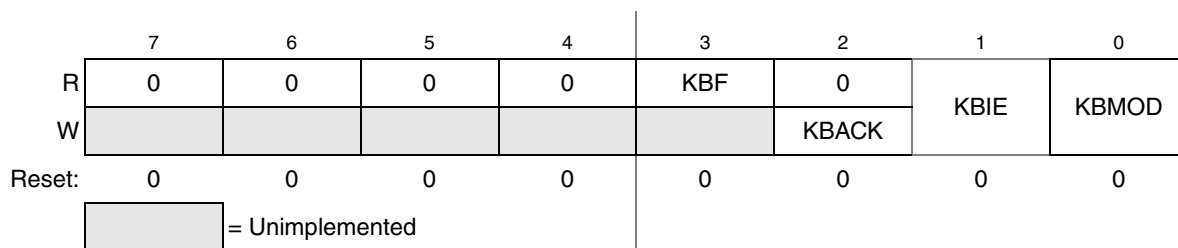


Figure 8-3. KBI Status and Control Register

Table 8-2. KBISC Register Field Descriptions

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

8.3.2 KBI Pin Enable Register (KBIPE)

KBIPE contains the pin enable control bits.

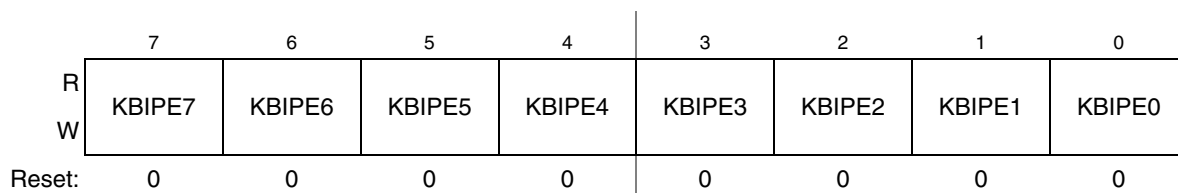


Figure 8-4. KBI Pin Enable Register

Table 8-3. KBIPE Register Field Descriptions

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

8.3.3 KBI Edge Select Register (KBIES)

KBIES contains the edge select control bits.

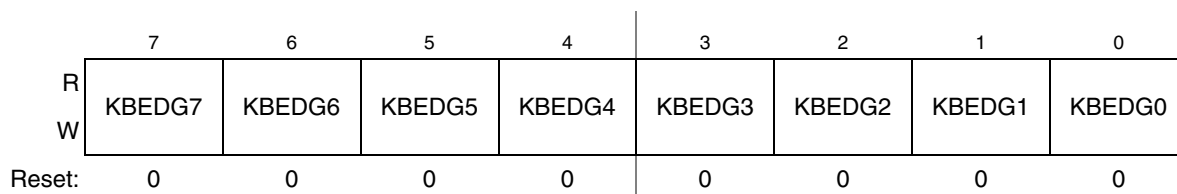


Figure 8-5. KBI Edge Select Register

Table 8-4. KBIES Register Field Descriptions

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

8.4 Functional Description

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

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

8.4.1 Edge Only Sensitivity

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

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

8.4.2 Edge and Level Sensitivity

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

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

8.4.3 KBI Pullup/Pulldown Resistors

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

8.4.4 KBI Initialization

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

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

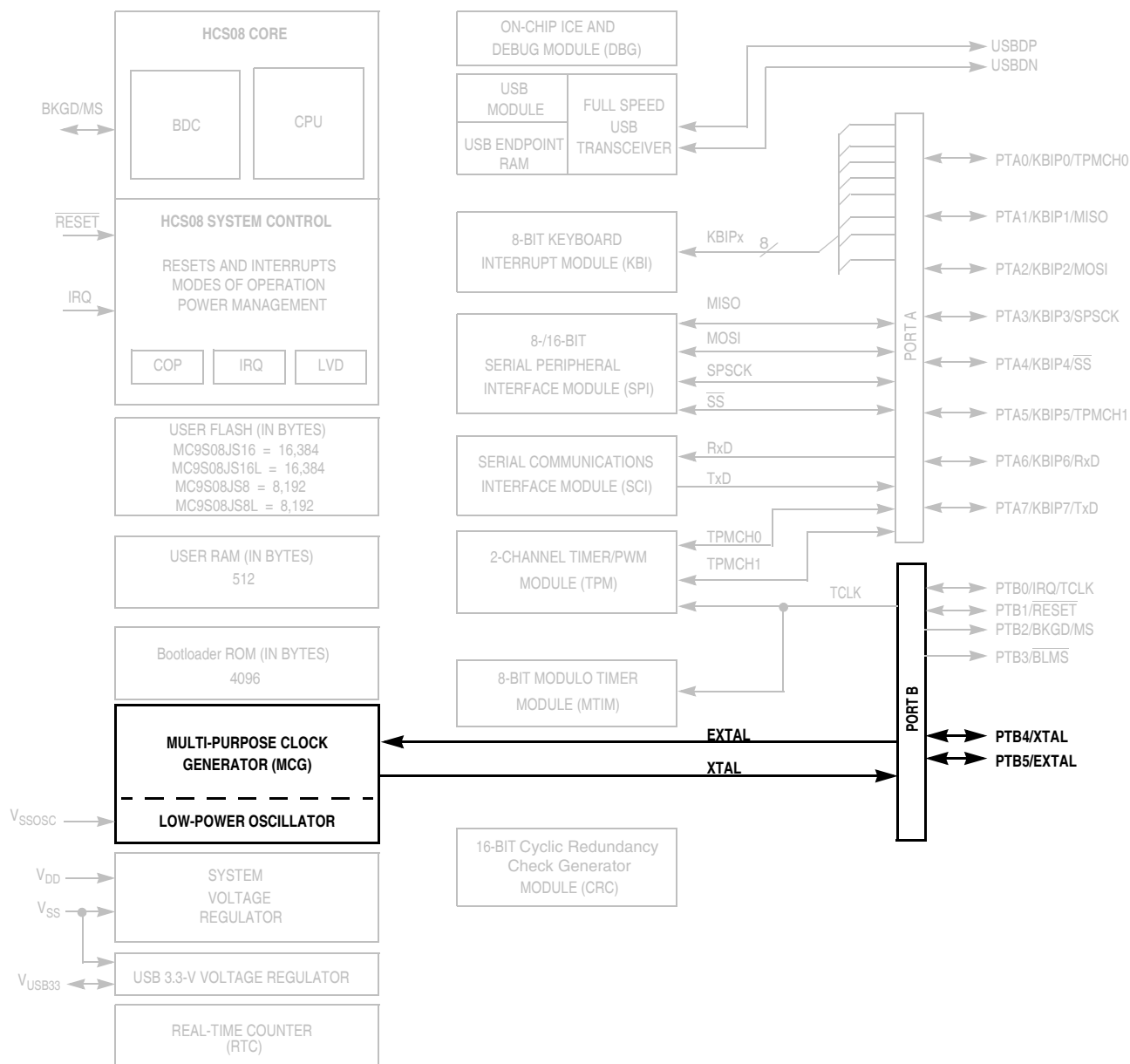
Chapter 9

Multi-Purpose Clock Generator (S08MCGV1)

9.1 Introduction

The multi-purpose clock generator (MCG) module provides several clock source choices for the MCU. 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, and either of the internal or external reference clocks as a source for the MCU system clock. Whichever clock source is chosen, it is passed through a reduced bus divider which allows a lower output clock frequency to be derived. The MCG also controls an external oscillator (XOSC) for the use of a crystal or resonator as the external reference clock.

For USB operation on the MC9S08JS16 series, the MCG must be configured for PLL engaged external (PEE) mode using a crystal to achieve an MCGOUT frequency of 48 MHz.



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1).
Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD} . IRQ must not be driven above V_{DD} .
4. \overline{RESET} contains integrated pullup device if PTB1 enabled as reset pin function (RSTPE = 1)
5. Pin contains integrated pullup device.
6. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 9-1. MC9S08JS16 Series Block Diagram Highlighting MCG Block and Pins

9.1.1 Features

Key features of the MCG module are:

- Frequency-locked loop (FLL)
 - 0.2% resolution using internal 32-kHz reference
 - 2% deviation over voltage and temperature using internal 32-kHz reference
 - Internal or external reference can be used to control the FLL
- Phase-locked loop (PLL)
 - Voltage-controlled oscillator (VCO)
 - Modulo VCO frequency divider
 - Phase/Frequency detector
 - Integrated loop filter
 - Lock detector with interrupt capability
- Internal reference clock
 - Nine trim bits for accuracy
 - Can be selected as the clock source for the MCU
- External reference clock
 - Control for external oscillator
 - Clock monitor with reset capability
 - Can be selected as the clock source for the MCU
- Reference divider is provided
- Clock source selected can be divided down by 1, 2, 4, or 8
- BDC clock (MCGLCLK) is provided as a constant divide by 2 of the DCO output whether in an FLL or PLL mode.

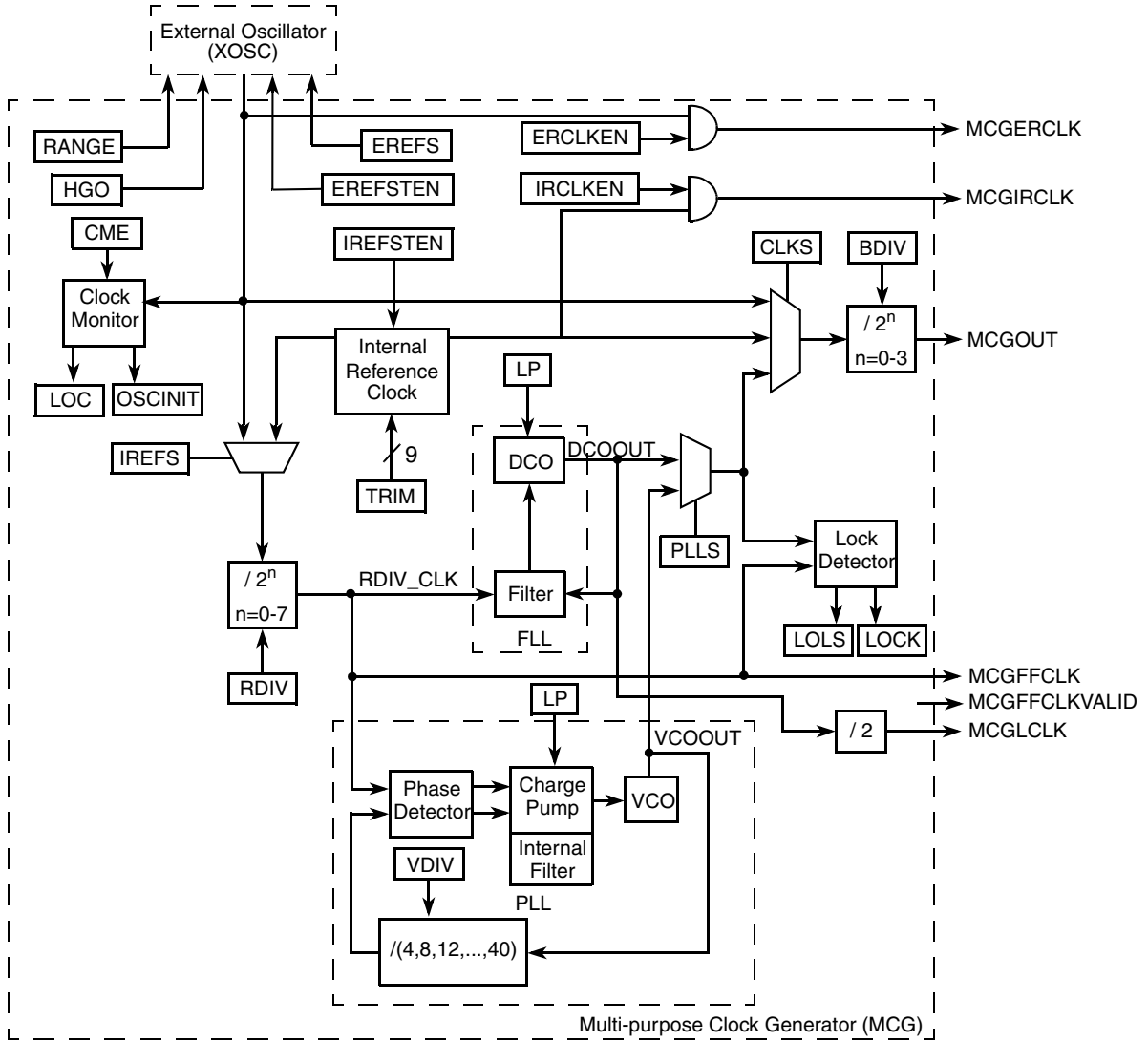


Figure 9-2. Multi-Purpose Clock Generator (MCG) Block Diagram

9.1.2 Modes of Operation

There are nine modes of operation for the MCG:

- FLL Engaged Internal (FEI)
- FLL Engaged External (FEE)
- FLL Bypassed Internal (FBI)
- FLL Bypassed External (FBE)
- PLL Engaged External (PEE)
- PLL Bypassed External (PBE)
- Bypassed Low Power Internal (BLPI)
- Bypassed Low Power External (BLPE)
- Stop

For details see [Section 9.4.1, “Operational Modes](#).

9.2 External Signal Description

There are no MCG signals that connect off chip.

9.3 Register Definition

9.3.1 MCG Control Register 1 (MCGC1)

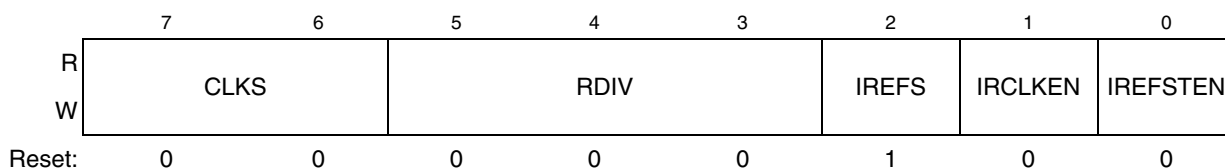


Figure 9-3. MCG Control Register 1 (MCGC1)

Table 9-1. MCG Control Register 1 Field Descriptions

Field	Description
7:6 CLKS	Clock Source Select — Selects the system clock source. 00 Encoding 0 — Output of FLL or PLL is selected. 01 Encoding 1 — Internal reference clock is selected. 10 Encoding 2 — External reference clock is selected. 11 Encoding 3 — Reserved, defaults to 00.
5:3 RDIV	Reference Divider — Selects the amount to divide down the reference clock selected by the IREFS bit. If the FLL is selected, the resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. If the PLL is selected, the resulting frequency must be in the range 1 MHz to 2 MHz. 000 Encoding 0 — Divides reference clock by 1 (reset default) 001 Encoding 1 — Divides reference clock by 2 010 Encoding 2 — Divides reference clock by 4 011 Encoding 3 — Divides reference clock by 8 100 Encoding 4 — Divides reference clock by 16 101 Encoding 5 — Divides reference clock by 32 110 Encoding 6 — Divides reference clock by 64 111 Encoding 7 — Divides reference clock by 128
2 IREFS	Internal Reference Select — Selects the reference clock source. 1 Internal reference clock selected 0 External reference clock selected
1 IRCLKEN	Internal Reference Clock Enable — Enables the internal reference clock for use as MCGIRCLK. 1 MCGIRCLK active 0 MCGIRCLK inactive
0 IREFSTEN	Internal Reference Stop Enable — Controls whether or not the internal reference clock remains enabled when the MCG enters stop mode. 1 Internal reference clock stays enabled in stop if IRCLKEN is set or if MCG is in FEI, FBI, or BLPI mode before entering stop 0 Internal reference clock is disabled in stop

9.3.2 MCG Control Register 2 (MCGC2)



Figure 9-4. MCG Control Register 2 (MCGC2)

Table 9-2. MCG Control Register 2 Field Descriptions

Field	Description
7:6 BDIV	Bus Frequency Divider — Selects the amount to divide down the clock source selected by the CLKS bits in the MCGC1 register. This controls the bus frequency. 00 Encoding 0 — Divides selected clock by 1 01 Encoding 1 — Divides selected clock by 2 (reset default) 10 Encoding 2 — Divides selected clock by 4 11 Encoding 3 — Divides selected clock by 8
5 RANGE	Frequency Range Select — Selects the frequency range for the external oscillator or external clock source. 1 High frequency range selected for the external oscillator of 1 MHz to 16 MHz (1 MHz to 40 MHz for external clock source) 0 Low frequency range selected for the external oscillator of 32 kHz to 100 kHz (32 kHz to 1 MHz for external clock source)
4 HGO	High Gain Oscillator Select — Controls the external oscillator mode of operation. 1 Configure external oscillator for high gain operation 0 Configure external oscillator for low power operation
3 LP	Low Power Select — Controls whether the FLL (or PLL) is disabled in bypassed modes. 1 FLL (or PLL) is disabled in bypass modes (lower power). 0 FLL (or PLL) is not disabled in bypass modes.
2 EREFS	External Reference Select — Selects the source for the external reference clock. 1 Oscillator requested 0 External Clock Source requested
1 ERCLKEN	External Reference Enable — Enables the external reference clock for use as MCGERCLK. 1 MCGERCLK active 0 MCGERCLK inactive
0 EREFSTEN	External Reference Stop Enable — Controls whether or not the external reference clock remains enabled when the MCG enters stop mode. 1 External reference clock stays enabled in stop if ERCLKEN is set or if MCG is in FEE, FBE, PEE, PBE, or BLPE mode before entering stop 0 External reference clock is disabled in stop

9.3.3 MCG Trim Register (MCGTRM)

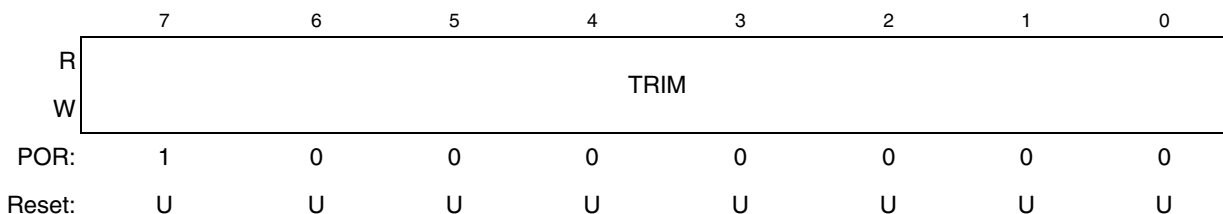


Figure 9-5. MCG Trim Register (MCGTRM)

Table 9-3. MCG Trim Register Field Descriptions

Field	Description
7:0 TRIM	<p>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.</p> <p>An additional fine trim bit is available in MCGSC as the FTRIM bit.</p> <p>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.</p>

9.3.4 MCG Status and Control Register (MCGSC)

	7	6	5	4	3	2	1	0
R	LOLS	LOCK	PLLST	IREFST	CLKST		OSCINIT	FTRIM
W								
POR:	0	0	0	1	0	0	0	0
Reset:	0	0	0	1	0	0	0	U

Figure 9-6. MCG Status and Control Register (MCGSC)

Table 9-4. MCG Status and Control Register Field Descriptions

Field	Description
7 LOLS	<p>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.</p> <p>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.</p>
6 LOCK	<p>Lock Status — Indicates whether the FLL or PLL has acquired lock. Lock detection is disabled when both the FLL and PLL are disabled. If the lock status bit is set then changing the value of any of the following bits IREFS, PLLS, RDIV[2:0], TRIM[7:0] (if in FEI or FBI modes), or VDIV[3:0] (if in PBE or PEE modes), will cause the lock status bit to clear and stay cleared until the FLL or PLL has reacquired lock. Stop mode entry will also cause the lock status bit to clear and stay cleared until the FLL or PLL has reacquired lock. Entry into BLPI or BLPE mode will also cause the lock status bit to clear and stay cleared until the MCG has exited these modes and the FLL or PLL has reacquired lock.</p> <p>0 FLL or PLL is currently unlocked. 1 FLL or PLL is currently locked.</p>
5 PLLST	<p>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.</p> <p>0 Source of PLLS clock is FLL clock. 1 Source of PLLS clock is PLL clock.</p>
4 IREFST	<p>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.</p> <p>0 Source of reference clock is external reference clock (oscillator or external clock source as determined by the IREFS bit in the MCGC2 register). 1 Source of reference clock is internal reference clock.</p>
3:2 CLKST	<p>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.</p> <p>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.</p>

Table 9-4. MCG Status and Control Register Field Descriptions (continued)

Field	Description
1 OSCINIT	OSC Initialization — If the external reference clock is selected by ERCLKEN or by the MCG being in FEE, FBE, PEE, PBE, or BLPE mode, and if EREFS is set, then this bit is set after the initialization cycles of the external oscillator clock have completed. This bit is only cleared when either EREFS is cleared or when the MCG is in either FEI, FBI, or BLPI mode and ERCLKEN is cleared.
0 FTRIM	MCG Fine Trim — Controls the smallest adjustment of the internal reference clock frequency. Setting FTRIM will increase the period and clearing FTRIM will decrease the period by the smallest amount possible. If an FTRIM value stored in nonvolatile memory is to be used, it's the user's responsibility to copy that value from the nonvolatile memory location to this register's FTRIM bit.

9.3.5 MCG Control Register 3 (MCGC3)

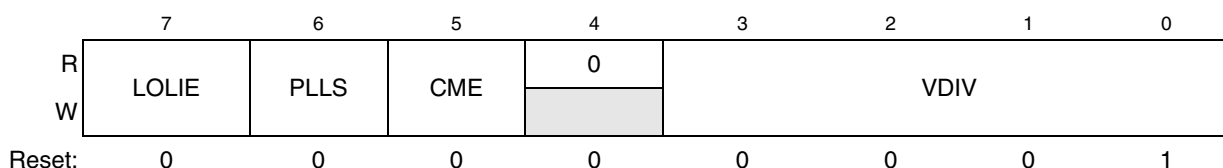


Figure 9-7. MCG PLL Register (MCGPLL)

Table 9-5. MCG PLL Register Field Descriptions

Field	Description
7 LOLIE	Loss of Lock Interrupt Enable — Determines if an interrupt request is made following a loss of lock indication. The LOLIE bit only has an effect when LOLS is set. 0 No request on loss of lock. 1 Generate an interrupt request on loss of lock.
6 PLLS	PLL Select — Controls whether the PLL or FLL is selected. If the PLLS bit is clear, the PLL is disabled in all modes. If the PLLS is set, the FLL is disabled in all modes. 1 PLL is selected 0 FLL is selected

Table 9-5. MCG PLL Register Field Descriptions (continued)

Field	Description
5 CME	<p>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.</p> <p>0 Clock monitor is disabled. 1 Generate a reset request on loss of external clock.</p>
3:0 VDIV	<p>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.</p> <p>0000 Encoding 0 — Reserved. 0001 Encoding 1 — Multiply by 4. 0010 Encoding 2 — Multiply by 8. 0011 Encoding 3 — Multiply by 12. 0100 Encoding 4 — Multiply by 16. 0101 Encoding 5 — Multiply by 20. 0110 Encoding 6 — Multiply by 24. 0111 Encoding 7 — Multiply by 28. 1000 Encoding 8 — Multiply by 32. 1001 Encoding 9 — Multiply by 36. 1010 Encoding 10 — Multiply by 40. 1011 Encoding 11 — Reserved (default to M=40). 11xx Encoding 12-15 — Reserved (default to M=40).</p>

9.4 Functional Description

9.4.1 Operational Modes

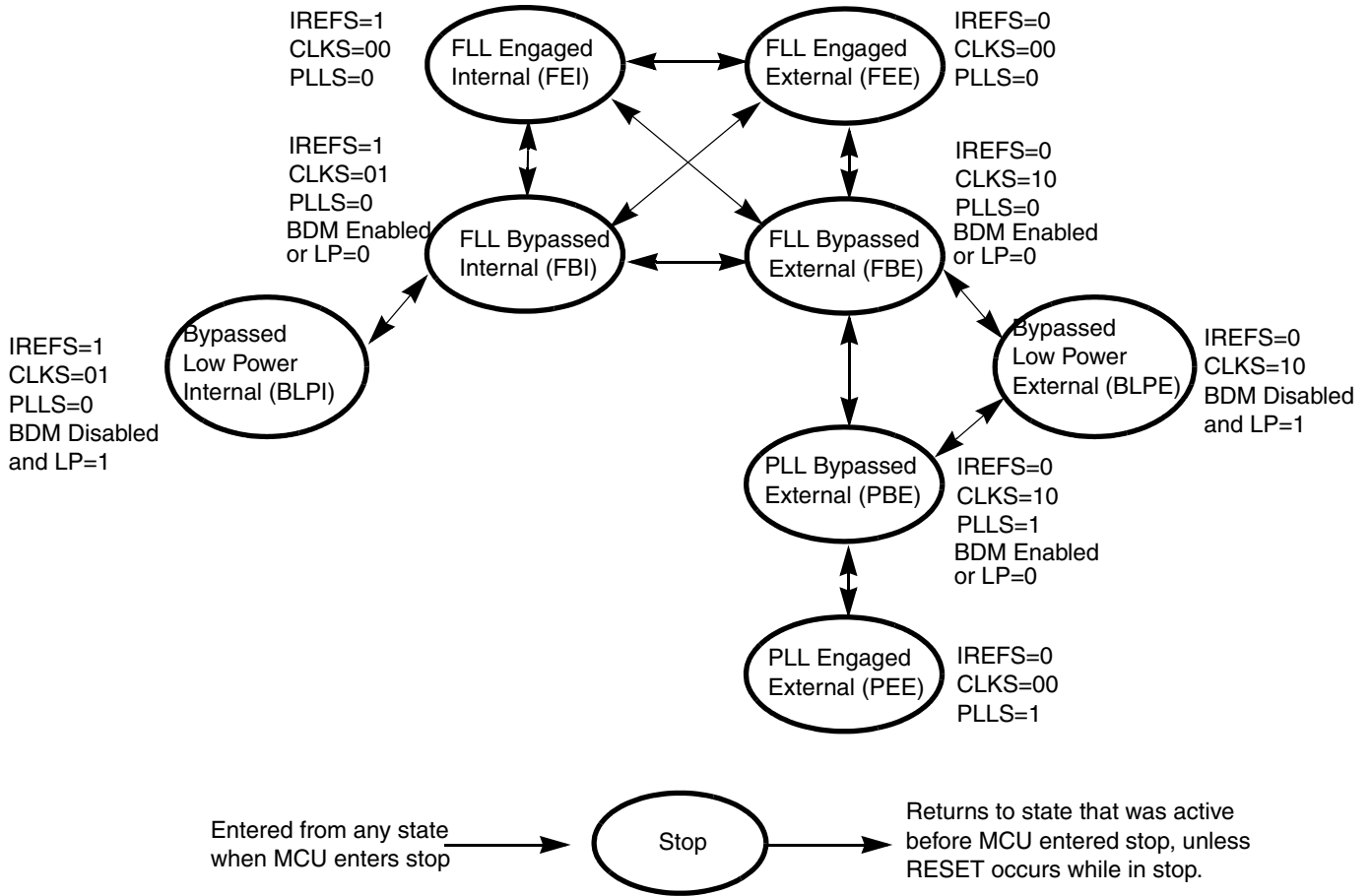


Figure 9-8. Clock Switching Modes

The nine states of the MCG are shown as a state diagram and are described below. The arrows indicate the allowed movements between the states.

9.4.1.1 FLL Engaged Internal (FEI)

FLL engaged internal (FEI) is the default mode of operation and is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 1
- PLLS bit is written to 0
- RDIV bits are written to 000. Since the internal reference clock frequency should already be in the range of 31.25 kHz to 39.0625 kHz after it is trimmed, no further frequency divide is necessary.

In FLL engaged internal mode, the MCGOUT clock is derived from the FLL clock, which is controlled by the internal reference clock. The FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

9.4.1.2 FLL Engaged External (FEE)

The FLL engaged external (FEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 0
- PLLS bit is written to 0
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz

In FLL engaged external mode, the MCGOUT clock is derived from the FLL clock which is controlled by the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source. The FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

9.4.1.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal (FBI) mode, the MCGOUT clock is derived from the internal reference clock and the FLL is operational but its output clock is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUT clock is driven from the internal reference clock.

The FLL bypassed internal mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1
- PLLS bit is written to 0
- RDIV bits are written to 000. Since the internal reference clock frequency should already be in the range of 31.25 kHz to 39.0625 kHz after it is trimmed, no further frequency divide is necessary.

- LP bit is written to 0

In FLL bypassed internal mode, the MCGOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

9.4.1.4 FLL Bypassed External (FBE)

In FLL bypassed external (FBE) mode, the MCGOUT clock is derived from the external reference clock and the FLL is operational but its output clock is not used. This mode is useful to allow the FLL to acquire its target frequency while the MCGOUT clock is driven from the external reference clock.

The FLL bypassed external mode is entered when all the following conditions occur:

- CLKS bits are written to 10
- IREFS bit is written to 0
- PLLS bit is written to 0
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz
- LP bit is written to 0

In FLL bypassed external mode, the MCGOUT clock is derived from the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source. The FLL clock is controlled by the external reference clock, and the FLL clock frequency locks to 1024 times the reference frequency, as selected by the RDIV bits. The MCGLCLK is derived from the FLL and the PLL is disabled in a low power state.

NOTE

It is possible to briefly operate in FBE mode with an FLL reference clock frequency that is greater than the specified maximum frequency. This can be necessary in applications that operate in PEE mode using an external crystal with a frequency above 5 MHz. Please see [9.5.2.4, “Example # 4: Moving from FEI to PEE Mode: External Crystal = 8 MHz, Bus Frequency = 8 MHz”](#) for a detailed example.

9.4.1.5 PLL Engaged External (PEE)

The PLL engaged external (PEE) mode is entered when all the following conditions occur:

- CLKS bits are written to 00
- IREFS bit is written to 0
- PLLS bit is written to 1
- RDIV bits are written to divide reference clock to be within the range of 1 MHz to 2 MHz

In PLL engaged external mode, the MCGOUT clock is derived from the PLL clock which is controlled by the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source. The PLL clock frequency locks to a

multiplication factor, as selected by the VDIV bits, times the reference frequency, as selected by the RDIV bits. If BDM is enabled then the MCGLCLK is derived from the DCO (open-loop mode) divided by two. If BDM is not enabled then the FLL is disabled in a low power state.

9.4.1.6 PLL Bypassed External (PBE)

In PLL bypassed external (PBE) mode, the MCGOUT clock is derived from the external reference clock and the PLL is operational but its output clock is not used. This mode is useful to allow the PLL to acquire its target frequency while the MCGOUT clock is driven from the external reference clock.

The PLL bypassed external mode is entered when all the following conditions occur:

- CLKS bits are written to 10
- IREFS bit is written to 0
- PLLS bit is written to 1
- RDIV bits are written to divide reference clock to be within the range of 1 MHz to 2 MHz
- LP bit is written to 0

In PLL bypassed external mode, the MCGOUT clock is derived from the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source. The PLL clock frequency locks to a multiplication factor, as selected by the VDIV bits, times the reference frequency, as selected by the RDIV bits. If BDM is enabled then the MCGLCLK is derived from the DCO (open-loop mode) divided by two. If BDM is not enabled then the FLL is disabled in a low power state.

9.4.1.7 Bypassed Low Power Internal (BLPI)

The bypassed low power internal (BLPI) mode is entered when all the following conditions occur:

- CLKS bits are written to 01
- IREFS bit is written to 1
- PLLS bit is written to 0
- LP bit is written to 1
- BDM mode is not active

In bypassed low power internal mode, the MCGOUT clock is derived from the internal reference clock.

The PLL and the FLL are disabled at all times in BLPI mode and the MCGLCLK will not be available for BDC communications. If the BDM becomes active the mode will switch to FLL bypassed internal (FBI) mode.

9.4.1.8 Bypassed Low Power External (BLPE)

The bypassed low power external (BLPE) mode is entered when all the following conditions occur:

- CLKS bits are written to 10
- IREFS bit is written to 0
- PLLS bit is written to 0 or 1

- LP bit is written to 1
- BDM mode is not active

In bypassed low power external mode, the MCGOUT clock is derived from the external reference clock. The external reference clock which is enabled can be an external crystal/resonator or it can be another external clock source.

The PLL and the FLL are disabled at all times in BLPE mode and the MCGLCLK will not be available for BDC communications. If the BDM becomes active the mode will switch to one of the bypassed external modes as determined by the state of the PLLS bit.

9.4.1.9 Stop

Stop mode is entered whenever the MCU enters a STOP state. In this mode, the FLL and PLL are disabled and all MCG clock signals are static except in the following cases:

MCGIRCLK will be active in stop mode when all the following conditions occur:

- IRCLKEN = 1
- IREFSTEN = 1

MCGERCLK will be active in stop mode when all the following conditions occur:

- ERCLKEN = 1
- EREFSTEN = 1

9.4.2 Mode Switching

When switching between engaged internal and engaged external modes the IREFS bit can be changed at anytime, but the RDIV bits must be changed simultaneously so that the reference frequency stays in the range required by the state of the PLLS bit (31.25 kHz to 39.0625 kHz if the FLL is selected, or 1 MHz to 2 MHz if the PLL is selected). After a change in the IREFS value the FLL or PLL will begin locking again after the switch is completed. The completion of the switch is shown by the IREFST bit.

For the special case of entering stop mode immediately after switching to FBE mode, if the external clock and the internal clock are disabled in stop mode, (EREFSTEN = 0 and IREFSTEN = 0), it is necessary to allow 100us after the IREFST bit is cleared to allow the internal reference to shutdown. For most cases the delay due to instruction execution times will be sufficient.

The CLKS bits can also be changed at anytime, but in order for the MCGLCLK to be configured correctly the RDIV bits must be changed simultaneously so that the reference frequency stays in the range required by the state of the PLLS bit (31.25 kHz to 39.0625 kHz if the FLL is selected, or 1 MHz to 2MHz if the PLL is selected). The actual switch to the newly selected clock will be shown by the CLKST bits. If the newly selected clock is not available, the previous clock will remain selected.

For details see [Figure 9-8](#).

9.4.3 Bus Frequency Divider

The BDIV bits can be changed at anytime and the actual switch to the new frequency will occur immediately.

9.4.4 Low Power Bit Usage

The low power bit (LP) is provided to allow the FLL or PLL to be disabled and thus conserve power when these systems are not being used. However, in some applications it may be desirable to enable the FLL or PLL and allow it to lock for maximum accuracy before switching to an engaged mode. Do this by writing the LP bit to 0.

9.4.5 Internal Reference Clock

When IRCLKEN is set the internal reference clock signal will be presented as MCGIRCLK, which can be used as an additional clock source. The MCGIRCLK frequency can be re-targeted by trimming the period of the internal reference clock. This can be done by writing a new value to the TRIM bits in the MCGTRM register. Writing a larger value will decrease the MCGIRCLK frequency, and writing a smaller value to the MCGTRM register will increase the MCGIRCLK frequency. The TRIM bits will effect the MCGOUT frequency if the MCG is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or bypassed low power internal (BLPI) mode. The TRIM and FTRIM value is initialized by POR but is not affected by other resets.

Until MCGIRCLK is trimmed, programming low reference divider (RDIV) factors may result in MCGOUT frequencies that exceed the maximum chip-level frequency and violate the chip-level clock timing specifications (see the [Device Overview](#) chapter).

If IREFSTEN and IRCLKEN bits are both set, the internal reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

9.4.6 External Reference Clock

The MCG module can support an external reference clock with frequencies between 31.25 kHz to 5 MHz in FEE and FBE modes, 1 MHz to 16 MHz in PEE and PBE modes, and 0 to 40 MHz in BLPE mode.

When ERCLKEN is set, the external reference clock signal will be presented as MCGERCLK, which can be used as an additional clock source. When IREFS = 1, the external reference clock will not be used by the FLL or PLL and will only be used as MCGERCLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications will support (see the [Device Overview](#) chapter).

If EREFSTEN and ERCLKEN bits are both set or the MCG is in FEE, FBE, PEE, PBE or BLPE mode, the external reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

If CME bit is written to 1, the clock monitor is enabled. If the external reference falls below a certain frequency (f_{loc_high} or f_{loc_low} depending on the RANGE bit in the MCGC2), the MCU will reset. The LOC bit in the System Reset Status (SRS) register will be set to indicate the error.

9.4.7 Fixed Frequency Clock

The MCG presents the divided reference clock as MCGFFCLK for use as an additional clock source. The MCGFFCLK frequency must be no more than 1/4 of the MCGOUT frequency to be valid. Because of this requirement, the MCGFFCLK is not valid in bypass modes for the following combinations of BDIV and RDIV values:

- BDIV=00 (divide by 1), RDIV < 010

BDIV=01 (divide by 2), RDIV < 011 When MCGFFCLK is valid then MCGFFCLKVALID is set to 1. When MCGFFCLK is not valid then MCGFFCLKVALID is set to 0.

9.5 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.

9.5.1 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_{refst} microseconds before the FLL can acquire lock. As soon as the internal reference is stable, the FLL will acquire lock in $t_{\text{fl_lock}}$ milliseconds.

Upon POR, the internal reference will require trimming to guarantee an accurate clock. Freescale recommends using FLASH location 0xFFAE for storing the fine trim bit, FTRIM in the MCGSC register, and 0xFFAF for storing the 8-bit trim value in the MCGTRM register. The MCU will not automatically copy the values in these FLASH locations to the respective registers. Therefore, user code must copy these values from FLASH to the registers.

NOTE

The BDIV value should not be changed to divide-by-1 without first trimming the internal reference. Failure to do so could result in the MCU running out of specification.

9.5.1.1 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 9-8](#)). Reaching any of the other modes requires first configuring the MCG for one of these three initial modes. Care must be taken to check relevant status bits in the MCGSC register reflecting all configuration changes within each mode.

To change from FEI mode to FEE or FBE modes, follow this procedure:

1. Enable the external clock source by setting the appropriate bits in MCGC2.
2. Write to MCGC1 to select the clock mode.
 - If entering FEE, set RDIV appropriately, clear the IREFS bit to switch to the external reference, and leave the CLKS bits at %00 so that the output of the FLL is selected as the system clock source.

- If entering FBE, clear the IREFS bit to switch to the external reference and change the CLKS bits to %10 so that the external reference clock is selected as the system clock source. The RDIV bits should also be set appropriately here according to the external reference frequency because although the FLL is bypassed, it is still on in FBE mode.
 - The internal reference can optionally be kept running by setting the IRCLKEN bit. This is useful if the application will switch back and forth between internal and external modes. For minimum power consumption, leave the internal reference disabled while in an external clock mode.
3. After the proper configuration bits have been set, wait for the affected bits in the MCGSC register to be changed appropriately, reflecting that the MCG has moved into the proper mode.
 - If ERCLKEN was set in step 1 or the MCG is in FEE, FBE, PEE, PBE, or BLPE mode, and EREFS was also set in step 1, wait here for the OSCINIT bit to become set indicating that the external clock source has finished its initialization cycles and stabilized. Typical crystal startup times are given in Appendix A, “Electrical Characteristics”.
 - If in FEE mode, check to make sure the IREFST bit is cleared and the LOCK bit is set before moving on.
 - If in FBE mode, check to make sure the IREFST bit is cleared, the LOCK bit is set, and the CLKST bits have changed to %10 indicating the external reference clock has been appropriately selected. Although the FLL is bypassed in FBE mode, it is still on and will lock in FBE mode.

To change from FEI clock mode to FBI clock mode, follow this procedure:

1. Change the CLKS bits to %01 so that the internal reference clock is selected as the system clock source.
2. Wait for the CLKST bits in the MCGSC register to change to %01, indicating that the internal reference clock has been appropriately selected.

9.5.2 MCG Mode Switching

When switching between operational modes of the MCG, certain configuration bits must be changed in order to properly move from one mode to another. Each time any of these bits are changed (PLLS, IREFS, CLKS, or EREFS), the corresponding bits in the MCGSC register (PLLST, IREFST, CLKST, or OSCINIT) must be checked before moving on in the application software.

Additionally, care must be taken to ensure that the reference clock divider (RDIV) is set properly for the mode being switched to. For instance, in PEE mode, if using a 4 MHz crystal, RDIV must be set to %001 (divide-by-2) or %010 (divide -by-4) in order to divide the external reference down to the required frequency between 1 and 2 MHz.

The RDIV and IREFS bits should always be set properly before changing the PLLS bit so that the FLL or PLL clock has an appropriate reference clock frequency to switch to.

The table below shows MCGOUT frequency calculations using RDIV, BDIV, and VDIV settings for each clock mode. The bus frequency is equal to MCGOUT divided by 2.

Table 9-6. MCGOUT Frequency Calculation Options

Clock Mode	f_{MCGOUT}^1	Note
FEI (FLL engaged internal)	$(f_{int} * 1024) / B$	Typical $f_{MCGOUT} = 16$ MHz immediately after reset. RDIV bits set to %000.
FEE (FLL engaged external)	$(f_{ext} / R * 1024) / B$	f_{ext} / R must be in the range of 31.25 kHz to 39.0625 kHz
FBE (FLL bypassed external)	f_{ext} / B	f_{ext} / R must be in the range of 31.25 kHz to 39.0625 kHz
FBI (FLL bypassed internal)	f_{int} / B	Typical $f_{int} = 32$ kHz
PEE (PLL engaged external)	$[(f_{ext} / R) * M] / B$	f_{ext} / R must be in the range of 1 MHz to 2 MHz
PBE (PLL bypassed external)	f_{ext} / B	f_{ext} / R must be in the range of 1 MHz to 2 MHz
BLPI (Bypassed low power internal)	f_{int} / B	
BLPE (Bypassed low power external)	f_{ext} / B	

¹ R is the reference divider selected by the RDIV bits, B is the bus frequency divider selected by the BDIV bits, and M is the multiplier selected by the VDIV bits.

This section will include 3 mode switching examples using a 4 MHz external crystal. If using an external clock source less than 1 MHz, the MCG should not be configured for any of the PLL modes (PEE and PBE).

9.5.2.1 Example # 1: Moving from FEI to PEE Mode: External Crystal = 4 MHz, Bus Frequency = 8 MHz

In this example, the MCG will move through the proper operational modes from FEI to PEE mode until the 4 MHz crystal reference frequency is set to achieve a bus frequency of 8 MHz. Because the MCG is in FEI mode out of reset, this example also shows how to initialize the MCG for PEE mode out of reset. First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, FEI must transition to FBE mode:
 - a) MCGC2 = 0x36 (%00110110)
 - BDIV (bits 7 and 6) set to %00, or divide-by-1
 - RANGE (bit 5) set to 1 because the frequency of 4 MHz is within the high frequency range
 - HGO (bit 4) set to 1 to configure external oscillator for high gain operation
 - EREFS (bit 2) set to 1, because a crystal is being used
 - ERCLKEN (bit 1) set to 1 to ensure the external reference clock is active
 - b) Loop until OSCINIT (bit 1) in MCGSC is 1, indicating the crystal selected by the EREFS bit has been initialized.

- c) MCGC1 = 0xB8 (%10111000)
 - CLKS (bits 7 and 6) set to %10 in order to select external reference clock as system clock source
 - RDIV (bits 5-3) set to %111, or divide-by-128 because $4 \text{ MHz} / 128 = 31.25 \text{ kHz}$ which is in the 31.25 kHz to 39.0625 kHz range required by the FLL
 - IREFS (bit 2) cleared to 0, selecting the external reference clock
 - d) Loop until IREFST (bit 4) in MCGSC is 0, indicating the external reference is the current source for the reference clock
 - e) Loop until CLKST (bits 3 and 2) in MCGSC are %10, indicating that the external reference clock is selected to feed MCGOUT
2. Then, FBE must transition either directly to PBE mode or first through BLPE mode and then to PBE mode:
 - a) BLPE: If a transition through BLPE mode is desired, first set LP (bit 3) in MCGC2 to 1.
 - b) BLPE/PBE: MCGC1 = 0x90 (%10010000)
 - RDIV (bits 5-3) set to %010, or divide-by-4 because $4 \text{ MHz} / 4 = 1 \text{ MHz}$ which is in the 1 MHz to 2 MHz range required by the PLL. In BLPE mode, the configuration of the RDIV does not matter because both the FLL and PLL are disabled. Changing them only sets up the the dividers for PLL usage in PBE mode
 - c) BLPE/PBE: MCGC3 = 0x44 (%01000100)
 - PLLS (bit 6) set to 1, selects the PLL. In BLPE mode, changing this bit only prepares the MCG for PLL usage in PBE mode
 - VDIV (bits 3-0) set to %0100, or multiply-by-16 because $1 \text{ MHz reference} * 16 = 16 \text{ MHz}$. In BLPE mode, the configuration of the VDIV bits does not matter because the PLL is disabled. Changing them only sets up the multiply value for PLL usage in PBE mode
 - d) BLPE: If transitioning through BLPE mode, clear LP (bit 3) in MCGC2 to 0 here to switch to PBE mode
 - e) PBE: Loop until PLLST (bit 5) in MCGSC is set, indicating that the current source for the PLLS clock is the PLL
 - f) PBE: Then loop until LOCK (bit 6) in MCGSC is set, indicating that the PLL has acquired lock
 3. Last, PBE mode transitions into PEE mode:
 - a) MCGC1 = 0x10 (%00010000)
 - CLKS (bits 7 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-4, a BDIV of divide-by-1, and a VDIV of multiply-by-16, $\text{MCGOUT} = [(4 \text{ MHz} / 4) * 16] / 1 = 16 \text{ MHz}$, and the bus frequency is $\text{MCGOUT} / 2$, or 8 MHz

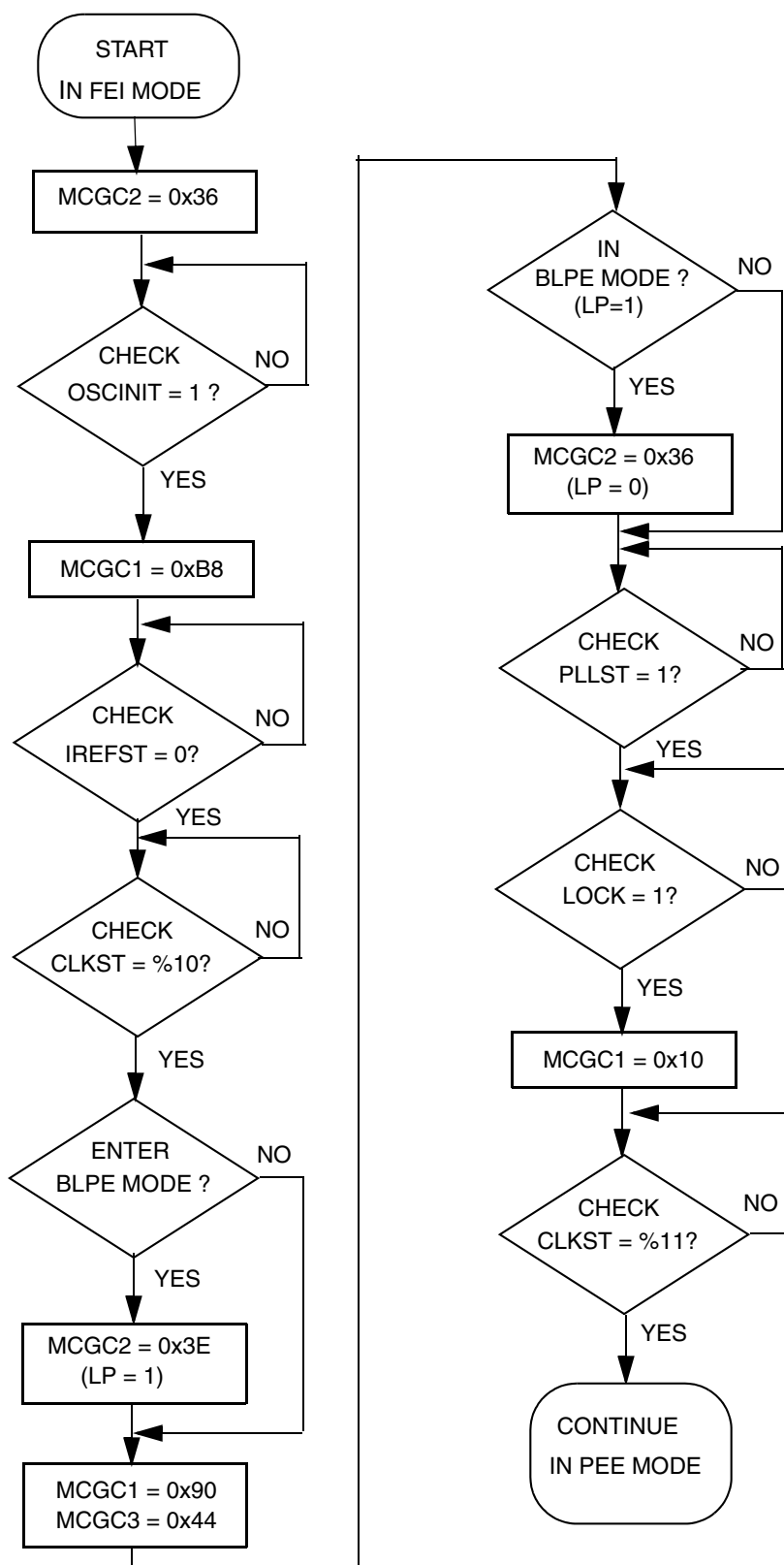


Figure 9-9. Flowchart of FEI to PEE Mode Transition using a 4 MHz crystal

9.5.2.2 Example # 2: Moving from PEE to BLPI Mode: External Crystal = 4 MHz, Bus Frequency = 16 kHz

In this example, the MCG will move through the proper operational modes from PEE mode with a 4 MHz crystal configured for an 8 MHz bus frequency (see previous example) to BLPI mode with a 16 kHz bus frequency. First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, PEE must transition to PBE mode:
 - a) MCGC1 = 0x90 (%10010000)
 - CLKS (bits 7 and 6) set to %10 in order to switch the system clock source to the external reference clock
 - b) Loop until CLKST (bits 3 and 2) in MCGSC are %10, indicating that the external reference clock is selected to feed MCGOUT
2. Then, PBE must transition either directly to FBE mode or first through BLPE mode and then to FBE mode:
 - a) BLPE: If a transition through BLPE mode is desired, first set LP (bit 3) in MCGC2 to 1
 - b) BLPE/FBE: MCGC1 = 0xB8 (%10111000)
 - RDIV (bits 5-3) set to %111, or divide-by-128 because $4 \text{ MHz} / 128 = 31.25 \text{ kHz}$ which is in the 31.25 kHz to 39.0625 kHz range required by the FLL. In BLPE mode, the configuration of the RDIV does not matter because both the FLL and PLL are disabled. Changing them only sets up the dividers for FLL usage in FBE mode
 - c) BLPE/FBE: MCGC3 = 0x04 (%00000100)
 - PLLS (bit 6) clear to 0 to select the FLL. In BLPE mode, changing this bit only prepares the MCG for FLL usage in FBE mode. With PLLS = 0, the VDIV value does not matter.
 - d) BLPE: If transitioning through BLPE mode, clear LP (bit 3) in MCGC2 to 0 here to switch to FBE mode
 - e) FBE: Loop until PLLST (bit 5) in MCGSC is clear, indicating that the current source for the PLLS clock is the FLL
 - f) FBE: Optionally, loop until LOCK (bit 6) in the MCGSC is set, indicating that the FLL has acquired lock. Although the FLL is bypassed in FBE mode, it is still enabled and running.
3. Next, FBE mode transitions into FBI mode:
 - a) MCGC1 = 0x44 (%01000100)
 - CLKS (bits 7 and 6) in MCGSC1 set to %01 in order to switch the system clock to the internal reference clock
 - IREFS (bit 2) set to 1 to select the internal reference clock as the reference clock source
 - RDIV (bits 5-3) set to %000, or divide-by-1 because the trimmed internal reference should be within the 31.25 kHz to 39.0625 kHz range required by the FLL
 - b) Loop until IREFST (bit 4) in MCGSC is 1, indicating the internal reference clock has been selected as the reference clock source
 - c) Loop until CLKST (bits 3 and 2) in MCGSC are %01, indicating that the internal reference clock is selected to feed MCGOUT

4. Lastly, FBI transitions into BLPI mode.
 - a) MCGC2 = 0x08 (%00001000)
 - LP (bit 3) in MCGSC is 1

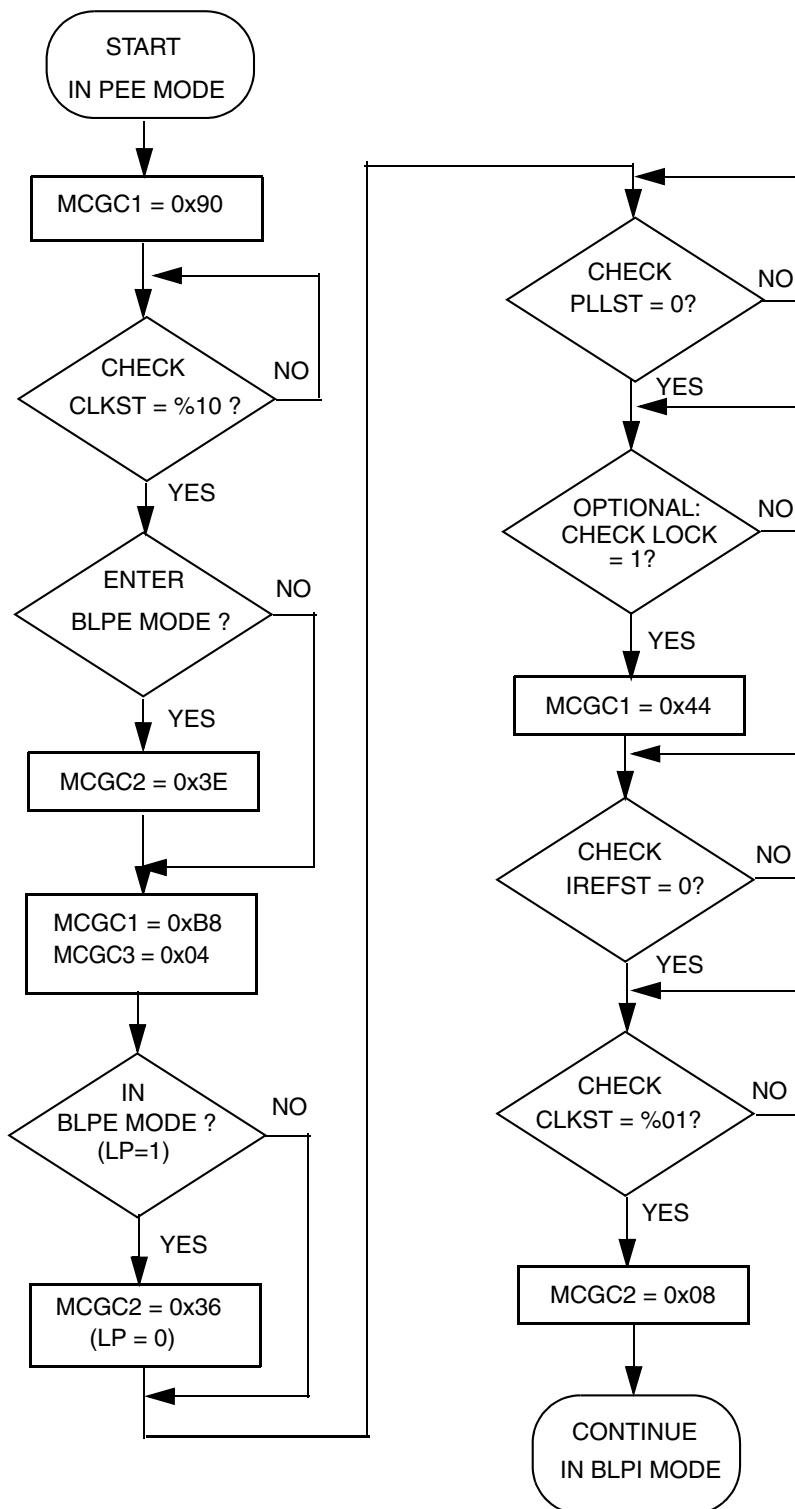


Figure 9-10. Flowchart of PEE to BLPI Mode Transition using a 4 MHz crystal

9.5.2.3 Example #3: Moving from BLPI to FEE Mode: External Crystal = 4 MHz, Bus Frequency = 16 MHz

In this example, the MCG will move through the proper operational modes from BLPI mode at a 16 kHz bus frequency running off of the internal reference clock (see previous example) to FEE mode using a 4 MHz crystal configured for a 16 MHz bus frequency. First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, BLPI must transition to FBI mode.
 - a) MCGC2 = 0x00 (%00000000)
 - LP (bit 3) in MCGSC is 0
 - b) Optionally, loop until LOCK (bit 6) in the MCGSC is set, indicating that the FLL has acquired lock. Although the FLL is bypassed in FBI mode, it is still enabled and running.
2. Next, FBI will transition to FEE mode.
 - a) MCGC2 = 0x36 (%00110110)
 - RANGE (bit 5) set to 1 because the frequency of 4 MHz is within the high frequency range
 - HGO (bit 4) set to 1 to configure external oscillator for high gain operation
 - EREFS (bit 2) set to 1, because a crystal is being used
 - ERCLKEN (bit 1) set to 1 to ensure the external reference clock is active
 - b) Loop until OSCINIT (bit 1) in MCGSC is 1, indicating the crystal selected by the EREFS bit has been initialized.
 - c) MCGC1 = 0x38 (%00111000)
 - CLKS (bits 7 and 6) set to %00 in order to select the output of the FLL as system clock source
 - RDIV (bits 5-3) set to %111, or divide-by-128 because $4 \text{ MHz} / 128 = 31.25 \text{ kHz}$ which is in the 31.25 kHz to 39.0625 kHz range required by the FLL
 - IREFS (bit 1) cleared to 0, selecting the external reference clock
 - d) Loop until IREFST (bit 4) in MCGSC is 0, indicating the external reference clock is the current source for the reference clock
 - e) Optionally, loop until LOCK (bit 6) in the MCGSC is set, indicating that the FLL has reacquired lock.
 - f) Loop until CLKST (bits 3 and 2) in MCGSC are %00, indicating that the output of the FLL is selected to feed MCGOUT

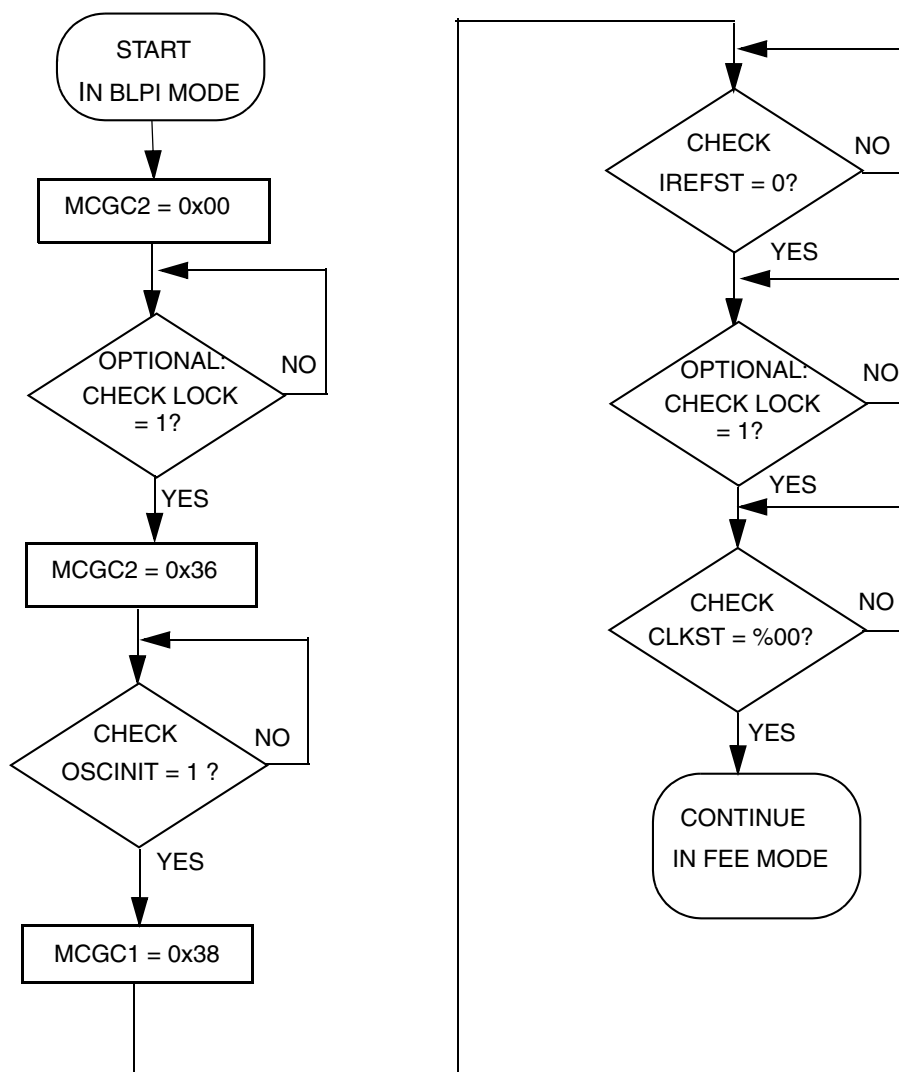


Figure 9-11. Flowchart of BLPI to FEE Mode Transition using a 4 MHz crystal

9.5.2.4 Example # 4: Moving from FEI to PEE Mode: External Crystal = 8 MHz, Bus Frequency = 8 MHz

In this example, the MCG will move through the proper operational modes from FEI to PEE mode until the 8 MHz crystal reference frequency is set to achieve a bus frequency of 8 MHz.

This example is similar to example number one except that in this case the frequency of the external crystal is 8 MHz instead of 4 MHz. Special consideration must be taken with this case since there is a period of time along the way from FEI mode to PEE mode where the FLL operates based on a reference clock with a frequency that is greater than the maximum allowed for the FLL. This occurs because with an 8 MHz

external crystal and a maximum reference divider factor of 128, the resulting frequency of the reference clock for the FLL is 62.5 kHz (greater than the 39.0625 kHz maximum allowed).

Care must be taken in the software to minimize the amount of time spent in this state where the FLL is operating in this condition.

The following code sequence describes how to move from FEI mode to PEE mode until the 8 MHz crystal reference frequency is set to achieve a bus frequency of 8 MHz. Because the MCG is in FEI mode out of reset, this example also shows how to initialize the MCG for PEE mode out of reset. First, the code sequence will be described. Then a flowchart will be included which illustrates the sequence.

1. First, FEI must transition to FBE mode:
 - a) MCGC2 = 0x36 (%00110110)
 - BDIV (bits 7 and 6) set to %00, or divide-by-1
 - RANGE (bit 5) set to 1 because the frequency of 8 MHz is within the high frequency range
 - HGO (bit 4) set to 1 to configure external oscillator for high gain operation
 - EREFS (bit 2) set to 1, because a crystal is being used
 - ERCLKEN (bit 1) set to 1 to ensure the external reference clock is active
 - b) Loop until OSCINIT (bit 1) in MCGSC is 1, indicating the crystal selected by the EREFS bit has been initialized.
 - c) Block Interrupts (If applicable by setting the interrupt bit in the CCR).
 - d) MCGC1 = 0xB8 (%10111000)
 - CLKS (bits 7 and 6) set to %10 in order to select external reference clock as system clock source
 - RDIV (bits 5-3) set to %111, or divide-by-128.

NOTE

8 MHz / 128 = 62.5 kHz which is greater than the 31.25 kHz to 39.0625 kHz range required by the FLL. Therefore after the transition to FBE is complete, software must progress through to BLPE mode immediately by setting the LP bit in MCGC2.

- IREFS (bit 2) cleared to 0, selecting the external reference clock
 - e) Loop until IREFST (bit 4) in MCGSC is 0, indicating the external reference is the current source for the reference clock
 - f) Loop until CLKST (bits 3 and 2) in MCGSC are %10, indicating that the external reference clock is selected to feed MCGOUT
2. Then, FBE mode transitions into BLPE mode:
 - a) MCGC2 = 0x3E (%00111110)
 - LP (bit 3) in MCGC2 to 1 (BLPE mode entered)

NOTE

There must be no extra steps (including interrupts) between steps 1d and 2a.

- b) Enable Interrupts (if applicable by clearing the interrupt bit in the CCR).

- c) MCGC1 = 0x98 (%10011000)
 - RDIV (bits 5-3) set to %011, or divide-by-8 because $8 \text{ MHz} / 8 = 1 \text{ MHz}$ which is in the 1 MHz to 2 MHz range required by the PLL. In BLPE mode, the configuration of the RDIV does not matter because both the FLL and PLL are disabled. Changing them only sets up the the dividers for PLL usage in PBE mode
 - d) MCGC3 = 0x44 (%01000100)
 - PLLS (bit 6) set to 1, selects the PLL. In BLPE mode, changing this bit only prepares the MCG for PLL usage in PBE mode
 - VDIV (bits 3-0) set to %0100, or multiply-by-16 because $1 \text{ MHz reference} * 16 = 16 \text{ MHz}$. In BLPE mode, the configuration of the VDIV bits does not matter because the PLL is disabled. Changing them only sets up the multiply value for PLL usage in PBE mode
 - e) Loop until PLLST (bit 5) in MCGSC is set, indicating that the current source for the PLLS clock is the PLL
3. Then, BLPE mode transitions into PBE mode:
 - a) Clear LP (bit 3) in MCGC2 to 0 here to switch to PBE mode
 - b) Then loop until LOCK (bit 6) in MCGSC is set, indicating that the PLL has acquired lock
 4. Last, PBE mode transitions into PEE mode:
 - a) MCGC1 = 0x18 (%00011000)
 - CLKS (bits 7 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-16, $\text{MCGOUT} = [(8 \text{ MHz} / 8) * 16] / 1 = 16 \text{ MHz}$, and the bus frequency is $\text{MCGOUT} / 2$, or 8 MHz

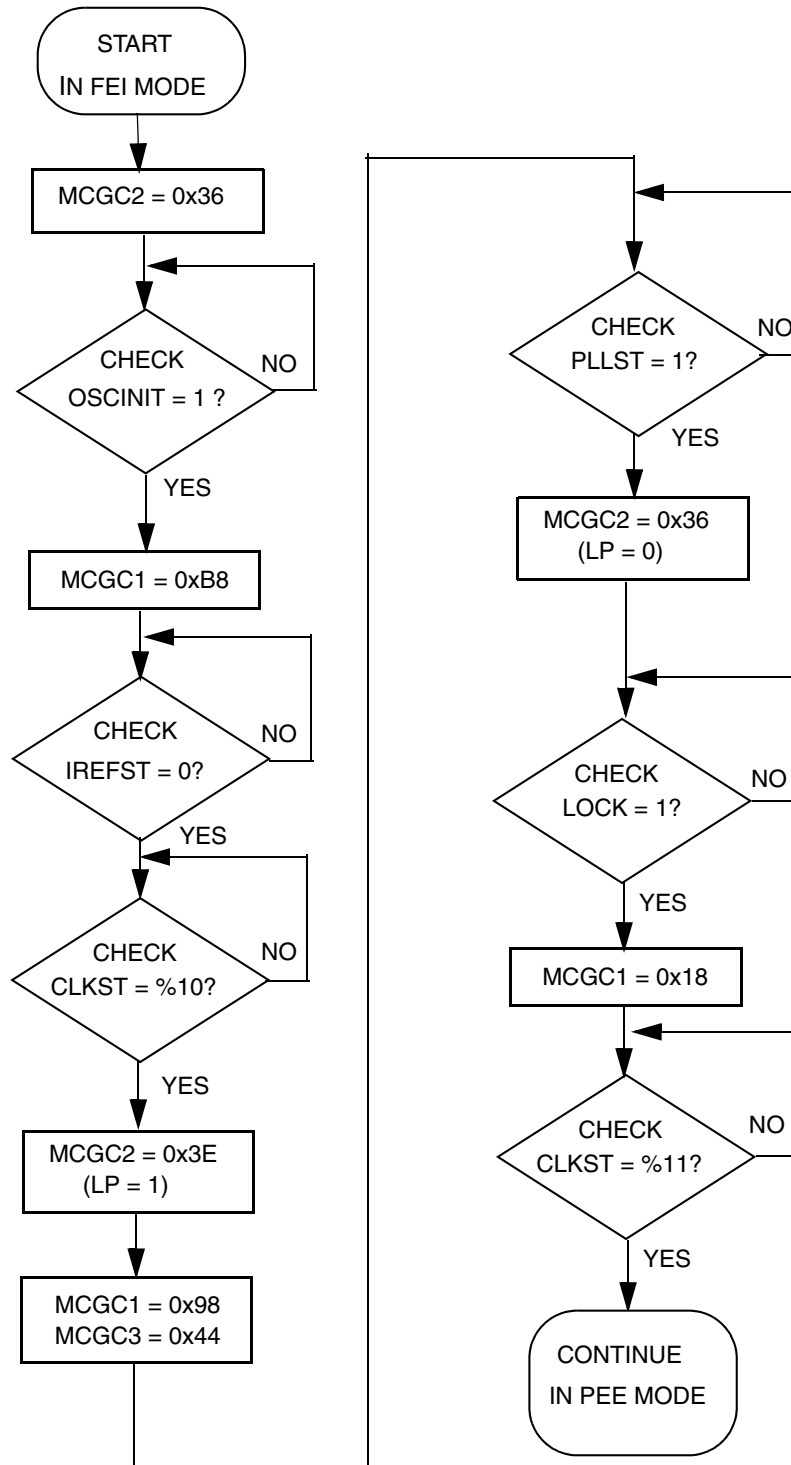


Figure 9-12. Flowchart of FEI to PEE Mode Transition using a 8 MHz crystal

9.5.3 Calibrating the Internal Reference Clock (IRC)

The IRC is calibrated by writing to the MCGTRM register first, then using the FTRIM bit to “fine tune” the frequency. We will refer to this total 9-bit value as the trim value, ranging from 0x000 to 0x1FF, where the FTRIM bit is the LSB.

The trim value after a POR is always 0x100 (MCGTRM = 0x80 and FTRIM = 0). Writing a larger value will decrease the frequency and smaller values will increase the frequency. The trim value is linear with the period, except that slight variations in wafer fab processing produce slight non-linearities between trim value and period. These non-linearities are why an iterative trimming approach to search for the best trim value is recommended. In Example #5: Internal Reference Clock Trim this approach will be demonstrated.

After a trim value has been found for a device, this value can be stored in FLASH memory to save the value. If power is removed from the device, the IRC can easily be re-trimmed by copying the saved value from FLASH to the MCG registers. Freescale identifies recommended FLASH locations for storing the trim value for each MCU. Consult the memory map in the data sheet for these locations. On devices that are factory trimmed, the factory trim value will be stored in these locations.

9.5.3.1 Example #5: Internal Reference Clock Trim

For applications that require a tight frequency tolerance, a trimming procedure is provided that will allow a very accurate internal clock source. This section outlines one example of trimming the internal oscillator. Many other possible trimming procedures are valid and can be used.

In the example below, the MCG trim will be calibrated for the 9-bit MCGTRM and FTRIM collective value. This value will be referred to as TRMVAL.

Initial conditions:

- 1) Clock supplied from ATE has 500 μ s duty period
- 2) MCG configured for internal reference with 8MHz bus

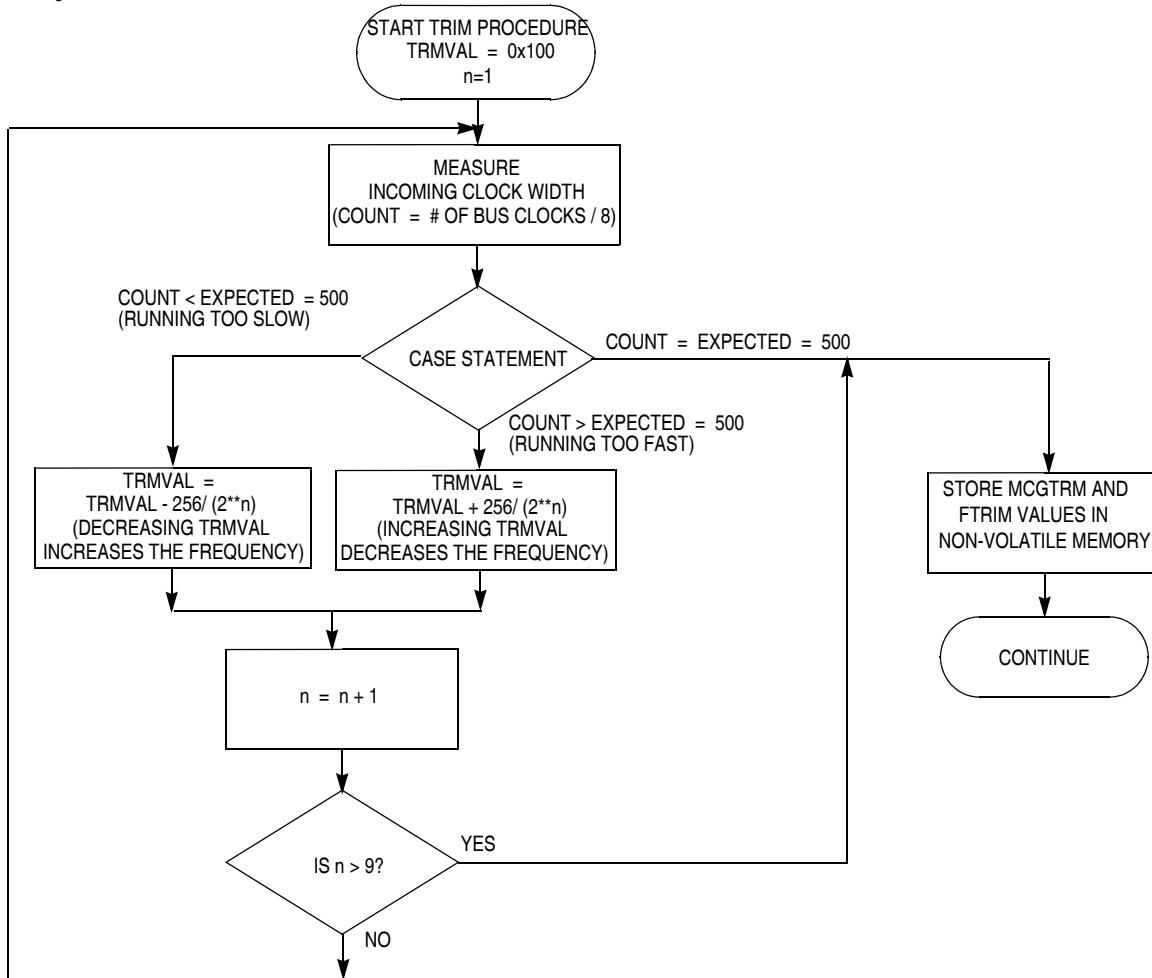


Figure 9-13. Trim Procedure

In this particular case, the MCU has been attached to a PCB and the entire assembly is undergoing final test with automated test equipment. A separate signal or message is provided to the MCU operating under user provided software control. The MCU initiates a trim procedure as outlined in [Figure 9-13](#) while the tester supplies a precision reference signal.

If the intended bus frequency is near the maximum allowed for the device, it is recommended to trim using a reference divider value (RDIV setting) of twice the final value. After the trim procedure is complete, the reference divider can be restored. This will prevent accidental overshoot of the maximum clock frequency.

Chapter 10

Modulo Timer (S08MTIMV1)

10.1 Introduction

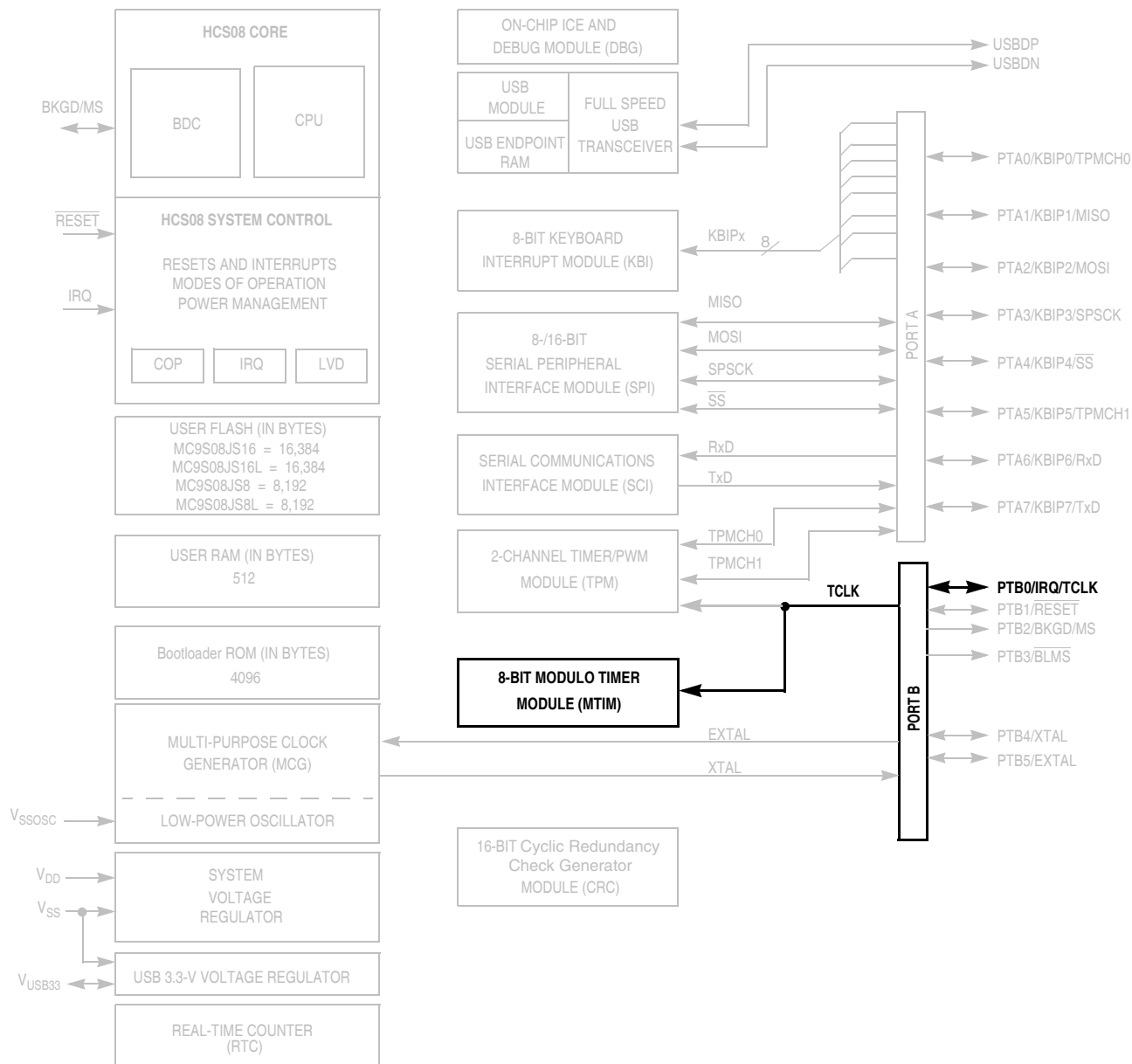
The MTIM is a simple 8-bit timer with several software selectable clock sources and a programmable interrupt.

The central component of the MTIM is the 8-bit counter, which can operate as a free-running counter or a modulo counter. A timer overflow interrupt can be enabled to generate periodic interrupts for time-based software loops.

Figure 10-1 shows the MC9S08JS16 series block diagram with the MTIM highlighted.

10.1.1 MTIM Configuration Information

The external clock for the MTIM module, $TCLK$, is selected by setting $CLKS = 1:1$ or $1:0$ in $MTIMCLK$, which selects the $TCLK$ pin input. The $TCLK$ input on $PTB0$ can be enabled as external clock inputs to both MTIM and TPM modules simultaneously.



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1).
Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD} . IRQ must not be driven above V_{DD} .
4. \overline{RESET} contains integrated pullup device if PTB1 enabled as reset pin function (RSTPE = 1)
5. Pin contains integrated pullup device.
6. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 10-1. MC9S08JS16 Series Block Diagram Highlighting MTIM Block and Pin

10.1.2 Features

Timer system features include:

- 8-bit up-counter
 - Free-running or 8-bit modulo limit
 - Software controllable interrupt on overflow
 - Counter reset bit (TRST)
 - Counter stop bit (TSTP)
- Four software selectable clock sources for input to prescaler:
 - System bus clock — rising edge
 - Fixed frequency clock (XCLK) — rising edge
 - External clock source on the TCLK pin — rising edge
 - External clock source on the TCLK pin — falling edge
- Nine selectable clock prescale values:
 - Clock source divide by 1, 2, 4, 8, 16, 32, 64, 128, or 256

10.1.3 Modes of Operation

This section defines the MTIM's operation in stop, wait and background debug modes.

10.1.3.1 MTIM in Wait Mode

The MTIM continues to run in wait mode if enabled before executing the WAIT instruction. Therefore, the MTIM can be used to bring the MCU out of wait mode if the timer overflow interrupt is enabled. For lowest possible current consumption, the MTIM should be stopped by software if not needed as an interrupt source during wait mode.

10.1.3.2 MTIM in Stop Modes

The MTIM is disabled in all stop modes, regardless of the settings before executing the STOP instruction. Therefore, the MTIM cannot be used as a wakeup source from stop modes.

Waking from stop1 and stop2 modes, the MTIM will be put into its reset state. If stop3 is exited with a reset, the MTIM will be put into its reset state. If stop3 is exited with an interrupt, the MTIM continues from the state it was in when stop3 was entered. If the counter was active upon entering stop3, the count will resume from the current value.

10.1.3.3 MTIM in Active Background Mode

The MTIM suspends all counting until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as an MTIM reset did not occur (TRST written to a 1 or MTIMMOD written).

10.1.4 Block Diagram

The block diagram for the modulo timer module is shown [Figure 10-2](#).

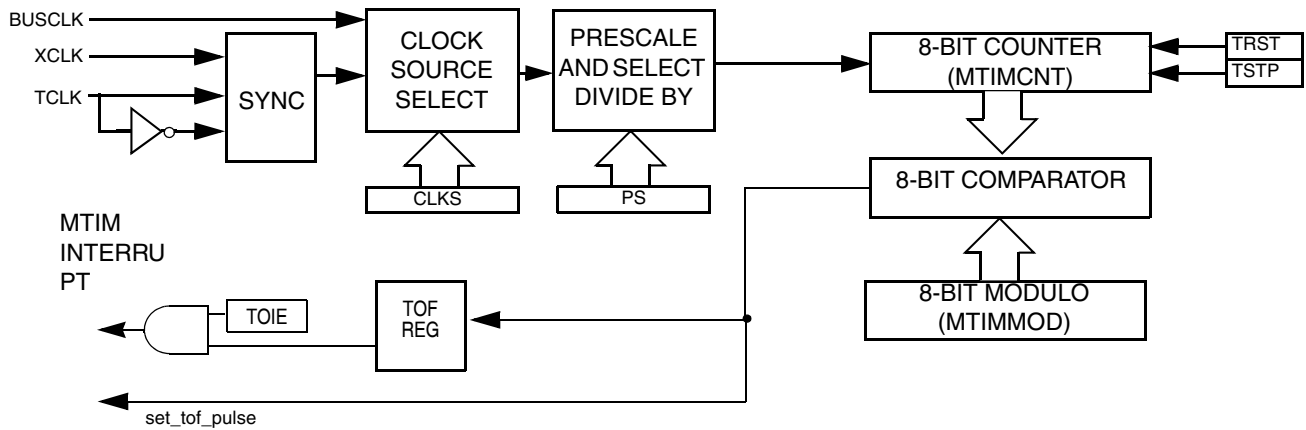


Figure 10-2. Modulo Timer (MTIM) Block Diagram

10.2 External Signal Description

The MTIM includes one external signal, TCLK, used to input an external clock when selected as the MTIM clock source. The signal properties of TCLK are shown in [Table 10-1](#).

Table 10-1. Signal Properties

Signal	Function	I/O
TCLK	External clock source input into MTIM	I

The TCLK input must be synchronized by the bus clock. Also, variations in duty cycle and clock jitter must be accommodated. Therefore, the TCLK signal must be limited to one-fourth of the bus frequency.

The TCLK pin can be muxed with a general-purpose port pin. See the [Pins and Connections](#) chapter for the pin location and priority of this function.

10.3 Register Definition

[Figure 10-3](#) is a summary of MTIM registers.

Figure 10-3. MTIM Register Summary

Name		7	6	5	4	3	2	1	0
MTIMSC	R	TOF	TOIE	0	TSTP	0	0	0	0
	W			TRST					
MTIMCLK	R	0	0	CLKS		PS			
	W								
MTIMCNT	R	COUNT							
	W								
MTIMMOD	R	MOD							
	W								

Each MTIM includes four registers:

- An 8-bit status and control register
- An 8-bit clock configuration register
- An 8-bit counter register
- An 8-bit modulo register

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet for the absolute address assignments for all MTIM registers. This section refers to registers and control bits only by their names and relative address offsets.

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

10.3.1 MTIM Status and Control Register (MTIMSC)

MTIMSC contains the overflow status flag and control bits which are used to configure the interrupt enable, reset the counter, and stop the counter.

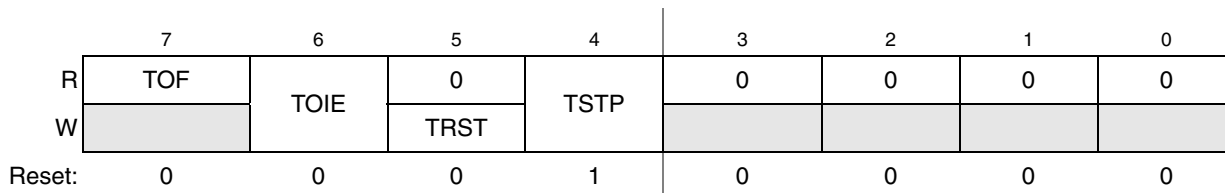


Figure 10-4. MTIM Status and Control Register

Table 10-2. MTIM Status and Control Register Field Descriptions

Field	Description
7 TOF	MTIM Overflow Flag — This read-only bit is set when the MTIM counter register overflows to 0x00 after reaching the value in the MTIM modulo register. Clear TOF by reading the MTIMSC register while TOF is set, then writing a 0 to TOF. TOF is also cleared when TRST is written to a 1 or when any value is written to the MTIMMOD register. 0 MTIM counter has not reached the overflow value in the MTIM modulo register. 1 MTIM counter has reached the overflow value in the MTIM modulo register.
6 TOIE	MTIM Overflow Interrupt Enable — This read/write bit enables MTIM overflow interrupts. If TOIE is set, then an interrupt is generated when TOF = 1. Reset clears TOIE. Do not set TOIE if TOF = 1. Clear TOF first, then set TOIE. 0 TOF interrupts are disabled. Use software polling. 1 TOF interrupts are enabled.
5 TRST	MTIM Counter Reset — When a 1 is written to this write-only bit, the MTIM counter register resets to 0x00 and TOF is cleared. Reading this bit always returns 0. 0 No effect. MTIM counter remains at current state. 1 MTIM counter is reset to 0x00.
4 TSTP	MTIM Counter Stop — When set, this read/write bit stops the MTIM counter at its current value. Counting resumes from the current value when TSTP is cleared. Reset sets TSTP to prevent the MTIM from counting. 0 MTIM counter is active. 1 MTIM counter is stopped.
3:0	Unused register bits, always read 0.

10.3.2 MTIM Clock Configuration Register (MTIMCLK)

MTIMCLK contains the clock select bits (CLKS) and the prescaler select bits (PS).

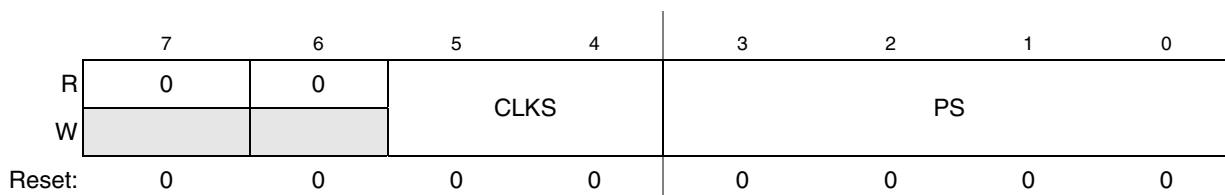


Figure 10-5. MTIM Clock Configuration Register

Table 10-3. MTIM Clock Configuration Register Field Description

Field	Description
7:6	Unused register bits, always read 0.
5:4 CLKS	<p>Clock Source Select — These two read/write bits select one of four different clock sources as the input to the MTIM prescaler. Changing the clock source while the counter is active does not clear the counter. The count continues with the new clock source. Reset clears CLKS to 000.</p> <p>00 Encoding 0. Bus clock (BUSCLK) 01 Encoding 1. Fixed-frequency clock (XCLK) 10 Encoding 3. External source (TCLK pin), falling edge 11 Encoding 4. External source (TCLK pin), rising edge All other encodings default to the bus clock (BUSCLK).</p>
3:0 PS	<p>Clock Source Prescaler — These four read/write bits select one of nine outputs from the 8-bit prescaler. Changing the prescaler value while the counter is active does not clear the counter. The count continues with the new prescaler value. Reset clears PS to 0000.</p> <p>0000 Encoding 0. MTIM clock source ÷ 1 0001 Encoding 1. MTIM clock source ÷ 2 0010 Encoding 2. MTIM clock source ÷ 4 0011 Encoding 3. MTIM clock source ÷ 8 0100 Encoding 4. MTIM clock source ÷ 16 0101 Encoding 5. MTIM clock source ÷ 32 0110 Encoding 6. MTIM clock source ÷ 64 0111 Encoding 7. MTIM clock source ÷ 128 1000 Encoding 8. MTIM clock source ÷ 256 All other encodings default to MTIM clock source ÷ 256.</p>

10.3.3 MTIM Counter Register (MTIMCNT)

MTIMCNT is the read-only value of the current MTIM count of the 8-bit counter.

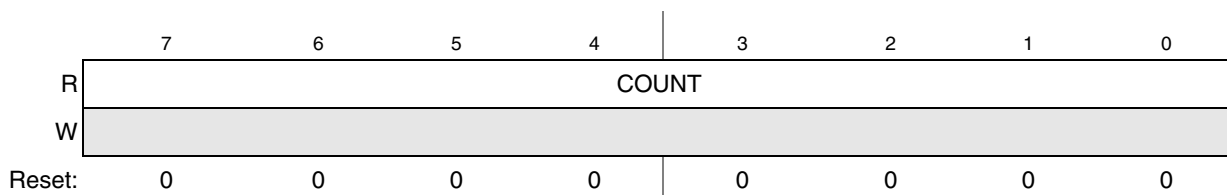


Figure 10-6. MTIM Counter Register

Table 10-4. MTIM Counter Register Field Description

Field	Description
7:0 COUNT	MTIM Count — These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset clears the count to 0x00.

10.3.4 MTIM Modulo Register (MTIMMOD)

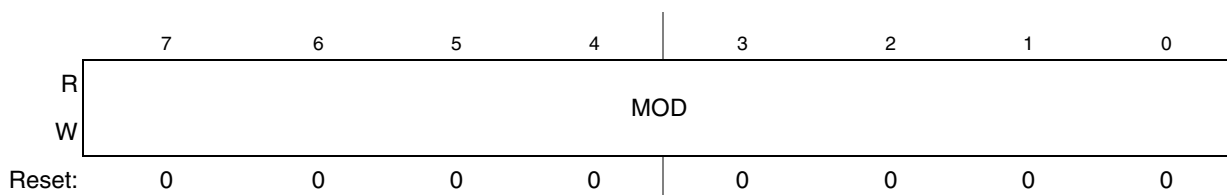


Figure 10-7. MTIM Modulo Register

Table 10-5. MTIM Modulo Register Field Descriptions

Field	Description
7:0 MOD	MTIM Modulo — These eight read/write bits contain the modulo value used to reset the count and set TOF. A value of 0x00 puts the MTIM in free-running mode. Writing to MTIMMOD resets the COUNT to 0x00 and clears TOF. Reset sets the modulo to 0x00.

10.4 Functional Description

The MTIM is composed of a main 8-bit up-counter with an 8-bit modulo register, a clock source selector, and a prescaler block with nine selectable values. The module also contains software selectable interrupt logic.

The MTIM counter (MTIMCNT) has three modes of operation: stopped, free-running, and modulo. Out of reset, the counter is stopped. If the counter is started without writing a new value to the modulo register, then the counter will be in free-running mode. The counter is in modulo mode when a value other than 0x00 is in the modulo register while the counter is running.

After any MCU reset, the counter is stopped and reset to 0x00, and the modulus is set to 0x00. The bus clock is selected as the default clock source and the prescale value is divide by 1. To start the MTIM in free-running mode, simply write to the MTIM status and control register (MTIMSC) and clear the MTIM stop bit (TSTP).

Four clock sources are software selectable: the internal bus clock, the fixed frequency clock (XCLK), and an external clock on the TCLK pin, selectable as incrementing on either rising or falling edges. The MTIM clock select bits (CLKS1:CLKS0) in MTIMSC are used to select the desired clock source. If the counter is active (TSTP = 0) when a new clock source is selected, the counter will continue counting from the previous value using the new clock source.

Nine prescale values are software selectable: clock source divided by 1, 2, 4, 8, 16, 32, 64, 128, or 256. The prescaler select bits (PS[3:0]) in MTIMSC select the desired prescale value. If the counter is active (TSTP = 0) when a new prescaler value is selected, the counter will continue counting from the previous value using the new prescaler value.

The MTIM modulo register (MTIMMOD) allows the overflow compare value to be set to any value from 0x01 to 0xFF. Reset clears the modulo value to 0x00, which results in a free running counter.

When the counter is active (TSTP = 0), the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter overflows to 0x00 and continues counting. The MTIM overflow flag (TOF) is set whenever the counter overflows. The flag sets on the transition from the modulo value to 0x00. Writing to MTIMMOD while the counter is active resets the counter to 0x00 and clears TOF.

Clearing TOF is a two-step process. The first step is to read the MTIMSC register while TOF is set. The second step is to write a 0 to TOF. If another overflow occurs between the first and second steps, the clearing process is reset and TOF will remain set after the second step is performed. This will prevent the second occurrence from being missed. TOF is also cleared when a 1 is written to TRST or when any value is written to the MTIMMOD register.

The MTIM allows for an optional interrupt to be generated whenever TOF is set. To enable the MTIM overflow interrupt, set the MTIM overflow interrupt enable bit (TOIE) in MTIMSC. TOIE should never be written to a 1 while TOF = 1. Instead, TOF should be cleared first, then the TOIE can be set to 1.

10.4.1 MTIM Operation Example

This section shows an example of the MTIM operation as the counter reaches a matching value from the modulo register.

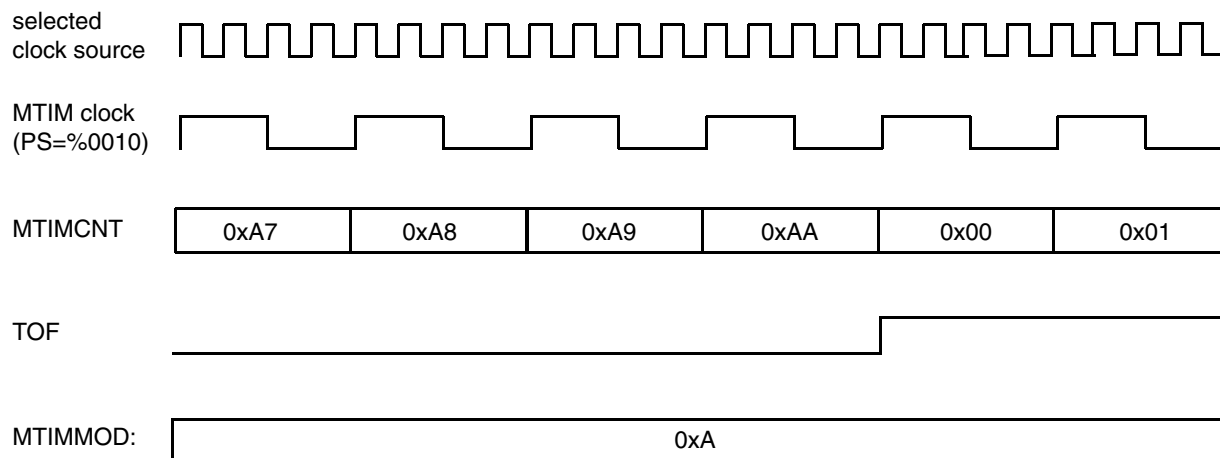


Figure 10-8. MTIM counter overflow example

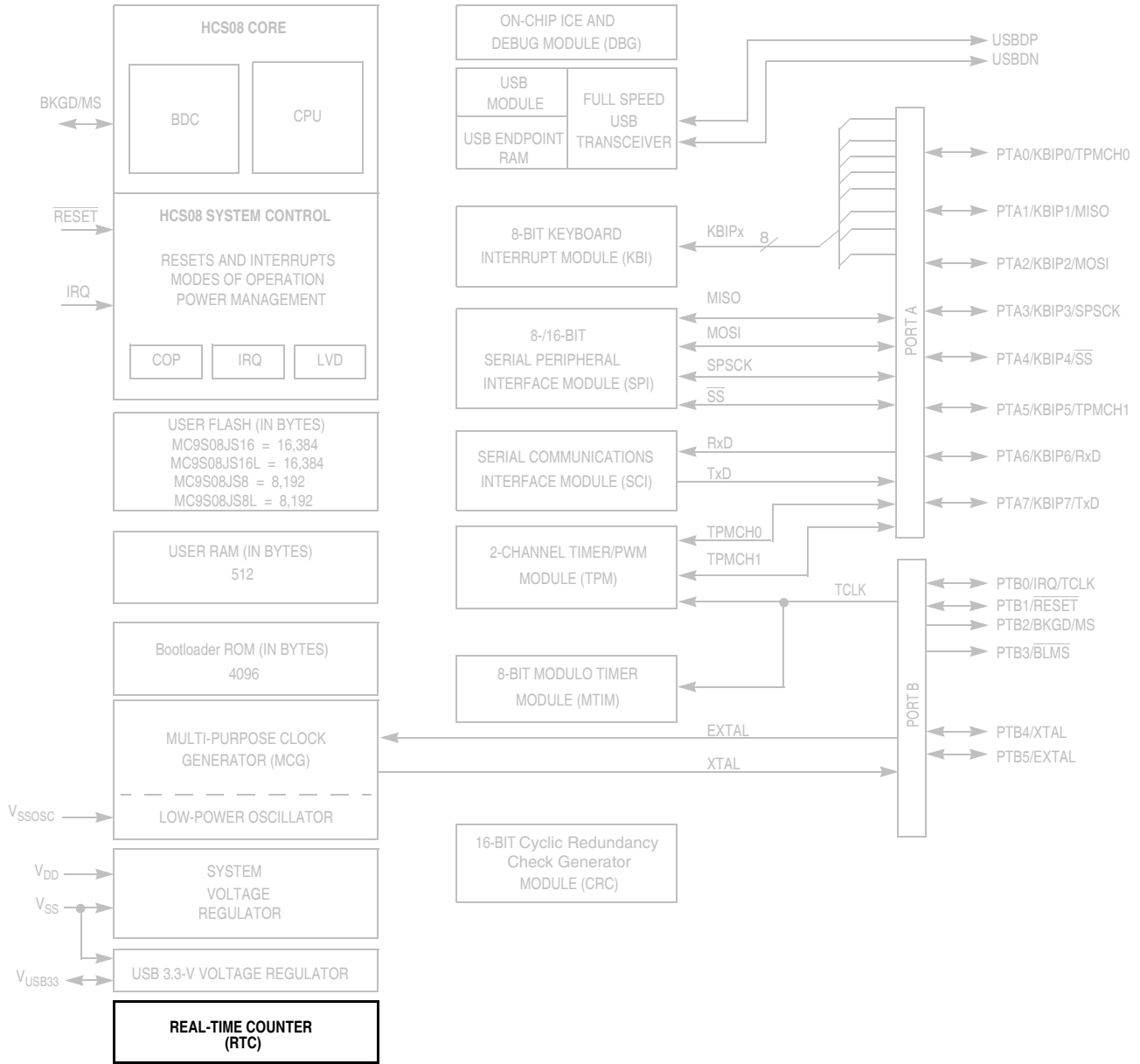
In the example of [Figure 10-8](#), the selected clock source could be any of the five possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIMMOD register is set to 0xAA. When the counter, MTIMCNT, reaches the modulo value of 0xAA, the counter overflows to 0x00 and continues counting. The timer overflow flag, TOF, sets when the counter value changes from 0xAA to 0x00. An MTIM overflow interrupt is generated when TOF is set, if TOIE = 1.

Chapter 11

Real-Time Counter (S08RTCV1)

11.1 Introduction

The real-time counter (RTC) consists of one 8-bit counter, one 8-bit comparator, several binary-based and decimal-based prescaler dividers, two clock sources, and one programmable periodic interrupt. This module can be used for time-of-day, calendar or any task scheduling functions. It can also serve as a cyclic wakeup from low power modes without the need of external components.



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1).
Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ must not be driven above V_{DD}.
4. RESET contains integrated pullup device if PTB1 enabled as reset pin function (RSTPE = 1)
5. Pin contains integrated pullup device.
6. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 11-1. MC9S08JS16 Series Block Diagram Highlighting RTC Block

11.1.1 Features

Features of the RTC module include:

- 8-bit up-counter
 - 8-bit modulo match limit
 - Software controllable periodic interrupt on match
- Three software selectable clock sources for input to prescaler with selectable binary-based and decimal-based divider values
 - 1-kHz internal low-power oscillator (LPO)
 - External clock (ERCLK)
 - 32-kHz internal clock (IRCLK)

11.1.2 Modes of Operation

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

11.1.2.1 Wait Mode

The RTC continues to run in wait mode if enabled before executing the appropriate instruction. Therefore, the RTC can bring the MCU out of wait mode if the real-time interrupt is enabled. For lowest possible current consumption, the RTC should be stopped by software if not needed as an interrupt source during wait mode.

11.1.2.2 Stop Modes

The RTC continues to run in stop2 or stop3 mode if the RTC is enabled before executing the STOP instruction. Therefore, the RTC can bring the MCU out of stop modes with no external components, if the real-time interrupt is enabled.

The LPO clock can be used in stop2 and stop3 modes. ERCLK and IRCLK clocks are only available in stop3 mode.

Power consumption is lower when all clock sources are disabled, but in that case, the real-time interrupt cannot wake the MCU from stop modes.

11.1.2.3 Active Background Mode

The RTC suspends all counting during active background mode until the microcontroller returns to normal user operating mode. Counting resumes from the suspended value as long as the RTCMOD register is not written and the RTCPS and RTCLKS bits are not altered.

11.1.3 Block Diagram

The block diagram for the RTC module is shown in [Figure 11-2](#).

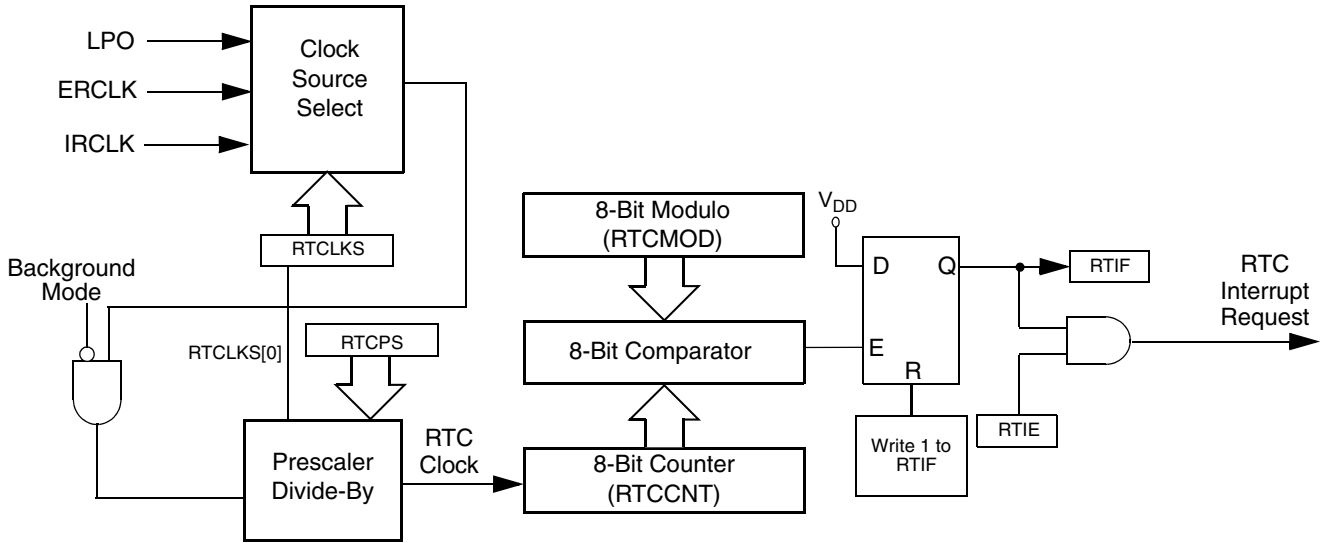


Figure 11-2. Real-Time Counter (RTC) Block Diagram

11.2 External Signal Description

The RTC does not include any off-chip signals.

11.3 Register Definition

The RTC includes a status and control register, an 8-bit counter register, and an 8-bit modulo register.

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

[Table 11-1](#) is a summary of RTC registers.

Table 11-1. RTC Register Summary

Name		7	6	5	4	3	2	1	0
RTCSC	R	RTIF	RTCLKS		RTIE	RTCPS			
	W								
RTCCNT	R	RTCCNT							
	W								
RTCMOD	R	RTCMOD							
	W								

11.3.1 RTC Status and Control Register (RTCSC)

RTCSC contains the real-time interrupt status flag (RTIF), the clock select bits (RTCLKS), the real-time interrupt enable bit (RTIE), and the prescaler select bits (RTCPS).

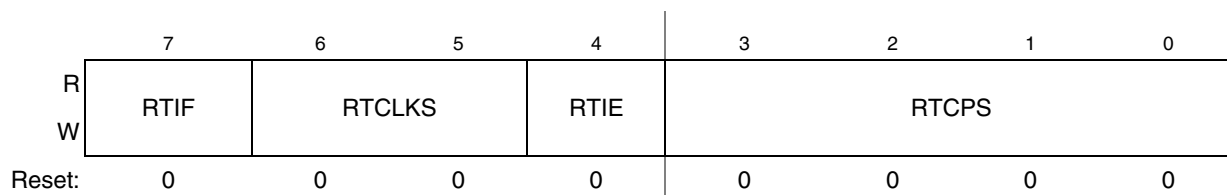


Figure 11-3. RTC Status and Control Register (RTCSC)

Table 11-2. RTCSC Field Descriptions

Field	Description
7 RTIF	Real-Time Interrupt Flag This status bit indicates the RTC counter register reached the value in the RTC modulo register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RTIF. 0 RTC counter has not reached the value in the RTC modulo register. 1 RTC counter has reached the value in the RTC modulo register.
6–5 RTCLKS	Real-Time Clock Source Select. These two read/write bits select the clock source input to the RTC prescaler. Changing the clock source clears the prescaler and RTCCNT counters. When selecting a clock source, ensure that the clock source is properly enabled (if applicable) to ensure correct operation of the RTC. Reset clears RTCLKS. 00 Real-time clock source is the 1-kHz low power oscillator (LPO) 01 Real-time clock source is the external clock (ERCLK) 1x Real-time clock source is the internal clock (IRCLK)
4 RTIE	Real-Time Interrupt Enable. This read/write bit enables real-time interrupts. If RTIE is set, then an interrupt is generated when RTIF is set. Reset clears RTIE. 0 Real-time interrupt requests are disabled. Use software polling. 1 Real-time interrupt requests are enabled.
3–0 RTCPS	Real-Time Clock Prescaler Select. These four read/write bits select binary-based or decimal-based divide-by values for the clock source. See Table 11-3 . Changing the prescaler value clears the prescaler and RTCCNT counters. Reset clears RTCPS.

Table 11-3. RTC Prescaler Divide-by values

RTCLKS[0]	RTCPS															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Off	2 ³	2 ⁵	2 ⁶	2 ⁷	2 ⁸	2 ⁹	2 ¹⁰	1	2	2 ²	10	2 ⁴	10 ²	5x10 ²	10 ³
1	Off	2 ¹⁰	2 ¹¹	2 ¹²	2 ¹³	2 ¹⁴	2 ¹⁵	2 ¹⁶	10 ³	2x10 ³	5x10 ³	10 ⁴	2x10 ⁴	5x10 ⁴	10 ⁵	2x10 ⁵

11.3.2 RTC Counter Register (RTCCNT)

RTCCNT is the read-only value of the current RTC count of the 8-bit counter.

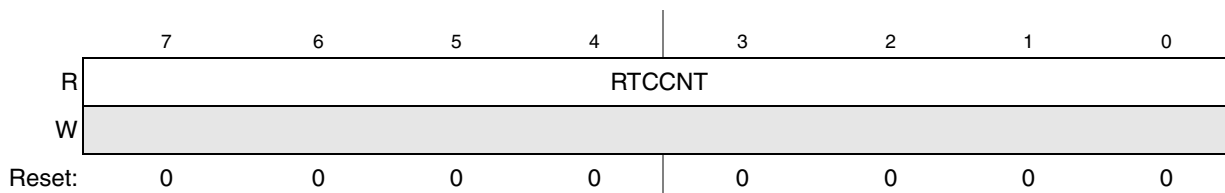


Figure 11-4. RTC Counter Register (RTCCNT)

Table 11-4. RTCCNT Field Descriptions

Field	Description
7:0 RTCCNT	RTC Count. These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset, writing to RTCMOD, or writing different values to RTCLKS and RTCPS clear the count to 0x00.

11.3.3 RTC Modulo Register (RTCMOD)

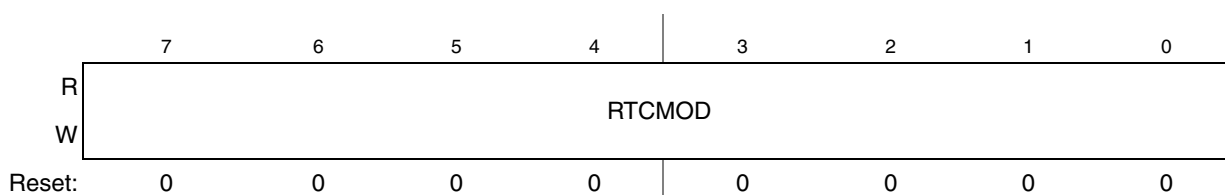


Figure 11-5. RTC Modulo Register (RTCMOD)

Table 11-5. RTCMOD Field Descriptions

Field	Description
7:0 RTCMOD	RTC Modulo. These eight read/write bits contain the modulo value used to reset the count to 0x00 upon a compare match and set the RTIF status bit. A value of 0x00 sets the RTIF bit on each rising edge of the prescaler output. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00. Reset sets the modulo to 0x00.

11.4 Functional Description

The RTC is composed of a main 8-bit up-counter with an 8-bit modulo register, a clock source selector, and a prescaler block with binary-based and decimal-based selectable values. The module also contains software selectable interrupt logic.

After any MCU reset, the counter is stopped and reset to 0x00, the modulus register is set to 0x00, and the prescaler is off. The 1-kHz internal oscillator clock is selected as the default clock source. To start the prescaler, write any value other than zero to the prescaler select bits (RTCPS).

Three clock sources are software selectable: the low power oscillator clock (LPO), the external clock (ERCLK), and the internal clock (IRCLK). The RTC clock select bits (RTCLKS) select the desired clock source. If a different value is written to RTCLKS, the prescaler and RTCCNT counters are reset to 0x00.

RTCPS and the RTCLKS[0] bit select the desired divide-by value. If a different value is written to RTCPS, the prescaler and RTCCNT counters are reset to 0x00. Table 11-6 shows different prescaler period values.

Table 11-6. Prescaler Period

RTCPS	1-kHz Internal Clock (RTCLKS = 00)	1-MHz External Clock (RTCLKS = 01)	32-kHz Internal Clock (RTCLKS = 10)	32-kHz Internal Clock (RTCLKS = 11)
0000	Off	Off	Off	Off
0001	8 ms	1.024 ms	250 μ s	32 ms
0010	32 ms	2.048 ms	1 ms	64 ms
0011	64 ms	4.096 ms	2 ms	128 ms
0100	128 ms	8.192 ms	4 ms	256 ms
0101	256 ms	16.4 ms	8 ms	512 ms
0110	512 ms	32.8 ms	16 ms	1.024 s
0111	1.024 s	65.5 ms	32 ms	2.048 s
1000	1 ms	1 ms	31.25 μ s	31.25 ms
1001	2 ms	2 ms	62.5 μ s	62.5 ms
1010	4 ms	5 ms	125 μ s	156.25 ms
1011	10 ms	10 ms	312.5 μ s	312.5 ms
1100	16 ms	20 ms	0.5 ms	0.625 s
1101	0.1 s	50 ms	3.125 ms	1.5625 s
1110	0.5 s	0.1 s	15.625 ms	3.125 s
1111	1 s	0.2 s	31.25 ms	6.25 s

The RTC modulo register (RTCMOD) allows the compare value to be set to any value from 0x00 to 0xFF. When the counter is active, the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter resets to 0x00 and continues counting. The real-time interrupt flag (RTIF) is set when a match occurs. The flag sets on the transition from the modulo value to 0x00. Writing to RTCMOD resets the prescaler and the RTCCNT counters to 0x00.

The RTC allows for an interrupt to be generated when RTIF is set. To enable the real-time interrupt, set the real-time interrupt enable bit (RTIE) in RTCSC. RTIF is cleared by writing a 1 to RTIF.

11.4.1 RTC Operation Example

This section shows an example of the RTC operation as the counter reaches a matching value from the modulo register.

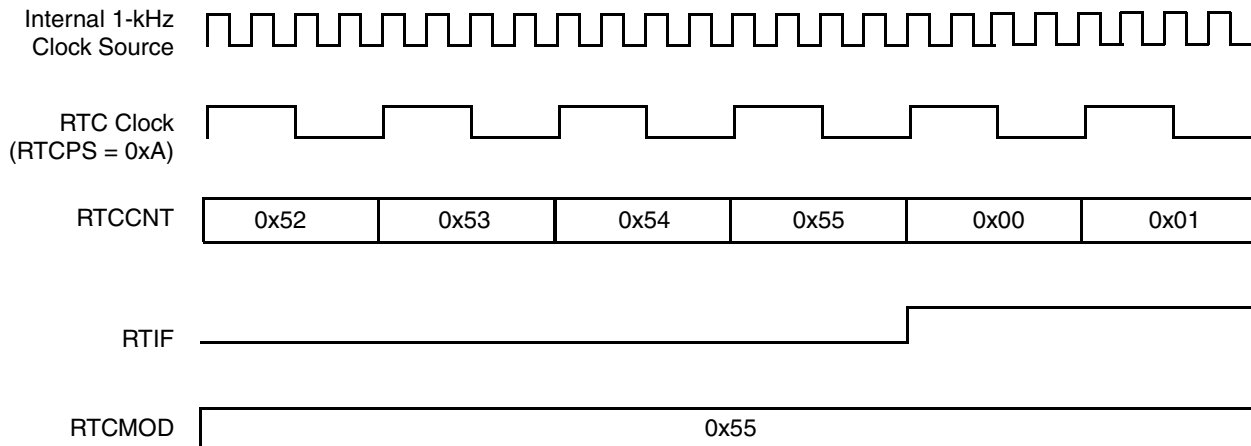


Figure 11-6. RTC Counter Overflow Example

In the example of [Figure 11-6](#), the selected clock source is the 1-kHz internal oscillator clock source. The prescaler (RTCPS) is set to 0xA or divide-by-4. The modulo value in the RTCMOD register is set to 0x55. When the counter, RTCCNT, reaches the modulo value of 0x55, the counter overflows to 0x00 and continues counting. The real-time interrupt flag, RTIF, sets when the counter value changes from 0x55 to 0x00. A real-time interrupt is generated when RTIF is set, if RTIE is set.

11.5 Initialization/Application Information

This section provides example code to give some basic direction to a user on how to initialize and configure the RTC module. The example software is implemented in C language.

The example below shows how to implement time of day with the RTC using the 1-kHz clock source to achieve the lowest possible power consumption. Because the 1-kHz clock source is not as accurate as a crystal, software can be added for any adjustments. For accuracy without adjustments at the expense of additional power consumption, the external clock (ERCLK) or the internal clock (IRCLK) can be selected with appropriate prescaler and modulo values.

```

/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;

/* Configure RTC to interrupt every 1 second from 1-kHz clock source */
RTCMOD.byte = 0x00;
RTCSC.byte = 0x1F;

/*****
Function Name : RTC_ISR
Notes : Interrupt service routine for RTC module.
*****/
#pragma TRAP_PROC
void RTC_ISR(void)
{
    /* Clear the interrupt flag */

```

```
RTCSC.byte = RTCSC.byte | 0x80;
/* RTC interrupts every 1 Second */
Seconds++;
/* 60 seconds in a minute */
if (Seconds > 59){
Minutes++;
Seconds = 0;
}
/* 60 minutes in an hour */
if (Minutes > 59){
Hours++;
Minutes = 0;
}
/* 24 hours in a day */
if (Hours > 23){
Days ++;
Hours = 0;
}
```



Chapter 12

Serial Communications Interface (S08SCIV4)

12.1 Introduction

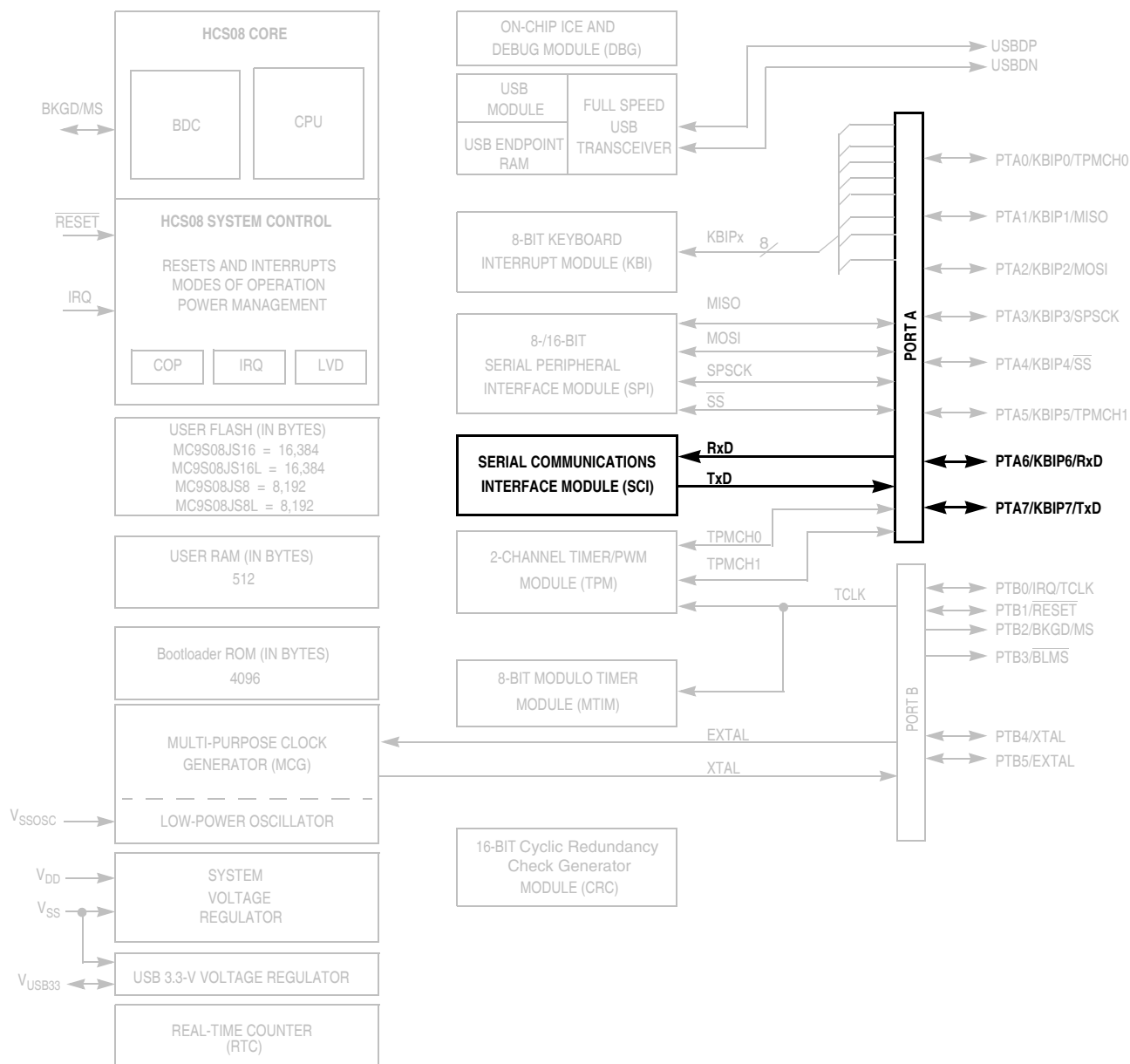
The MC9S08JS16 series include two independent serial communications interface (SCI) modules which are sometimes called universal asynchronous receiver/transmitters (UARTs). Typically, these systems are used to connect to the RS232 serial input/output (I/O) port of a personal computer or workstation, but they can also be used to communicate with other embedded controllers.

A flexible, 13-bit, modulo-based baud rate generator supports a broad range of standard baud rates beyond 115.2 kbaud. Transmit and receive within the same SCI use a common baud rate, and each SCI module has a separate baud rate generator.

This SCI system offers many advanced features not commonly found on other asynchronous serial I/O peripherals on other embedded controllers. The receiver employs an advanced data sampling technique that ensures reliable communication and noise detection. Hardware parity, receiver wakeup, and double buffering on transmit and receive are also included.

NOTE

MC9S08JS16 series devices operate at a higher voltage range (2.7 V to 5.5 V) and do not include stop1 mode. Therefore, please disregard references to stop1.



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1). Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD} . IRQ must not be driven above V_{DD} .
4. \overline{RESET} contains integrated pullup device if PTB1 enabled as reset pin function (RSTPE = 1)
5. Pin contains integrated pullup device.
6. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 12-1. MC9S08JS16 Series Block Diagram Highlighting the SCI Module and Pins

12.1.1 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
 - Transmit data register empty and transmission complete
 - Receive data register full
 - Receive overrun, parity error, framing error, and noise error
 - Idle receiver detect
 - Active edge on receive pin
 - Break detect supporting LIN
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark
- Optional 13-bit break character generation / 11-bit break character detection
- Selectable transmitter output polarity

12.1.2 Modes of Operation

See [Section 12.3, “Functional Description,”](#) For details concerning SCI operation in these modes:

- 8- and 9-bit data modes
- Stop mode operation
- Loop mode
- Single-wire mode

12.1.3 Block Diagram

[Figure 12-2](#) shows the transmitter portion of the SCI.

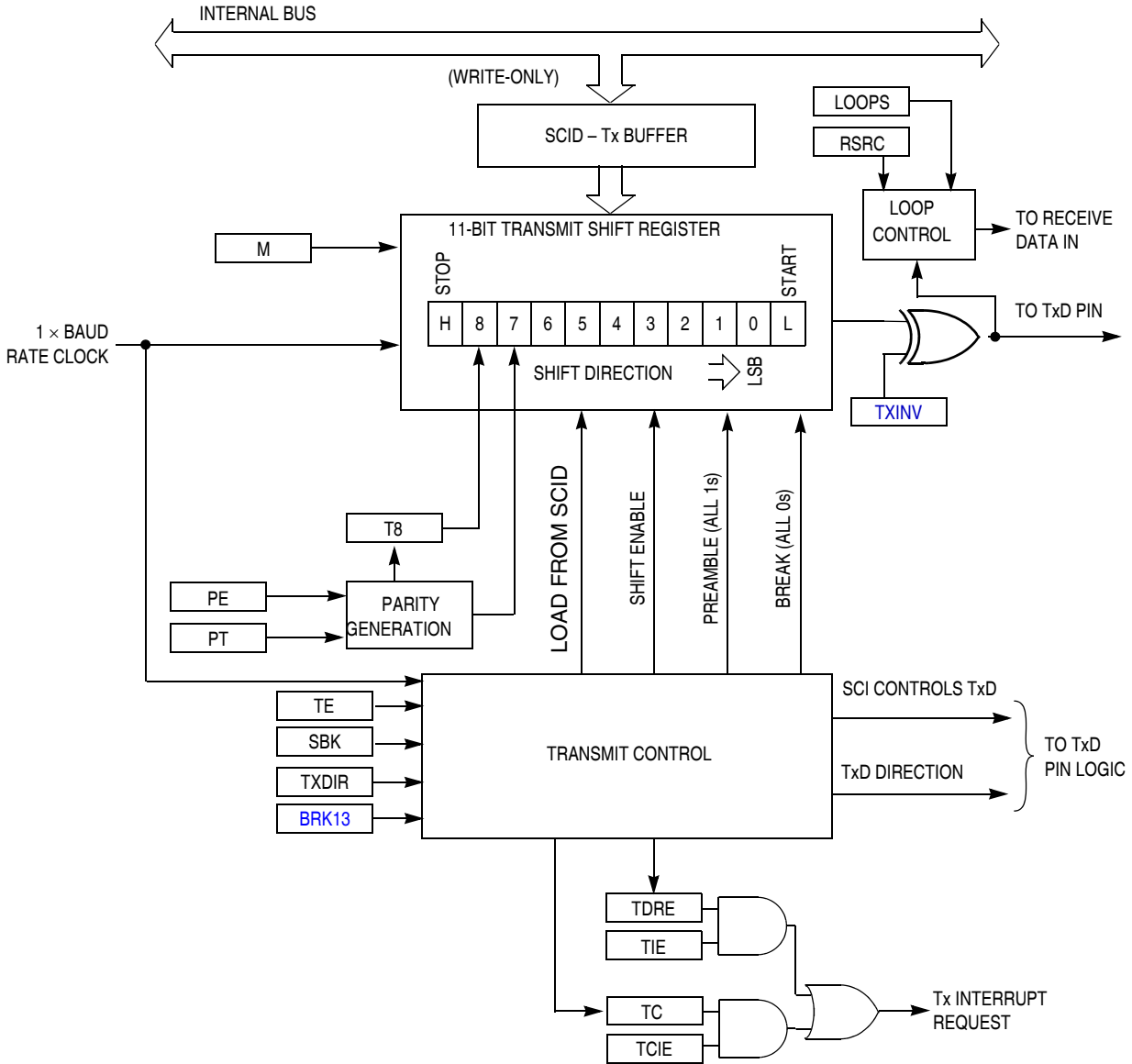


Figure 12-2. SCI Transmitter Block Diagram

Figure 12-3 shows the receiver portion of the SCI.

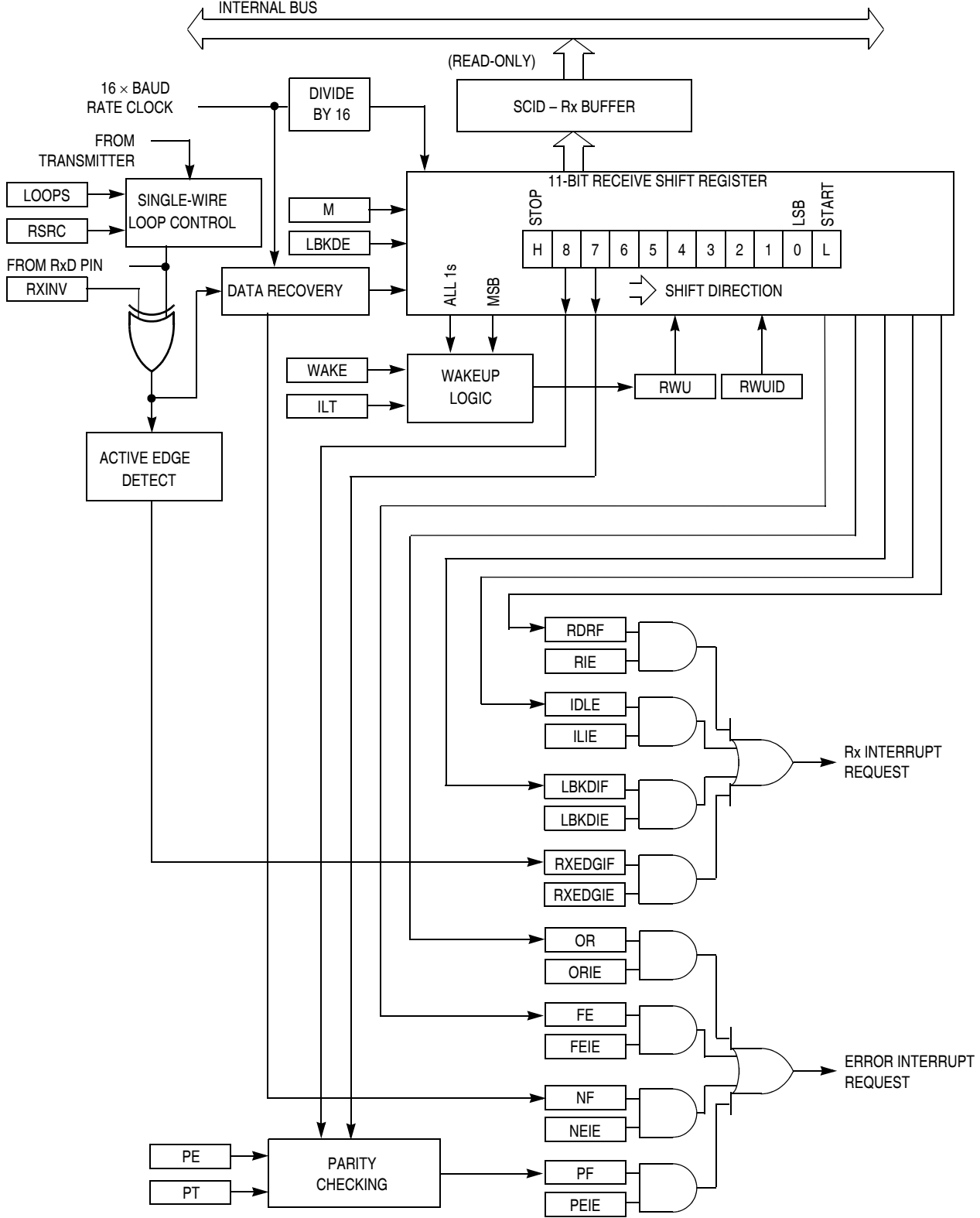


Figure 12-3. SCI Receiver Block Diagram

12.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.

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

12.2.1 SCI Baud Rate Registers (SCIBDH, SCIBDL)

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 SCIBDH to buffer the high half of the new value and then write to SCIBDL. The working value in SCIBDH does not change until SCIBDL is written.

SCIBDL 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 SCIC2 are written to 1).

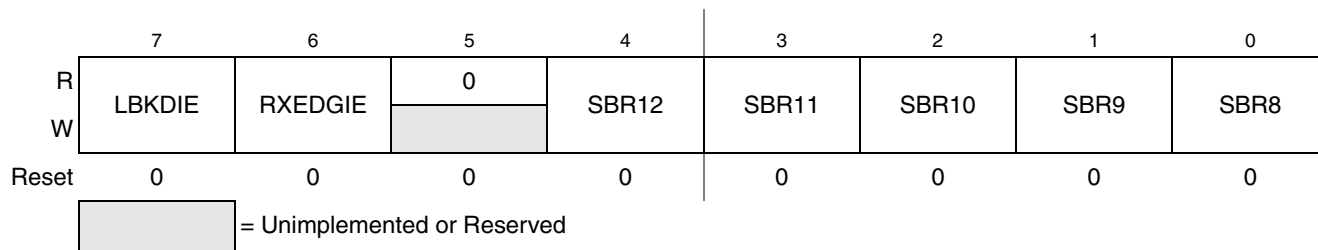


Figure 12-4. SCI Baud Rate Register (SCIBDH)

Table 12-1. SCIBDH 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 12-2 .

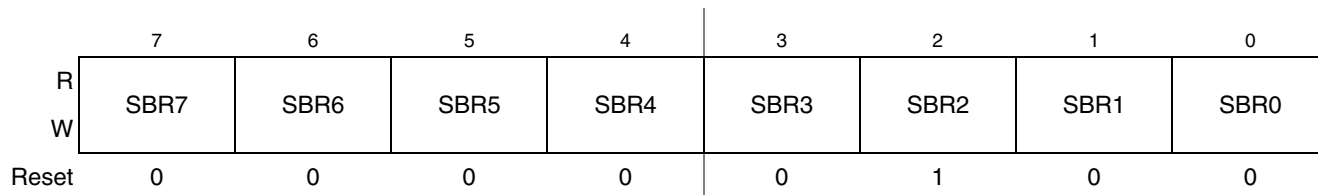


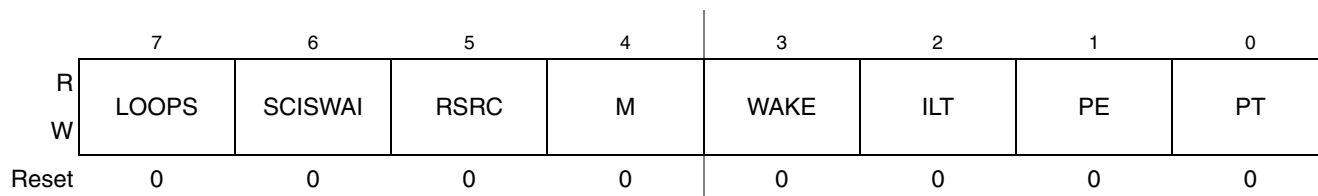
Figure 12-5. SCI Baud Rate Register (SCIBDL)

Table 12-2. SCIBDL 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 12-1 .

12.2.2 SCI Control Register 1 (SCIC1)

This read/write register is used to control various optional features of the SCI system.


Figure 12-6. SCI Control Register 1 (SCIC1)
Table 12-3. SCIC1 Field Descriptions

Field	Description
7 LOOPS	Loop Mode Select — Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input. 0 Normal operation — RxD and TxD use separate pins. 1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See RSRC bit.) RxD pin is not used by SCI.
6 SCISWAI	SCI Stops in Wait Mode 0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU. 1 SCI clocks freeze while CPU is in wait mode.
5 RSRC	Receiver Source Select — This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output. 0 Provided LOOPS = 1, RSRC = 0 selects internal loop back mode and the SCI does not use the RxD pins. 1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input.
4 M	9-Bit or 8-Bit Mode Select 0 Normal — start + 8 data bits (LSB first) + stop. 1 Receiver and transmitter use 9-bit data characters start + 8 data bits (LSB first) + 9th data bit + stop.
3 WAKE	Receiver Wakeup Method Select — Refer to Section 12.3.3.2, “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 12.3.3.2.1, “Idle-Line Wakeup” for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.

Table 12-3. SCIC1 Field Descriptions (continued)

Field	Description
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.

12.2.3 SCI Control Register 2 (SCIC2)

This register can be read or written at any time.

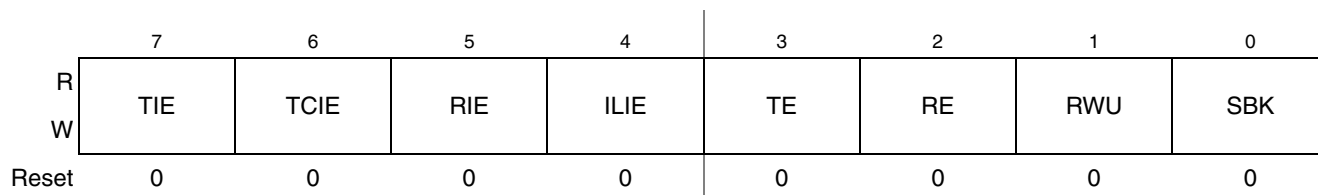


Figure 12-7. SCI Control Register 2 (SCIC2)

Table 12-4. SCIC2 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.
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 12.3.2.1, “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.

Table 12-4. SCIC2 Field Descriptions (continued)

Field	Description
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 12.3.3.2, “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 12.3.2.1, “Send Break and Queued Idle” for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

12.2.4 SCI Status Register 1 (SCIS1)

This register has eight read-only status flags. Writes have no effect. Special software sequences (which do not involve writing to this register) are used to clear these status flags.

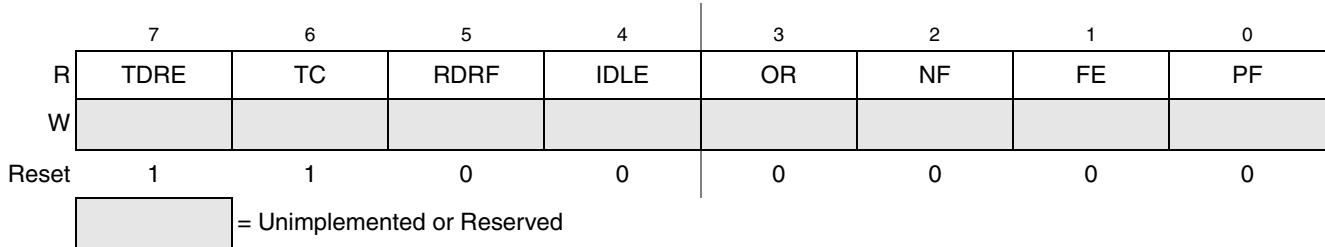


Figure 12-8. SCI Status Register 1 (SCIS1)

Table 12-5. SCIS1 Field Descriptions

Field	Description
7 TDRE	<p>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 SCIS1 with TDRE = 1 and then write to the SCI data register (SCID).</p> <p>0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.</p>
6 TC	<p>Transmission Complete Flag — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted.</p> <p>0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete).</p> <p>TC is cleared automatically by reading SCIS1 with TC = 1 and then doing one of the following three things:</p> <ul style="list-style-type: none"> • Write to the SCI data register (SCID) to transmit new data • Queue a preamble by changing TE from 0 to 1 • Queue a break character by writing 1 to SBK in SCIC2
5 RDRF	<p>Receive Data Register Full Flag — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCID). To clear RDRF, read SCIS1 with RDRF = 1 and then read the SCI data register (SCID).</p> <p>0 Receive data register empty. 1 Receive data register full.</p>
4 IDLE	<p>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.</p> <p>To clear IDLE, read SCIS1 with IDLE = 1 and then read the SCI data register (SCID). 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.</p> <p>0 No idle line detected. 1 Idle line was detected.</p>
3 OR	<p>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 SCID yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCID. To clear OR, read SCIS1 with OR = 1 and then read the SCI data register (SCID).</p> <p>0 No overrun. 1 Receive overrun (new SCI data lost).</p>
2 NF	<p>Noise Flag — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCIS1 and then read the SCI data register (SCID).</p> <p>0 No noise detected. 1 Noise detected in the received character in SCID.</p>

Table 12-5. SCIS1 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 SCIS1 with FE = 1 and then read the SCI data register (SCID). 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 SCIS1 and then read the SCI data register (SCID). 0 No parity error. 1 Parity error.

12.2.5 SCI Status Register 2 (SCIS2)

This register has one read-only status flag.

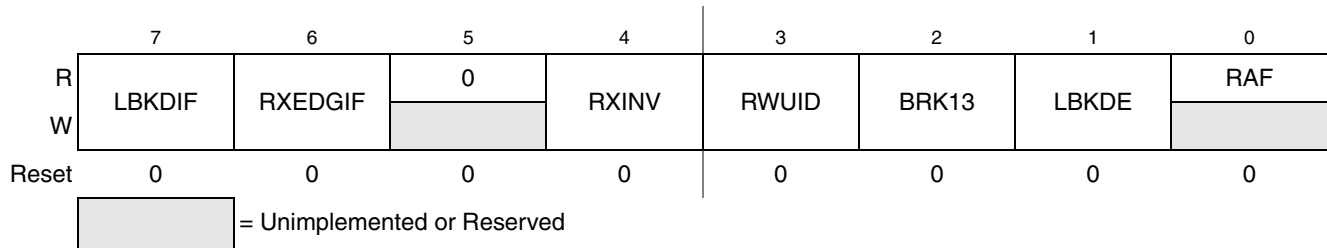


Figure 12-9. SCI Status Register 2 (SCIS2)

Table 12-6. SCIS2 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 WakeUp Idle Detect — RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. 0 During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. 1 During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character.
2 BRK13	Break Character Generation Length — BRK13 is used to select a longer transmitted break character length. Detection of a framing error is not affected by the state of this bit. 0 Break character is transmitted with length of 10 bit times (11 if M = 1) 1 Break character is transmitted with length of 13 bit times (14 if M = 1)

Table 12-6. SCIS2 Field Descriptions (continued)

Field	Description
1 LBKDE	LIN Break Detection Enable — LBKDE is used to select a longer break character detection length. While LBKDE is set, framing error (FE) and receive data register full (RDRF) flags are prevented from setting. 0 Break character is detected at length of 10 bit times (11 if M = 1). 1 Break character is detected at length of 11 bit times (12 if M = 1).
0 RAF	Receiver Active Flag — RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode. 0 SCI receiver idle waiting for a start bit. 1 SCI receiver active (RxD input not idle).

¹ Setting RXINV inverts the RxD input for all cases: data bits, start and stop bits, break, and idle.

When using an internal oscillator in a LIN system, it is necessary to raise the break detection threshold by one bit time. Under the worst case timing conditions allowed in LIN, it is possible that a 0x00 data character can appear to be 10.26 bit times long at a slave which is running 14% faster than the master. This would trigger normal break detection circuitry which is designed to detect a 10 bit break symbol. When the LBKDE bit is set, framing errors are inhibited and the break detection threshold changes from 10 bits to 11 bits, preventing false detection of a 0x00 data character as a LIN break symbol.

12.2.6 SCI Control Register 3 (SCIC3)

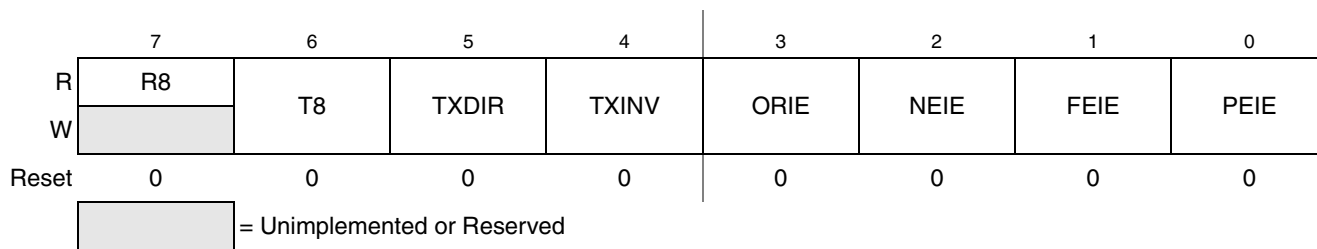


Figure 12-10. SCI Control Register 3 (SCIC3)

Table 12-7. SCIC3 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 SCID register. When reading 9-bit data, read R8 before reading SCID because reading SCID completes automatic flag clearing sequences which could allow R8 and SCID 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 SCID register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCID is written so T8 should be written (if it needs to change from its previous value) before SCID 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 SCID 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 12-7. SCIC3 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.

12.2.7 SCI Data Register (SCID)

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 12-11. SCI Data Register (SCID)

12.3 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.

12.3.1 Baud Rate Generation

As shown in [Figure 12-12](#), the clock source for the SCI baud rate generator is the bus-rate clock.

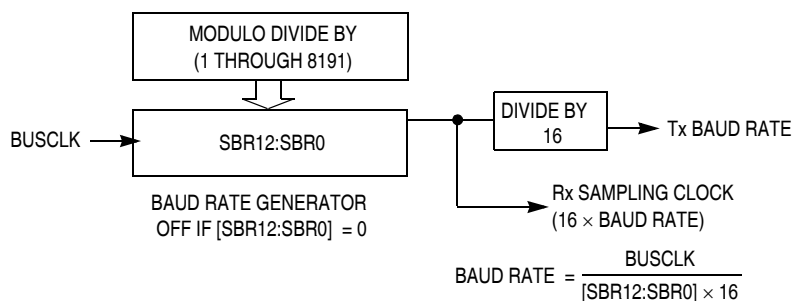


Figure 12-12. SCI Baud Rate Generation

SCI communications require the transmitter and receiver (which typically derive baud rates from independent clock sources) to use the same baud rate. Allowed tolerance on this baud frequency depends on the details of how the receiver synchronizes to the leading edge of the start bit and how bit sampling is performed.

The MCU resynchronizes to bit boundaries on every high-to-low transition, but in the worst case, there are no such transitions in the full 10- or 11-bit time character frame so any mismatch in baud rate is accumulated for the whole character time. For a Freescale Semiconductor SCI system whose bus frequency is driven by a crystal, the allowed baud rate mismatch is about ± 4.5 percent for 8-bit data format and about ± 4 percent for 9-bit data format. Although baud rate modulo divider settings do not always produce baud rates that exactly match standard rates, it is normally possible to get within a few percent, which is acceptable for reliable communications.

12.3.2 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 12-2](#).

The transmitter output (TxD) idle state defaults to logic high (TXINV = 0 following reset). The transmitter output is inverted by setting TXINV = 1. The transmitter is enabled by setting the TE bit in SCIC2. 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 (SCID).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, we will assume M = 0, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCID.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD high, waiting for more characters to transmit.

Writing 0 to TE does not immediately release the pin to be a general-purpose I/O pin. Any transmit activity that is in progress must first be completed. This includes data characters in progress, queued idle characters, and queued break characters.

12.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIC2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13 = 1. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake any sleeping receivers. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 0 and then write 1 to the TE bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while TE = 0, the SCI transmitter never actually releases control of the TxD pin. If there is a possibility of the shifter finishing while TE = 0, set the general-purpose I/O controls so the pin that is shared with TxD is an output driving a logic 1. This ensures that the TxD line will look like a normal idle line even if the SCI loses control of the port pin between writing 0 and then 1 to TE.

The length of the break character is affected by the BRK13 and M bits as shown below.

Table 12-8. Break Character Length

BRK13	M	Break Character Length
0	0	10 bit times
0	1	11 bit times
1	0	13 bit times
1	1	14 bit times

12.3.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 12-3) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV = 1. The receiver is enabled by setting the RE bit in SCIC2. 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 12.3.5.1, “8- and 9-Bit Data Modes.”](#) For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF)

status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full ($RDRF = 1$), it gets the data from the receive data register by reading SCID. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to [Section 12.3.4, "Interrupts and Status Flags"](#) for more details about flag clearing.

12.3.3.1 Data Sampling Technique

The SCI receiver uses a $16\times$ baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The $16\times$ baud rate clock is used to divide the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic level for that bit, the noise flag (NF) will be set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges, and if an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE is still set.

12.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wakeup (RWU) control bit in SCIC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant message

characters. At the end of a message, or at the beginning of the next message, all receivers automatically force RWU to 0 so all receivers wake in time to look at the first character(s) of the next message.

12.3.3.2.1 Idle-Line Wakeup

When WAKE = 0, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits).

When RWU is one and RWUID is zero, the idle condition that wakes up the receiver does not set the IDLE flag. The receiver wakes up and waits for the first data character of the next message which will set the RDRF flag and generate an interrupt if enabled. When RWUID is one, any idle condition sets the IDLE flag and generates an interrupt if enabled, regardless of whether RWU is zero or one.

The idle-line type (ILT) control bit selects one of two ways to detect an idle line. When ILT = 0, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle. When ILT = 1, the idle bit counter does not start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

12.3.3.2.2 Address-Mark Wakeup

When WAKE = 1, the receiver is configured for address-mark wakeup. In this mode, RWU is cleared automatically when the receiver detects a logic 1 in the most significant bit of a received character (eighth bit in M = 0 mode and ninth bit in M = 1 mode).

Address-mark wakeup allows messages to contain idle characters but requires that the MSB be reserved for use in address frames. The logic 1 MSB of an address frame clears the RWU bit before the stop bit is received and sets the RDRF flag. In this case the character with the MSB set is received even though the receiver was sleeping during most of this character time.

12.3.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF, IDLE, RXEDGIF and LBKDIF events, and a third vector is used for OR, NF, FE, and PF error conditions. Each of these ten interrupt sources can be separately masked by local interrupt enable masks. The flags can still be polled by software when the local masks are cleared to disable generation of hardware interrupt requests.

The SCI transmitter has two status flags that optionally can generate hardware interrupt requests. Transmit data register empty (TDRE) indicates when there is room in the transmit data buffer to write another transmit character to SCID. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt will be requested whenever TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TxD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (TCIE) bit is set, a hardware interrupt will be requested whenever TC = 1.

Instead of hardware interrupts, software polling may be used to monitor the TDRE and TC status flags if the corresponding TIE or TCIE local interrupt masks are 0s.

When a program detects that the receive data register is full ($RDRF = 1$), it gets the data from the receive data register by reading SCID. The RDRF flag is cleared by reading SCIS1 while $RDRF = 1$ and then reading SCID.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIS1 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 SCIS1 while $IDLE = 1$ and then reading SCID. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

If the associated error was detected in the received character that caused RDRF to be set, the error flags — noise flag (NF), framing error (FE), and parity error flag (PF) — get set at the same time as RDRF. These flags are not set in overrun cases.

If RDRF was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (OR) flag gets set instead the data along with any associated NF, FE, or PF condition is lost.

At any time, an active edge on the RxD serial data input pin causes the RXEDGIF flag to set. The RXEDGIF flag is cleared by writing a “1” to it. This function does depend on the receiver being enabled ($RE = 1$).

12.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

12.3.5.1 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 SCIC1. 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 SCIC3. For the receiver, the ninth bit is held in R8 in SCIC3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCID.

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 SCID to the shifter.

9-bit data mode typically is used in conjunction with parity to allow eight bits of data plus the parity in the ninth bit. Or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. In custom protocols, the ninth bit can also serve as a software-controlled marker.

12.3.5.2 Stop Mode Operation

During all stop modes, clocks to the SCI module are halted.

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes. No SCI module registers are affected in stop3 mode.

The receive input active edge detect circuit is still active in stop3 mode, but not in stop2.. An active edge on the receive input brings the CPU out of stop3 mode if the interrupt is not masked (RXEDGIE = 1).

Note, because the clocks are halted, the SCI module will resume operation upon exit from stop (only in stop3 mode). Software should ensure stop mode is not entered while there is a character being transmitted out of or received into the SCI module.

12.3.5.3 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.

12.3.5.4 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 SCIC3 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 13

16-Bit Serial Peripheral Interface (S08SPI16V1)

13.1 Introduction

The 8- or 16-bit selectable serial peripheral interface (SPI) module provides for full-duplex, synchronous, serial communication between the MCU and peripheral devices. These peripheral devices can include other microcontrollers, analog-to-digital converters, shift registers, sensors, memories, etc.

The SPI runs at a baud rate up to the bus clock divided by two in master mode and up to the bus clock divided by four in slave mode. Software can poll the status flags, or SPI operation can be interrupt driven.

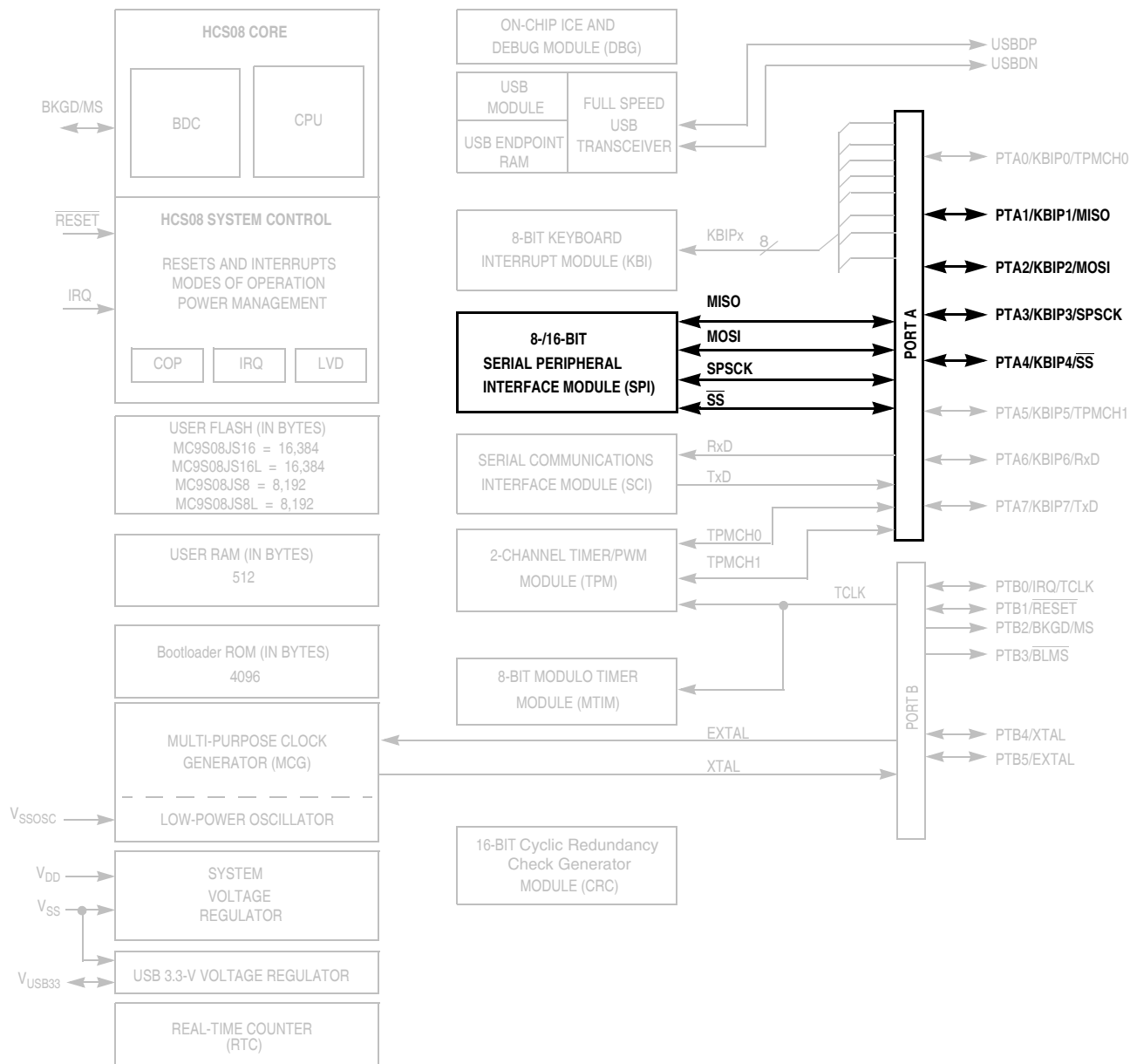
The SPI also supports a data length of 8 or 16 bits and provides a hardware match feature for the receive data buffer.

The MC9S08JS16 series have one serial peripheral interface module (SPI). The four pins associated with SPI functionality are shared with PTA[4:1] See “MC9S08JS16 Series *Data Sheet*,” for SPI electrical parametric information.

13.1.1 SPI Port Configuration Information

To configure the SPI for high speed operation (operating frequency of greater than 6 MHz), the input filters on the SPI port pins must be disabled by clearing the SPIFE in SOPT2. Doing so allows data transfers to occur at the higher frequency but at a risk of introducing noise during data transfers. If running the SPI at a baud rate of 8 MHz or greater, the application must enable the high output drive strength selection on the port pins corresponding to the affected SPI port pins.

By default, the input filters on the SPI port pins will be enabled (SPIFE=1), which restricts the SPI data rate to 6 MHz, but protects the SPI from noise during data transfers.



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1).
Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD} . IRQ must not be driven above V_{DD} .
4. \overline{RESET} contains integrated pullup device if PTB1 enabled as reset pin function (RSTPE = 1)
5. Pin contains integrated pullup device.
6. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 13-1. MC9S08JS16 Series Block Diagram Highlighting the SPI Module and Pins

Module Initialization (Slave):															
Write:	SPIC1	to configure	interrupts, set primary SPI options, slave mode select, and system enable.												
Write:	SPIC2	to configure	optional SPI features, hardware match interrupt enable, and 8- or 16-bit data transmission length												
Write:	SPIMH:SPIML	to set	hardware compare value that triggers SPMF (optional) when value in receive data buffer equals this value.												
Module Initialization (Master):															
Write:	SPIC1	to configure	interrupts, set primary SPI options, master mode select, and system enable.												
Write:	SPIC2	to configure	optional SPI features, hardware match interrupt enable, and 8- or 16-bit data transmission length												
Write:	SPIBR	to set	baud rate												
Write:	SPIMH:SPIML	to set	hardware compare value that triggers SPMF (optional) when value in receive data buffer equals this value.												
Module Use:															
After SPI master initiates transfer by checking that SPTEF = 1 and then writing data to SPIDH/L:															
Wait for SPRF, then read from SPIDH/L															
Wait for SPTEF, then write to SPIDH/L															
Data transmissions can be 8- or 16-bits long, and mode fault detection can be enabled for master mode in cases where more than one SPI device might become a master at the same time. Also, some applications may utilize the receive data buffer hardware match feature to trigger specific actions, such as when command data can be sent through the SPI or to indicate the end of an SPI transmission.															
SPIC1	<table border="1"> <tr> <td>SPIE</td> <td>SPE</td> <td>SPTIE</td> <td>MSTR</td> <td>CPOL</td> <td>CPHA</td> <td>SSOE</td> <td>LSBFE</td> </tr> </table> Module/interrupt enables and configuration							SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE								
SPIC2	<table border="1"> <tr> <td>SPMIE</td> <td>SPIMODE</td> <td></td> <td>MODFEN</td> <td>BIDIROE</td> <td></td> <td>SPISWAI</td> <td>SPC0</td> </tr> </table> Additional configuration options.							SPMIE	SPIMODE		MODFEN	BIDIROE		SPISWAI	SPC0
SPMIE	SPIMODE		MODFEN	BIDIROE		SPISWAI	SPC0								
SPIBR	<table border="1"> <tr> <td></td> <td>SPPR2</td> <td>SPPR1</td> <td>SPPR0</td> <td></td> <td>SPR2</td> <td>SPR1</td> <td>SPR0</td> </tr> </table> Baud rate = (BUSCLK/SPPR[2:0])/SPR2[2:0]								SPPR2	SPPR1	SPPR0		SPR2	SPR1	SPR0
	SPPR2	SPPR1	SPPR0		SPR2	SPR1	SPR0								
SPIDH	<table border="1"> <tr> <td>Bit 15</td> <td>Bit 14</td> <td>Bit 13</td> <td>Bit 12</td> <td>Bit 11</td> <td>Bit 10</td> <td>Bit 9</td> <td>Bit 8</td> </tr> </table>							Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8								
SPIDL	<table border="1"> <tr> <td>Bit 7</td> <td>Bit 6</td> <td>Bit 5</td> <td>Bit 4</td> <td>Bit 3</td> <td>Bit 2</td> <td>Bit 1</td> <td>Bit 0</td> </tr> </table>							Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0								
SPIMH	<table border="1"> <tr> <td>Bit 15</td> <td>Bit 14</td> <td>Bit 13</td> <td>Bit 12</td> <td>Bit 11</td> <td>Bit 10</td> <td>Bit 9</td> <td>Bit 8</td> </tr> </table>							Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8								
SPIML	<table border="1"> <tr> <td>Bit 7</td> <td>Bit 6</td> <td>Bit 5</td> <td>Bit 4</td> <td>Bit 3</td> <td>Bit 2</td> <td>Bit 1</td> <td>Bit 0</td> </tr> </table> Hardware Match Value							Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0								
SPIS	<table border="1"> <tr> <td>SPRF</td> <td>SPMF</td> <td>SPTEF</td> <td>MODF</td> <td></td> <td></td> <td></td> <td></td> </tr> </table>							SPRF	SPMF	SPTEF	MODF				
SPRF	SPMF	SPTEF	MODF												

Figure 13-2. SPI Module Quick Start

13.1.2 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

13.1.3 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 SPIC2 register. In wait mode, if the SPISWAI bit is clear, the SPI operates like in Run Mode. If the SPISWAI bit is set, the SPI goes into a power conservative state, with the SPI clock generation turned off. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU goes into Run Mode. If the SPI is configured as a slave, reception and transmission of a byte continues, so that the slave stays synchronized to the master.
- Stop Mode

The SPI is inactive in stop3 mode for reduced power consumption. If the SPI is configured as a master, any transmission in progress stops, but is resumed after the CPU goes into Run Mode. If the SPI is configured as a slave, reception and transmission of a data continues, so that the slave stays synchronized to the master.

The SPI is completely disabled in all other stop modes. When the CPU wakes from these stop modes, all SPI register content will be reset.

This is a high level description only, detailed descriptions of operating modes are contained in [Section 13.4.9, “Low Power Mode Options.”](#)

13.1.4 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.

13.1.4.1 SPI System Block Diagram

Figure 13-3 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input (\overline{SS} pin). In this system, the master device has configured its \overline{SS} pin as an optional slave select output.

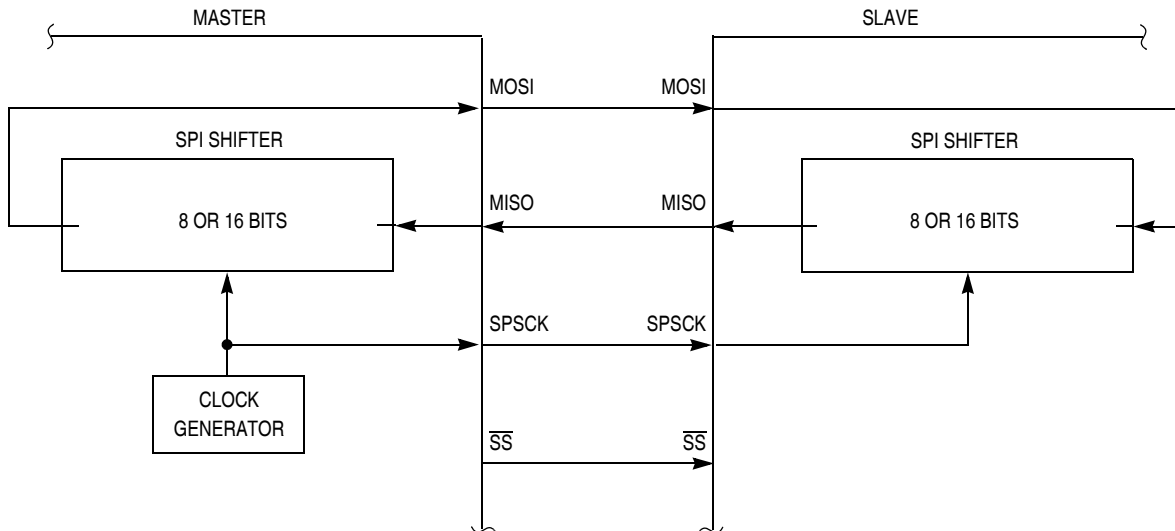


Figure 13-3. SPI System Connections

13.1.4.2 SPI Module Block Diagram

Figure 13-4 is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPIDH:SPIDL) 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 SPIDH:SPIDL). 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.

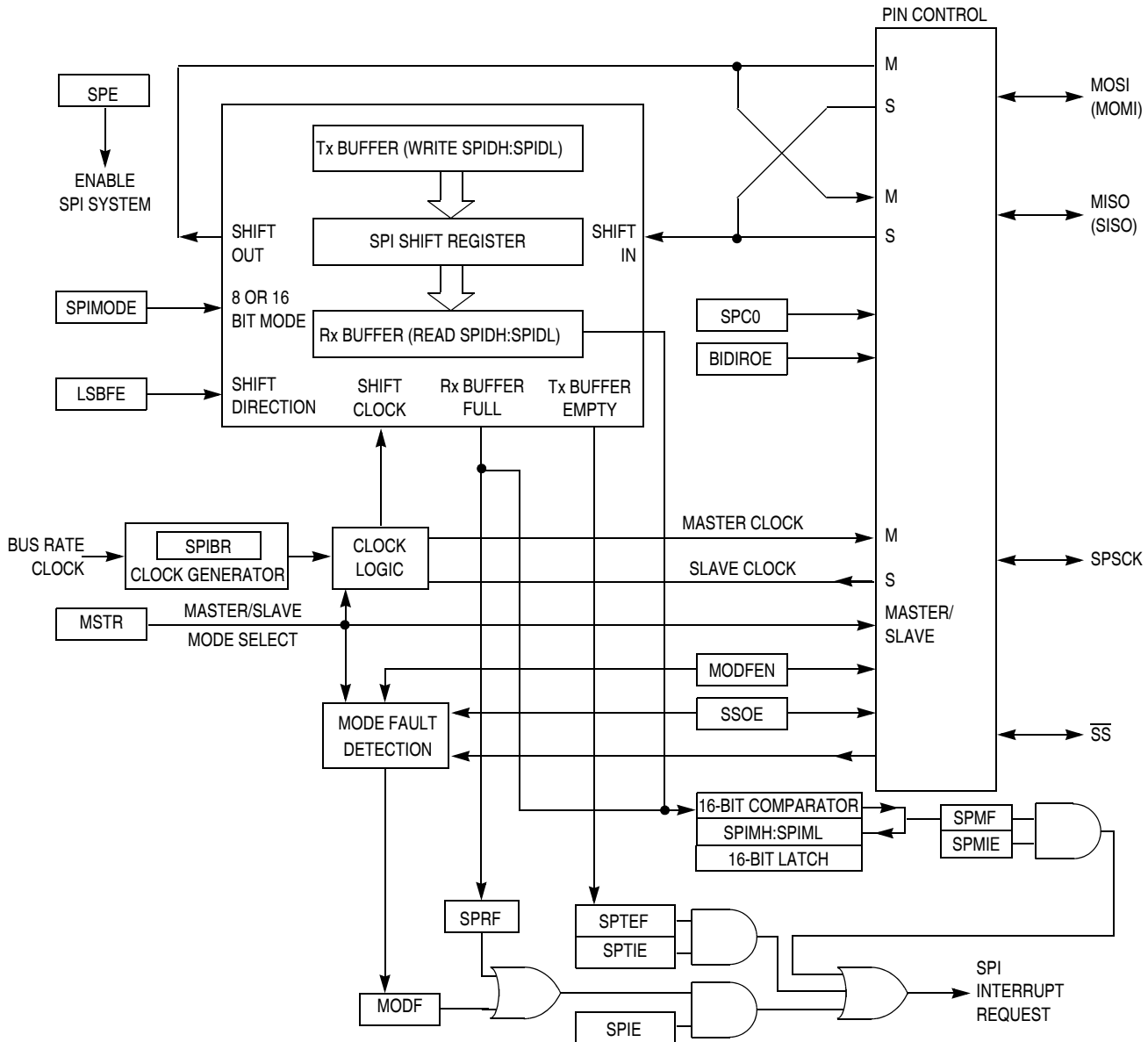


Figure 13-4. SPI Module Block Diagram

13.2 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.

13.2.1 SPCK — 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.

13.2.2 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.

13.2.3 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.

13.2.4 \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).

13.3 Register Definition

The SPI has eight 8-bit registers to select SPI options, control baud rate, report SPI status, hold an SPI data match value, and for transmit/receive data.

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

13.3.1 SPI Control Register 1 (SPIC1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

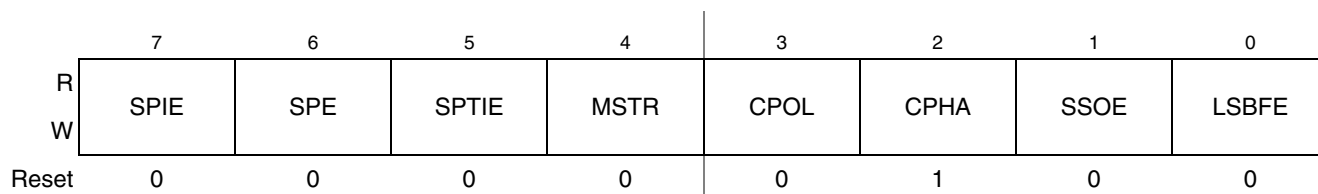


Figure 13-5. SPI Control Register 1 (SPIC1)

Table 13-1. SPIC1 Field Descriptions

Field	Description
7 SPIE	SPI Interrupt Enable (for SPRF and MODF) — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF inhibited (use polling) 1 When SPRF or MODF is 1, request a hardware interrupt
6 SPE	SPI System Enable — This bit enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, SPI is disabled and forced into idle state, and all status bits in the SPIS register are reset. 0 SPI system inactive 1 SPI system enabled
5 SPTIE	SPI Transmit Interrupt Enable — This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). 0 Interrupts from SPTEF inhibited (use polling) 1 When SPTEF is 1, hardware interrupt requested
4 MSTR	Master/Slave Mode Select — This bit selects master or slave mode operation. 0 SPI module configured as a slave SPI device 1 SPI module configured as a master SPI device
3 CPOL	Clock Polarity — This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to Section 13.4.5, “SPI Clock Formats” for more details. 0 Active-high SPI clock (idles low) 1 Active-low SPI clock (idles high)
2 CPHA	Clock Phase — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to Section 13.4.5, “SPI Clock Formats” for more details. 0 First edge on SPSCCK occurs at the middle of the first cycle of a data transfer 1 First edge on SPSCCK 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 SPIC2 and the master/slave (MSTR) control bit to determine the function of the \overline{SS} pin as shown in Table 13-2 .
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 13-2. \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

13.3.2 SPI Control Register 2 (SPIC2)

This read/write register is used to control optional features of the SPI system. Bits 6 and 5 are not implemented and always read 0.

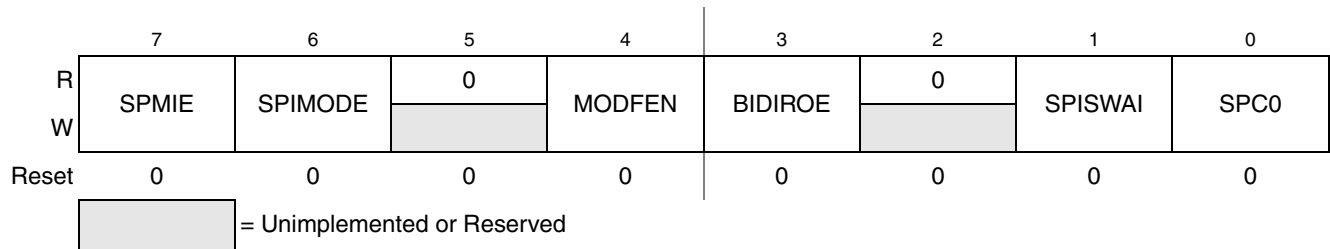


Figure 13-6. SPI Control Register 2 (SPIC2)

Table 13-3. SPIC2 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 SPIS register. Refer to section Section 13.4.4, “Data Transmission Length,” for details. 0 8-bit SPI shift register, match register, and buffers. 1 16-bit SPI shift register, match register, and buffers.
4 MODFEN	Master Mode-Fault Function Enable — When the SPI is configured for slave mode, this bit has no meaning or effect. (The \overline{SS} pin is the slave select input.) In master mode, this bit determines how the \overline{SS} pin is used (refer to Table 13-2 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 13-4 . 0 SPI uses separate pins for data input and data output. 1 SPI configured for single-wire bidirectional operation.

Table 13-4. Bidirectional Pin Configurations

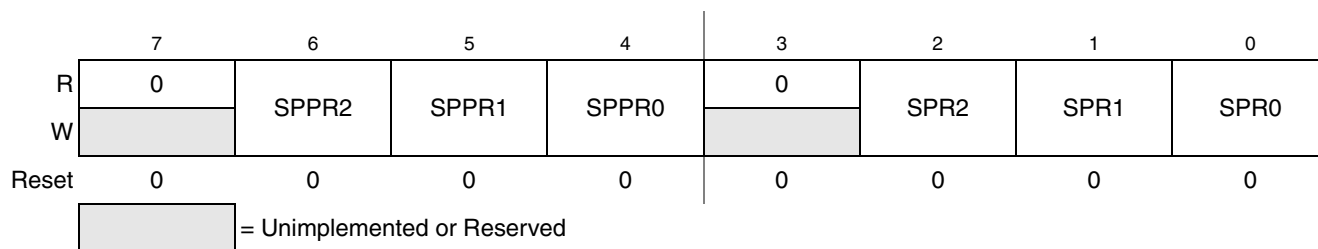
Pin Mode	SPC0	BIDIROE	MISO	MOSI
Master Mode of Operation				

Table 13-4. Bidirectional Pin Configurations

Pin Mode	SPC0	BIDIROE	MISO	MOSI
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	SlaveIn
Bidirectional	1	0	Slave In	MOSI not used by SPI
		1	Slave I/O	

13.3.3 SPI Baud Rate Register (SPIBR)

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.


Figure 13-7. SPI Baud Rate Register (SPIBR)
Table 13-5. SPIBR 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 13-6 . 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 13-15). See Section 13.4.6, “SPI Baud Rate Generation,” for details.
2:0 SPR[2:0]	SPI Baud Rate Divisor — This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown in Table 13-7 . The input to this divider comes from the SPI baud rate prescaler (see Figure 13-15). See Section 13.4.6, “SPI Baud Rate Generation,” for details.

Table 13-6. SPI Baud Rate Prescaler Divisor

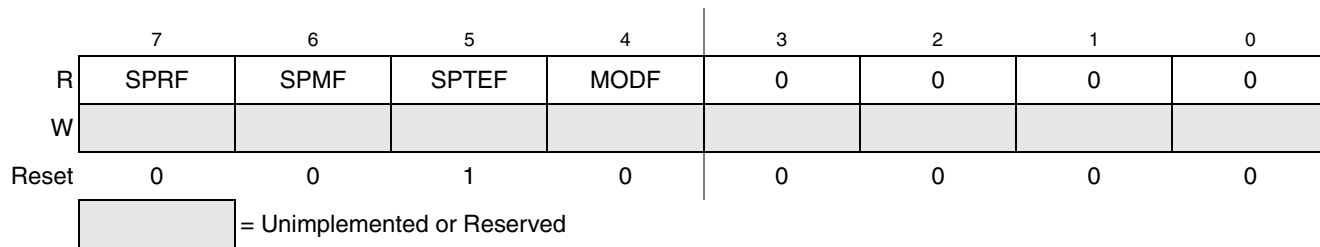
SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

Table 13-7. SPI Baud Rate Divisor

SPR2:SPR1:SPR0	Rate Divisor
0:0:0	2
0:0:1	4
0:1:0	8
0:1:1	16
1:0:0	32
1:0:1	64
1:1:0	128
1:1:1	256

13.3.4 SPI Status Register (SPIS)

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.


Figure 13-8. SPI Status Register (SPIS)
Table 13-8. SPIS 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 (SPIDH:SPIDL). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register. 0 No data available in the receive data buffer. 1 Data available in the receive data buffer.
6 SPMF	SPI Match Flag — SPMF is set after SPRF = 1 when the value in the receive data buffer matches the value in SPIMH:SPIML. To clear the flag, read SPMF when it is set, then write a 1 to it. 0 Value in the receive data buffer does not match the value in SPIMH:SPIML registers. 1 Value in the receive data buffer matches the value in SPIMH:SPIML registers.

Table 13-8. SPIS Register Field Descriptions

Field	Description
5 SPTEF	<p>SPI Transmit Buffer Empty Flag — This bit is set when the transmit data buffer is empty. It is cleared by reading SPIS with SPTEF set, followed by writing a data value to the transmit buffer at SPIDH:SPIDL. SPIS must be read with SPTEF = 1 before writing data to SPIDH:SPIDL or the SPIDH:SPIDL 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 SPIDH:SPIDL 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.</p> <p>0 SPI transmit buffer not empty 1 SPI transmit buffer empty</p>
4 MODF	<p>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 (SPIC1).</p> <p>0 No mode fault error 1 Mode fault error detected</p>

13.3.5 SPI Data Registers (SPIDH:SPIDL)

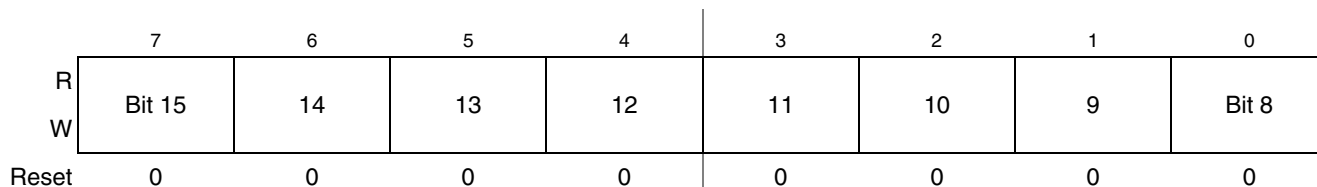


Figure 13-9. SPI Data Register High (SPIDH)

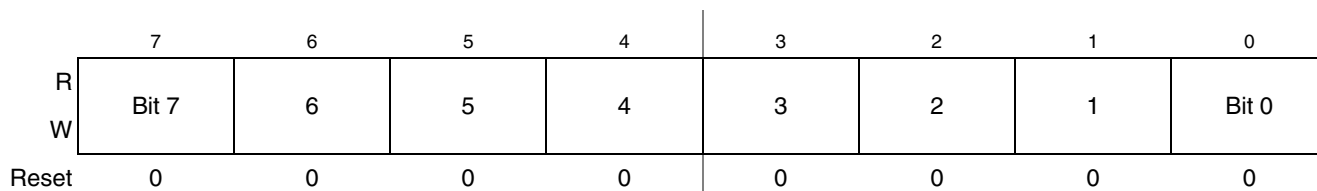


Figure 13-10. SPI Data Register Low (SPIDL)

The SPI data registers (SPIDH:SPIDL) 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 SPIS register indicates when the transmit data buffer is ready to accept new data. SPIS must be read when SPTEF is set before writing to the SPI data registers, or the write will be ignored.

Data may be read from SPIDH:SPIDL 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 SPIDL is available. Reads of SPIDH will return all 0s. Writes to SPIDH will be ignored.

In 16-bit mode, reading either byte (SPIDH or SPIDL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIDH or SPIDL) 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.

13.3.6 SPI Match Registers (SPIMH:SPIML)

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 SPIMH:SPIML registers.

In 8-bit mode, only SPIML is available. Reads of SPIMH will return all 0s. Writes to SPIMH will be ignored.

In 16-bit mode, reading either byte (SPIMH or SPIML) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIMH or SPIML) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent value into the SPI match registers.

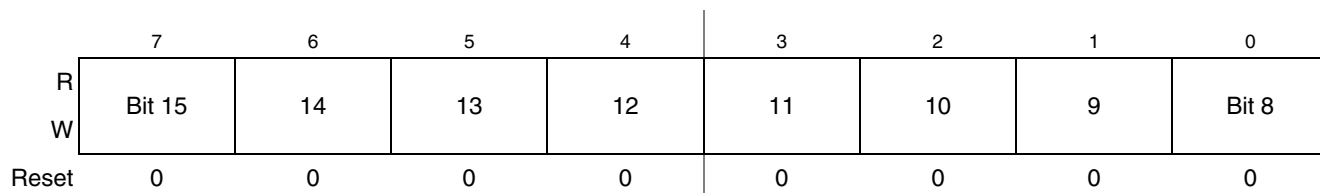


Figure 13-11. SPI Match Register High (SPIMH)

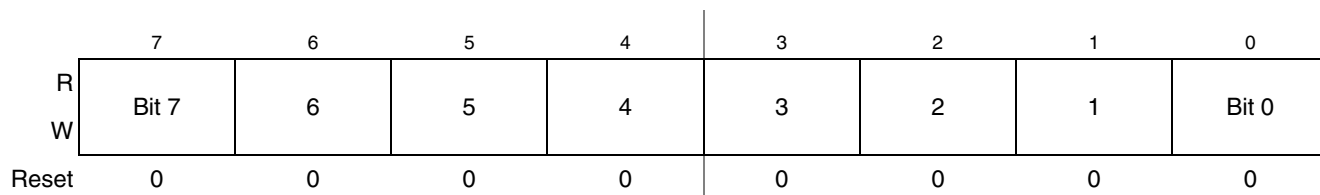


Figure 13-12. SPI Match Register Low (SPIML)

13.4 Functional Description

13.4.1 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 (SPIS) when $SPTEF = 1$ and then writing data to the transmit data buffer (write to SPIDH:SPIDL). When a transfer is complete, received data is moved into the receive data buffer. The SPIDH:SPIDL 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 (SPIC1) 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.

13.4.2 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 SPIS register while $SPTEF = 1$ and writing to the master SPI data registers. If the shift register is empty, the byte immediately transfers to the shift register. The data begins shifting out on the MOSI pin under the control of the serial clock.

- SPSCK

The SPR2, SPR1, and SPR0 baud rate selection bits in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI Baud Rate register control the baud rate generator and determine the speed of the transmission. The SPSCK pin is the SPI clock output. Through the SPSCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.

- MOSI, MISO pin

In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.

- \overline{SS} pin

If MODFEN and SSOE bit are set, the \overline{SS} pin is configured as slave select output. The \overline{SS} output becomes low during each transmission and is high when the SPI is in idle state.

If MODFEN is set and SSOE is cleared, the \overline{SS} pin is configured as input for detecting mode fault error. If the \overline{SS} input becomes low this indicates a mode fault error where another master tries to drive the MOSI and SPSCK lines. In this case, the SPI immediately switches to slave mode, by clearing the MSTR bit and also disables the slave output buffer MISO (or SISO in bidirectional mode). So the result is that all outputs are disabled and SPSCK, MOSI and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state.

This mode fault error also sets the mode fault (MODF) flag in the SPI Status Register (SPIS). 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 SPSCCK-cycle delay. After the delay, SPSCCK 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 13.4.5, “SPI Clock Formats”](#)).

NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, BIDIROE with SPC0 set, SPIMODE, SPPR2-SPPR0 and SPR2-SPR0 in master mode will abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master has to ensure that the remote slave is set back to idle state.

13.4.3 Slave Mode

The SPI operates in slave mode when the MSTR bit in SPI Control Register1 is clear.

- SPSCCK

In slave mode, SPSCCK 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 SPSCCK 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 SPSCCK 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 SPSCCK input cause the data at the serial data input pin to be latched. Odd numbered edges cause the value previously latched from the serial data input pin to shift into the LSB or MSB of the SPI shift register, depending on the LSBFE bit.

When CPHA is set, the first edge is used to get the first data bit onto the serial data output pin. When CPHA is clear and the \overline{SS} input is low (slave selected), the first bit of the SPI data is driven out of the serial data output pin. After the eighth (SPIMODE = 0) or sixteenth (SPIMODE = 1) shift, the transfer is considered complete and the received data is transferred into the SPI data registers. To indicate transfer is complete, the SPRF flag in the SPI Status Register is set.

NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0 and BIDIROE with SPC0 set and SPIMODE in slave mode will corrupt a transmission in progress and has to be avoided.

13.4.4 Data Transmission Length

The SPI can support data lengths of 8 or 16 bits. The length can be configured with the SPIMODE bit in the SPIC2 register.

In 8-bit mode (SPIMODE = 0), the SPI Data Register is comprised of one byte: SPIDL. The SPI Match Register is also comprised of only one byte: SPIML. Reads of SPIDH and SPIMH will return zero. Writes to SPIDH and SPIMH will be ignored.

In 16-bit mode (SPIMODE = 1), the SPI Data Register is comprised of two bytes: SPIDH and SPIDL. Reading either byte (SPIDH or SPIDL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIDH or SPIDL) 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: SPIMH and SPIML. Reading either byte (SPIMH or SPIML) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. Writing to either byte (SPIMH or SPIML) 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 SPIS register. To initiate a transfer after writing to SPIMODE, the SPIS register must be read with SPTEF = 1, and data must be written to SPIDH:SPIDL in 16-bit mode (SPIMODE = 1) or SPIDL in 8-bit mode (SPIMODE = 0).

In slave mode, user software must 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.

13.4.5 SPI Clock Formats

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the SPI system has a clock polarity (CPOL) bit and a clock phase (CPHA) control bit to select one of four clock formats for data transfers. CPOL selectively inserts an inverter in series with the clock. CPHA chooses between two different clock phase relationships between the clock and data.

Figure 13-13 shows the clock formats when SPIMODE = 0 (8-bit mode) and CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCCK edge and bit 8 ending one-half SPSCCK cycle after the sixteenth SPSCCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCCK 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 SPSCCK 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.

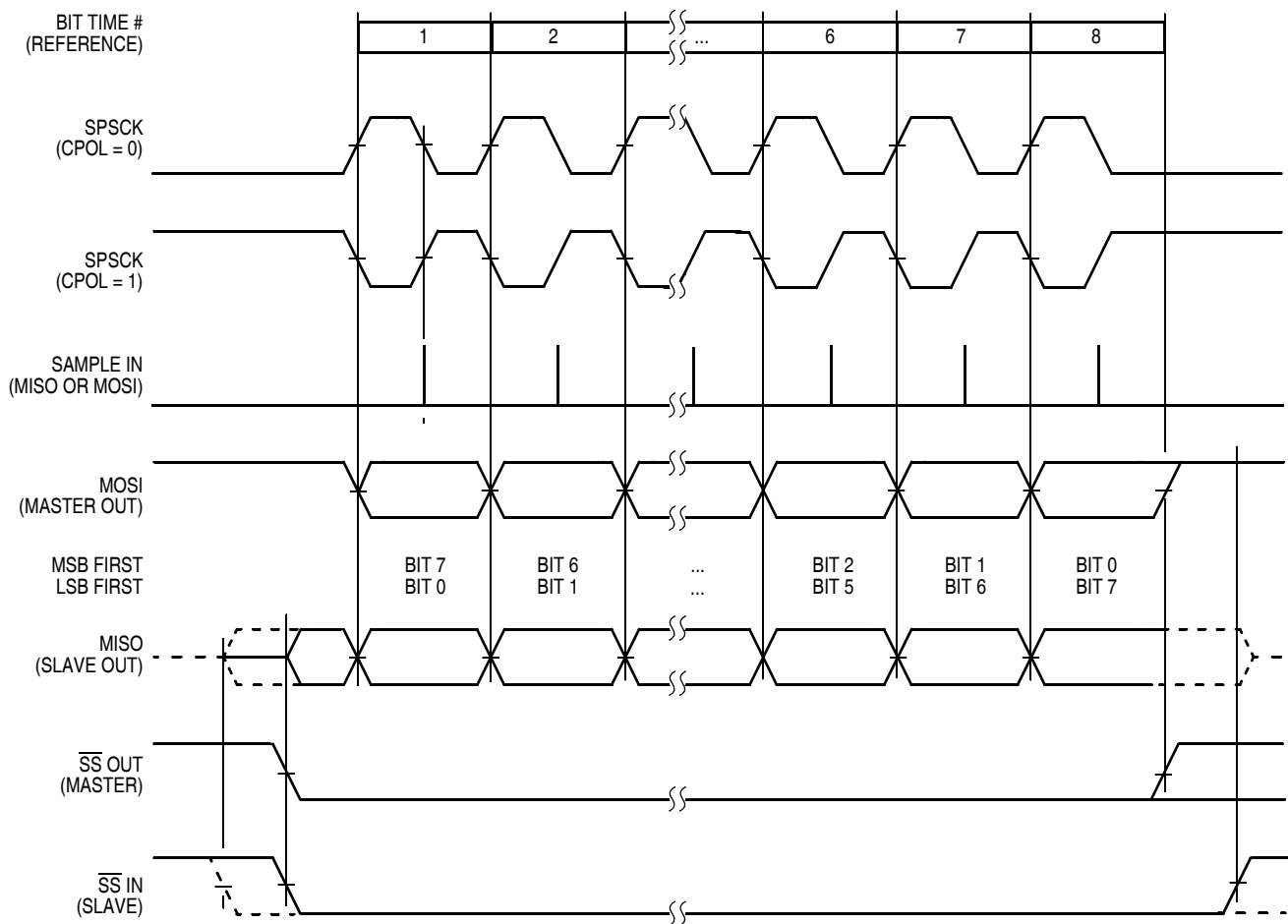


Figure 13-13. SPI Clock Formats (CPHA = 1)

When $CPHA = 1$, the slave begins to drive its MISO output when \overline{SS} goes to active low, but the data is not defined until the first SPSCCK edge. The first SPSCCK 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 SPSCCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCCK 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 13-14 shows the clock formats when $SPIMODE = 0$ and $CPHA = 0$. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected (\overline{SS} IN goes low), and bit 8 ends at the last SPSCCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in $LSBFE$. Both variations of SPSCCK 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 SPSCCK 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.

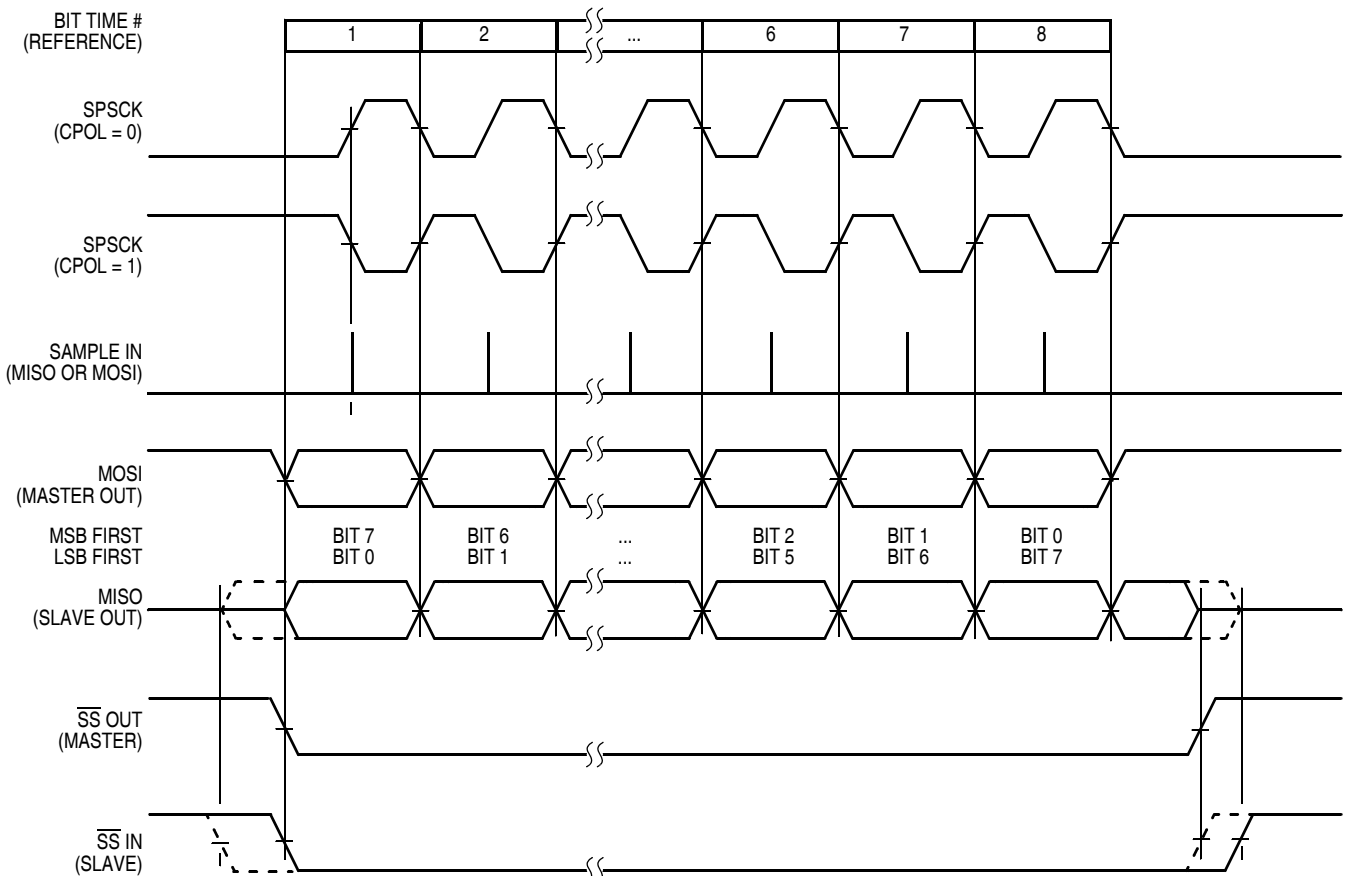


Figure 13-14. SPI Clock Formats ($CPHA = 0$)

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when \overline{SS} goes to active low. The first SPSCCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCCK 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.

13.4.6 SPI Baud Rate Generation

As shown in [Figure 13-15](#), the clock source for the SPI baud rate generator is the bus clock. The three prescale bits (SPPR2:SPPR1:SPPR0) choose a prescale divisor of 1, 2, 3, 4, 5, 6, 7, or 8. The three rate select bits (SPR2:SPR1:SPR0) divide the output of the prescaler stage by 2, 4, 8, 16, 32, 64, 128, or 256 to get the internal SPI master mode bit-rate clock.

The baud rate generator is activated only when the SPI is in the master mode and a serial transfer is taking place. In the other cases, the divider is disabled to decrease I_{DD} current.

The baud rate divisor equation is as follows:

$$\text{BaudRateDivisor} = (\text{SPPR} + 1) \cdot 2^{(\text{SPR} + 1)}$$

The baud rate can be calculated with the following equation:

$$\text{Baud Rate} = \text{BusClock} / \text{BaudRateDivisor}$$

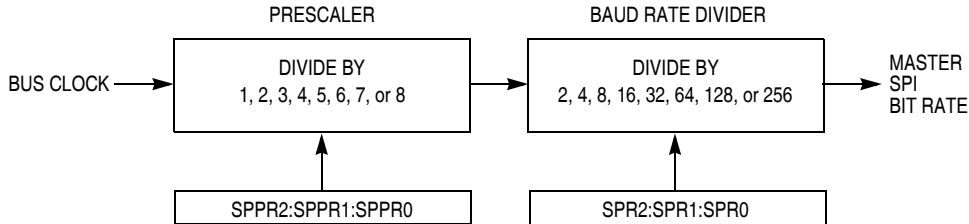


Figure 13-15. SPI Baud Rate Generation

13.4.7 Special Features

13.4.7.1 \overline{SS} Output

The \overline{SS} output feature automatically drives the \overline{SS} pin low during transmission to select external devices and drives it high during idle to deselect external devices. When \overline{SS} output is selected, the \overline{SS} output pin is connected to the \overline{SS} input pin of the external device.

The \overline{SS} output is available only in master mode during normal SPI operation by asserting the SSOE and MODFEN bits as shown in [Table 13-2](#).

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.

13.4.7.2 Bidirectional Mode (MOMI or SISO)

The bidirectional mode is selected when the SPC0 bit is set in SPI Control Register 2 (see Section Table 13-9., “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 13-9. Normal Mode and Bidirectional Mode

When SPE = 1	Master Mode MSTR = 1	Slave Mode MSTR = 0
<p>Normal Mode SPC0 = 0</p>		
<p>Bidirectional Mode SPC0 = 1</p>		

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 SPSCCK 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 SPSCCK 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.

13.4.8 Error Conditions

The SPI has one error condition:

- Mode fault error

13.4.8.1 Mode Fault Error

If the \overline{SS} input becomes low while the SPI is configured as a master, it indicates a system error where more than one master may be trying to drive the MOSI and SPSCK lines simultaneously. This condition is not permitted in normal operation, and the MODF bit in the SPI status register is set automatically provided the MODFEN bit is set.

In the special case where the SPI is in master mode and MODFEN bit is cleared, the \overline{SS} pin is not used by the SPI. In this special case, the mode fault error function is inhibited and MODF remains cleared. In case the SPI system is configured as a slave, the \overline{SS} pin is a dedicated input pin. Mode fault error doesn't occur in slave mode.

If a mode fault error occurs the SPI is switched to slave mode, with the exception that the slave output buffer is disabled. So SPSCK, MISO and MOSI pins are forced to be high impedance inputs to avoid any possibility of conflict with another output driver. A transmission in progress is aborted and the SPI is forced into idle state.

If the mode fault error occurs in the bidirectional mode for a SPI system configured in master mode, output enable of the MOMI (MOSI in bidirectional mode) is cleared if it was set. No mode fault error occurs in the bidirectional mode for the SPI system configured in slave mode.

The mode fault flag is cleared automatically by a read of the SPI Status Register (with MODF set) followed by a write to SPI Control Register 1. If the mode fault flag is cleared, the SPI becomes a normal master or slave again.

13.4.9 Low Power Mode Options

13.4.9.1 SPI in Run Mode

In run mode with the SPI system enable (SPE) bit in the SPI control register clear, the SPI system is in a low-power, disabled state. SPI registers can still be accessed, but clocks to the core of this module are disabled.

13.4.9.2 SPI in Wait Mode

SPI operation in wait mode depends upon the state of the SPISWAI bit in SPI Control Register 2.

- If SPISWAI is clear, the SPI operates normally when the CPU is in wait mode
- If SPISWAI is set, SPI clock generation ceases and the SPI module enters a power conservation state when the CPU is in wait mode.
 - If SPISWAI is set and the SPI is configured for master, any transmission and reception in progress stops at wait mode entry. The transmission and reception resumes when the SPI exits wait mode.
 - If SPISWAI is set and the SPI is configured as a slave, any transmission and reception in progress continues if the SPSCCK continues to be driven from the master. This keeps the slave synchronized to the master and the SPSCCK.

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 SPIDH:SPIDL 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 SPIDH:SPIDL registers until after the slave SPI has exited wait or stop mode. A SPRF flag and SPIDH:SPIDL copy is only generated if wait mode is entered or exited during a transmission. If the slave enters wait mode in idle mode and exits wait mode in idle mode, neither a SPRF nor a SPIDH:SPIDL copy will occur.

13.4.9.3 SPI in Stop Mode

Stop3 mode is dependent on the SPI system. Upon entry to stop3 mode, the SPI module clock is disabled (held high or low). If the SPI is in master mode and exchanging data when the CPU enters stop mode, the transmission is frozen until the CPU exits stop mode. After stop, data to and from the external SPI is exchanged correctly. In slave mode, the SPI will stay synchronized with the master.

The stop mode is not dependent on the SPISWAI bit.

In all other stop modes, the SPI module is completely disabled. After stop, all registers are reset to their default values, and the SPI module must be re-initialized.

13.4.9.4 Reset

The reset values of registers and signals are described in [Section 13.3, “Register Definition.”](#) which details the registers and their bit-fields.

- If a data transmission occurs in slave mode after reset without a write to SPIDH:SPIDL, it will transmit garbage, or the data last received from the master before the reset.
- Reading from the SPIDH:SPIDL after reset will always read zeros.

13.4.9.5 Interrupts

The SPI only originates interrupt requests when the SPI is enabled (SPE bit in SPIC1 set). The following is a description of how the SPI makes a request and how the MCU must acknowledge that request. The interrupt vector offset and interrupt priority are chip dependent.

13.4.10 SPI Interrupts

There are four flag bits, three interrupt mask bits, and one interrupt vector associated with the SPI system. The SPI interrupt enable mask (SPIE) enables interrupts from the SPI receiver full flag (SPRF) and mode fault flag (MODF). The SPI transmit interrupt enable mask (SPTIE) enables interrupts from the SPI transmit buffer empty flag (SPTEF). The SPI match interrupt enable mask bit (SPIMIE) enables interrupts from the SPI match flag (SPMF). When one of the flag bits is set, and the associated interrupt mask bit is set, a hardware interrupt request is sent to the CPU. If the interrupt mask bits are cleared, software can poll the associated flag bits instead of using interrupts. The SPI interrupt service routine (ISR) must check the flag bits to determine what event caused the interrupt. The service routine must also clear the flag bit(s) before returning from the ISR (usually near the beginning of the ISR).

13.4.10.1 MODF

MODF occurs when the master detects an error on the \overline{SS} pin. The master SPI must be configured for the MODF feature (see [Table 13-2](#)). Once MODF is set, the current transfer is aborted and the following bit is changed:

- MSTR=0, The master bit in SPIC1 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 13.3.4, “SPI Status Register \(SPIS\).”](#)

13.4.10.2 SPRF

SPRF occurs when new data has been received and copied to the SPI receive data buffer. In 8-bit mode, SPRF is set only after all 8 bits have been shifted out of the shift register and into SPIDL. In 16-bit mode, SPRF is set only after all 16 bits have been shifted out of the shift register and into SPIDH:SPIDL.

Once SPRF is set, it does not clear until it is serviced. SPRF has an automatic clearing process which is described in [Section 13.3.4, “SPI Status Register \(SPIS\).”](#) 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 SPIDH:SPIDL.

13.4.10.3 SPTEF

SPTEF occurs when the SPI transmit buffer is ready to accept new data. In 8-bit mode, SPTEF is set only after all 8 bits have been moved from SPIDL into the shifter. In 16-bit mode, SPTEF is set only after all 16 bits have been moved from SPIDH:SPIDL 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 13.3.4, “SPI Status Register \(SPIS\)”](#).

13.4.10.4 SPMF

SPMF occurs when the data in the receive data buffer is equal to the data in the SPI match register. In 8-bit mode, SPMF is set only after bits 8–0 in the receive data buffer are determined to be equivalent to the value in SPIML. 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 SPIMH:SPIML.

13.5 Initialization/Application Information

13.5.1 SPI Module Initialization Example

13.5.1.1 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 (SPIC1) 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 (SPIC2) 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 (SPIBR) to set the prescaler and bit rate divisor for an SPI master.
4. Update the hardware match register (SPIMH:SPIML) 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 SPIS while SPTEF = 1, and then write to the transmit data register (SPIDH:SPIDL) to begin transfer.

13.5.1.2 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 SPSCCK occurs at the start of the first cycle of a data transfer.

Table 13-10.
SPIC1=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 SPSCCK 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

Table 13-11.
SPIC2 = 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

Table 13-12.
SPIBR = 0x00(%00000000)

Bit 7		= 0	Unimplemented
Bit 6:4		= 000	Sets prescale divisor to 1
Bit 3		= 0	Unimplemented
Bit 2:0		= 000	Sets baud rate divisor to 2

Table 13-13.
SPIS = 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

Table 13-13.

SPIS = 0x00(%00000000)

Bit 4	MODF	= 0	Mode fault flag for master mode
Bit 3:0		= 0	Unimplemented

Table 13-14.

SPIMH = 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.

Table 13-15.

SPIML = 0xXX

Holds bits 0–7 of the hardware match buffer.

Table 13-16.

SPIDH = 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.

Table 13-17.

SPIDL = 0xxx

Holds bits 0–7 of the data to be transmitted by the transmit buffer and received by the receive buffer.

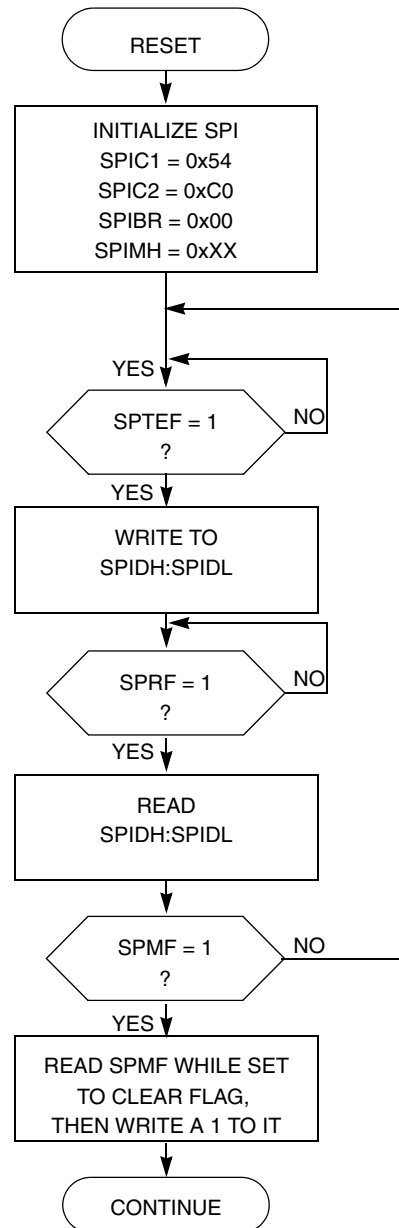


Figure 13-16. Initialization Flowchart Example for SPI Master Device in 16-bit Mode

Chapter 14

Timer/Pulse-Width Modulator (S08TPMV3)

14.1 Introduction

The MC9S08JS16 series include one independent timer/PWM (TPM) module that supports traditional input capture, output compare, or buffered edge-aligned pulse-width modulation (PWM) on each channel. A control bit in TPM configures all channels in that timer to operate as center-aligned PWM functions. In TPM, timing functions are based on a separate 16-bit counter with prescaler and modulo features to control frequency and range (period between overflows) of the time reference.

The use of the fixed system clock, XCLK, as the clock source for the TPM module allows the TPM prescaler to run at a frequency no greater than MCGOUT divided by 4 (see [Section 9.4.7, “Fixed Frequency Clock”](#)).

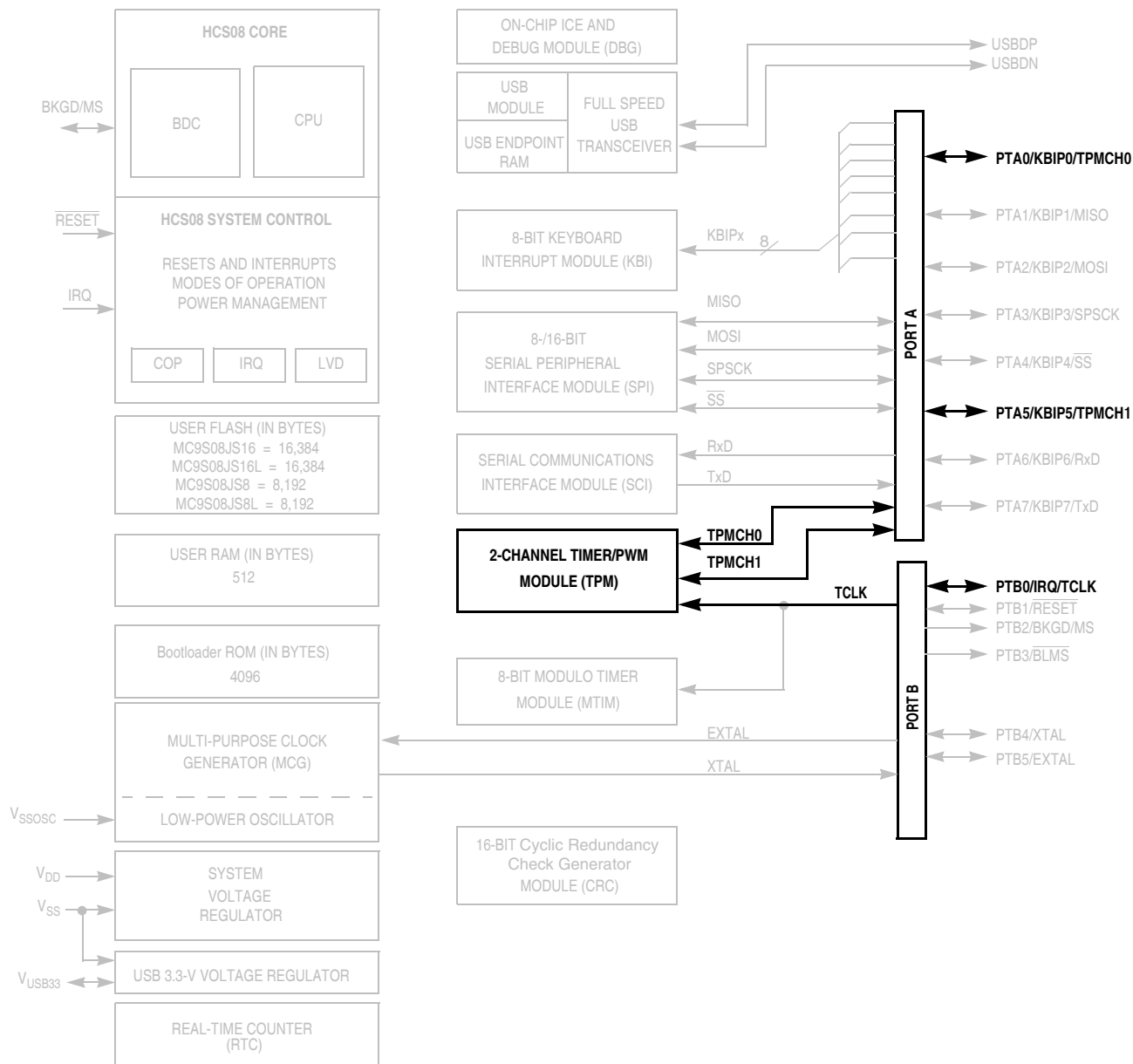
NOTE

TCLK, external input clock source for TPM and MTIM, is referenced as TPMCLK in TPM chapter.

14.2 Features

The timer system in the MC9S08JS16 series include one 2-channel TPM. Timer system features include:

- A total of two channels:
 - Each channel may be input capture, output compare, or buffered 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
- TPM may be configured for buffered, center-aligned pulse-width modulation (CPWM) on all channels
- Clock source to prescaler for TPM is independently selectable as bus clock, fixed system clock, or an external pin:
 - Prescale taps for divide by 1, 2, 4, 8, 16, 32, 64, or 128
 - External clock input: TPMCLK for TPM
- 16-bit free-running or up/down (CPWM) count operation
- 16-bit modulus register to control counter range
- Timer system enable
- One interrupt per channel plus a terminal count interrupt for TPM module



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1).
Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD} . IRQ must not be driven above V_{DD} .
4. \overline{RESET} contains integrated pullup device if PTB1 enabled as reset pin function (RSTPE = 1)
5. Pin contains integrated pullup device.
6. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 14-1. MC9S08JS16 Series Block Diagram Highlighting the TPM Module and Pins

14.3 TPMV3 Differences from Previous Versions

The TPMV3 is the latest version of the Timer/PWM module that addresses errata found in previous versions. The following section outlines the differences between TPMV3 and TPMV2 modules, and any considerations that should be taken when porting code.

Table 14-1. TPMV2 and TPMV3 Porting Considerations

Action	TPMV3	TPMV2
Write to TPMxCnTH:L registers¹		
Any write to TPMCNTH or TPMCNTL registers	Clears the TPM counter (TPMCNTH:L) and the prescaler counter.	Clears the TPM counter (TPMCNTH:L) only.
Read of TPMxCNTH:L registers¹		
In BDM mode, any read of TPMCNTH:L registers	Returns the value of the TPM counter that is frozen.	If only one byte of the TPMCNTH:L registers was read before the BDM mode became active, returns the latched value of TPMCNTH:L from the read buffer (instead of the frozen TPM counter value).
In BDM mode, a write to TPMSC, TPMCNTH or TPMCNTL	Clears this read coherency mechanism.	Does not clear this read coherency mechanism.
Read of TPMCnVH:L registers²		
In BDM mode, any read of TPMCnVH:L registers	Returns the value of the TPMCnVH:L register.	If only one byte of the TPMCnVH:L registers was read before the BDM mode became active, returns the latched value of TPMCNTH:L from the read buffer (instead of the value in the TPMCnVH:L registers).
In BDM mode, a write to TPMCnSC	Clears this read coherency mechanism.	Does not clear this read coherency mechanism.
Write to TPMCnVH:L registers		
In Input Capture mode, writes to TPMCnVH:L registers ³	Not allowed.	Allowed.
In Output Compare mode, when (CLKSB:CLKSA not = 0:0), writes to TPMCnVH:L registers ³	Update the TPMCnVH:L registers with the value of their write buffer at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.	Always update these registers when their second byte is written.

Table 14-1. TPMV2 and TPMV3 Porting Considerations (continued)

Action	TPMV3	TPMV2
In Edge-Aligned PWM mode when (CLKSB:CLKSA not = 00), writes to TPMCnVH:L registers	Update the TPMCnVH:L registers with the value of their write buffer after both bytes were written and when the TPM counter changes from (TPMMODH:L – 1) to (TPMMODH:L). Note: If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from 0xFFFE to 0xFFFF.	Update after both bytes are written and when the TPM counter changes from TPMMODH:L to 0x0000.
In Center-Aligned PWM mode when (CLKSB:CLKSA not = 00), writes to TPMCnVH:L registers ⁴	Update the TPMxVH:L registers with the value of their write buffer after both bytes are written and when the TPM counter changes from (TPMMODH:L – 1) to (TPMMODH:L). Note: If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from 0xFFFE to 0xFFFF.	Update after both bytes are written and when the TPM counter changes from TPMMODH:L to (TPMMODH:L – 1).
Center-Aligned PWM		
When TPMCnVH:L = TPMMODH:L ⁵	Produces 100% duty cycle.	Produces 0% duty cycle.
When TPMCnVH:L = (TPMMODH:L – 1) ⁶	Produces a near 100% duty cycle.	Produces 0% duty cycle.
TPMCnVH:L is changed from 0x0000 to a non-zero value ⁷	Waits for the start of a new PWM period to begin using the new duty cycle setting.	Changes the channel output at the middle of the current PWM period (when the count reaches 0x0000).
TPMCnVH:L is changed from a non-zero value to 0x0000 ⁸	Finishes the current PWM period using the old duty cycle setting.	Finishes the current PWM period using the new duty cycle setting.
Write to TPMMODH:L registers in BDM mode		
In BDM mode, a write to TPMxSC register	Clears the write coherency mechanism of TPMxMODH:L registers.	Does not clear the write coherency mechanism.

¹ For more information, refer to [Section 14.5.2, “TPM-Counter Registers \(TPMCNTH:TPMCNTL\)”](#) [SE110-TPM case 7]

² For more information, refer to [Section 14.5.5, “TPM Channel Value Registers \(TPMCnVH:TPMCnVL\)”](#).

³ For more information, refer to [Section 14.6.2.1, “Input Capture Mode.”](#)

⁴ For more information, refer to [Section 14.6.2.4, “Center-Aligned PWM Mode.”](#)

⁵ For more information, refer to [Section 14.6.2.4, “Center-Aligned PWM Mode.”](#) [SE110-TPM case 1]

⁶ For more information, refer to [Section 14.6.2.4, “Center-Aligned PWM Mode.”](#) [SE110-TPM case 2]

⁷ For more information, refer to [Section 14.6.2.4, “Center-Aligned PWM Mode.”](#) [SE110-TPM case 3 and 5]

⁸ For more information, refer to [Section 14.6.2.4, “Center-Aligned PWM Mode.”](#) [SE110-TPM case 4]

14.3.1 Migrating from TPMV1

In addition to [Section 14.3, “TPMV3 Differences from Previous Versions,”](#) keep in mind the following considerations when migrating from a device that uses TPMV1.

- You can write to the Channel Value register (TPMCnV) when the timer is not in input capture mode for TPMV2, not TPMV3.
- In edge- or center- aligned modes, the Channel Value register (TPMCnV) registers only update when the timer changes from TPMMOD-1 to TPMMOD, or in the case of a free running timer from 0xFFFE to 0xFFFF.
- Also, when configuring the TPM modules, it is best to write to TPMSC before TPMCnV as a write to TPMxSC resets the coherency mechanism on the TPMCnV registers.

Table 14-2. Migrating to TPMV3 Considerations

When...	Action / Best Practice
Writing to the Channel Value Register (TPMCnV) register...	Timer must be in Input Capture mode.
Updating the Channel Value Register (TPMCnV) register in edge-aligned or center-aligned modes...	Only occurs when the timer changes from TPMMOD – 1 to TPMMOD (or in the case of a free running timer, from 0xFFFE to 0xFFFF).
Resetting the coherency mechanism for the Channel Value Register (TPMCnV) register...	Write to TPMSC.
Configuring the TPM modules...	Write first to TPMSC and then to TPMCnV register.

14.3.2 Features

The TPM includes these distinctive features:

- One to eight channels:
 - Each channel may be input capture, output compare, or edge-aligned PWM
 - Rising-Edge, falling-edge, or any-edge input capture trigger
 - Set, clear, or toggle output compare action
 - Selectable polarity on PWM outputs
- Module may be configured for buffered, center-aligned pulse-width-modulation (CPWM) on all channels
- Timer clock source selectable as prescaled bus clock, fixed system clock, or an external clock pin
 - Prescale taps for divide-by 1, 2, 4, 8, 16, 32, 64, or 128
 - Fixed system clock source are synchronized to the bus clock by an on-chip synchronization circuit
 - External clock pin may be shared with any timer channel pin or a separated input pin
- 16-bit free-running or modulo up/down count operation
- Timer system enable
- One interrupt per channel plus terminal count interrupt

14.3.3 Modes of Operation

In general, TPM channels may be independently configured to operate in input capture, output compare, or edge-aligned PWM modes. A control bit allows the whole TPM (all channels) to switch to center-aligned PWM mode. When center-aligned PWM mode is selected, input capture, output compare, and edge-aligned PWM functions are not available on any channels of this TPM module.

When the microcontroller is in active BDM background or BDM foreground mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all system clocks, including the main oscillator, are stopped; therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally. Provided the TPM does not need to produce a real time reference or provide the interrupt source(s) needed to wake the MCU from wait mode, the user can save power by disabling TPM functions before entering wait mode.

- Input capture mode

When a selected edge event occurs on the associated MCU pin, the current value of the 16-bit timer counter is captured into the channel value register and an interrupt flag bit is set. Rising edges, falling edges, any edge, or no edge (disable channel) may be selected as the active edge which triggers the input capture.
- Output compare mode

When the value in the timer counter register matches the channel value register, an interrupt flag bit is set, and a selected output action is forced on the associated MCU pin. The output compare action may be selected to force the pin to zero, force the pin to one, toggle the pin, or ignore the pin (used for software timing functions).

- Edge-aligned PWM mode
The value of a 16-bit modulo register plus 1 sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. The user may also choose the polarity of the PWM output signal. Interrupts are available at the end of the period and at the duty-cycle transition point. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within a TPM.
- Center-aligned PWM mode
Twice the value of a 16-bit modulo register sets the period of the PWM output, and the channel-value register sets the half-duty-cycle duration. The timer counter counts up until it reaches the modulo value and then counts down until it reaches zero. As the count matches the channel value register while counting down, the PWM output becomes active. When the count matches the channel value register while counting up, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. This type of PWM is required for types of motors used in small appliances.

This is a high-level description only. Detailed descriptions of operating modes are in later sections.

14.3.4 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMCH_n (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 14-2 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, TPMMODH:TPMMODL, 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 TPMCNT counter resets the counter, regardless of the data value written.

Timer/Pulse-Width Modulator (S08TPMV3)

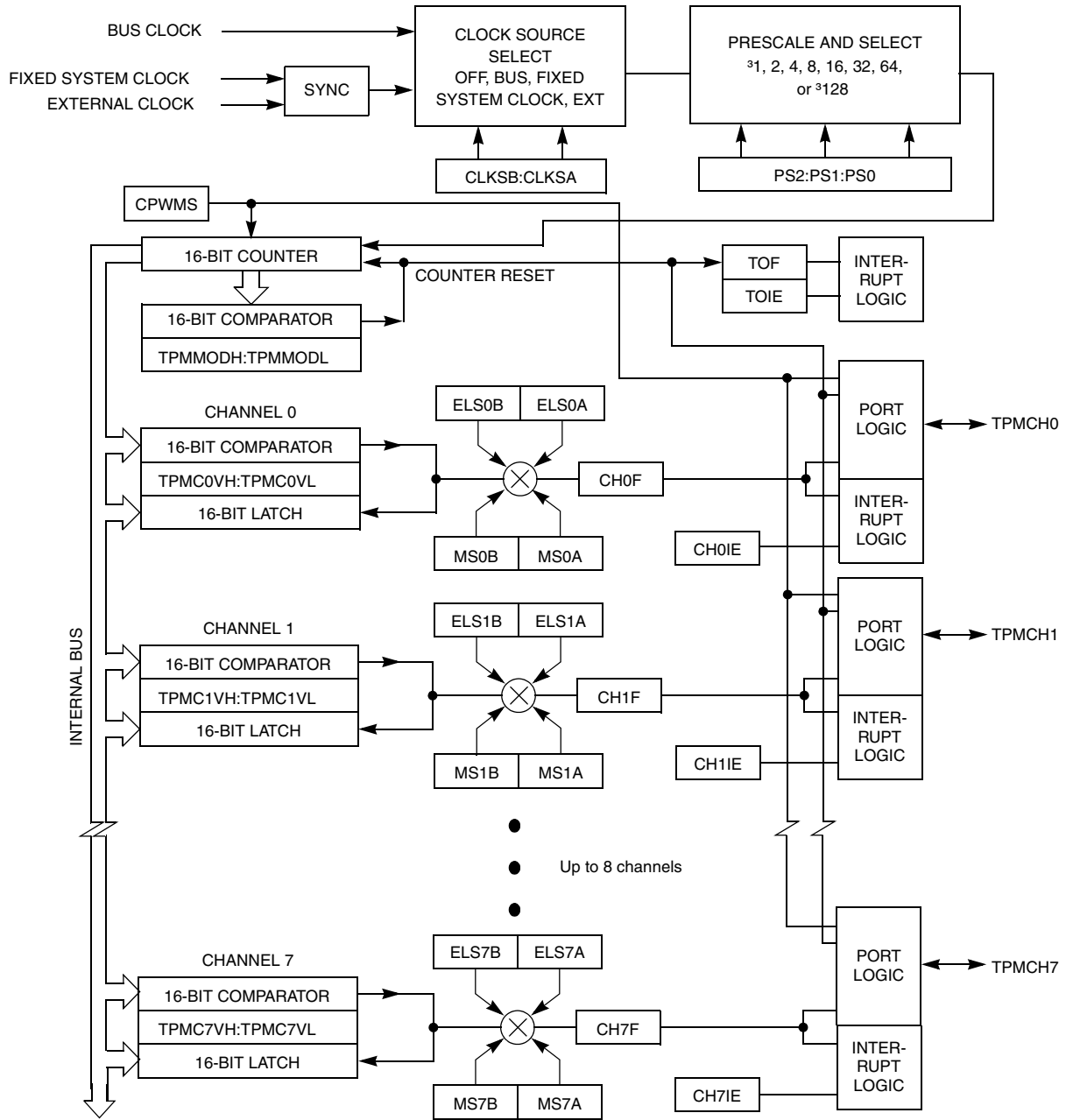


Figure 14-2. TPM Block Diagram

The TPM channels are programmable independently as input capture, output compare, or edge-aligned PWM channels. Alternately, the TPM can be configured to produce CPWM outputs on all channels. When the TPM is configured for CPWMs, the counter operates as an up/down counter; input capture, output compare, and EPWM functions are not practical.

If a channel is configured as input capture, an internal pullup device may be enabled for that channel. The details of how a module interacts with pin controls depends upon the chip implementation because the I/O pins and associated general purpose I/O controls are not part of the module. Refer to the discussion of the I/O port logic in a full-chip specification.

Because center-aligned PWMs are usually used to drive 3-phase AC-induction motors and brushless DC motors, they are typically used in sets of three or six channels.

14.4 Signal Description

Table 14-3 shows the user-accessible signals for the TPM. The number of channels may be varied from one to eight. When an external clock is included, it can be shared with the same pin as any TPM channel; however, it could be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

Table 14-3. Signal Properties

Name	Function
EXTCLK ¹	External clock source which may be selected to drive the TPM counter.
TPMCHn ²	I/O pin associated with TPM channel n

¹ When preset, this signal can share any channel pin; however depending upon full-chip implementation, this signal could be connected to a separate external pin.

² n=channel number (1 to 8)

Refer to documentation for the full-chip for details about reset states, port connections, and whether there is any pullup device on these pins.

TPM channel pins can be associated with general purpose I/O pins and have passive pullup devices which can be enabled with a control bit when the TPM or general purpose I/O controls have configured the associated pin as an input. When no TPM function is enabled to use a corresponding pin, the pin reverts to being controlled by general purpose I/O controls, including the port-data and data-direction registers. Immediately after reset, no TPM functions are enabled, so all associated pins revert to general purpose I/O control.

14.4.1 Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 14-3 grouped all channel pins together, any TPM pin can be shared with the external clock source signal. Since I/O pin logic is not part of the TPM, refer to full-chip documentation for a specific derivative for more details about the interaction of TPM pin functions and general purpose I/O controls including port data, data direction, and pullup controls.

14.4.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow the user to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock which drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer; the frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock, to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin will not be usable for channel I/O function when selected as the external clock source. It is the user's responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can still be used in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

14.4.1.2 TPMCHn — TPM Channel n I/O Pin(s)

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled whenever a port pin is acting as an input.

The TPM channel does not control the I/O pin when (ELSnB:ELSnA = 0:0) or when (CLKSB:CLKSA = 0:0) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the TPMCHn pins are all controlled by the TPM system. When CPWMS=0, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPMCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges will trigger input-capture events. A synchronizer based on the bus clock is used to synchronize input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA = 0:1 and ELSnB:ELSnA not = 0:0), the associated data direction control is overridden, the TPMCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPMCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event—then the pin is toggled.

When a channel is configured for edge-aligned PWM (CPWMS=0, MSnB=1 and ELSnB:ELSnA not = 0:0), the data direction is overridden, the TPMCHn pin is forced to be an output controlled by the TPM, and ELSnA controls the polarity of the PWM output signal on the pin. When ELSnB:ELSnA=1:0, the TPMCHn pin is forced high at the start of each new period (TPMCNT=0x0000), and the pin is forced low when the channel value register matches the timer counter. When ELSnA=1, the TPMCHn pin is forced low at the start of each new period (TPMCNT=0x0000), and the pin is forced high when the channel value register matches the timer counter.

TPMMODH:TPMMODL = 0x0008
 TPMMODH:TPMMODL = 0x0005

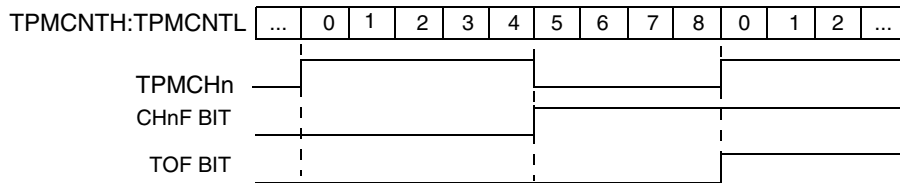


Figure 14-3. High-True Pulse of an Edge-Aligned PWM

TPMMODH:TPMMODL = 0x0008
 TPMMODH:TPMMODL = 0x0005

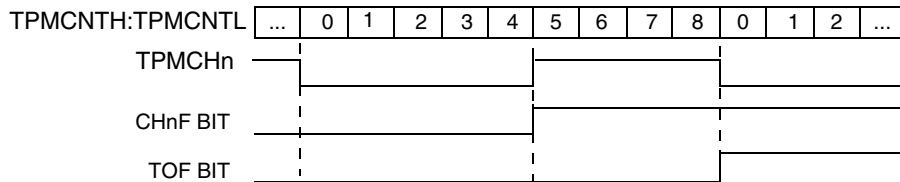


Figure 14-4. Low-True Pulse of an Edge-Aligned PWM

When the TPM is configured for center-aligned PWM (and ELSnB:ELSnA not = 0:0), the data direction for all channels in this TPM are overridden, the TPMCHn pins are forced to be outputs controlled by the TPM, and the ELSnA bits control the polarity of each TPMCHn output. If ELSnB:ELSnA=1:0, the corresponding TPMCHn pin is cleared when the timer counter is counting up, and the channel value register matches the timer counter; the TPMCHn pin is set when the timer counter is counting down, and the channel value register matches the timer counter. If ELSnA=1, the corresponding TPMCHn pin is set when the timer counter is counting up and the channel value register matches the timer counter; the TPMCHn pin is cleared when the timer counter is counting down and the channel value register matches the timer counter.

TPMMODH:TPMMODL = 0x0008
 TPMMODH:TPMMODL = 0x0005

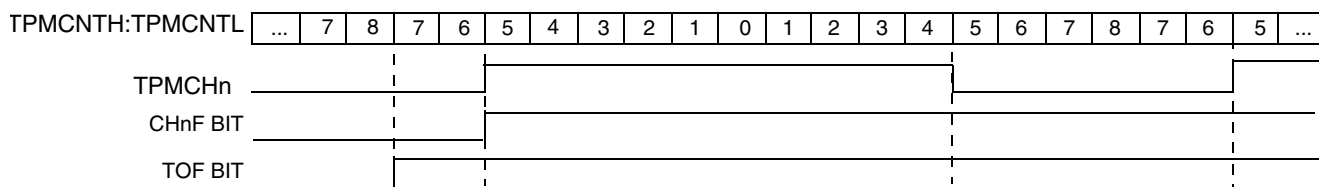


Figure 14-5. High-True Pulse of a Center-Aligned PWM

TPMMODH:TPMMODL = 0x0008
 TPMMODH:TPMMODL = 0x0005

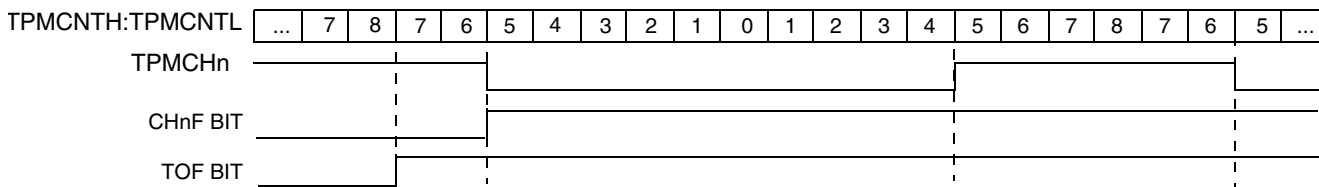


Figure 14-6. Low-True Pulse of a Center-Aligned PWM

14.5 Register Definition

This section consists of register descriptions in address order.

14.5.1 TPM Status and Control Register (TPMSC)

TPMSC 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.

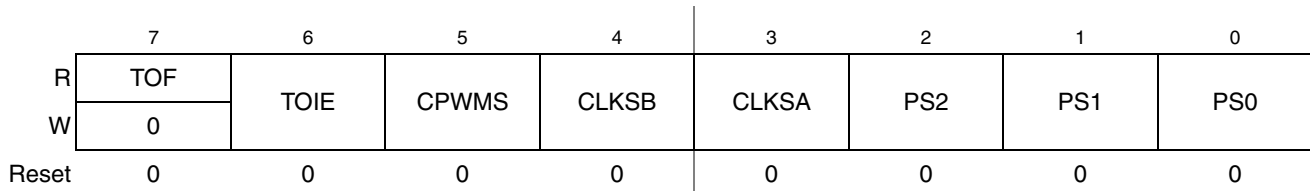


Figure 14-7. TPM Status and Control Register (TPMSC)

Table 14-4. TPMSC Field Descriptions

Field	Description
7 TOF	<p>Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect.</p> <p>0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed</p>
6 TOIE	<p>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.</p> <p>0 TOF interrupts inhibited (use for software polling) 1 TOF interrupts enabled</p>
5 CPWMS	<p>Center-aligned PWM select. When present, this read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS.</p> <p>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.</p>
4–3 CLKS[B:A]	<p>Clock source selects. As shown in Table 14-5, this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The fixed system clock source is only meaningful in systems with a PLL-based or FLL-based system clock. When there is no PLL or FLL, the fixed-system clock source is the same as the bus rate clock. The external source is synchronized to the bus clock by TPM module, and the fixed system clock source (when a PLL or FLL is present) is synchronized to the bus clock by an on-chip synchronization circuit. When a PLL or FLL is present but not enabled, the fixed-system clock source is the same as the bus-rate clock.</p>
2–0 PS[2:0]	<p>Prescale factor select. This 3-bit field selects one of 8 division factors for the TPM clock input as shown in Table 14-6. This prescaler is located after any clock source synchronization or clock source selection so it affects the clock source selected to drive the TPM system. The new prescale factor will affect the clock source on the next system clock cycle after the new value is updated into the register bits.</p>

Table 14-5. TPM-Clock-Source Selection

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disable)
01	Bus rate clock
10	Fixed system clock
11	External source

Table 14-6. Prescale Factor Selection

PS2:PS1:PS0	TPM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

14.5.2 TPM-Counter Registers (TPMCNTH:TPMCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMCNTH or TPMCNTL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This allows coherent 16-bit reads in either big-endian or little-endian order which makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the timer status/control register (TPMSC).

Reset clears the TPM counter registers. Writing any value to TPMCNTH or TPMCNTL also clears the TPM counter (TPMCNTH:TPMCNTL) and resets the coherency mechanism, regardless of the data involved in the write.

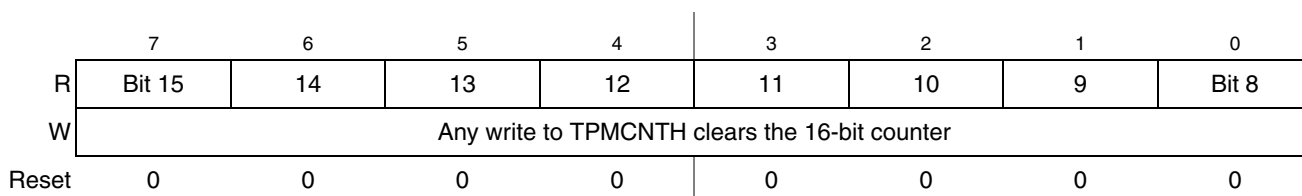
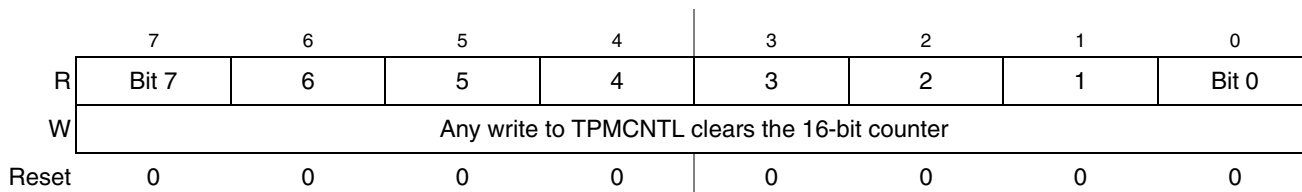


Figure 14-8. TPM Counter Register High (TPMCNTH)


Figure 14-9. TPM Counter Register Low (TPMCNTL)

When BDM is active, the timer counter is frozen (this is the value that will be read by user); the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both counter halves are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution.

In BDM mode, writing any value to TPMSC, TPMCNTH or TPMCNTL registers resets the read coherency mechanism of the TPMCNTH:L registers, regardless of the data involved in the write.

14.5.3 TPM Counter Modulo Registers (TPMMODH:TPMMODL)

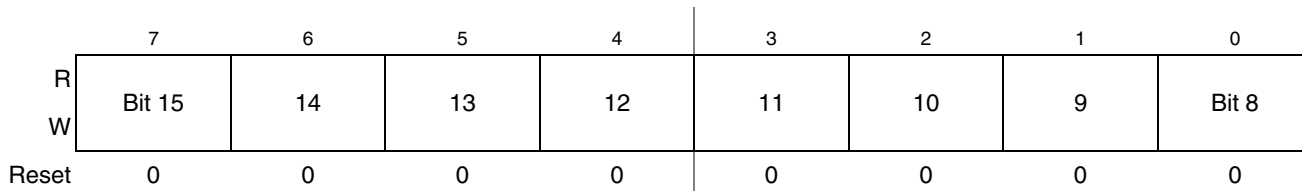
The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock, and the overflow flag (TOF) becomes set. Writing to TPMMODH or TPMMODL inhibits the TOF bit and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000 which results in a free running timer counter (modulo disabled).

Writing to either byte (TPMMODH or TPMMODL) latches the value into a buffer and the registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), then the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), then the registers are updated after both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF

The latching mechanism may be manually reset by writing to the TPMSC address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.


Figure 14-10. TPM Counter Modulo Register High (TPMMODH)

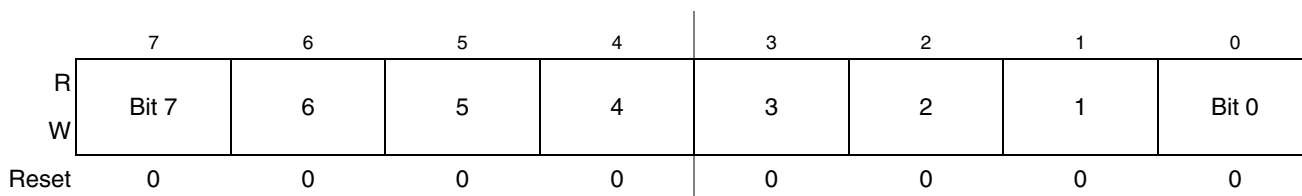


Figure 14-11. TPM Counter Modulo Register Low (TPMMODL)

Reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

14.5.4 TPM Channel n Status and Control Register (TPMCnSC)

TPMCnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

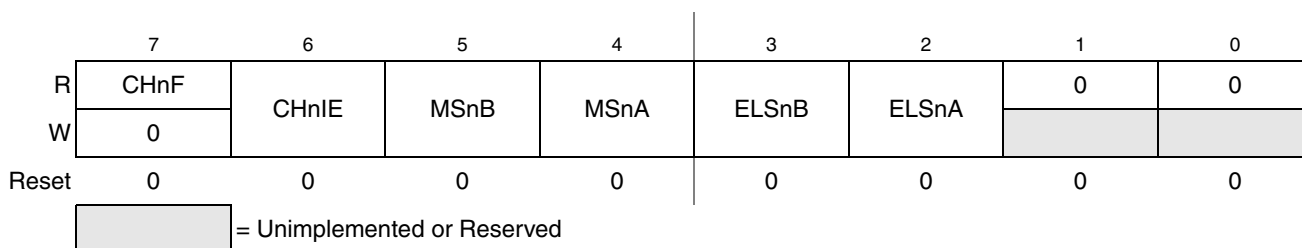


Figure 14-12. TPM Channel n Status and Control Register (TPMCnSC)

Table 14-7. TPMCnSC Field Descriptions

Field	Description
7 CHnF	Channel n flag. When channel n is an input-capture channel, this read/write bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF will not be set even when the value in the TPM counter registers matches the value in the TPM channel n value registers. A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPMCnSC while CHnF is set and then writing a logic 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF remains set after the clear sequence completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost due to clearing a previous CHnF. Reset clears the CHnF bit. Writing a logic 1 to CHnF has no effect. 0 No input capture or output compare event occurred on channel n 1 Input capture or output compare event on channel n
6 CHnIE	Channel n interrupt enable. This read/write bit enables interrupts from channel n. Reset clears CHnIE. 0 Channel n interrupt requests disabled (use for software polling) 1 Channel n interrupt requests enabled
5 MSnB	Mode select B for TPM channel n. When CPWMS=0, MSnB=1 configures TPM channel n for edge-aligned PWM mode. Refer to the summary of channel mode and setup controls in Table 14-8 .

Table 14-7. TPMCnSC Field Descriptions (continued)

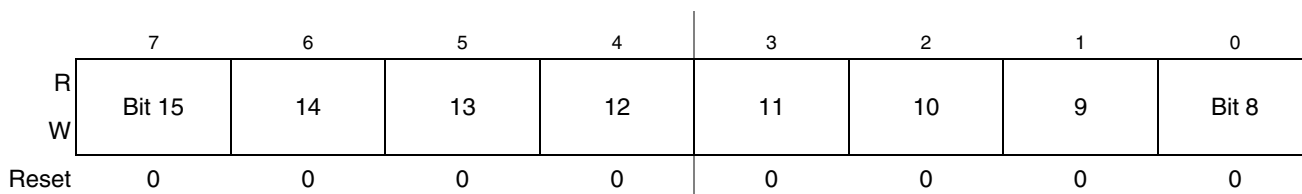
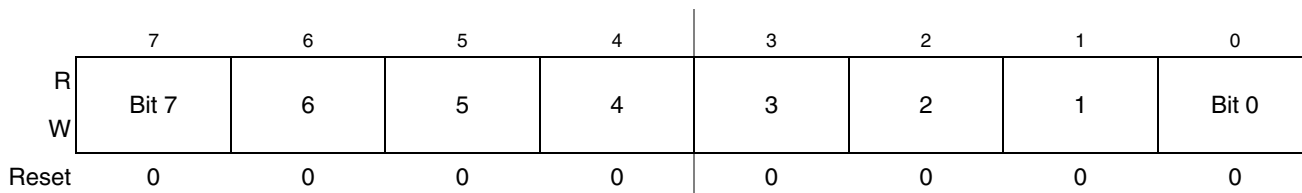
Field	Description
4 MSnA	Mode select A for TPM channel n. When CPWMS=0 and MSnB=0, MSnA configures TPM channel n for input-capture mode or output compare mode. Refer to Table 14-8 for a summary of channel mode and setup controls. Note: If the associated port pin is not stable for at least two bus clock 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 14-8 , these bits select the polarity of the input edge that triggers an input capture event, select the level that will be driven in response to an output compare match, or select the polarity of the PWM output. Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general purpose I/O pin not related to any timer functions. This function is typically used to temporarily disable an input capture channel or to make the timer pin available as a general purpose I/O pin when the associated timer channel is set up as a software timer that does not require the use of a pin.

Table 14-8. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	XX	00	Pin not used for TPM - revert to general purpose I/O or other peripheral control	
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	01	Output compare	Toggle output on compare
		10		Clear output on compare
		11		Set output on compare
1X	10	Edge-aligned PWM	High-true pulses (clear output on compare)	
	X1		Low-true pulses (set output on compare)	
1	XX	10	Center-aligned PWM	High-true pulses (clear output on compare-up)
		X1		Low-true pulses (set output on compare-up)

14.5.5 TPM Channel Value Registers (TPMCnVH:TPMCnVL)

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.


Figure 14-13. TPM Channel Value Register High (TPMCnVH)

Figure 14-14. TPM Channel Value Register Low (TPMCnVL)

In input capture mode, reading either byte (TPMCnVH or TPMCnVL) 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 TPMCnSC register is written (whether BDM mode is active or not). Any write to the channel registers will be ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMCnSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the TPMCnVH and TPMCnVL 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 (TPMCnVH or TPMCnVL) latches the value into a buffer. After both bytes are written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKS_B:CLKS_A bits and the selected mode, so:

- If (CLKS_B:CLKS_A = 0:0), then the registers are updated when the second byte is written.
- If (CLKS_B:CLKS_A not = 0:0 and in output compare mode) then the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).
- If (CLKS_B:CLKS_A not = 0:0 and in EPWM or CPWM modes), then the registers are updated after the both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

The latching mechanism may be manually reset by writing to the TPMCnSC register (whether BDM mode is active or not). This latching mechanism allows coherent 16-bit writes in either big-endian or little-endian order which is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active even if one or both halves of the channel register are written while BDM is active. Any write to the channel registers bypasses the buffer latches and directly write to the channel register while BDM is active. The values written to the channel register while BDM is active are used for PWM & output compare operation once normal execution resumes. Writes to the channel

registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

14.6 Functional Description

All TPM functions are associated with a central 16-bit counter which allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

The following sections describe the main counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend upon the operating mode, these topics will be covered in the associated mode explanation sections.

14.6.1 Counter

All timer functions are based on the main 16-bit counter (TPMCNTH:TPMCNTL). This section discusses selection of the clock source, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

14.6.1.1 Counter Clock Source

The 2-bit field, CLKS_B:CLKS_A, in the timer status and control register (TPMSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See [Table 14-5](#). After any MCU reset, CLKS_B:CLKS_A=0:0 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKS_B:CLKS_A field) does not affect the values in the counter or other timer registers.

Table 14-9. TPM Clock Source Selection

CLKSB:CLKSA	TPM Clock Source to Prescaler Input
00	No clock selected (TPM counter disabled)
01	Bus rate clock
10	Fixed system clock
11	External source

The bus rate clock is the main system bus clock for the MCU. This clock source requires no synchronization because it is the clock that is used for all internal MCU activities including operation of the CPU and buses.

In MCUs that have no PLL and FLL or the PLL and FLL are not engaged, the fixed system clock source is the same as the bus-rate-clock source, and it does not go through a synchronizer. When a PLL or FLL is present and engaged, a synchronizer is required between the crystal divided-by two clock source and the timer counter so counter transitions will be properly aligned to bus-clock transitions. A synchronizer will be used at chip level to synchronize the crystal-related source clock to the bus clock.

The external clock source may be connected to any TPM channel pin. This clock source always has to pass through a synchronizer to assure that counter transitions are properly aligned to bus clock transitions. The bus-rate clock drives the synchronizer; therefore, to meet Nyquist criteria even with jitter, the frequency of the external clock source must not be faster than the bus rate divided-by four. With ideal clocks the external clock can be as fast as bus clock divided by four.

When the external clock source shares the TPM channel pin, this pin should not be used for other channel timing functions. For example, it would be ambiguous to configure channel 0 for input capture when the TPM channel 0 pin was also being used as the timer external clock source. (It is the user's responsibility to avoid such settings.) The TPM channel could still be used in output compare mode for software timing functions (pin controls set not to affect the TPM channel pin).

14.6.1.2 Counter Overflow and Modulo Reset

An interrupt flag and enable are associated with the 16-bit main counter. The flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE=0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE=1) where a static hardware interrupt is generated whenever the TOF flag is equal to one.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS=1). In the simplest mode, there is no modulus limit and the TPM is not in CPWMS=1 mode. In this case, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS=1), the TOF flag gets set as the counter changes direction at the end of the count value set in the modulus register (that is, at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

14.6.1.3 Counting Modes

The main timer counter has two counting modes. When center-aligned PWM is selected (CPWMS=1), the counter operates in up/down counting mode. Otherwise, the counter operates as a simple up counter. As an up counter, the timer counter counts from 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMMODH:TPMMODL.

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. Both 0x0000 and the terminal count value are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) becomes set at the end of the terminal-count period (as the count changes to the next lower count value).

14.6.1.4 Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to either half of TPMCNTH or TPMCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only half of the counter was read before resetting the count.

14.6.2 Channel Mode Selection

Provided CPWMS=0, the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

14.6.2.1 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 (TPMCnVH:TPMCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

In input capture mode, the TPMCnVH and TPMCnVL registers are read only.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMCnSC).

An input capture event sets a flag bit (CHnF) which may optionally generate a CPU interrupt request.

While in BDM, the input capture function works as configured by the user. When an external event occurs, the TPM latches the contents of the TPM counter (which is frozen because of the BDM mode) into the channel value registers and sets the flag bit.

14.6.2.2 Output Compare Mode

With the output-compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel-value registers of an output-compare channel, the TPM can set, clear, or toggle the channel pin.

In output compare mode, values are transferred to the corresponding timer channel registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMCnSC).

An output compare event sets a flag bit (CHnF) which may optionally generate a CPU-interrupt request.

14.6.2.3 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 (TPMMODH:TPMMODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPMCnVH:TPMCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 14-15). The time between the modulus overflow and the output compare is the pulse width. If ELSnA=0, the counter overflow forces the PWM signal high, and the output compare forces the PWM signal low. If ELSnA=1, the counter overflow forces the PWM signal low, and the output compare forces the PWM signal high.

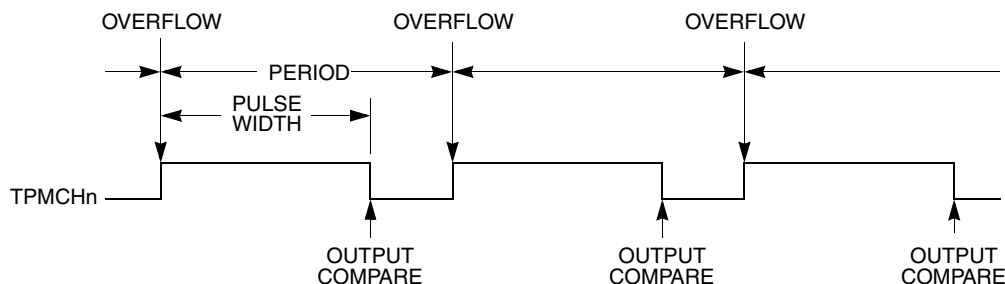


Figure 14-15. PWM Period and Pulse Width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. 100% duty cycle can be achieved by setting the timer-channel register (TPMCnVH:TPMCnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF in order to get 100% duty cycle.

Because the TPM may be used in an 8-bit MCU, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMCnVH and TPMCnVL, actually write to buffer registers. In edge-aligned PWM mode, values are transferred to the corresponding timer-channel registers according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the

TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

14.6.2.4 Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The output compare value in TPMCnVH:TPMCnVL determines the pulse width (duty cycle) of the PWM signal while the period is determined by the value in TPMMODH:TPMMODL. TPMMODH:TPMMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

$$\text{pulse width} = 2 \times (\text{TPMCnVH:TPMCnVL})$$

$$\text{period} = 2 \times (\text{TPMMODH:TPMMODL}); \text{TPMMODH:TPMMODL}=0\text{x}0001\text{-}0\text{x}7\text{FFF}$$

If the channel-value register TPMCnVH:TPMCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMCnVH:TPMCnVL is a positive value (bit 15 clear) and is greater than the (non-zero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if you do not need to generate 100% duty cycle). This is not a significant limitation. The resulting period would be much longer than required for normal applications.

TPMMODH:TPMMODL=0x0000 is a special case that should not be used with center-aligned PWM mode. When CPWMS=0, this case corresponds to the counter running free from 0x0000 through 0xFFFF, but when CPWMS=1 the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

The output compare value in the TPM channel registers (times 2) determines the pulse width (duty cycle) of the CPWM signal (Figure 14-16). If ELSnA=0, a compare occurred while counting up forces the CPWM output signal low and a compare occurred while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPMMODH:TPMMODL, then counts down until it reaches zero. This sets the period equal to two times TPMMODH:TPMMODL.

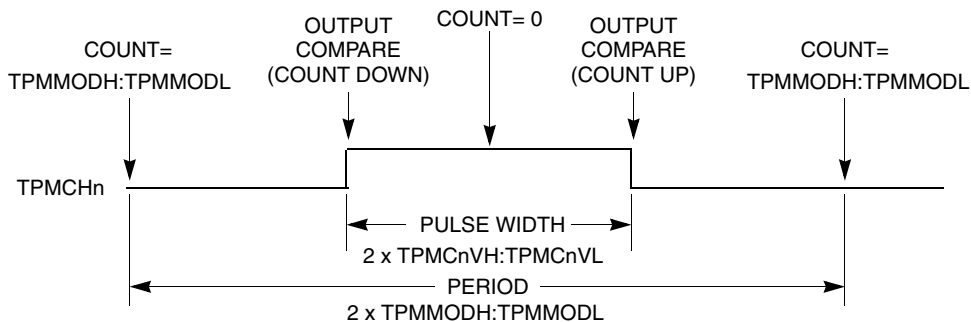


Figure 14-16. CPWM Period and Pulse Width (ELSnA=0)

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Input capture, output compare, and edge-aligned PWM functions do not make sense when the counter is operating in up/down counting mode so this implies that all active channels within a TPM must be used in CPWM mode when CPWMS=1.

The TPM may be used in an 8-bit MCU. The settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMMODH, TPMMODL, TPMCnVH, and TPMCnVL, actually write to buffer registers.

In center-aligned PWM mode, the TPMCnVH:L registers are updated with the value of their write buffer according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMMODH:TPMMODL - 1) to (TPMMODH:TPMMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

When TPMCNTH:TPMCNTL=TPMMODH:TPMMODL, the TPM can optionally generate a TOF interrupt (at the end of this count).

Writing to TPMSC cancels any values written to TPMMODH and/or TPMMODL and resets the coherency mechanism for the modulo registers. Writing to TPMCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMCnVH:TPMCnVL.

14.7 Reset Overview

14.7.1 General

The TPM is reset whenever any MCU reset occurs.

14.7.2 Description of Reset Operation

Reset clears the TPMSC register which disables clocks to the TPM and disables timer overflow interrupts (TOIE=0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared which configures all TPM channels for input-capture operation with the associated pins disconnected from I/O pin logic (so all MCU pins related to the TPM revert to general purpose I/O pins).

14.8 Interrupts

14.8.1 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 14-10](#) which shows the interrupt name, the name of any local enable that can block the interrupt request from leaving the TPM and getting recognized by the separate interrupt processing logic.

Table 14-10. Interrupt Summary

Interrupt	Local Enable	Source	Description
TOF	TOIE	Counter overflow	Set each time the timer counter reaches its terminal count (at transition to next count value which is usually 0x0000)
CHnF	CHnIE	Channel event	An input capture or output compare event took place on channel n

The TPM module will provide a high-true interrupt signal. Vectors and priorities are determined at chip integration time in the interrupt module so refer to the user's guide for the interrupt module or to the chip's complete documentation for details.

14.8.2 Description of Interrupt Operation

For each interrupt source in the TPM, a flag bit is set upon recognition of the interrupt condition such as timer overflow, channel-input capture, or output-compare events. This flag may be read (polled) by software to determine that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt will generate whenever the associated interrupt flag equals one. The user's software must perform a sequence of steps to clear the interrupt flag before returning from the interrupt-service routine.

TPM interrupt flags are cleared by a two-step process including a read of the flag bit while it is set (1) followed by a write of zero (0) to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

14.8.2.1 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.

14.8.2.1.1 Normal Case

Normally TOF is set when the timer counter changes from 0xFFFF to 0x0000. When the TPM is not configured for center-aligned PWM (CPWMS=0), TOF gets set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. This case corresponds to the normal meaning of counter overflow.

14.8.2.1.2 Center-Aligned PWM Case

When CPWMS=1, TOF gets set when the timer counter changes direction from up-counting to down-counting at the end of the terminal count (the value in the modulo register). In this case the TOF corresponds to the end of a PWM period.

14.8.2.2 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).

14.8.2.2.1 Input Capture Events

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select no edge (off), rising edges, falling edges or any edge as the edge which triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the two-step sequence described in [Section 14.8.2, "Description of Interrupt Operation."](#)

14.8.2.2.2 Output Compare Events

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the two-step sequence described [Section 14.8.2, "Description of Interrupt Operation."](#)

14.8.2.2.3 PWM End-of-Duty-Cycle Events

For channels configured for PWM operation there are two possibilities. When the channel is configured for edge-aligned PWM, the channel flag gets set when the timer counter matches the channel value register which marks the end of the active duty cycle period. When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle period which are the times when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described [Section 14.8.2, "Description of Interrupt Operation."](#)

1. Write to TPMxCnTH:L registers ([Section 14.5.2, "TPM-Counter Registers \(TPMCNTH:TPMCNTL\)"](#)) [SE110-TPM case 7]
 Any write to TPMxCNTH or TPMxCNTL registers in TPM v3 clears the TPM counter (TPMxCNTH:L) and the prescaler counter. Instead, in the TPM v2 only the TPM counter is cleared in this case.
2. Read of TPMxCnTH:L registers ([Section 14.5.2, "TPM-Counter Registers \(TPMCNTH:TPMCNTL\)"](#))
 - In TPM v3, any read of TPMxCNTH:L registers during BDM mode returns the value of the TPM counter that is frozen. In TPM v2, if only one byte of the TPMxCNTH:L registers was read before the BDM mode became active, then any read of TPMxCNTH:L registers during BDM mode returns the latched value of TPMxCNTH:L from the read buffer instead of the frozen TPM counter value.

- This read coherency mechanism is cleared in TPM v3 in BDM mode if there is a write to TPMxSC, TPMxCNTH or TPMxCNTL. Instead, in these conditions the TPM v2 does not clear this read coherency mechanism.
3. Read of TPMxCnVH:L registers (Section 14.5.5, “TPM Channel Value Registers (TPMCnVH:TPMCnVL)”)
 - In TPM v3, any read of TPMxCnVH:L registers during BDM mode returns the value of the TPMxCnVH:L register. In TPM v2, if only one byte of the TPMxCnVH:L registers was read before the BDM mode became active, then any read of TPMxCnVH:L registers during BDM mode returns the latched value of TPMxCNTH:L from the read buffer instead of the value in the TPMxCnVH:L registers.
 - This read coherency mechanism is cleared in TPM v3 in BDM mode if there is a write to TPMxCnSC. Instead, in this condition the TPM v2 does not clear this read coherency mechanism.
 4. Write to TPMxCnVH:L registers
 - Input Capture Mode (Section 14.6.2.1, “Input Capture Mode”)

In this mode the TPM v3 does not allow the writes to TPMxCnVH:L registers. Instead, the TPM v2 allows these writes.
 - Output Compare Mode (Section 14.6.2.2, “Output Compare Mode”)

In this mode and if (CLKSB:CLKSA not = 0:0), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer at the next change of the TPM counter (end of the prescaler counting) after the second byte is written. Instead, the TPM v2 always updates these registers when their second byte is written.
 - Edge-Aligned PWM (Section 14.6.2.3, “Edge-Aligned PWM Mode”)

In this mode and if (CLKSB:CLKSA not = 00), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer after that the both bytes were written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from 0xFFFFE to 0xFFFF. Instead, the TPM v2 makes this update after that the both bytes were written and when the TPM counter changes from TPMxMODH:L to 0x0000.
 - Center-Aligned PWM (Section 14.6.2.4, “Center-Aligned PWM Mode”)

In this mode and if (CLKSB:CLKSA not = 00), the TPM v3 updates the TPMxCnVH:L registers with the value of their write buffer after that the both bytes were written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from 0xFFFFE to 0xFFFF. Instead, the TPM v2 makes this update after that the both bytes were written and when the TPM counter changes from TPMxMODH:L to (TPMxMODH:L - 1).
 5. Center-Aligned PWM (Section 14.6.2.4, “Center-Aligned PWM Mode”)
 - $TPMxCnVH:L = TPMxMODH:L$ [SE110-TPM case 1]

In this case, the TPM v3 produces 100% duty cycle. Instead, the TPM v2 produces 0% duty cycle.
 - $TPMxCnVH:L = (TPMxMODH:L - 1)$ [SE110-TPM case 2]

In this case, the TPM v3 produces almost 100% duty cycle. Instead, the TPM v2 produces 0% duty cycle.

- TPMxCnVH:L is changed from 0x0000 to a non-zero value [SE110-TPM case 3 and 5]

In this case, the TPM v3 waits for the start of a new PWM period to begin using the new duty cycle setting. Instead, the TPM v2 changes the channel output at the middle of the current PWM period (when the count reaches 0x0000).

- TPMxCnVH:L is changed from a non-zero value to 0x0000 [SE110-TPM case 4]

In this case, the TPM v3 finishes the current PWM period using the old duty cycle setting. Instead, the TPM v2 finishes the current PWM period using the new duty cycle setting.

6. Write to TPMxMODH:L registers in BDM mode ([Section 14.5.3, “TPM Counter Modulo Registers \(TPMMODH:TPMMODL\)”](#))

In the TPM v3 a write to TPMxSC register in BDM mode clears the write coherency mechanism of TPMxMODH:L registers. Instead, in the TPM v2 this coherency mechanism is not cleared when there is a write to TPMxSC register.

Chapter 15

Universal Serial Bus Device Controller (S08USBV1)

15.1 Introduction

This section describes a universal serial bus (USB) 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, P/S2, and IEEE 1284 for PC peripherals.

This USB module supports full-speed (FS) USB signaling including an interface to an on-chip FS USB 2.0 compatible transceiver. The on-chip USB transceiver is used with an on-chip 3.3-V voltage regulator. This USB module also provides control options for both the on-chip USB transceiver and voltage regulator.

15.1.1 Clocking Requirements

The S08USBV1 requires two clock sources, the 24 MHz bus clock and a 48 MHz reference clock. The 48 MHz clock is sourced directly from MCGOUT. To achieve the 48 MHz clock rate, the MCG must be configured properly for PLL engaged external (PEE) mode with an external crystal.

For USB operation, examples of MCG configuration using PEE mode include:

- 2 MHz crystal — RDIV = 000 and VDIV = 0110
- 4 MHz crystal — RDIV = 001 and VDIV = 0110

15.1.2 Current Consumption in USB Suspend

In USB suspend mode, the S08USBV1 current consumption is limited to 500 μ A. When the USB device goes into suspend mode, the firmware typically enters stop3 to meet the USB suspend requirements on current consumption.

NOTE

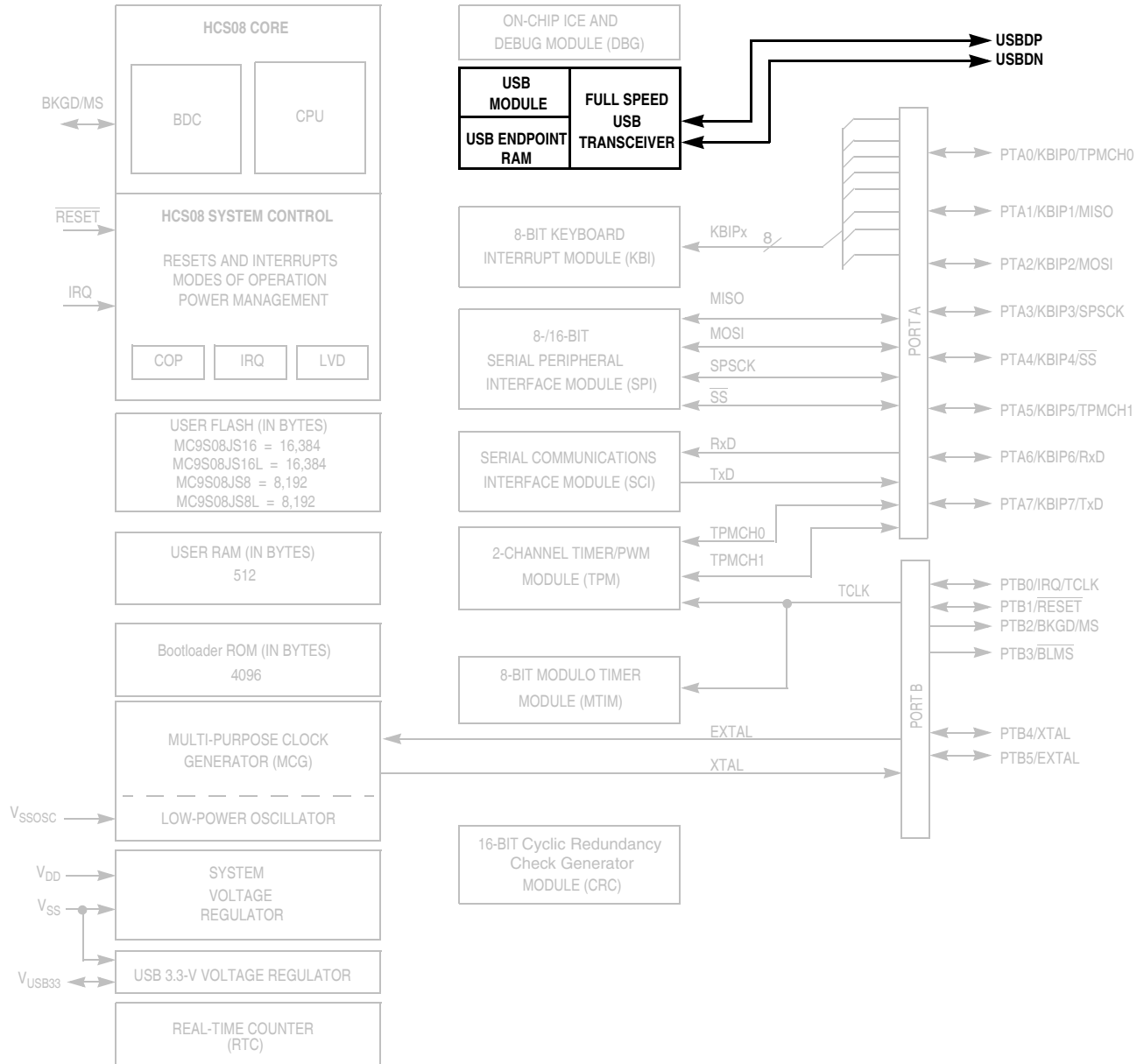
Enabling LVD increases current consumption in stop3. Consequently, disabling LVD before entering stop3 to satisfy USB suspend requirements.

15.1.3 3.3 V Regulator

If using an external 3.3 V regulator as an input to V_{USB33} (only when USBVREN = 0), the supply voltage, V_{DD} , must not fall below the input voltage at the V_{USB33} pin. If using the internal 3.3 V regulator (USBVREN = 1), do not connect an external supply to the V_{USB33} pin. In this case, V_{DD} must fall between 3.9 V and 5.0 V for the internal 3.3 V regulator to operate correctly.

Table 15-1. USBVREN Configuration

USBVREN	3.3 V Regulator	V_{DD} Supply Voltage Range
0	External 3.3 V Regulator (as input to V_{USB33} pin)	$V_{USB33} \leq V_{DD}$ Supply Voltage
1	Internal 3.3 V Regulator (no external supply connected to V_{USB33} pin)	$3.9\text{ V} \leq V_{DD}$ Supply Voltage $\leq 5.0\text{ V}$



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1).
Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD} . IRQ must not be driven above V_{DD} .
4. \overline{RESET} contains integrated pullup device if PTB1 enabled as reset pin function (RSTPE = 1)
5. Pin contains integrated pullup device.
6. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 15-1. MC9S08JS16 Block Diagram Highlighting USB Block and Pins

15.1.4 Features

Features of the USB module include:

- USB 2.0 compliant
 - 12 Mbps full-speed (FS) data rate
 - USB data control logic:
 - Packet identification and decoding/generation
 - CRC generation and checking
 - NRZI (non-return-to-zero inverted) encoding/decoding
 - Bit-stuffing
 - Sync detection
 - End-of-packet detection
- Seven USB endpoints
 - Bidirectional endpoint 0
 - Six unidirectional data endpoints configurable as interrupt, bulk, or isochronous
 - Endpoints 5 and 6 support double-buffering
- USB RAM
 - 256 bytes of buffer RAM shared between system and USB module
 - RAM may be allocated as buffers for USB controller or extra system RAM resource
- USB reset options
 - USB module reset generated by MCU
 - Bus reset generated by the host, which triggers a CPU interrupt
- Suspend and resume operations with remote wakeup support
- Transceiver features
 - Converts USB differential voltages to digital logic signal levels
- On-chip USB pullup resistor
- On-chip 3.3-V regulator

15.1.5 Modes of Operation

Table 15-2. Operating Modes

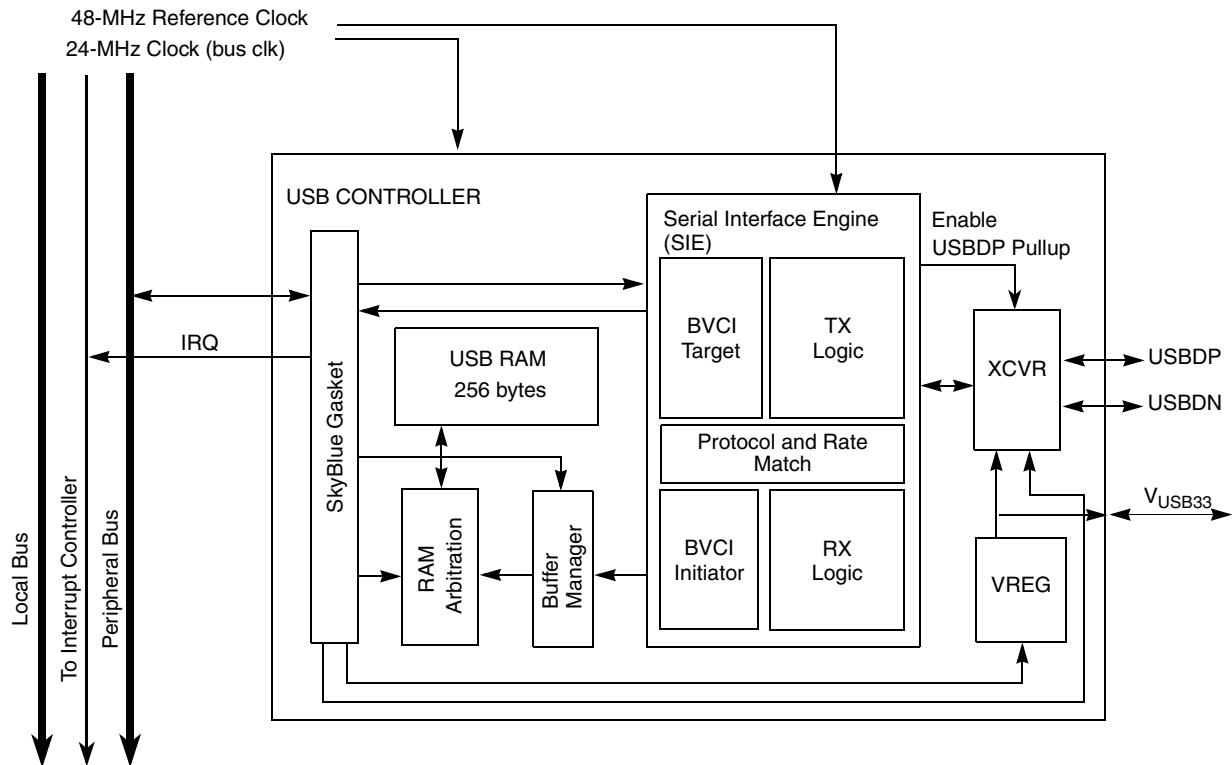
Mode	Description
Stop1	USB module is not functional. Before entering stop1, the internal USB voltage regulator and USB transceiver enter shutdown mode; therefore, the USB voltage regulator and USB transceiver must be disabled by firmware.
Stop2	USB module is not functional. Before entering stop2, the internal USB voltage regulator and USB transceiver enter shutdown mode; therefore, the USB voltage regulator and USB transceiver must be disabled by firmware.

Table 15-2. Operating Modes (continued)

Mode	Description
Stop3	<p>The USB module is optionally available in stop3.</p> <p>A reduced current consumption mode may be required for USB suspend mode per USB Specification Rev. 2.0, and stop3 mode is useful for achieving lower current consumption for the MCU and hence the overall USB device. Before entering stop3 via firmware, the user must ensure that the device settings are configured for stop3 to achieve USB suspend current consumption targets.</p> <p>The USB module is notified about entering suspend mode when the SLEEPF flag is set; this occurs after the USB bus is idle for 3 ms. The device USB suspend mode current consumption level requirements are defined by the USB Specification Rev. 2.0 (500 μA for low-power and 2.5 mA for high-power with remote-wakeup enabled).</p> <p>If USBRESMEN in USBCTL0 is set, and a K-state (resume signaling) is detected on the USB bus, the LPRESF bit in USBCTL0 will be set. This triggers an asynchronous interrupt that will wakeup the MCU from stop3 mode and enable clocks to the USB module. The USBRESMEN bit must then be cleared immediately after stop3 recovery to clear the LPRESF flag bit.</p>
Wait	USB module is operational.

15.1.6 Block Diagram

Figure 15-2 is a block diagram of the USB module.


Figure 15-2. USB Module Block Diagram

15.2 External Signal Description

The USB module requires both data and power pins. The following table describes each of the USB external pin

Name	Port	Direction	Function	Reset State
Positive USB differential signal	USBDP	I/O	Differential USB signaling.	High impedance
Negative USB differential signal	USBDN	I/O	Differential USB signaling.	High impedance
USB voltage regulator power pin	V _{USB33}	Power	3.3 V USB voltage regulator output or 3.3 V USB transceiver/resistor supply input.	—

15.2.1 USBDP

USBDP is the positive USB differential signal. In a USB peripheral application, connect an external $33\ \Omega \pm 1\%$ resistor in series with this signal in order to meet the USB Specification Rev. 2.0 impedance requirement.

15.2.2 USBDN

USBDN is the negative USB differential signal. In a USB peripheral application, connect an external $33\ \Omega \pm 1\%$ resistor in series with this signal in order to meet the USB Specification, Rev. 2.0 impedance requirement.

15.2.3 V_{USB33}

V_{USB33} is connected to the on-chip 3.3-V voltage regulator (VREG). V_{USB33} maintains an output voltage of 3.3 V and can only source enough current for USB internal transceiver (XCVR) and USB pullup resistor. If the VREG is disabled by software, the application must input an external 3.3 V power supply to the USB module via V_{USB33}.

15.3 Register Definition

This section describes the memory map and control/status registers for the USB module.

15.3.1 USB Control Register 0 (USBCTL0)

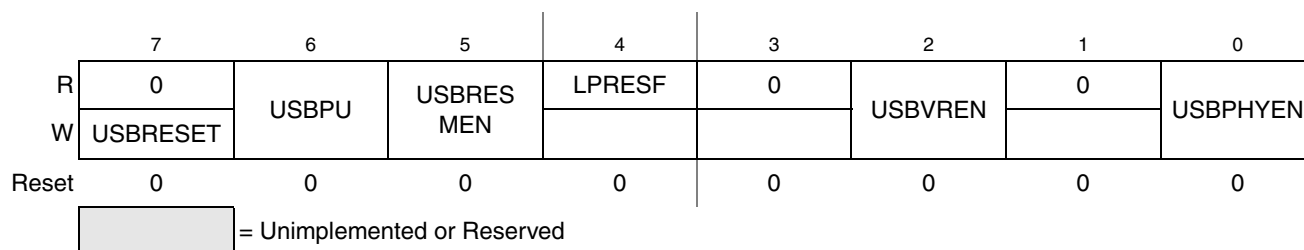


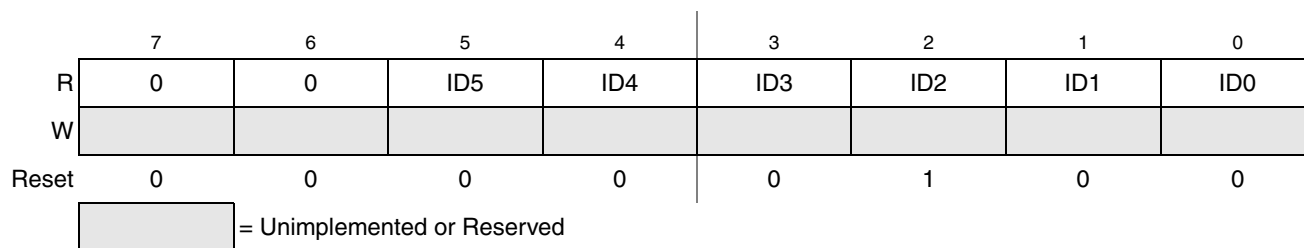
Figure 15-3. USB Transceiver and Regulator Control Register 0 (USBCTL0)

Table 15-3. 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 USB Transceiver and USB voltage regulator). When set to 1, this bit automatically clears when the reset occurs. 0 USB module normal operation 1 Returns the USB module to its reset state
6 USBPU	Pull Up Source — This bit determines the source of the pullup resistor on the USBDP line. 0 Internal USBDP pullup resistor is disabled; The application can use an external pullup resistor 1 Internal USBDP pullup resistor is enabled
5 USBRESMEN	USB Low-Power Resume Event Enable — This bit, when set, enables the USB module to send an asynchronous wakeup interrupt to the MCU upon detection that the LPRESF bit has been set, indicating a K-state on the USB bus. This bit must be set before entering low-power stop3 mode only after SLEEPF=1 (USB is entering suspend mode). It must be cleared immediately after stop3 recovery in order to clear the Low-Power Resume Flag. 0 USB asynchronous wakeup from suspend mode disabled 1 USB asynchronous wakeup from suspend mode enabled
4 LPRESF	Low-Power Resume Flag — This bit becomes set in USB suspend mode if USBRESMEN=1 and a K-state is detected on the USB bus, indicating resume signaling while the device is in a low-power stop3 mode. This flag bit will trigger an asynchronous interrupt, which will wake the device from stop3. Firmware must then clear the USBRESMEN bit in order to clear the LPRESF bit. 0 No K-state detected on the USB bus while the device is in stop3 and the USB is suspended. 1 K-state detected on the USB bus when USBRESMEN=1, the device is in stop3, and the USB is suspended.
2 USBVREN	USB Voltage Regulator Enable — This bit enables the on-chip 3.3V USB voltage regulator. 0 On-chip USB voltage regulator is disabled (OFF MODE) 1 On-chip USB voltage regulator is enabled for active or standby mode
0 USBPHYEN	USB Transceiver Enable — When the USB Transceiver (XCVR) is disabled, USBDP and USBDN are hi-Z. It is recommended that the XCVR be enabled before setting the USBEN bit in the CTL register. The firmware must ensure that the XCVR remains enabled when entering USB SUSPEND mode. 0 On-chip XCVR is disabled 1 On-chip XCVR is enabled

15.3.2 Peripheral ID Register (PERID)

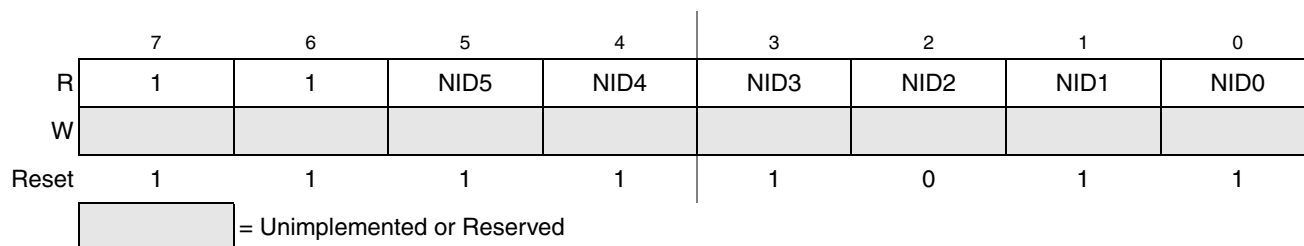
The PERID reads back the value of 0x04. This value is defined for the USB module peripheral.


Figure 15-4. Peripheral ID Register (PERID)
Table 15-4. 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.

15.3.3 Peripheral ID Complement Register (IDCOMP)

The IDCOMP reads back the complement of the peripheral ID register. For the USB module peripheral this will be 0xFB.


Figure 15-5. Peripheral ID Complement Register (IDCOMP)
Table 15-5. IDCOMP Field Descriptions

Field	Description
5:0 NID[5:0]	Compliment ID Number — One's complement version of ID[5:0].

15.3.4 Peripheral Revision Register (REV)

The REV reads back the value of the USB peripheral revision.

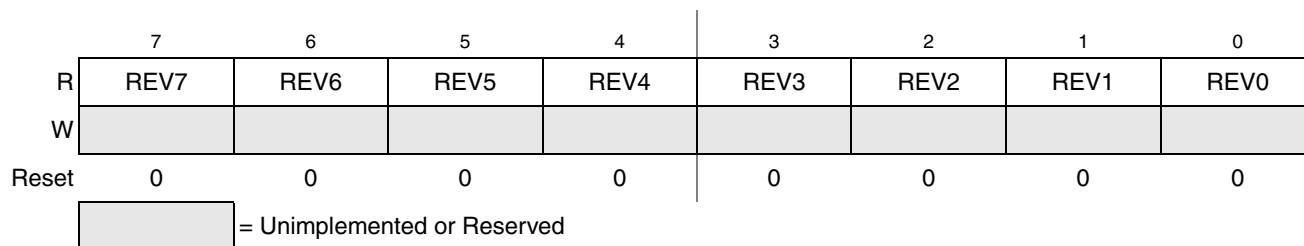
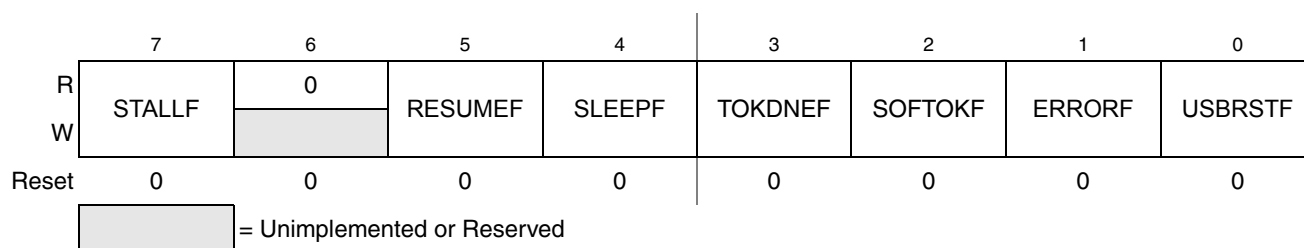

Figure 15-6. Peripheral Revision Register (REV)

Table 15-6. REV Field Descriptions

Field	Description
8–0 REV[7:0]	Revision — Revision number of the USB module.

15.3.5 Interrupt Status Register (INTSTAT)

The INTSTAT contains bits for each of the interrupt source within the USB module. Each of these bits is qualified with its respective interrupt enable bits (see the interrupt enable register). All bits of the register are logically OR'ed together to form a single interrupt source for the microcontroller. Once an interrupt bit has been set, it may only be cleared by writing a 1 to the respective interrupt bit. This register will contain the value of 0x00 after a reset.


Figure 15-8. Interrupt Status Register (INTSTAT)
Table 15-8. INTSTAT Field Descriptions

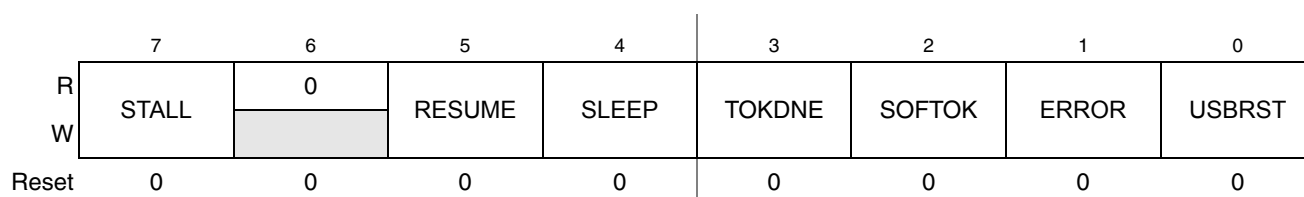
Field	Description
7 STALLF	Stall Flag — The stall interrupt is used in device mode. In device mode the stall flag is asserted when a STALL handshake is sent by the serial interface engine (SIE). 0 A STALL handshake has not been sent 1 A STALL handshake has been sent
5 RESUMEF	Resume Flag — This bit is set 2.5 μ s after clocks to the USB module have restarted following resume signaling. It can be used to indicate remote wakeup signaling on the USB bus. This interrupt is enabled only when the USB module is about to enter suspend mode (usually when SLEEPF interrupt detected). 0 No RESUME observed 1 RESUME detected (K-state is observed on the USBDP/USBDM signals for 2.5 μ s)
4 SLEEPF	Sleep Flag — This bit is set if the USB module has detected a constant idle on the USB bus for 3 ms, indicating that the USB module will go into suspend mode. The sleep timer is reset by activity on the USB bus. 0 No constant idle state of 3 ms has been detected on the USB bus 1 A constant idle state of 3 ms has been detected on the USB bus
3 TOKDNEF	Token Complete Flag — This bit is set when the current transaction is completed. The firmware must immediately read the STAT register to determine the endpoint and BD information. Clearing this bit (by setting it to 1) causes the STAT register to be cleared or the STAT FIFO holding register to be loaded into the STAT register. 0 No tokens being processed are complete 1 Current token being processed is complete
2 SOFTOKF	SOF Token Flag — This bit is set if the USB module has received a start of frame (SOF) token. 0 The USB module has not received an SOF token 1 The USB module has received an SOF token

Table 15-8. INTSTAT Field Descriptions (continued)

Field	Description
1 ERRORF	Error Flag — This bit is set when any of the error conditions within the ERRSTAT register has occurred. The firmware must then read the ERRSTAT register to determine the source of the error. 0 No error conditions within the ERRSTAT register have been detected 1 Error conditions within the ERRSTAT register have been detected
0 USBRSTF	USB Reset Flag — This bit is set when the USB module has decoded a valid USB reset. When asserted, this bit will inform the MCU to automatically write 0x00 to the address register and to enable endpoint 0. USBRSTF is set once a USB reset has been detected for 2.5 μ s. 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

15.3.6 Interrupt Enable Register (INTENB)

The INTENB contains enabling bits for each of the interrupt sources within the USB module. Setting any of these bits will enable the respective interrupt source in the INTSTAT register. This register will contain the value of 0x00 after a reset, i.e. all interrupts disabled.


Figure 15-9. Interrupt Enable Register (INTENB)
Table 15-9. INTENB Field Descriptions

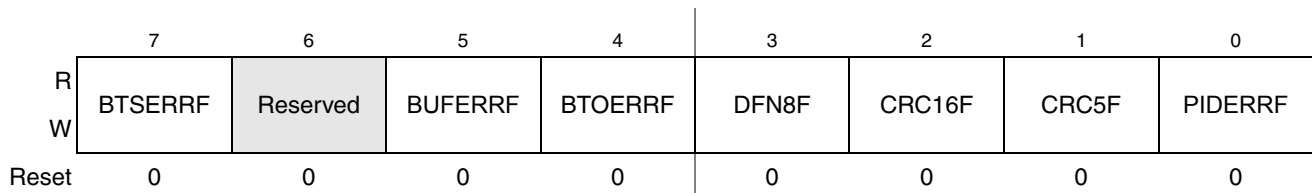
Field	Description
7 STALL	STALL Interrupt Enable — Setting this bit will enable STALL interrupts. 0 Interrupt disabled 1 Interrupt enabled
5 RESUME	RESUME Interrupt Enable — Setting this bit will enable RESUME interrupts. 0 Interrupt disabled 1 Interrupt enabled
4 SLEEP	SLEEP Interrupt Enable — Setting this bit will enable SLEEP interrupts. 0 Interrupt disabled 1 Interrupt enabled
3 TOKDNE	TOKDNE Interrupt Enable — Setting this bit will enable TOKDNE interrupts. 0 Interrupt disabled 1 Interrupt enabled
2 SOFTOK	SOFTOK Interrupt Enable — Setting this bit will enable SOFTOK interrupts. 0 Interrupt disabled 1 Interrupt enabled

Table 15-9. INTENB Field Descriptions (continued)

Field	Description
1 ERROR	ERROR Interrupt Enable — Setting this bit will enable ERROR interrupts. 0 Interrupt disabled 1 Interrupt enabled
0 USBRST	USBRST Interrupt Enable — Setting this bit will enable USBRST interrupts. 0 Interrupt disabled 1 Interrupt enabled

15.3.7 Error Interrupt Status Register (ERRSTAT)

The ERRSTAT contains bits for each of the error sources within the USB module. Each of these bits corresponds to its respective error enable bit (See [Section 15.3.8, “Error Interrupt Enable Register \(ERREN B\)”](#).) The result is OR'ed together and sent to the ERROR bit of the INTSTAT register. Once an interrupt bit has been set, it may only be cleared by writing a 1 to the corresponding flag bit. Each bit is set as soon as the error condition is detected. Thus, the interrupt will typically not correspond with the end of a token being processed. This register will contain the value of 0x00 after reset.

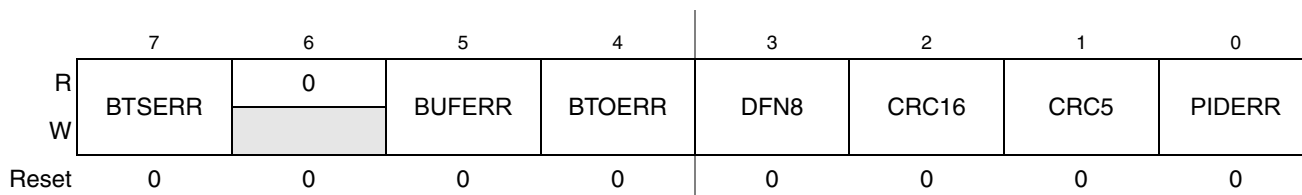

Figure 15-10. Error Interrupt Status Register (ERRSTAT)
Table 15-10. ERRSTAT Field Descriptions

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
4 BTOERRF	Bus Turnaround Error Timeout Flag — This bit is set if a bus turnaround timeout error has occurred. The USB module uses a bus turnaround timer to keep track of the amount of time elapsed between the token and data phases of a SETUP or OUT TOKEN or the data and handshake phases of an IN TOKEN. If more than 16-bit times are counted from the previous EOP before a transition from IDLE, a bus turnaround timeout error will occur. 0 No bus turnaround timeout error has been detected 1 A bus turnaround timeout error has occurred

Table 15-10. ERRSTAT Field Descriptions (continued)

Field	Description
3 DFN8F	Data Field Error Flag — The data field received was not an interval of 8 bits. The USB Specification specifies that the data field must be an integer number of bytes. If the data field was not an integer number of bytes, this bit will be set. 0 The data field was an integer number of bytes 1 The data field was not an integer number of bytes
2 CRC16F	CRC16 Error Flag — The CRC16 failed. If set, the data packet was rejected due to a CRC16 error. 0 No CRC16 error detected 1 CRC16 error detected
1 CRC5F	CRC5 Error Flag — This bit will detect a CRC5 error in the token packets generated by the host. If set, the token packet was rejected due to a CRC5 error. 0 No CRC5 error detected 1 CRC5 error detected, and the token packet was rejected.
0 PIDERRF	PID Error Flag — The PID check failed. 0 No PID check error detected 1 PID check error detected

15.3.8 Error Interrupt Enable Register (ERRENB)


Figure 15-11. Error Interrupt Enable Register (ERRENB)
Table 15-11. ERRSTAT Field Descriptions

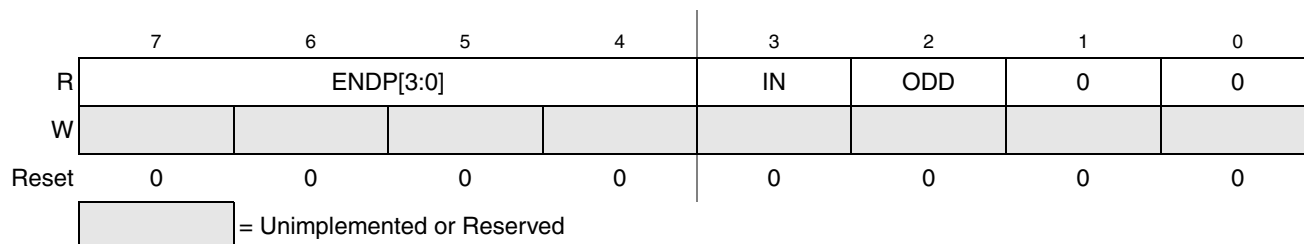
Field	Description
7 BTSERR	BTSERR Interrupt Enable — Setting this bit will enable BTSERR interrupts. 0 Interrupt disabled 1 Interrupt enabled
5 BUFERR	BUFERR Interrupt Enable — Setting this bit will enable BUFERR interrupts. 0 Interrupt disabled 1 Interrupt enabled
4 BTOERR	BTOERR Interrupt Enable — Setting this bit will enable BTOERR interrupts. 0 Interrupt disabled 1 Interrupt enabled
3 DFN8	DFN8 Interrupt Enable — Setting this bit will enable DFN8 interrupts. 0 Interrupt disabled 1 Interrupt enabled
2 CRC16	CRC16 Interrupt Enable — Setting this bit will enable CRC16 interrupts. 0 Interrupt disabled 1 Interrupt enabled

Table 15-11. ERRSTAT Field Descriptions (continued)

Field	Description
1 CRC5	CRC5 Interrupt Enable — Setting this bit will enable CRC5 interrupts. 0 Interrupt disabled 1 Interrupt enabled
0 PIDERR	PIDERR Interrupt Enable — Setting this bit will enable PIDERR interrupts. 0 Interrupt disabled 1 Interrupt enabled

15.3.9 Status Register (STAT)

The STAT reports the transaction status within the USB module. When the MCU receives a TOKDNE interrupt, the STAT is read to determine the status of the previous endpoint communication. The data in the status register is valid only when the TOKDNEF interrupt flag is asserted. The STAT register is actually a read window into a status FIFO maintained by the USB module. When the USB module uses a BD, it updates the status register. If another USB transaction is performed before the TOKDNE interrupt is serviced, the USB module will store the status of the next transaction in the STAT FIFO. Thus, the STAT register is actually a four byte FIFO which allows the microcontroller to process one transaction while the serial interface engine (SIE) is processing the next. Clearing the TOKDNEF bit in the INTSTAT register causes the SIE to update the STAT register with the contents of the next STAT value. If the next data in the STAT FIFO holding register is valid, the SIE will immediately reassert the TOKDNE interrupt.


Figure 15-12. Status Register (STAT)
Table 15-12. STAT Field Descriptions

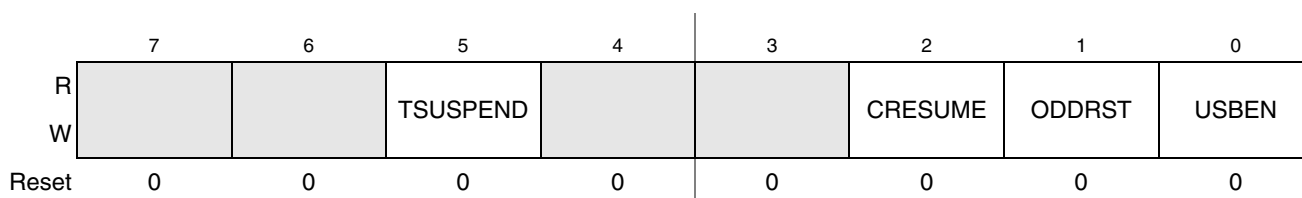
Field	Description
7–4 ENDP[3:0]	Endpoint Number — These four bits encode the endpoint address that received or transmitted the previous token. This allows the microcontroller to determine which BDT entry was updated by the last USB transaction. 0000 Endpoint 0 0001 Endpoint 1 0010 Endpoint 2 0011 Endpoint 3 0100 Endpoint 4 0101 Endpoint 5 0110 Endpoint 6

Table 15-12. STAT Field Descriptions (continued)

Field	Description
3 IN	In/Out Transaction — This bit indicates whether the last BDT updated was for a transmit (IN) transfer or a receive (OUT) data transfer. 0 Last transaction was a receive (OUT) data transfer 1 Last BDT updated was for transmit (IN) transfer
2 ODD	Odd/Even Transaction — This bit indicates whether the last buffer descriptor updated was in the odd bank of the BDT or the even bank of the BDT, See earlier section for more information on BDT address generation. 0 Last buffer descriptor updated was in the EVEN bank 1 Last buffer descriptor updated was in the ODD bank

15.3.10 Control Register (CTL)

The CTL provides various control and configuration information for the USB module.


Figure 15-13. Control Register (CTL)
Table 15-13. 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 and then clear it to 0. 0 Do not execute remote wakeup 1 Execute resume signaling - remote wakeup
1 ODDRST	Odd Reset — Setting this bit will reset all the buffer descriptor ODD ping-pong bits to 0 which will then specify the EVEN descriptor bank. This bit is used with double-buffered endpoints 5 and 6. This bit has no effect on endpoints 0 through 4. 0 Do not reset 1 Reset all the buffer descriptor ODD ping/pong bits to 0 which will then specify the EVEN descriptor bank
0 USBEN	USB Enable Setting this bit will enable the USB module to operate. Setting this bit causes the SIE to reset all of its ODD bits to the BDTs. Thus, setting this bit will reset much of the logic in the SIE. 0 Disable the USB module 1 Enable the USB module for operation, will not affect Transceiver and VREG.

15.3.11 Address Register (ADDR)

The ADDR register contains the unique 7-bit address the device will be recognized as through USB. The register is reset to 0x00 after the reset input has gone active or the USB module has decoded USB reset signaling. That will initialize the address register to decode address 0x00 as required by the USB specification. Firmware will change the value when it processes a SET_ADDRESS request.

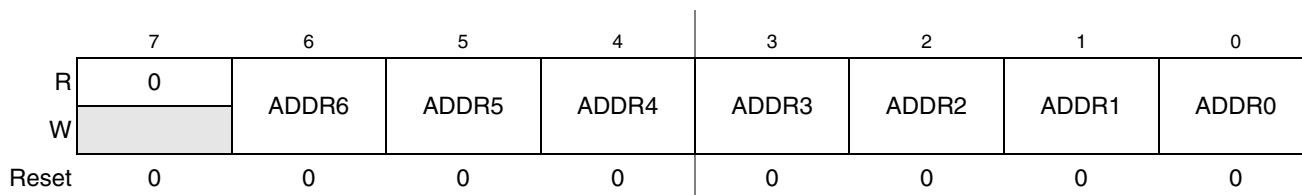


Figure 15-14. Address Register (ADDR)

Table 15-14. 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

15.3.12 Frame Number Register (FRMNUML, FRMNUMH)

The frame number registers contains the 11-bit frame number. The frame number registers require two 8-bit registers to implement. The low order byte is contained in FRMNUML, and the high order byte is contained in FRMNUMH. These registers are updated with the current frame number whenever a SOF TOKEN is received.

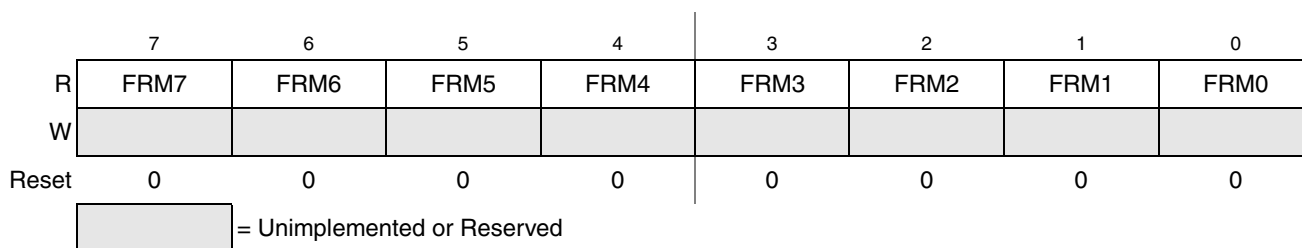
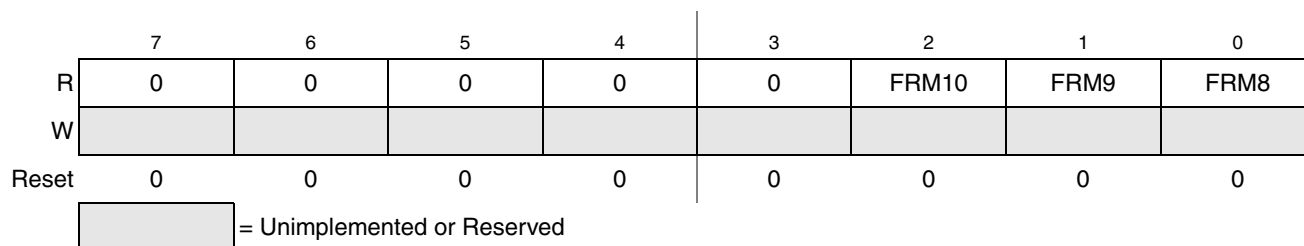


Figure 15-15. Frame Number Register Low (FRMNUML)

Table 15-15. FRMNUML Field Descriptions

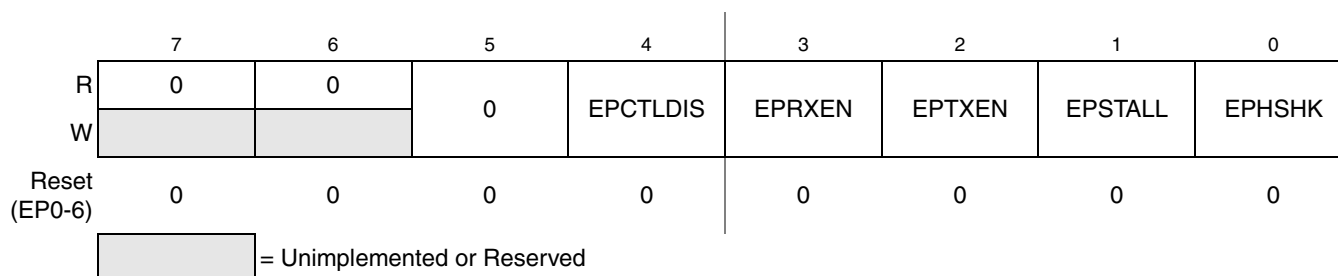
Field	Description
7–0 FRM[7:0]	Frame Number — These bits represent the low order bits of the 11 bit frame number.


Figure 15-16. Frame Number Register High (FRMNUMH)
Table 15-16. 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.

15.3.13 Endpoint Control Register (EPCTLn, n=0-6)

The endpoint control registers contains the endpoint control bits (EPCTLDIS, EPRXEN, EPTXEN, and EPHSHK) for each endpoint available within the USB module for a decoded address. These four bits define all of the control necessary for any one endpoint. The formats for these registers are shown in the tables below. Endpoint 0 (ENDP0) is associated with control pipe 0 which is required by the USB for all functions. Therefore, after a USBRST interrupt has been received, the microcontroller must set EPCTL0 to contain 0x0D.


Figure 15-17. Endpoint Control Register (EPCTLn)
Table 15-17. 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 15-18 .
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 15-18 .
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 15-18 .

Table 15-17. EPCTLn Field Descriptions (continued)

Field	Description
1 EPSTALL	Endpoint Stall — When set, this bit indicates that the endpoint is stalled. This bit has priority over all other control bits in the endpoint control register, but is only valid if EPTXEN=1 or EPRXEN=1. Any access to this endpoint will cause the USB module to return a STALL handshake. Once an endpoint is stalled it requires intervention from the host controller. 0 Endpoint n is not stalled 1 Endpoint n is stalled
0 EPHSHK	Endpoint Handshake — This bit determines if the endpoint will perform handshaking during a transaction to the endpoint. This bit will generally be set unless the endpoint is isochronous. 0 No handshaking performed during a transaction to this endpoint (usually for isochronous endpoints) 1 Handshaking performed during a transaction to this endpoint

Table 15-18. 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	RESERVED

15.4 Functional Description

This section describes the functional behavior of the USB module. It documents data packet processing for endpoint 0 and data endpoints, USB suspend and resume states, SOF token processing, reset conditions and interrupts.

15.4.1 Block Descriptions

Figure 15-2 is the block diagram. The module's sub-blocks and external signals are described in the following sections. The module involves several major blocks — USB transceiver (XCVR), USB serial interface engine (SIE), a 3.3 V regulator (VREG), endpoint buffer manager, shared RAM arbitration, USB RAM and the SkyBlue gasket.

15.4.1.1 USB Serial Interface Engine (SIE)

The SIE is composed of two major functions: TX Logic and RX Logic. These major functions are described below in more detail. The TX and RX logic are connected by a USB protocol engine which manages packet flow to and from the USB module. The SIE is connected to the rest of the system via

internal basic virtual component interface (BVCI) compliant target and initiator buses. The BVCI target interface is used to configure the USB SIE and to provide status and interrupts to CPU. The BVCI initiator interface provides the integrated DMA controller access to the buffer descriptor table (BDT), and transfers USB data to or from USB RAM memory.

15.4.1.1.1 Serial Interface Engine (SIE) Transmitter Logic

The SIE transmitter logic has two primary functions. The first is to format the USB data packets that have been stored in the endpoint buffers. The second is to transmit data packets via the USB transceiver.

All of the necessary USB data formatting is performed by the SIE transmitter logic, including:

- NRZI encoding
- bit-stuffing
- CRC computation
- addition of the SYNC field
- addition of the End-of-packet (EOP)

The CPU typically places data in the endpoint buffers as part of the application. When the buffer is configured as an IN buffer and the USB host requests a packet, the SIE responds with a properly formatted data packet.

The transmitter logic is also used to generate responses to packets received from the USB host. When a properly formatted packet is received from the USB host, the transmitter logic responds with the appropriate ACK, NAK or STALL handshake.

When the SIE transmitter logic is transmitting data from the buffer space for a particular endpoint, CPU access to that endpoint buffer space is not recommended.

15.4.1.1.2 Serial Interface Engine (SIE) Receiver Logic

The SIE receiver logic receives USB data and stores USB packets in USB RAM for processing by the CPU and the application software. Serial data from the transceiver is converted to a byte-wide parallel data stream, checked for proper packet framing, and stored in the USB RAM memory.

Received bitstream processing includes the following operations:

- decodes an NRZI USB serial data stream
- Sync detection
- Bit-stuff removal (and error detection)
- End-of-packet (EOP) detection
- CRC validation
- PID check
- other USB protocol layer checks.

The SIE receiver logic provides error detection including:

- Bad CRC
- Timeout detection for EOP

- Bit stuffing violation

If a properly formatted packet is received, the receiver logic initiates a handshake response to the host. If the packet is not decoded correctly due to bit stuff violation, CRC error or other packet level problem, the receiver ignores it. The USB host will eventually time-out waiting for a response, and retransmit the packet.

When the SIE receiver logic is receiving data in the buffer space for a particular endpoint, CPU access to that buffer space is not recommended.

15.4.1.2 MCU/Memory Interfaces

15.4.1.2.1 SkyBlue Gasket

The SkyBlue gasket connects the USB module to the SoC internal peripheral bus. The gasket maps accesses to the endpoint buffer descriptors or the endpoint buffers into the shared RAM block, and it also maps accesses to the peripherals register set into the serial interface engine (SIE) register space. The SkyBlue gasket interface includes registers to control the USB transceiver and voltage regulator.

15.4.1.2.2 Endpoint Buffer Manager

Each endpoint supported by the USB device transmits data to and from buffers stored in the shared buffer memory. The serial interface engine (SIE) uses a table of descriptors, the Buffer Descriptor Table (BDT), which is also stored in the USB RAM to describe the characteristics of each endpoint. The endpoint buffer manager is responsible for mapping requests to access endpoint buffer descriptors into physical addresses within the USB RAM block.

15.4.1.2.3 RAM Arbitration

The arbitration block allows access to the USB RAM block from the SkyBlue gasket block and from the SIE.

15.4.1.3 USB RAM

The USB module includes 256 bytes of high speed RAM, accessible by the USB serial interface engine (SIE) and the CPU. The USB RAM runs at twice the speed of the bus clock to allow interleaved non-blocked access by the CPU and SIE. The USB RAM is used for storage of the buffer descriptor table (BDT) and endpoint buffers. USB RAM that is not allocated for the BDT and endpoint buffers can be used as system memory. If the USB module is not enabled, then the entire USB RAM may be used as unsecured system memory.

15.4.1.4 USB Transceiver (XCVR)

The USB transceiver is electrically compliant to the *Universal Serial Bus Specification 2.0*. This block provides the necessary 2-wire differential NRZI signaling for USB communication. The transceiver is on-chip to provide a cost effective single chip USB peripheral solution.

15.4.1.5 USB On-Chip Voltage Regulator (VREG)

The on-chip 3.3-V regulator provides a stable power source to power the USB internal transceiver and provide for the termination of an internal or external pullup resistor. When the on-chip regulator is enabled, it requires a voltage supply input in the range from 3.9 V to 5.5 V, and the voltage regulator output will be in the range of 3.0 V to 3.6 V.

With a dedicated on-chip USB 3.3-V regulator and a separate power supply for the MCU, the MCU and USB can operate at different voltages (See the USB electricals regarding the USB voltage regulator electrical characteristics). When the on-chip 3.3-V regulator is disabled, a 3.3-V source must be provided through the V_{USB33} pin to power the USB transceiver. In this case, the power supply voltage to the MCU must not fall below the input voltage at the V_{USB33} pin.

The 3.3-V regulator has 3 modes including:

- Active mode — This mode is entered when USB is active. Current requirement is sufficient to power the transceiver and the USBDP pullup resistor.
- Standby — The voltage regulator standby mode is entered automatically when the USB device is in suspend mode. When the USB device is forced into suspend mode by the USB bus, the firmware must configure the MCU for stop3 mode. In standby mode, the requirement is to maintain the USBDP pin voltage at 3.0 V to 3.6 V, with a 900 Ω (worst-case) pullup.
- Power off — This mode is entered anytime when stop2 or stop1 is entered or when the voltage regulator is disabled.

15.4.1.6 USB On-Chip USBDP Pullup Resistor

The pullup resistor on the USBDP line required for full-speed operation by the USB Specification Rev. 2.0 can be internal or external to the MCU, depending on the application requirements. An on-chip pullup resistor, implemented as specified in the USB 2.0 resistor ECN, is optionally available via firmware configuration. Alternatively, this on-chip pullup resistor can be disabled, and the USB module can be configured to use an external pullup resistor for the USBDP line instead. If using an external pullup resistor on the USBDP line, the resistor must comply with the requirements in the USB 2.0 resistor ECN found at <http://www.usb.org>.

The USBPU bit in the USBCTL0 register can be used to indicate if the pullup resistor is internal or external to the MCU. If USBPU is clear, the internal pullup resistor on USBDP is disabled, and an external USBDP pullup can be used. When using an external USBDP pullup, if the voltage regulator is enabled, the V_{USB33} voltage output can be used with the USBDP pullup. While the use of the internal USBDP pullup resistor is generally recommended, the figure below shows the USBDP pullup resistor configuration for a USB device using an external resistor tied to V_{USB33} .

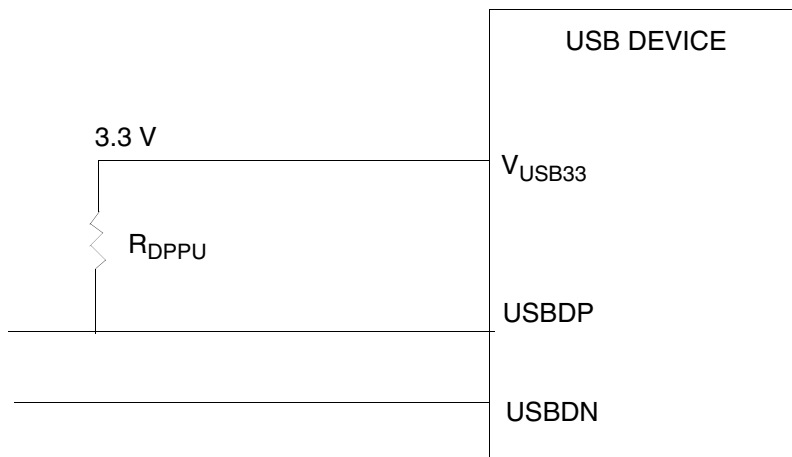


Figure 15-18. USBDP/USB DN Pullup Resistor Configuration for USB module

15.4.1.7 USB Powering and USB DP Pullup Enable Options

The USB module provides a single-chip solution for USB device applications that are self-powered or bus-powered. The USB device needs to know when it has a valid USB connection in order to enable or disable the pullup resistor on the USB DP line. For the USB module on this device, the pullup on USB DP is only applied when a valid VBUS connection is sensed, as required by the USB specification.

In bus-powered applications, system power must be derived from VBUS. Because VBUS is only available when a valid USB connection from host to device is made, the VBUS sensing is built-in, and the USB DP pullup can be enabled accordingly.

With self-powered applications, determining when a valid USB connection is made is different from that of bus-powered applications. In self-powered applications, VBUS sensing must be built into the application. For instance, a KBI pin interrupt can be utilized (if available). When a valid VBUS connection is made, the KBI interrupt can notify the application that a valid USB connection is available, and the internal pullup resistor can be enabled using the USBPU bit. If an external pullup resistor is used instead of the internal one, the VBUS sensing mechanism must be included in the system design.

Table 15-19 summarizes the differences in enabling the USB DP pullup for different USB power modes.

Table 15-19. USB DP Pullup Enable for Different USB Power Modes

Power	USB DP Pullup	Pullup Enable
Bus Power (Built-in VBUS sense)	Internal	Set USBPU bit
	External	Build into application
Self Power (Build VBUS sense into application)	Internal	Set USBPU bit
	External	Build into application

15.4.2 Buffer Descriptor Table (BDT)

To efficiently manage USB endpoint communications, the USB module implements a buffer descriptor table (BDT) comprised of buffer descriptors (BD) in the local USB RAM. The BD entries provide status or control information for a corresponding endpoint. The BD entries also provide an address to the endpoint's buffer. A single BD for an endpoint direction requires 3-bytes. A detailed description of the BDT format is provided in the next sections.

The software API intelligently manages buffers for the USB module by updating the BDT when needed. This allows the USB module to efficiently handle data transmission and reception, while the microcontroller performs communication overhead processing and other function dependent applications.

Because the buffers are shared between the microcontroller and the USB module, a simple semaphore mechanism is used to distinguish who is allowed to update the BDT and buffers in buffer memory. A semaphore bit, the OWN bit, is cleared to 0 when the BD entry is owned by the microcontroller. The microcontroller is allowed read and write access to the BD entry and the data buffer when the OWN bit is 0. When the OWN bit is set to 1, the BD entry and the data buffer are owned by the USB module. The USB module now has full read and write access and the microcontroller must not modify the BD or its corresponding data buffer.

15.4.2.1 Multiple Buffer Descriptor Table Entries for a Single Endpoint

Every endpoint direction requires at least one three-byte Buffer Descriptor entry. Thus, endpoint 0, a bidirectional control endpoint, requires one BDT entry for the IN direction, and one for the OUT direction.

Using two BD entries also allows for double-buffering. Double-buffering BDs allows the USB module to easily transfer data at the maximum throughput provided by the USB module. Double buffering allows the MCU to process one BD while the USB module is processing the other BD.

To facilitate double-buffering, two buffer descriptor (BD) entries are needed for each endpoint direction. One BD entry is the EVEN BD and the other is the ODD BD.

15.4.2.2 Addressing Buffer Descriptor Table Entries

The BDT addressing is hardwired into the module. The BDT occupies the first portion of the USB RAM. To access endpoint data via the USB or MCU, the addressing mechanism of the buffer descriptor table must be understood.

All enabled IN and OUT endpoint BD entries are indexed into the BDT to allow easy access via the USB module or the MCU. The figure below shows the USB RAM organization. The figure shows that the first entries in the USB RAM are dedicated to storage of the BDT entries - i.e. the first 30 bytes of the USB RAM (0x00 to 0x1D) are used to implement the BDT.

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
		Endpoint 6, Buffer ODD
0x1D		
0x1E	RESERVED	
0x1F	RESERVED	
0x20	USB RAM available for endpoint buffers	
0xFF		

When the USB module receives a USB token on an enabled endpoint, it interrogates the BDT. The USB module reads the corresponding endpoint BD entry and determines if it owns the BD and corresponding data buffer.

15.4.2.3 Buffer Descriptor Formats

The buffer descriptors (BDs) are groups of registers that provide endpoint buffer control information for the USB module and the MCU. The BDs have different meanings based on who is reading the BD in memory.

The USB module uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- Release Own upon packet completion
- Data toggle synchronization enable
- How much data to be transmitted or received
- Where the buffer resides in the buffer RAM.

The microcontroller uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- The received TOKEN PID
- How much data was transmitted or received.
- Where the buffer resides in buffer memory

The BDT is composed of buffer descriptors (BD) which are used to define and control the actual buffers in the USB RAM space. BDs always occur as a 3-bytes block. See [Figure 15-19](#) for the BD example of Endpoint 0 IN start from USB RAM offset 0x00.

The format for the buffer descriptor is shown in [Table 15-20](#).

Offset		7	6	5	4	3	2	1	0	
0x00	R	OWN	DATA0/1	BDTKPID[3]	BDTKPID[3]	BDTKPID[1]	BDTKPID[0]	0	0	
	W			0	0	DTS	BDTSTALL			
0x01	R	BC[7:0]								
	W									
0x02	R	EPADR[9:4]								
	W									

Figure 15-19. Buffer Descriptor Example

Table 15-20. Buffer Descriptor Table Fields

Field	Description
OWN	<p>OWN — This OWN bit determines who currently owns the buffer. The USB SIE generally writes a 0 to this bit when it has completed a token. The USB module ignores all other fields in the BD when OWN=0. Once the BD has been assigned to the USB module (OWN=1), the MCU must not change it in any way. This byte of the BD must always be the last byte the MCU (firmware) updates when it initializes a BD. Although the hardware will not block the MCU from accessing the BD while owned by the USB SIE, doing so may cause undefined behavior and is generally not recommended.</p> <p>0 The MCU has exclusive access to the entire BD 1 The USB module has exclusive access to the BD</p>
DATA0/1	<p>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.</p> <p>0 Data 0 packet 1 Data 1 packet</p>
BDTKPID[3:0]	<p>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.</p>
DTS	<p>Data Toggle Synchronization— This bit enables data toggle synchronization.</p> <p>0 No data toggle synchronization is performed. 1 Data toggle synchronization is performed.</p>

Table 15-20. Buffer Descriptor Table Fields (continued)

Field	Description
BDTSTALL	<p>BDT Stall — Setting this bit will cause the USB module to issue a STALL handshake if a token is received by the SIE that would use the BDT in this location. The BDT is not consumed by the SIE (the OWN bit remains and the rest of the BD is unchanged) when the BDTSTALL bit is set.</p> <p>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</p>
BC[7:0]	<p>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.</p>
EPADR[9:4]	<p>Endpoint Address— The endpoint address bits represent the upper 6 bits of the 10-bit buffer address within the module's local USB RAM. Bits [3:0] of EPADR are always zero, therefore the address of the buffer must always start on a 16-byte aligned address within the local RAM. These bits are unchanged by the USB module. This is NOT the address of the memory on the system bus. EPADR is relative to the start of the local USB RAM.</p>

15.4.3 USB Transactions

When the USB module transmits or receives data, it will first compute the BDT address based on the endpoint number, data direction, and which buffer is being used (even or odd), then it will read the BD.

Once the BD has been read, and if the OWN bit equals 1, the serial interface engine (SIE) will transfer the packet data to or receive the packet data from the buffer pointed to by the EPADR field of the BD. When the USB TOKEN is complete, the USB module will update the BDT and change the OWN bit to 0.

The STAT register is updated and the TOKDNE interrupt is set. When the microcontroller processes the TOKDNE interrupt, it reads the status register. This gives the microcontroller all the information it needs to process the endpoint. At this point the microcontroller can allocate a new BD, so additional USB data can be transmitted or received for that endpoint, and it can process the previous BD. [Figure 15-20](#) shows a timeline for how a typical USB token would be processed.

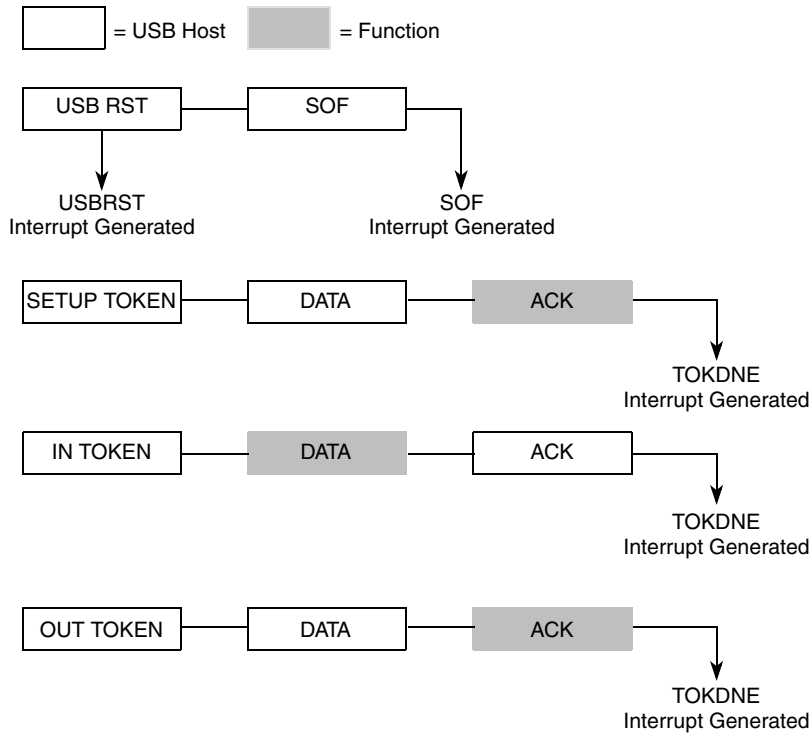


Figure 15-20. USB Packet Flow

The USB has two sources of data overrun error:

- The memory latency to the local USB RAM interface may be too high and cause the receive buffer to overflow. This is predominantly a hardware performance issue, usually caused by transient memory access issues.
- The packet received may be larger than the negotiated MAXPACKET size. This is caused by a software bug.

In the first case, the USB will respond with a NAK or bus timeout (BTO) as appropriate for the class of transaction. The BTOERR bit will be set in the ERRSTAT register. Depending on the values of the INTENB and ERRENB register, USB module may assert an interrupt to notify the CPU of the error. In device mode the BDT is not written back nor is the TOKDNE interrupt triggered because it is assumed that a second attempt will be queued at future time and will succeed.

In the second case of oversized data packets, the USB specification assumes correct software drivers on both sides. The overrun is not due to memory latency but to a lack of space to put the excess data. NAK'ing the packet will likely cause another retransmission of the already oversized packet data. In response to oversized packets, the USB module will still ACK the packet for non-isochronous transfers. The data written to memory is clipped to the MAXPACKET size so as not to corrupt the buffer space. The USB module will assert the BUFERRF bit of the ERRSTAT register (which could trigger an interrupt, as above) and a TOKDNE interrupt fails. The BDTKPID field of the BDT will not be "1111" because the BUFERRF is *not* due to latency. The packet length field written back to the BDT will be the MAXPACKET value to represent the length of the clipped data actually written to memory. From here the software can decide an

appropriate course of action for future transactions — stalling the endpoint, canceling the transfer, disabling the endpoint, etc.

15.4.4 USB Packet Processing

Packet processing for a USB device consists of managing buffers for IN (to the USB Host) and OUT (to the USB device) transactions. Packet processing is further divided into request processing on Endpoint 0, and data packet processing on the data endpoints.

15.4.4.1 USB Data Pipe Processing

Data pipe processing is essentially a buffer management task. The firmware is responsible for managing the shared buffer RAM to ensure that a BD is always ready for the hardware to process (OWN bit = 1).

The device allocates buffers within the shared RAM, sets up the buffer descriptors, and waits for interrupts. On receipt of a TOKDNE interrupt, the firmware reads the STAT register to determine which endpoint is affected, then reads the corresponding BDT entry to determine what to do next.

When processing data packets, firmware is responsible for managing the size of the packet buffers to be in compliance with the USB specification, and the physical limitations of this module. Packet sizes up to 64 bytes are supported on all endpoints. Isochronous endpoints also can only specify packet sizes up to 64 bytes.

Firmware is also responsible for setting the appropriate bits in the BDT. For most applications using bulk packets (control, bulk, and interrupt-type transfers), the firmware will set the DTS, BC and EPADR fields for each BD. For isochronous packets, firmware will set BC and EPADR fields. In all cases, firmware will set the OWN bit to enable the endpoint for data transfers.

15.4.4.2 Request Processing on Endpoint 0

In most cases, commands to the USB device are directed to Endpoint 0. The host uses the “Standard Requests” described in Chapter 9 of the USB specification to enumerate and configure the device. Class drivers or product specific drivers running on the host send class (HID, Mass Storage, Imaging) and vendor specific commands to the device on endpoint 0.

USB requests always follow a specific format:

- Host sends a SETUP token, followed by an 8-byte setup packet, and the device hardware can send a handshake packet.
- If the setup packet specifies a data phase, the host and device may transfer up to 64 Kbytes of data (either IN or OUT, not both).
- The request is terminated by a status phase.

Device firmware monitors the INTSTAT and STAT registers, the endpoint 0 buffer descriptors (BD's), and the contents of the setup packet to correctly execute the host's request.

The flow for processing endpoint 0 requests is as follows:

1. Allocate 8-byte buffers for endpoint 0 OUT.

2. Create BDT entries for Endpoint 0 OUT, and set the DTS and OWN bits to 1.
3. Wait for interrupt TOKDNE.
4. Read STAT register.
 - The status register must show Endpoint 0, RX. If it does not, then assert the EPSTALL bit in the endpoint control register.
5. Read Endpoint 0 OUT BD.
 - Verify that the token type is a SETUP token. If it is not, then assert the EPSTALL bit in the endpoint control register.
6. Decode and process the setup packet.
 - If the direction field in the setup packet indicates an OUT transfer, then process the out data phase to receive exactly the number of bytes specified in the wLength field of the setup packet.
 - If the direction field in the setup packet indicates an IN transfer, then process the in data phase to deliver no more than the number of bytes specified in the wLength field. Note that it is common for the host to request more bytes than it needs, expecting the device to only send as much as it needs to.
7. After processing the data phase (if there was one), create a zero-byte status phase transaction.
 - This is accomplished for an OUT data phase (IN status phase) by setting the BC to 0 in the next BD, while also setting OWN=1. For an IN data phase (OUT status phase), the host will send a zero-byte packet to the device.
 - Firmware can verify completion of the data phase by verifying the received token in the BD on receipt of the TOKDNE interrupt. If the data phase was of type IN, then the status phase token will be OUT. If the data phase was of type OUT, then the status phase token will be IN.

15.4.4.3 Endpoint 0 Exception Conditions

The USB includes a number of error checking and recovery mechanisms to ensure reliable data transfer. One such exception occurs when the host sends a SETUP packet to a device, and the host never receives the acknowledge handshake from the device. In this case, the host will retry the SETUP packet.

Endpoint 0 request handlers on the device must be aware of the possibility that after receiving a correct SETUP packet, they could receive another SETUP packet before the data phase actually begins.

15.4.5 Start of Frame Processing

The USB host allocates time in 1.0 ms chunks called “Frames” for the purposes of packet scheduling. The USB host starts each frame with a broadcast token called SOF (start of frame) that includes an 11-bit sequence number. The TOKSOF interrupt is used to notify firmware when an SOF token was received.

Firmware can read the current frame number from the FRMNUML/FRMNUMH registers.

In general, the SOF interrupt is only monitored by devices using isochronous endpoints to help ensure that the device and host remain synchronized.

15.4.6 Suspend/Resume

The USB supports a single low-power mode called suspend. Getting into and out of the suspend state is described in the following sections.

15.4.6.1 Suspend

The USB host can put a single device or the entire bus into the suspend state at any time. The MCU supports suspend mode for low power. Suspend mode will be entered when the USB data lines are in the idle state for more than 3 ms. Entry into suspend mode is announced by the SLEEPF bit in the INTSTAT register.

Per the USB specification, a low-power bus-powered USB device is required to draw less than 500 μA in suspend state. A high-power device that supports remote wakeup and has its remote wake-up feature enabled by the host can draw up to 2.5 mA of current. After the initial 3-ms idle, the USB device will reach this state within 7 ms. This low-current requirement means that firmware is responsible for entering stop3 mode once the SLEEPF flag has been set and before the USB module has been placed in the suspend state.

On receipt of resume signaling from the USB, the module can generate an asynchronous interrupt to the MCU which brings the device out of stop mode and wakes up the clocks. Setting the USBRESMEN bit in the USBCTL0 register immediately after the SLEEPF bit is set enables this asynchronous notification feature. The USB resume signaling will then cause the LPRESF bit to be set, indicating a low-power SUSPEND resume, which will wake the CPU from stop3 mode.

During normal operation, while the host is sending SOF packets, the USB module will not enter suspend mode.

15.4.6.2 Resume

There are three ways to get out of the suspend state. When the USB module is in suspend state, the resume detection is active even if all the clocks are disabled and the MCU is in stop3 mode. The MCU can be activated from the suspend state by normal bus activity, a USB reset signal, or upstream resume (remote wakeup).

15.4.6.2.1 Host Initiated Resume

The host signals a resume from suspend by initiating resume signaling (K state) for at least 20 ms followed by a standard low-speed EOP signal. This 20 ms ensures that all devices in the USB network are awakened. After resuming the bus, the host must begin sending bus traffic within 3 ms to prevent the device from re-entering suspend mode.

Depending on the power mode the device is in while suspended, the notification for a host initiated resume will be different:

- Run mode - RESUME must be set after SLEEPF becomes set to enable the RESUMEF interrupt. Then, upon resume signaling, the RESUMEF interrupt will trigger after a K-state has been observed on the USBDP/USBDN lines for 2.5 μs .
- Stop3 mode - USBRESMEN must be set after SLEEPF becomes set to arm the LPRESF bit. Then, upon a K-state on the bus while the device is in stop3 mode, the LPRESF bit will be set, indicating

a resume from low-power suspend. This will trigger an asynchronous interrupt to wake the CPU from stop3 mode and resume clocks to the USB module.

NOTE

As a precaution, after LPRESF is set, firmware must check the state of the USB bus to see if the K-state was a result of a transient event and not a true host-initiated resume. If this is the case, then the device can drop back into stop3 if necessary. To do this, the RESUME interrupt can be enabled in conjunction with the USBRESMEN feature. Then, after LPRESF is set, and a K-state is still detected approximately 2.5 μ s after clocks have restarted, firmware can check that the RESUMEF interrupt has triggered, indicating resume signaling from the host.

15.4.6.2.2 USB Reset Signaling

Reset can wake a device from the suspend state.

15.4.6.2.3 Remote Wakeup

The USB device can send a resume event to the host by writing to the CRESUME bit. Firmware must first set the bit for the time period required by the USB Specification Rev. 2.0 (Section 7.1.7.7) and then clear it to 0.

15.4.7 Resets

The module supports multiple types of resets. The first is a bus reset generated by the USB Host, the second is a module reset generated by the MCU.

15.4.7.1 USB Bus Reset

At any time, the USB host may issue a reset to one or all of the devices attached to the bus. A USB reset is defined as a period of single ended zero (SE0) on the cable for greater than 2.5 μ s. 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.

15.4.7.2 USB Module Reset

USB module resets are initiated on-chip. During a module reset, the USB module is configured in the default mode. The USB module can also be forced into its reset state by setting the USBRESET bit in the USBCTL0 register. The default mode includes the following settings:

- Interrupts masked.
- USB clock enabled
- USB voltage regulator disabled

- USB transceiver disabled
- USBDP pullup disabled
- Endpoints disabled
- USB address register set to zero

15.4.8 Interrupts

Interrupts from the INTSTAT register signify events which occur during normal operation — USB start of frame tokens (TOKSOF), packet completion (TOKDNE), USB bus reset (USBRST), endpoint errors (ERROR), suspend and resume (SLEEP and RESUME), and endpoint stalled (STALL).

The ERRSTAT interrupts carry information about specific types of errors, which is needed on an application specific basis. Using ERRSTAT, an application can determine exactly why a packet transfer failed — due to CRC error, PID check error and so on.

Both registers are maskable via the INTENB and ERRENB registers. The INTSTAT and ERRSTAT are used to signal interrupts in a two-level structure. Unmasked interrupts from the ERRSTAT register are reported in the INTSTAT register.

Note that the interrupt registers work in concert with the STAT register. On receipt of an INTSTAT interrupt, software can check the STAT register and determine which BDT entry was affected by the transaction.

Chapter 16

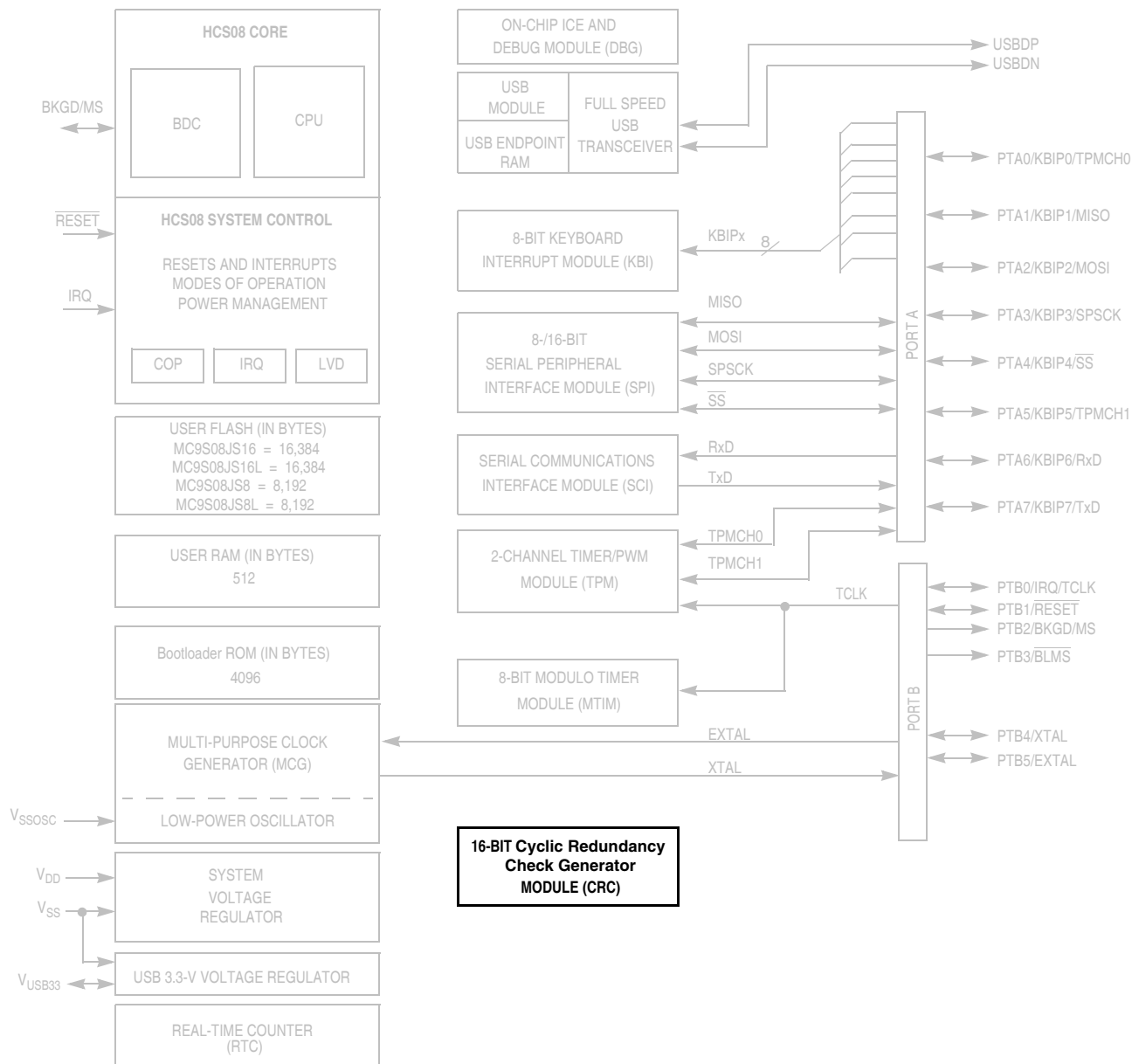
Cyclic Redundancy Check Generator (S08CRCV2)

16.1 Introduction

This chapter describes the cyclic redundancy check (CRC) generator module which uses the 16-bit CRC-CCITT polynomial, $x^{16} + x^{12} + x^5 + 1$, to generate a CRC code for error detection. The 16-bit code is calculated by 8-bit of data at a time, and provides a simple check for all accessible memory locations in flash or RAM.

NOTE

Stop1 mode is not available in this device.



NOTES:

1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1). Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD} . IRQ must not be driven above V_{DD} .
4. \overline{RESET} contains integrated pullup device if PTB1 enabled as reset pin function (RSTPE = 1)
5. Pin contains integrated pullup device.
6. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

Figure 16-1. MC9S08JS16 Series Block Diagram Highlighting the CRC Module

16.1.1 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

16.1.2 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 1 and 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.

16.1.3 Block Diagram

Figure 16-2 provides a block diagram of the CRC module

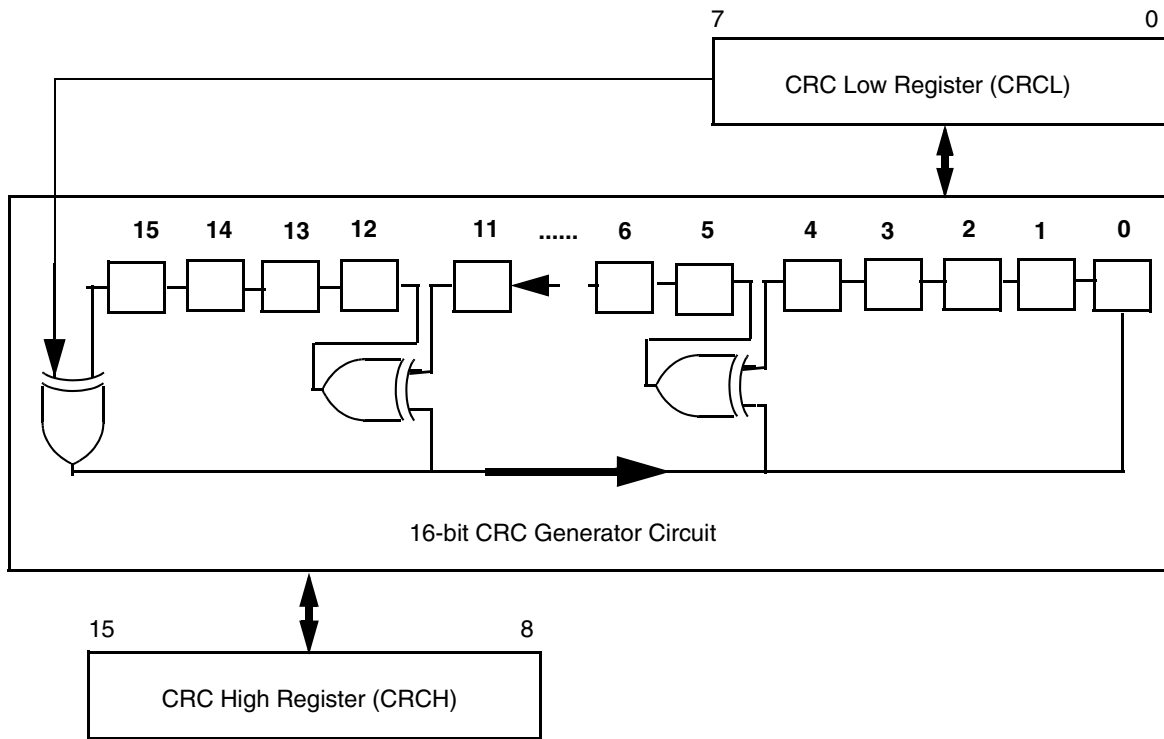


Figure 16-2. Cyclic Redundancy Check (CRC) Module Block Diagram

16.2 External Signal Description

There are no CRC signals that connect off chip.

16.3 Register Definition

16.3.1 Memory Map

Table 16-1. 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								
CRCL (offset=1)	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								

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. See [Section 16.5, “Initialization Information”](#) for more details.

16.3.2 Register Descriptions

The CRC module includes:

- A 16-bit CRC result and seed register (CRCH:CRCL)

Refer to the direct-page register summary in the [Chapter 4, “Memory,”](#) 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.

16.3.2.1 CRC High Register (CRCH)

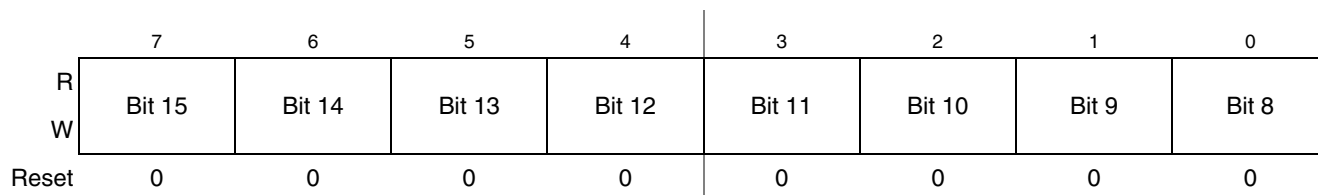


Figure 16-3. CRC High Register (CRCH)

Table 16-2. 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.

16.3.2.2 CRC Low Register (CRCL)

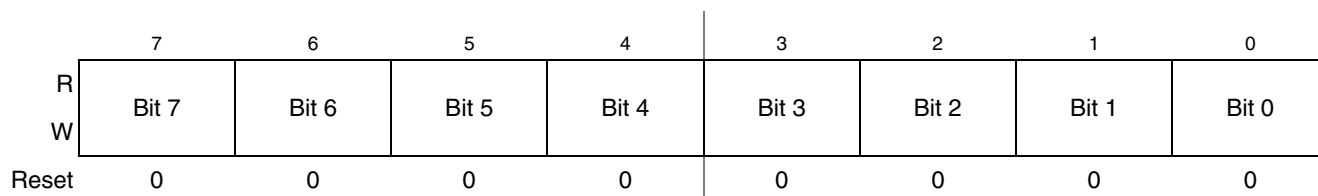


Figure 16-4. CRC Low Register (CRCL)

Table 16-3. 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.

16.4 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. After all 8 bits have been shifted into the CRC generator (in the next bus cycle after the data write to CRCL), 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.

16.4.1 ITU-T (CCITT) Recommendations & 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 16-2](#), but it recommends a SEED = 0x0000.
- ITU-T T.30 and ITU-T X.25 implement the same circuit shown in [Figure 16-2](#), 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 16-4 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 16-4. 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	0xab3e
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.

16.5 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 16.5, “Initialization Information”).
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 17

Development Support

17.1 Introduction

Development support systems in the HCS08 include the background debug controller (BDC) and the on-chip debug module (DBG). The BDC provides a single-wire debug interface to the target MCU that provides a convenient interface for programming the on-chip flash and other nonvolatile memories. The BDC is also the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modify, breakpoints, and single instruction trace commands.

In the HCS08 family, address and data bus signals are not available on external pins (not even in test mode). Debug is done through commands fed into the target MCU via the single-wire background debug interface. The debug module provides a means to selectively trigger and capture bus information so an external development system can reconstruct what happened inside the MCU on a cycle-by-cycle basis without having external access to the address and data signals.

The alternate BDC clock source for MC9S08JS16 series is the MCGLCLK. See [Chapter 9, “Multi-Purpose Clock Generator \(S08MCGV1\)”](#), for more information about MCGLCLK and how to select clock sources.

17.1.1 Features

Features of the BDC module include:

- Single pin for mode selection and background communications
- BDC registers are not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from stop or wait modes
- One hardware address breakpoint built into BDC
- Oscillator runs in stop mode, if BDC enabled
- COP watchdog disabled while in active background mode

Features of the ICE system include:

- Two trigger comparators: Two address + read/write (R/W) or one full address + data + R/W
- Flexible 8-word by 16-bit FIFO (first-in, first-out) buffer for capture information:
 - Change-of-flow addresses or
 - Event-only data
- Two types of breakpoints:
 - Tag breakpoints for instruction opcodes
 - Force breakpoints for any address access
- Nine trigger modes:
 - Basic: A-only, A OR B
 - Sequence: A then B
 - Full: A AND B data, A AND NOT B data
 - Event (store data): Event-only B, A then event-only B
 - Range: Inside range ($A \leq \text{address} \leq B$), outside range ($\text{address} < A$ or $\text{address} > B$)

17.2 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.

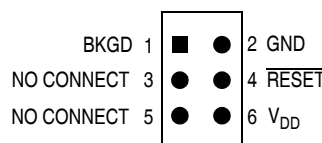


Figure 17-1. BDM Tool Connector

17.2.1 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 17.2.2, “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 17.2.2, “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.

17.2.2 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 17-2](#) 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.

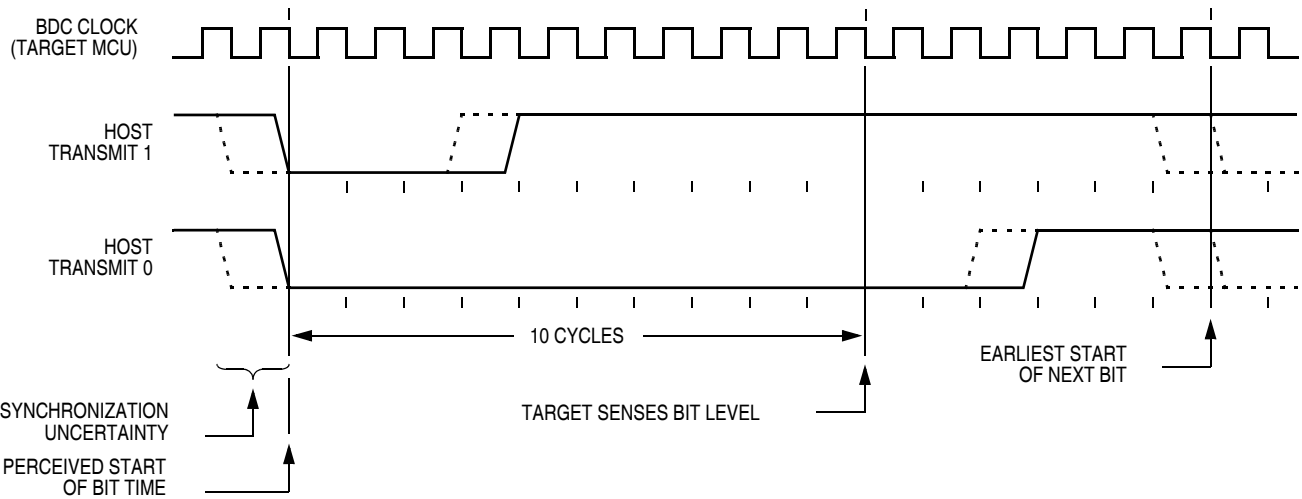


Figure 17-2. BDC Host-to-Target Serial Bit Timing

Figure 17-3 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 must sample the bit level about 10 cycles after it started the bit time.

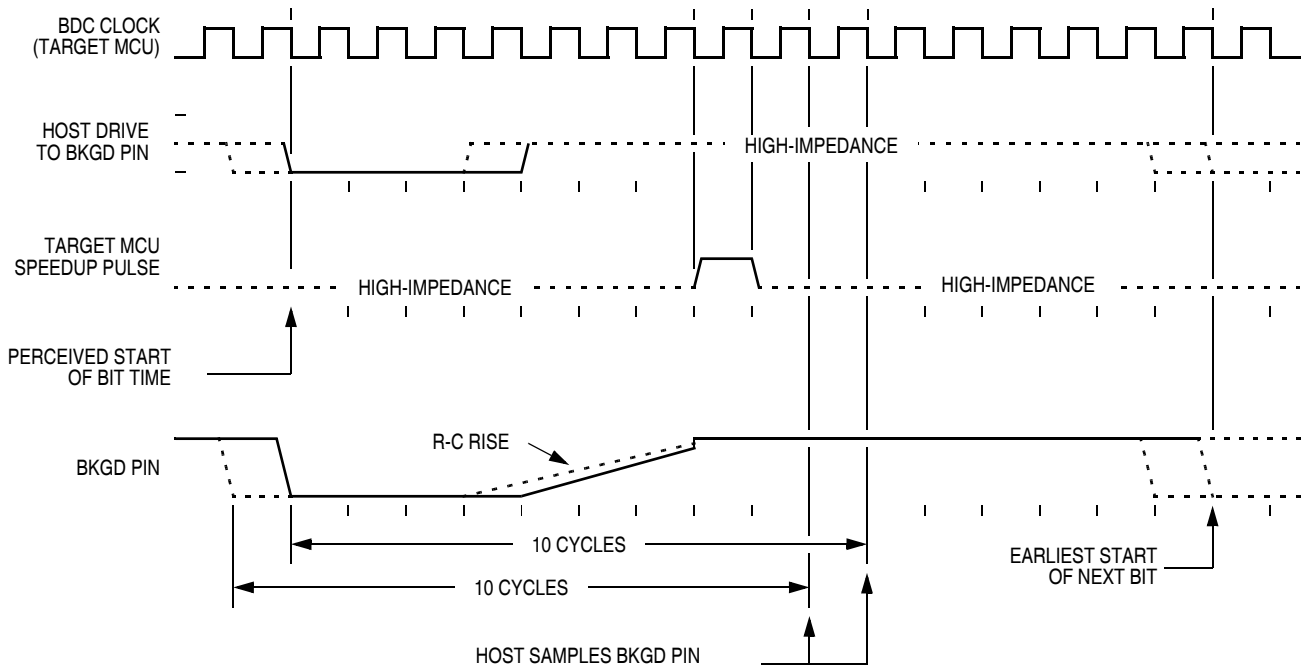


Figure 17-3. BDC Target-to-Host Serial Bit Timing (Logic 1)

Figure 17-4 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.

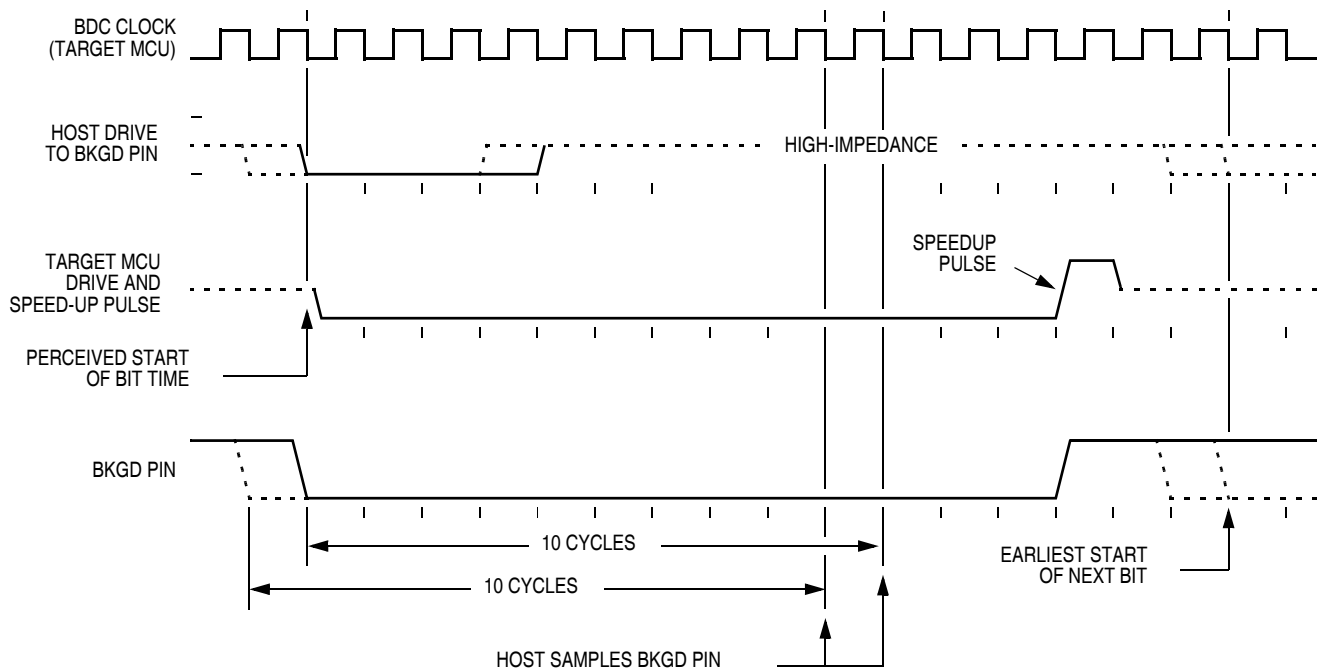


Figure 17-4. BDM Target-to-Host Serial Bit Timing (Logic 0)

17.2.3 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 17-1 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 17-1 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 17-1. 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.

17.2.4 BDC Hardware Breakpoint

The BDC includes one relatively simple hardware breakpoint that compares the CPU address bus to a 16-bit match value in the BDCBKPT register. This breakpoint can generate a forced breakpoint or a tagged breakpoint. A forced breakpoint causes the CPU to enter active background mode at the first instruction boundary following any access to the breakpoint address. The tagged breakpoint causes the instruction opcode at the breakpoint address to be tagged so that the CPU will enter active background mode rather than executing that instruction if and when it reaches the end of the instruction queue. This implies that tagged breakpoints can only be placed at the address of an instruction opcode while forced breakpoints can be set at any address.

The breakpoint enable (BKPTEN) control bit in the BDC status and control register (BDCSCR) is used to enable the breakpoint logic (BKPTEN = 1). When BKPTEN = 0, its default value after reset, the breakpoint logic is disabled and no BDC breakpoints are requested regardless of the values in other BDC breakpoint registers and control bits. The force/tag select (FTS) control bit in BDCSCR is used to select forced (FTS = 1) or tagged (FTS = 0) type breakpoints.

The on-chip debug module (DBG) includes circuitry for two additional hardware breakpoints that are more flexible than the simple breakpoint in the BDC module.

17.3 On-Chip Debug System (DBG)

Because HCS08 devices do not have external address and data buses, the most important functions of an in-circuit emulator have been built onto the chip with the MCU. The debug system consists of an 8-stage FIFO that can store address or data bus information, and a flexible trigger system to decide when to capture bus information and what information to capture. The system relies on the single-wire background debug system to access debug control registers and to read results out of the eight stage FIFO.

The debug module includes control and status registers that are accessible in the user's memory map. These registers are located in the high register space to avoid using valuable direct page memory space.

Most of the debug module's functions are used during development, and user programs rarely access any of the control and status registers for the debug module. The one exception is that the debug system can provide the means to implement a form of ROM patching. This topic is discussed in greater detail in [Section 17.3.6, "Hardware Breakpoints."](#)

17.3.1 Comparators A and B

Two 16-bit comparators (A and B) can optionally be qualified with the R/W signal and an opcode tracking circuit. Separate control bits allow you to ignore R/W for each comparator. The opcode tracking circuitry optionally allows you to specify that a trigger will occur only if the opcode at the specified address is actually executed as opposed to only being read from memory into the instruction queue. The comparators are also capable of magnitude comparisons to support the inside range and outside range trigger modes. Comparators are disabled temporarily during all BDC accesses.

The A comparator is always associated with the 16-bit CPU address. The B comparator compares to the CPU address or the 8-bit CPU data bus, depending on the trigger mode selected. Because the CPU data bus is separated into a read data bus and a write data bus, the RWAEN and RWA control bits have an additional purpose, in full address plus data comparisons they are used to decide which of these buses to use in the comparator B data bus comparisons. If RWAEN = 1 (enabled) and RWA = 0 (write), the CPU's write data bus is used. Otherwise, the CPU's read data bus is used.

The currently selected trigger mode determines what the debugger logic does when a comparator detects a qualified match condition. A match can cause:

- Generation of a breakpoint to the CPU
- Storage of data bus values into the FIFO
- Starting to store change-of-flow addresses into the FIFO (begin type trace)
- Stopping the storage of change-of-flow addresses into the FIFO (end type trace)

17.3.2 Bus Capture Information and FIFO Operation

The usual way to use the FIFO is to setup the trigger mode and other control options, then arm the debugger. When the FIFO has filled or the debugger has stopped storing data into the FIFO, you would read the information out of it in the order it was stored into the FIFO. Status bits indicate the number of words of valid information that are in the FIFO as data is stored into it. If a trace run is manually halted by writing 0 to ARM before the FIFO is full (CNT = 1:0:0:0), the information is shifted by one position and

the host must perform $((8 - \text{CNT}) - 1)$ dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see [Section 17.3.5, “Trigger Modes”](#)), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

In trigger modes where the FIFO is storing change-of-flow addresses, there is a delay between CPU addresses and the input side of the FIFO. Because of this delay, if the trigger event itself is a change-of-flow address or a change-of-flow address appears during the next two bus cycles after a trigger event starts the FIFO, it will not be saved into the FIFO. In the case of an end-trace, if the trigger event is a change-of-flow, it will be saved as the last change-of-flow entry for that debug run.

The FIFO can also be used to generate a profile of executed instruction addresses when the debugger is not armed. When $\text{ARM} = 0$, reading DBGFL causes the address of the most-recently fetched opcode to be saved in the FIFO. To use the profiling feature, a host debugger would read addresses out of the FIFO by reading DBGFH then DBGFL at regular periodic intervals. The first eight values would be discarded because they correspond to the eight DBGFL reads needed to initially fill the FIFO. Additional periodic reads of DBGFH and DBGFL return delayed information about executed instructions so the host debugger can develop a profile of executed instruction addresses.

17.3.3 Change-of-Flow Information

To minimize the amount of information stored in the FIFO, only information related to instructions that cause a change to the normal sequential execution of instructions is stored. With knowledge of the source and object code program stored in the target system, an external debugger system can reconstruct the path of execution through many instructions from the change-of-flow information stored in the FIFO.

For conditional branch instructions where the branch is taken (branch condition was true), the source address is stored (the address of the conditional branch opcode). Because BRA and BRN instructions are not conditional, these events do not cause change-of-flow information to be stored in the FIFO.

Indirect JMP and JSR instructions use the current contents of the H:X index register pair to determine the destination address, so the debug system stores the run-time destination address for any indirect JMP or JSR. For interrupts, RTI, or RTS, the destination address is stored in the FIFO as change-of-flow information.

17.3.4 Tag vs. Force Breakpoints and Triggers

Tagging is a term that refers to identifying an instruction opcode as it is fetched into the instruction queue, but not taking any other action until and unless that instruction is actually executed by the CPU. This distinction is important because any change-of-flow from a jump, branch, subroutine call, or interrupt causes some instructions that have been fetched into the instruction queue to be thrown away without being executed.

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the DBGTR register is set to select tag-type operation, the output from comparator A or B is qualified by a block of logic in the debug module that tracks opcodes and only produces a trigger to the debugger if the opcode at the compare address is actually executed. There is separate opcode tracking logic for each comparator so more than one compare event can be tracked through the instruction queue at a time.

17.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGTR register selects one of nine trigger modes. When TRGSEL = 1 in the DBGTR register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGTR chooses whether the FIFO begins storing data when the qualified trigger is detected (begin trace), or the FIFO stores data in a circular fashion from the time it is armed until the qualified trigger is detected (end trigger).

A debug run is started by writing a 1 to the ARM bit in the DBGCR register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGSR. A begin-trace debug run ends when the FIFO gets full. An end-trace run ends when the selected trigger event occurs. Any debug run can be stopped manually by writing a 0 to ARM or DBGEN in DBGCR.

In all trigger modes except event-only modes, the FIFO stores change-of-flow addresses. In event-only trigger modes, the FIFO stores data in the low-order eight bits of the FIFO.

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following trigger mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

A-Only — Trigger when the address matches the value in comparator A

A OR B — Trigger when the address matches either the value in comparator A or the value in comparator B

A Then B — Trigger when the address matches the value in comparator B but only after the address for another cycle matched the value in comparator A. There can be any number of cycles after the A match and before the B match.

A AND B Data (Full Mode) — This is called a full mode because address, data, and R/W (optionally) must match within the same bus cycle to cause a trigger event. Comparator A checks address, the low byte of comparator B checks data, and R/W is checked against RWA if RWAEN = 1. The high-order half of comparator B is not used.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

A AND NOT B Data (Full Mode) — Address must match comparator A, data must not match the low half of comparator B, and R/W must match RWA if RWAEN = 1. All three conditions must be met within the same bus cycle to cause a trigger.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

Event-Only B (Store Data) — Trigger events occur each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

A Then Event-Only B (Store Data) — After the address has matched the value in comparator A, a trigger event occurs each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

Inside Range ($A \leq \text{Address} \leq B$) — A trigger occurs when the address is greater than or equal to the value in comparator A and less than or equal to the value in comparator B at the same time.

Outside Range ($\text{Address} < A$ or $\text{Address} > B$) — A trigger occurs when the address is either less than the value in comparator A or greater than the value in comparator B.

17.3.6 Hardware Breakpoints

The BRKEN control bit in the DBGCR register may be set to 1 to allow any of the trigger conditions described in [Section 17.3.5, “Trigger Modes,”](#) to be used to generate a hardware breakpoint request to the CPU. TAG in DBGCR controls whether the breakpoint request will be treated as a tag-type breakpoint or a force-type breakpoint. A tag breakpoint causes the current opcode to be marked as it enters the instruction queue. If a tagged opcode reaches the end of the pipe, the CPU executes a BGND instruction to go to active background mode rather than executing the tagged opcode. A force-type breakpoint causes the CPU to finish the current instruction and then go to active background mode.

If the background mode has not been enabled (ENBDM = 1) by a serial WRITE_CONTROL command through the BKGD pin, the CPU will execute an SWI instruction instead of going to active background mode.

17.4 Register Definition

This section contains the descriptions of the BDC and DBG registers and control bits.

Refer to the high-page register summary in the device overview chapter of this data sheet for the absolute address assignments for all DBG registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

17.4.1 BDC Registers and Control Bits

The BDC has two registers:

- The BDC status and control register (BDCSCR) is an 8-bit register containing control and status bits for the background debug controller.
- The BDC breakpoint match register (BDCBKPT) holds a 16-bit breakpoint match address.

These registers are accessed with dedicated serial BDC commands and are not located in the memory space of the target MCU (so they do not have addresses and cannot be accessed by user programs).

Some of the bits in the BDCSCR have write limitations; otherwise, these registers may be read or written at any time. For example, the ENBDM control bit may not be written while the MCU is in active background mode. (This prevents the ambiguous condition of the control bit forbidding active background mode while the MCU is already in active background mode.) Also, the four status bits (BDMACT, WS, WSF, and DVF) are read-only status indicators and can never be written by the WRITE_CONTROL serial BDC command. The clock switch (CLKSW) control bit may be read or written at any time.

17.4.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ_STATUS and WRITE_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

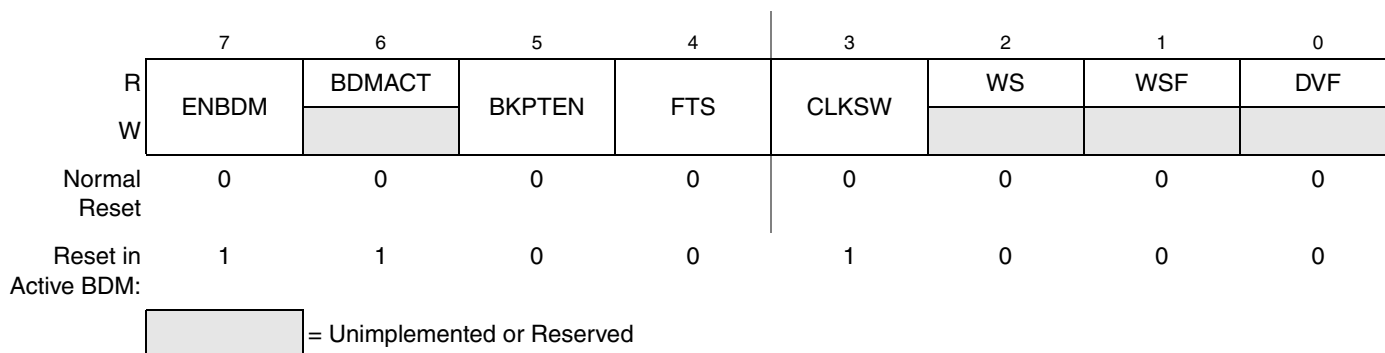


Figure 17-5. BDC Status and Control Register (BDCSCR)

Table 17-2. 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 17-2. BDCSCR Register Field Descriptions (continued)

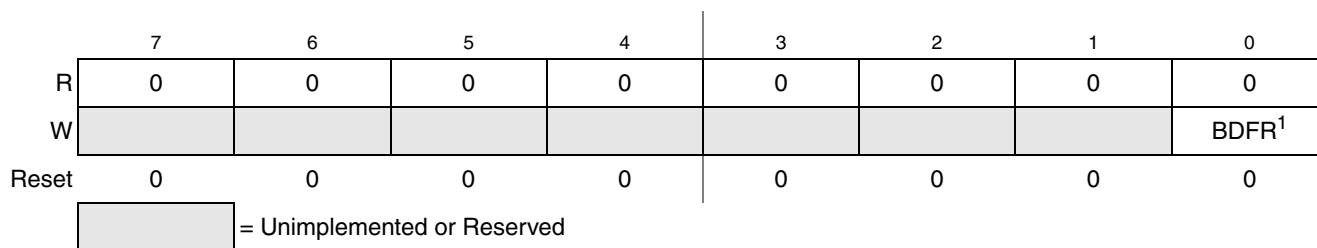
Field	Description
2 WS	<p>Wait or Stop Status — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host must issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands.</p> <p>0 Target CPU is running user application code or in active background mode (was not in wait or stop mode when background became active)</p> <p>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</p>
1 WSF	<p>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.)</p> <p>0 Memory access did not conflict with a wait or stop instruction</p> <p>1 Memory access command failed because the CPU entered wait or stop mode</p>
0 DVF	<p>Data Valid Failure Status — This status bit is not used in the MC9S08JS16 series because it does not have any slow access memory.</p> <p>0 Memory access did not conflict with a slow memory access</p> <p>1 Memory access command failed because CPU was not finished with a slow memory access</p>

17.4.1.2 BDC Breakpoint Match Register (BDCBKPT)

This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ_BKPT and WRITE_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU. Breakpoints are normally set while the target MCU is in active background mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to [Section 17.2.4, “BDC Hardware Breakpoint.”](#)

17.4.2 System Background Debug Force Reset Register (SBDFR)

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



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

Figure 17-6. System Background Debug Force Reset Register (SBDFR)

Table 17-3. 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.

17.4.3 DBG Registers and Control Bits

The debug module includes nine bytes of register space for three 16-bit registers and three 8-bit control and status registers. These registers are located in the high register space of the normal memory map so they are accessible to normal application programs. These registers are rarely if ever accessed by normal user application programs with the possible exception of a ROM patching mechanism that uses the breakpoint logic.

17.4.3.1 Debug Comparator A High Register (DBGCAH)

This register contains compare value bits for the high-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

17.4.3.2 Debug Comparator A Low Register (DBGCAL)

This register contains compare value bits for the low-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

17.4.3.3 Debug Comparator B High Register (DBGCBH)

This register contains compare value bits for the high-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

17.4.3.4 Debug Comparator B Low Register (DBGCBL)

This register contains compare value bits for the low-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

17.4.3.5 Debug FIFO High Register (DBGFH)

This register provides read-only access to the high-order eight bits of the FIFO. Writes to this register have no meaning or effect. In the event-only trigger modes, the FIFO only stores data into the low-order byte of each FIFO word, so this register is not used and will read 0x00.

Reading DBGFH does not cause the FIFO to shift to the next word. When reading 16-bit words out of the FIFO, read DBGFH before reading DBGFL because reading DBGFL causes the FIFO to advance to the next word of information.

17.4.3.6 Debug FIFO Low Register (DBGFL)

This register provides read-only access to the low-order eight bits of the FIFO. Writes to this register have no meaning or effect.

Reading DBGFL causes the FIFO to shift to the next available word of information. When the debug module is operating in event-only modes, only 8-bit data is stored into the FIFO (high-order half of each FIFO word is unused). When reading 8-bit words out of the FIFO, simply read DBGFL repeatedly to get successive bytes of data from the FIFO. It isn't necessary to read DBGFH in this case.

Do not attempt to read data from the FIFO while it is still armed (after arming but before the FIFO is filled or ARMF is cleared) because the FIFO is prevented from advancing during reads of DBGFL. This can interfere with normal sequencing of reads from the FIFO.

Reading DBGFL while the debugger is not armed causes the address of the most-recently fetched opcode to be stored to the last location in the FIFO. By reading DBGFH then DBGFL periodically, external host software can develop a profile of program execution. After eight reads from the FIFO, the ninth read will return the information that was stored as a result of the first read. To use the profiling feature, read the FIFO eight times without using the data to prime the sequence and then begin using the data to get a delayed picture of what addresses were being executed. The information stored into the FIFO on reads of DBGFL (while the FIFO is not armed) is the address of the most-recently fetched opcode.

17.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

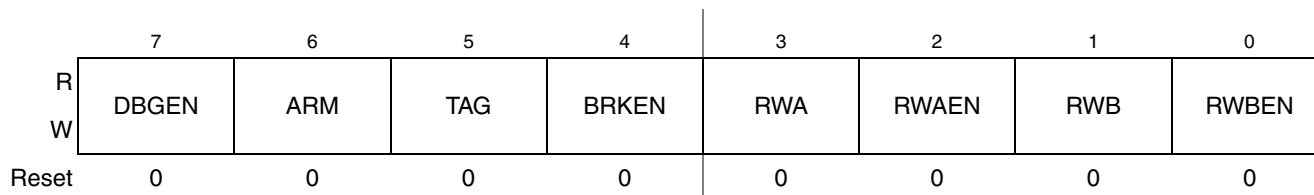


Figure 17-7. Debug Control Register (DBGC)

Table 17-4. DBGC Register Field Descriptions

Field	Description
7 DBGEN	Debug Module Enable — Used to enable the debug module. DBGEN cannot be set to 1 if the MCU is secure. 0 DBG disabled 1 DBG enabled
6 ARM	Arm Control — Controls whether the debugger is comparing and storing information in the FIFO. A write is used to set this bit (and ARMF) and completion of a debug run automatically clears it. Any debug run can be manually stopped by writing 0 to ARM or to DBGEN. 0 Debugger not armed 1 Debugger armed
5 TAG	Tag/Force Select — Controls whether break requests to the CPU will be tag or force type requests. If BRKEN = 0, this bit has no meaning or effect. 0 CPU breaks requested as force type requests 1 CPU breaks requested as tag type requests
4 BRKEN	Break Enable — Controls whether a trigger event will generate a break request to the CPU. Trigger events can cause information to be stored in the FIFO without generating a break request to the CPU. For an end trace, CPU break requests are issued to the CPU when the comparator(s) and R/W meet the trigger requirements. For a begin trace, CPU break requests are issued when the FIFO becomes full. TRGSEL does not affect the timing of CPU break requests. 0 CPU break requests not enabled 1 Triggers cause a break request to the CPU
3 RWA	R/W Comparison Value for Comparator A — When RWAEN = 1, this bit determines whether a read or a write access qualifies comparator A. When RWAEN = 0, RWA and the R/W signal do not affect comparator A. 0 Comparator A can only match on a write cycle 1 Comparator A can only match on a read cycle
2 RWAEN	Enable R/W for Comparator A — Controls whether the level of R/W is considered for a comparator A match. 0 R/W is not used in comparison A 1 R/W is used in comparison A
1 RWB	R/W Comparison Value for Comparator B — When RWBEN = 1, this bit determines whether a read or a write access qualifies comparator B. When RWBEN = 0, RWB and the R/W signal do not affect comparator B. 0 Comparator B can match only on a write cycle 1 Comparator B can match only on a read cycle
0 RWBEN	Enable R/W for Comparator B — Controls whether the level of R/W is considered for a comparator B match. 0 R/W is not used in comparison B 1 R/W is used in comparison B

17.4.3.8 Debug Trigger Register (DBGT)

This register can be read any time, but may be written only if ARM = 0, except bits 4 and 5 are hard-wired to 0s.

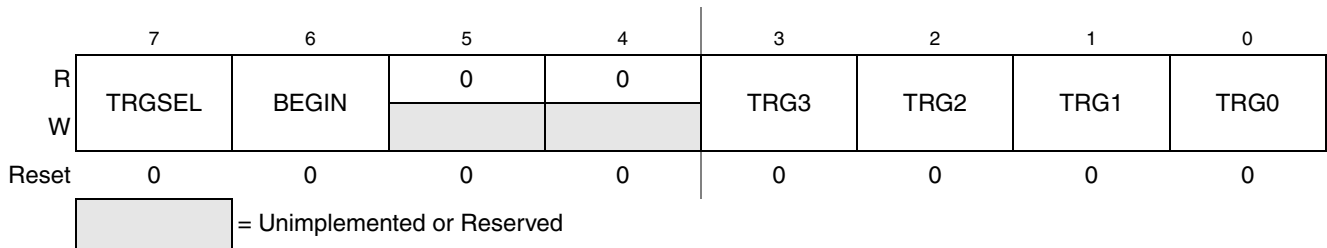


Figure 17-8. Debug Trigger Register (DBGT)

Table 17-5. DBGT Register Field Descriptions

Field	Description
7 TRGSEL	Trigger Type — Controls whether the match outputs from comparators A and B are qualified with the opcode tracking logic in the debug module. If TRGSEL is set, a match signal from comparator A or B must propagate through the opcode tracking logic and a trigger event is only signalled to the FIFO logic if the opcode at the match address is actually executed. 0 Trigger on access to compare address (force) 1 Trigger if opcode at compare address is executed (tag)
6 BEGIN	Begin/End Trigger Select — Controls whether the FIFO starts filling at a trigger or fills in a circular manner until a trigger ends the capture of information. In event-only trigger modes, this bit is ignored and all debug runs are assumed to be begin traces. 0 Data stored in FIFO until trigger (end trace) 1 Trigger initiates data storage (begin trace)
3:0 TRG[3:0]	Select Trigger Mode — Selects one of nine triggering modes, as described below. 0000 A-only 0001 A OR B 0010 A Then B 0011 Event-only B (store data) 0100 A then event-only B (store data) 0101 A AND B data (full mode) 0110 A AND NOT B data (full mode) 0111 Inside range: $A \leq \text{address} \leq B$ 1000 Outside range: $\text{address} < A$ or $\text{address} > B$ 1001 – 1111 (No trigger)

17.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.

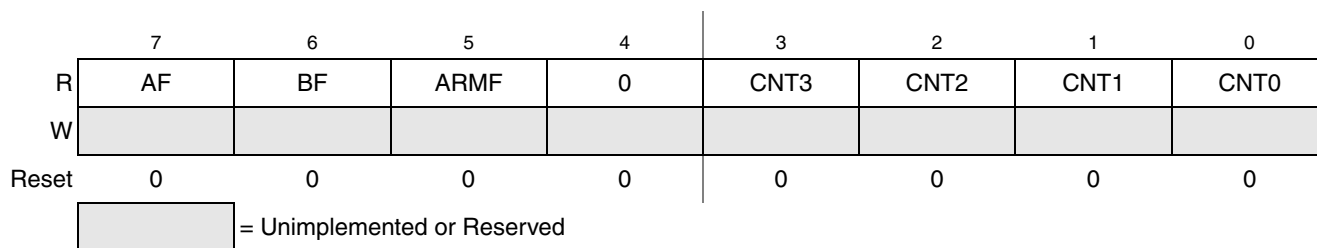


Figure 17-9. Debug Status Register (DBGS)

Table 17-6. DBGS Register Field Descriptions

Field	Description
7 AF	Trigger Match A Flag — AF is cleared at the start of a debug run and indicates whether a trigger match A condition was met since arming. 0 Comparator A has not matched 1 Comparator A match
6 BF	Trigger Match B Flag — BF is cleared at the start of a debug run and indicates whether a trigger match B condition was met since arming. 0 Comparator B has not matched 1 Comparator B match
5 ARMF	Arm Flag — While DBGEN = 1, this status bit is a read-only image of ARM in DBG. This bit is set by writing 1 to the ARM control bit in DBG (while DBGEN = 1) and is automatically cleared at the end of a debug run. A debug run is completed when the FIFO is full (begin trace) or when a trigger event is detected (end trace). A debug run can also be ended manually by writing 0 to ARM or DBGEN in DBG. 0 Debugger not armed 1 Debugger armed
3:0 CNT[3:0]	FIFO Valid Count — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO. 0000 Number of valid words in FIFO = No valid data 0001 Number of valid words in FIFO = 1 0010 Number of valid words in FIFO = 2 0011 Number of valid words in FIFO = 3 0100 Number of valid words in FIFO = 4 0101 Number of valid words in FIFO = 5 0110 Number of valid words in FIFO = 6 0111 Number of valid words in FIFO = 7 1000 Number of valid words in FIFO = 8

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

<http://www.freescale.com/support>

USA/Europe or Locations Not Listed:

Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

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
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.



Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2008-2009. All rights reserved.