



# SC9S08MZ16

Data Sheet: Technical Data

**HCS08**  
**Microcontrollers**

SC9S08MZ16  
Rev. 1  
9/2008

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





# SC9S08MZ16 Features

## 8-Bit HCS08 Central Processor Unit (CPU)

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

## Memory Options

- Up to 16 KB of on-chip in-circuit programmable flash memory with block protection and security options
- Up to 1 KB of on-chip RAM

## Clock Source Options

- Clock source options include crystal, resonator, external clock, or internally generated clock with precision NVM trimming

## System Protection

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

## Power-Saving Modes

- Two stop modes

## Peripherals

- **ADC** — 8-channel, 8-bit analog-to-digital converter with automatic compare function
- **SCI** — Two serial communications interface modules with optional 13-bit break
- **Timers** — Three 16-bit timer/pulse-width modulator (TPM) modules. Two 2-channel and one 4-channel; each has selectable input capture, output compare, and edge-aligned PWM capability on each channel. Each timer module may be configured for buffered, centered PWM (CPWM) on all channels

## Input/Output

- Up to 34 general-purpose input/output (I/O) pins
- Software selectable pullups on ports when used as inputs
- Software selectable slew rate control on ports when used as outputs
- Software selectable drive strength on ports when used as outputs
- Master reset pin and power-on reset (POR)
- Internal pullup on RESET, IRQ, and BKGD/MS pins to reduce customer system cost

## Package Options

- 44-pin low-profile quad flat package (LQFP)
- 32-pin low-profile quad flat package (LQFP)

## Full Functional Operational Conditions

- Supply voltage: 2.7 V to 5.5 V
- Ambient temperature: -40 °C to 85 °C



---

# SC9S08MZ16 Data Sheet

SC9S08MZ16  
Rev. 1  
9/2008



# 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. For your convenience, the page number designators have been linked to the appropriate location.

Revision Number	Revision Date	Description of Changes
1	9/8/2008	Initial public released.

This product incorporates SuperFlash<sup>®</sup> technology licensed from SST.

Freescale<sup>™</sup> and the Freescale logo are trademarks of Freescale Semiconductor, Inc.  
© Freescale Semiconductor, Inc., 2008. All rights reserved.

# List of Chapters

<b>Chapter Number</b>	<b>Title</b>	<b>Page</b>
<b>Chapter 1</b>	<b>Introduction</b>	<b>17</b>
<b>Chapter 2</b>	<b>Pins and Connections</b>	<b>21</b>
<b>Chapter 3</b>	<b>Modes of Operation</b>	<b>31</b>
<b>Chapter 4</b>	<b>Memory</b>	<b>37</b>
<b>Chapter 5</b>	<b>Resets, Interrupts, and System Configuration</b>	<b>59</b>
<b>Chapter 6</b>	<b>Parallel Input/Output</b>	<b>77</b>
<b>Chapter 7</b>	<b>Central Processor Unit (S08CPUV2)</b>	<b>101</b>
<b>Chapter 8</b>	<b>Internal Clock Generator (S08ICGV4)</b>	<b>121</b>
<b>Chapter 9</b>	<b>Timer/PWM (S08TPMV3)</b>	<b>147</b>
<b>Chapter 10</b>	<b>Serial Communications Interface (S08SCIV4)</b>	<b>175</b>
<b>Chapter 11</b>	<b>Analog-to-Digital Converter (S08ADC10V1)</b>	<b>195</b>
<b>Chapter 12</b>	<b>Development Support</b>	<b>223</b>
<b>Appendix A</b>	<b>Electrical Characteristics and Timing Specifications</b>	<b>245</b>
<b>Appendix B</b>	<b>Ordering Information and Mechanical Drawings</b>	<b>261</b>





# Contents

Section Number	Title	Page
<b>Chapter 1</b>		
<b>Introduction</b>		
1.1	Overview .....	17
1.2	MCU Block Diagrams .....	17
1.3	System Clock Distribution .....	19
<b>Chapter 2</b>		
<b>Pins and Connections</b>		
2.1	Introduction .....	21
2.2	Device Pin Assignment .....	21
2.3	Recommended System Connections .....	25
2.3.1	Power ( $V_{DD}$ , $2 \times V_{SS}$ , $V_{DDAD}$ , $V_{SSAD}$ ) .....	27
2.3.2	Oscillator (XTAL, EXTAL) .....	27
2.3.3	RESET Pin .....	27
2.3.4	Background/Mode Select (BKGD/MS) .....	28
2.3.5	ADC Reference Pins ( $V_{REFH}$ , $V_{REFL}$ ) .....	28
2.3.6	External Interrupt Pin (IRQ) .....	28
2.3.7	General-Purpose I/O and Peripheral Ports .....	29
<b>Chapter 3</b>		
<b>Modes of Operation</b>		
3.1	Introduction .....	31
3.2	Features .....	31
3.3	Run Mode .....	31
3.4	Active Background Mode .....	31
3.5	Stop Modes .....	32
3.5.1	Stop2 Mode .....	33
3.5.2	Stop3 Mode .....	34
3.5.3	Active BDM Enabled in Stop Mode .....	34
3.5.4	LVD Enabled in Stop Mode .....	35
3.5.5	On-Chip Peripheral Modules in Stop Modes .....	35
<b>Chapter 4</b>		
<b>Memory</b>		
4.1	SC9S08MZ16 Memory Map .....	37
4.1.1	Reset and Interrupt Vector Assignments .....	38
4.2	Register Addresses and Bit Assignments .....	39
4.3	RAM .....	44
4.4	Flash .....	45

4.4.1	Features .....	45
4.4.2	Program and Erase Times .....	46
4.4.3	Program and Erase Command Execution .....	46
4.4.4	Burst Program Execution .....	48
4.4.5	Access Errors .....	50
4.4.6	Flash Block Protection .....	50
4.4.7	Vector Redirection .....	51
4.5	Security .....	51
4.6	Flash Registers and Control Bits .....	52
4.6.1	Flash Clock Divider Register (FCDIV) .....	53
4.6.2	Flash Options Register (FOPT and NVOPT) .....	54
4.6.3	Flash Configuration Register (FCNFG) .....	55
4.6.4	Flash Protection Register (FPROT and NVPROT) .....	56
4.6.5	Flash Status Register (FSTAT) .....	56
4.6.6	Flash Command Register (FCMD) .....	58

## Chapter 5 Resets, Interrupts, and System Configuration

5.1	Introduction .....	59
5.2	Features .....	59
5.3	MCU Reset .....	59
5.4	Computer Operating Properly (COP) Watchdog .....	60
5.5	Interrupts .....	61
5.5.1	Interrupt Stack Frame .....	62
5.5.2	External Interrupt Request (IRQ) Pin .....	62
5.5.3	Interrupt Vectors, Sources, and Local Masks .....	63
5.6	Low-Voltage Detect (LVD) System .....	65
5.6.1	Power-On Reset Operation .....	65
5.6.2	LVD Reset Operation .....	65
5.6.3	LVD Interrupt Operation .....	65
5.6.4	Low-Voltage Warning (LVW) .....	65
5.7	Real-Time Interrupt (RTI) .....	65
5.8	MCLK Output .....	66
5.9	Reset, Interrupt, and System Control Registers and Control Bits .....	66
5.9.1	Interrupt Pin Request Status and Control Register (IRQSC) .....	67
5.9.2	System Reset Status Register (SRS) .....	68
5.9.3	System Background Debug Force Reset Register (SBDFFR) .....	69
5.9.4	System Options Register (SOPT) .....	69
5.9.5	System MCLK Control Register (SMCLK) .....	70
5.9.6	System Device Identification Register (SDIDH, SDIDL) .....	71
5.9.7	System Real-Time Interrupt Status and Control Register (SRTISC) .....	72
5.9.8	System Power Management Status and Control 1 Register (SPMSC1) .....	73
5.9.9	System Power Management Status and Control 2 Register (SPMSC2) .....	74
5.9.10	System Options Register 2 (SOPT2) .....	75

## Chapter 6 Parallel Input/Output

6.1	Introduction .....	77
6.2	Features .....	79
6.3	Pin Descriptions .....	79
6.3.1	Port A .....	79
6.3.2	Port B .....	80
6.3.3	Port C .....	80
6.3.4	Port D .....	81
6.3.5	Port E .....	81
6.3.6	Port F .....	82
6.3.7	Port G .....	82
6.4	Parallel I/O Control .....	82
6.5	Pin Control .....	83
6.5.1	Internal Pullup Enable .....	84
6.5.2	Output Slew Rate Control Enable .....	84
6.5.3	Output Drive Strength Select .....	84
6.6	Pin Behavior in Stop Modes .....	85
6.7	Parallel I/O and Pin Control Registers .....	85
6.7.1	Port A I/O Registers (PTAD and PTADD) .....	85
6.7.2	Port A Pin Control Registers (PTAPE, PTASE, PTADS) .....	86
6.7.3	Port B I/O Registers (PTBD and PTBDD) .....	87
6.7.4	Port B Pin Control Registers (PTBPE, PTBSE, PTBDS) .....	88
6.7.5	Port C I/O Registers (PTCD and PTCDD) .....	89
6.7.6	Port C Pin Control Registers (PTCPE, PTCSE, PTCDS) .....	90
6.7.7	Port D I/O Registers (PTDD and PTDDD) .....	91
6.7.8	Port D Pin Control Registers (PTDPE, PTDSE, PTDDS) .....	92
6.7.9	Port E I/O Registers (PTED and PTEDD) .....	93
6.7.10	Port E Pin Control Registers (PTEPE, PTESE, PTEDS) .....	94
6.7.11	Port F I/O Registers (PTFD and PTFDD) .....	95
6.7.12	Port F Pin Control Registers (PTFPE, PTFSE, PTFDS) .....	96
6.7.13	Port G I/O Registers (PTGD and PTGDD) .....	97
6.7.14	Port G Pin Control Registers (PTGPE, PTGSE, PTGDS) .....	98

## Chapter 7 Central Processor Unit (S08CPUV2)

7.1	Introduction .....	101
7.1.1	Features .....	101
7.2	Programmer's Model and CPU Registers .....	102
7.2.1	Accumulator (A) .....	102
7.2.2	Index Register (H:X) .....	102
7.2.3	Stack Pointer (SP) .....	103
7.2.4	Program Counter (PC) .....	103
7.2.5	Condition Code Register (CCR) .....	103
7.3	Addressing Modes .....	104

7.3.1	Inherent Addressing Mode (INH)	105
7.3.2	Relative Addressing Mode (REL)	105
7.3.3	Immediate Addressing Mode (IMM)	105
7.3.4	Direct Addressing Mode (DIR)	105
7.3.5	Extended Addressing Mode (EXT)	105
7.3.6	Indexed Addressing Mode	105
7.4	Special Operations	106
7.4.1	Reset Sequence	107
7.4.2	Interrupt Sequence	107
7.4.3	Wait Mode Operation	108
7.4.4	Stop Mode Operation	108
7.4.5	BGND Instruction	108
7.5	HCS08 Instruction Set Summary	109

## Chapter 8 Internal Clock Generator (S08ICGV4)

8.1	Introduction	123
8.1.1	Features	123
8.1.2	Modes of Operation	124
8.1.3	Block Diagram	125
8.2	External Signal Description	125
8.2.1	EXTAL — External Reference Clock / Oscillator Input	125
8.2.2	XTAL — Oscillator Output	125
8.2.3	External Clock Connections	126
8.2.4	External Crystal/Resonator Connections	126
8.3	Register Definition	127
8.3.1	ICG Control Register 1 (ICGC1)	127
8.4	ICG Control Register 2 (ICGC2)	128
8.4.1	ICG Status Register 1 (ICGS1)	129
8.4.2	ICG Status Register 2 (ICGS2)	130
8.4.3	ICG Filter Registers (ICGFLTU, ICGFLTL)	130
8.4.4	ICG Trim Register (ICGTRM)	131
8.5	Functional Description	131
8.5.1	Off Mode (Off)	132
8.5.2	Self-Clocked Mode (SCM)	132
8.5.3	FLL Engaged, Internal Clock (FEI) Mode	133
8.5.4	FLL Engaged Internal Unlocked	134
8.5.5	FLL Engaged Internal Locked	134
8.5.6	FLL Bypassed, External Clock (FBE) Mode	134
8.5.7	FLL Engaged, External Clock (FEE) Mode	134
8.5.8	FLL Lock and Loss-of-Lock Detection	135
8.5.9	FLL Loss-of-Clock Detection	136
8.5.10	Clock Mode Requirements	137
8.5.11	Fixed Frequency Clock	138
8.5.12	High Gain Oscillator	138

8.6	Initialization/Application Information .....	138
8.6.1	Introduction .....	138
8.6.2	Example #1: External Crystal = 32 kHz, Bus Frequency = 4.19 MHz .....	140
8.6.3	Example #2: External Crystal = 4 MHz, Bus Frequency = 20 MHz .....	141
8.6.4	Example #3: No External Crystal Connection, 5.4 MHz Bus Frequency .....	143
8.6.5	Example #4: Internal Clock Generator Trim .....	145

## Chapter 9 Timer/PWM (S08TPMV3)

9.1	Introduction .....	147
9.1.1	TPMV3 Differences from Previous Versions .....	149
9.1.2	Migrating from TPMV1 .....	151
9.1.3	Features .....	151
9.1.4	Modes of Operation .....	153
9.1.5	Block Diagram .....	154
9.2	Signal Description .....	156
9.2.1	Detailed Signal Descriptions .....	156
9.3	Register Definition .....	160
9.3.1	TPM Status and Control Register (TPMxSC) .....	160
9.3.2	TPM-Counter Registers (TPMxCNTH:TPMxCNTL) .....	161
9.3.3	TPM Counter Modulo Registers (TPMxMODH:TPMxMODL) .....	162
9.3.4	TPM Channel n Status and Control Register (TPMxCnSC) .....	163
9.3.5	TPM Channel Value Registers (TPMxCnVH:TPMxCnVL) .....	165
9.4	Functional Description .....	166
9.4.1	Counter .....	167
9.4.2	Channel Mode Selection .....	168
9.5	Reset Overview .....	172
9.5.1	General .....	172
9.5.2	Description of Reset Operation .....	172
9.6	Interrupts .....	172
9.6.1	General .....	172
9.6.2	Description of Interrupt Operation .....	172

## Chapter 10 Serial Communications Interface (S08SCIV4)

10.1	Introduction .....	175
10.1.1	Features .....	177
10.1.2	Modes of Operation .....	177
10.1.3	Block Diagram .....	177
10.2	Register Definition .....	180
10.2.1	SCI Baud Rate Registers (SCIxBDH, SCIxBDL) .....	180
10.2.2	SCI Control Register 1 (SCIxC1) .....	181
10.2.3	SCI Control Register 2 (SCIxC2) .....	182
10.2.4	SCI Status Register 1 (SCIxS1) .....	183
10.2.5	SCI Status Register 2 (SCIxS2) .....	185

10.2.6	SCI Control Register 3 (SCIxC3)	186
10.2.7	SCI Data Register (SCIxD)	187
10.3	Functional Description	187
10.3.1	Baud Rate Generation	187
10.3.2	Transmitter Functional Description	188
10.3.3	Receiver Functional Description	189
10.3.4	Interrupts and Status Flags	191
10.3.5	Additional SCI Functions	192

## Chapter 11 Analog-to-Digital Converter (S08ADC10V1)

11.1	Overview	195
11.2	Channel Assignments	197
11.2.1	Alternate Clock	197
11.2.2	Hardware Trigger	198
11.2.3	Temperature Sensor	198
11.2.4	Features	199
11.2.5	Block Diagram	199
11.3	External Signal Description	200
11.3.1	Analog Power ( $V_{DDAD}$ )	201
11.3.2	Analog Ground ( $V_{SSAD}$ )	201
11.3.3	Voltage Reference High ( $V_{REFH}$ )	201
11.3.4	Voltage Reference Low ( $V_{REFL}$ )	201
11.3.5	Analog Channel Inputs (ADx)	201
11.4	Register Definition	201
11.4.1	Status and Control Register 1 (ADC1SC1)	201
11.4.2	Status and Control Register 2 (ADC1SC2)	203
11.4.3	Data Result High Register (ADC1RH)	204
11.4.4	Data Result Low Register (ADC1RL)	204
11.4.5	Compare Value High Register (ADC1CVH)	205
11.4.6	Compare Value Low Register (ADC1CVL)	205
11.4.7	Configuration Register (ADC1CFG)	205
11.4.8	Pin Control 1 Register (APCTL1)	207
11.4.9	Pin Control 2 Register (APCTL2)	208
11.4.10	Pin Control 3 Register (APCTL3)	209
11.5	Functional Description	210
11.5.1	Clock Select and Divide Control	210
11.5.2	Input Select and Pin Control	211
11.5.3	Hardware Trigger	211
11.5.4	Conversion Control	211
11.5.5	Automatic Compare Function	214
11.5.6	MCU Wait Mode Operation	214
11.5.7	MCU Stop3 Mode Operation	214
11.5.8	MCU Stop1 and Stop2 Mode Operation	215
11.6	Initialization Information	215

11.6.1	ADC Module Initialization Example .....	215
11.7	Application Information .....	217
11.7.1	External Pins and Routing .....	217
11.7.2	Sources of Error .....	219

## Chapter 12 Development Support

12.1	Introduction .....	223
12.1.1	Features .....	224
12.2	Background Debug Controller (BDC) .....	224
12.2.1	BKGD Pin Description .....	225
12.2.2	Communication Details .....	226
12.2.3	BDC Commands .....	229
12.2.4	BDC Hardware Breakpoint .....	232
12.3	On-Chip Debug System (DBG) .....	233
12.3.1	Comparators A and B .....	233
12.3.2	Bus Capture Information and FIFO Operation .....	233
12.3.3	Change-of-Flow Information .....	234
12.3.4	Tag vs. Force Breakpoints and Triggers .....	234
12.3.5	Trigger Modes .....	235
12.3.6	Hardware Breakpoints .....	237
12.4	Register Definition .....	237
12.4.1	BDC Registers and Control Bits .....	237
12.4.2	System Background Debug Force Reset Register (SBDFR) .....	239
12.4.3	DBG Registers and Control Bits .....	240

## Appendix A Electrical Characteristics and Timing Specifications

A.1	Introduction .....	245
A.2	Parameter Classification.....	245
A.3	Absolute Maximum Ratings.....	245
A.4	Thermal Characteristics.....	246
A.5	DC Characteristics.....	247
A.6	Supply Current Characteristics.....	249
A.7	ADC Characteristics.....	252
A.8	Internal Clock Generation Module Characteristics.....	254
A.8.1	ICG Frequency Specifications .....	255
A.9	AC Characteristics.....	257
A.9.1	Control Timing .....	258
A.9.2	Timer/PWM (TPM) Module Timing.....	259
A.10	Flash Specifications.....	259
A.11	EMC Performance.....	260

## Appendix B Ordering Information and Mechanical Drawings

---

B.1	Ordering Information .....	261
B.2	Mechanical Drawings.....	261



# Chapter 1

## Introduction

### 1.1 Overview

The SC9S08MZ16 device is a member 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. Refer to [Table 1-1](#) for memory sizes and package types.

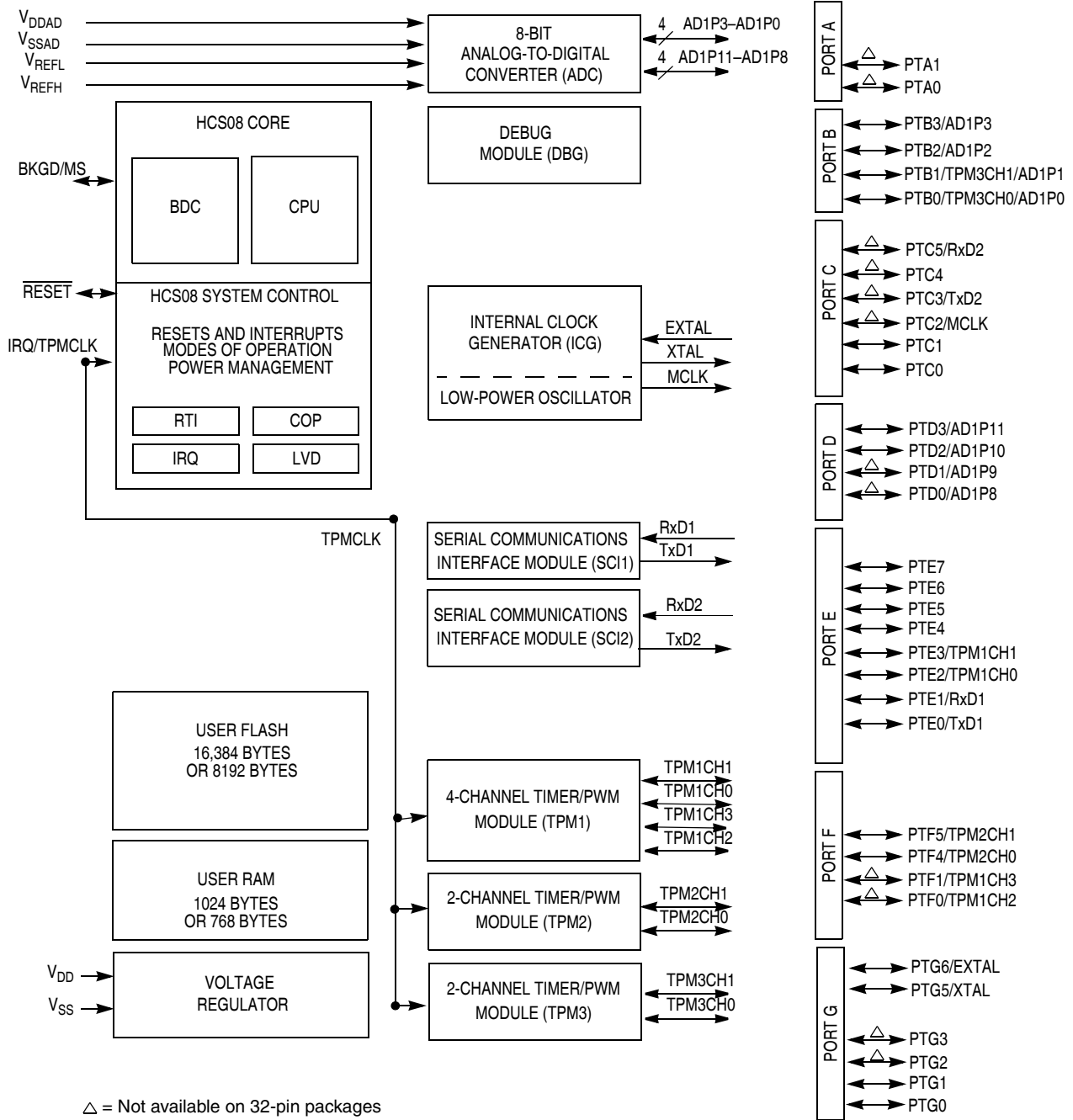
[Table 1-1](#) summarizes the feature set available in the MCU.

**Table 1-1. Features by MCU and Package**

Feature	SC9S08MZ16	
Flash size (bytes)	16 K	
RAM size (bytes)	1024	
Pin quantity	44	32
ADC channels	8	6
TPM1 channels	4	2
TPM2 channels	2	2
TPM3 channels	2	2
GPIO pins	34	22

### 1.2 MCU Block Diagrams

The block diagram shows the structure of the SC9S08MZ16 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  $V_{DD}$ . Do not drive an IRQ above  $V_{DD}$ .
4. Pin contains integrated pullup device.
5. PTD3 and PTD2 contain both pullup and pulldown devices.

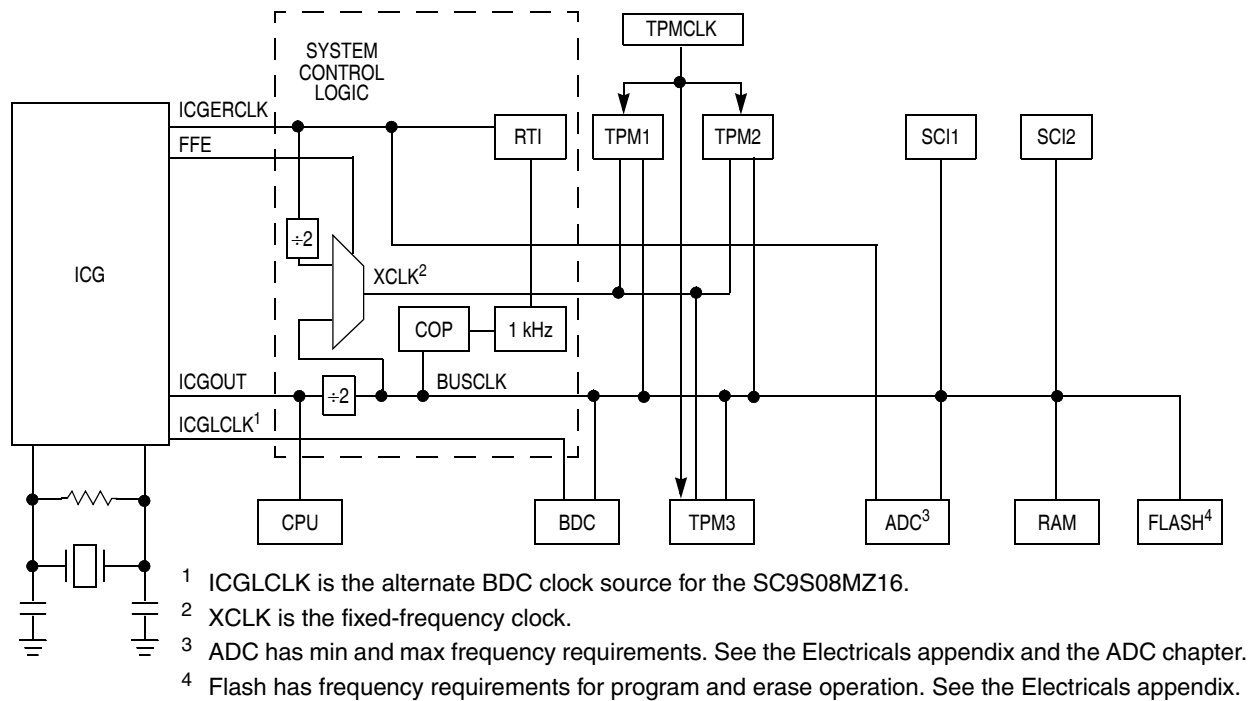
**Figure 1-1. SC9S08MZ16 Block Diagram**

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

**Table 1-2. Versions of On-Chip Modules**

Module	Version
Analog-to-Digital Converter (ADC)	1
Internal Clock Generator (ICG)	4
Serial Communications Interface (SCI)	4
Timer Pulse-Width Modulator (TPM)	3
Central Processing Unit (CPU)	2

### 1.3 System Clock Distribution



**Figure 1-2. System Clock Distribution Diagram**

Some of the modules inside the MCU have clock source choices. Figure 1-2 shows a simplified clock connection diagram. The ICG supplies the clock sources:

- ICGOUT is one of the following output options from the ICG module:
  - The external crystal oscillator
  - An external clock source
  - The output of the digitally-controlled oscillator (DCO) in the frequency-locked loop sub-module
  - Control bits inside the ICG determine which source is connected

- FFE is a control signal generated inside the ICG. If the frequency of ICGOUT is great than  $4 \times$  the frequency of ICGERCLK, this signal is a logic 1 and the fixed-frequency clock is ICGERCLK/2. Otherwise the fixed-frequency clock is BUSCLK.
- ICGLCLK — Development tools can select the internal self-clocked source (~ 8 MHz) to speed up BDC communications in systems where the bus clock is slow.
- ICGERCLK — External reference clock can be selected as the real-time interrupt clock source. Can also be used as the ALTCLK input to the ADC module.

---

## **Chapter 2**

# **Pins and Connections**

### **2.1 Introduction**

This chapter describes the 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. shows the 44-pin LQFP pin assignments for the SC9S08MZ16 device.

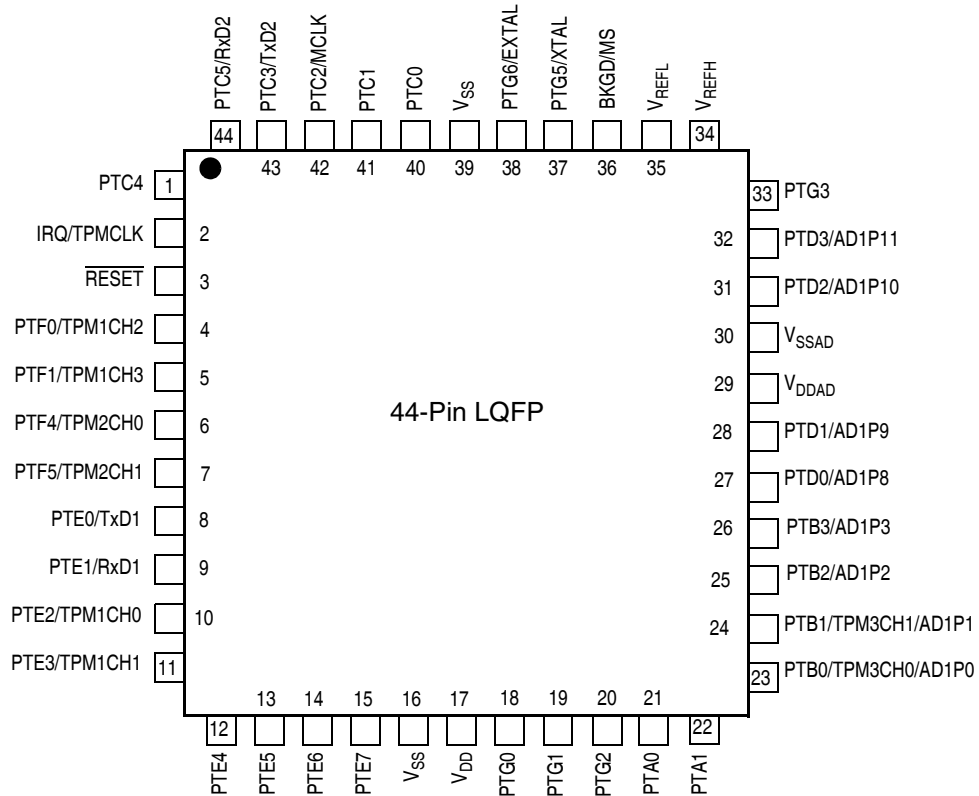


Figure 2-1. SC9S08MZ16 in 44-Pin LQFP Package

Figure 2-2. shows the 32-pin LQFP pin assignments for the SC9S08MZ16 device.

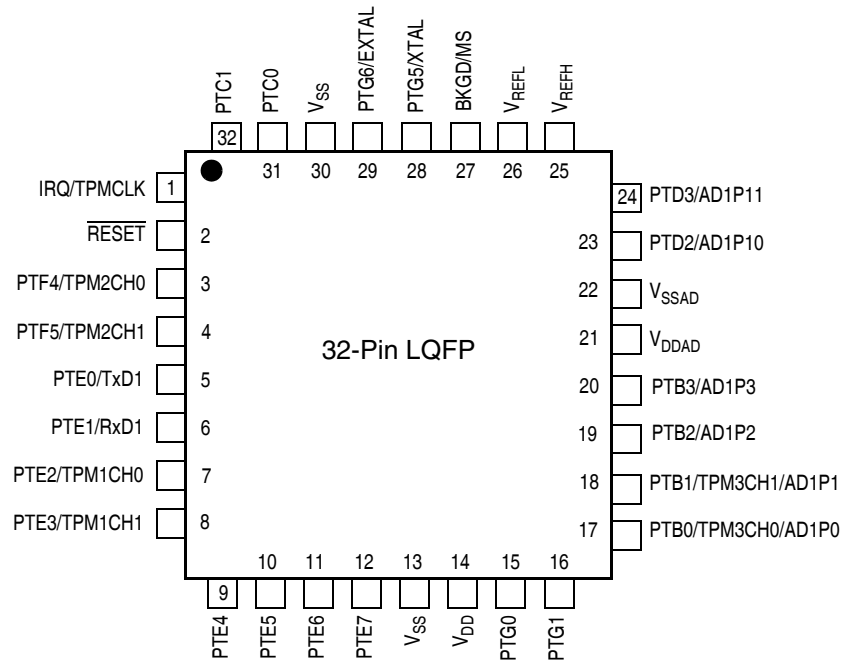


Figure 2-2. SC9S08MZ16 in 32-Pin LQFP Package

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

Pin Number		<-- Lowest Priority --> Highest		
44	32	Port Pin	Alt 1	Alt 2
1	—	PTC4		
2	1		IRQ	TPMCLK
3	2			RESET
4	—	PTF0	TPM1CH2	
5	—	PTF1	TPM1CH3	
6	3	PTF4	TPM2CH0	
7	4	PTF5	TPM2CH1	
8	5	PTE0		TxD1
9	6	PTE1		RxD1
10	7	PTE2	TPM1CH0	
11	8	PTE3	TPM1CH1	
12	9	PTE4		
13	10	PTE5		
14	11	PTE6		
15	12	PTE7		
16	13			V <sub>SS</sub>
17	14			V <sub>DD</sub>
18	15	PTG0		
19	16	PTG1		
20	—	PTG2		
21	—	PTA0		
22	—	PTA1		

Pin Number		<-- Lowest Priority --> Highest		
44	32	Port Pin	Alt 1	Alt 2
23	17	PTB0	TPM3CH0	AD1P0
24	18	PTB1	TPM3CH1	AD1P1
25	19	PTB2	AD1P2	
26	20	PTB3	AD1P3	
27	—	PTD0	AD1P8	
28	—	PTD1	AD1P9	
29	21			V <sub>DDAD</sub>
30	22			V <sub>SSAD</sub>
31	23	PTD2	AD1P10	
32	24	PTD3	AD1P11	
33	—	PTG3		
34	25			V <sub>REFH</sub>
35	26			V <sub>REFL</sub>
36	27	BKGD	MS	
37	28	PTG5	XTAL	
38	29	PTG6	EXTAL	
39	30			V <sub>SS</sub>
40	31	PTC0		
41	32	PTC1		
42	—	PTC2		MCLK
43	—	PTC3		TxD2
44	—	PTC5		RxD2

Table 2-2. Pin Function Reference

Signal Function	Example(s)	Reference
Port Pins	PTAx, PTBx	<a href="#">Chapter 6, “Parallel Input/Output”</a>
Timer/PWM	TCLK, TPMCHx	<a href="#">Chapter 9, “Timer/PWM (S08TPMV3)”</a>
Serial communications interface	TxD, RxD	<a href="#">Chapter 10, “Serial Communications Interface (S08SCIV4)”</a>
Oscillator/clocking	EXTAL, XTAL	<a href="#">Chapter 8, “Internal Clock Generator (S08ICGV4)”</a>
Analog-to-digital	ADPx	<a href="#">Chapter 11, “Analog-to-Digital Converter (S08ADC10V1)”</a>
Power/core	BKGD/MS, V <sub>DD</sub> , V <sub>SS</sub>	<a href="#">Chapter 2, “Pins and Connections”</a>

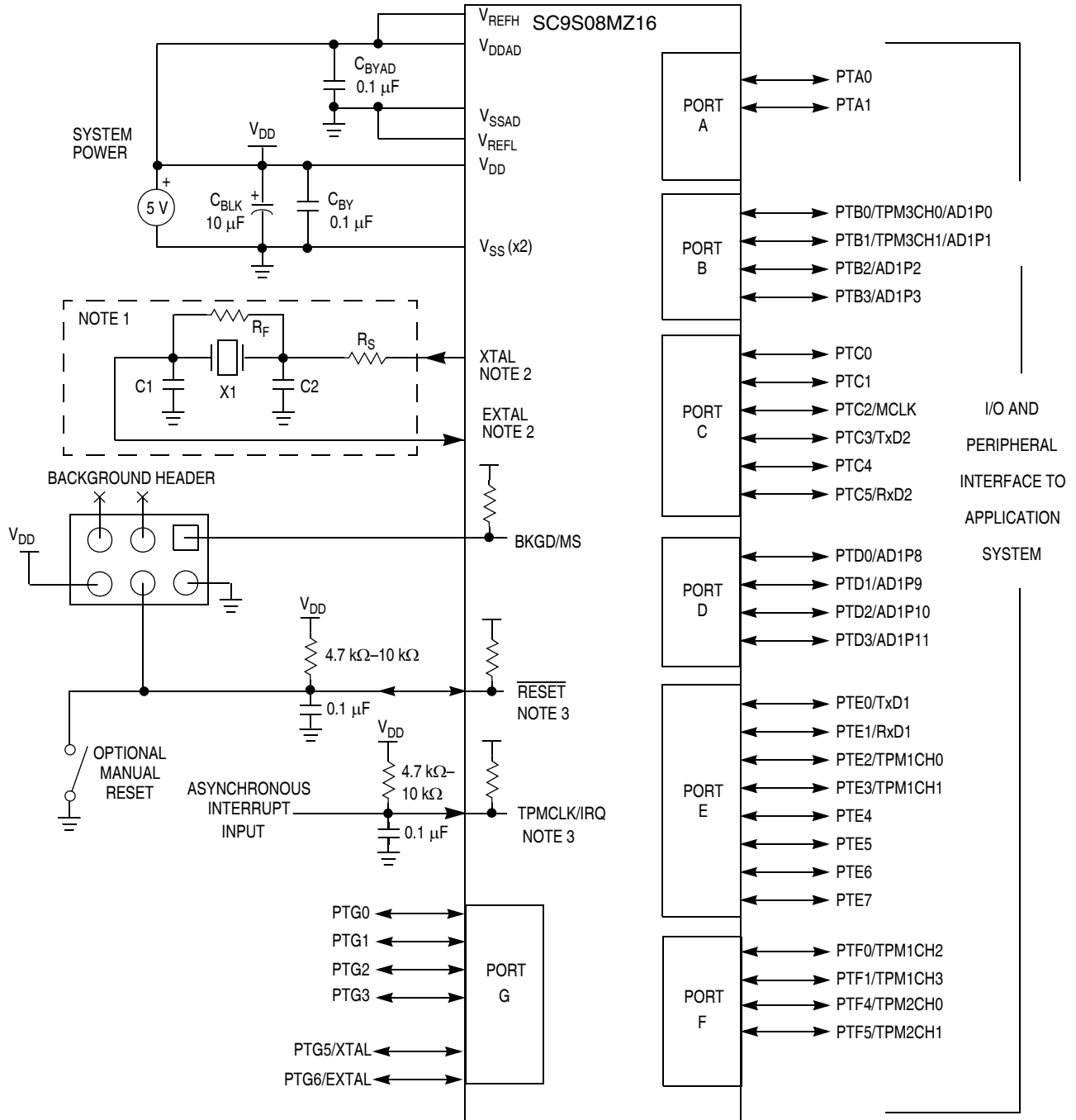


Table 2-2. Pin Function Reference

Signal Function	Example(s)	Reference
Reset and interrupts	$\overline{\text{RESET}}$ , IRQ	<a href="#">Chapter 5, “Resets, Interrupts, and System Configuration”</a>

## 2.3 Recommended System Connections

[Figure 2-3](#) shows pin connections that are common to almost all SC9S08MZ16 application systems.



NOTES:

1. Not required if using the internal clock option.
2. XTAL and EXTAL are PTG5 and PTG6 respectively.
3. RC filters on RESET and IRQ are recommended for EMC-sensitive applications.

Figure 2-3. Basic System Connections

### 2.3.1 Power ( $V_{DD}$ , $2 \times V_{SS}$ , $V_{DDAD}$ , $V_{SSAD}$ )

$V_{DD}$  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 should be a bulk electrolytic capacitor, such as a 10  $\mu$ F tantalum capacitor, to provide bulk charge storage for the overall system and a 0.1  $\mu$ F ceramic bypass capacitor located as near to the paired  $V_{DD}$  and  $V_{SS}$  power pins as practical to suppress high-frequency noise. The SC9S08MZ16 has a second  $V_{SS}$  pin. This pin should be connected to the system ground plane or to the primary  $V_{SS}$  pin through a low-impedance connection.

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

### 2.3.2 Oscillator (XTAL, EXTAL)

Out of reset the MCU uses an internally generated clock (self-clocked mode —  $f_{Self\_reset}$ ) equivalent to about 8 MHz crystal rate. This frequency source is used during reset startup and can be enabled as the clock source for stop recovery to avoid the need for a long crystal startup delay. This MCU also contains a trimmable internal clock generator (ICG) module that can be used to run the MCU. For more information on the ICG, see the [Chapter 8, “Internal Clock Generator \(S08ICGV4\).”](#)

The MCU has a Pierce oscillator that can accommodate a crystal or ceramic resonator in either of two frequency ranges selected by the RANGE bit in the ICGC1 register. Rather than a crystal or ceramic resonator, an external oscillator can be connected to the EXTAL input pin.

Refer to [Figure 2-3](#) for the following discussion.  $R_S$  (when used) and  $R_F$  should be low-inductance resistors such as carbon composition resistors. Wire-wound resistors, and some metal film resistors, have too much inductance. C1 and C2 normally should be high-quality ceramic capacitors that are specifically designed for high-frequency applications.

$R_F$  is used to provide a bias path to keep the EXTAL input in its linear range during crystal startup and its value is not generally critical. Typical systems use 1 M $\Omega$  to 10 M $\Omega$ . 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 sizing C1 and C2. The crystal manufacturer typically specifies a load capacitance which is the series combination of C1 and C2 which are usually the same size. As a first-order approximation, use 10 pF as an estimate of combined pin and PCB capacitance for each oscillator pin (EXTAL and XTAL).

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

$\overline{\text{RESET}}$  is a dedicated pin with a pullup device built in. It has input hysteresis, a high current output driver, and no output slew rate control. Internal power-on reset and low-voltage reset circuitry typically make

external reset circuitry unnecessary. This pin is normally connected to the standard 6-pin background debug connector so a development system can directly reset the MCU system. If desired, a manual external reset can be added by supplying a simple switch to ground (pull reset pin low to force a reset).

Whenever any reset is initiated (whether from an external signal or from an internal system), the reset pin is driven low for approximately 34 bus cycles. The reset circuitry decodes the cause of reset and records it by setting a corresponding bit in the system control reset status register (SRS).

In EMC-sensitive applications, an external RC filter is recommended on the reset pin. See [Figure 2-3](#) for an example.

### 2.3.4 Background/Mode Select (BKGD/MS)

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

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

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

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

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

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

### 2.3.6 External Interrupt Pin (IRQ)

The IRQ pin is the input source for the IRQ interrupt and is also the input for the BIH and BIL instructions. If the IRQ function is not enabled, this pin does not perform any function.

In EMC-sensitive applications, an external RC filter is recommended on the IRQ pin. See [Figure 2-3](#) for an example.

### 2.3.7 General-Purpose I/O and Peripheral Ports

The remaining pins are shared among general-purpose I/O and on-chip peripheral functions such as timers and serial I/O systems. Immediately after reset, all of these pins are configured as high-impedance general-purpose inputs with internal pullup devices disabled.

#### NOTE

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

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, refer to the appropriate chapter from [Table 2-2](#).

When an on-chip peripheral system is controlling a pin, data direction control bits still determine what is read from port data registers even though the peripheral module controls the pin direction by controlling the enable for the pin’s output buffer. See the [Chapter 6, “Parallel Input/Output”](#) chapter for more details.

Pullup enable bits for each input pin control whether on-chip pullup devices are enabled whenever the pin is acting as an input even if it is being controlled by an on-chip peripheral module. When IRQ is configured as the IRQ input and is set to detect rising edges, the pullup enable control bit enables a pulldown device rather than a pullup device.

#### NOTE

When an alternate function is first enabled, a spurious edge to a module can occur. To avoid this, 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. It is recommended that all modules that share a pin be disabled before enabling another module.



# Chapter 3

## Modes of Operation

### 3.1 Introduction

The operating modes of the SC9S08MZ16 are described in this chapter. Entry into each mode, exit from each mode, and functionality while in each of the modes are described.

### 3.2 Features

- Run mode
- Active background mode for code development
- Stop modes:
  - System clocks stopped; voltage regulator in standby
  - Stop2 — Partial power down of internal circuits, RAM contents retained
  - Stop3 — All internal circuits powered for fast recovery

#### NOTE

SC9S08MZ16 does not support wait mode. In [Chapter 7, “Central Processor Unit \(S08CPUV2\)”](#), all wait mode description is not available in this device.

### 3.3 Run Mode

This is the normal operating mode for the SC9S08MZ16. This mode is selected when the BKGD/MS pin is high at the rising-edge of reset. After a reset occurs in this mode, the CPU executes code fetched from internal memory beginning at 0xFFFFE:0xFFFF.

### 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 debug module (DBG), provide the means for analyzing MCU operation during software development.

Active background mode is entered in any of five ways:

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

- When encountering a DBG breakpoint

After entering active background mode, the CPU is held in a suspended state waiting for serial background commands rather than executing instructions from the user's 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's 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 SC9S08MZ16 is shipped from the Freescale Semiconductor 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 12, "Development Support."](#)

## 3.5 Stop Modes

One of two stop modes is entered upon execution of a STOP instruction when the stop mode enable bit (STOPE) in the system option register is set. In both stop modes, all internal clocks are halted. If the STOPE bit is not set when the CPU executes a STOP instruction, the MCU does not enter either of the stop modes and an illegal opcode reset is forced. The stop modes are selected by setting the appropriate bits in SPMSC2.

The SC9S08MZ16 device does not include stop1 mode.

[Table 3-1](#) summarizes the behavior of the MCU in each of the stop modes.



Table 3-1. Stop Mode Behavior

Mode	PPDC <sup>1</sup>	CPU, Digital Peripherals, Flash	RAM	ICG	ADC	Regulator	I/O Pins	RTI
Stop2	1	Off	Standby	Off	Disabled	Standby	States held	Optionally on
Stop3	0	Standby	Standby	Off <sup>2</sup>	Optionally on	Standby	States held	Optionally on

<sup>1</sup> PPDC is the partial power down control bit, see [Section 5.9.9, “System Power Management Status and Control 2 Register \(SPMSC2\),”](#) for details.

<sup>2</sup> Crystal oscillator can be configured to run in stop3. See the ICG registers.

### 3.5.1 Stop2 Mode

Stop2 mode provides very low standby power consumption and maintains the contents of RAM and the current state of all of the I/O pins. To enter stop2, the user must execute a STOP instruction with stop2 selected (PPDC = 1) and stop mode enabled (STOPE = 1). In addition, the LVD must not be enabled to operate in stop (LVDSE = LVDE = 1). If the LVD is enabled in stop, then the MCU enters stop3 upon the execution of the STOP instruction regardless of the state of PPDC.

Before entering stop2 mode, the user must save the contents of the I/O port registers, as well as any other memory-mapped registers which they want to restore after exit of stop2, to locations in RAM. Upon exit of stop2, these values can be restored by user software before pin latches are opened.

When the MCU is in stop2 mode, all internal circuits that are powered from the voltage regulator are turned off, except for the RAM. The voltage regulator is in a low-power standby state, as is the ADC. Upon entry into stop2, the states of the I/O pins are latched. The states are held while in stop2 mode and after exiting stop2 mode until a logic 1 is written to PPDACK in SPMSC2.

Exit from stop2 is done by asserting either of the wake-up pins:  $\overline{\text{RESET}}$  or IRQ/TPMCLK, or by an RTI interrupt. IRQ/TPMCLK is always an active low input when the MCU is in stop2, regardless of how it was configured before entering stop2.

Upon wake-up from stop2 mode, the MCU will start up as from a power-on reset (POR) except pin states remain latched. The CPU will take the reset vector. The system and all peripherals will be in their default reset states and must be initialized.

After waking up from stop2, the PPDF bit in SPMSC2 is set. This flag may be 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 logic 1 is written to PPDACK in SPMSC2.

To maintain I/O state for pins that were configured as general-purpose I/O, 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 register bits will assume their reset states when the I/O pin latches are opened and the I/O pins will switch to their reset states.

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

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

### 3.5.2 Stop3 Mode

To enter stop3, the user must execute a STOP instruction with stop3 selected (PPDC = 0) and stop mode enabled (STOPE = 1). Upon entering the stop3 mode, all of the clocks in the MCU, including the oscillator itself, are halted. The ICG enters its standby state, as does the voltage regulator and the ADC. The states of all of the internal registers and logic, as well as the RAM content, are maintained. The I/O pin states are not latched at the pin as in stop2. Instead they are maintained by virtue of the states of the internal logic driving the pins being maintained.

Exit from stop3 is done by asserting  $\overline{\text{RESET}}$  or by an interrupt from one of the following sources: the real-time interrupt (RTI), LVD system, ADC, IRQ, or SCI.

If stop3 is exited by means of the  $\overline{\text{RESET}}$  pin, then the MCU will be reset and operation will resume after taking the reset vector. Exit by means of an asynchronous interrupt or the real-time interrupt will result in the MCU taking the appropriate interrupt vector.

A separate self-clocked source ( $\approx 1$  kHz) for the real-time interrupt allows a wakeup from stop2 or stop3 mode with no external components. When RTIS2:RTIS1:RTIS0 = 0:0:0, the real-time interrupt function and this 1 kHz source are disabled. Power consumption is lower when the 1 kHz source is disabled, but in that case the real-time interrupt cannot wake the MCU from stop.

### 3.5.3 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if the ENBDM bit in BDCSCR is set. This register is described in [Chapter 12, “Development Support”](#) of this data sheet. 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 so background debug communication is still 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 instead enter stop3.

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 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. [Table 3-2](#) summarizes the behavior of the MCU in stop when entry into the background debug mode is enabled.

**Table 3-2. BDM Enabled Stop Mode Behavior**

Mode	PPDC	CPU, Digital Peripherals, Flash	RAM	ICG	ADC	Regulator	I/O Pins	RTI
Stop3	0	Standby	Standby	Active	Optionally on	Active	States held	Optionally on

### 3.5.4 LVD Enabled in Stop Mode

The LVD system is capable of generating either an interrupt or a reset when the supply voltage drops below the LVD voltage. If the LVD is enabled in stop by setting the LVDE and the LVDSE bits, 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 instead enter stop3. [Table 3-3](#) summarizes the behavior of the MCU in stop when the LVD is enabled.

**Table 3-3. LVD Enabled Stop Mode Behavior**

Mode	PPDC	CPU, Digital Peripherals, Flash	RAM	ICG	ADC	Regulator	I/O Pins	RTI
Stop3	0	Standby	Standby	Off	Optionally on	Active	States held	Optionally on

### 3.5.5 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 are kept alive to the background debug logic, clocks to the peripheral systems are halted to reduce power consumption. Refer to [Section 3.5.2, “Stop3 Mode”](#) for specific information on system behavior in stop modes.

#### I/O Pins

- All I/O pin states remain unchanged when the MCU enters stop3 mode.
- If the MCU is configured to go into stop2 mode, all I/O pins states are latched before entering stop.

#### Memory

- All RAM and register contents are preserved while the MCU is in stop3 mode.
- All registers will be reset upon wake-up from stop2, but the contents of RAM are preserved and pin states remain latched until the PPDACK bit is written. The user may save any memory-mapped register data into RAM before entering stop2 and restore the data upon exit from stop2.
- The contents of the flash memory are non-volatile and are preserved in any of the stop modes.

**ICG** — In stop3 mode, the ICG enters its low-power standby state. The oscillator may be kept running when the ICG is in standby by setting OSCSTEN. In stop2 mode, the ICG is turned off. The oscillator cannot be kept running in stop2 even if OSCSTEN is set. If the MCU is configured to go into stop2 mode, the ICG will be reset upon wake-up from stop and must be reinitialized.

**TPM** — When the MCU enters stop mode, the clock to the TPM1 and TPM2 modules stop. The modules halt operation. If the MCU is configured to go into stop2 mode, the TPM modules will be reset upon wake-up from stop and must be reinitialized.

**ADC** — When the MCU enters stop mode, the ADC will enter a low-power standby state unless the asynchronous clock source, ADACK, is enabled. Conversions can occur in stop3 if ADACK is enabled. If the MCU is configured to go into stop2 mode, the ADC will be reset upon wake-up from stop and must be re-initialized.

**SCI** — When the MCU enters stop mode, the clocks to the SCI1 and SCI2 modules stop. The modules halt operation. If the MCU is configured to go into stop2 mode, the SCI modules will be reset upon wake-up from stop and must be reinitialized.

**Voltage Regulator** — The voltage regulator enters a low-power standby state when the MCU enters either of the stop modes unless the LVD is enabled in stop mode or BDM is enabled.

# Chapter 4 Memory

## 4.1 SC9S08MZ16 Memory Map

Figure 4-1 shows the memory maps for the SC9S08MZ16 MCU. On-chip memory in the SC9S08MZ16 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 0x006F)
- High-page registers (0x1800 through 0x185F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

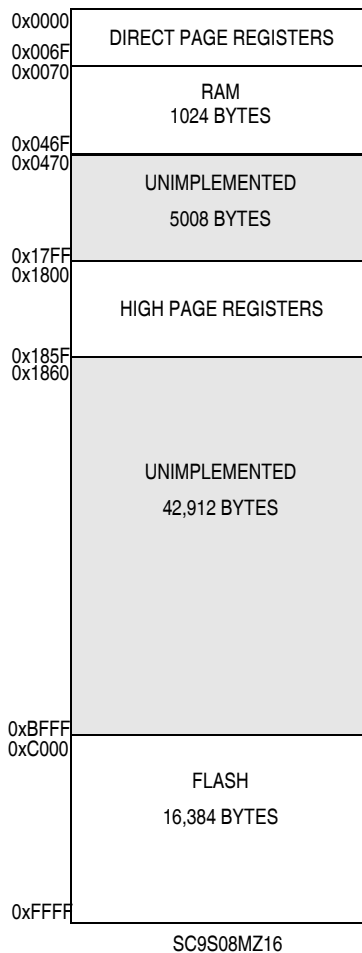


Figure 4-1. SC9S08MZ16 Memory Map

### 4.1.1 Reset and Interrupt Vector Assignments

Figure 4-1 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the Freescale-provided equate file for the SC9S08MZ16. 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:FFC1 through 0xFFC4:FFC5	Unused Vector Space (available for user program)	—
0xFFC6:FFC	TPM3 overflow	Vtpm3ovf
0xFFC8:FFC9	TPM3 channel 1	Vtpm3ch1
0xFFCA:FFCB	TPM3 channel 0	Vtpm3ch0
0xFFCC:FFCD	RTI	Vrti
0xFFCE:FFCF	Unused	—
0xFFD0:FFD1	ADC Conversion	Vadc1
0xFFD2:FFD3	Unused	—
0xFFD4:FFD5	SCI2 Transmit	Vsci2tx
0xFFD6:FFD7	SCI2 Receive	Vsci2rx
0xFFD8:FFD9	SCI2 Error	Vsci2err
0xFFDA:FFDB	SCI1 Transmit	Vsci1tx
0xFFDC:FFDD	SCI1 Receive	Vsci1rx
0xFFDE:FFDF	SCI1 Error	Vsci1err
0xFFE0:FFE1	Unused	—
0xFFE2:FFE3	TPM2 Overflow	Vtpm2ovf
0xFFE4:FFE5	TPM2 Channel 1	Vtpm2ch1
0xFFE6:FFE7	TPM2 Channel 0	Vtpm2ch0
0xFFE8:FFE9	TPM1 Overflow	Vtpm1ovf
0xFFEA:FFEB	Unused	—
0xFFEC:FFED	Unused	—
0xFFEE:FFEF	TPM1 Channel 3	Vtpm1ch3
0xFFFF0:FFF1	TPM1 Channel 2	Vtpm1ch2
0xFFFF2:FFF3	TPM1 Channel 1	Vtpm1ch1
0xFFFF4:FFF5	TPM1 Channel 0	Vtpm1ch0
0xFFFF6:FFF7	ICG	Vicg
0xFFFF8:FFF9	Low Voltage Detect	Vlvd
0xFFFFA:FFFB	IRQ	Virq
0xFFFFC:FFFD	SWI	Vswi
0xFFFFE:FFFF	Reset	Vreset

## 4.2 Register Addresses and Bit Assignments

The registers in the SC9S08MZ16 are divided into these three groups:

- Direct-page registers are located in the first 112 locations in the memory map, so they are accessible with efficient direct addressing mode instructions.
- High-page registers are used much less often, so they are located above 0x1800 in the memory map. This leaves more room in the direct page for more frequently used registers and variables.
- The nonvolatile register area consists of a block of 16 locations in flash memory at 0xFFB0–0xFFBF.

Nonvolatile register locations include:

- Three values which are loaded into working registers at reset
- An 8-byte backdoor comparison key which optionally allows a user to gain controlled access to secure memory

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

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

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

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000	PTAD	R	R	R	R	R	R	PTAD1	PTAD0
0x0001	PTADD	R	R	R	R	R	R	PTADD1	PTADD0
0x0002	PTBD	R	R	R	R	PTBD3	PTBD2	PTBD1	PTBD0
0x0003	PTBDD	R	R	R	R	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x0004	PTCD	0	R	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
0x0005	PTCDD	0	R	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
0x0006	PTDD	R	R	R	R	PTDD3	PTDD2	PTDD1	PTDD0
0x0007	PTDDD	R	R	R	R	PTDDD3	PTDDD2	PTDDD1	PTDDD0
0x0008	PTED	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
0x0009	PTEDD	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
0x000A	PTFD	R	R	PTFD5	PTFD4	R	R	PTFD1	PTFD0
0x000B	PTFDD	R	R	PTFDD5	PTFDD4	R	R	PTFDD1	PTFDD0
0x000C	PTGD	0	PTGD6	PTGD5	0	PTGD3	PTGD2	PTGD1	PTGD0
0x000D	PTGDD	0	PTGDD6	PTGDD5	0	PTGDD3	PTGDD2	PTGDD1	PTGDD0
0x000E– 0x000F	Reserved	—	—	—	—	—	—	—	—
0x0010	ADCSC1	COCO	AIEN	ADCO	ADCH				
0x0011	ADCSC2	ADACT	ADTRG	ACFE	ACFGT	0	0	R	R
0x0012	ADCRH	0	0	0	0	0	0	0	0
0x0013	ADCRL	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
0x0014	ADCCVH	0	0	0	0	—	—	0	0
0x0015	ADCCVL	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
0x0016	ADCCFG	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x0017	APCTL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x0018	APCTL2	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
0x0019	APCTL3	ADPC23	ADPC22	ADPC21	ADPC20	ADPC19	ADPC18	ADPC17	ADPC16
0x001A– 0x001B	Reserved	—	—	—	—	—	—	—	—
0x001C	IRQSC	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x001D	Reserved	—	—	—	—	—	—	—	—
0x001E	Reserved	—	—	—	—	—	—	—	—
0x001F	Reserved	—	—	—	—	—	—	—	—
0x0020	TPM1SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0021	TPM1CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0022	TPM1CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0023	TPM1MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0024	TPM1MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0025	TPM1C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0026	TPM1C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0027	TPM1C0VL	Bit 7	6	5	4	3	2	1	Bit 0



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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0028	TPM1C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0029	TPM1C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x002A	TPM1C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x002B	TPM1C2SC	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x002C	TPM1C2VH	Bit 15	14	13	12	11	10	9	Bit 8
0x002D	TPM1C2VL	Bit 7	6	5	4	3	2	1	Bit 0
0x002E	TPM1C3SC	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	0	0
0x002F	TPM1C3VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0030	TPM1C3VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0031– 0x0037	Reserved	—	—	—	—	—	—	—	—
0x0038	SCI1BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0039	SCI1BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x003A	SCI1C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x003B	SCI1C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x003C	SCI1S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x003D	SCI1S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x003E	SCI1C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x003F	SCI1D	Bit 7	6	5	4	3	2	1	Bit 0
0x0040	SCI2BDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0041	SCI2BDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x0042	SCI2C1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x0043	SCI2C2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x0044	SCI2S1	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
0x0045	SCI2S2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x0046	SCI2C3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x0047	SCI2D	Bit 7	6	5	4	3	2	1	Bit 0
0x0048	ICGC1	HGO	RANGE	REFS	CLKS		OSCSTEN	LOCD	0
0x0049	ICGC2	LOLRE	MFD			LOCRE	RFD		
0x004A	ICGS1	CLKST		REFST	LOLS	LOCK	LOCS	ERCS	ICGIF
0x004B	ICGS2	0	0	0	0	0	0	0	DCOS
0x004C	ICGFLTU	0	0	0	0	FLT			
0x004D	ICGFLTL	FLT							
0x004E	ICGTRM	TRIM							
0x004F– 0x005F	Reserved	—	—	—	—	—	—	—	—
0x0060	TPM2SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0061	TPM2CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0062	TPM2CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0063	TPM2MODH	Bit 15	14	13	12	11	10	9	Bit 8

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0064	TPM2MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0065	TPM2C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0066	TPM2C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0067	TPM2C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0068	TPM2C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0069	TPM2C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x006A	TPM2C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x006B– 0x006F	Reserved	—	—	—	—	—	—	—	—

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

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	ICG	LVD	0
0x1801	SBDFR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT	COPE	COPT	STOPE	—	0	0	—	—
0x1803	SMCLK	0	0	0	MPE	0	MCSEL		
0x1804– 0x1805	Reserved	—	—	—	—	—	—	—	—
0x1806	SDIDH	—	—	—	—	ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	SRTISC	RTIF	RTIACK	RTICLKS	RTIE	0	RTIS2	RTIS1	RTIS0
0x1809	SPMSC1	LVDF	LVDACK	LVDIE	LVDRE	LVDSE	LVDE	0 <sup>1</sup>	BGBE
0x180A	SPMSC2	LVWF	LVWACK	LVDV	LVWV	PPDF	PPDACK	—	PPDC
0x180B	Reserved	—	—	—	—	—	—	—	—
0x180C	SOPT2	COPCLKS	0	0	0	0	0	0	0
0x180D– 0x180F	Reserved	—	—	—	—	—	—	—	—
0x1810	DBGCAH	Bit 15	14	13	12	11	10	9	Bit 8
0x1811	DBGCAL	Bit 7	6	5	4	3	2	1	Bit 0
0x1812	DBGCBH	Bit 15	14	13	12	11	10	9	Bit 8
0x1813	DBGCBL	Bit 7	6	5	4	3	2	1	Bit 0
0x1814	DBGFH	Bit 15	14	13	12	11	10	9	Bit 8
0x1815	DBGFL	Bit 7	6	5	4	3	2	1	Bit 0
0x1816	DBGC	DBGGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
0x1817	DBGT	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
0x1818	DBGS	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
0x1819– 0x181F	Reserved	—	—	—	—	—	—	—	—

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
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– 0x182F	Reserved	—	—	—	—	—	—	—	—
0x1830	TPM3SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x1831	TPM3CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x1832	TPM3CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x1833	TPM3MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x1834	TPM3MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x1835	TPM3C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x1836	TPM3C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x1837	TPM3C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x1838	TPM3C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x1839	TPM3C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x183A	TPM3C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x183B– 0x183F	Reserved	—	—	—	—	—	—	—	—
0x1840	PTAPE	R	R	R	R	R	R	PTAPE1	PTAPE0
0x1841	PTASE	R	R	R	R	R	R	PTASE1	PTASE0
0x1842	PTADS	R	R	R	R	R	R	PTADS1	PTADS0
0x1843	Reserved	—	—	—	—	—	—	—	—
0x1844	PTBPE	R	R	R	R	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x1845	PTBSE	R	R	R	R	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x1846	PTBDS	R	R	R	R	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x1847	Reserved	—	—	—	—	—	—	—	—
0x1848	PTCPE	0	R	PTCPE5	PTCPE4	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x1849	PTCSE	0	R	PTCSE5	PTCSE4	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x184A	PTCDS	0	R	PTCDS5	PTCDS4	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x184B	Reserved	—	—	—	—	—	—	—	—
0x184C	PTDPE	R	R	R	R	PTDPE3	PTDPE2	PTDPE1	PTDPE0
0x184D	PTDSE	R	R	R	R	PTDSE3	PTDSE2	PTDSE1	PTDSE0
0x184E	PTDDS	R	R	R	R	PTDDS3	PTDDS2	PTDDS1	PTDDS0
0x184F	Reserved	—	—	—	—	—	—	—	—
0x1850	PTEPE	PTEPE7	PTEPE6	PTEPE5	PTEPE4	PTEPE3	PTEPE2	PTEPE1	PTEPE0
0x1851	PTESE	PTESE7	PTESE6	PTESE5	PTESE4	PTESE3	PTESE2	PTESE1	PTESE0

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1852	PTEDS	PTEDS7	PTEDS6	PTEDS5	PTEDS4	PTEDS3	PTEDS2	PTEDS1	PTEDS0
0x1853	Reserved	—	—	—	—	—	—	—	—
0x1854	PTFPE	R	R	PTFPE5	PTFPE4	R	R	PTFPE1	PTFPE0
0x1855	PTFSE	R	R	PTFSE5	PTFSE4	R	R	PTFSE1	PTFSE0
0x1856	PTFDS	R	R	PTFDS5	PTFDS4	R	R	PTFDS1	PTFDS0
0x1857	Reserved	—	—	—	—	—	—	—	—
0x1858	PTGPE	0	PTGPE6	PTGPE5	0	PTGPE3	PTGPE2	PTGPE1	PTGPE0
0x1859	PTGSE	0	PTGSE6	PTGSE5	0	PTGSE3	PTGSE2	PTGSE1	PTGSE0
0x185A	PTGDS	0	PTGDS6	PTGDS5	0	PTGDS3	PTGDS2	PTGDS1	PTGDS0
0x185B– 0x185F	Reserved	—	—	—	—	—	—	—	—

<sup>1</sup> This reserved bit must always be written to 0.

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

Table 4-4. Nonvolatile Register Summary

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0xFFB0– 0xFFB7	NVBACKKEY	8-Byte Comparison Key							
0xFFB8– 0xFFBC	Reserved	—	—	—	—	—	—	—	—
0xFFBD	NVPROT	FPS7	FPS6	FPS5	FPS4	FPS3	FPS2	FPS1	FPDIS
0xFFBE	Reserved <sup>1</sup>	—	—	—	—	—	—	—	—
0xFFBF	NVOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00

<sup>1</sup> This location can be used to store the factory trim value for the ICG.

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

## 4.3 RAM

The SC9S08MZ16 includes static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit

manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

The RAM retains data when the MCU is in low-power 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 SC9S08MZ16, it is usually best to re-initialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM in the Freescale-provided equate file).

---

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

---

When security is enabled, the RAM is considered a secure memory resource and is not accessible through BDM or through code executing from non-secure memory. See [Section 4.5, “Security”](#) for a detailed description of the security feature.

## 4.4 Flash

The flash memory is intended primarily for program storage. In-circuit programming allows the operating program to be loaded into the flash memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because no special voltages are needed for flash erase and programming operations, in-application programming is also possible through other software-controlled communication paths. 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.

### 4.4.1 Features

Features of the flash memory include:

- Flash Size — 16,384 bytes (32 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

## 4.4.2 Program and Erase Times

Before any program or erase command can be accepted, the flash clock divider register (FCDIV) must be written to set the internal clock for the flash module to a frequency ( $f_{\text{FCLK}}$ ) between 150 kHz and 200 kHz (see Section 4.6.1, “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 are used by the command processor to complete a program or erase command.

Table 4-5 shows program and erase times. The bus clock frequency and FCDIV determine the frequency of FCLK ( $f_{\text{FCLK}}$ ). 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 times shown include overhead for the command state machine and enabling and disabling of program and erase voltages.

**Table 4-5. Program and Erase Times**

Parameter	Cycles of FCLK	Time if FCLK = 200 kHz
Byte program	9	45 $\mu\text{s}$
Byte program (burst)	4	20 $\mu\text{s}$ <sup>1</sup>
Page erase	4000	20 ms
Mass erase	20,000	100 ms

<sup>1</sup> Excluding start/end overhead

## 4.4.3 Program and Erase Command Execution

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

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

### NOTE

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

- 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.
- Write a 1 to the FCBEF bit in FSTAT to clear FCBEF and launch the command (including its address and data information).

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

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

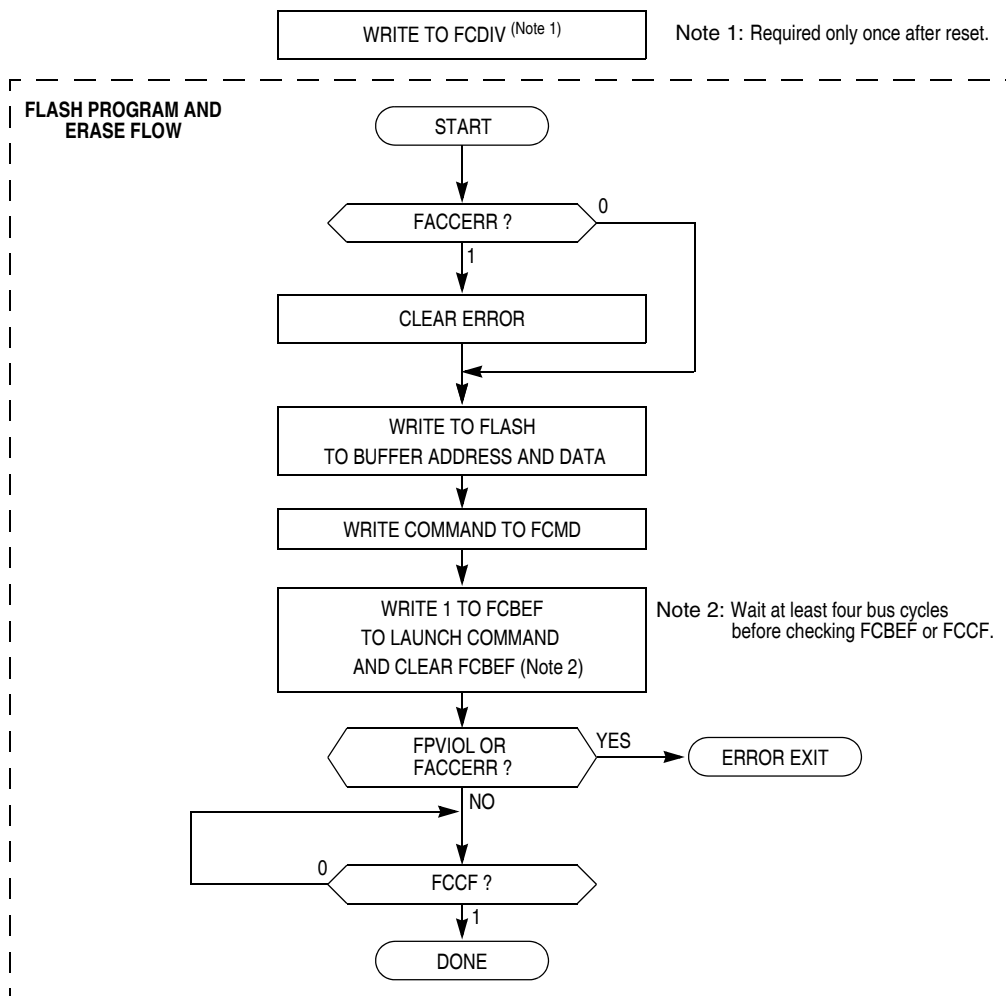


Figure 4-2. Flash Program and Erase Flowchart

#### 4.4.4 Burst Program Execution

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

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

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



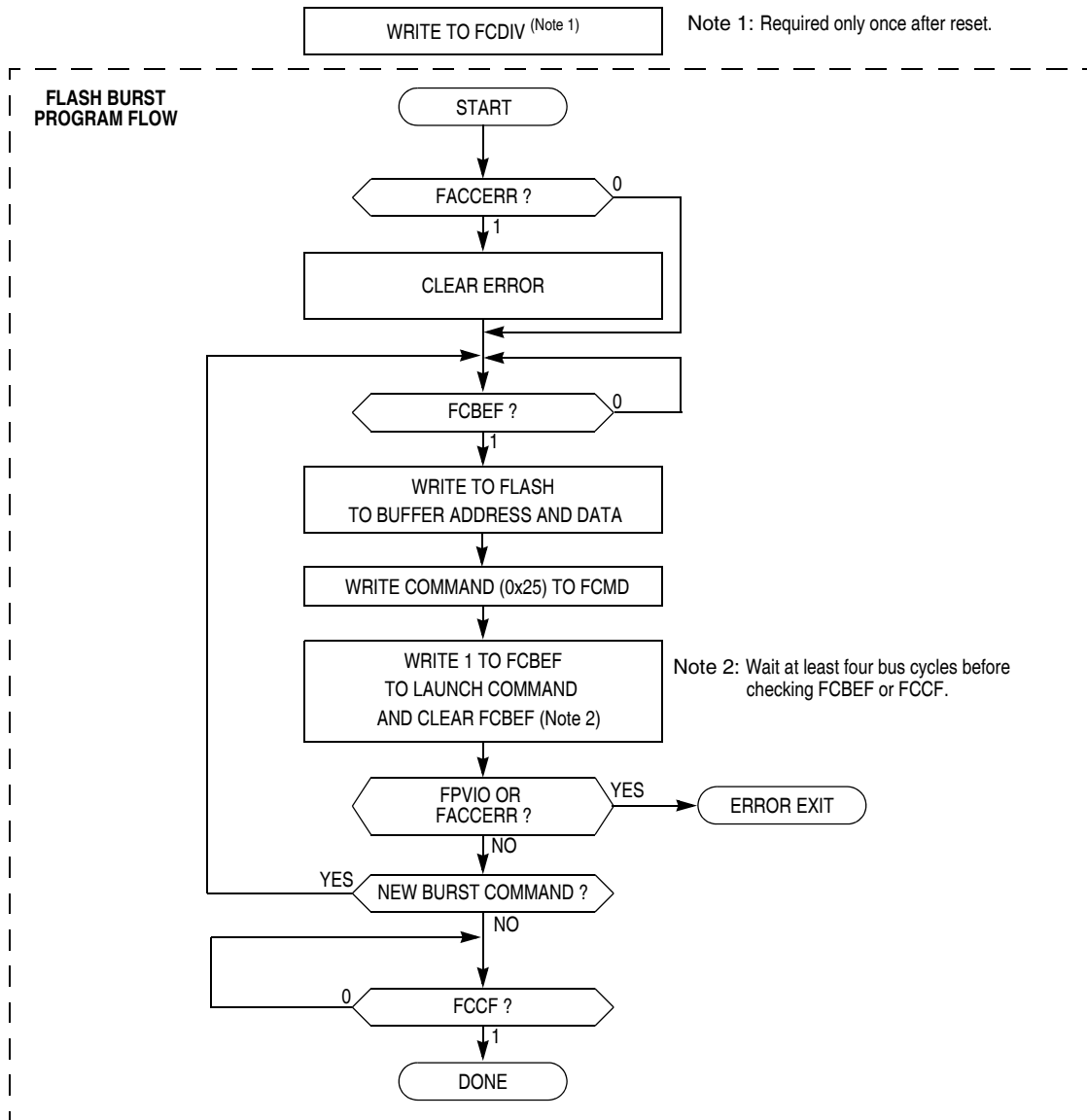


Figure 4-3. Flash Burst Program Flowchart

### 4.4.5 Access Errors

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

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

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

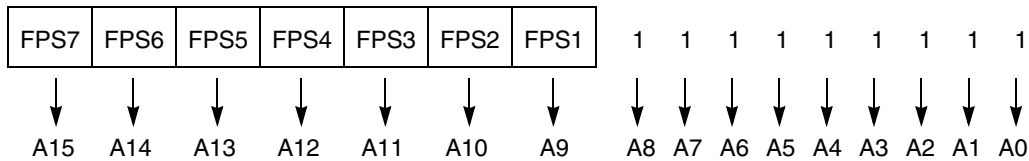
### 4.4.6 Flash Block Protection

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 4.6.4, “Flash Protection Register \(FPROT and NVPROT\)”](#)).

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

The block protection mechanism is illustrated in [Figure 4-4](#). The FPS bits are used as the upper bits of the last address of unprotected memory. This address is formed by concatenating FPS7:FPS1 with logic 1 bits as shown. For example, 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 4-4. Block Protection Mechanism**

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

#### 4.4.7 Vector Redirection

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

For example, if 512 bytes of flash are protected, the protected address region is from 0xFE00 through 0xFFFF. The interrupt vectors (0xFFC0–0xFFFFD) are redirected to the locations 0xFDC0–0xFDFD. Now, if an RTI interrupt is taken for instance, the values in the locations 0xFDCC:FDCD are used for the vector instead of the values in the locations 0xFFCC:FFCD. 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.

### 4.5 Security

The SC9S08MZ16 includes circuitry to prevent unauthorized access to the contents of flash and RAM memory. When security is engaged, flash and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources.

Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

Security is engaged or disengaged based on the state of two nonvolatile register bits (SEC01:SEC00) in the FOPT register. During reset, the contents of the nonvolatile location NVOPT are copied from flash into the working FOPT register in high-page register space. A user engages security by programming the NVOPT location which can be done at the same time the flash memory is programmed. The 1:0 state disengages security and the other three combinations engage security. Notice the erased state (1:1) makes

the MCU secure. During development, whenever the flash is erased, it is good practice to immediately program the SEC00 bit to 0 in NVOPT so SEC01:SEC00 = 1:0. This would allow the MCU to remain unsecured after a subsequent reset.

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

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

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

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

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

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

1. Disable any block protections by writing FPROT. FPROT can be written only with background debug commands, not from application software.
2. Mass erase flash if necessary.
3. Blank check flash. Provided flash is completely erased, security is disengaged until the next reset. To avoid returning to secure mode after the next reset, program NVOPT so SEC01:SEC00 = 1:0.

## 4.6 Flash Registers and Control Bits

The flash module has nine 8-bit registers in the high-page register space, three locations in the nonvolatile register space in flash memory which are copied into three corresponding high-page control registers at

reset. There is also an 8-byte comparison key in flash memory. Refer to [Table 4-3](#) and [Table 4-4](#) for the absolute address assignments for all flash registers. This section refers to registers and control bits only by their names. A Freescale-provided equate or header file normally is used to translate these names into the appropriate absolute addresses.

### 4.6.1 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.

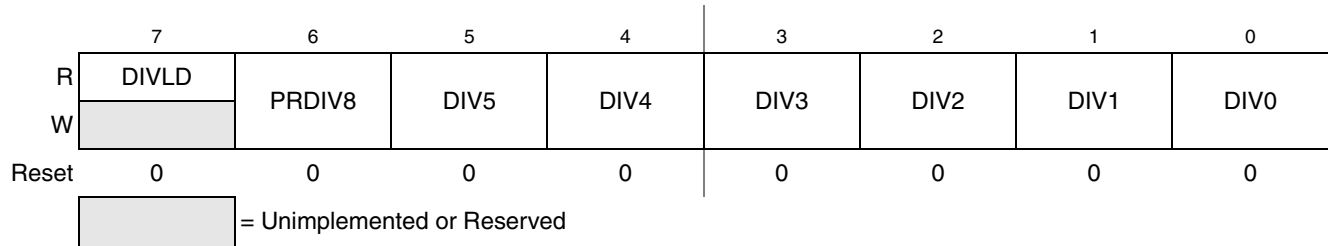


Figure 4-5. Flash Clock Divider Register (FCDIV)

Table 4-6. FCDIV Register Field Descriptions

Field	Description
7 DIVLD	<b>Divisor Loaded Status Flag</b> — When set, this read-only status flag indicates that the FCDIV register has been written since reset. Reset clears this bit and the first write to this register causes this bit to become set regardless of the data written. 0 FCDIV has not been written since reset; erase and program operations disabled for flash. 1 FCDIV has been written since reset; erase and program operations enabled for flash.
6 PRDIV8	<b>Prescale (Divide) Flash Clock by 8</b> 0 Clock input to the flash clock divider is the bus rate clock. 1 Clock input to the flash clock divider is the bus rate clock divided by 8.
5:0 DIV[5:0]	<b>Divisor for Flash Clock Divider</b> — 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 $\mu$ s to 6.7 $\mu$ s. The automated programming logic uses an integer number of these pulses to complete an erase or program operation. See <a href="#">Equation 4-1</a> , <a href="#">Equation 4-2</a> , and <a href="#">Table 4-6</a> .

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

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

[Table 4-7](#) shows the appropriate values for PRDIV8 and DIV5:DIV0 for selected bus frequencies.

Table 4-7. Flash Clock Divider Settings

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

## 4.6.2 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.

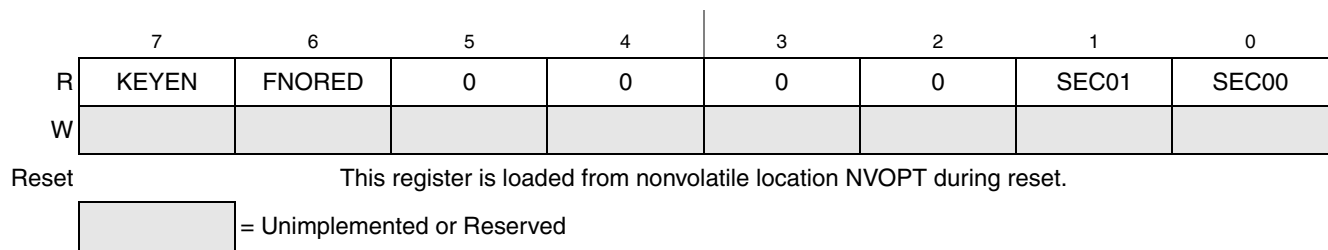


Figure 4-6. Flash Options Register (FOPT)

Table 4-8. FOPT Register Field Descriptions

Field	Description
7 KEYEN	<p><b>Backdoor Key Mechanism Enable</b> — 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 <a href="#">Section 4.5, “Security.”</a></p> <p>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.</p>
6 FNORED	<p><b>Vector Redirection Disable</b> — When this bit is 1, then vector redirection is disabled.</p> <p>0 Vector redirection enabled. 1 Vector redirection disabled.</p>
1:0 SEC0[1:0]	<p><b>Security State Code</b> — This 2-bit field determines the security state of the MCU as shown in <a href="#">Table 4-9</a>. 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 <a href="#">Section 4.5, “Security.”</a></p>

Table 4-9. Security States

SEC01:SEC00	Description
0:0	Secure
0:1	Secure
1:0	Unsecured
1:1	Secure

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

### 4.6.3 Flash Configuration Register (FCNFG)

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

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


 = Unimplemented or Reserved

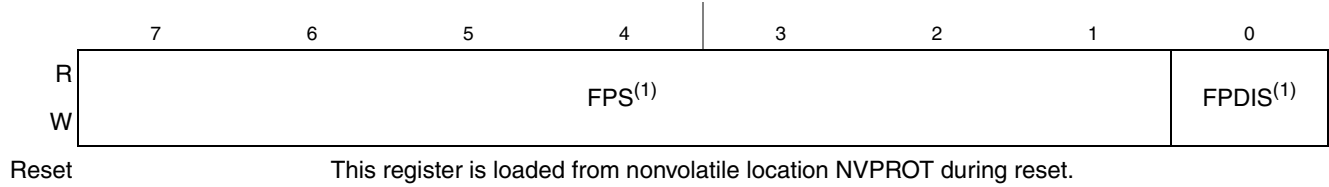
Figure 4-7. Flash Configuration Register (FCNFG)

Table 4-10. FCNFG Register Field Descriptions

Field	Description
5 KEYACC	<b>Enable Writing of Access Key</b> — This bit enables writing of the backdoor comparison key. For more detailed information about the backdoor key mechanism, refer to <a href="#">Section 4.5, “Security.”</a> 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.

### 4.6.4 Flash Protection Register (FPROT and NVPROT)

During reset, the contents of the nonvolatile location NVPROT are copied from flash into FPROT. This register can be read at any time. If FPDIS = 0, protection can be increased, i.e., a smaller value of FPS can be written. If FPDIS = 1, writes do not change protection.



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

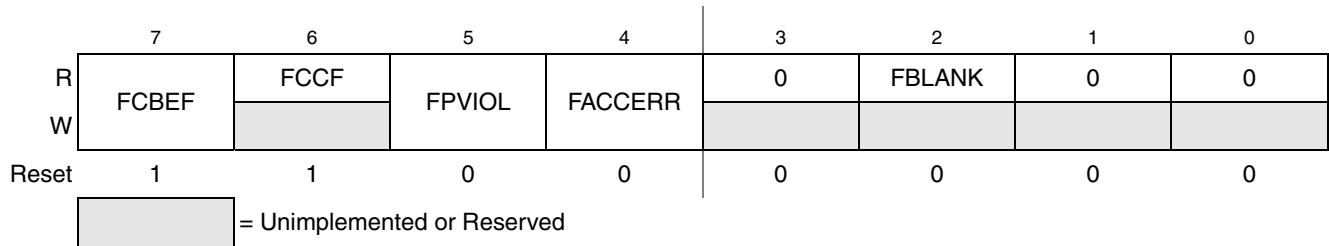
**Figure 4-8. Flash Protection Register (FPROT)**

**Table 4-11. FPROT Register Field Descriptions**

Field	Description
7:1 FPS[7:1]	<b>Flash Protect Select Bits</b> — 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	<b>Flash Protection Disable</b> 0 Flash block specified by FPS[7:1] is block protected (program and erase not allowed). 1 No flash block is protected.

### 4.6.5 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 4-9. Flash Status Register (FSTAT)**



Table 4-12. FSTAT Register Field Descriptions

Field	Description
7 FCBEF	<b>Flash Command Buffer Empty Flag</b> — The FCBEF bit is used to launch commands. It also indicates that the command buffer is empty so that a new command sequence can be executed when performing burst programming. The FCBEF bit is cleared by writing a one to it or when a burst program command is transferred to the array for programming. Only burst program commands can be buffered. 0 Command buffer is full (not ready for additional commands). 1 A new burst program command may be written to the command buffer.
6 FCCF	<b>Flash Command Complete Flag</b> — 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	<b>Protection Violation Flag</b> — FPVIOL is set automatically when FCBEF is cleared to register a command that attempts to erase or program a location in a protected block (the erroneous command is ignored). FPVIOL is cleared by writing a 1 to FPVIOL. 0 No protection violation. 1 An attempt was made to erase or program a protected location.
4 FACCERR	<b>Access Error Flag</b> — 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 <a href="#">Section 4.4.5, "Access Errors."</a> 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	<b>Flash Verified as All Blank (erased) Flag</b> — 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).

## 4.6.6 Flash Command Register (FCMD)

Only five command codes are recognized in normal user modes as shown in [Table 4-14](#). Refer to [Section 4.4.3, “Program and Erase Command Execution”](#) for a detailed discussion of flash programming and erase operations.

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

**Figure 4-10. Flash Command Register (FCMD)**

**Table 4-13. FCMD Register Field Descriptions**

Field	Description
7:0 FCMD[7:0]	<b>Flash Command Bits</b> — See <a href="#">Table 4-14</a>

**Table 4-14. Flash Commands**

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

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

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

# Chapter 5

## Resets, Interrupts, and System Configuration

### 5.1 Introduction

This chapter discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the SC9S08MZ16. Some interrupt sources from peripheral modules are discussed in greater detail within other chapters of this data manual. This chapter gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog and real-time interrupt (RTI), 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:
  - Power-on detection (POR)
  - Low voltage detection (LVD) with enable
  - External  $\overline{\text{RESET}}$  pin
  - COP watchdog with enable and two timeout choices
  - Illegal opcode
  - Illegal address
  - Serial command from a background debug host
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vectors for each module (reduces polling overhead) (see [Table 5-11](#))

### 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 (0xFFFF: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 following sources of reset are available on the SC9S08MZ16:

- Power-on reset (POR)
- Low-voltage detect (LVD)

- Computer operating properly (COP) timer
- Illegal opcode detect
- Illegal address detect
- Background debug forced reset
- The reset pin ( $\overline{\text{RESET}}$ )
- Clock generator loss of lock and loss of clock reset

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

## 5.4 Computer Operating Properly (COP) Watchdog

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

After any reset, the COPE becomes set in SOPT enabling the COP watchdog (see [Section 5.9.4, “System Options Register \(SOPT\),”](#) for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPE. The COP counter is reset by writing any value to the address of SRS. This write does not affect the data in the read-only SRS. Instead, the act of writing to this address is decoded and sends a reset signal to the COP counter.

The COPCLKS bit in SOPT2 (see [Section 5.9.10, “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 is an associated short and long time-out controlled by COPT in SOPT. [Table 5-1](#) summarizes the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the bus clock source and the associated long time-out ( $2^{18}$  cycles).

**Table 5-1. COP Configuration Options**

Control Bits		Clock Source	COP Overflow Count
COPCLKS	COPT		
0	0	~1 kHz	$2^5$ cycles (32 ms) <sup>1</sup>
0	1	~1 kHz	$2^8$ cycles (256 ms) <sup>1</sup>
1	0	Bus	$2^{13}$ cycles
1	1	Bus	$2^{18}$ cycles

<sup>1</sup> Values are shown in this column based on  $t_{RTI} = 1$  ms. See  $t_{RTI}$  in the appendix [Section A.9.1, “Control Timing,”](#) for the tolerance of this value.

Even if the application will use the reset default settings of COPE, COPCLKS, and COPT, the user must write to the write-once SOPT and SOPT2 registers during reset initialization to lock in the settings. That way, they cannot be changed accidentally if the application program gets lost. The initial writes to SOPT and SOPT2 will reset the COP counter.

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.

In background debug mode, the COP counter will not increment.

When the bus clock source is selected, the COP counter does not increment while the system is in stop mode. The COP counter resumes as soon as the MCU exits stop mode.

When the 1 kHz clock source is selected, the COP counter is re-initialized to zero upon entry to stop mode. The COP counter begins from zero after the MCU exits stop mode.

## 5.5 Interrupts

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

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

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

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

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

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

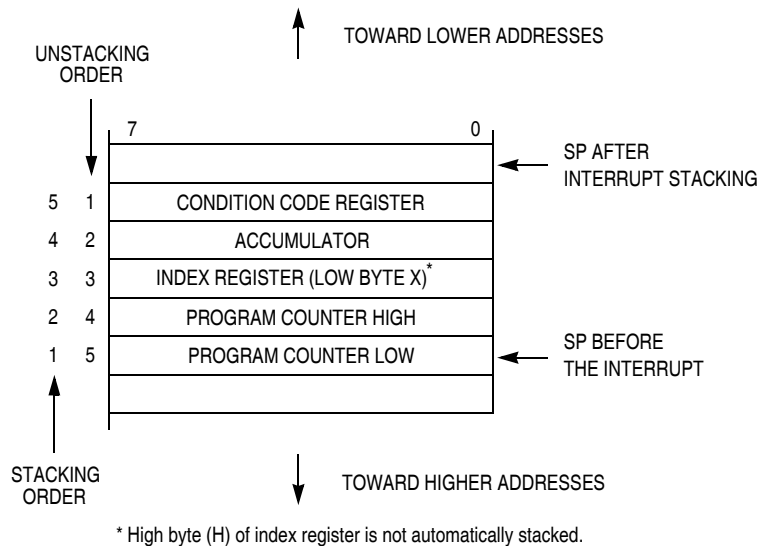
**NOTE**

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

When two or more interrupts are pending when the I bit is cleared, the highest priority source is serviced first (see Table 5-2).

**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 should 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 IRQ status and control register, IRQSC. 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 IRQ Pin Configuration Options

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

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

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

#### NOTE

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

#### 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 reconfigures the detection logic so it detects edge events and pin levels. In the edge and level 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 5-2 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 5-2. Vector Summary

Vector Priority	Vector No.	Address (High/Low)	Vector Name	Module	Source	Enable	Description	
Lower   Higher	29 – 31	0xFFC0/FFC1 – 0xFFC4/0xFFC5	Unused vector space (available for user program)					
	28	0xFFC6/FFC7	Vtpm3ovf	TPM3	TOF	TOIE	TPM3 overflow	
	27	0xFFC8/FFC9	Vtpm3ch1	TPM3	CH1F	CH1IE	TPM3 channel 1	
	26	0xFFCA/FFCB	Vtpm3ch0	TPM3	CH0F	CH0IE	TPM3 channel 0	
	25	0xFFCC/FFCD	Vrti	System control	RTIF	RTIE	Real-time interrupt	
	24	0xFFCE/FFCF	Unused vector space					
	23	0xFFD0/FFD1	Vadc	ADC	COCO	AIEN	ADC	
	22	0xFFD2/FFD3	Unused vector space					
	21	0xFFD4/FFD5	Vsci2tx	SCI2	TDRE TC	TIE TCIE	SCI2 transmit	
	20	0xFFD6/FFD7	Vsci2rx	SCI2	IDLE RDRF	ILIE RIE	SCI2 receive	
	19	0xFFD8/FFD9	Vsci2err	SCI2	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI2 error	
	18	0xFFDA/FFDB	Vsci1tx	SCI1	TDRE TC	TIE TCIE	SCI1 transmit	
	17	0xFFDC/FFDD	Vsci1rx	SCI1	IDLE RDRF	ILIE RIE	SCI1 receive	
	16	0xFFDE/FFDF	Vsci1err	SCI1	OR NF FE PF	ORIE NFIE FEIE PFIE	SCI1 error	
	15	0xFFE0/FFE1	Unused vector space					
	14	0xFFE2/FFE3	Vtpm2ovf	TPM2	TOF	TOIE	TPM2 overflow	
	13	0xFFE4/FFE5	Vtpm2ch1	TPM2	CH1F	CH1IE	TPM2 channel 1	
	12	0xFFE6/FFE7	Vtpm2ch0	TPM2	CH0F	CH0IE	TPM2 channel 0	
	11	0xFFE8/FFE9	Vtpm1ovf	TPM1	TOF	TOIE	TPM1 overflow	
	10	0xFFEA/FFEB	Unused vector space					
	9	0xFFEC/FFED	Unused vector space					
	8	0xFFEE/FFEF	Vtpm1ch3	TPM1	CH3F	CH3IE	TPM1 channel 3	
	7	0xFFFF0/FFF1	Vtpm1ch2	TPM1	CH2F	CH2IE	TPM1 channel 2	
	6	0xFFFF2/FFF3	Vtpm1ch1	TPM1	CH1F	CH1IE	TPM1 channel 1	
	5	0xFFFF4/FFF5	Vtpm1ch0	TPM1	CH0F	CH0IE	TPM1 channel 0	
	4	0xFFFF6/FFF7	Vicg	ICG	ICGIF (LOLS/LOCS)	LOLRE/LOCRE	ICG	
	3	0xFFFF8/FFF9	Vlvd	System control	LVDF	LVDIE	Low-voltage detect	
	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	COPE LVDRE — —	Watchdog timer Low-voltage detect External pin Illegal opcode	



## 5.6 Low-Voltage Detect (LVD) System

The SC9S08MZ16 includes a system to protect against low voltage conditions in order to protect memory contents and control MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and an LVD circuit with a user selectable trip voltage, either high ( $V_{LV\text{DH}}$ ) or low ( $V_{LV\text{DL}}$ ). The LVD circuit is enabled when LVDE in SPMSC1 is high and the trip voltage is selected by LVDV in SPMSC2. The LVD is disabled upon entering any of the stop modes unless the LVDSE bit is set. If LVDSE and LVDE are both set, then the MCU cannot enter stop2, and the current consumption in stop3 with the LVD enabled will be greater.

### 5.6.1 Power-On Reset Operation

When power is initially applied to the MCU, or when the supply voltage drops below the  $V_{\text{POR}}$  level, the POR circuit will cause a reset condition. As the supply voltage rises, the LVD circuit will hold the chip in reset until the supply has risen above the  $V_{LV\text{DL}}$  level. Both the POR bit and the LVD bit in SRS are set following a POR.

### 5.6.2 LVD Reset Operation

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

### 5.6.3 LVD Interrupt Operation

When a low voltage condition is detected and the LVD circuit is configured for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), then LVDF will be set and an LVD interrupt will occur.

### 5.6.4 Low-Voltage Warning (LVW)

The LVD system has a low voltage warning flag to indicate to the user that the supply voltage is approaching, but is still above, the LVD voltage. The LVW does not have an interrupt associated with it. There are two user selectable trip voltages for the LVW, one high ( $V_{LV\text{WH}}$ ) and one low ( $V_{LV\text{WL}}$ ). The trip voltage is selected by LVWV in SPMSC2. Setting the LVW trip voltage equal to the LVD trip voltage is not recommended. Typical use of the LVW would be to select  $V_{LV\text{WH}}$  and  $V_{LV\text{DL}}$ .

## 5.7 Real-Time Interrupt (RTI)

The real-time interrupt function can be used to generate periodic interrupts. The RTI can accept two sources of clocks, the 1 kHz internal clock or an external clock if available. The 1 kHz internal clock source is completely independent of any bus clock source and is used only by the RTI module and, on some MCUs, the COP watchdog. To use an external clock source, it must be available and active. The RTICLKS bit in SRTISC is used to select the RTI clock source.

Either RTI clock source can be used when the MCU is in run or stop3 mode. When using the external oscillator in stop3, it must be enabled in stop (OSCSTEN = 1) and configured for low bandwidth operation (RANGE = 0). Only the internal 1 kHz clock source can be selected to wake the MCU from stop2 mode.

The SRTISC register includes a read-only status flag, a write-only acknowledge bit, and a 3-bit control value (RTIS2:RTIS1:RTIS0) used to disable the clock source to the real-time interrupt or select one of seven wakeup periods. The RTI has a local interrupt enable, RTIE, to allow masking of the real-time interrupt. The RTI can be disabled by writing each bit of RTIS to zeroes, and no interrupts will be generated. See [Section 5.9.7, “System Real-Time Interrupt Status and Control Register \(SRTISC\),”](#) for detailed information about this register.

## 5.8 MCLK Output

The PTC2 pin is shared with the MCLK clock output. Setting the pin enable bit, MPE, causes the PTC2 pin to output a divided version of the internal MCU bus clock. The divide ratio is determined by the MCSEL bits. When MPE is set, the PTC2 pin is forced to operate as an output pin regardless of the state of the port data direction control bit for the pin. If the MCSEL bits are all 0s, the pin is driven low. The slew rate and drive strength for the pin are controlled by PTCSE2 and PTCDS2, respectively. The maximum clock output frequency is limited if slew rate control is enabled, see the electrical chapter for pin rise and fall times with slew rate enabled.

## 5.9 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 SOPT 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.”](#)

## 5.9.1 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.

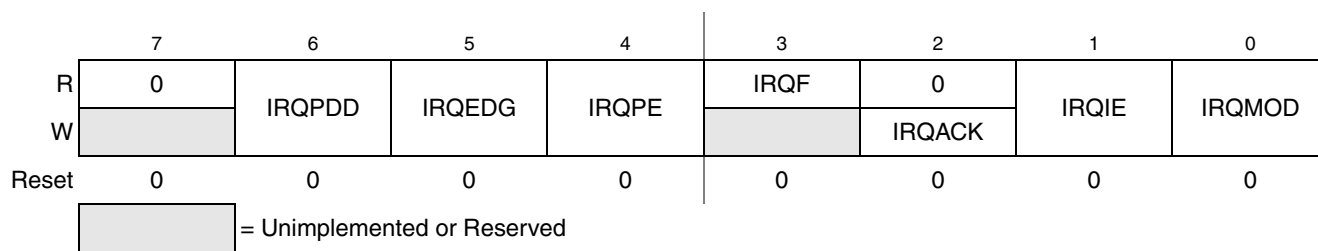


Figure 5-2. Interrupt Request Status and Control Register (IRQSC)

Table 5-3. IRQSC Register Field Descriptions

Field	Description
6 IRQPDD	<b>Interrupt Request (IRQ) Pull Device Disable</b> — This read/write control bit is used to disable the internal pull-up/pull-down 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	<b>Interrupt Request (IRQ) Edge Select</b> — 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, it has a pull-down. When the IRQ pin is enabled as the IRQ input and is configured to detect falling edges, it has a pull-up. 0 IRQ is falling edge or falling edge/low-level sensitive. 1 IRQ is rising edge or rising edge/high-level sensitive.
4 IRQPE	<b>IRQ Pin Enable</b> — This read/write control bit enables the IRQ pin function. When this bit is set the IRQ pin can be used as an interrupt request. 0 IRQ pin function is disabled. 1 IRQ pin function is enabled.
3 IRQF	<b>IRQ Flag</b> — This read-only status bit indicates when an interrupt request event has occurred. 0 No IRQ request. 1 IRQ event detected.
2 IRQACK	<b>IRQ Acknowledge</b> — 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	<b>IRQ Interrupt Enable</b> — 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	<b>IRQ Detection Mode</b> — This read/write control bit selects either edge-only detection or edge-and-level detection. The IRQEDG control bit determines the polarity of edges and levels that are detected as interrupt request events. See <a href="#">Section 5.5.2.2, “Edge and Level Sensitivity”</a> for more details. 0 IRQ event on falling edges or rising edges only. 1 IRQ event on falling edges and low levels or on rising edges and high levels.

## 5.9.2 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 SBDPR register, none of the status bits in SRS will be set. Writing any value to this register 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	ICG	LVD	0
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	Note <sup>(1)</sup>	Note <sup>(1)</sup>	Note <sup>(1)</sup>	0	Note <sup>(1)</sup>	0	0

U = Unaffected by reset

<sup>1</sup> 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 5-4. SRS Register Field Descriptions**

Field	Description
7 POR	<b>Power-On Reset</b> — 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	<b>External Reset Pin</b> — 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	<b>Computer Operating Properly (COP) Watchdog</b> — 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	<b>Illegal Opcode</b> — 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.

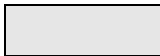
Table 5-4. SRS Register Field Descriptions (continued)

Field	Description
3 ILAD	<p><b>Illegal Address</b> — Reset was caused by an attempt to access a designated illegal address.</p> <p>0 Reset not caused by an illegal address access. 1 Reset caused by an illegal address access.</p> <p>Illegal address areas in the SC9S08MZ16 are: 0x0470 - 0x17FF — Gap from end of RAM to start of high page registers 0x1860 - 0xBFFF — Gap from end of high page registers to start of Flash memory</p> <p>Unused and reserved locations in register areas are not considered illegal addresses and do not trigger illegal address resets.</p>
2 ICG	<p><b>Internal Clock Generation Module Reset</b> — Reset was caused by an ICG module reset.</p> <p>0 Reset not caused by ICG module. 1 Reset caused by ICG module.</p>
1 LVD	<p><b>Low Voltage Detect</b> — If the LVDRE and LVDSE bits are set and the supply drops below the LVD trip voltage, an LVD reset will occur. This bit is also set by POR.</p> <p>0 Reset not caused by LVD trip or POR. 1 Reset caused by LVD trip or POR.</p>

### 5.9.3 System Background Debug Force Reset Register (SBDFR)

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

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								BDFR <sup>1</sup>
Reset	0	0	0	0	0	0	0	0

 = Unimplemented or Reserved

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

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

Table 5-5. SBDFR Register Field Descriptions

Field	Description
0 BDFR	<p><b>Background Debug Force Reset</b> — 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.</p>

### 5.9.4 System Options Register (SOPT)

This register may be read at any time. Bits 3 and 2 are unimplemented and always read 0. This is a write-once register so only the first write after reset is honored. Any subsequent attempt to write to SOPT

(intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. SOPT should be written during the user’s reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

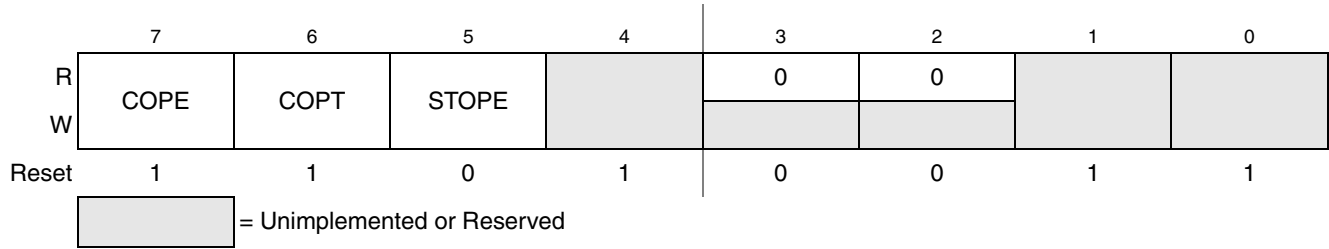


Figure 5-5. System Options Register (SOPT)

Table 5-6. SOPT Register Field Descriptions

Field	Description
7 COPE	<b>COP Watchdog Enable</b> — This write-once bit defaults to 1 after reset. 0 COP watchdog timer disabled. 1 COP watchdog timer enabled (force reset on timeout).
6 COPT	<b>COP Watchdog Timeout</b> — This write-once bit defaults to 1 after reset. 0 Short timeout period selected. 1 Long timeout period selected.
5 STOPE	<b>Stop Mode Enable</b> — 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.

### 5.9.5 System MCLK Control Register (SMCLK)

This register is used to control the MCLK clock output.

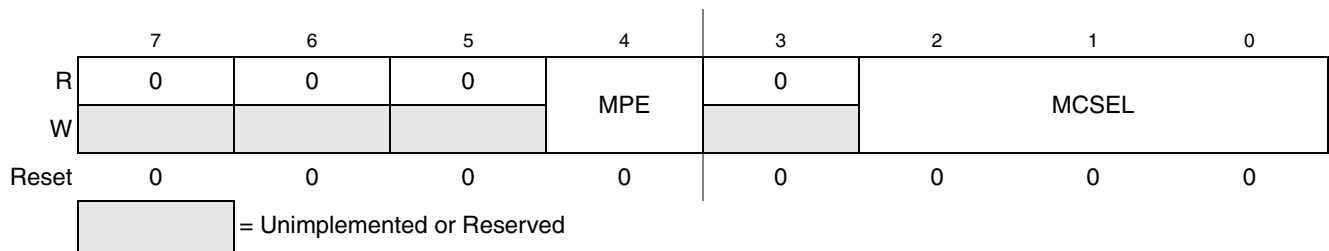


Figure 5-6. System MCLK Control Register (SMCLK)

Table 5-7. SMCLK Register Field Descriptions

Field	Description
4 MPE	<b>MCLK Pin Enable</b> — This bit is used to enable the MCLK function. 0 MCLK output disabled. 1 MCLK output enabled on PTC2 pin.
2:0 MCSEL	<b>MCLK Divide Select</b> — These bits are used to select the divide ratio for the MCLK output according to the formula below when the MCSEL bits are not equal to all zeroes. In the case that the MCSEL bits are all zero and MPE is set, the pin is driven low. See <a href="#">Equation 5-1</a> .

$$\text{MCLK frequency} = \text{Bus Clock frequency} \div (2 \times \text{MCSEL})$$

Eqn. 5-1

## 5.9.6 System Device Identification Register (SDIDH, SDIDL)

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

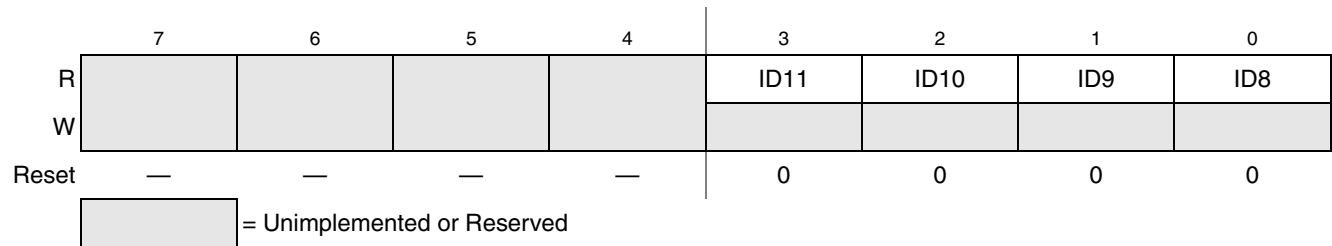


Figure 5-7. System Device Identification Register — High (SDIDH)

Table 5-8. SDIDH Register Field Descriptions

Field	Description
7:4 Reserved	<b>Bits 7:4 are reserved. Reading these bits will result in an indeterminate value; writes have no effect.</b>
3:0 ID[11:8]	<b>Part Identification Number</b> — Each derivative in the HCS08 Family has a unique identification number. The SC9S08MZ16 is hard coded to the value 0x012. See also ID bits in <a href="#">Table 5-9</a> .

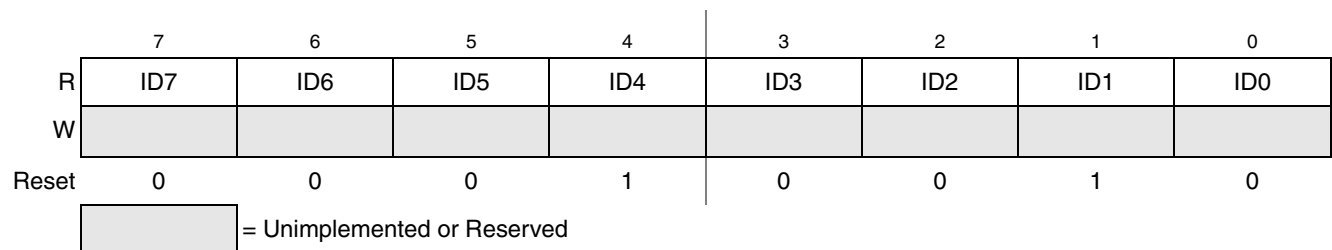


Figure 5-8. System Device Identification Register — Low (SDIDL)

Table 5-9. SDIDL Register Field Descriptions

Field	Description
7:0 ID[7:0]	<b>Part Identification Number</b> — Each derivative in the HCS08 Family has a unique identification number. The SC9S08MZ16 is hard coded to the value 0x012. See also ID bits in <a href="#">Table 5-8</a> .

## 5.9.7 System Real-Time Interrupt Status and Control Register (SRTISC)

This register contains one read-only status flag, one write-only acknowledge bit, three read/write delay selects, and three unimplemented bits, which always read 0.

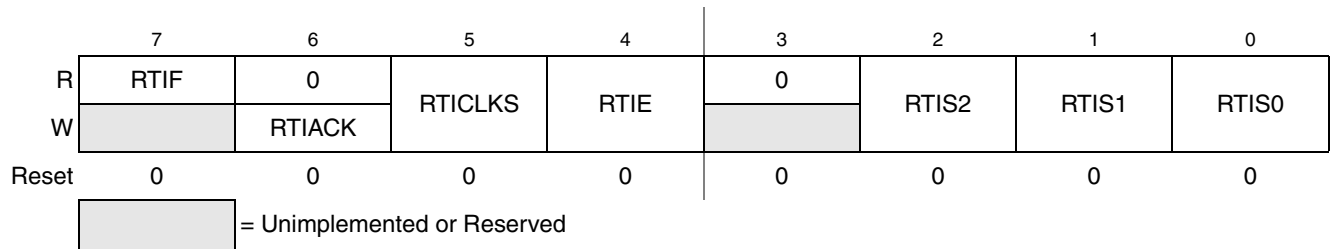


Figure 5-9. System RTI Status and Control Register (SRTISC)

Table 5-10. SRTISC Register Field Descriptions

Field	Description
7 RTIF	<b>Real-Time Interrupt Flag</b> — This read-only status bit indicates the periodic wakeup timer has timed out. 0 Periodic wakeup timer not timed out. 1 Periodic wakeup timer timed out.
6 RTIACK	<b>Real-Time Interrupt Acknowledge</b> — This write-only bit is used to acknowledge real-time interrupt request (write 1 to clear RTIF). Writing 0 has no meaning or effect. Reads always return logic 0.
5 RTICLKs	<b>Real-Time Interrupt Clock Select</b> — This read/write bit selects the clock source for the real-time interrupt. 0 Real-time interrupt request clock source is internal 1 kHz oscillator. 1 Real-time interrupt request clock source is external clock.
4 RTIE	<b>Real-Time Interrupt Enable</b> — This read-write bit enables real-time interrupts. 0 Real-time interrupts disabled. 1 Real-time interrupts enabled.
2:0 RTIS[2:0]	<b>Real-Time Interrupt Delay Selects</b> — These read/write bits select the wakeup delay for the RTI. The clock source for the real-time interrupt is a self-clocked source which oscillates at about 1 kHz, is independent of other MCU clock sources. Using external clock source the delays will be crystal frequency divided by value in RTIS2:RTIS1:RTIS0. See <a href="#">Table 5-11</a> .

Table 5-11. Real-Time Interrupt Frequency

RTIS2:RTIS1:RTIS0	1 kHz Clock Source Delay <sup>1</sup>	Using External Clock Source Delay (Crystal Frequency)
0:0:0	Disable periodic wakeup timer	Disable periodic wakeup timer
0:0:1	8 ms	divide by 256
0:1:0	32 ms	divide by 1024
0:1:1	64 ms	divide by 2048
1:0:0	128 ms	divide by 4096
1:0:1	256 ms	divide by 8192
1:1:0	512 ms	divide by 16384
1:1:1	1.024 s	divide by 32768

<sup>1</sup> Normal values are shown in this column based on  $f_{RTI} = 1$  kHz. See [Appendix A, “Electrical Characteristics and Timing Specifications,”](#)  $f_{RTI}$  for the tolerance on these values.



## 5.9.8 System Power Management Status and Control 1 Register (SPMSC1)

	7	6	5	4	3	2	1 <sup>1</sup>	0
R	LVDF	0	LVDIE	LVDRE <sup>(2)</sup>	LVDSE <sup>(2)</sup>	LVDE <sup>(2)</sup>		BGBE
W		LVDACK						
Reset	0	0	0	1	1	1	0	0

= Unimplemented or Reserved

<sup>1</sup> Bit 1 is a reserved bit that must always be written to 0.

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

**Figure 5-10. System Power Management Status and Control 1 Register (SPMSC1)**

**Table 5-12. SPMSC1 Register Field Descriptions**

Field	Description
7 LVDF	<b>Low-Voltage Detect Flag</b> — Provided LVDE = 1, this read-only status bit indicates a low-voltage detect event.
6 LVDACK	<b>Low-Voltage Detect Acknowledge</b> — This write-only bit is used to acknowledge low voltage detection errors (write 1 to clear LVDF). Reads always return 0.
5 LVDIE	<b>Low-Voltage Detect Interrupt Enable</b> — This read/write bit enables hardware interrupt requests for LVDF. 0 Hardware interrupt disabled (use polling). 1 Request a hardware interrupt when LVDF = 1.
4 LVDRE	<b>Low-Voltage Detect Reset Enable</b> — This read/write bit enables LVDF events to generate a hardware reset (provided LVDE = 1). 0 LVDF does not generate hardware resets. 1 Force an MCU reset when LVDF = 1.
3 LVDSE	<b>Low-Voltage Detect Stop Enable</b> — 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	<b>Low-Voltage Detect Enable</b> — This read/write bit enables low-voltage detect logic and qualifies the operation of other bits in this register. 0 LVD logic disabled. 1 LVD logic enabled.
0 BGBE	<b>Bandgap Buffer Enable</b> — The BGBE bit is used to enable an internal buffer for the bandgap voltage reference for use by the ADC module on one of its internal channels. 0 Bandgap buffer disabled. 1 Bandgap buffer enabled.

## 5.9.9 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	LVWF	0	LVDV	LVWV	PPDF	0		PPDC <sup>1</sup>
W		LVWACK						
Power-on reset:	0 <sup>(2)</sup>	0	0	0	0	0	0	0
LVD reset:	0 <sup>(2)</sup>	0	U	U	0	0	0	0
Any other reset:	0 <sup>(2)</sup>	0	U	U	0	0	0	0

= Unimplemented or Reserved                      U = Unaffected by reset

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

<sup>2</sup> LVWF will be set in the case when  $V_{Supply}$  transitions below the trip point or after reset and  $V_{Supply}$  is already below  $V_{LVW}$ .

**Figure 5-11. System Power Management Status and Control 2 Register (SPMSC2)**


**Table 5-13. SPMSC2 Register Field Descriptions**

Field	Description
7 LVWF	<b>Low-Voltage Warning Flag</b> — The LVWF bit indicates the low voltage warning status. 0 Low voltage warning <b>not</b> present. 1 Low voltage warning is present or was present.
6 LVWACK	<b>Low-Voltage Warning Acknowledge</b> — The LVWACK bit is the low-voltage warning acknowledge. Writing a 1 to LVWACK clears LVWF to a 0 if a low voltage warning is not present.
5 LVDV	<b>Low-Voltage Detect Voltage Select</b> — The LVDV bit selects the LVD trip point voltage ( $V_{LVD}$ ). 0 Low trip point selected ( $V_{LVD} = V_{LVDDL}$ ). 1 High trip point selected ( $V_{LVD} = V_{LVVDH}$ ).
4 LVWV	<b>Low-Voltage Warning Voltage Select</b> — The LVWV bit selects the LVW trip point voltage ( $V_{LVW}$ ). 0 Low trip point selected ( $V_{LVW} = V_{LVWDL}$ ). 1 High trip point selected ( $V_{LVW} = V_{LVWDH}$ ).
3 PPDF	<b>Partial Power Down Flag</b> — The PPDF bit indicates that the MCU has exited the stop2 mode. 0 Not stop2 mode recovery. 1 Stop2 mode recovery.
2 PPDACK	<b>Partial Power Down Acknowledge</b> — Writing a 1 to PPDACK clears the PPDF bit.
0 PPDC	<b>Partial Power Down Control</b> — The write-once PPDC bit controls whether stop2 or stop3 mode is selected. 0 Stop3 mode enabled. 1 Stop2, partial power down, mode enabled.

## 5.9.10 System Options Register 2 (SOPT2)

This high page register contains bits to configure MCU specific features on the SC9S08MZ16 device.

	7	6	5	4	3	2	1	0
R	COPCLKS <sup>1</sup>	0	0	0	0	0	0	0
W								
Reset:	1	0	0	0	0	0	0	0

 = Unimplemented or Reserved

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

**Figure 5-12. System Options Register 2 (SOPT2)**

**Table 5-14. SOPT2 Register Field Descriptions**

Field	Description
7 COPCLKS	<b>COP Watchdog Clock Select</b> — 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.



# Chapter 6

## Parallel Input/Output

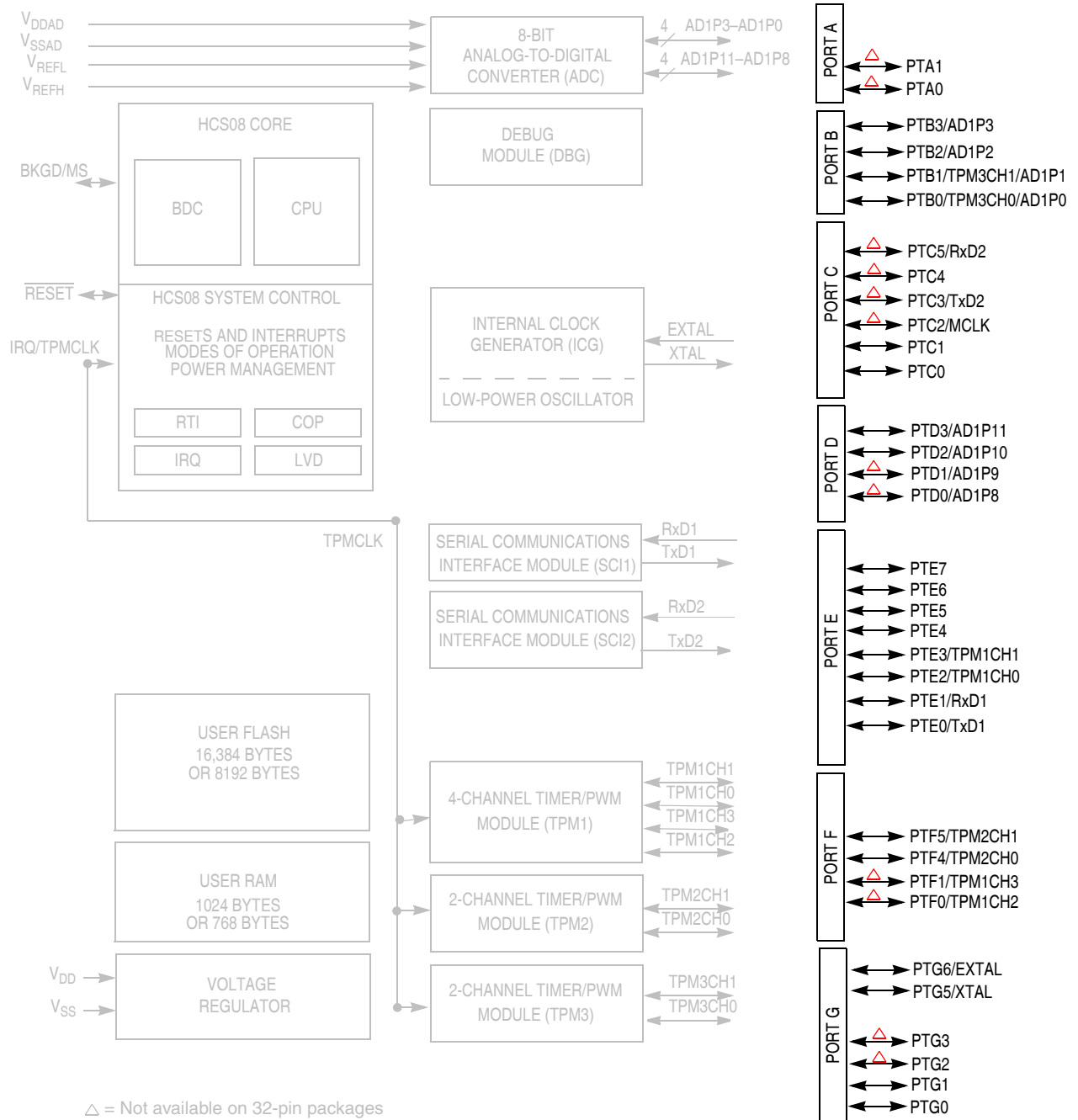
### 6.1 Introduction

This chapter explains software controls related to parallel input/output (I/O). The SC9S08MZ16 has seven I/O ports which include a total of 34 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 these pins are shared with on-chip peripherals such as timer systems and communication systems. When these other modules are not controlling the port pins, they revert to general-purpose I/O control.

#### NOTE

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



**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 should not be driven above  $V_{DD}$ .
4. Pin contains integrated pullup device.
5. PTD3 and PTD2 contain both pullup and pulldown devices.

**Figure 6-1. Block Diagram Highlighting Parallel Input/Output Pins**

## 6.2 Features

Parallel I/O and pin control features, depending on package choice, include:

- A total of 34 general-purpose I/O pins in seven ports
- Hysteresis input buffers
- Software-controlled pullups on each input pin
- Software-controlled slew rate output buffers
- Two port A pins
- Four port B pins shared with ADC and TPM3
- Six port C pins shared with MCLK and SCI2
- Four port D pins shared with ADC
- Eight port E pins shared with SCI1 and TPM1
- Four port F pins shared with TPM1 and TPM2
- Six port G pins shared with EXTAL and XTAL

## 6.3 Pin Descriptions

The SC9S08MZ16 has a total of 34 parallel I/O pins in seven ports (PTA–PTG). Not all pins are bonded out in all packages. Consult the pin assignment in [Chapter 2, “Pins and Connections,”](#) for available parallel I/O pins. All of these pins are available for general-purpose I/O when they are not used by other on-chip peripheral systems.

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

The following paragraphs discuss each port and the software controls that determine each pin’s use.

### 6.3.1 Port A

Port A	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	R	R	R	R	R	R	PTA1	PTA0

**Figure 6-2. Port A Pin Names**

Port A pins are general-purpose I/O pins. Parallel I/O function is controlled by the port A data (PTAD) and data direction (PTADD) registers which are located in page zero register space. The pin control registers, pullup enable (PTAPE), slew rate control (PTASE), and drive strength select (PTADS) are located in the high page registers. Refer to [Section 6.4, “Parallel I/O Control”](#) for more information about general-purpose I/O control and [Section 6.5, “Pin Control”](#) for more information about pin control.

### 6.3.2 Port B

Port B	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	R	R	R	R	PTB3/ AD1P3	PTB2/ AD1P2	PTB1/ TPM3CH1/ AD1P1	PTB0/ TPM3CH0/ AD1P0

**Figure 6-3. Port B Pin Names**

Port B pins are general-purpose I/O pins. Parallel I/O function is controlled by the port B data (PTBD) and data direction (PTBDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTBPE), slew rate control (PTBSE), and drive strength select (PTBDS) are located in the high page registers. Refer to [Section 6.4, “Parallel I/O Control”](#) for more information about general-purpose I/O control and [Section 6.5, “Pin Control”](#) for more information about pin control.

Port B general-purpose I/O are shared with the ADC and TPM3 timer channels. Any pin enabled as an ADC input will have the general-purpose I/O function disabled. When any TPM3 function is enabled, the direction (input or output) is controlled by the TPM3 and not by the data direction register of the parallel I/O port. Refer to [Chapter 9, “Timer/PWM \(S08TPMV3\)”](#) for more information about using port B pins as TPM channels. Refer to [Chapter 11, “Analog-to-Digital Converter \(S08ADC10V1\)”](#) for more information about using port B as analog inputs.

### 6.3.3 Port C

Port C	Bit 7	6	5	3	3	2	1	Bit 0
MCU Pin:	0	R	PTC5/ RxD2	PTC4	PTC3/ TxD2	PTC2/ MCLK	PTC1	PTC0

**Figure 6-4. Port C Pin Names**

Port C pins are general-purpose I/O pins. Parallel I/O function is controlled by the port C data (PTCD) and data direction (PTCDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTCPE), slew rate control (PTCSE), and drive strength select (PTCDS) are located in the high page registers. Refer to [Section 6.4, “Parallel I/O Control”](#) for more information about general-purpose I/O control and [Section 6.5, “Pin Control”](#) for more information about pin control.

Port C general-purpose I/O is shared with SCI2 and MCLK. When any shared function is enabled, the direction, input or output, is controlled by the shared function and not by the data direction register of the parallel I/O port. Also, for pins which are configured as outputs by the shared function, the output data is controlled by the shared function and not by the port data register.

Refer to [Chapter 10, “Serial Communications Interface \(S08SCIV4\)”](#) for more information about using port C pins as SCI pins.

Refer to [Chapter 5, “Resets, Interrupts, and System Configuration”](#) for more information about using PTC2 as the MCLK pin.



### 6.3.4 Port D

Port D	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	R	R	R	R	PTD3/ AD1P11	PTD2/ AD1P10	PTD1/ AD1P9	PTD0/ AD1P8

**Figure 6-5. Port D Pin Names**

Port D pins are general-purpose I/O pins. Parallel I/O function is controlled by the port D data (PTDD) and data direction (PTDDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTDPE), slew rate control (PTDSE), and drive strength select (PTDDS) are located in the high page registers. Refer to [Section 6.4, “Parallel I/O Control”](#) for more information about general-purpose I/O control and [Section 6.5, “Pin Control”](#) for more information about pin control.

Port D general-purpose I/O are shared with the ADC. When any of these shared functions is enabled, the direction, input or output, is controlled by the shared function and not by the data direction register of the parallel I/O port. When a pin is shared with both the ADC and a digital peripheral function, the ADC has higher priority. For example, in the case that the ADC are configured to use PTD3, then the pin is controlled by the ADC module.

Refer to [Chapter 11, “Analog-to-Digital Converter \(S08ADC10V1\)”](#) for more information about using port D pins as analog inputs.

### 6.3.5 Port E

Port E	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	PTE7	PTE6	PTE5	PTE4	PTE3/ TPM1CH1	PTE2/ TPM1CH0	PTE1/ RxD1	PTE0/ TxD1

**Figure 6-6. Port E Pin Names**

Port E pins are general-purpose I/O pins. Parallel I/O function is controlled by the port E data (PTED) and data direction (PTEDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTEPE), slew rate control (PTESE), and drive strength select (PTEDS) are located in the high page registers. Refer to [Section 6.4, “Parallel I/O Control”](#) for more information about general-purpose I/O control and [Section 6.5, “Pin Control”](#) for more information about pin control.

Port E general-purpose I/O is shared with TPM1 timer channels and SCI. When any of these shared functions is enabled, the direction, input or output, is controlled by the shared function and not by the data direction register of the parallel I/O port. Also, for pins which are configured as outputs by the shared function, the output data is controlled by the shared function and not by the port data register.

Refer to [Chapter 10, “Serial Communications Interface \(S08SCIV4\)”](#) for more information about using port E pins as SCI pins.

Refer to [Chapter 9, “Timer/PWM \(S08TPMV3\)”](#) for more information about using port E pins as TPM channel pins.

### 6.3.6 Port F

Port F	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	R	R	PTF5/ TPM2CH1	PTF4/ TPM2CH0	R	R	PTF1/ TPM1CH3	PTF0/ TPM1CH2

**Figure 6-7. Port F Pin Names**

Port F pins are general-purpose I/O pins. Parallel I/O function is controlled by the port F data (PTFD) and data direction (PTFDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTFPE), slew rate control (PTFSE), and drive strength select (PTFDS) are located in the high page registers. Refer to [Section 6.4, “Parallel I/O Control”](#) for more information about general-purpose I/O control and [Section 6.5, “Pin Control”](#) for more information about pin control.

Port F general-purpose I/O is shared with TPM1 and TPM2 timer channels. When any of these shared functions is enabled, the direction, input or output, is controlled by the shared function and not by the data direction register of the parallel I/O port. Also, for pins which are configured as outputs by the shared function, the output data is controlled by the shared function and not by the port data register.

Refer to [Chapter 9, “Timer/PWM \(S08TPMV3\)”](#) for more information about using port F pins as TPM channel pins.

### 6.3.7 Port G

Port G	Bit 7	6	5	4	3	2	1	Bit 0
MCU Pin:	R	PTG6/ EXTAL	PTG5/ XTAL	R	PTG3	PTG2	PTG1	PTG0

**Figure 6-8. Port G Pin Names**

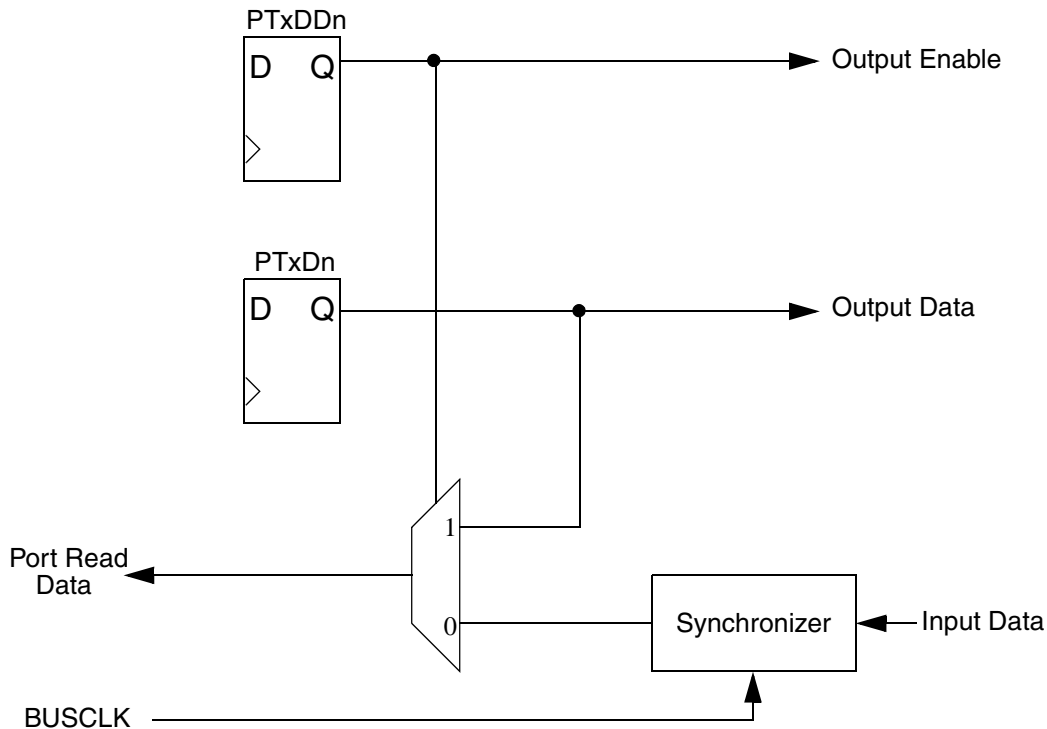
Port G pins are general-purpose I/O pins. Parallel I/O function is controlled by the port G data (PTGD) and data direction (PTGDD) registers which are located in page zero register space. The pin control registers, pullup enable (PTGPE), slew rate control (PTGSE), and drive strength select (PTGDS) are located in the high page registers. Refer to [Section 6.4, “Parallel I/O Control”](#) for more information about general-purpose I/O control and [Section 6.5, “Pin Control”](#) for more information about pin control.

Port G general-purpose I/O is shared with XTAL and EXTAL. When the external oscillator is enabled, PTG5 and PTG6 function as oscillator pins. In this case the associated parallel I/O and pin control registers have no control of the pins.

Refer to [Chapter 8, “Internal Clock Generator \(S08ICGV4\)”](#) for more information about using port G pins as XTAL and EXTAL pins.

## 6.4 Parallel I/O Control

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 6-9. Parallel I/O Block Diagram**

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

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

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

## 6.5 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.

### 6.5.1 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.

### 6.5.2 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.

### 6.5.3 Output Drive Strength Select

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

## 6.6 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 should 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 should examine the state of the PPDF bit in the SPMSC2 register. If the PPDF bit is 0, I/O must be initialized as if a power on reset had occurred. If the PPDF bit is 1, I/O data previously stored in RAM, before the STOP instruction was executed, peripherals may require being initialized and restored to their pre-stop condition. The user must then write a 1 to the PPDACK bit in the SPMSC2 register. Access to I/O is now permitted again in the user's application program.
- In stop3 mode, all I/O is maintained because internal logic circuitry stays powered up. Upon recovery, normal I/O function is available to the user.

## 6.7 Parallel I/O and Pin Control Registers

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

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

### 6.7.1 Port A I/O Registers (PTAD and PTADD)

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

	7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	PTAD1	PTAD0
W								
Reset	0	0	0	0	0	0	0	0

**Note:** Bits 7 through 2 are reserved bits that must always be written to 0.

**Figure 6-10. Port A Data Register (PTAD)**

**Table 6-1. PTAD Register Field Descriptions**

Field	Description
1:0 PTADn	<b>Port A Data Register Bits</b> — 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.

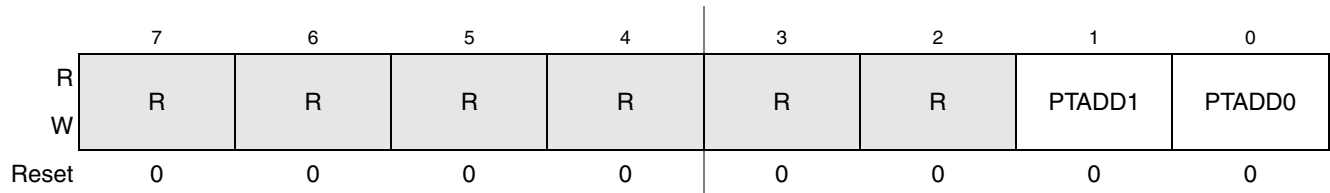


Figure 6-11. Data Direction for Port A Register (PTADD)<sup>1</sup>

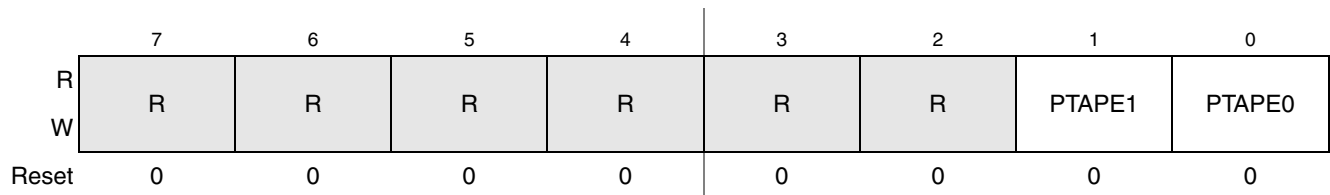
<sup>1</sup> Bits 7 through 2 are reserved bits that must always be written to 0.

Table 6-2. PTADD Register Field Descriptions

Field	Description
1:0 PTADDn	<b>Data Direction for Port A Bits</b> — 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.

### 6.7.2 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.

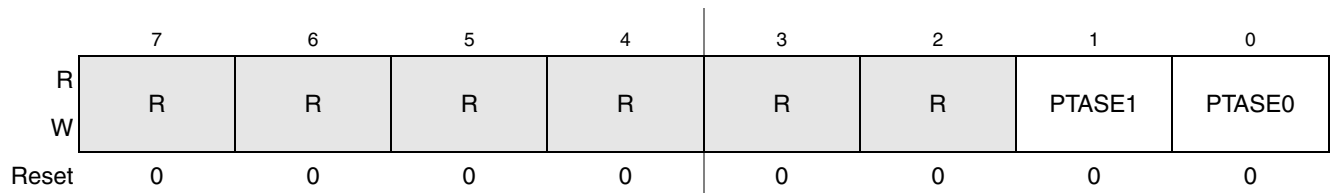


**Note:** Bits 7 through 2 are reserved bits that must always be written to 0.

Figure 6-12. Internal Pullup Enable for Port A (PTAPE)

Table 6-3. PTADD Register Field Descriptions

Field	Description
1:0 PTAPEn	<b>Internal Pullup Enable for Port A Bits</b> — 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.



**Note:** Bits 7 through 2 are reserved bits that must always be written to 0.

Figure 6-13. Output Slew Rate Control Enable for Port A (PTASE)

Table 6-4. PTASE Register Field Descriptions

Field	Description
1:0 PTASEn	<p><b>Output Slew Rate Control Enable for Port A Bits</b> — 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.</p> <p>0 Output slew rate control disabled for port A bit n. 1 Output slew rate control enabled for port A bit n.</p>

	7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	PTADS1	PTADS0
W								
Reset	0	0	0	0	0	0	0	0

**Note:** Bits 7 through 2 are reserved bits that must always be written to 0.

Figure 6-14. Output Drive Strength Selection for Port A (PTADS)

Table 6-5. PTADS Register Field Descriptions

Field	Description
1:0 PTADS <sub>n</sub>	<p><b>Output Drive Strength Selection for Port A Bits</b> — Each of these control bits selects between low and high output drive for the associated PTA pin.</p> <p>0 Low output drive enabled for port A bit n. 1 High output drive enabled for port A bit n.</p>

### 6.7.3 Port B I/O Registers (PTBD and PTBDD)

Port B parallel I/O function is controlled by the registers in this section.

	7	6	5	4	3	2	1	0
R	R	R	R	R	PTBD3	PTBD2	PTBD1	PTBD0
W								
Reset	0	0	0	0	0	0	0	0

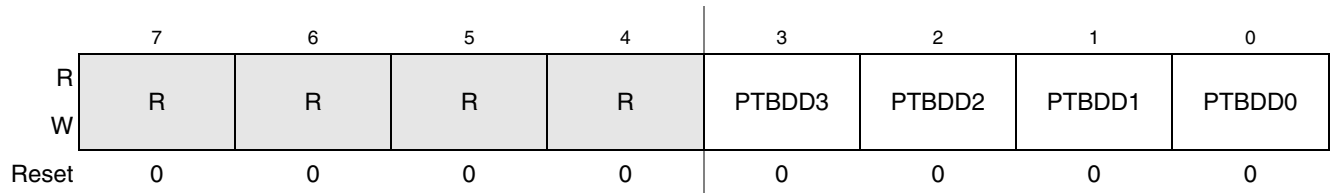
**Note:** Bits 7 through 4 are reserved bits that must always be written to 0.

Figure 6-15. Port B Data Register (PTBD)

Table 6-6. PTBD Register Field Descriptions

Field	Description
3:0 PTBD[3:0]	<p><b>Port B Data Register Bits</b> — 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.</p> <p>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.</p>

1.



**Note:** Bits 7 through 4 are reserved bits that must always be written to 0.

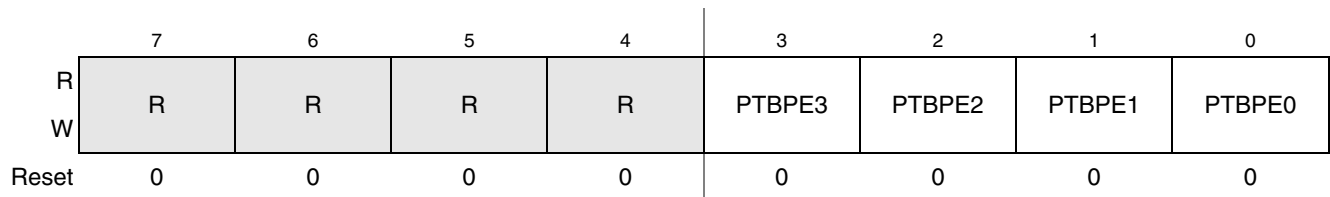
**Figure 6-16. Data Direction for Port B (PTBDD)**

**Table 6-7. PTBDD Register Field Descriptions**

Field	Description
3:0 PTBDD[3:0]	<p><b>Data Direction for Port B Bits</b> — These read/write bits control the direction of port B pins and what is read for PTBD reads.</p> <p>0 Input (output driver disabled) and reads return the pin value.</p> <p>1 Output driver enabled for port B bit n and PTBD reads return the contents of PTBDn.</p>

### 6.7.4 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.

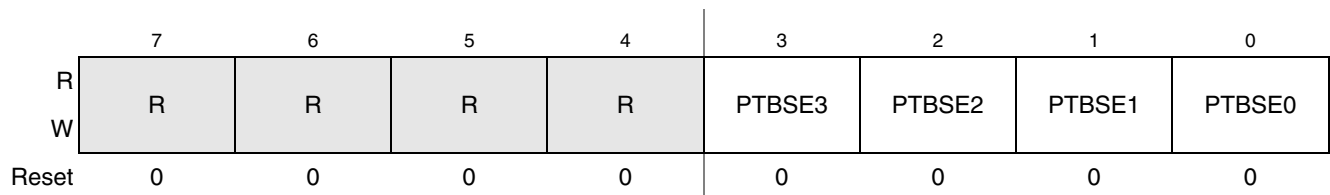


**Note:** Bits 7 through 4 are reserved bits that must always be written to 0.

**Figure 6-17. Internal Pullup Enable for Port B (PTBPE)**

**Table 6-8. PTBPE Register Field Descriptions**

Field	Description
3:0 PTBPE[3:0]	<p><b>Internal Pullup Enable for Port B Bits</b> — 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.</p> <p>0 Internal pullup device disabled for port B bit n.</p> <p>1 Internal pullup device enabled for port B bit n.</p>



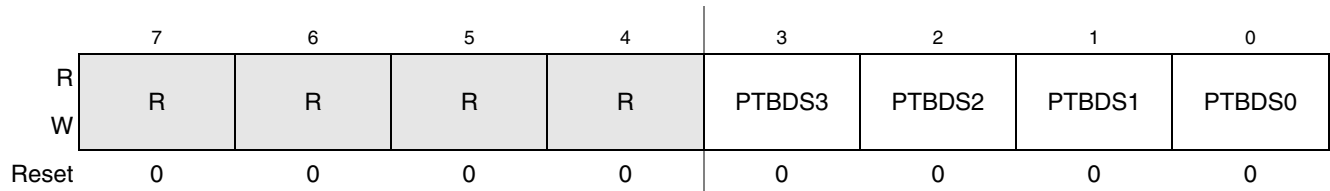
**Note:** Bits 7 through 4 are reserved bits that must always be written to 0.

**Figure 6-18. Output Slew Rate Control Enable for Port B (PTBSE)**



Table 6-9. PTBSE Register Field Descriptions

Field	Description
3:0 PTBSE[3:0]	<p><b>Output Slew Rate Control Enable for Port B Bits</b>— 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.</p> <p>0 Output slew rate control disabled for port B bit n. 1 Output slew rate control enabled for port B bit n.</p>



**Note:** Bits 7 through 4 are reserved bits that must always be written to 0.

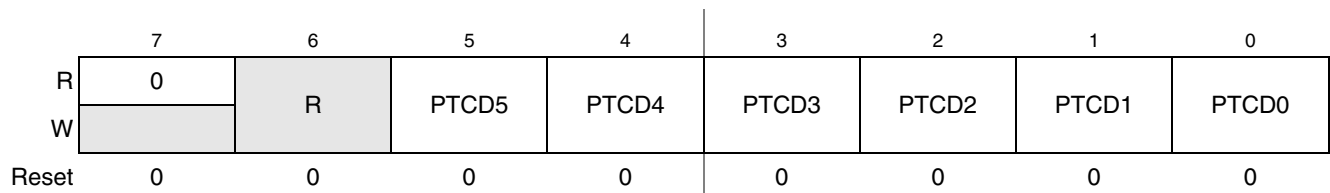
Figure 6-19. Output Drive Strength Selection for Port B (PTBDS)

Table 6-10. PTBDS Register Field Descriptions

Field	Description
3:0 PTBDS[3:0]	<p><b>Output Drive Strength Selection for Port B Bits</b> — 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. 1 High output drive enabled for port B bit n.</p>

## 6.7.5 Port C I/O Registers (PTCD and PTCDD)

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

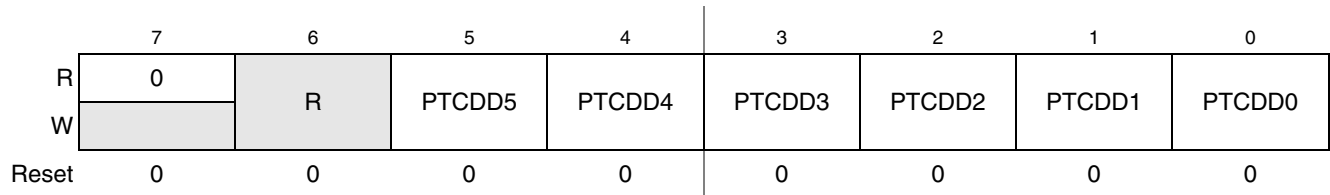


**Note:** Bit 6 is a reserved bit that must always be written to 0.

Figure 6-20. Port C Data Register (PTCD)

Table 6-11. PTCD Register Field Descriptions

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



**Note:** Bit 6 is a reserved bit that must always be written to 0.

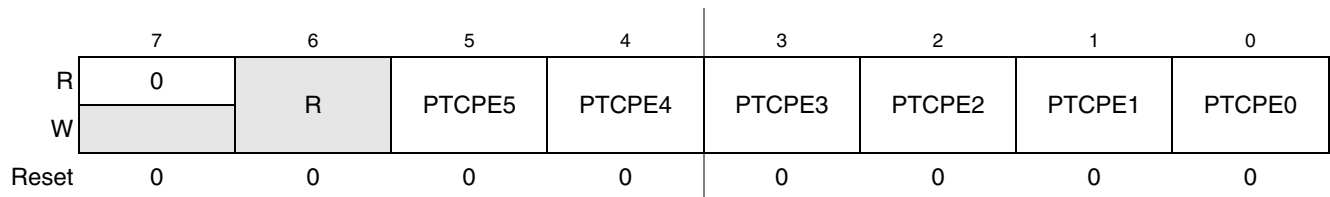
**Figure 6-21. Data Direction for Port C (PTCDD)**

**Table 6-12. PTCDD Register Field Descriptions**

Field	Description
5:0 PTCDD[5:0]	<p><b>Data Direction for Port C Bits</b> — These read/write bits control the direction of port C pins and what is read for PTCDD reads.</p> <p>0 Input (output driver disabled) and reads return the pin value.</p> <p>1 Output driver enabled for port C bit n and PTCDD reads return the contents of PTCDDn.</p>

### 6.7.6 Port C Pin Control Registers (PTCPE, PTCSE, PTCDS)

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

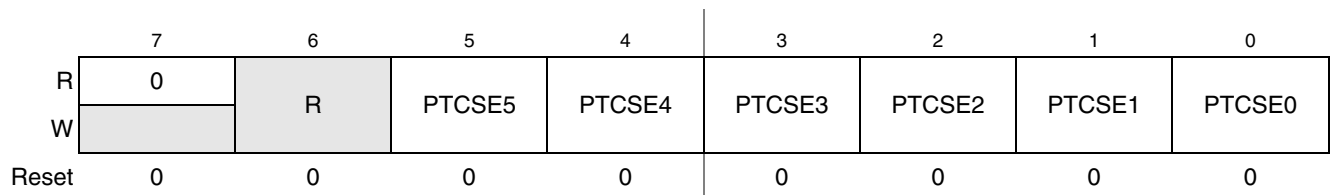


**Note:** Bit 6 is a reserved bit that must always be written to 0.

**Figure 6-22. Internal Pullup Enable for Port C (PTCPE)**

**Table 6-13. PTCPE Register Field Descriptions**

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

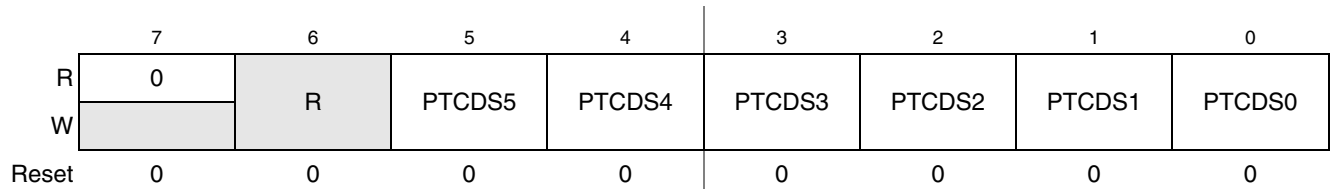


**Note:** Bit 6 is a reserved bit that must always be written to 0.

**Figure 6-23. Output Slew Rate Control Enable for Port C (PTCSE)**

Table 6-14. PTCSE Register Field Descriptions

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



**Note:** Bit 6 is a reserved bit that must always be written to 0.

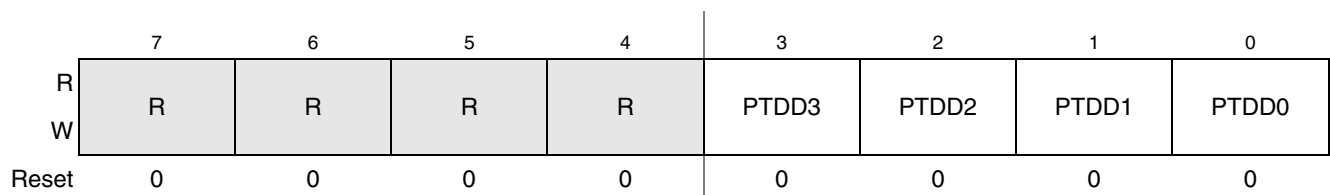
Figure 6-24. Output Drive Strength Selection for Port C (PTCDS)

Table 6-15. PTCDS Register Field Descriptions

Field	Description
5:0 PTCDS[5:0]	<p><b>Output Drive Strength Selection for Port C Bits</b> — Each of these control bits selects between low and high output drive for the associated PTC pin.</p> <p>0 Low output drive enabled for port C bit n. 1 High output drive enabled for port C bit n.</p>

## 6.7.7 Port D I/O Registers (PTDD and PTDDD)

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

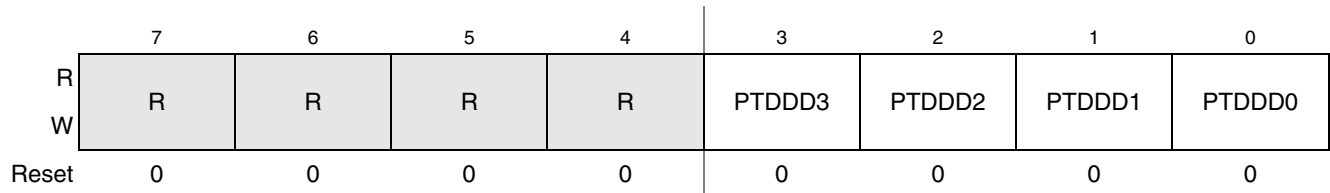


**Note:** Bits 7 through 4 are reserved bits that must always be written to 0.

Figure 6-25. Port D Data Register (PTDD)

Table 6-16. PTDD Register Field Descriptions

Field	Description
3:0 PTDD[3:0]	<p><b>Port D Data Register Bits</b> — For port D pins that are inputs, reads return the logic level on the pin. For port D pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port D pins that are configured as outputs, the logic level is driven out the corresponding MCU pin.</p> <p>Reset forces PTDD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.</p>



**Note:** Bits 7 through 4 are reserved bits that must always be written to 0.

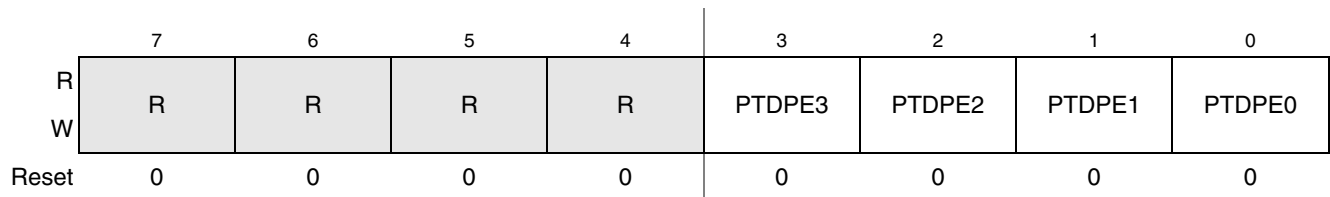
**Figure 6-26. Data Direction for Port D (PTDDD)**

**Table 6-17. PTDDD Register Field Descriptions**

Field	Description
3:0 PTDDD[3:0]	<p><b>Data Direction for Port D Bits</b> — These read/write bits control the direction of port D pins and what is read for PTDD reads.</p> <p>0 Input (output driver disabled) and reads return the pin value.</p> <p>1 Output driver enabled for port D bit n and PTDD reads return the contents of PTDDn.</p>

### 6.7.8 Port D Pin Control Registers (PTDPE, PTDSE, PTDDS)

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

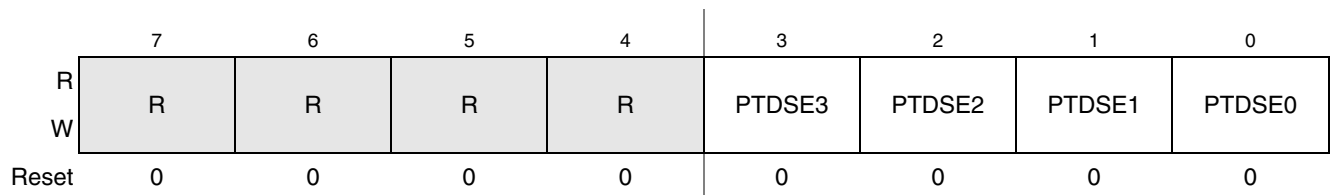


**Note:** Bits 7 through 4 are reserved bits that must always be written to 0.

**Figure 6-27. Internal Pullup Enable for Port D (PTDPE)**

**Table 6-18. PTDPE Register Field Descriptions**

Field	Description
3:0 PTDPE[3:0]	<p><b>Internal Pullup Enable for Port D Bits</b> — Each of these control bits determines if the internal pullup device is enabled for the associated PTD pin. For port D pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled.</p> <p>0 Internal pullup device disabled for port D bit n.</p> <p>1 Internal pullup device enabled for port D bit n.</p>

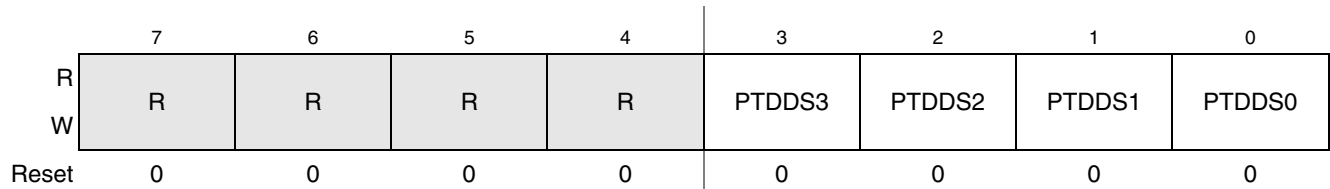


**Note:** Bits 7 through 4 are reserved bits that must always be written to 0.

**Figure 6-28. Output Slew Rate Control Enable for Port D (PTDSE)**

Table 6-19. PTDSE Register Field Descriptions

Field	Description
3:0 PTDSE[3:0]	<p><b>Output Slew Rate Control Enable for Port D Bits</b> — Each of these control bits determine whether output slew rate control is enabled for the associated PTD pin. For port D pins that are configured as inputs, these bits have no effect.</p> <p>0 Output slew rate control disabled for port D bit n. 1 Output slew rate control enabled for port D bit n.</p>



**Note:** Bits 7 through 4 are reserved bits that must always be written to 0.

Figure 6-29. Output Drive Strength Selection for Port D (PTDDS)

Table 6-20. PTDDS Register Field Descriptions

Field	Description
3:0 PTDDS[3:0]	<p><b>Output Drive Strength Selection for Port D Bits</b> — Each of these control bits selects between low and high output drive for the associated PTD pin.</p> <p>0 Low output drive enabled for port D bit n. 1 High output drive enabled for port D bit n.</p>

## 6.7.9 Port E I/O Registers (PTED and PTEDD)

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

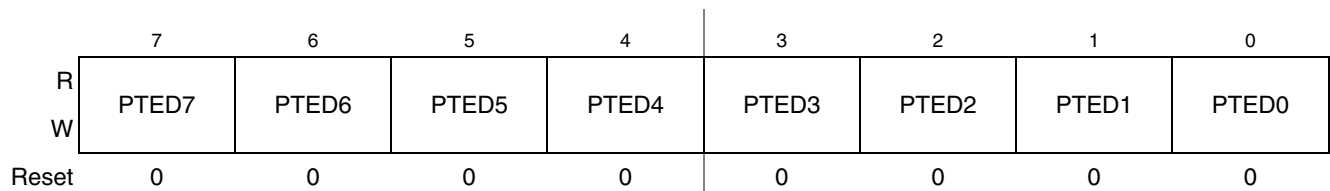


Figure 6-30. Port E Data Register (PTED)

Table 6-21. PTED Register Field Descriptions

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

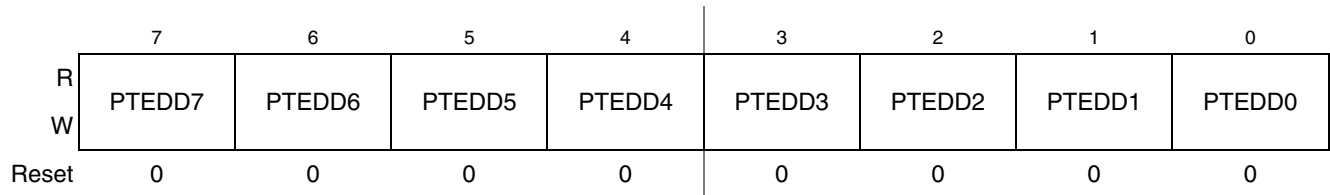


Figure 6-31. Data Direction for Port E (PTEDD)

Table 6-22. PTEDD Register Field Descriptions

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

### 6.7.10 Port E Pin Control Registers (PTEPE, PTESE, PTEDS)

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

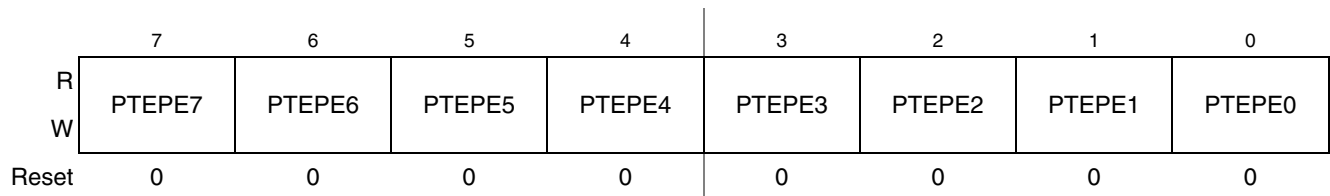


Figure 6-32. Internal Pullup Enable for Port E (PTEPE)

Table 6-23. PTEPE Register Field Descriptions

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

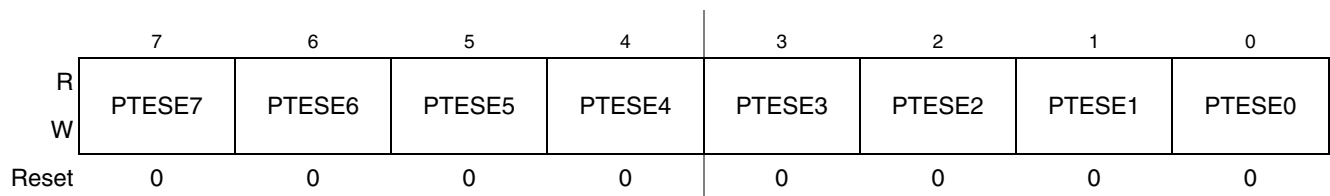


Figure 6-33. Output Slew Rate Control Enable for Port E (PTESE)

Table 6-24. PTESE Register Field Descriptions

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

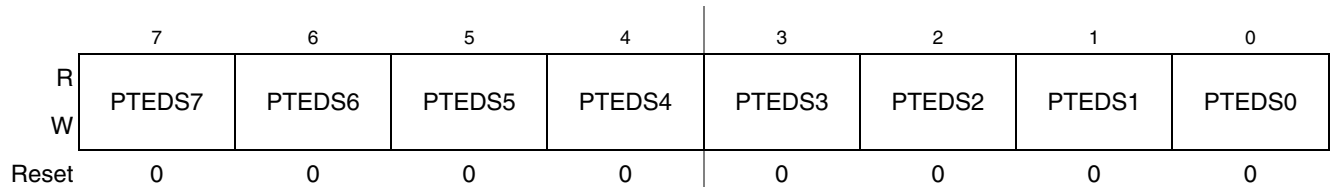


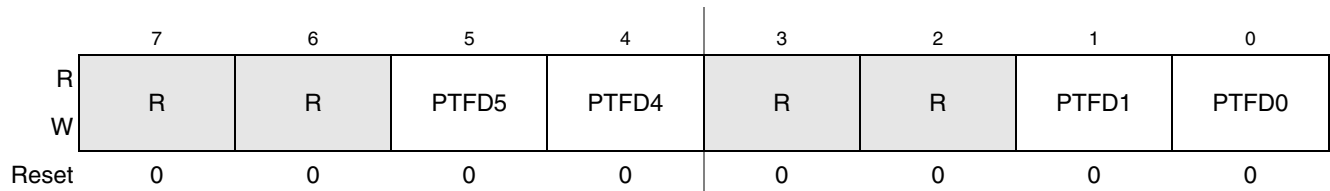
Figure 6-34. Output Drive Strength Selection for Port E (PTEDS)

Table 6-25. PTEDS Register Field Descriptions

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

### 6.7.11 Port F I/O Registers (PTFD and PTFDD)

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

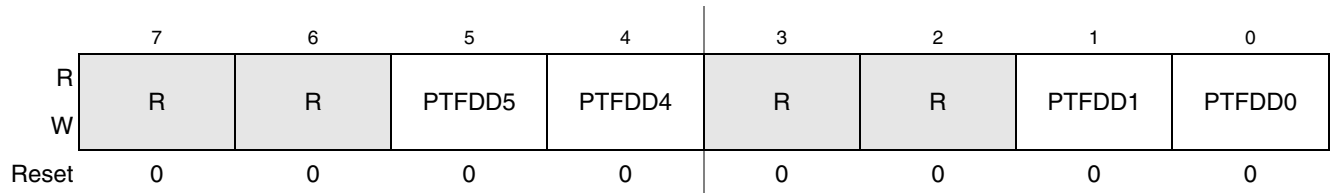


**Note:** Bits 7, 6, 3 and 2 are reserved bits that must always be written to 0.

Figure 6-35. Port F Data Register (PTFD)

Table 6-26. PTFD Register Field Descriptions

Field	Description
[5:4, 1:0] PTFDn	<p><b>Port F Data Register Bits</b>— For port F pins that are inputs, reads return the logic level on the pin. For port F pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port F pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTFD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.</p>



**Note:** Bits 7, 6, 3 and 2 are reserved bits that must always be written to 0.

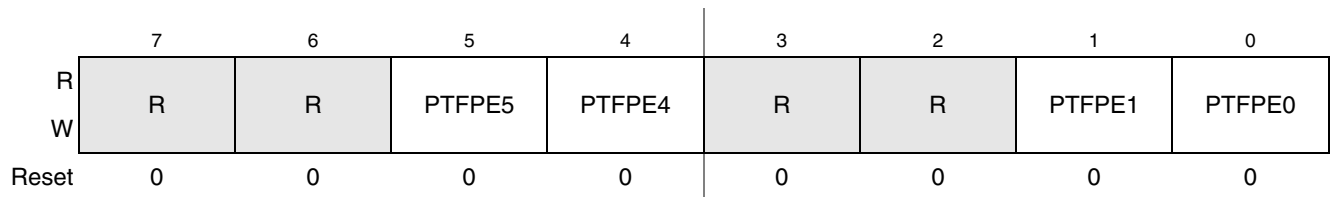
**Figure 6-36. Data Direction for Port F (PTFDD)**

**Table 6-27. PTFDD Register Field Descriptions**

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

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

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

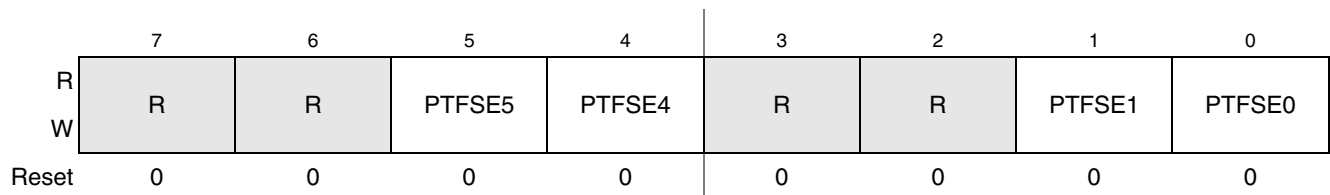


**Note:** Bits 7, 6, 3 and 2 are reserved bits that must always be written to 0.

**Figure 6-37. Internal Pullup Enable for Port F (PTFPE)**

**Table 6-28. PTFPE Register Field Descriptions**

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



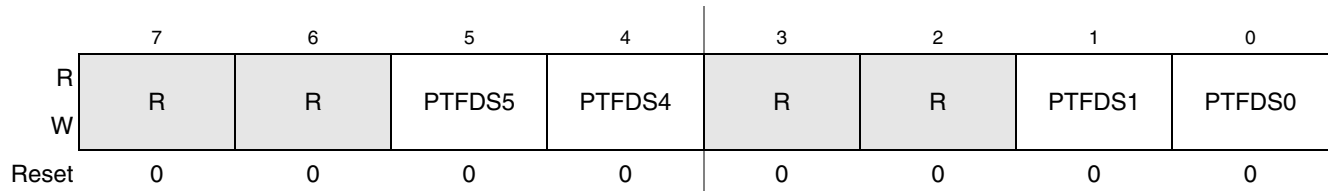
**Note:** Bits 7, 6, 3 and 2 are reserved bits that must always be written to 0.

**Figure 6-38. Output Slew Rate Control Enable for Port F (PTFSE)**



Table 6-29. PTFSE Register Field Descriptions

Field	Description
[5:4, 1:0] PTFSEn	<b>Output Slew Rate Control Enable for Port F Bits</b> — Each of these control bits determine whether output slew rate control is enabled for the associated PTF pin. For port F pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port F bit n. 1 Output slew rate control enabled for port F bit n.



**Note:** Bits 7, 6, 3 and 2 are reserved bits that must always be written to 0.

Figure 6-39. Output Drive Strength Selection for Port F (PTFDS)

Table 6-30. PTFDS Register Field Descriptions

Field	Description
[5:4, 1:0] PTFDSn	<b>Output Drive Strength Selection for Port F Bits</b> — Each of these control bits selects between low and high output drive for the associated PTF pin. 0 Low output drive enabled for port F bit n. 1 High output drive enabled for port F bit n.

### 6.7.13 Port G I/O Registers (PTGD and PTGDD)

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

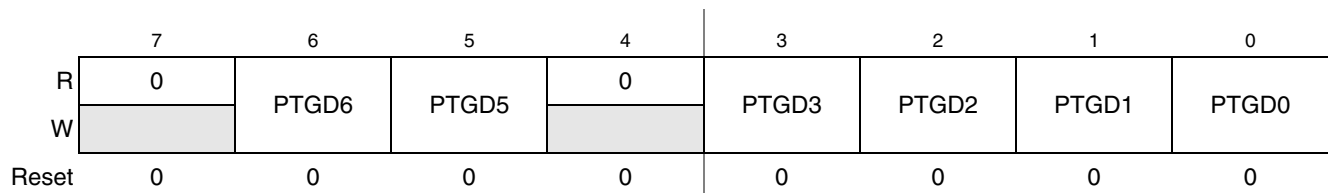


Figure 6-40. Port G Data Register (PTGD)

Table 6-31. PTGD Register Field Descriptions

Field	Description
6:5, 3:0 PTGD[6:5, 3:0]	<b>Port G Data Register Bits</b> — For port G pins that are inputs, reads return the logic level on the pin. For port G pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port G pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTGD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

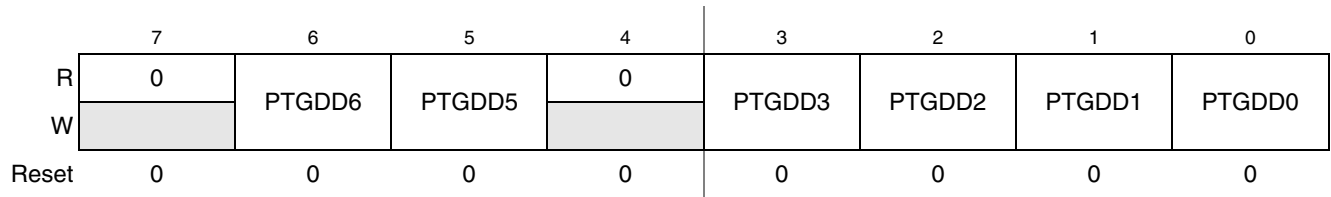


Figure 6-41. Data Direction for Port G (PTGDD)

Table 6-32. PTGDD Register Field Descriptions

Field	Description
6:5, 3:0 PTGDD[6:5, 3:0]	<b>Data Direction for Port G Bits</b> — These read/write bits control the direction of port G pins and what is read for PTGD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port G bit n and PTGD reads return the contents of PTGDn.

### 6.7.14 Port G Pin Control Registers (PTGPE, PTGSE, PTGDS)

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

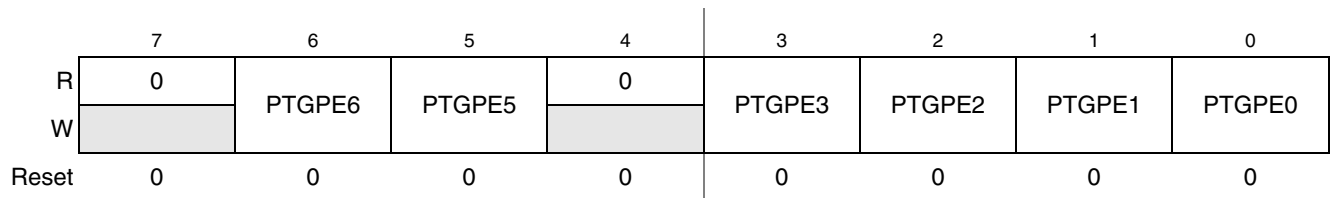


Figure 6-42. Internal Pullup Enable for Port G Bits (PTGPE)

Table 6-33. PTGPE Register Field Descriptions

Field	Description
6:5, 3:0 PTGPE[6:5, 3:0]	<b>Internal Pullup Enable for Port G Bits</b> — Each of these control bits determines if the internal pullup device is enabled for the associated PTG pin. For port G pins that are configured as outputs, these bits have no effect and the internal pullup devices are disabled. 0 Internal pullup device disabled for port G bit n. 1 Internal pullup device enabled for port G bit n.

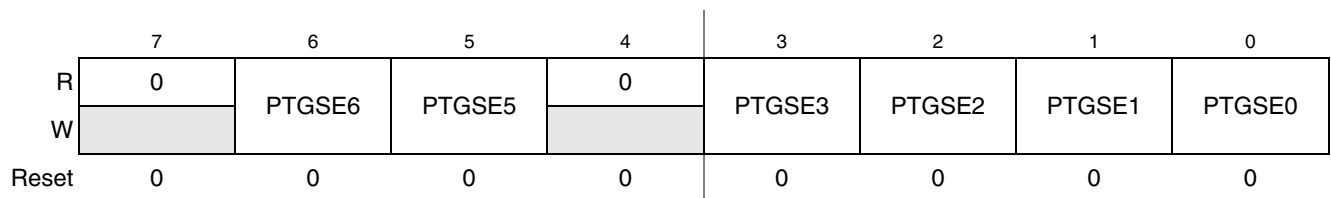


Figure 6-43. Output Slew Rate Control Enable for Port G Bits (PTGSE)

Table 6-34. PTGSE Register Field Descriptions

Field	Description
6:5, 3:0 PTGSE[6:5, 3:0]	<p><b>Output Slew Rate Control Enable for Port G Bits</b>— Each of these control bits determine whether output slew rate control is enabled for the associated PTG pin. For port G pins that are configured as inputs, these bits have no effect.</p> <p>0 Output slew rate control disabled for port G bit n. 1 Output slew rate control enabled for port G bit n.</p>

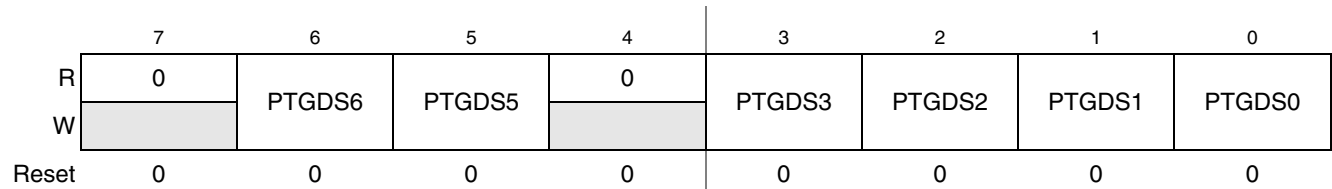


Figure 6-44. Output Drive Strength Selection for Port G (PTGDS)

Table 6-35. PTGDS Register Field Descriptions

Field	Description
6:5, 3:0 PTGDS[6:5, 3:0]	<p><b>Output Drive Strength Selection for Port G Bits</b> — Each of these control bits selects between low and high output drive for the associated PTG pin.</p> <p>0 Low output drive enabled for port G bit n. 1 High output drive enabled for port G 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*.

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.

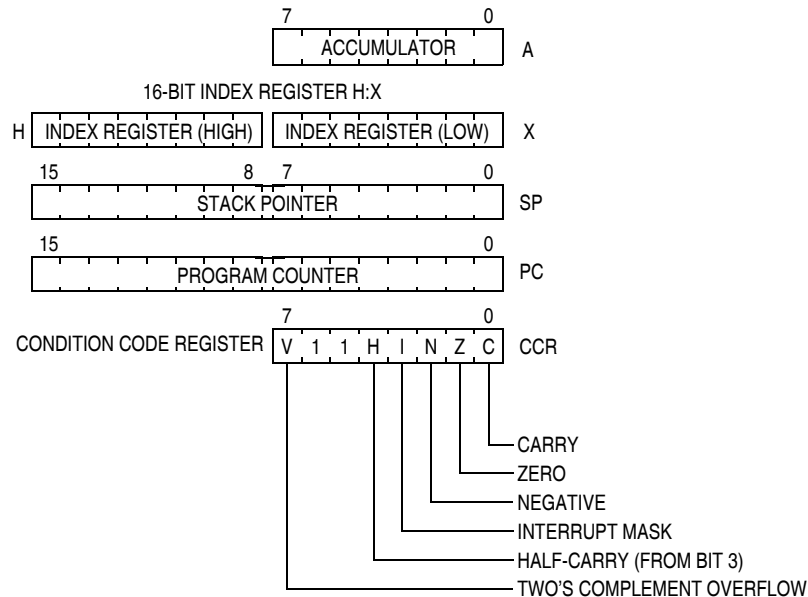


Figure 7-1. CPU Registers

### 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*.

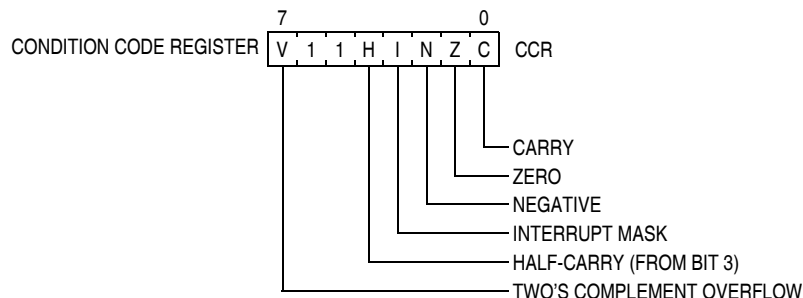


Figure 7-2. Condition Code Register

Table 7-1. CCR Register Field Descriptions

Field	Description
7 V	<b>Two's Complement Overflow Flag</b> — 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	<b>Half-Carry Flag</b> — 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	<b>Interrupt Mask Bit</b> — 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	<b>Negative Flag</b> — 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	<b>Zero Flag</b> — 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	<b>Carry/Borrow Flag</b> — 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 up the target MCU when it is time to resume processing. Unlike the earlier M68HC05 and M68HC08 MCUs, the HCS08 can be configured to keep a minimum set of clocks running in stop mode. This optionally allows an internal periodic signal to wake the target MCU from stop mode.

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

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

### 7.4.5 BGND Instruction

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

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

## 7.5 HCS08 Instruction Set Summary

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

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

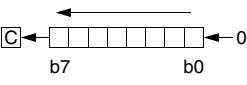
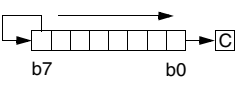
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	I N Z C
ADC #opr8i ADC opr8a ADC opr16a ADC oprx16,X ADC oprx8,X ADC ,X ADC oprx16,SP ADC oprx8,SP	Add with Carry $A \leftarrow (A) + (M) + (C)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 ii B9 dd C9 hh ll D9 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	↑ 1 1 ↓	- ↓ ↓ ↓
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	↑ 1 1 ↓	- ↓ ↓ ↓
AIS #opr8i	Add Immediate Value (Signed) to Stack Pointer $SP \leftarrow (SP) + (M)$	IMM	A7 ii	2	pp	- 1 1 -	- - - - -
AIX #opr8i	Add Immediate Value (Signed) to Index Register (H:X) $H:X \leftarrow (H:X) + (M)$	IMM	AF ii	2	pp	- 1 1 -	- - - - -
AND #opr8i AND opr8a AND opr16a AND oprx16,X AND oprx8,X AND ,X AND oprx16,SP AND oprx8,SP	Logical AND $A \leftarrow (A) \& (M)$	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 1 1 -	- ↓ ↓ -
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	↑ 1 1 -	- ↓ ↓ ↓
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	↑ 1 1 -	- ↓ ↓ ↓
BCC rel	Branch if Carry Bit Clear (if C = 0)	REL	24 rr	3	ppp	- 1 1 -	- - - - -

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					
						V	I	H	I	N	Z
BCLR <i>n,opr8a</i>	Clear Bit n in Memory (Mn ← 0)	DIR (b0)	11 dd	5	rfwpp	- 1 1 -	- - - - -				
		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	- 1 1 -	- - - - -				
BEQ <i>rel</i>	Branch if Equal (if Z = 1)	REL	27 rr	3	ppp	- 1 1 -	- - - - -				
BGE <i>rel</i>	Branch if Greater Than or Equal To (if $N \oplus V = 0$ ) (Signed)	REL	90 rr	3	ppp	- 1 1 -	- - - - -				
BGND	Enter active background if ENBDM=1 Waits for and processes BDM commands until GO, TRACE1, or TAGGO	INH	82	5+	fp...ppp	- 1 1 -	- - - - -				
BGT <i>rel</i>	Branch if Greater Than (if $Z \mid (N \oplus V) = 0$ ) (Signed)	REL	92 rr	3	ppp	- 1 1 -	- - - - -				
BHCC <i>rel</i>	Branch if Half Carry Bit Clear (if H = 0)	REL	28 rr	3	ppp	- 1 1 -	- - - - -				
BHCS <i>rel</i>	Branch if Half Carry Bit Set (if H = 1)	REL	29 rr	3	ppp	- 1 1 -	- - - - -				
BHI <i>rel</i>	Branch if Higher (if $C \mid Z = 0$ )	REL	22 rr	3	ppp	- 1 1 -	- - - - -				
BHS <i>rel</i>	Branch if Higher or Same (if C = 0) (Same as BCC)	REL	24 rr	3	ppp	- 1 1 -	- - - - -				
BIH <i>rel</i>	Branch if IRQ Pin High (if IRQ pin = 1)	REL	2F rr	3	ppp	- 1 1 -	- - - - -				
BIL <i>rel</i>	Branch if IRQ Pin Low (if IRQ pin = 0)	REL	2E rr	3	ppp	- 1 1 -	- - - - -				
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 1 1 -	- $\updownarrow$ -				
BLE <i>rel</i>	Branch if Less Than or Equal To (if $Z \mid (N \oplus V) = 1$ ) (Signed)	REL	93 rr	3	ppp	- 1 1 -	- - - - -				
BLO <i>rel</i>	Branch if Lower (if C = 1) (Same as BCS)	REL	25 rr	3	ppp	- 1 1 -	- - - - -				
BLS <i>rel</i>	Branch if Lower or Same (if $C \mid Z = 1$ )	REL	23 rr	3	ppp	- 1 1 -	- - - - -				
BLT <i>rel</i>	Branch if Less Than (if $N \oplus V = 1$ ) (Signed)	REL	91 rr	3	ppp	- 1 1 -	- - - - -				
BMC <i>rel</i>	Branch if Interrupt Mask Clear (if I = 0)	REL	2C rr	3	ppp	- 1 1 -	- - - - -				
BMI <i>rel</i>	Branch if Minus (if N = 1)	REL	2B rr	3	ppp	- 1 1 -	- - - - -				
BMS <i>rel</i>	Branch if Interrupt Mask Set (if I = 1)	REL	2D rr	3	ppp	- 1 1 -	- - - - -				
BNE <i>rel</i>	Branch if Not Equal (if Z = 0)	REL	26 rr	3	ppp	- 1 1 -	- - - - -				
BPL <i>rel</i>	Branch if Plus (if N = 0)	REL	2A rr	3	ppp	- 1 1 -	- - - - -				

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							
						V	I	H	I	N	Z	C	
BRA <i>rel</i>	Branch Always (if I = 1)	REL	20 rr	3	ppp	-	1	1	-	-	-	-	-
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	-	1	1	-	-	-	-	↓
		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	-	1	1	-	-	-	-	-
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	-	1	1	-	-	-	-	↓
		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	rfwpp	-	1	1	-	-	-	-	-
		DIR (b1)	12 dd	5	rfwpp								
		DIR (b2)	14 dd	5	rfwpp								
		DIR (b3)	16 dd	5	rfwpp								
		DIR (b4)	18 dd	5	rfwpp								
		DIR (b5)	1A dd	5	rfwpp								
		DIR (b6)	1C dd	5	rfwpp								
		DIR (b7)	1E dd	5	rfwpp								
BSR <i>rel</i>	Branch to Subroutine PC ← (PC) + \$0002 push (PCL); SP ← (SP) – \$0001 push (PCH); SP ← (SP) – \$0001 PC ← (PC) + <i>rel</i>	REL	AD rr	5	ssppp	-	1	1	-	-	-	-	-
CBEQ <i>opr8a,rel</i>	Compare and... Branch if (A) = (M)	DIR	31 dd rr	5	rpppp	-	1	1	-	-	-	-	-
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	rfppp								
CBEQ <i>opr8,SP,rel</i>		SP1	9E 61 ff rr	6	prpppp								
CLC		Clear Carry Bit (C ← 0)	INH	98	1								
CLI	Clear Interrupt Mask Bit (I ← 0)	INH	9A	1	p	-	1	1	-	0	-	-	-
CLR <i>opr8a</i>	Clear M ← \$00 A ← \$00 X ← \$00 H ← \$00 M ← \$00 M ← \$00 M ← \$00	DIR	3F dd	5	rfwpp	0	1	1	-	-	0	1	-
CLRA		INH	4F	1	p								
CLRX		INH	5F	1	p								
CLRH		INH	8C	1	p								
CLR <i>opr8,X</i>		IX1	6F ff	5	rfwpp								
CLR <i>,X</i>		IX	7F	4	rfwp								
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								
						V	I	H	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		↑	1	1	–	–	↑	↓	↑
COM opr8a COMA COMX COM oprx8,X COM ,X COM oprx8,SP	Complement (One's Complement) M ← $(\overline{M}) = \$FF - (M)$ A ← $(\overline{A}) = \$FF - (A)$ X ← $(\overline{X}) = \$FF - (X)$ M ← $(\overline{M}) = \$FF - (M)$ M ← $(\overline{M}) = \$FF - (M)$ M ← $(\overline{M}) = \$FF - (M)$	DIR INH INH IX1 IX SP1	33 dd 43 53 63 ff 73 9E 63 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp		0	1	1	–	–	↑	↓	↑
CPHX opr16a CPHX #opr16i CPHX opr8a CPHX oprx8,SP	Compare Index Register (H:X) with Memory (H:X) – (M:M + \$0001) (CCR Updated But Operands Not Changed)	EXT IMM DIR SP1	3E hh ll 65 jj kk 75 dd 9E F3 ff	6 3 5 6	prrfpp ppp rrfpp prrfpp		↑	1	1	–	–	↑	↓	↑
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		↑	1	1	–	–	↑	↓	↑
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	INH	72	1	p		U	1	1	–	–	↑	↓	↑
DBNZ opr8a,rel DBNZA rel DBNZX rel DBNZ oprx8,X,rel DBNZ ,X,rel DBNZ oprx8,SP,rel	Decrement A, X, or M and Branch if Not Zero (if (result) ≠ 0) DBNZX Affects X Not H	DIR INH INH IX1 IX SP1	3B dd rr 4B rr 5B rr 6B ff rr 7B rr 9E 6B ff rr	7 4 4 7 6 8	rfwpppp fppp fppp rfwpppp rfwppp prfwpppp		–	1	1	–	–	–	–	–
DEC opr8a DECA DECX DEC oprx8,X DEC ,X DEC oprx8,SP	Decrement M ← (M) – \$01 A ← (A) – \$01 X ← (X) – \$01 M ← (M) – \$01 M ← (M) – \$01 M ← (M) – \$01	DIR INH INH IX1 IX SP1	3A dd 4A 5A 6A ff 7A 9E 6A ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp		↑	1	1	–	–	↑	↓	–
DIV	Divide A ← (H:A) ÷ (X); H ← Remainder	INH	52	6	ffffffp		–	1	1	–	–	–	↑	↓
EOR #opr8i EOR opr8a EOR opr16a EOR oprx16,X EOR oprx8,X EOR ,X EOR oprx16,SP EOR oprx8,SP	Exclusive OR Memory with Accumulator A ← (A ⊕ M)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A8 ii B8 dd C8 hh ll D8 ee ff E8 ff F8 9E D8 ee ff 9E E8 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp		0	1	1	–	–	↑	↓	–



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

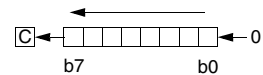
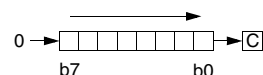
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR													
						V	I	H	I	N	Z	C							
INC <i>opr8a</i> INCA INCX INC <i>opr8,X</i> INC ,X INC <i>opr8,SP</i>	Increment $M \leftarrow (M) + \$01$ $A \leftarrow (A) + \$01$ $X \leftarrow (X) + \$01$ $M \leftarrow (M) + \$01$ $M \leftarrow (M) + \$01$ $M \leftarrow (M) + \$01$	DIR INH INH IX1 IX SP1	3C dd 4C 5C 6C ff 7C 9E 6C ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp														↑1 1 - - ↓↑ -
JMP <i>opr8a</i> JMP <i>opr16a</i> JMP <i>opr16,X</i> JMP <i>opr8,X</i> JMP ,X	Jump $PC \leftarrow \text{Jump Address}$	DIR EXT IX2 IX1 IX	BC dd CC hh ll DC ee ff EC ff FC	3 4 4 3 3	ppp pppp pppp ppp ppp														- 1 1 - - - - -
JSR <i>opr8a</i> JSR <i>opr16a</i> JSR <i>opr16,X</i> JSR <i>opr8,X</i> JSR ,X	Jump to Subroutine $PC \leftarrow (PC) + n$ ( $n = 1, 2, \text{ or } 3$ ) Push (PCL); $SP \leftarrow (SP) - \$0001$ Push (PCH); $SP \leftarrow (SP) - \$0001$ $PC \leftarrow \text{Unconditional Address}$	DIR EXT IX2 IX1 IX	BD dd CD hh ll DD ee ff ED ff FD	5 6 6 5 5	ssppp pssppp pssppp ssppp ssppp														- 1 1 - - - - -
LDA # <i>opr8i</i> LDA <i>opr8a</i> LDA <i>opr16a</i> LDA <i>opr16,X</i> LDA <i>opr8,X</i> LDA ,X LDA <i>opr16,SP</i> LDA <i>opr8,SP</i>	Load Accumulator from Memory $A \leftarrow (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	A6 ii B6 dd C6 hh ll D6 ee ff E6 ff F6 9E D6 ee ff 9E E6 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp														0 1 1 - - ↓↑ -
LDHX # <i>opr16i</i> LDHX <i>opr8a</i> LDHX <i>opr16a</i> LDHX ,X LDHX <i>opr16,X</i> LDHX <i>opr8,X</i> LDHX <i>opr8,SP</i>	Load Index Register (H:X) $H:X \leftarrow (M:M + \$0001)$	IMM DIR EXT IX IX2 IX1 SP1	45 jj kk 55 dd 32 hh ll 9E AE 9E BE ee ff 9E CE ff 9E FE ff	3 4 5 5 6 6 5	ppp rrpp prpp prfp pprrpp prpp prpp														0 1 1 - - ↓↑ -
LDX # <i>opr8i</i> LDX <i>opr8a</i> LDX <i>opr16a</i> LDX <i>opr16,X</i> LDX <i>opr8,X</i> LDX ,X LDX <i>opr16,SP</i> LDX <i>opr8,SP</i>	Load X (Index Register Low) from Memory $X \leftarrow (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE ii BE dd CE hh ll DE ee ff EE ff FE 9E DE ee ff 9E EE ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp														0 1 1 - - ↓↑ -
LSL <i>opr8a</i> LSLA LSLX LSL <i>opr8,X</i> LSL ,X LSL <i>opr8,SP</i>	Logical Shift Left  (Same as ASL)	DIR INH INH IX1 IX SP1	38 dd 48 58 68 ff 78 9E 68 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp														↑1 1 - - ↓↑↑
LSR <i>opr8a</i> LSRA LSRX LSR <i>opr8,X</i> LSR ,X LSR <i>opr8,SP</i>	Logical Shift Right 	DIR INH INH IX1 IX SP1	34 dd 44 54 64 ff 74 9E 64 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp														↑1 1 - - 0↑↑

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

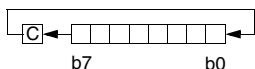
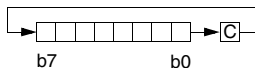
Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V I 1 H	I N Z C
MOV <i>opr8a,opr8a</i> MOV <i>opr8a,X+</i> MOV <i>#opr8i,opr8a</i> MOV <i>,X+,opr8a</i>	Move $(M)_{\text{destination}} \leftarrow (M)_{\text{source}}$ In IX+/DIR and DIR/IX+ Modes, $H:X \leftarrow (H:X) + \$0001$	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E dd dd 5E dd 6E ii dd 7E dd	5 5 4 5	rpwpp rfwpp pwpp rfwpp	0 1 1 -	- $\uparrow$ $\downarrow$ -
MUL	Unsigned multiply $X:A \leftarrow (X) \times (A)$	INH	42	5	ffffp	- 1 1 0	- - - - 0
NEG <i>opr8a</i> NEGA NEGX NEG <i>opr8,X</i> NEG <i>,X</i> NEG <i>opr8,SP</i>	Negate $M \leftarrow -(M) = \$00 - (M)$ (Two's Complement) $A \leftarrow -(A) = \$00 - (A)$ $X \leftarrow -(X) = \$00 - (X)$ $M \leftarrow -(M) = \$00 - (M)$ $M \leftarrow -(M) = \$00 - (M)$ $M \leftarrow -(M) = \$00 - (M)$	DIR INH INH IX1 IX SP1	30 dd 40 50 60 ff 70 9E 60 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	$\uparrow$ 1 1 -	- $\uparrow$ $\downarrow$ $\uparrow$ $\downarrow$
NOP	No Operation — Uses 1 Bus Cycle	INH	9D	1	p	- 1 1 -	- - - - -
NSA	Nibble Swap Accumulator $A \leftarrow (A[3:0]:A[7:4])$	INH	62	1	p	- 1 1 -	- - - - -
ORA <i>#opr8i</i> ORA <i>opr8a</i> ORA <i>opr16a</i> ORA <i>opr8,X</i> ORA <i>opr8,X</i> ORA <i>,X</i> ORA <i>opr8,SP</i> ORA <i>opr16,SP</i> ORA <i>opr8,SP</i>	Inclusive OR Accumulator and Memory $A \leftarrow (A) \mid (M)$	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA ii BA dd CA hh ll DA ee ff EA ff FA 9E DA ee ff 9E EA ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 1 1 -	- $\uparrow$ $\downarrow$ -
PSHA	Push Accumulator onto Stack Push (A); $SP \leftarrow (SP) - \$0001$	INH	87	2	sp	- 1 1 -	- - - - -
PSHH	Push H (Index Register High) onto Stack Push (H); $SP \leftarrow (SP) - \$0001$	INH	8B	2	sp	- 1 1 -	- - - - -
PSHX	Push X (Index Register Low) onto Stack Push (X); $SP \leftarrow (SP) - \$0001$	INH	89	2	sp	- 1 1 -	- - - - -
PULA	Pull Accumulator from Stack $SP \leftarrow (SP + \$0001)$ ; Pull (A)	INH	86	3	ufp	- 1 1 -	- - - - -
PULH	Pull H (Index Register High) from Stack $SP \leftarrow (SP + \$0001)$ ; Pull (H)	INH	8A	3	ufp	- 1 1 -	- - - - -
PULX	Pull X (Index Register Low) from Stack $SP \leftarrow (SP + \$0001)$ ; Pull (X)	INH	88	3	ufp	- 1 1 -	- - - - -
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	$\uparrow$ 1 1 -	- $\uparrow$ $\downarrow$ $\uparrow$ $\downarrow$
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	$\uparrow$ 1 1 -	- $\uparrow$ $\downarrow$ $\uparrow$ $\downarrow$

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

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								
						V	1	H	I	N	Z	C		
SUB #opr8i SUB opr8a SUB opr16a SUB oprx16,X SUB oprx8,X SUB ,X SUB oprx16,SP SUB oprx8,SP	Subtract $A \leftarrow (A) - (M)$	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 $PC \leftarrow (PC) + \$0001$ Push (PCL); $SP \leftarrow (SP) - \$0001$ Push (PCH); $SP \leftarrow (SP) - \$0001$ Push (X); $SP \leftarrow (SP) - \$0001$ Push (A); $SP \leftarrow (SP) - \$0001$ Push (CCR); $SP \leftarrow (SP) - \$0001$ $I \leftarrow 1$ ; PCH $\leftarrow$ Interrupt Vector High Byte PCL $\leftarrow$ Interrupt Vector Low Byte	INH	83	11	sssssvvfppp	-	1	1	-	1	-	-	-	-
TAP	Transfer Accumulator to CCR $CCR \leftarrow (A)$	INH	84	1	p	$\uparrow$	1	$\uparrow$		$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
TAX	Transfer Accumulator to X (Index Register Low) $X \leftarrow (A)$	INH	97	1	p	-	1	1	-	-	-	-	-	-
TPA	Transfer CCR to Accumulator $A \leftarrow (CCR)$	INH	85	1	p	-	1	1	-	-	-	-	-	-
TST opr8a TSTA TSTX TST oprx8,X TST ,X TST oprx8,SP	Test for Negative or Zero (M) - \$00 (A) - \$00 (X) - \$00 (M) - \$00 (M) - \$00 (M) - \$00	DIR INH INH IX1 IX SP1	3D dd 4D 5D 6D ff 7D 9E 6D ff	4 1 1 4 3 5	rfpp p p rfpp rfp prfpp	0	1	1	-					
TSX	Transfer SP to Index Reg. $H:X \leftarrow (SP) + \$0001$	INH	95	2	fp	-	1	1	-	-	-	-	-	-
TXA	Transfer X (Index Reg. Low) to Accumulator $A \leftarrow (X)$	INH	9F	1	p	-	1	1	-	-	-	-	-	-

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							
						V	1	H	I	N	Z	C	
TXS	Transfer Index Reg. to SP SP ← (H:X) – \$0001	INH	94	2	f <sub>p</sub>	-	1	1	-	-	-	-	-
WAIT	Enable Interrupts; Wait for Interrupt I bit ← 0; Halt CPU	INH	8F	2+	f <sub>p</sub> . . .	-	1	1	-	0	-	-	-

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

- n* Any label or expression that evaluates to a single integer in the range 0-7.
- opr8i* Any label or expression that evaluates to an 8-bit immediate value.
- opr16i* Any label or expression that evaluates to a 16-bit immediate value.
- opr8a* Any label or expression that evaluates to an 8-bit direct-page address (\$00xx).
- opr16a* Any label or expression that evaluates to a 16-bit address.
- 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 \$FFxx (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 3 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 2	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 5 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 2	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	NOP INH	9D 1 1	NOP INH	AD 5 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 2 1	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 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   **IX+** Indexed, 1-Byte Offset with  
**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

Number of Bytes





## Chapter 8

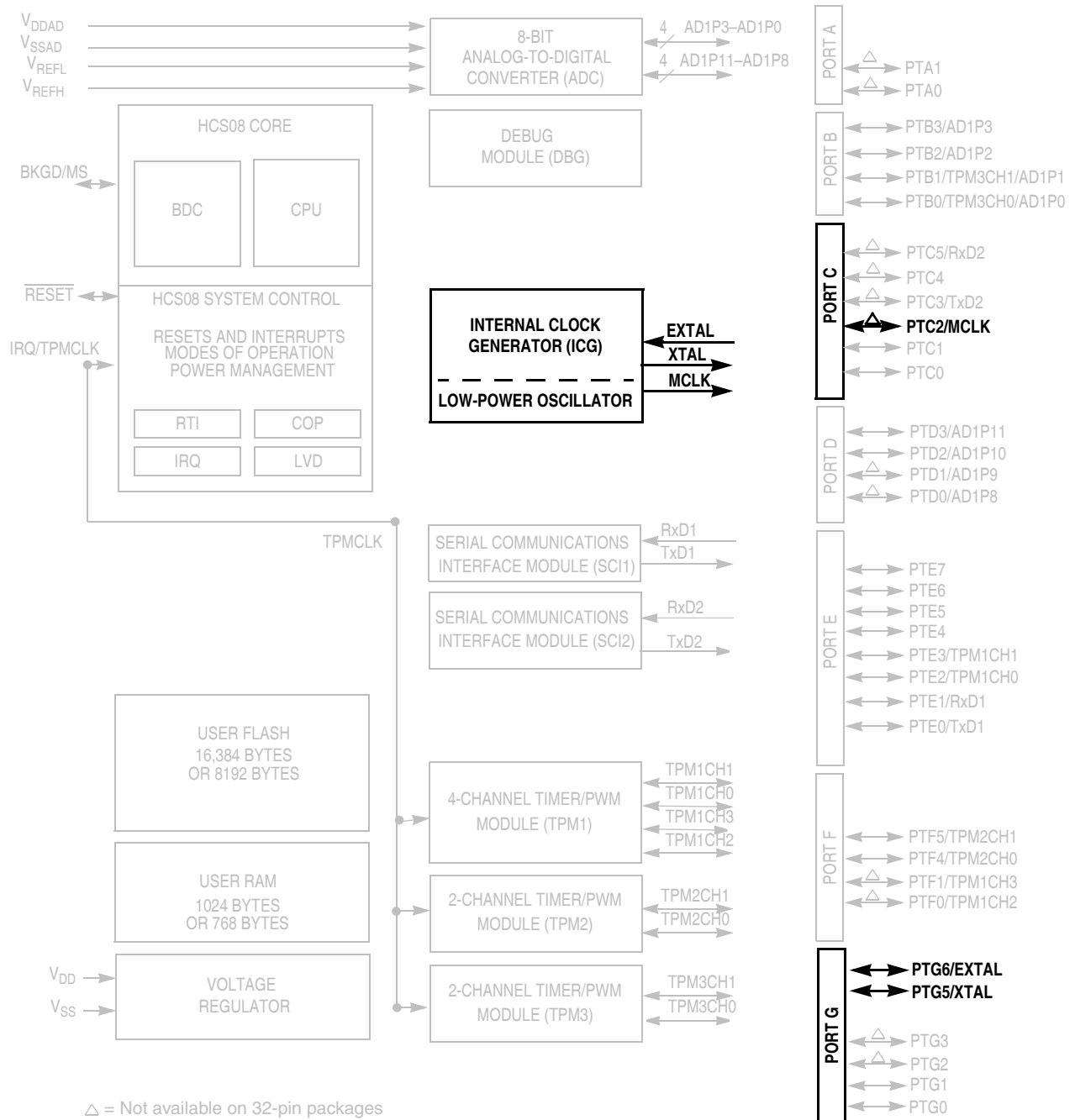
# Internal Clock Generator (S08ICGV4)

The internal clock generation (ICG) module is used to generate the system clocks for the SC9S08MZ16 MCU. The analog supply lines  $V_{DDA}$  and  $V_{SSA}$  are internally derived from the MCU's  $V_{DD}$  and  $V_{SS}$  pins. Refer to [Appendix A, “Electrical Characteristics and Timing Specifications.”](#) for electrical data.

### NOTE

Freescale Semiconductor recommends that flash location 0xFFBE be reserved to store a nonvolatile version of ICGTRM. This will allow debugger and programmer to perform a manual trim operation and store the resultant ICGTRM value for users to access at a later time.

The internal bus frequency in SC9S08MZ16 is up to 10 MHz. This chapter reserves the description of up to 20 MHz to be compatible with the data sheet of other Freescale 8-bit MCUs.



**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 should not be driven above  $V_{DD}$ .
4. Pin contains integrated pullup device.
5. PTD3 and PTD2 contain both pullup and pulldown devices.

**Figure 8-1. SC9S08MZ16 Block Diagram Highlighting ICG Block and Pins**

## 8.1 Introduction

The ICG provides multiple options for clock sources. This offers a user great flexibility when making choices between cost, precision, current draw, and performance. As seen in [Figure 8-2](#), the ICG consists of four functional blocks. Each of these is briefly described here and then in more detail in a later section.

- **Oscillator block** — The oscillator block provides means for connecting an external crystal or resonator. Two frequency ranges are software selectable to allow optimal startup and stability. Alternatively, the oscillator block can be used to route an external square wave to the system clock. External sources can provide a very precise clock source. The oscillator is capable of being configured for low power mode or high amplitude mode as selected by HGO.
- **Internal reference generator** — The internal reference generator consists of two controlled clock sources. One is designed to be approximately 8 MHz and can be selected as a local clock for the background debug controller. The other internal reference clock source is typically 243 kHz and can be trimmed for finer accuracy via software when a precise timed event is input to the MCU. This provides a highly reliable, low-cost clock source.
- **Frequency-locked loop** — A frequency-locked loop (FLL) stage takes either the internal or external clock source and multiplies it to a higher frequency. Status bits provide information when the circuit has achieved lock and when it falls out of lock. Additionally, this block can monitor the external reference clock and signals whether the clock is valid or not.
- **Clock select block** — The clock select block provides several switch options for connecting different clock sources to the system clock tree. ICGDCLK is the multiplied clock frequency out of the FLL, ICGERCLK is the reference clock frequency from the crystal or external clock source, and FFE (fixed frequency enable) is a control signal used to control the system fixed frequency clock (XCLK). ICGLCLK is the clock source for the background debug controller (BDC).

### 8.1.1 Features

The module is intended to be very user friendly with many of the features occurring automatically without user intervention. To quickly configure the module, go to [Section 8.6, “Initialization/Application Information”](#) and pick an example that best suits the application needs.

Features of the ICG and clock distribution system:

- Several options for the primary clock source allow a wide range of cost, frequency, and precision choices:
  - 32 kHz–100 kHz crystal or resonator
  - 1 MHz–16 MHz crystal or resonator
  - External clock
  - Internal reference generator
- Defaults to self-clocked mode to minimize startup delays
- Frequency-locked loop (FLL) generates 8 MHz to 40 MHz (for bus rates up to 20 MHz)
  - Uses external or internal clock as reference frequency
- Automatic lockout of non-running clock sources
- Reset or interrupt on loss of clock or loss of FLL lock

- Digitally-controlled oscillator (DCO) preserves previous frequency settings, allowing fast frequency lock when recovering from stop3 mode
- DCO will maintain operating frequency during a loss or removal of reference clock
- Post-FLL divider selects 1 of 8 bus rate divisors (/1 through /128)
- Separate self-clocked source for real-time interrupt
- Trimmable internal clock source supports SCI communications without additional external components
- Automatic FLL engagement after lock is acquired
- External oscillator selectable for low power or high gain

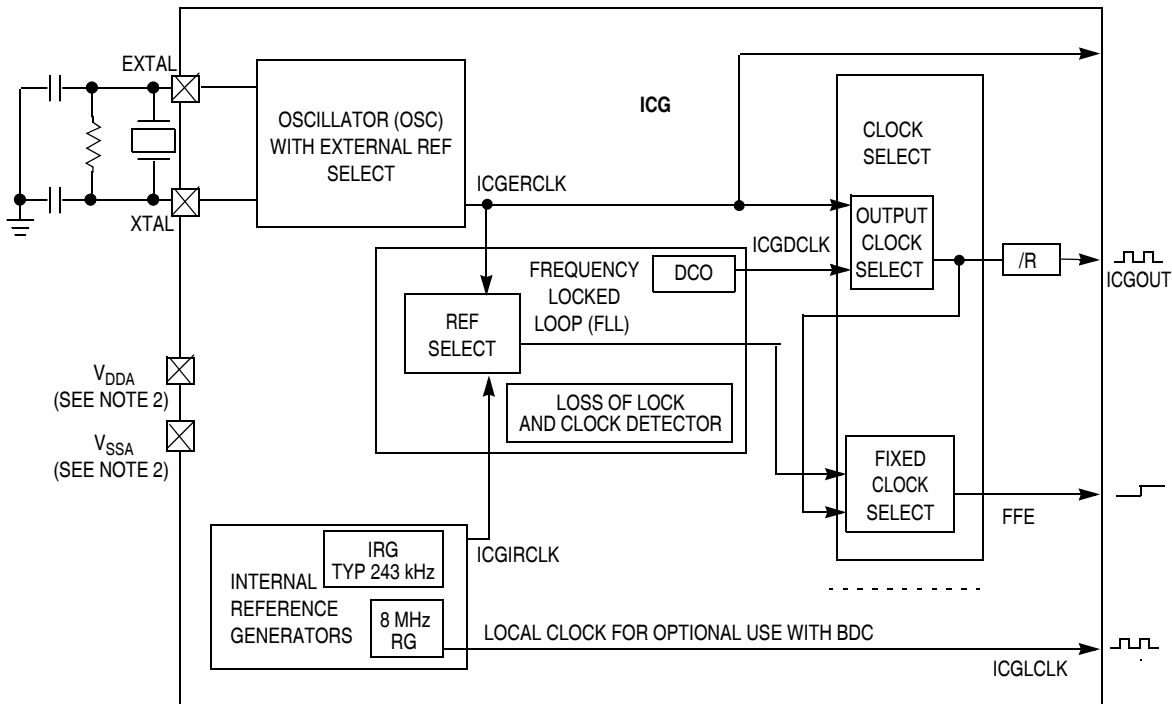
## 8.1.2 Modes of Operation

This is a high-level description only. Detailed descriptions of operating modes are contained in [Section 8.5, “Functional Description.”](#)

- Mode 1 — Off  
The output clock, ICGOUT, is static. This mode may be entered when the STOP instruction is executed.
- Mode 2 — Self-clocked (SCM)  
Default mode of operation that is entered immediately after reset. The ICG’s FLL is open loop and the digitally controlled oscillator (DCO) is free running at a frequency set by the filter bits.
- Mode 3 — FLL engaged internal (FEI)  
In this mode, the ICG’s FLL is used to create frequencies that are programmable multiples of the internal reference clock.
  - FLL engaged internal unlocked is a transition state that occurs while the FLL is attempting to lock. The FLL DCO frequency is off target and the FLL is adjusting the DCO to match the target frequency.
  - FLL engaged internal locked is a state that occurs when the FLL detects that the DCO is locked to a multiple of the internal reference.
- Mode 4 — FLL bypassed external (FBE)  
In this mode, the ICG is configured to bypass the FLL and use an external clock as the clock source.
- Mode 5 — FLL engaged external (FEE)  
The ICG’s FLL is used to generate frequencies that are programmable multiples of the external clock reference.
  - FLL engaged external unlocked is a transition state that occurs while the FLL is attempting to lock. The FLL DCO frequency is off target and the FLL is adjusting the DCO to match the target frequency.
  - FLL engaged external locked is a state which occurs when the FLL detects that the DCO is locked to a multiple of the internal reference.

### 8.1.3 Block Diagram

Figure 8-2 is a top-level diagram that shows the functional organization of the internal clock generation (ICG) module. This section includes a general description and a feature list.



NOTES:

1. See Table 8-1 for specific use of ICGOUT, FFE, ICGLCLK, ICGERCLK
2. Not all HCS08 microcontrollers have unique supply pins for the ICG. See the device pin assignments.

Figure 8-2. ICG Block Diagram

## 8.2 External Signal Description

The oscillator pins are used to provide an external clock source for the MCU. The oscillator pins are gain controlled in low-power mode (default). Oscillator amplitudes in low-power mode are limited to approximately 1 V, peak-to-peak.

### 8.2.1 EXTAL — External Reference Clock / Oscillator Input

If upon the first write to ICGC1, either the FEE mode or FBE mode is selected, this pin functions as either the external clock input or the input of the oscillator circuit as determined by REFS. If upon the first write to ICGC1, either the FEI mode or SCM mode is selected, this pin is not used by the ICG.

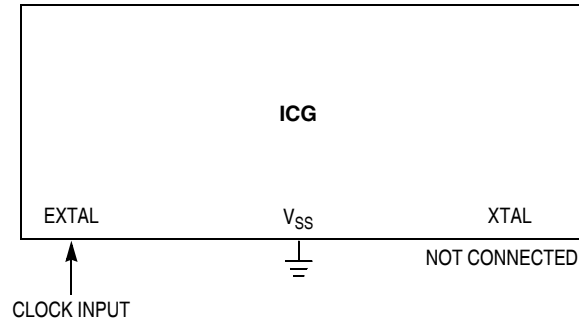
### 8.2.2 XTAL — Oscillator Output

If upon the first write to ICGC1, either the FEE mode or FBE mode is selected, this pin functions as the output of the oscillator circuit. If upon the first write to ICGC1, either the FEI mode or SCM mode is

selected, this pin is not used by the ICG. The oscillator is capable of being configured to provide a higher amplitude output for improved noise immunity. This mode of operation is selected by  $HGO = 1$ .

### 8.2.3 External Clock Connections

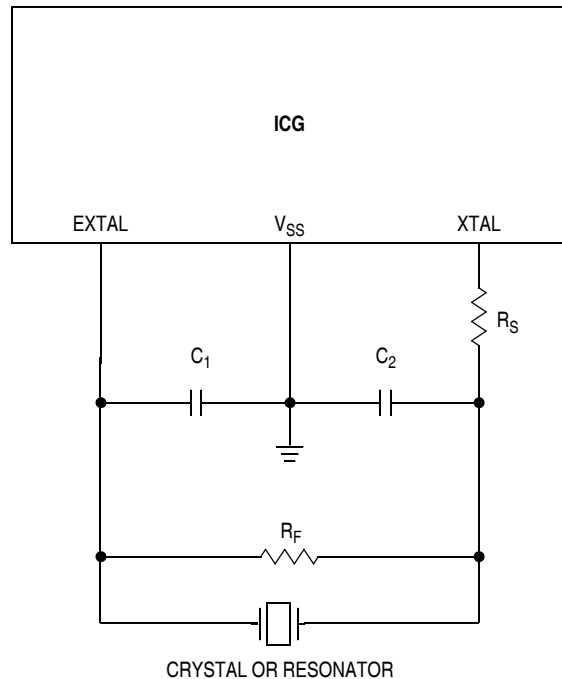
If an external clock is used, then the pins are connected as shown [Figure 8-3](#).



**Figure 8-3. External Clock Connections**

### 8.2.4 External Crystal/Resonator Connections

If an external crystal/resonator frequency reference is used, then the pins are connected as shown in [Figure 8-4](#). Recommended component values are listed in the [Electrical Characteristics](#) chapter.

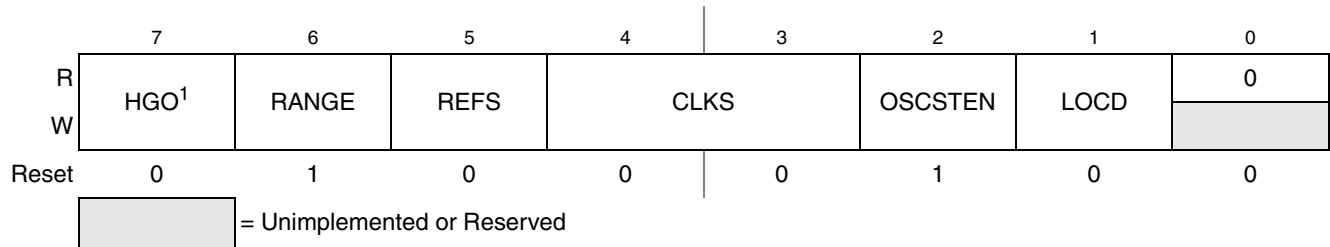


**Figure 8-4. External Frequency Reference Connection**

## 8.3 Register Definition

Refer to the direct-page register summary in the [Memory](#) chapter of this data sheet for the absolute address assignments for all ICG 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.

### 8.3.1 ICG Control Register 1 (ICGC1)



<sup>1</sup> This bit can be written only once after reset. Additional writes are ignored.

**Figure 8-5. ICG Control Register 1 (ICGC1)**

**Table 8-1. ICGC1 Register Field Descriptions**

Field	Description
7 HGO	<b>High Gain Oscillator Select</b> — The HGO bit is used to select between low power operation and high gain operation for improved noise immunity. This bit is write-once after reset. 0 Oscillator configured for low power operation. 1 Oscillator configured for high gain operation.
6 RANGE	<b>Frequency Range Select</b> — The RANGE bit controls the oscillator, reference divider, and FLL loop prescaler multiplication factor (P). It selects one of two reference frequency ranges for the ICG. The RANGE bit is write-once after a reset. The RANGE bit only has an effect in FLL engaged external and FLL bypassed external modes. 0 Oscillator configured for low frequency range. FLL loop prescale factor P is 64. 1 Oscillator configured for high frequency range. FLL loop prescale factor P is 1.
5 REFS	<b>External Reference Select</b> — The REFS bit controls the external reference clock source for ICGERCLK. The REFS bit is write-once after a reset. 0 External clock requested. 1 Oscillator using crystal or resonator requested.
4:3 CLKS	<b>Clock Mode Select</b> — The CLKS bits control the clock mode as described below. If FLL bypassed external is requested, it will not be selected until ERCS = 1. If the ICG enters off mode, the CLKS bits will remain unchanged. Writes to the CLKS bits will not take effect if a previous write is not complete. 00 Self-clocked 01 FLL engaged, internal reference 10 FLL bypassed, external reference 11 FLL engaged, external reference The CLKS bits are writable at any time, unless the first write after a reset was CLKS = 0X, the CLKS bits cannot be written to 1X until after the next reset (because the EXTAL pin was not reserved).

Table 8-1. ICGC1 Register Field Descriptions (continued)

Field	Description
2 OSCSTEN	<b>Enable Oscillator in Off Mode</b> — The OSCSTEN bit controls whether or not the oscillator circuit remains enabled when the ICG enters off mode. This bit has no effect if HGO = 1 and RANGE = 1. 0 Oscillator disabled when ICG is in off mode unless ENABLE is high, CLKS = 10, and REFST = 1. 1 Oscillator enabled when ICG is in off mode, CLKS = 1X and REFST = 1.
1 LOCD	<b>Loss of Clock Disable</b> 0 Loss of clock detection enabled. 1 Loss of clock detection disabled.

## 8.4 ICG Control Register 2 (ICGC2)



Figure 8-6. ICG Control Register 2 (ICGC2)

Table 8-2. ICGC2 Register Field Descriptions

Field	Description
7 LOLRE	<b>Loss of Lock Reset Enable</b> — The LOLRE bit determines what type of request is made by the ICG following a loss of lock indication. The LOLRE bit only has an effect when LOLS is set. 0 Generate an interrupt request on loss of lock. 1 Generate a reset request on loss of lock.
6:4 MFD	<b>Multiplication Factor</b> — The MFD bits control the programmable multiplication factor in the FLL loop. The value specified by the MFD bits establishes the multiplication factor (N) applied to the reference frequency. Writes to the MFD bits will not take effect if a previous write is not complete. Select a low enough value for N such that $f_{ICGDCLK}$ does not exceed its maximum specified value. 000 Multiplication factor = 4 001 Multiplication factor = 6 010 Multiplication factor = 8 011 Multiplication factor = 10 100 Multiplication factor = 12 101 Multiplication factor = 14 110 Multiplication factor = 16 111 Multiplication factor = 18



Table 8-2. ICGC2 Register Field Descriptions (continued)

Field	Description
3 LOCRE	<b>Loss of Clock Reset Enable</b> — The LOCRE bit determines how the system manages a loss of clock condition. 0 Generate an interrupt request on loss of clock. 1 Generate a reset request on loss of clock.
2:0 RFD	<b>Reduced Frequency Divider</b> — The RFD bits control the value of the divider following the clock select circuitry. The value specified by the RFD bits establishes the division factor (R) applied to the selected output clock source. Writes to the RFD bits will not take effect if a previous write is not complete. 000 Division factor = 1 001 Division factor = 2 010 Division factor = 4 011 Division factor = 8 100 Division factor = 16 101 Division factor = 32 110 Division factor = 64 111 Division factor = 128

### 8.4.1 ICG Status Register 1 (ICGS1)

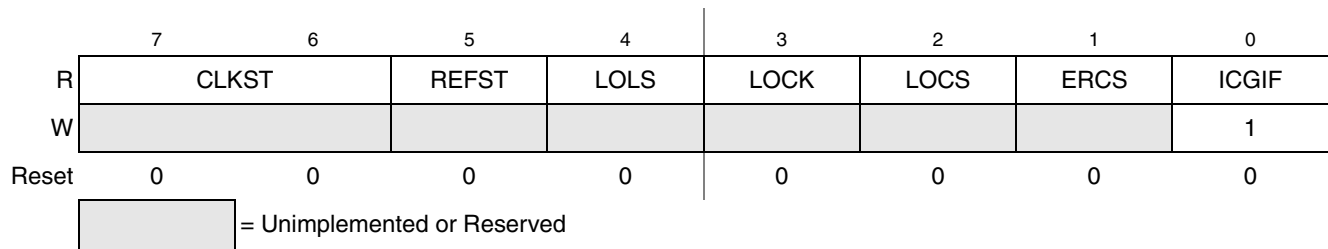


Figure 8-7. ICG Status Register 1 (ICGS1)

Table 8-3. ICGS1 Register Field Descriptions

Field	Description
7:6 CLKST	<b>Clock Mode Status</b> — The CLKST bits indicate the current clock mode. The CLKST bits don't update immediately after a write to the CLKS bits due to internal synchronization between clock domains. 00 Self-clocked 01 FLL engaged, internal reference 10 FLL bypassed, external reference 11 FLL engaged, external reference
5 REFST	<b>Reference Clock Status</b> — The REFST bit indicates which clock reference is currently selected by the Reference Select circuit. 0 External Clock selected. 1 Crystal/Resonator selected.
4 LOLS	<b>FLL Loss of Lock Status</b> — The LOLS bit is a sticky indication of FLL lock status. 0 FLL has not unexpectedly lost lock since LOLS was last cleared. 1 FLL has unexpectedly lost lock since LOLS was last cleared, LOLRE determines action taken. FLL has unexpectedly lost lock since LOLS was last cleared, LOLRE determines action taken.
3 LOCK	<b>FLL Lock Status</b> — The LOCK bit indicates whether the FLL has acquired lock. The LOCK bit is cleared in off, self-clocked, and FLL bypassed modes. 0 FLL is currently unlocked. 1 FLL is currently locked.

Table 8-3. ICGS1 Register Field Descriptions (continued)

Field	Description
2 LOCS	<b>Loss Of Clock Status</b> — The LOCS bit is an indication of ICG loss of clock status. 0 ICG has not lost clock since LOCS was last cleared. 1 ICG has lost clock since LOCS was last cleared, LOCRE determines action taken.
1 ERCS	<b>External Reference Clock Status</b> — The ERCS bit is an indication of whether or not the external reference clock (ICGERCLK) meets the minimum frequency requirement. 0 External reference clock is not stable, frequency requirement is not met. 1 External reference clock is stable, frequency requirement is met.
0 ICGIF	<b>ICG Interrupt Flag</b> — The ICGIF read/write flag is set when an ICG interrupt request is pending. It is cleared by a reset or by reading the ICG status register when ICGIF is set and then writing a logic 1 to ICGIF. If another ICG interrupt occurs before the clearing sequence is complete, the sequence is reset so ICGIF would remain set after the clear sequence was completed for the earlier interrupt. Writing a logic 0 to ICGIF has no effect. 0 No ICG interrupt request is pending. 1 An ICG interrupt request is pending.

## 8.4.2 ICG Status Register 2 (ICGS2)

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

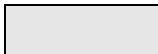
 = Unimplemented or Reserved

Figure 8-8. ICG Status Register 2 (ICGS2)

Table 8-4. ICGS2 Register Field Descriptions

Field	Description
0 DCOS	<b>DCO Clock Stable</b> — The DCOS bit is set when the DCO clock (ICG2DCLK) is stable, meaning the count error has not changed by more than $n_{unlock}$ for two consecutive samples and the DCO clock is not static. This bit is used when exiting off state if $CLKS = X1$ to determine when to switch to the requested clock mode. It is also used in self-clocked mode to determine when to start monitoring the DCO clock. This bit is cleared upon entering the off state. 0 DCO clock is unstable. 1 DCO clock is stable.

## 8.4.3 ICG Filter Registers (ICGFLTU, ICGFLTL)

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


 = Unimplemented or Reserved

Figure 8-9. ICG Upper Filter Register (ICGFLTU)

Table 8-5. ICGFLTU Register Field Descriptions

Field	Description
3:0 FLT	<b>Filter Value</b> — The FLT bits indicate the current filter value, which controls the DCO frequency. The FLT bits are read only except when the CLKS bits are programmed to self-clocked mode (CLKS = 00). In self-clocked mode, any write to ICGFLTU updates the current 12-bit filter value. Writes to the ICGFLTU register will not affect FLT if a previous latch sequence is not complete.

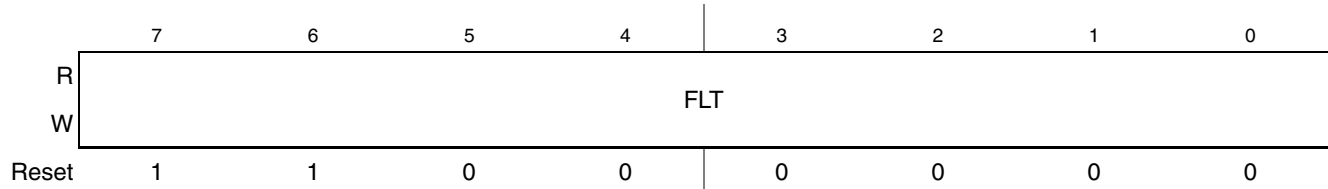
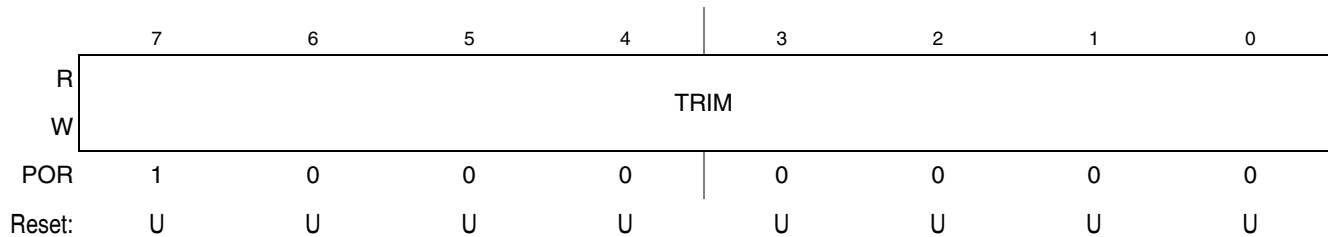


Figure 8-10. ICG Lower Filter Register (ICGFLTU)

Table 8-6. ICGFLTU Register Field Descriptions

Field	Description
7:0 FLT	<b>Filter Value</b> — The FLT bits indicate the current filter value, which controls the DCO frequency. The FLT bits are read only except when the CLKS bits are programmed to self-clocked mode (CLKS = 00). In self-clocked mode, any write to ICGFLTU updates the current 12-bit filter value. Writes to the ICGFLTU register will not affect FLT if a previous latch sequence is not complete. The filter registers show the filter value (FLT).

#### 8.4.4 ICG Trim Register (ICGTRM)



U = Unaffected by MCU reset

Figure 8-11. ICG Trim Register (ICGTRM)

Table 8-7. ICGTRM Register Field Descriptions

Field	Description
7 TRIM	<b>ICG Trim Setting</b> — The TRIM bits control the internal reference generator frequency. They allow a $\pm 25\%$ adjustment of the nominal (POR) period. The bit's effect on period is binary weighted (i.e., bit 1 will adjust twice as much as changing bit 0). Increasing the binary value in TRIM will increase the period and decreasing the value will decrease the period.

## 8.5 Functional Description

This section provides a functional description of each of the five operating modes of the ICG. Also discussed are the loss of clock and loss of lock errors and requirements for entry into each mode. The ICG

is very flexible, and in some configurations, it is possible to exceed certain clock specifications. When using the FLL, configure the ICG so that the frequency of ICGDCLK does not exceed its maximum value to ensure proper MCU operation.

### 8.5.1 Off Mode (Off)

Normally when the CPU enters stop mode, the ICG will cease all clock activity and is in the off state. However there are two cases to consider when clock activity continues while the CPU is in stop mode,

#### 8.5.1.1 BDM Active

When the BDM is enabled, the ICG continues activity as originally programmed. This allows access to memory and control registers via the BDC controller.

#### 8.5.1.2 OSCSTEN Bit Set

When the oscillator is enabled in stop mode ( $OSCSTEN = 1$ ), the individual clock generators are enabled but the clock feed to the rest of the MCU is turned off. This option is provided to avoid long oscillator startup times if necessary, or to run the RTI from the oscillator during stop3.

#### 8.5.1.3 Stop/Off Mode Recovery

Upon the CPU exiting stop mode due to an interrupt, the previously set control bits are valid and the system clock feed resumes. If FEE is selected, the ICG will source the internal reference until the external clock is stable. If FBE is selected, the ICG will wait for the external clock to stabilize before enabling ICGOUT.

Upon the CPU exiting stop mode due to a reset, the previously set ICG control bits are ignored and the default reset values applied. Therefore the ICG will exit stop in SCM mode configured for an approximately 8 MHz DCO output (4 MHz bus clock) with trim value maintained. If using a crystal, 4096 clocks are detected prior to engaging ICGERCLK. This is incorporated in crystal start-up time.

### 8.5.2 Self-Clocked Mode (SCM)

Self-clocked mode (SCM) is the default mode of operation and is entered when any of the following conditions occur:

- After any reset.
- Exiting from off mode when  $CLKS$  does not equal 10. If  $CLKS = X1$ , the ICG enters this state temporarily until the DCO is stable ( $DCOS = 1$ ).
- $CLKS$  bits are written from X1 to 00.
- $CLKS = 1X$  and ICGERCLK is not detected (both  $ERCS = 0$  and  $LOCS = 1$ ).

In this state, the FLL loop is open. The DCO is on, and the output clock signal ICGOUT frequency is given by  $f_{ICGDCLK} / R$ . The ICGDCLK frequency can be varied from 8 MHz to 40 MHz by writing a new value into the filter registers (ICGFLTH and ICGFLTL). This is the only mode in which the filter registers can be written.

If this mode is entered due to a reset,  $f_{ICGDCLK}$  will default to  $f_{Self\_reset}$  which is nominally 8 MHz. If this mode is entered from FLL engaged internal,  $f_{ICGDCLK}$  will maintain the previous frequency. If this mode is entered from FLL engaged external (either by programming CLKS or due to a loss of external reference clock),  $f_{ICGDCLK}$  will maintain the previous frequency, but ICGOUT will double if the FLL was unlocked. If this mode is entered from off mode,  $f_{ICGDCLK}$  will be equal to the frequency of ICGDCLK before entering off mode. If CLKS bits are set to 01 or 11 coming out of the Off state, the ICG enters this mode until ICGDCLK is stable as determined by the DCOS bit. After ICGDCLK is considered stable, the ICG automatically closes the loop by switching to FLL engaged (internal or external) as selected by the CLKS bits.

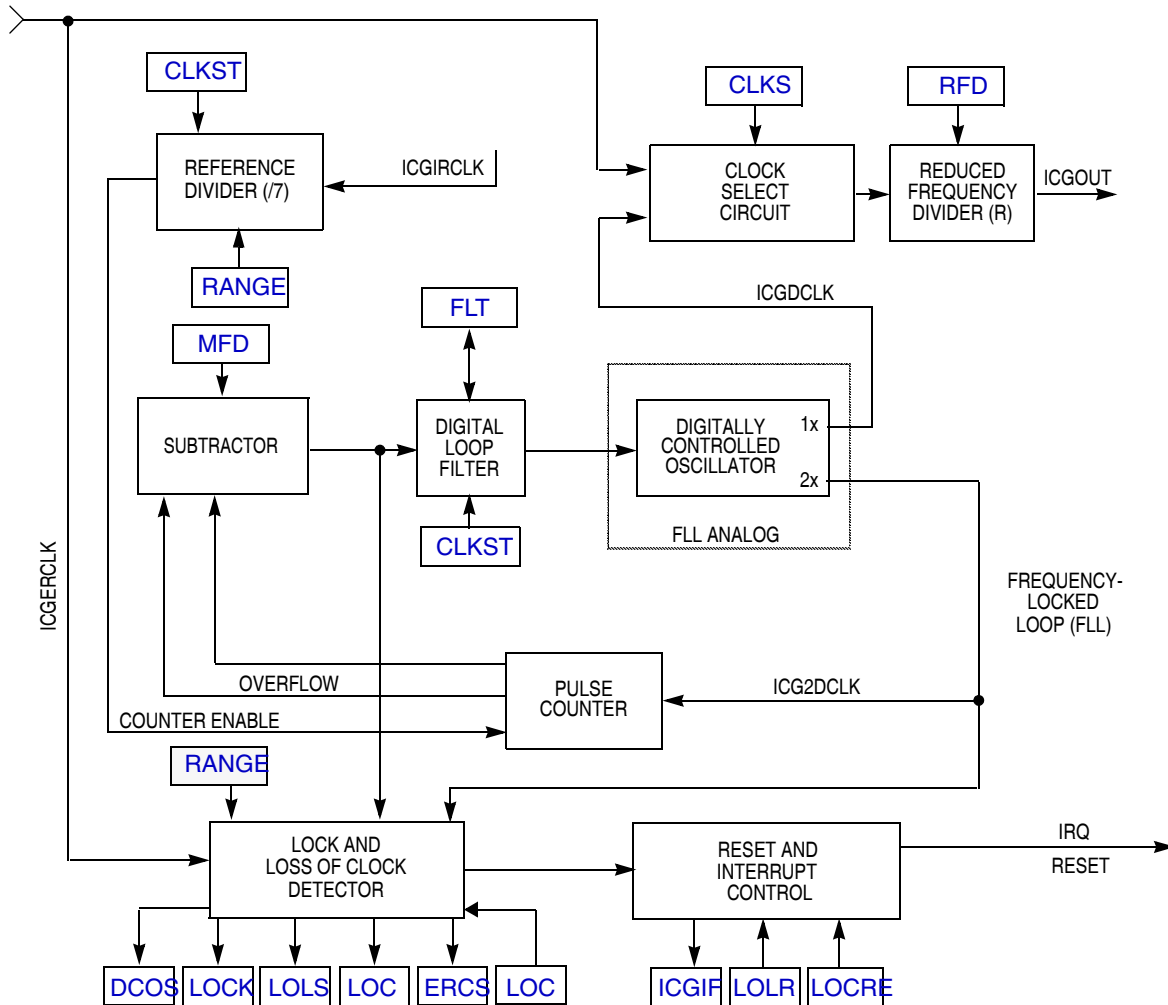


Figure 8-12. Detailed Frequency-Locked Loop Block Diagram

### 8.5.3 FLL Engaged, Internal Clock (FEI) Mode

FLL engaged internal (FEI) is entered when any of the following conditions occur:

- CLKS bits are written to 01
- The DCO clock stabilizes ( $DCOS = 1$ ) while in SCM upon exiting the off state with  $CLKS = 01$

In FLL engaged internal mode, the reference clock is derived from the internal reference clock ICGIRCLK, and the FLL loop will attempt to lock the ICGDCLK frequency to the desired value, as selected by the MFD bits.

### 8.5.4 FLL Engaged Internal Unlocked

FEI unlocked is a temporary state that is entered when FEI is entered and the count error ( $\Delta n$ ) output from the subtractor is greater than the maximum  $n_{\text{unlock}}$  or less than the minimum  $n_{\text{unlock}}$ , as required by the lock detector to detect the unlock condition.

The ICG will remain in this state while the count error ( $\Delta n$ ) is greater than the maximum  $n_{\text{lock}}$  or less than the minimum  $n_{\text{lock}}$ , as required by the lock detector to detect the lock condition.

In this state the output clock signal ICGOUT frequency is given by  $f_{\text{ICGDCLK}} / R$ .

### 8.5.5 FLL Engaged Internal Locked

FLL engaged internal locked is entered from FEI unlocked when the count error ( $\Delta n$ ), which comes from the subtractor, is less than  $n_{\text{lock}}$  (max) and greater than  $n_{\text{lock}}$  (min) for a given number of samples, as required by the lock detector to detect the lock condition. The output clock signal ICGOUT frequency is given by  $f_{\text{ICGDCLK}} / R$ . In FEI locked, the filter value is updated only once every four comparison cycles. The update made is an average of the error measurements taken in the four previous comparisons.

### 8.5.6 FLL Bypassed, External Clock (FBE) Mode

FLL bypassed external (FBE) is entered when any of the following conditions occur:

- From SCM when  $\text{CLKS} = 10$  and ERCS is high
- When  $\text{CLKS} = 10$ , ERCS = 1 upon entering off mode, and off is then exited
- From FLL engaged external mode if a loss of DCO clock occurs and the external reference remains valid (both LOCS = 1 and ERCS = 1)

In this state, the DCO and IRG are off and the reference clock is derived from the external reference clock, ICGERCLK. The output clock signal ICGOUT frequency is given by  $f_{\text{ICGERCLK}} / R$ . If an external clock source is used (REFS = 0), then the input frequency on the EXTAL pin can be anywhere in the range 0 MHz to 40 MHz. If a crystal or resonator is used (REFS = 1), then frequency range is either low for RANGE = 0 or high for RANGE = 1.

### 8.5.7 FLL Engaged, External Clock (FEE) Mode

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

- $\text{CLKS} = 11$  and ERCS and DCOS are both high.
- The DCO stabilizes (DCOS = 1) while in SCM upon exiting the off state with  $\text{CLKS} = 11$ .

In FEE mode, the reference clock is derived from the external reference clock ICGERCLK, and the FLL loop will attempt to lock the ICGDCLK frequency to the desired value, as selected by the MFD bits. To run in FEE mode, there must be a working 32 kHz–100 kHz or 2 MHz–10 MHz external clock source. The

maximum external clock frequency is limited to 10 MHz in FEE mode to prevent over-clocking the DCO. The minimum multiplier for the FLL, from Table 8-12 is 4. Because  $4 \times 10 \text{ MHz}$  is 40MHz, which is the operational limit of the DCO, the reference clock cannot be any faster than 10 MHz.

### 8.5.7.1 FLL Engaged External Unlocked

FEE unlocked is entered when FEE is entered and the count error ( $\Delta n$ ) output from the subtractor is greater than the maximum  $n_{\text{unlock}}$  or less than the minimum  $n_{\text{unlock}}$ , as required by the lock detector to detect the unlock condition.

The ICG will remain in this state while the count error ( $\Delta n$ ) is greater than the maximum  $n_{\text{lock}}$  or less than the minimum  $n_{\text{lock}}$ , as required by the lock detector to detect the lock condition.

In this state, the pulse counter, subtractor, digital loop filter, and DCO form a closed loop and attempt to lock it according to their operational descriptions later in this section. Upon entering this state and until the FLL becomes locked, the output clock signal ICGOUT frequency is given by  $f_{\text{ICGDCLK}} / (2 \times R)$ . This extra divide by two prevents frequency overshoots during the initial locking process from exceeding chip-level maximum frequency specifications. After the FLL has locked, if an unexpected loss of lock causes it to re-enter the unlocked state while the ICG remains in FEE mode, the output clock signal ICGOUT frequency is given by  $f_{\text{ICGDCLK}} / R$ .

### 8.5.7.2 FLL Engaged External Locked

FEE locked is entered from FEE unlocked when the count error ( $\Delta n$ ) is less than  $n_{\text{lock}}$  (max) and greater than  $n_{\text{lock}}$  (min) for a given number of samples, as required by the lock detector to detect the lock condition. The output clock signal ICGOUT frequency is given by  $f_{\text{ICGDCLK}}/R$ . In FLL engaged external locked, the filter value is updated only once every four comparison cycles. The update made is an average of the error measurements taken in the four previous comparisons.

## 8.5.8 FLL Lock and Loss-of-Lock Detection

To determine the FLL locked and loss-of-lock conditions, the pulse counter counts the pulses of the DCO for one comparison cycle (see Table 8-9 for explanation of a comparison cycle) and passes this number to the subtractor. The subtractor compares this value to the value in MFD and produces a count error,  $\Delta n$ . To achieve locked status,  $\Delta n$  must be between  $n_{\text{lock}}$  (min) and  $n_{\text{lock}}$  (max). After the FLL has locked,  $\Delta n$  must stay between  $n_{\text{unlock}}$  (min) and  $n_{\text{unlock}}$  (max) to remain locked. If  $\Delta n$  goes outside this range unexpectedly, the LOLS status bit is set and remains set until cleared by software or until the MCU is reset. LOLS is cleared by reading ICGS1 then writing 1 to ICGIF (LOLRE = 0), or by a loss-of-lock induced reset (LOLRE = 1), or by any MCU reset.

If the ICG enters the off state due to stop mode when ENBDM = OSCSTEN = 0, the FLL loses locked status (LOCK is cleared), but LOLS remains unchanged because this is not an unexpected loss-of-lock condition. Though it would be unusual, if ENBDM is cleared to 0 while the MCU is in stop, the ICG enters the off state. Because this is an unexpected stopping of clocks, LOLS will be set when the MCU wakes up from stop.

Expected loss of lock occurs when the MFD or CLKS bits are changed or in FEI mode only, when the TRIM bits are changed. In these cases, the LOCK bit will be cleared until the FLL regains lock, but the LOLS will not be set.

### 8.5.9 FLL Loss-of-Clock Detection

The reference clock and the DCO clock are monitored under different conditions (see Table 8-8). Provided the reference frequency is being monitored, ERCS = 1 indicates that the reference clock meets minimum frequency requirements. When the reference and/or DCO clock(s) are being monitored, if either one falls below a certain frequency,  $f_{LOR}$  and  $f_{LOD}$ , respectively, the LOCS status bit will be set to indicate the error. LOCS will remain set until it is acknowledged or until the MCU is reset. LOCS is cleared by reading ICGS1 then writing 1 to ICGIF (LOCRE = 0), or by a loss-of-clock induced reset (LOCRE = 1), or by any MCU reset.

If the ICG is in FEE, a loss of reference clock causes the ICG to enter SCM, and a loss of DCO clock causes the ICG to enter FBE mode. If the ICG is in FBE mode, a loss of reference clock will cause the ICG to enter SCM. In each case, the CLKST and CLKS bits will be automatically changed to reflect the new state.

If the ICG is in FEE mode when a loss of clock occurs and the ERCS is still set to 1, then the CLKST bits are set to 10 and the ICG reverts to FBE mode.

A loss of clock will also cause a loss of lock when in FEE or FEI modes. Because the method of clearing the LOCS and LOLS bits is the same, this would only be an issue in the unlikely case that LOLRE = 1 and LOCRE = 0. In this case, the interrupt would be overridden by the reset for the loss of lock.

**Table 8-8. Clock Monitoring (When LOCD = 0)**

Mode	CLKS	REFST	ERCS	External Reference Clock Monitored?	DCO Clock Monitored?
Off	0X or 11	X	Forced Low	No	No
	10	0	Forced Low	No	No
	10	1	Real-Time <sup>1</sup>	Yes <sup>(1)</sup>	No
SCM (CLKST = 00)	0X	X	Forced Low	No	Yes <sup>2</sup>
	10	0	Forced High	No	Yes <sup>(2)</sup>
	10	1	Real-Time	Yes	Yes <sup>(2)</sup>
	11	X	Real-Time	Yes	Yes <sup>(2)</sup>
FEI (CLKST = 01)	0X	X	Forced Low	No	Yes
	11	X	Real-Time	Yes	Yes
FBE (CLKST = 10)	10	0	Forced High	No	No
	10	1	Real-Time	Yes	No
FEE (CLKST = 11)	11	X	Real-Time	Yes	Yes

<sup>1</sup> If ENABLE is high (waiting for external crystal start-up after exiting stop).

<sup>2</sup> DCO clock will not be monitored until DCOS = 1 upon entering SCM from off or FLL bypassed external mode.



## 8.5.10 Clock Mode Requirements

A clock mode is requested by writing to CLKS1:CLKS0 and the actual clock mode is indicated by CLKST1:CLKST0. Provided minimum conditions are met, the status shown in CLKST1:CLKST0 should be the same as the requested mode in CLKS1:CLKS0. Table 8-9 shows the relationship between CLKS, CLKST, and ICGOUT. It also shows the conditions for CLKS = CLKST or the reason CLKS ≠ CLKST.

### NOTE

If a crystal will be used before the next reset, then be sure to set REFS = 1 and CLKS = 1x on the first write to the ICGC1 register. Failure to do so will result in “locking” REFS = 0 which will prevent the oscillator amplifier from being enabled until the next reset occurs.

Table 8-9. ICG State Table

Actual Mode (CLKST)	Desired Mode (CLKS)	Range	Reference Frequency ( $f_{\text{REFERENCE}}$ )	Comparison Cycle Time	ICGOUT	Conditions <sup>1</sup> for CLKS = CLKST	Reason CLKS1 ≠ CLKST
Off (XX)	Off (XX)	X	0	—	0	—	—
	FBE (10)	X	0	—	0	—	ERCS = 0
SCM (00)	SCM (00)	X	$f_{\text{ICGIRCLK}}/7^2$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	Not switching from FBE to SCM	—
	FEI (01)	0	$f_{\text{ICGIRCLK}}/7^{(1)}$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	—	DCOS = 0
	FBE (10)	X	$f_{\text{ICGIRCLK}}/7^{(1)}$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	—	ERCS = 0
	FEE (11)	X	$f_{\text{ICGIRCLK}}/7^{(1)}$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	—	DCOS = 0 or ERCS = 0
FEI (01)	FEI (01)	0	$f_{\text{ICGIRCLK}}/7$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	DCOS = 1	—
	FEE (11)	X	$f_{\text{ICGIRCLK}}/7$	$8/f_{\text{ICGIRCLK}}$	ICGDCLK/R	—	ERCS = 0
FBE (10)	FBE (10)	X	0	—	ICGERCLK/R	ERCS = 1	—
	FEE (11)	X	0	—	ICGERCLK/R	—	LOCS = 1 & ERCS = 1
FEE (11)	FEE (11)	0	$f_{\text{ICGERCLK}}$	$2/f_{\text{ICGERCLK}}$	ICGDCLK/R <sup>3</sup>	ERCS = 1 and DCOS = 1	—
		1	$f_{\text{ICGERCLK}}$	$128/f_{\text{ICGERCLK}}$	ICGDCLK/R <sup>(2)</sup>	ERCS = 1 and DCOS = 1	—

<sup>1</sup> CLKST will not update immediately after a write to CLKS. Several bus cycles are required before CLKST updates to the new value.

<sup>2</sup> The reference frequency has no effect on ICGOUT in SCM, but the reference frequency is still used in making the comparisons that determine the DCOS bit

<sup>3</sup> After initial LOCK; will be ICGDCLK/2R during initial locking process and while FLL is re-locking after the MFD bits are changed.

## 8.5.11 Fixed Frequency Clock

The ICG provides a fixed frequency clock output, XCLK, for use by on-chip peripherals. This output is equal to the internal bus clock, BUSCLK, in all modes except FEE. In FEE mode, XCLK is equal to  $ICGERCLK \div 2$  when the following conditions are met:

- $(P \times N) \div R \geq 4$  where P is determined by RANGE (see [Table 8-11](#)), N and R are determined by MFD and RFD respectively (see [Table 8-12](#)).
- LOCK = 1.

If the above conditions are not true, then XCLK is equal to BUSCLK.

When the ICG is in either FEI or SCM mode, XCLK is turned off. Any peripherals which can use XCLK as a clock source must not do so when the ICG is in FEI or SCM mode.

## 8.5.12 High Gain Oscillator

The oscillator has the option of running in a high gain oscillator (HGO) mode, which improves the oscillator's resistance to EMC noise when running in FBE or FEE modes. This option is selected by writing a 1 to the HGO bit in the ICGC1 register. HGO is used with both the high and low range oscillators but is only valid when REFS = 1 in the ICGC1 register. When HGO = 0, the standard low-power oscillator is selected. This bit is writable only once after any reset.

## 8.6 Initialization/Application Information

### 8.6.1 Introduction

The section is intended to give some basic direction on which configuration a user would want to select when initializing the ICG. For some applications, the serial communication link may dictate the accuracy of the clock reference. For other applications, lowest power consumption may be the chief clock consideration. Still others may have lowest cost as the primary goal. The ICG allows great flexibility in choosing which is best for any application.

Table 8-10. ICG Configuration Consideration

	Clock Reference Source = Internal	Clock Reference Source = External
<b>FLL Engaged</b>	<b>FEI</b> $4 \text{ MHz} < f_{\text{Bus}} < 20 \text{ MHz}$ . Medium power (will be less than FEE if oscillator range = high) Good clock accuracy (After IRG is trimmed) <b>Lowest system cost</b> (no external components required) IRG is on. DCO is on. <sup>1</sup>	<b>FEE</b> $4 \text{ MHz} < f_{\text{Bus}} < 20 \text{ MHz}$ Medium power (will be less than FEI if oscillator range = low) High clock accuracy Medium/High system cost (crystal, resonator or external clock source required) IRG is off. DCO is on.
<b>FLL Bypassed</b>	<b>SCM</b> This mode is mainly provided for quick and reliable system startup. $3 \text{ MHz} < f_{\text{Bus}} < 5 \text{ MHz}$ (default). $3 \text{ MHz} < f_{\text{Bus}} < 20 \text{ MHz}$ (via filter bits). Medium power Poor accuracy. IRG is off. DCO is on and open loop.	<b>FBE</b> $f_{\text{Bus}}$ range $\leq 8 \text{ MHz}$ when crystal or resonator is used. <b>Lowest power</b> Highest clock accuracy Medium/High system cost (Crystal, resonator or external clock source required) IRG is off. DCO is off.

<sup>1</sup> The IRG typically consumes 100  $\mu\text{A}$ . The FLL and DCO typically consumes 0.5 to 2.5 mA, depending upon output frequency. For minimum power consumption and minimum jitter, choose N and R to be as small as possible.

The following sections contain initialization examples for various configurations.

#### NOTE

Hexadecimal values designated by a preceding \$, binary values designated by a preceding %, and decimal values have no preceding character.

Important configuration information is repeated here for reference.

Table 8-11. ICGOUT Frequency Calculation Options

Clock Scheme	$f_{\text{ICGOUT}}^1$	P	Note
SCM — self-clocked mode (FLL bypassed internal)	$f_{\text{ICGDCLK}} / R$	NA	Typical $f_{\text{ICGOUT}} = 8 \text{ MHz}$ immediately after reset
FBE — FLL bypassed external	$f_{\text{ext}} / R$	NA	
FEI — FLL engaged internal	$(f_{\text{IRG}} / 7) * 64 * N / R$	64	Typical $f_{\text{IRG}} = 243 \text{ kHz}$
FEE — FLL engaged external	$f_{\text{ext}} * P * N / R$	Range = 0; P = 64 Range = 1; P = 1	

<sup>1</sup> Ensure that  $f_{\text{ICGDCLK}}$ , which is equal to  $f_{\text{ICGOUT}} * R$ , does not exceed  $f_{\text{ICGDCLKmax}}$ .

Table 8-12. MFD and RFD Decode Table

MFD Value	Multiplication Factor (N)	RFD	Division Factor (R)
000	4	000	÷1
001	6	001	÷2
010	8	010	÷4
011	10	011	÷8
100	12	100	÷16
101	14	101	÷32
110	16	110	÷64
111	18	111	÷128

### 8.6.2 Example #1: External Crystal = 32 kHz, Bus Frequency = 4.19 MHz

In this example, the FLL will be used (in FEE mode) to multiply the external 32 kHz oscillator up to 8.38 MHz to achieve 4.19 MHz bus frequency.

After the MCU is released from reset, the ICG is in self-clocked mode (SCM) and supplies approximately 8 MHz on ICGOUT, which corresponds to a 4 MHz bus frequency ( $f_{\text{Bus}}$ ).

The clock scheme will be FLL engaged, external (FEE). So

$$f_{\text{ICGOUT}} = f_{\text{ext}} * P * N / R; P = 64, f_{\text{ext}} = 32 \text{ kHz} \quad \text{Eqn. 8-1}$$

Solving for N / R gives:

$$N / R = 8.38 \text{ MHz} / (32 \text{ kHz} * 64) = 4; \text{ we can choose } N = 4 \text{ and } R = 1 \quad \text{Eqn. 8-2}$$

The values needed in each register to set up the desired operation are:

**ICGC1 = \$38 (%00111000)**

Bit 7	HGO	0	Configures oscillator for low power
Bit 6	RANGE	0	Configures oscillator for low-frequency range; FLL prescale factor is 64
Bit 5	REFS	1	Oscillator using crystal or resonator is requested
Bits 4:3	CLKS	11	FLL engaged, external reference clock mode
Bit 2	OSCSTEN	0	Oscillator disabled
Bit 1	LOCD	0	Loss-of-clock detection enabled
Bit 0		0	Unimplemented or reserved, always reads zero

**ICGC2 = \$00 (%00000000)**

Bit 7	LOLRE	0	Generates an interrupt request on loss of lock
Bits 6:4	MFD	000	Sets the MFD multiplication factor to 4
Bit 3	LOCRE	0	Generates an interrupt request on loss of clock
Bits 2:0	RFD	000	Sets the RFD division factor to ÷1

**ICGS1 = \$xx**

This is read only except for clearing interrupt flag

**ICGS2 = \$xx**

This is read only; should read DCOS = 1 before performing any time critical tasks

### ICGFLTLU/L = \$xx

Only needed in self-clocked mode; FLT will be adjusted by loop to give 8.38 MHz DCO clock

Bits 15:12 unused 0000

Bits 11:0 FLT No need for user initialization

### ICGTRM = \$xx

Bits 7:0 TRIM Only need to write when trimming internal oscillator; not used when external crystal is clock source

Figure 8-13 shows flow charts for three conditions requiring ICG initialization.

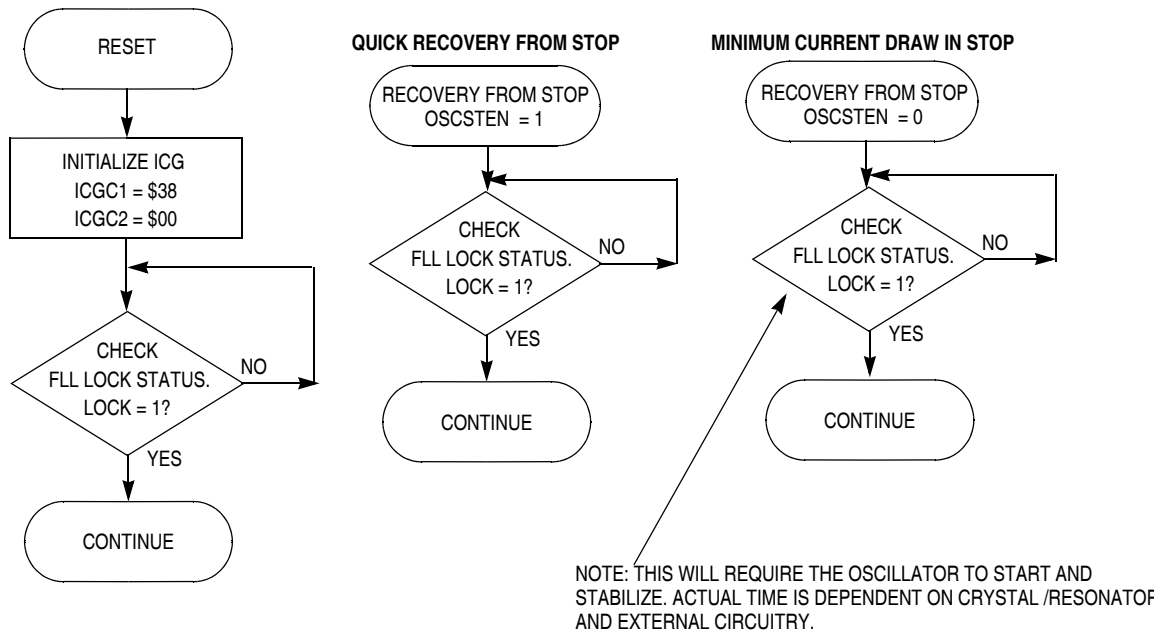


Figure 8-13. ICG Initialization for FEE in Example #1

### 8.6.3 Example #2: External Crystal = 4 MHz, Bus Frequency = 20 MHz

In this example, the FLL will be used (in FEE mode) to multiply the external 4 MHz oscillator up to 40 MHz to achieve 20 MHz bus frequency.

After the MCU is released from reset, the ICG is in self-clocked mode (SCM) and supplies approximately 8 MHz on ICGOUT which corresponds to a 4 MHz bus frequency ( $f_{BUS}$ ).

During reset initialization software, the clock scheme will be set to FLL engaged, external (FEE). So

$$f_{ICGOUT} = f_{ext} * P * N / R; P = 1, f_{ext} = 4.00 \text{ MHz} \quad \text{Eqn. 8-3}$$

Solving for N / R gives:

$$N / R = 40 \text{ MHz} / (4 \text{ MHz} * 1) = 10; \text{ We can choose } N = 10 \text{ and } R = 1 \quad \text{Eqn. 8-4}$$

The values needed in each register to set up the desired operation are:

**ICGC1 = \$78 (%01111000)**

Bit 7	HGO	0	Configures oscillator for low power
Bit 6	RANGE	1	Configures oscillator for high-frequency range; FLL prescale factor is 1
Bit 5	REFS	1	Requests an oscillator
Bits 4:3	CLKS	11	FLL engaged, external reference clock mode
Bit 2	OSCSTEN	0	Disables the oscillator
Bit 1	LOCD	0	Loss-of-clock detection enabled
Bit 0		0	Unimplemented or reserved, always reads zero

**ICGC2 = \$30 (%00110000)**

Bit 7	LOLRE	0	Generates an interrupt request on loss of lock
Bit 6:4	MFD	011	Sets the MFD multiplication factor to 10
Bit 3	LOCRE	0	Generates an interrupt request on loss of clock
Bit 2:0	RFD	000	Sets the RFD division factor to ÷1

**ICGS1 = \$xx**

This is read only except for clearing interrupt flag

**ICGS2 = \$xx**

This is read only. Should read DCOS before performing any time critical tasks

**ICGFLTLU/L = \$xx**

Not used in this example

**ICGTRM**

Not used in this example

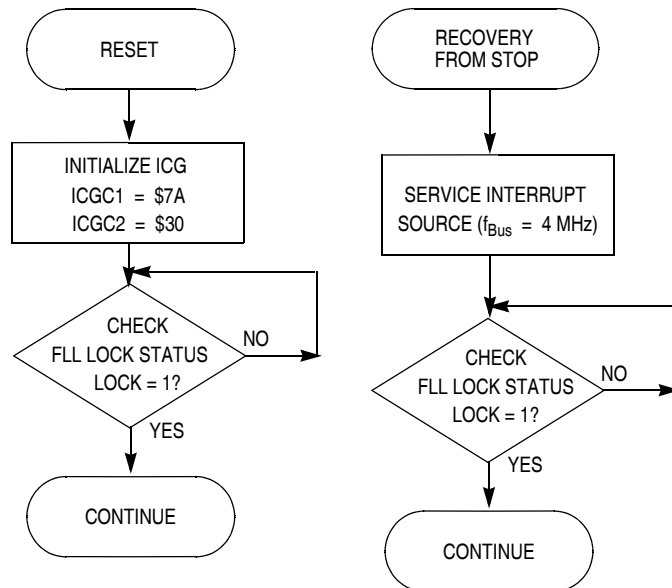


Figure 8-14. ICG Initialization and Stop Recovery for Example #2

### 8.6.4 Example #3: No External Crystal Connection, 5.4 MHz Bus Frequency

In this example, the FLL will be used (in FEI mode) to multiply the internal 243 kHz (approximate) reference clock up to 10.8 MHz to achieve 5.4 MHz bus frequency. This system will also use the trim function to fine tune the frequency based on an external reference signal.

After the MCU is released from reset, the ICG is in self-clocked mode (SCM) and supplies approximately 8 MHz on ICGOUT which corresponds to a 4 MHz bus frequency ( $f_{Bus}$ ).

The clock scheme will be FLL engaged, internal (FEI). So

$$f_{ICGOUT} = (f_{IRG} / 7) * P * N / R; P = 64, f_{IRG} = 243 \text{ kHz} \quad \text{Eqn. 8-5}$$

Solving for N / R gives:

$$N / R = 10.8 \text{ MHz} / (243/7 \text{ kHz} * 64) = 4.86; \text{ We can choose } N = 10 \text{ and } R = 2. \quad \text{Eqn. 8-6}$$

A trim procedure will be required to hone the frequency to exactly 5.4 MHz. An example of the trim procedure is shown in example #4.

The values needed in each register to set up the desired operation are:

**ICGC1 = \$28 (%00101000)**

Bit 7	HGO	0	Configures oscillator for low power
Bit 6	RANGE	0	Configures oscillator for low-frequency range; FLL prescale factor is 64
Bit 5	REFS	1	Oscillator using crystal or resonator requested (bit is really a don't care)
Bits 4:3	CLKS	01	FLL engaged, internal reference clock mode
Bit 2	OSCSTEN	0	Disables the oscillator
Bit 1	LOCD	0	Loss-of-clock enabled

**Internal Clock Generator (S08ICGV4)**

Bit 0                    0     Unimplemented or reserved, always reads zero

**ICGC2 = \$31 (%00110001)**

Bit 7    LOLRE    0     Generates an interrupt request on loss of lock  
 Bit 6:4   MFD     011   Sets the MFD multiplication factor to 10  
 Bit 3    LOCRE    0     Generates an interrupt request on loss of clock  
 Bit 2:0   RFD     001   Sets the RFD division factor to ÷2

**ICGS1 = \$xx**

This is read only except for clearing interrupt flag

**ICGS2 = \$xx**

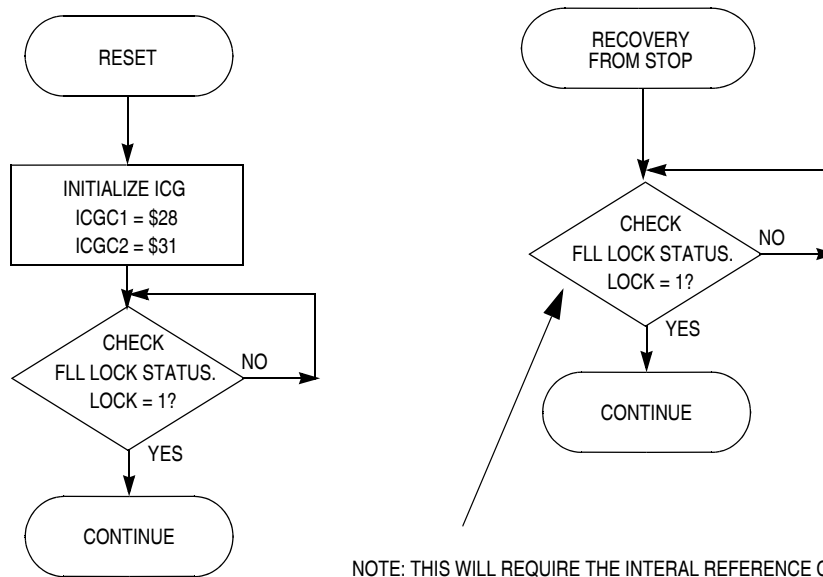
This is read only; good idea to read this before performing time critical operations

**ICGFLTLU/L = \$xx**

Not used in this example

**ICGTRM = \$xx**

Bit 7:0   TRIM                    Only need to write when trimming internal oscillator; done in separate operation (see example #4)



**Figure 8-15. ICG Initialization and Stop Recovery for Example #3**



## 8.6.5 Example #4: Internal Clock Generator Trim

The internally generated clock source is guaranteed to have a period  $\pm 25\%$  of the nominal value. In some cases, this may be sufficient accuracy. For other applications that require a tight frequency tolerance, a trimming procedure is provided that will allow a very accurate source. This section outlines one example of trimming the internal oscillator. Many other possible trimming procedures are valid and can be used.

Initial conditions:

- 1) Clock supplied from ATE has 500  $\mu\text{sec}$  duty period
- 2) ICG configured for internal reference with 4 MHz bus

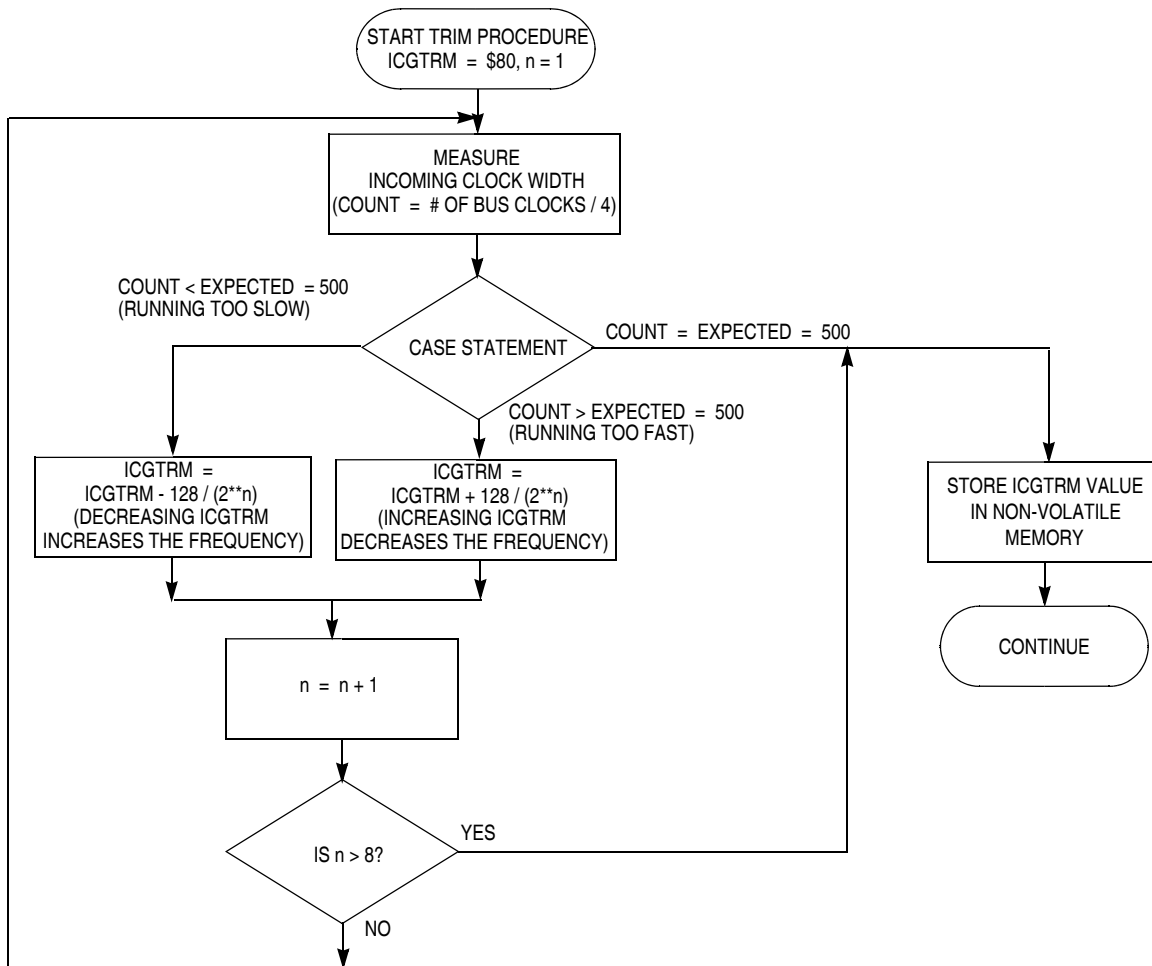


Figure 8-16. Trim Procedure

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 8-16 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 reduction divisor (R) twice the final value. After the trim procedure is complete, the reduction divisor can be restored. This will prevent accidental overshoot of the maximum clock frequency.



# Chapter 9

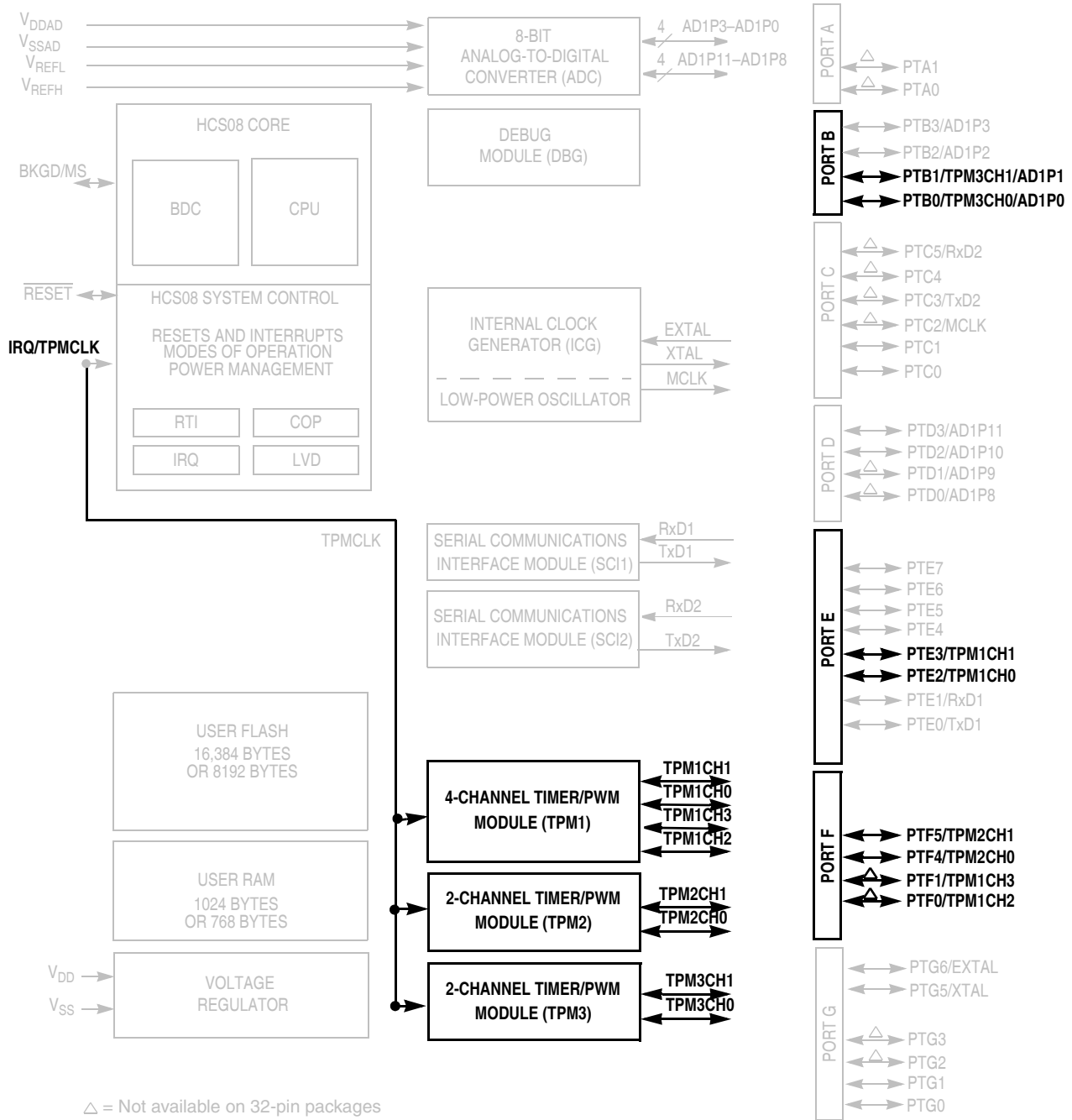
## Timer/PWM (S08TPMV3)

### 9.1 Introduction

The SC9S08MZ16 includes three independent timer/PWM (TPM) modules which support traditional input capture, output compare, or buffered edge-aligned pulse-width modulation (PWM) on each channel. A control bit in each TPM configures all channels in that timer to operate as center-aligned PWM functions. In each 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. This timing system is ideally suited for a wide range of control applications, and the center-aligned PWM capability on the TPM extends the field of applications to motor control in small appliances.

The use of the fixed system clock, XCLK, as the clock source for any of the TPM modules allows the TPM prescaler to run using the oscillator rate divided by two (ICGERCLK/2). This option is only available if the ICG is configured in FEE mode and the proper conditions are met (see [Section 8.4.11, “Fixed Frequency Clock”](#)). In all other ICG modes this selection is redundant because XCLK is the same as BUSCLK.

An external clock source can be connected to the TPMCLK pin. The maximum frequency for TPMCLK is the bus clock frequency divided by 4. All three TPM modules can independently select TPMCLK as the clock source.



**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<sub>DD</sub>. IRQ should not be driven above V<sub>DD</sub>.
4. Pin contains integrated pullup device.
5. PTD3 and PTD2 contain both pullup and pulldown devices.

**Figure 9-1. SC9S08MZ16 Block Diagram Highlighting TPM Blocks and Pins**

## 9.1.1 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 9-1. TPMV2 and TPMV3 Porting Considerations**

Action	TPMV3	TPMV2
<b>Write to TPMxCnTH:L registers<sup>1</sup></b>		
Any write to TPMxCNTH or TPMxCNTL registers	Clears the TPM counter (TPMxCNTH:L) and the prescaler counter.	Clears the TPM counter (TPMxCNTH:L) only.
<b>Read of TPMxCNTH:L registers<sup>1</sup></b>		
In BDM mode, any read of TPMxCNTH:L registers	Returns the value of the TPM counter that is frozen.	If only one byte of the TPMxCNTH:L registers was read before the BDM mode became active, returns the latched value of TPMxCNTH:L from the read buffer (instead of the frozen TPM counter value).
In BDM mode, a write to TPMxSC, TPMxCNTH or TPMxCNTL	Clears this read coherency mechanism.	Does not clear this read coherency mechanism.
<b>Read of TPMxCnVH:L registers<sup>2</sup></b>		
In BDM mode, any read of TPMxCnVH:L registers	Returns the value of the TPMxCnVH:L register.	If only one byte of the TPMxCnVH:L registers was read before the BDM mode became active, returns the latched value of TPMxCNTH:L from the read buffer (instead of the value in the TPMxCnVH:L registers).
In BDM mode, a write to TPMxCnSC	Clears this read coherency mechanism.	Does not clear this read coherency mechanism.
<b>Write to TPMxCnVH:L registers</b>		
In Input Capture mode, writes to TPMxCnVH:L registers <sup>3</sup>	Not allowed.	Allowed.
In Output Compare mode, when (CLKSB:CLKSA not = 0:0), writes to TPMxCnVH:L registers <sup>3</sup>	Update 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.	Always update these registers when their second byte is written.

Table 9-1. TPMV2 and TPMV3 Porting Considerations (continued)

Action	TPMV3	TPMV2
In Edge-Aligned PWM mode when (CLKSB:CLKSA not = 00), writes to TPMxCnVH:L registers	Update the TPMxCnVH:L registers with the value of their write buffer after both bytes were written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). <b>Note:</b> If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from \$FFFE to \$FFFF.	Update after both bytes are written and when the TPM counter changes from TPMxMODH:L to \$0000.
In Center-Aligned PWM mode when (CLKSB:CLKSA not = 00), writes to TPMxCnVH:L registers <sup>4</sup>	Update the TPMxCnVH:L registers with the value of their write buffer after both bytes are written and when the TPM counter changes from (TPMxMODH:L - 1) to (TPMxMODH:L). <b>Note:</b> If the TPM counter is a free-running counter, then this update is made when the TPM counter changes from \$FFFE to \$FFFF.	Update after both bytes are written and when the TPM counter changes from TPMxMODH:L to (TPMxMODH:L - 1).
<b>Center-Aligned PWM</b>		
When TPMxCnVH:L = TPMxMODH:L <sup>5</sup>	Produces 100% duty cycle.	Produces 0% duty cycle.
When TPMxCnVH:L = (TPMxMODH:L - 1) <sup>6</sup>	Produces a near 100% duty cycle.	Produces 0% duty cycle.
TPMxCnVH:L is changed from 0x0000 to a non-zero value <sup>7</sup>	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).
TPMxCnVH:L is changed from a non-zero value to 0x0000 <sup>8</sup>	Finishes the current PWM period using the old duty cycle setting.	Finishes the current PWM period using the new duty cycle setting.
<b>Write to TPMxMODH:L registers in BDM mode</b>		
In BDM mode, a write to TPMxSC register	Clears the write coherency mechanism of TPMxMODH:L registers.	Does not clear the write coherency mechanism.

<sup>1</sup> For more information, refer to [Section 9.3.2, “TPM-Counter Registers \(TPMxCNTH:TPMxCNTL\)”](#) [SE110-TPM case 7]

<sup>2</sup> For more information, refer to [Section 9.3.5, “TPM Channel Value Registers \(TPMxCnVH:TPMxCnVL\)”](#)

<sup>3</sup> For more information, refer to [Section 9.4.2.1, “Input Capture Mode.”](#)

<sup>4</sup> For more information, refer to [Section 9.4.2.4, “Center-Aligned PWM Mode.”](#)

<sup>5</sup> For more information, refer to [Section 9.4.2.4, “Center-Aligned PWM Mode.”](#) [SE110-TPM case 1]

<sup>6</sup> For more information, refer to [Section 9.4.2.4, “Center-Aligned PWM Mode.”](#) [SE110-TPM case 2]

<sup>7</sup> For more information, refer to [Section 9.4.2.4, “Center-Aligned PWM Mode.”](#) [SE110-TPM case 3 and 5]

<sup>8</sup> For more information, refer to [Section 9.4.2.4, “Center-Aligned PWM Mode.”](#) [SE110-TPM case 4]

## 9.1.2 Migrating from TPMV1

In addition to [Section 9.1.1, “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 (TPMxCnV) when the timer is not in input capture mode for TPMV2, not TPMV3.
- In edge- or center- aligned modes, the Channel Value register (TPMxCnV) 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 TPMxSC before TPMxCnV as a write to TPMxSC resets the coherency mechanism on the TPMxCnV registers.

**Table 9-2. Migrating to TPMV3 Considerations**

When...	Action / Best Practice
Writing to the Channel Value Register (TPMxCnV) register...	Timer must be in Input Capture mode.
Updating the Channel Value Register (TPMxCnV) 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 (TPMxCnV) register...	Write to TPMxSC.
Configuring the TPM modules...	Write first to TPMxSC and then to TPMxCnV register.

## 9.1.3 Features

The timer system in the SC9S08MZ16 includes a 4-channel TPM1, a separate 2-channel TPM2, and a separate 2-channel TPM3. Timer system features include:

- A total of up to eight 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
- Each TPM may be configured for buffered, center-aligned pulse-width modulation (CPWM) on all channels
- Clock source to prescaler for each 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 use by TPM1, TPM2, and/or TPM3
- 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 each TPM module



## 9.1.4 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.

### 9.1.5 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMxCHn (timer channel n) where n is the channel number (1-8). The TPM shares its I/O pins with general purpose I/O port pins (refer to I/O pin descriptions in full-chip specification for the specific chip implementation).

Figure 9-2 shows the TPM structure. The central component of the TPM is the 16-bit counter that can operate as a free-running counter or a modulo up/down counter. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMxMODH:TPMxMODL, control the modulo value of the counter (the values 0x0000 or 0xFFFF effectively make the counter free running). Software can read the counter value at any time without affecting the counting sequence. Any write to either half of the TPMxCNT counter resets the counter, regardless of the data value written.

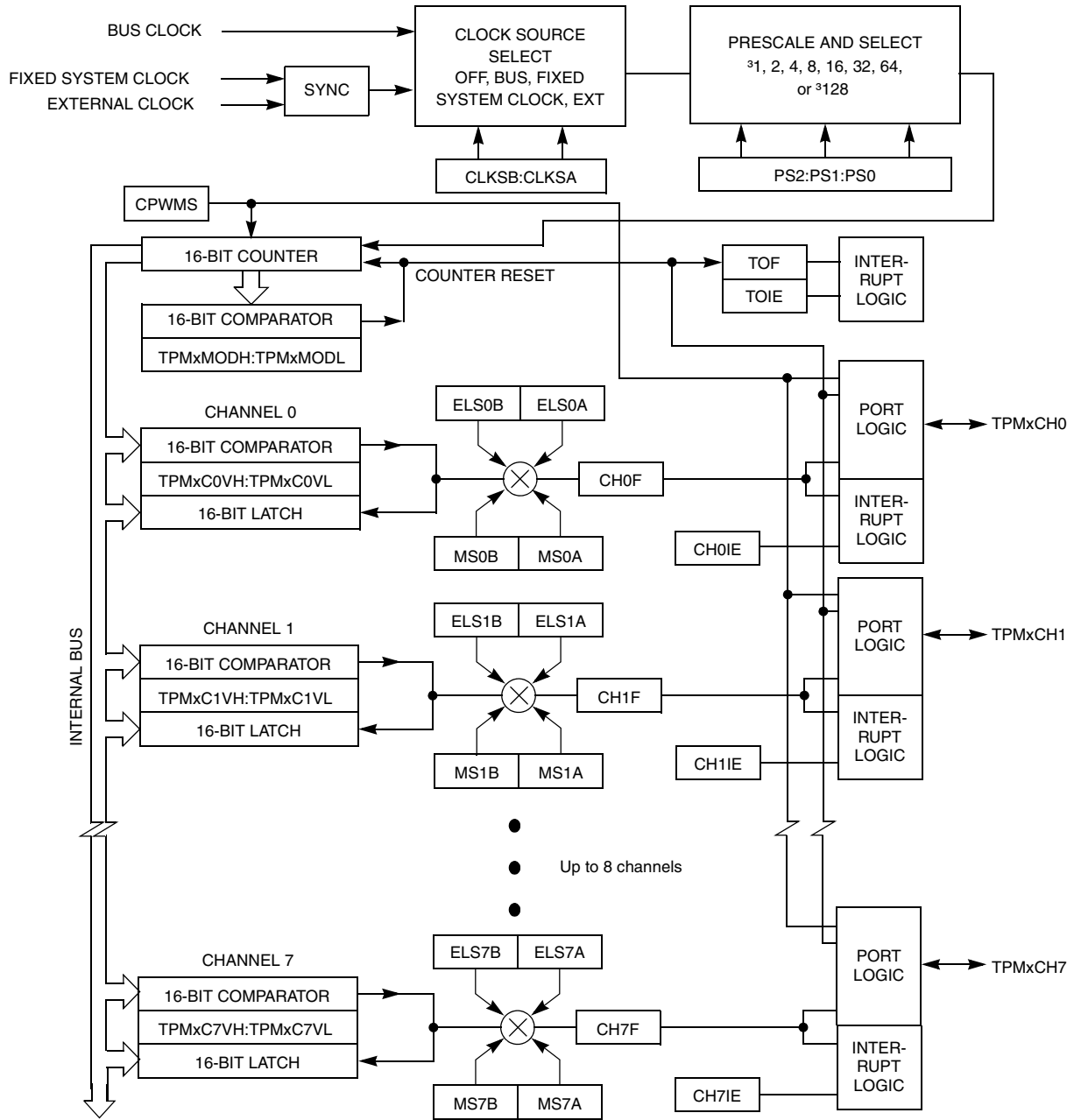


Figure 9-2. TPM Block Diagram

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.

## 9.2 Signal Description

Table 9-3 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 9-3. Signal Properties**

Name	Function
EXTCLK <sup>1</sup>	External clock source which may be selected to drive the TPM counter.
TPMxCHn <sup>2</sup>	I/O pin associated with TPM channel n

<sup>1</sup> 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.

<sup>2</sup> 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.

### 9.2.1 Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 9-3 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.

### 9.2.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow the user to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock which drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer; the frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock, to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin will not be usable for channel I/O function when selected as the external clock source. It is the user's responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can still be used in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

### 9.2.1.2 TPMxCHn — TPM Channel n I/O Pin(s)

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled whenever a port pin is acting as an input.

The TPM channel does not control the I/O pin when (ELSnB:ELSnA = 0:0) or when (CLKSB:CLKSA = 0:0) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the TPMxCHn pins are all controlled by the TPM system. When CPWMS=0, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

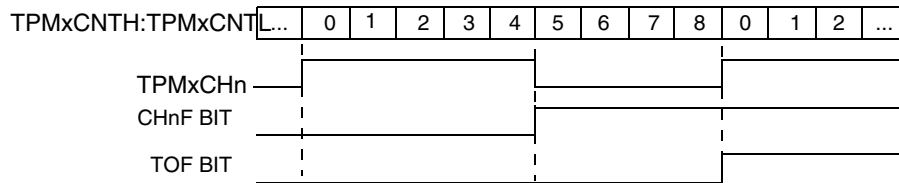
When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPMxCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges will trigger input-capture events. A synchronizer based on the bus clock is used to synchronize input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA = 0:1 and ELSnB:ELSnA not = 0:0), the associated data direction control is overridden, the TPMxCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPMxCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event—then the pin is toggled.

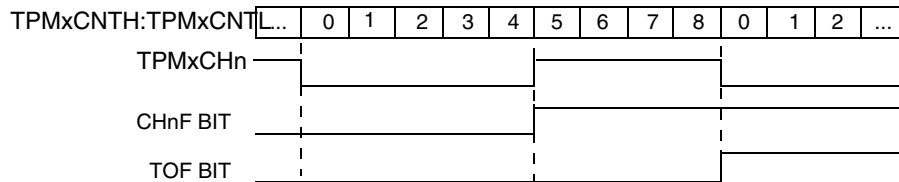
When a channel is configured for edge-aligned PWM (CPWMS=0, MSnB=1 and ELSnB:ELSnA not = 0:0), the data direction is overridden, the TPMxCHn pin is forced to be an output controlled by the TPM, and ELSnA controls the polarity of the PWM output signal on the pin. When ELSnB:ELSnA=1:0, the TPMxCHn pin is forced high at the start of each new period (TPMxCNT=0x0000), and the pin is forced low when the channel value register matches the timer counter. When ELSnA=1, the TPMxCHn pin is forced low at the start of each new period (TPMxCNT=0x0000), and the pin is forced high when the channel value register matches the timer counter.

TPMxMODH:TPMxMODL = 0x0008  
 TPMxCnVH:TPMxCnVL = 0x0005



**Figure 9-3. High-True Pulse of an Edge-Aligned PWM**

TPMxMODH:TPMxMODL = 0x0008  
 TPMxCnVH:TPMxCnVL = 0x0005



**Figure 9-4. Low-True Pulse of an Edge-Aligned PWM**

When the TPM is configured for center-aligned PWM (and ELSnB:ELSnA not = 0:0), the data direction for all channels in this TPM are overridden, the TPMxCHn pins are forced to be outputs controlled by the TPM, and the ELSnA bits control the polarity of each TPMxCHn output. If ELSnB:ELSnA=1:0, the corresponding TPMxCHn pin is cleared when the timer counter is counting up, and the channel value register matches the timer counter; the TPMxCHn pin is set when the timer counter is counting down, and the channel value register matches the timer counter. If ELSnA=1, the corresponding TPMxCHn pin is set when the timer counter is counting up and the channel value register matches the timer counter; the TPMxCHn pin is cleared when the timer counter is counting down and the channel value register matches the timer counter.

TPMxMODH:TPMxMODL = 0x0008  
 TPMxCnVH:TPMxCnVL = 0x0005

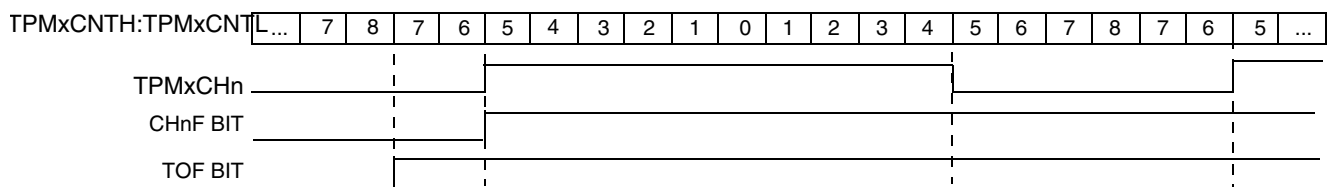


Figure 9-5. High-True Pulse of a Center-Aligned PWM

TPMxMODH:TPMxMODL = 0x0008  
 TPMxCnVH:TPMxCnVL = 0x0005

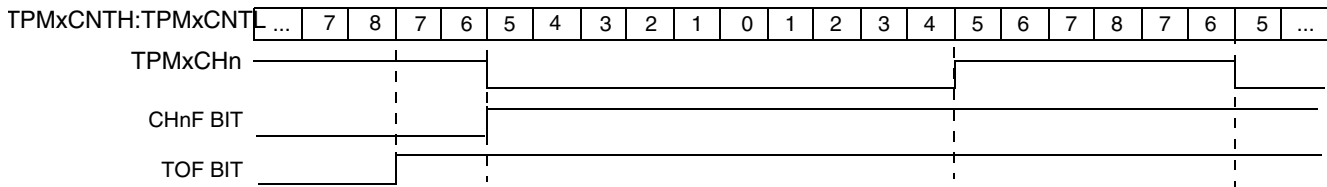


Figure 9-6. Low-True Pulse of a Center-Aligned PWM

### 9.3 Register Definition

This section consists of register descriptions in address order. A typical MCU system may contain multiple TPMs, and each TPM may have one to eight channels, so register names include placeholder characters to identify which TPM and which channel is being referenced. For example, TPMxCnSC refers to timer (TPM) x, channel n. TPM1C2SC would be the status and control register for channel 2 of timer 1.

#### 9.3.1 TPM Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits used to configure the interrupt enable, TPM configuration, clock source, and prescale factor. These controls relate to all channels within this timer module.

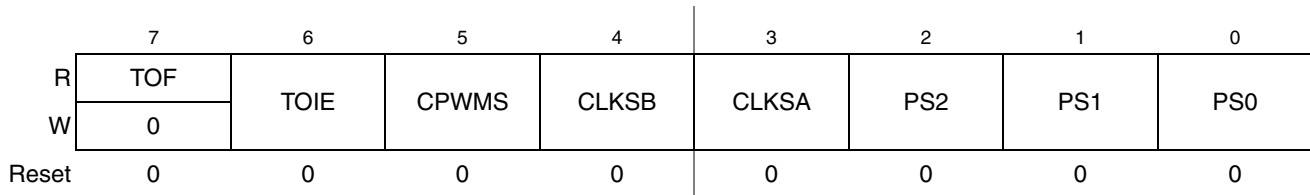


Figure 9-7. TPM Status and Control Register (TPMxSC)

Table 9-4. TPMxSC Field Descriptions

Field	Description
7 TOF	Timer overflow flag. This read/write flag is set when the TPM counter resets to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. Clear TOF by reading the TPM status and control register when TOF is set and then writing a logic 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. This is done so a TOF interrupt request cannot be lost during the clearing sequence for a previous TOF. Reset clears TOF. Writing a logic 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed
6 TOIE	Timer overflow interrupt enable. This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals one. Reset clears TOIE. 0 TOF interrupts inhibited (use for software polling) 1 TOF interrupts enabled
5 CPWMS	Center-aligned PWM select. When present, this read/write bit selects CPWM operating mode. By default, the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up/down counting mode for CPWM functions. Reset clears CPWMS. 0 All channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register. 1 All channels operate in center-aligned PWM mode.



**Table 9-4. TPMxSC Field Descriptions (continued)**

Field	Description
4–3 CLKS[B:A]	Clock source selects. As shown in <a href="#">Table 9-5</a> , 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.
2–0 PS[2:0]	Prescale factor select. This 3-bit field selects one of 8 division factors for the TPM clock input as shown in <a href="#">Table 9-6</a> . This prescaler is located after any clock source synchronization or clock source selection so it affects the clock source selected to drive the TPM system. The new prescale factor will affect the clock source on the next system clock cycle after the new value is updated into the register bits.

**Table 9-5. 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

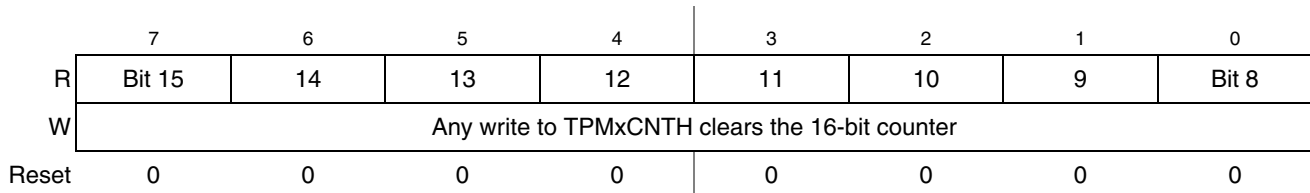
**Table 9-6. Prescale Factor Selection**

PS2:PS1:PS0	TPM Clock Source Divided-by
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

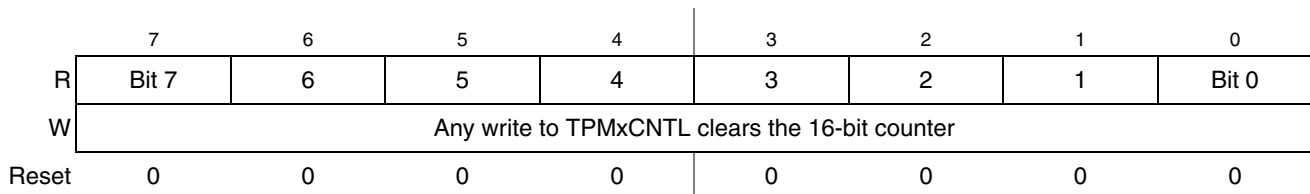
### 9.3.2 TPM-Counter Registers (TPMxCNTH:TPMxCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMxCNTH or TPMxCNTL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This allows coherent 16-bit reads in either big-endian or little-endian order which makes this more friendly to various compiler implementations. The coherency mechanism is automatically restarted by an MCU reset or any write to the timer status/control register (TPMxSC).

Reset clears the TPM counter registers. Writing any value to TPMxCNTH or TPMxCNTL also clears the TPM counter (TPMxCNTH:TPMxCNTL) and resets the coherency mechanism, regardless of the data involved in the write.



**Figure 9-8. TPM Counter Register High (TPMxCNTH)**



**Figure 9-9. TPM Counter Register Low (TPMxCNTL)**

When BDM is active, the timer counter is frozen (this is the value that will be read by user); the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active, even if one or both counter halves are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution.

In BDM mode, writing any value to TPMxSC, TPMxCNTH or TPMxCNTL registers resets the read coherency mechanism of the TPMxCNTH:L registers, regardless of the data involved in the write.

### 9.3.3 TPM Counter Modulo Registers (TPMxMODH:TPMxMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock, and the overflow flag (TOF) becomes set. Writing to TPMxMODH or TPMxMODL inhibits the TOF bit and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000 which results in a free running timer counter (modulo disabled).

Writing to either byte (TPMxMODH or TPMxMODL) latches the value into a buffer and the registers are updated with the value of their write buffer according to the value of CLKSb:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), then the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), then the registers are updated after both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF

The latching mechanism may be manually reset by writing to the TPMxSC address (whether BDM is active or not).

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the modulo register are written while BDM is active. Any write to the modulo registers bypasses the buffer latches and directly writes to the modulo register while BDM is active.

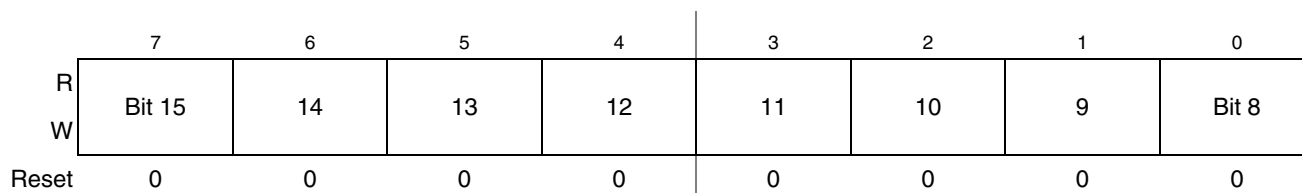


Figure 9-10. TPM Counter Modulo Register High (TPMxMODH)

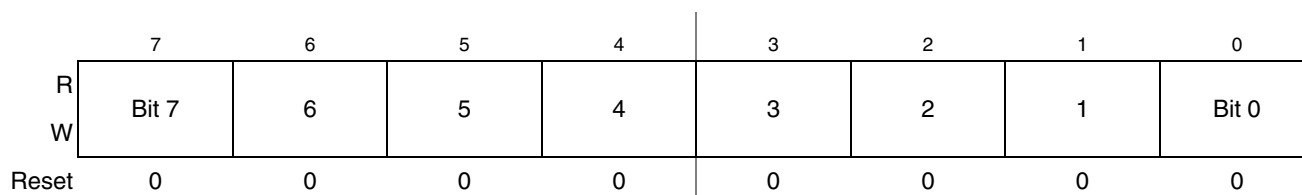


Figure 9-11. TPM Counter Modulo Register Low (TPMxMODL)

Reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

### 9.3.4 TPM Channel n Status and Control Register (TPMxCnSC)

TPMxCnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

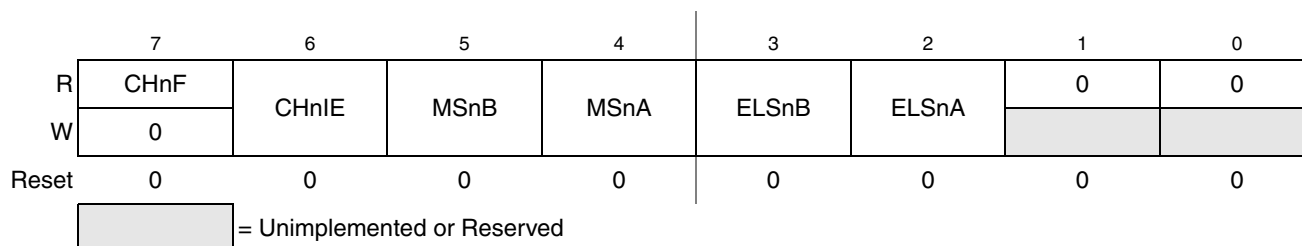


Figure 9-12. TPM Channel n Status and Control Register (TPMxCnSC)

**Table 9-7. TPMxCnSC Field Descriptions**

Field	Description
7 CHnF	<p>Channel n flag. When channel n is an input-capture channel, this read/write bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned/center-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. When channel n is an edge-aligned/center-aligned PWM channel and the duty cycle is set to 0% or 100%, CHnF will not be set even when the value in the TPM counter registers matches the value in the TPM channel n value registers. A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPMxCnSC while CHnF is set and then writing a logic 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF remains set after the clear sequence completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost due to clearing a previous CHnF.</p> <p>Reset clears the CHnF bit. Writing a logic 1 to CHnF has no effect.</p> <p>0 No input capture or output compare event occurred on channel n 1 Input capture or output compare event on channel n</p>
6 CHnIE	<p>Channel n interrupt enable. This read/write bit enables interrupts from channel n. Reset clears CHnIE.</p> <p>0 Channel n interrupt requests disabled (use for software polling) 1 Channel n interrupt requests enabled</p>
5 MSnB	<p>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 <a href="#">Table 9-8</a>.</p>
4 MSnA	<p>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 <a href="#">Table 9-8</a> for a summary of channel mode and setup controls.</p> <p><b>Note:</b> 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.</p>
3–2 ELSnB ELSnA	<p>Edge/level select bits. Depending upon the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in <a href="#">Table 9-8</a>, 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.</p>

**Table 9-8. 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

Table 9-8. Mode, Edge, and Level Selection (continued)

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
0	00	01	Input capture	Capture on rising edge only
		10		Capture on falling edge only
		11		Capture on rising or falling edge
	01	00	Output compare	Software compare only
		01		Toggle output on 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)

### 9.3.5 TPM Channel Value Registers (TPMxCnVH:TPMxCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel registers are cleared by reset.

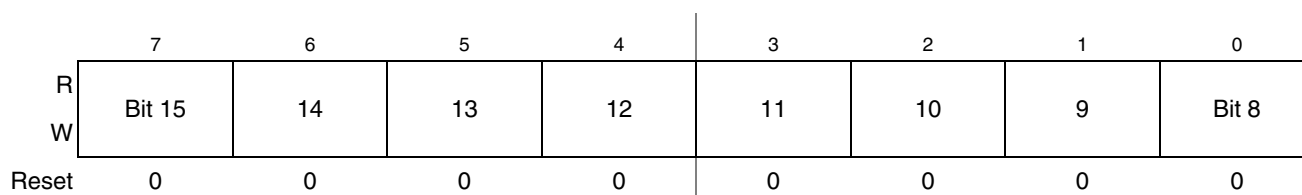


Figure 9-13. TPM Channel Value Register High (TPMxCnVH)

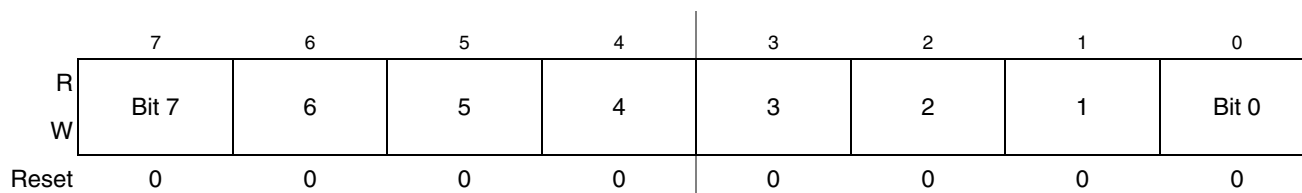


Figure 9-14. TPM Channel Value Register Low (TPMxCnVL)

In input capture mode, reading either byte (TPMxCnVH or TPMxCnVL) latches the contents of both bytes into a buffer where they remain latched until the other half is read. This latching mechanism also resets (becomes unlatched) when the TPMxCnSC register is written (whether BDM mode is active or not). Any write to the channel registers will be ignored during the input capture mode.

When BDM is active, the coherency mechanism is frozen (unless reset by writing to TPMxCnSC register) such that the buffer latches remain in the state they were in when the BDM became active, even if one or both halves of the channel register are read while BDM is active. This assures that if the user was in the middle of reading a 16-bit register when BDM became active, it will read the appropriate value from the other half of the 16-bit value after returning to normal execution. The value read from the TPMxCnVH and TPMxCnVL registers in BDM mode is the value of these registers and not the value of their read buffer.

In output compare or PWM modes, writing to either byte (TPMxCnVH or TPMxCnVL) latches the value into a buffer. After both bytes are written, they are transferred as a coherent 16-bit value into the timer-channel registers according to the value of CLKSB:CLKSA bits and the selected mode, so:

- If (CLKSB:CLKSA = 0:0), then the registers are updated when the second byte is written.
- If (CLKSB:CLKSA not = 0:0 and in output compare mode) then the registers are updated after the second byte is written and on the next change of the TPM counter (end of the prescaler counting).
- If (CLKSB:CLKSA not = 0:0 and in EPWM or CPWM modes), then the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFFE to 0xFFFF.

The latching mechanism may be manually reset by writing to the TPMxCnSC register (whether BDM mode is active or not). This latching mechanism allows coherent 16-bit writes in either big-endian or little-endian order which is friendly to various compiler implementations.

When BDM is active, the coherency mechanism is frozen such that the buffer latches remain in the state they were in when the BDM became active even if one or both halves of the channel register are written while BDM is active. Any write to the channel registers bypasses the buffer latches and directly write to the channel register while BDM is active. The values written to the channel register while BDM is active are used for PWM & output compare operation once normal execution resumes. Writes to the channel registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

## 9.4 Functional Description

All TPM functions are associated with a central 16-bit counter which allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

The following sections describe the main counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and

interrupt activity depend upon the operating mode, these topics will be covered in the associated mode explanation sections.

## 9.4.1 Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock source, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

### 9.4.1.1 Counter Clock Source

The 2-bit field, CLKSB:CLKSA, in the timer status and control register (TPMxSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See [Table 9-5](#). After any MCU reset, CLKSB:CLKSA=0:0 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKSB:CLKSA field) does not affect the values in the counter or other timer registers.

**Table 9-9. 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).

### 9.4.1.2 Counter Overflow and Modulo Reset

An interrupt flag and enable are associated with the 16-bit main counter. The flag (TOF) is a software-accessible indication that the timer counter has overflowed. The enable signal selects between software polling (TOIE=0) where no hardware interrupt is generated, or interrupt-driven operation (TOIE=1) where a static hardware interrupt is generated whenever the TOF flag is equal to one.

The conditions causing TOF to become set depend on whether the TPM is configured for center-aligned PWM (CPWMS=1). In the simplest mode, there is no modulus limit and the TPM is not in CPWMS=1 mode. In this case, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the TPM is in center-aligned PWM mode (CPWMS=1), the TOF flag gets set as the counter changes direction at the end of the count value set in the modulus register (that is, at the transition from the value set in the modulus register to the next lower count value). This corresponds to the end of a PWM period (the 0x0000 count value corresponds to the center of a period).

### 9.4.1.3 Counting Modes

The main timer counter has two counting modes. When center-aligned PWM is selected (CPWMS=1), the counter operates in up/down counting mode. Otherwise, the counter operates as a simple up counter. As an up counter, the timer counter counts from 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMxMODH:TPMxMODL.

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. Both 0x0000 and the terminal count value are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) becomes set at the end of the terminal-count period (as the count changes to the next lower count value).

### 9.4.1.4 Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to either half of TPMxCNTH or TPMxCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only half of the counter was read before resetting the count.

## 9.4.2 Channel Mode Selection

Provided CPWMS=0, the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

### 9.4.2.1 Input Capture Mode

With the input-capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input-capture channel, the TPM latches the contents of the TPM counter into the channel-value registers (TPMxCnVH:TPMxCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.



In input capture mode, the TPMxCnVH and TPMxCnVL registers are read only.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An input capture event sets a flag bit (CHnF) which may optionally generate a CPU interrupt request.

While in BDM, the input capture function works as configured by the user. When an external event occurs, the TPM latches the contents of the TPM counter (which is frozen because of the BDM mode) into the channel value registers and sets the flag bit.

### 9.4.2.2 Output Compare Mode

With the output-compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel-value registers of an output-compare channel, the TPM can set, clear, or toggle the channel pin.

In output compare mode, values are transferred to the corresponding timer channel registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKSb:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) which may optionally generate a CPU-interrupt request.

### 9.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS=0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the value of the modulus register (TPMxMODH:TPMxMODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 9-15). 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.

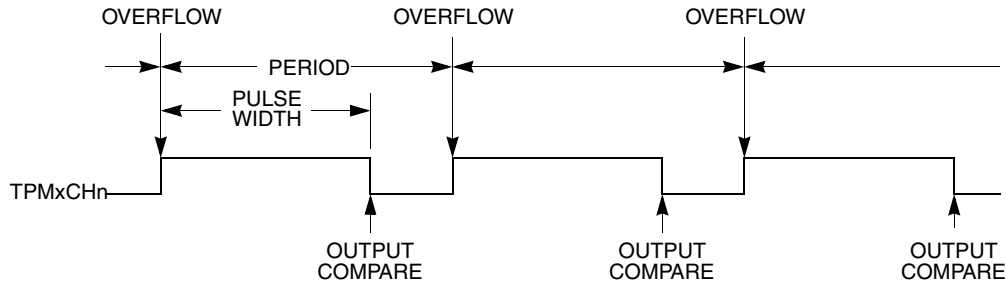


Figure 9-15. PWM Period and Pulse Width (ELSnA=0)

When the channel value register is set to 0x0000, the duty cycle is 0%. 100% duty cycle can be achieved by setting the timer-channel register (TPMxCnVH:TPMxCnVL) to a value greater than the modulus setting. This implies that the modulus setting must be less than 0xFFFF in order to get 100% duty cycle.

Because the TPM may be used in an 8-bit MCU, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers TPMxCnVH and TPMxCnVL, actually write to buffer registers. In edge-aligned PWM mode, values are transferred to the corresponding timer-channel registers according to the value of CLKSb:CLKSA bits, so:

- If (CLKSb:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSb:CLKSA not = 0:0), the registers are updated after the both bytes were written, and the TPM counter changes from (TPMxMODH:TPMxMODL - 1) to (TPMxMODH:TPMxMODL). If the TPM counter is a free-running counter then the update is made when the TPM counter changes from 0xFFFE to 0xFFFF.

#### 9.4.2.4 Center-Aligned PWM Mode

This type of PWM output uses the up/down counting mode of the timer counter (CPWMS=1). The output compare value in TPMxCnVH:TPMxCnVL determines the pulse width (duty cycle) of the PWM signal while the period is determined by the value in TPMxMODH:TPMxMODL. TPMxMODH:TPMxMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

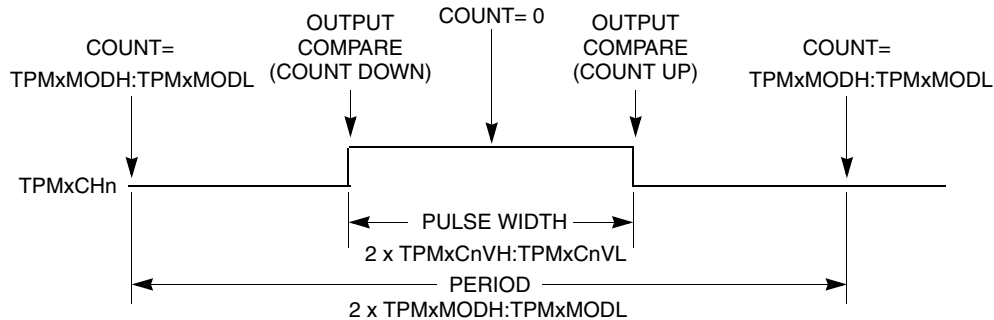
$$\text{pulse width} = 2 \times (\text{TPMxCnVH:TPMxCnVL})$$

$$\text{period} = 2 \times (\text{TPMxMODH:TPMxMODL}); \text{TPMxMODH:TPMxMODL}=0\text{x}0001\text{-}0\text{x}7\text{FFF}$$

If the channel-value register TPMxCnVH:TPMxCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMxCnVH:TPMxCnVL is a positive value (bit 15 clear) and is greater than the (non-zero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if you do not need to generate 100% duty cycle). This is not a significant limitation. The resulting period would be much longer than required for normal applications.

TPMxMODH:TPMxMODL=0x0000 is a special case that should not be used with center-aligned PWM mode. When CPWMS=0, this case corresponds to the counter running free from 0x0000 through 0xFFFF, but when CPWMS=1 the counter needs a valid match to the modulus register somewhere other than at 0x0000 in order to change directions from up-counting to down-counting.

The output compare value in the TPM channel registers (times 2) determines the pulse width (duty cycle) of the CPWM signal (Figure 9-16). If  $ELSnA=0$ , a compare occurred while counting up forces the CPWM output signal low and a compare occurred while counting down forces the output high. The counter counts up until it reaches the modulo setting in  $TPMxMODH:TPMxMODL$ , then counts down until it reaches zero. This sets the period equal to two times  $TPMxMODH:TPMxMODL$ .



**Figure 9-16. CPWM Period and Pulse Width ( $ELSnA=0$ )**

Center-aligned PWM outputs typically produce less noise than edge-aligned PWMs because fewer I/O pin transitions are lined up at the same system clock edge. This type of PWM is also required for some types of motor drives.

Input capture, output compare, and edge-aligned PWM functions do not make sense when the counter is operating in up/down counting mode so this implies that all active channels within a TPM must be used in CPWM mode when  $CPWMS=1$ .

The TPM may be used in an 8-bit MCU. The settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to any of the registers  $TPMxMODH$ ,  $TPMxMODL$ ,  $TPMxCnVH$ , and  $TPMxCnVL$ , actually write to buffer registers.

In center-aligned PWM mode, the  $TPMxCnVH:L$  registers are updated with the value of their write buffer according to the value of  $CLKSB:CLKSA$  bits, so:

- If ( $CLKSB:CLKSA = 0:0$ ), the registers are updated when the second byte is written
- If ( $CLKSB:CLKSA \text{ not } = 0:0$ ), the registers are updated after the both bytes were written, and the TPM counter changes from ( $TPMxMODH:TPMxMODL - 1$ ) to ( $TPMxMODH:TPMxMODL$ ). If the TPM counter is a free-running counter, the update is made when the TPM counter changes from  $0xFFFFE$  to  $0xFFFF$ .

When  $TPMxCNTH:TPMxCNTL=TPMxMODH:TPMxMODL$ , the TPM can optionally generate a TOF interrupt (at the end of this count).

Writing to  $TPMxSC$  cancels any values written to  $TPMxMODH$  and/or  $TPMxMODL$  and resets the coherency mechanism for the modulo registers. Writing to  $TPMxCnSC$  cancels any values written to the channel value registers and resets the coherency mechanism for  $TPMxCnVH:TPMxCnVL$ .

## 9.5 Reset Overview

### 9.5.1 General

The TPM is reset whenever any MCU reset occurs.

### 9.5.2 Description of Reset Operation

Reset clears the TPMxSC register which disables clocks to the TPM and disables timer overflow interrupts (TOIE=0). CPWMS, MSnB, MSnA, ELSnB, and ELSnA are all cleared which configures all TPM channels for input-capture operation with the associated pins disconnected from I/O pin logic (so all MCU pins related to the TPM revert to general purpose I/O pins).

## 9.6 Interrupts

### 9.6.1 General

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on each channel's mode of operation. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register.

All TPM interrupts are listed in [Table 9-10](#) 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 9-10. 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.

### 9.6.2 Description of Interrupt Operation

For each interrupt source in the TPM, a flag bit is set upon recognition of the interrupt condition such as timer overflow, channel-input capture, or output-compare events. This flag may be read (polled) by software to determine that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set

to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt will generate whenever the associated interrupt flag equals one. The user's software must perform a sequence of steps to clear the interrupt flag before returning from the interrupt-service routine.

TPM interrupt flags are cleared by a two-step process including a read of the flag bit while it is set (1) followed by a write of zero (0) to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

### 9.6.2.1 Timer Overflow Interrupt (TOF) Description

The meaning and details of operation for TOF interrupts varies slightly depending upon the mode of operation of the TPM system (general purpose timing functions versus center-aligned PWM operation). The flag is cleared by the two step sequence described above.

#### 9.6.2.1.1 Normal Case

Normally TOF is set when the timer counter changes from 0xFFFF to 0x0000. When the TPM is not configured for center-aligned PWM (CPWMS=0), TOF gets set when the timer counter changes from the terminal count (the value in the modulo register) to 0x0000. This case corresponds to the normal meaning of counter overflow.

#### 9.6.2.1.2 Center-Aligned PWM Case

When CPWMS=1, TOF gets set when the timer counter changes direction from up-counting to down-counting at the end of the terminal count (the value in the modulo register). In this case the TOF corresponds to the end of a PWM period.

### 9.6.2.2 Channel Event Interrupt Description

The meaning of channel interrupts depends on the channel's current mode (input-capture, output-compare, edge-aligned PWM, or center-aligned PWM).

#### 9.6.2.2.1 Input Capture Events

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select no edge (off), rising edges, falling edges or any edge as the edge which triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the two-step sequence described in [Section 9.6.2, "Description of Interrupt Operation."](#)

#### 9.6.2.2.2 Output Compare Events

When a channel is configured as an output compare channel, the interrupt flag is set each time the main timer counter matches the 16-bit value in the channel value register. The flag is cleared by the two-step sequence described [Section 9.6.2, "Description of Interrupt Operation."](#)

### 9.6.2.2.3 PWM End-of-Duty-Cycle Events

For channels configured for PWM operation there are two possibilities. When the channel is configured for edge-aligned PWM, the channel flag gets set when the timer counter matches the channel value register which marks the end of the active duty cycle period. When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle period which are the times when the timer counter matches the channel value register. The flag is cleared by the two-step sequence described [Section 9.6.2, “Description of Interrupt Operation.”](#)

# Chapter 10

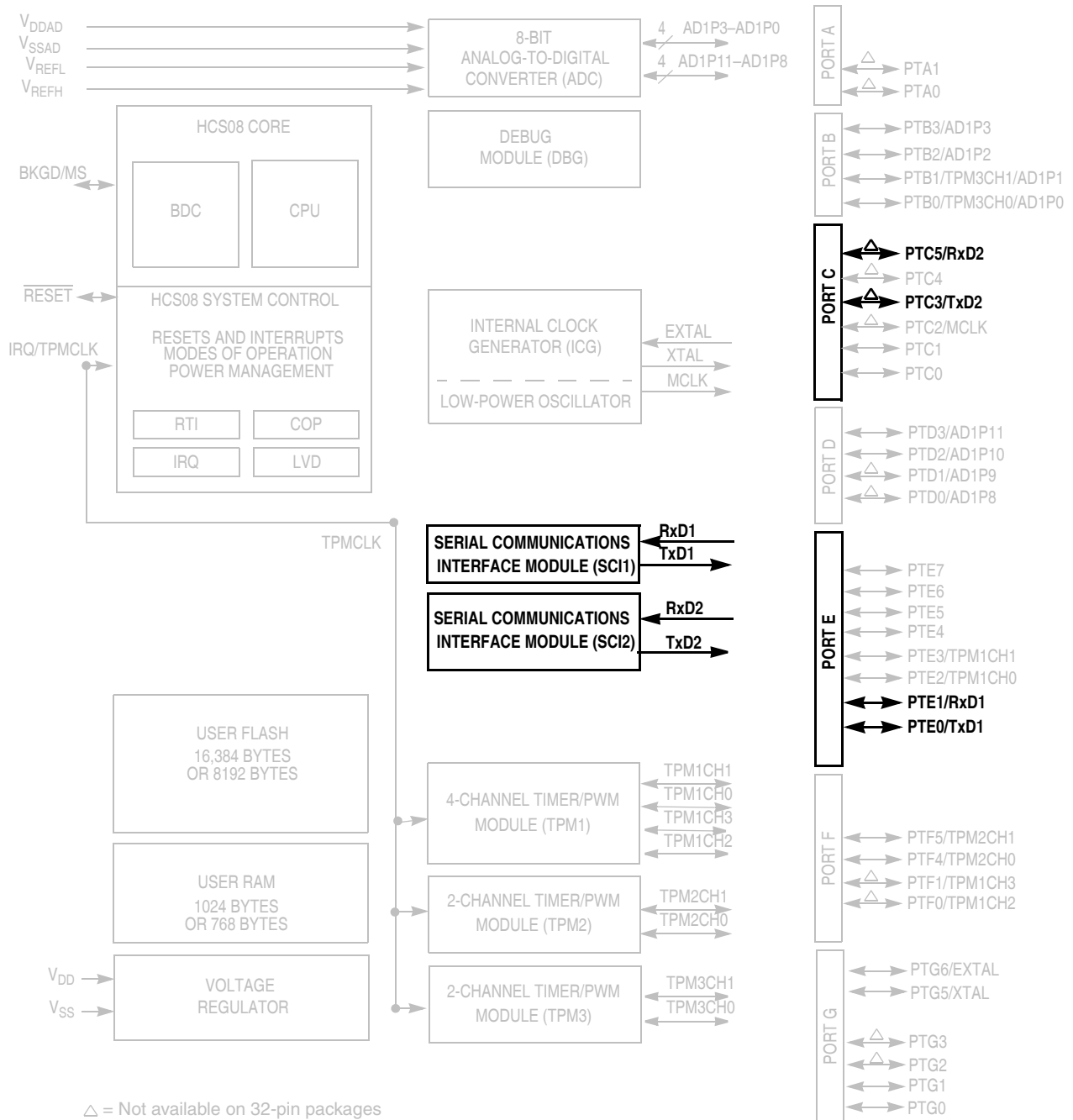
## Serial Communications Interface (S08SCIV4)

### 10.1 Introduction

The SC9S08MZ16 includes 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.



**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<sub>DD</sub>. IRQ should not be driven above V<sub>DD</sub>.
4. Pin contains integrated pullup device.
5. PTD3 and PTD2 contain both pullup and pulldown devices.

**Figure 10-1. SC9S08MZ16 Block Diagram Highlighting SCI Blocks and Pins**



## 10.1.1 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

## 10.1.2 Modes of Operation

See [Section 10.3, “Functional Description,”](#) For details concerning SCI operation in these modes:

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

## 10.1.3 Block Diagram

[Figure 10-2](#) shows the transmitter portion of the SCI.

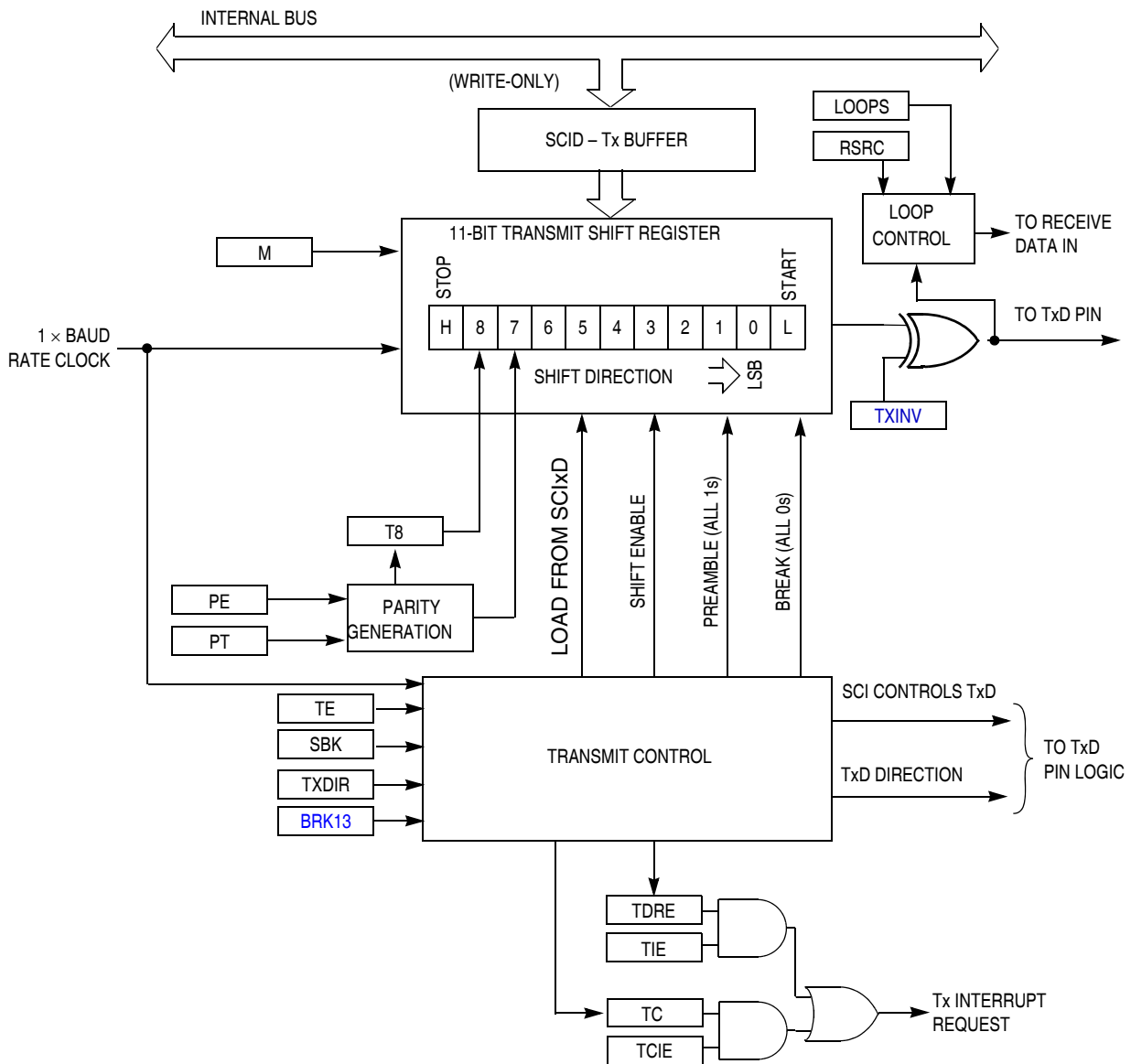


Figure 10-2. SCI Transmitter Block Diagram

Figure 10-3 shows the receiver portion of the SCI.

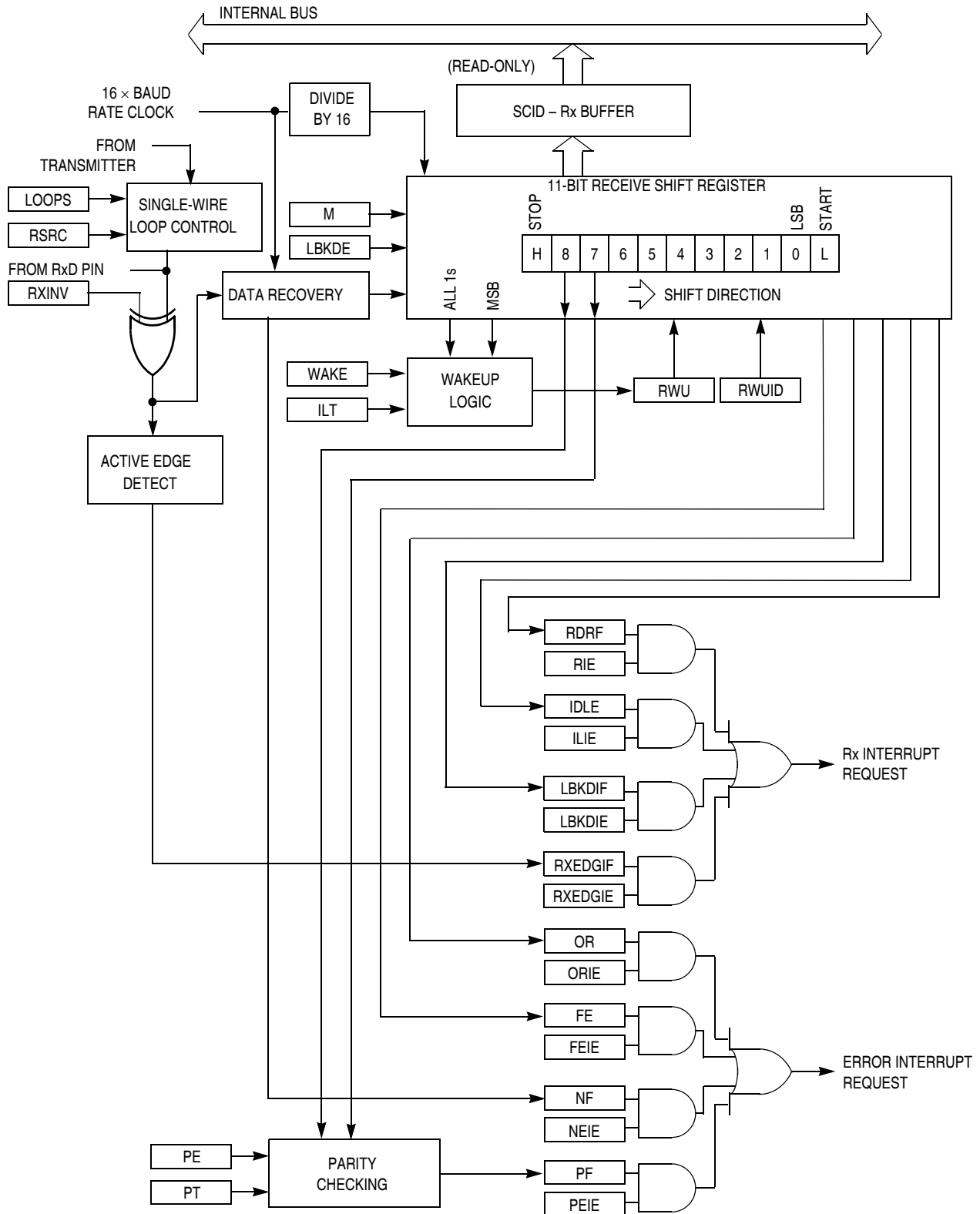


Figure 10-3. SCI Receiver Block Diagram

## 10.2 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.

### 10.2.1 SCI Baud Rate Registers (SCIxBDH, SCIxBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIxBDH to buffer the high half of the new value and then write to SCIxBDL. The working value in SCIxBDH does not change until SCIxBDL is written.

SCIxBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIxC2 are written to 1).

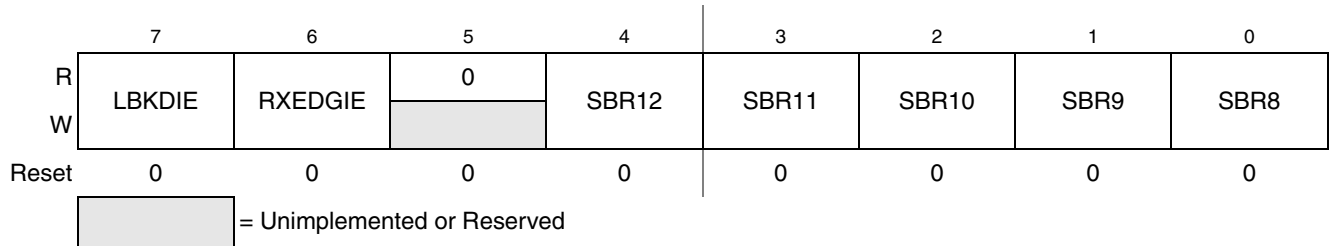


Figure 10-4. SCI Baud Rate Register (SCIxBDH)

Table 10-1. SCIxBDH Field Descriptions

Field	Description
7 LBKDIE	<b>LIN Break Detect Interrupt Enable (for LBKDIF)</b> 0 Hardware interrupts from LBKDIF disabled (use polling). 1 Hardware interrupt requested when LBKDIF flag is 1.
6 RXEDGIE	<b>RxD Input Active Edge Interrupt Enable (for RXEDGIF)</b> 0 Hardware interrupts from RXEDGIF disabled (use polling). 1 Hardware interrupt requested when RXEDGIF flag is 1.
4:0 SBR[12:8]	<b>Baud Rate Modulo Divisor</b> — 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 <a href="#">Table 10-2</a> .

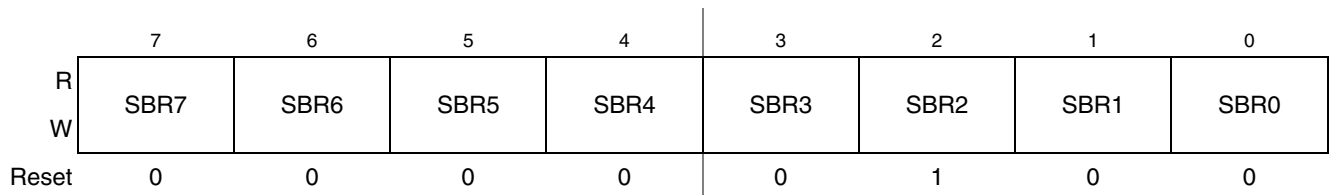


Figure 10-5. SCI Baud Rate Register (SCIxBDL)

Table 10-2. SCIxBDL Field Descriptions

Field	Description
7:0 SBR[7:0]	<b>Baud Rate Modulo Divisor</b> — 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 <a href="#">Table 10-1</a> .

## 10.2.2 SCI Control Register 1 (SCIxC1)

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

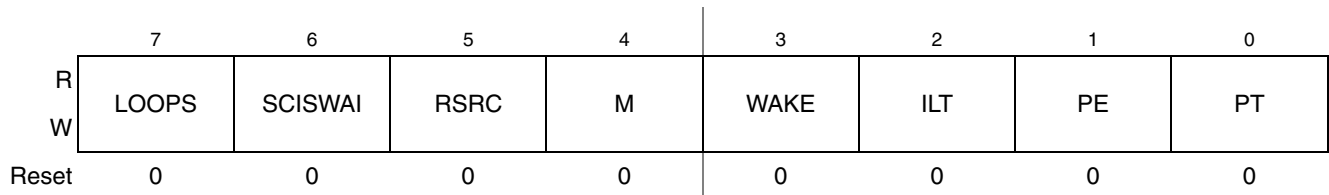


Figure 10-6. SCI Control Register 1 (SCIxC1)

Table 10-3. SCIxC1 Field Descriptions

Field	Description
7 LOOPS	<b>Loop Mode Select</b> — 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 <a href="#">RSRC</a> bit.) RxD pin is not used by SCI.
6 SCISWAI	<b>SCI Stops in Wait Mode</b> 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	<b>Receiver Source Select</b> — 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	<b>9-Bit or 8-Bit Mode Select</b> 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	<b>Receiver Wakeup Method Select</b> — Refer to <a href="#">Section 10.3.3.2, “Receiver Wakeup Operation”</a> for more information. 0 Idle-line wakeup. 1 Address-mark wakeup.
2 ILT	<b>Idle Line Type Select</b> — 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 <a href="#">Section 10.3.3.2.1, “Idle-Line Wakeup”</a> for more information. 0 Idle character bit count starts after start bit. 1 Idle character bit count starts after stop bit.

Table 10-3. SC1xC1 Field Descriptions (continued)

Field	Description
1 PE	<b>Parity Enable</b> — 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	<b>Parity Type</b> — 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.

### 10.2.3 SCI Control Register 2 (SC1xC2)

This register can be read or written at any time.

	7	6	5	4	3	2	1	0
R	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
W								
Reset	0	0	0	0	0	0	0	0

Figure 10-7. SCI Control Register 2 (SC1xC2)

Table 10-4. SC1xC2 Field Descriptions

Field	Description
7 TIE	<b>Transmit Interrupt Enable (for TDRE)</b> 0 Hardware interrupts from TDRE disabled (use polling). 1 Hardware interrupt requested when TDRE flag is 1.
6 TCIE	<b>Transmission Complete Interrupt Enable (for TC)</b> 0 Hardware interrupts from TC disabled (use polling). 1 Hardware interrupt requested when TC flag is 1.
5 RIE	<b>Receiver Interrupt Enable (for RDRF)</b> 0 Hardware interrupts from RDRF disabled (use polling). 1 Hardware interrupt requested when RDRF flag is 1.
4 ILIE	<b>Idle Line Interrupt Enable (for IDLE)</b> 0 Hardware interrupts from IDLE disabled (use polling). 1 Hardware interrupt requested when IDLE flag is 1.
3 TE	<b>Transmitter Enable</b> 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 <a href="#">Section 10.3.2.1, “Send Break and Queued Idle”</a> 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 10-4. SC1xC2 Field Descriptions (continued)

Field	Description
2 RE	<b>Receiver Enable</b> — 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	<b>Receiver Wakeup Control</b> — 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 <a href="#">Section 10.3.3.2, “Receiver Wakeup Operation”</a> for more details. 0 Normal SCI receiver operation. 1 SCI receiver in standby waiting for wakeup condition.
0 SBK	<b>Send Break</b> — 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 <a href="#">Section 10.3.2.1, “Send Break and Queued Idle”</a> for more details. 0 Normal transmitter operation. 1 Queue break character(s) to be sent.

## 10.2.4 SCI Status Register 1 (SC1xS1)

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.

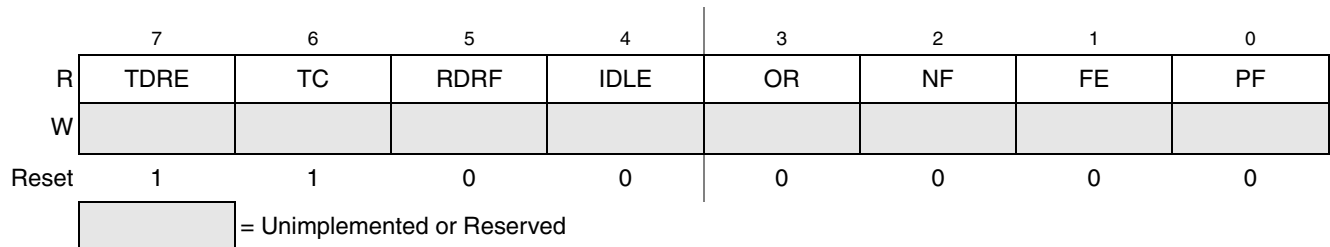


Figure 10-8. SCI Status Register 1 (SC1xS1)

Table 10-5. SC1xS1 Field Descriptions

Field	Description
7 TDRE	<b>Transmit Data Register Empty Flag</b> — 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 SC1xS1 with TDRE = 1 and then write to the SCI data register (SC1xD). 0 Transmit data register (buffer) full. 1 Transmit data register (buffer) empty.
6 TC	<b>Transmission Complete Flag</b> — TC is set out of reset and when TDRE = 1 and no data, preamble, or break character is being transmitted. 0 Transmitter active (sending data, a preamble, or a break). 1 Transmitter idle (transmission activity complete). TC is cleared automatically by reading SC1xS1 with TC = 1 and then doing one of the following three things: <ul style="list-style-type: none"> <li>• Write to the SCI data register (SC1xD) to transmit new data</li> <li>• Queue a preamble by changing TE from 0 to 1</li> <li>• Queue a break character by writing 1 to SBK in SC1xC2</li> </ul>
5 RDRF	<b>Receive Data Register Full Flag</b> — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SC1xD). To clear RDRF, read SC1xS1 with RDRF = 1 and then read the SCI data register (SC1xD). 0 Receive data register empty. 1 Receive data register full.
4 IDLE	<b>Idle Line Flag</b> — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. To clear IDLE, read SC1xS1 with IDLE = 1 and then read the SCI data register (SC1xD). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period. 0 No idle line detected. 1 Idle line was detected.
3 OR	<b>Receiver Overrun Flag</b> — 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 SC1xD yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SC1xD. To clear OR, read SC1xS1 with OR = 1 and then read the SCI data register (SC1xD). 0 No overrun. 1 Receive overrun (new SCI data lost).
2 NF	<b>Noise Flag</b> — 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 SC1xS1 and then read the SCI data register (SC1xD). 0 No noise detected. 1 Noise detected in the received character in SC1xD.



Table 10-5. SCiXs1 Field Descriptions (continued)

Field	Description
1 FE	<b>Framing Error Flag</b> — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCiXs1 with FE = 1 and then read the SCI data register (SCiXD). 0 No framing error detected. This does not guarantee the framing is correct. 1 Framing error.
0 PF	<b>Parity Error Flag</b> — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCiXs1 and then read the SCI data register (SCiXD). 0 No parity error. 1 Parity error.

## 10.2.5 SCI Status Register 2 (SCiXs2)

This register has one read-only status flag.

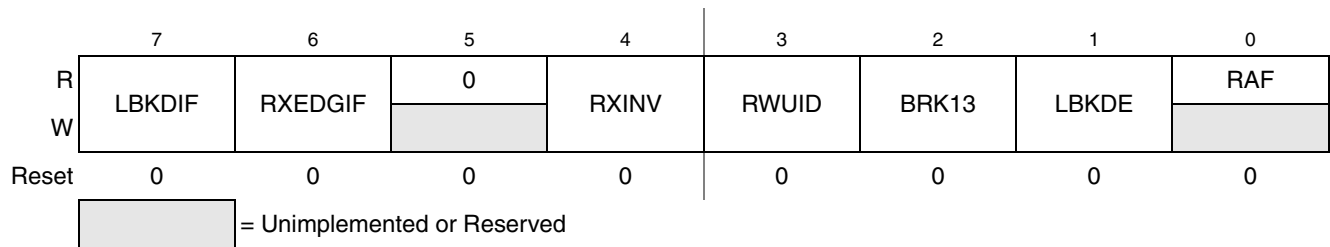


Figure 10-9. SCI Status Register 2 (SCiXs2)

Table 10-6. SCiXs2 Field Descriptions

Field	Description
7 LBKDIF	<b>LIN Break Detect Interrupt Flag</b> — 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	<b>RxD Pin Active Edge Interrupt Flag</b> — 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 <sup>1</sup>	<b>Receive Data Inversion</b> — Setting this bit reverses the polarity of the received data input. 0 Receive data not inverted 1 Receive data inverted
3 RWUID	<b>Receive Wake Up Idle Detect</b> — 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	<b>Break Character Generation Length</b> — 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 10-6. SCiX2 Field Descriptions (continued)

Field	Description
1 LBKDE	<b>LIN Break Detection Enable</b> —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	<b>Receiver Active Flag</b> — 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).

<sup>1</sup> 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.

## 10.2.6 SCI Control Register 3 (SCiXC3)

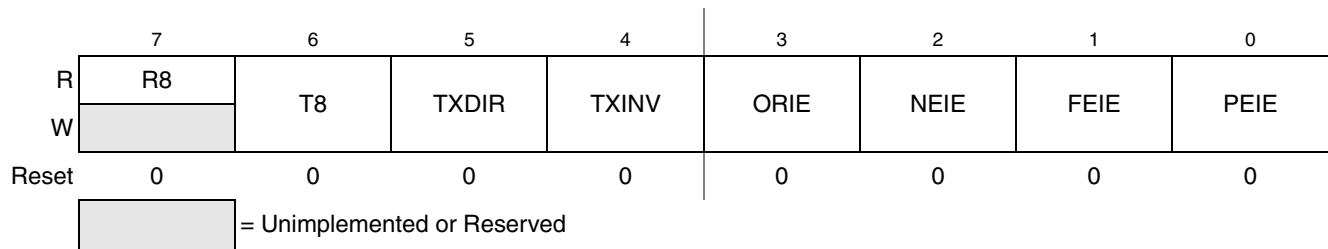


Figure 10-10. SCI Control Register 3 (SCiXC3)

Table 10-7. SCiXC3 Field Descriptions

Field	Description
7 R8	<b>Ninth Data Bit for Receiver</b> — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the SCiXD register. When reading 9-bit data, read R8 before reading SCiXD because reading SCiXD completes automatic flag clearing sequences which could allow R8 and SCiXD to be overwritten with new data.
6 T8	<b>Ninth Data Bit for Transmitter</b> — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the SCiXD register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCiXD is written so T8 should be written (if it needs to change from its previous value) before SCiXD is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCiXD is written.
5 TXDIR	<b>TxD Pin Direction in Single-Wire Mode</b> — 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 10-7. SCIxC3 Field Descriptions (continued)

Field	Description
4 TXINV <sup>1</sup>	<b>Transmit Data Inversion</b> — Setting this bit reverses the polarity of the transmitted data output. 0 Transmit data not inverted 1 Transmit data inverted
3 ORIE	<b>Overrun Interrupt Enable</b> — 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	<b>Noise Error Interrupt Enable</b> — 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	<b>Framing Error Interrupt Enable</b> — 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	<b>Parity Error Interrupt Enable</b> — 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.

<sup>1</sup> Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

## 10.2.7 SCI Data Register (SClxD)

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

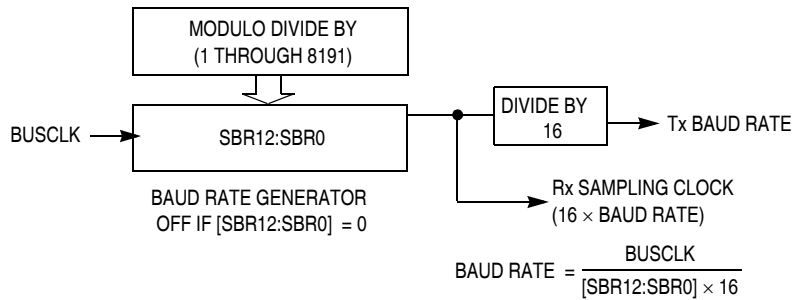
Figure 10-11. SCI Data Register (SClxD)

## 10.3 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.

### 10.3.1 Baud Rate Generation

As shown in [Figure 10-12](#), the clock source for the SCI baud rate generator is the bus-rate clock.



**Figure 10-12. 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  $\pm 4.5$  percent for 8-bit data format and about  $\pm 4$  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.

### 10.3.2 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 10-2](#).

The transmitter output (TxD) idle state defaults to logic high ( $\text{TXINV} = 0$  following reset). The transmitter output is inverted by setting  $\text{TXINV} = 1$ . The transmitter is enabled by setting the TE bit in  $\text{SCIx}C2$ . 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 ( $\text{SCIxD}$ ).

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

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

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

### 10.3.2.1 Send Break and Queued Idle

The SBK control bit in SCIxC2 is used to send break characters which were originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0 (10 bit times including the start and stop bits). A longer break of 13 bit times can be enabled by setting BRK13 = 1. Normally, a program would wait for TDRE to become set to indicate the last character of a message has moved to the transmit shifter, then write 1 and then write 0 to the SBK bit. This action queues a break character to be sent as soon as the shifter is available. If SBK is still 1 when the queued break moves into the shifter (synchronized to the baud rate clock), an additional break character is queued. If the receiving device is another Freescale Semiconductor SCI, the break characters will be received as 0s in all eight data bits and a framing error (FE = 1) occurs.

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

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

**Table 10-8. 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

### 10.3.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 10-3) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver input is inverted by setting RXINV = 1. The receiver is enabled by setting the RE bit in SCIxC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section 10.3.5.1, “8- and 9-Bit Data Modes.” For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

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

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

When a program detects that the receive data register is full ( $RDRF = 1$ ), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to [Section 10.3.4, "Interrupts and Status Flags"](#) for more details about flag clearing.

### 10.3.3.1 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.

### 10.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIxC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant

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

#### 10.3.3.2.1 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.

#### 10.3.3.2.2 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.

### 10.3.4 Interrupts and Status Flags

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

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

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

When a program detects that the receive data register is full ( $RDRF = 1$ ), it gets the data from the receive data register by reading  $SCIxD$ . The  $RDRF$  flag is cleared by reading  $SCIxS1$  while  $RDRF = 1$  and then reading  $SCIxD$ .

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used,  $SCIxS1$  must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the  $RxD$  line remains idle for an extended period of time. IDLE is cleared by reading  $SCIxS1$  while  $IDLE = 1$  and then reading  $SCIxD$ . After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set  $RDRF$ .

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

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

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

### 10.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

#### 10.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in  $SCIxC1$ . In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in  $SCIxC3$ . For the receiver, the ninth bit is held in R8 in  $SCIxC3$ .

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

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from  $SCIxD$  to the shifter.

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



### 10.3.5.2 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.

### 10.3.5.3 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.

### 10.3.5.4 Single-Wire Operation

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD pin. The RxD pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIxC3 controls the direction of serial data on the TxD pin. When TXDIR = 0, the TxD pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD pin so an external device can send serial data to the receiver. When TXDIR = 1, the TxD pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.



---

# Chapter 11

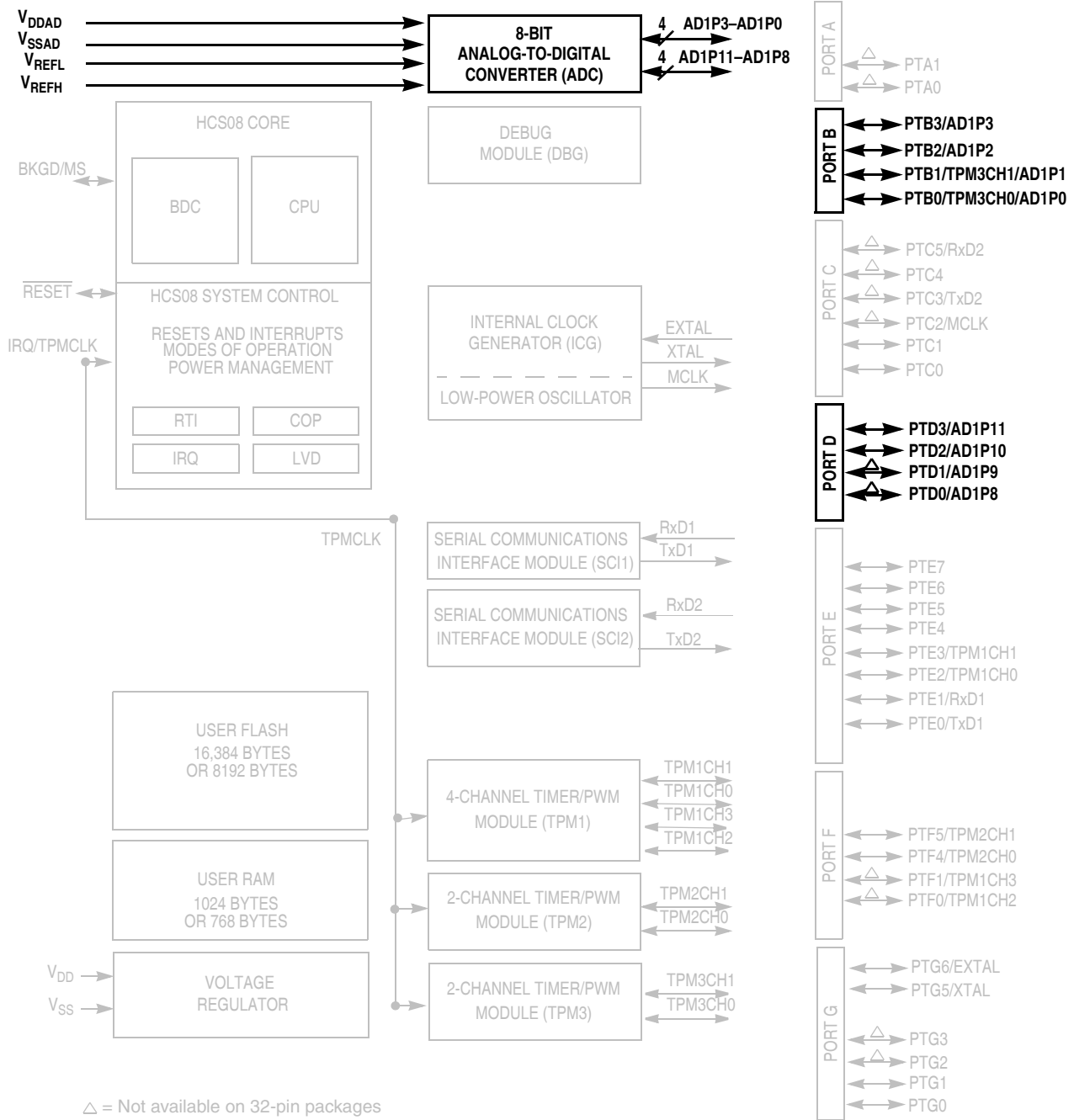
## Analog-to-Digital Converter (S08ADC10V1)

### 11.1 Overview

The 8-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip. The ADC module design supports up to 28 separate analog inputs (AD0-AD27). Only 8 (AD0-AD3, and AD8-AD11) of the possible inputs are implemented on the SC9S08MZ16. These inputs are selected by the ADCH bits. Some inputs are shared with I/O pins as shown in [Figure 11-1](#). All of the channel assignments of the ADC for the SC9S08MZ16 are summarized in [Table 11-1](#).

#### NOTE

The ADC in SC9S08MZ16 only provides 8-bit operation. The 10-bit and 12-bit operations are not supported. This chapter reserves the description of 10-bit and 12-bit operations to be compatible with the reference manuals of other Freescale 8-bit MCUs.



**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<sub>DD</sub>. IRQ should not be driven above V<sub>DD</sub>.
4. Pin contains integrated pullup device.
5. PTD3 and PTD2 contain both pullup and pulldown devices.

**Figure 11-1. SC9S08MZ16 Block Diagram Highlighting ADC Block and Pins**

## 11.2 Channel Assignments

The ADC channel assignments for the SC9S08MZ16 are shown in the table below. Channels that are unimplemented are internally connected to  $V_{REFL}$ . Reserved channels convert to an unknown value. Channels which are connected to an I/O pin have an associated pin control bit as shown.

Table 11-1. ADC Channel Assignment

ADCH	Channel	Input	Pin Control	ADCH	Channel	Input	Pin Control
00000	AD0	PTB0/AD1P0	ADPC0	10000	AD16	$V_{REFL}$	N/A
00001	AD1	PTB1/AD1P1	ADPC1	10001	AD17	$V_{REFL}$	N/A
00010	AD2	PTB2/TPM3CH1 /AD1P2	ADPC2	10010	AD18	$V_{REFL}$	N/A
00011	AD3	PTB3/TPM3CH0 /AD1P3	ADPC3	10011	AD19	$V_{REFL}$	N/A
00100	AD4	$V_{REFL}$	N/A	10100	AD20	$V_{REFL}$	N/A
00101	AD5	$V_{REFL}$	N/A	10101	AD21	$V_{REFL}$	N/A
00110	AD6	$V_{REFL}$	N/A	10110	AD22	Reserved	N/A
00111	AD7	$V_{REFL}$	N/A	10111	AD23	Reserved	N/A
01000	AD8	PTD0/AD1P8	ADPC8	11000	AD24	Reserved	N/A
01001	AD9	PTD1/AD1P9	ADPC9	11001	AD25	Reserved	N/A
01010	AD10	PTD2/AD1P10	ADPC10	11010	AD26	Temp Sensor	N/A
01011	AD11	PTD3/AD1P11	ADPC11	11011	AD27	Internal Bandgap	N/A
01100	AD12	$V_{REFL}$	N/A	11100		Reserved	N/A
01101	AD13	$V_{REFL}$	N/A	11101	$V_{REFH}$	$V_{REFH}$	N/A
01110	AD14	$V_{REFL}$	N/A	11110	$V_{REFL}$	$V_{REFL}$	N/A
01111	AD15	$V_{REFL}$	N/A	11111	module disabled	None	N/A

### NOTE

Selecting the internal bandgap channel requires  $BGBE = 1$  in  $SPMSC1$  see [Section 5.9.8, “System Power Management Status and Control 1 Register \(SPMSC1\).”](#) For value of bandgap voltage reference see [Section A.5, “DC Characteristics.”](#)

### 11.2.1 Alternate Clock

The ADC module is capable of performing conversions using the MCU bus clock, the bus clock divided by two, the local asynchronous clock (ADACK) within the module, or the alternate clock, ALTCLK. The alternate clock for the SC9S08MZ16 MCU devices is the external reference clock (ICGERCLK) from the internal clock generator (ICG) module.

Because ICGERCLK is active only while an external clock source is enabled, the ICG must be configured for either FBE or FEE mode ( $CLKS1 = 1$ ). ICGERCLK must run at a frequency such that the ADC conversion clock (ADCK) runs at a frequency within its specified range ( $f_{ADCK}$ ) after being divided down from the ALTCLK input as determined by the ADIV bits. For example, if the ADIV bits are set up to divide by four, then the minimum frequency for ALTCLK (ICGERCLK) is four times the minimum value for

$f_{\text{ADCK}}$  and the maximum frequency is four times the maximum value for  $f_{\text{ADCK}}$ . Because of the minimum frequency requirement, when an oscillator circuit is used it must be configured for high range operation (RANGE = 1).

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

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

## 11.2.2 Hardware Trigger

The ADC hardware trigger, ADHWT, is output from the real time interrupt (RTI) counter. The RTI counter can be clocked by either ICGERCLK or a nominal 1 kHz clock source within the RTI block. The 1 kHz clock source can be used with the MCU in run, wait, or stop3. With the ICG configured for either FBE or FEE mode, ICGERCLK can be used with the MCU in run or wait.

The period of the RTI is determined by the input clock frequency and the RTIS bits. When the ADC hardware trigger is enabled, a conversion is initiated upon an RTI counter overflow. The RTI counter is a free running counter that generates an overflow at the RTI rate determined by the RTIS bits.

### 11.2.2.1 Analog Pin Enables

The ADC on SC9S08MZ16 contains only two analog pin enable registers, APCTL1 and APCTL2.

### 11.2.2.2 Low-Power Mode Operation

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

## 11.2.3 Temperature Sensor

The ADC1 module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 11-1 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{\text{TEMP}} - V_{\text{TEMP25}}) \div m) \quad \text{Eqn. 11-1}$$

where:

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

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

In application code, the user reads the temperature sensor channel, calculates  $V_{\text{TEMP}}$  and compares to  $V_{\text{TEMP25}}$ . If  $V_{\text{TEMP}}$  is greater than  $V_{\text{TEMP25}}$  the cold slope value is applied in Equation 11-1. If  $V_{\text{TEMP}}$  is less than  $V_{\text{TEMP25}}$  the hot slope value is applied in Equation 11-1.

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

## 11.2.4 Features

Features of the ADC module include:

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

## 11.2.5 Block Diagram

[Figure 11-2](#) provides a block diagram of the ADC module

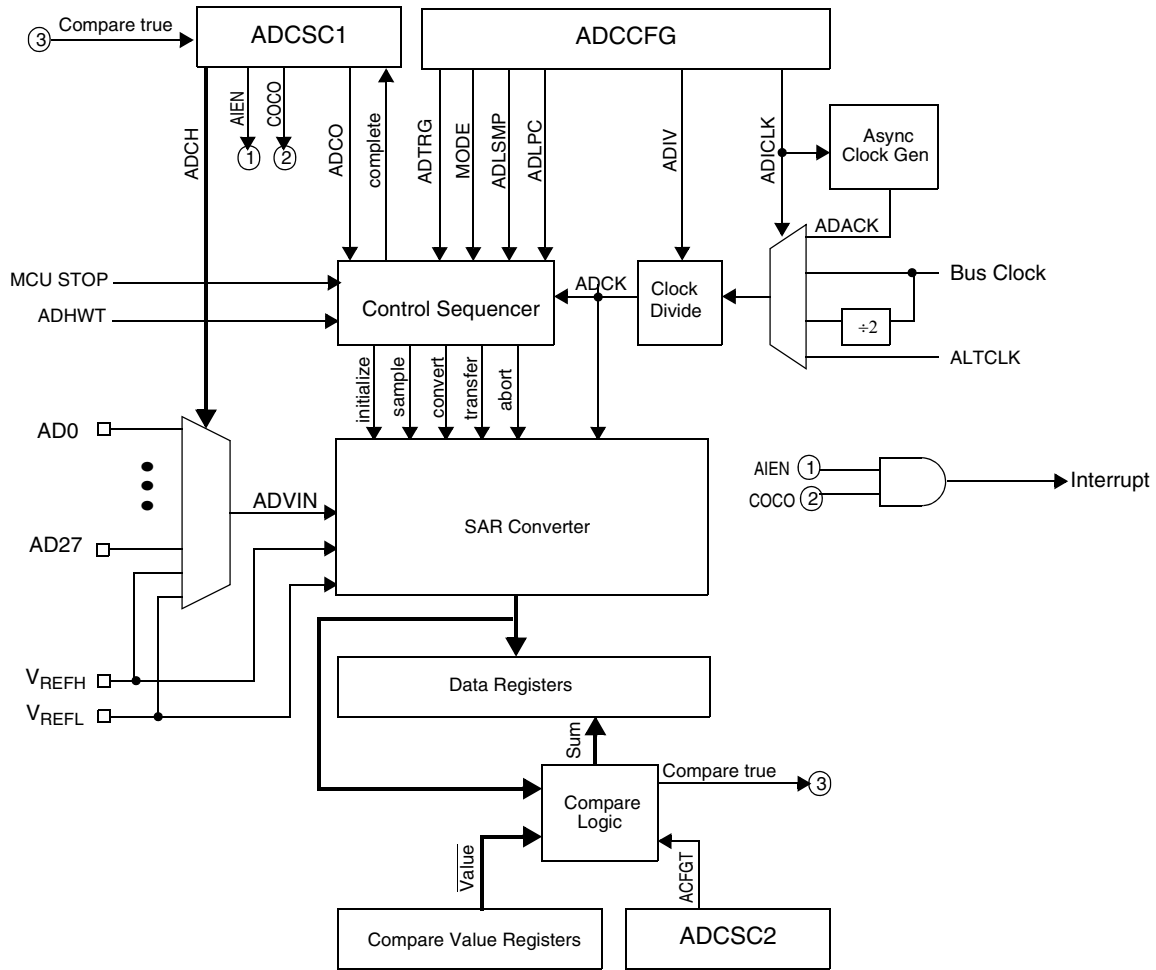


Figure 11-2. ADC Block Diagram

### 11.3 External Signal Description

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

Table 11-2. Signal Properties

Name	Function
AD27–AD0	Analog Channel inputs
V <sub>REFH</sub>	High reference voltage
V <sub>REFL</sub>	Low reference voltage
V <sub>DDAD</sub>	Analog power supply
V <sub>SSAD</sub>	Analog ground



### 11.3.1 Analog Power ( $V_{DDAD}$ )

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

### 11.3.2 Analog Ground ( $V_{SSAD}$ )

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

### 11.3.3 Voltage Reference High ( $V_{REFH}$ )

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

### 11.3.4 Voltage Reference Low ( $V_{REFL}$ )

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

### 11.3.5 Analog Channel Inputs (ADx)

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

## 11.4 Register Definition

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

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

### 11.4.1 Status and Control Register 1 (ADCSC1)

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

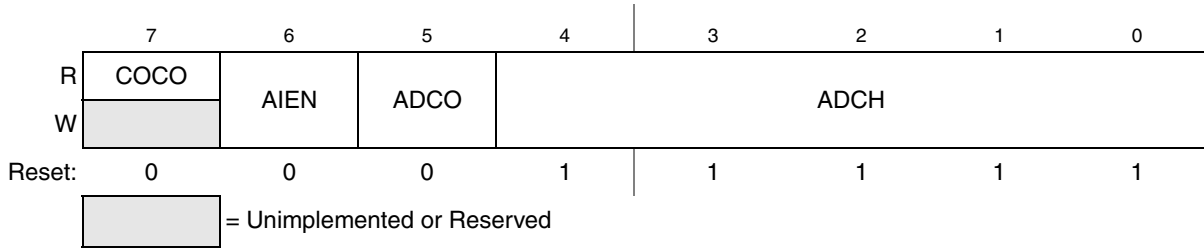


Figure 11-3. Status and Control Register (ADCSC1)

Table 11-3. ADCSC1 Register Field Descriptions

Field	Description
7 COCO	<b>Conversion Complete Flag</b> — The COCO flag is a read-only bit which is set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1) the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared whenever ADCSC1 is written or whenever ADCRL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	<b>Interrupt Enable</b> — AIEN is used to enable conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 ADCO	<b>Continuous Conversion Enable</b> — ADCO is used to enable continuous conversions. 0 One conversion following a write to the ADCSC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected. 1 Continuous conversions initiated following a write to ADCSC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected.
4:0 ADCH	<b>Input Channel Select</b> — The ADCH bits form a 5-bit field which is used to select one of the input channels. The input channels are detailed in Figure 11-4. The successive approximation converter subsystem is turned off when the channel select bits are all set to 1. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way will prevent an additional, single conversion from being performed. It is not necessary to set the channel select bits to all 1s to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

Figure 11-4. Input Channel Select

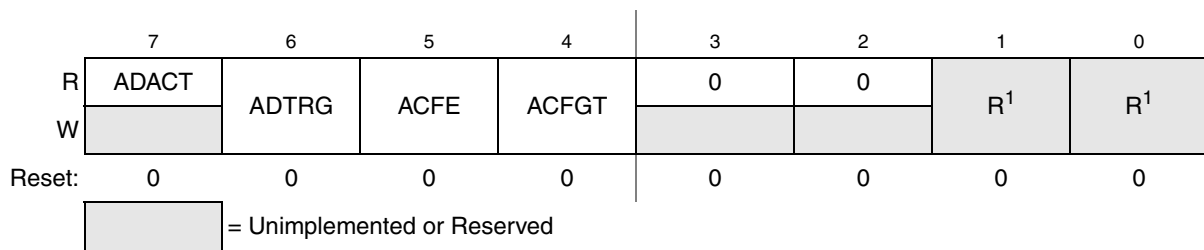
ADCH	Input Select	ADCH	Input Select
00000	AD0	10000	AD16
00001	AD1	10001	AD17
00010	AD2	10010	AD18
00011	AD3	10011	AD19
00100	AD4	10100	AD20
00101	AD5	10101	AD21
00110	AD6	10110	AD22

Figure 11-4. Input Channel Select (continued)

ADCH	Input Select	ADCH	Input Select
00111	AD7	10111	AD23
01000	AD8	11000	AD24
01001	AD9	11001	AD25
01010	AD10	11010	AD26
01011	AD11	11011	AD27
01100	AD12	11100	Reserved
01101	AD13	11101	V <sub>REFH</sub>
01110	AD14	11110	V <sub>REFL</sub>
01111	AD15	11111	Module disabled

## 11.4.2 Status and Control Register 2 (ADCSC2)

The ADCSC2 register is used to control the compare function, conversion trigger and conversion active of the ADC module.



<sup>1</sup> Bits 1 and 0 are reserved bits that must always be written to 0.

Figure 11-5. Status and Control Register 2 (ADCSC2)

Table 11-4. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	<b>Conversion Active</b> — ADACT indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress 1 Conversion in progress
6 ADTRG	<b>Conversion Trigger Select</b> — ADTRG is used to select the type of trigger to be used for initiating a conversion. Two types of trigger are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input. 0 Software trigger selected 1 Hardware trigger selected

Table 11-4. ADCSC2 Register Field Descriptions (continued)

Field	Description
5 ACFE	<b>Compare Function Enable</b> — ACFE is used to enable the compare function. 0 Compare function disabled 1 Compare function enabled
4 ACFGT	<b>Compare Function Greater Than Enable</b> — ACFGT is used to configure the compare function to trigger when the result of the conversion of the input being monitored is greater than or equal to the compare value. The compare function defaults to triggering when the result of the compare of the input being monitored is less than the compare value. 0 Compare triggers when input is less than compare level 1 Compare triggers when input is greater than or equal to compare level

### 11.4.3 Data Result High Register (ADCRH)

ADCRH contains the upper two bits of the result of a 10-bit conversion. When configured for 8-bit conversions both ADR8 and ADR9 are equal to zero. ADCRH is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. In 10-bit MODE, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion is completed, then the intermediate conversion result will be lost. In 8-bit mode there is no interlocking with ADCRL. In the case that the MODE bits are changed, any data in ADCRH becomes invalid.

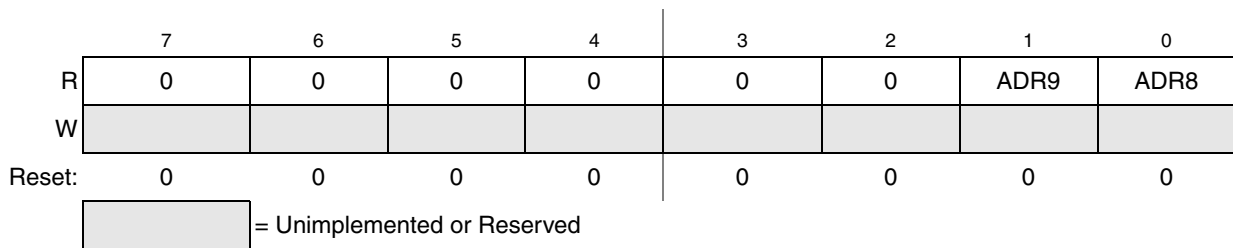


Figure 11-6. Data Result High Register (ADCRH)

### 11.4.4 Data Result Low Register (ADCRL)

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



Figure 11-7. Data Result Low Register (ADCRL)

### 11.4.5 Compare Value High Register (ADCCVH)

This register holds the upper two bits of the 10-bit compare value. These bits are compared to the upper two bits of the result following a conversion in 10-bit mode when the compare function is enabled. In 8-bit operation, ADCCVH is not used during compare.

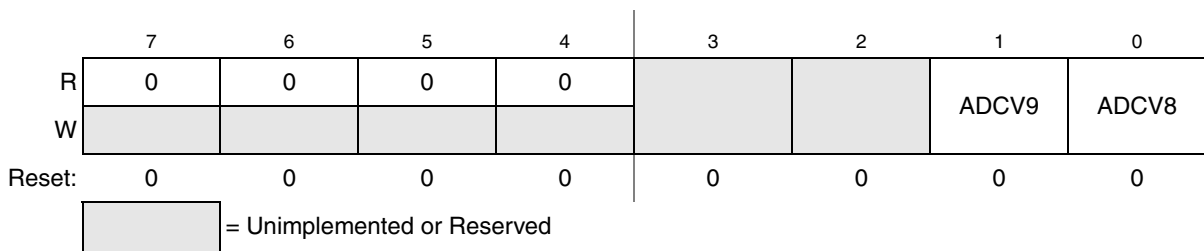


Figure 11-8. Compare Value High Register (ADCCVH)

### 11.4.6 Compare Value Low Register (ADCCVL)

This register holds the lower 8 bits of the 10-bit compare value, or all 8 bits of the 8-bit compare value. Bits ADCV7:ADCV0 are compared to the lower 8 bits of the result following a conversion in either 10-bit or 8-bit mode.

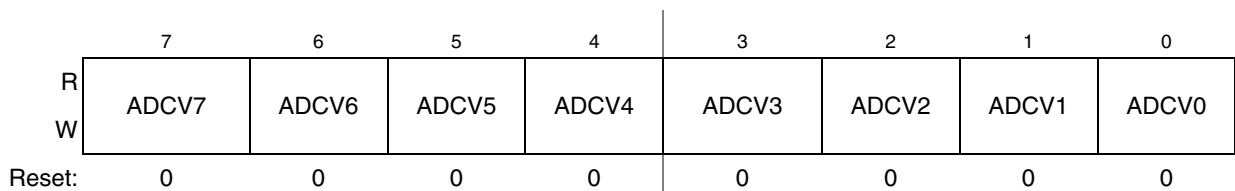
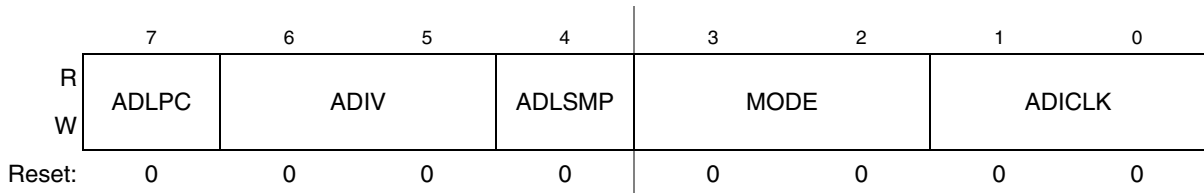


Figure 11-9. Compare Value Low Register(ADCCVL)

### 11.4.7 Configuration Register (ADCCFG)

ADCCFG is used to select the mode of operation, clock source, clock divide, and configure for low power or long sample time.



**Figure 11-10. Configuration Register (ADCCFG)**

**Table 11-5. ADCCFG Register Field Descriptions**

Field	Description
7 ADLPC	<b>Low Power Configuration</b> — ADLPC controls the speed and power configuration of the successive approximation converter. This is used to optimize power consumption when higher sample rates are not required. 0 High speed configuration 1 Low power configuration: {FC31}The power is reduced at the expense of maximum clock speed.
6:5 ADIV	<b>Clock Divide Select</b> — ADIV select the divide ratio used by the ADC to generate the internal clock ADCK. <a href="#">Table 11-6</a> shows the available clock configurations.
4 ADLSMP	<b>Long Sample Time Configuration</b> — ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. 0 Short sample time 1 Long sample time
3:2 MODE	<b>Conversion Mode Selection</b> — MODE bits are used to select between 10- or 8-bit operation. See <a href="#">Table 11-7</a> .
1:0 ADICLK	<b>Input Clock Select</b> — ADICLK bits select the input clock source to generate the internal clock ADCK. See <a href="#">Table 11-8</a> .

**Table 11-6. Clock Divide Select**

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

**Table 11-7. Conversion Modes**

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

Table 11-8. Input Clock Select

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

### 11.4.8 Pin Control 1 Register (APCTL1)

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

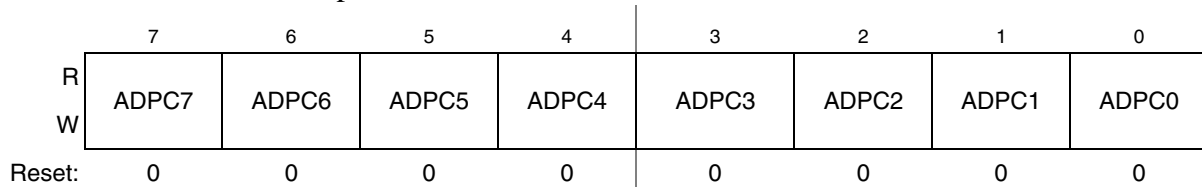


Figure 11-11. Pin Control 1 Register (APCTL1)

Table 11-9. APCTL1 Register Field Descriptions

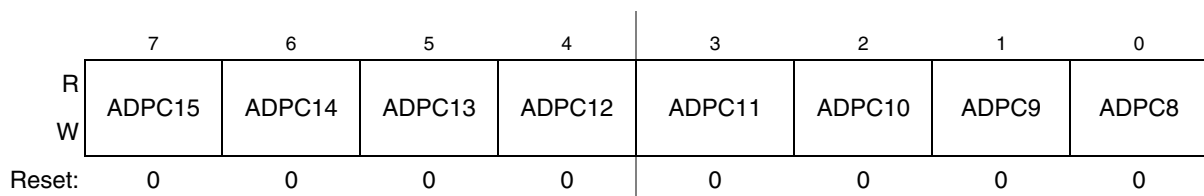
Field	Description
7 ADPC7	<b>ADC Pin Control 7</b> — ADPC7 is used to control the pin associated with channel AD7. 0 AD7 pin I/O control enabled 1 AD7 pin I/O control disabled
6 ADPC6	<b>ADC Pin Control 6</b> — ADPC6 is used to control the pin associated with channel AD6. 0 AD6 pin I/O control enabled 1 AD6 pin I/O control disabled
5 ADPC5	<b>ADC Pin Control 5</b> — ADPC5 is used to control the pin associated with channel AD5. 0 AD5 pin I/O control enabled 1 AD5 pin I/O control disabled
4 ADPC4	<b>ADC Pin Control 4</b> — ADPC4 is used to control the pin associated with channel AD4. 0 AD4 pin I/O control enabled 1 AD4 pin I/O control disabled
3 ADPC3	<b>ADC Pin Control 3</b> — ADPC3 is used to control the pin associated with channel AD3. 0 AD3 pin I/O control enabled 1 AD3 pin I/O control disabled
2 ADPC2	<b>ADC Pin Control 2</b> — ADPC2 is used to control the pin associated with channel AD2. 0 AD2 pin I/O control enabled 1 AD2 pin I/O control disabled

**Table 11-9. APCTL1 Register Field Descriptions (continued)**

Field	Description
1 ADPC1	<b>ADC Pin Control 1</b> — ADPC1 is used to control the pin associated with channel AD1. 0 AD1 pin I/O control enabled 1 AD1 pin I/O control disabled
0 ADPC0	<b>ADC Pin Control 0</b> — ADPC0 is used to control the pin associated with channel AD0. 0 AD0 pin I/O control enabled 1 AD0 pin I/O control disabled

### 11.4.9 Pin Control 2 Register (APCTL2)

APCTL2 is used to control channels 8–15 of the ADC module.

**Figure 11-12. Pin Control 2 Register (APCTL2)****Table 11-10. APCTL2 Register Field Descriptions**

Field	Description
7 ADPC15	<b>ADC Pin Control 15</b> — ADPC15 is used to control the pin associated with channel AD15. 0 AD15 pin I/O control enabled 1 AD15 pin I/O control disabled
6 ADPC14	<b>ADC Pin Control 14</b> — ADPC14 is used to control the pin associated with channel AD14. 0 AD14 pin I/O control enabled 1 AD14 pin I/O control disabled
5 ADPC13	<b>ADC Pin Control 13</b> — ADPC13 is used to control the pin associated with channel AD13. 0 AD13 pin I/O control enabled 1 AD13 pin I/O control disabled
4 ADPC12	<b>ADC Pin Control 12</b> — ADPC12 is used to control the pin associated with channel AD12. 0 AD12 pin I/O control enabled 1 AD12 pin I/O control disabled
3 ADPC11	<b>ADC Pin Control 11</b> — ADPC11 is used to control the pin associated with channel AD11. 0 AD11 pin I/O control enabled 1 AD11 pin I/O control disabled
2 ADPC10	<b>ADC Pin Control 10</b> — ADPC10 is used to control the pin associated with channel AD10. 0 AD10 pin I/O control enabled 1 AD10 pin I/O control disabled



Table 11-10. APCTL2 Register Field Descriptions (continued)

Field	Description
1 ADPC9	<b>ADC Pin Control 9</b> — ADPC9 is used to control the pin associated with channel AD9. 0 AD9 pin I/O control enabled 1 AD9 pin I/O control disabled
0 ADPC8	<b>ADC Pin Control 8</b> — ADPC8 is used to control the pin associated with channel AD8. 0 AD8 pin I/O control enabled 1 AD8 pin I/O control disabled

### 11.4.10 Pin Control 3 Register (APCTL3)

APCTL3 is used to control channels 16–23 of the ADC module.

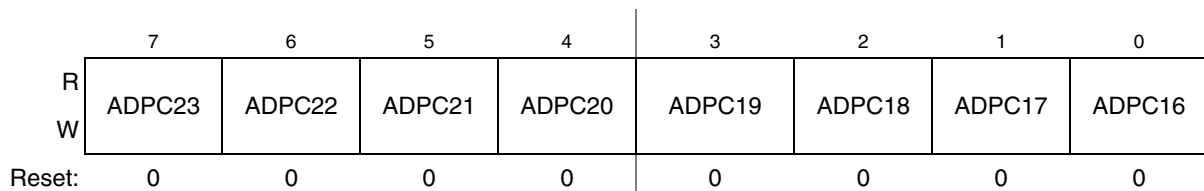


Figure 11-13. Pin Control 3 Register (APCTL3)

Table 11-11. APCTL3 Register Field Descriptions

Field	Description
7 ADPC23	<b>ADC Pin Control 23</b> — ADPC23 is used to control the pin associated with channel AD23. 0 AD23 pin I/O control enabled 1 AD23 pin I/O control disabled
6 ADPC22	<b>ADC Pin Control 22</b> — ADPC22 is used to control the pin associated with channel AD22. 0 AD22 pin I/O control enabled 1 AD22 pin I/O control disabled
5 ADPC21	<b>ADC Pin Control 21</b> — ADPC21 is used to control the pin associated with channel AD21. 0 AD21 pin I/O control enabled 1 AD21 pin I/O control disabled
4 ADPC20	<b>ADC Pin Control 20</b> — ADPC20 is used to control the pin associated with channel AD20. 0 AD20 pin I/O control enabled 1 AD20 pin I/O control disabled
3 ADPC19	<b>ADC Pin Control 19</b> — ADPC19 is used to control the pin associated with channel AD19. 0 AD19 pin I/O control enabled 1 AD19 pin I/O control disabled
2 ADPC18	<b>ADC Pin Control 18</b> — ADPC18 is used to control the pin associated with channel AD18. 0 AD18 pin I/O control enabled 1 AD18 pin I/O control disabled

Table 11-11. APCTL3 Register Field Descriptions (continued)

Field	Description
1 ADPC17	<b>ADC Pin Control 17</b> — ADPC17 is used to control the pin associated with channel AD17. 0 AD17 pin I/O control enabled 1 AD17 pin I/O control disabled
0 ADPC16	<b>ADC Pin Control 16</b> — ADPC16 is used to control the pin associated with channel AD16. 0 AD16 pin I/O control enabled 1 AD16 pin I/O control disabled

## 11.5 Functional Description

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

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. The selected channel voltage is converted by a successive approximation algorithm into an 11-bit digital result. In 8-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 9-bit digital result.

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

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

### 11.5.1 Clock Select and Divide Control

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

- The bus clock, which is equal to the frequency at which software is executed. This is the default selection following reset.
- The bus clock divided by 2. For higher bus clock rates, this allows a maximum divide by 16 of the bus clock.
- ALTCLK, as defined for this MCU (See module section introduction).
- The asynchronous clock (ADACK) – This clock is generated from a clock source within the ADC module. When selected as the clock source this clock remains active while the MCU is in wait or stop3 mode and allows conversions in these modes for lower noise operation.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC will not perform according to specifications. If the available clocks

are too fast, then the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

## 11.5.2 Input Select and Pin Control

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

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

## 11.5.3 Hardware Trigger

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

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

## 11.5.4 Conversion Control

Conversions can be performed in either 10-bit mode or 8-bit mode as determined by the MODE bits. Conversions can be initiated by either a software or hardware trigger. In addition, the ADC module can be configured for low power operation, long sample time, continuous conversion, and automatic compare of the conversion result to a software determined compare value.

### 11.5.4.1 Initiating Conversions

A conversion is initiated:

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

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

### 11.5.4.2 Completing Conversions

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

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

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

### 11.5.4.3 Aborting Conversions

Any conversion in progress will be aborted when:

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

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

### 11.5.4.4 Power Control

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

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

### 11.5.4.5 Total Conversion Time

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

result of the conversion is transferred to ADCRH and ADCRL upon completion of the conversion algorithm.

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

The maximum total conversion time for different conditions is summarized in [Table 11-12](#).

**Table 11-12. Total Conversion Time vs. Control Conditions**

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

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

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

$$\text{Number of bus cycles} = 3.5 \mu\text{s} \times 8 \text{ MHz} = 28 \text{ cycles}$$

#### NOTE

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

## 11.5.5 Automatic Compare Function

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

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

### NOTE

The compare function can be used to monitor the voltage on a channel while the MCU is in either wait or stop3 mode. The ADC interrupt will wake the MCU when the compare condition is met.

## 11.5.6 MCU Wait Mode Operation

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

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

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

## 11.5.7 MCU Stop3 Mode Operation

The STOP instruction is used to put the MCU in a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

### 11.5.7.1 Stop3 Mode With ADACK Disabled

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

### 11.5.7.2 Stop3 Mode With ADACK Enabled

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

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

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

#### NOTE

It is possible for the ADC module to wake the system from low power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure that the data transfer blocking mechanism (discussed in [Section 11.5.4.2, "Completing Conversions"](#)) is cleared when entering stop3 and continuing ADC conversions.

### 11.5.8 MCU Stop1 and Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters either stop1 or stop2 mode. All module registers contain their reset values following exit from stop1 or stop2. Therefore the module must be re-enabled and re-configured following exit from stop1 or stop2.

## 11.6 Initialization Information

This section gives an example which provides some basic direction on how a user would initialize and configure the ADC module. The user has the flexibility of choosing between configuring the module for 8-bit or 10-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to [Table 11-6](#), [Table 11-7](#), and [Table 11-8](#) for information used in this example.

#### NOTE

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

### 11.6.1 ADC Module Initialization Example

#### 11.6.1.1 Initialization Sequence

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

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.

2. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.
3. Update status and control register 1 (ADCSC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

### 11.6.1.2 Pseudo — Code Example

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

#### ADCCFG = 0x98 (%10011000)

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

#### ADCSC2 = 0x00 (%00000000)

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

#### ADCSC1 = 0x41 (%01000001)

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

#### ADCRH/L = 0xxx

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

#### ADCCVH/L = 0xxx

Holds compare value when compare function enabled

#### APCTL1=0x02

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

#### APCTL2=0x00

All other AD pins remain general purpose I/O pins



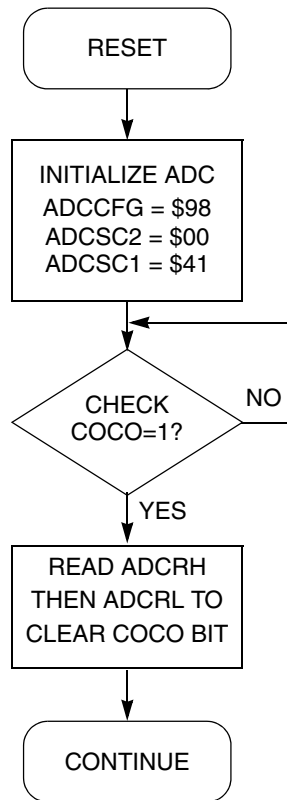


Figure 11-14. Initialization Flowchart for Example

## 11.7 Application Information

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

### 11.7.1 External Pins and Routing

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

#### 11.7.1.1 Analog Supply Pins

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

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

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

### 11.7.1.2 Analog Reference Pins

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

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

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

### 11.7.1.3 Analog Input Pins

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

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

For proper conversion, the input voltage must fall between  $V_{REFH}$  and  $V_{REFL}$ . If the input is equal to or exceeds  $V_{REFH}$ , the converter circuit converts the signal to \$3FF (full scale 10-bit representation) or \$FF (full scale 8-bit representation). If the input is equal to or less than  $V_{REFL}$ , the converter circuit converts it to \$000. Input voltages between  $V_{REFH}$  and  $V_{REFL}$  are straight-line linear conversions. There will be a brief current associated with  $V_{REFL}$  when the sampling capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

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

## 11.7.2 Sources of Error

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

### 11.7.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately  $7\text{k}\Omega$  and input capacitance of approximately  $5.5\text{ pF}$ , sampling to within  $1/4\text{LSB}$  (at 10-bit resolution) can be achieved within the minimum sample window (3.5 cycles @  $8\text{ MHz}$  maximum ADCK frequency) provided the resistance of the external analog source ( $R_{AS}$ ) is kept below  $5\text{ k}\Omega$ .

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

### 11.7.2.2 Pin Leakage Error

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

### 11.7.2.3 Noise-Induced Errors

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

- There is a  $0.1\text{ }\mu\text{F}$  low-ESR capacitor from  $V_{REFH}$  to  $V_{REFL}$ .
- There is a  $0.1\text{ }\mu\text{F}$  low-ESR capacitor from  $V_{DDAD}$  to  $V_{SSAD}$ .
- If inductive isolation is used from the primary supply, an additional  $1\text{ }\mu\text{F}$  capacitor is placed from  $V_{DDAD}$  to  $V_{SSAD}$ .
- $V_{SSAD}$  (and  $V_{REFL}$ , if connected) is connected to  $V_{SS}$  at a quiet point in the ground plane.
- Operate the MCU in wait or stop3 mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
  - For software triggered conversions, immediately follow the write to the ADCSC1 with a WAIT instruction or STOP instruction.
  - For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces  $V_{DD}$  noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

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

- Place a  $0.01\text{ }\mu\text{F}$  capacitor ( $C_{AS}$ ) on the selected input channel to  $V_{REFL}$  or  $V_{SSAD}$  (this will improve noise issues but will affect sample rate based on the external analog source resistance).

- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

### 11.7.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 1024 steps (in 10-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8 or 10), defined as 1LSB, is:

$$1\text{LSB} = (V_{\text{REFH}} - V_{\text{REFL}}) / 2^N \quad \text{Eqn. 11-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code will transition when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be  $\pm 1/2\text{LSB}$  in 8- or 10-bit mode. As a consequence, however, the code width of the first (\$000) conversion is only  $1/2\text{LSB}$  and the code width of the last (\$FF or \$3FF) is  $1.5\text{LSB}$ .

### 11.7.2.5 Linearity Errors

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

- Zero-scale error ( $E_{\text{ZS}}$ ) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width ( $1/2\text{LSB}$ ). Note, if the first conversion is \$001, then the difference between the actual \$001 code width and its ideal (1LSB) is used.
- Full-scale error ( $E_{\text{FS}}$ ) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width ( $1.5\text{LSB}$ ). Note, if the last conversion is \$3FE, then the difference between the actual \$3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function, and therefore includes all forms of error.

### 11.7.2.6 Code Jitter, Non-Monotonicity and Missing Codes

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

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the

converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around  $\pm 1/2$  LSB and will increase with noise. This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in [Section 11.7.2.3](#) will reduce this error.

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

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



# Chapter 12

## Development Support

### 12.1 Introduction

Development support systems in the HCS08 include the background debug controller (BDC) and the on-chip debug module (DBG). The BDC provides a single-wire debug interface to the target MCU that provides a convenient interface for programming the on-chip flash and other nonvolatile memories. The BDC is also the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modify, breakpoints, and single instruction trace commands.

In the HCS08 family, address and data bus signals are not available on external pins (not even in test modes). Debug is done through commands fed into the target MCU via the single-wire background debug interface. The debug module provides a means to selectively trigger and capture bus information so an external development system can reconstruct what happened inside the MCU on a cycle-by-cycle basis without having external access to the address and data signals.

The alternate BDC clock source for SC9S08MZ16 is the ICGCLK. See [Chapter 8, “Internal Clock Generator \(S08ICGV4\)”](#) for more information about ICGCLK and how to select clock sources.

## 12.1.1 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  $\text{address} > B$ )

## 12.2 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  $V_{\text{DD}}$ . 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  $V_{\text{DD}}$  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.

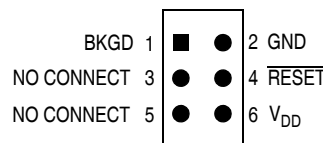


Figure 12-1. BDM Tool Connector

## 12.2.1 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 12.2.2, "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 12.2.2, “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.

## 12.2.2 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.

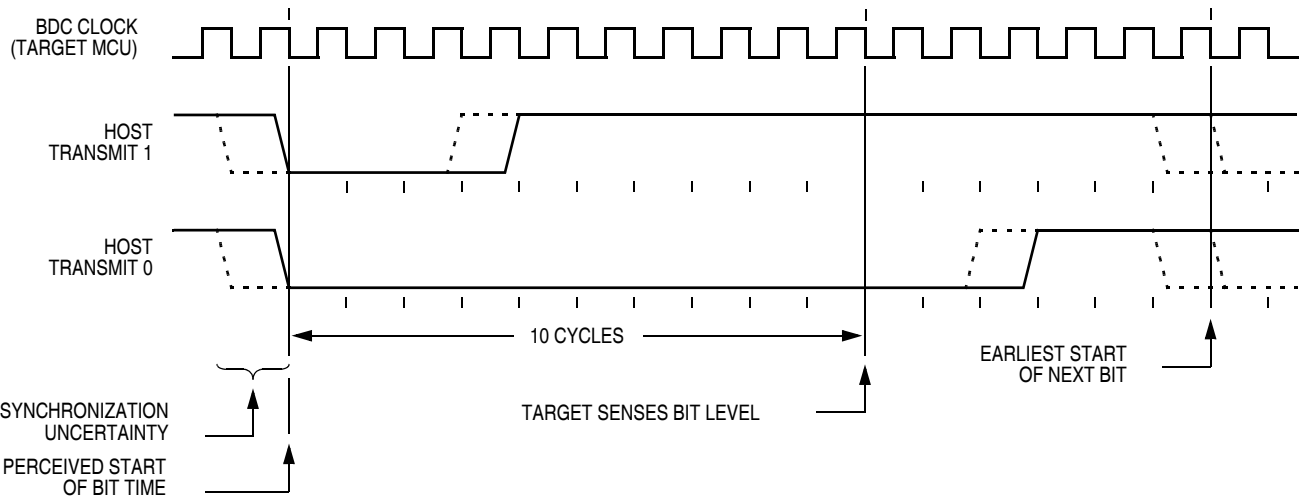
BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 BDC clock cycles per bit (nominal speed). The interface times out if 512 BDC clock cycles occur between falling edges from the host. Any BDC command that was in progress 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.

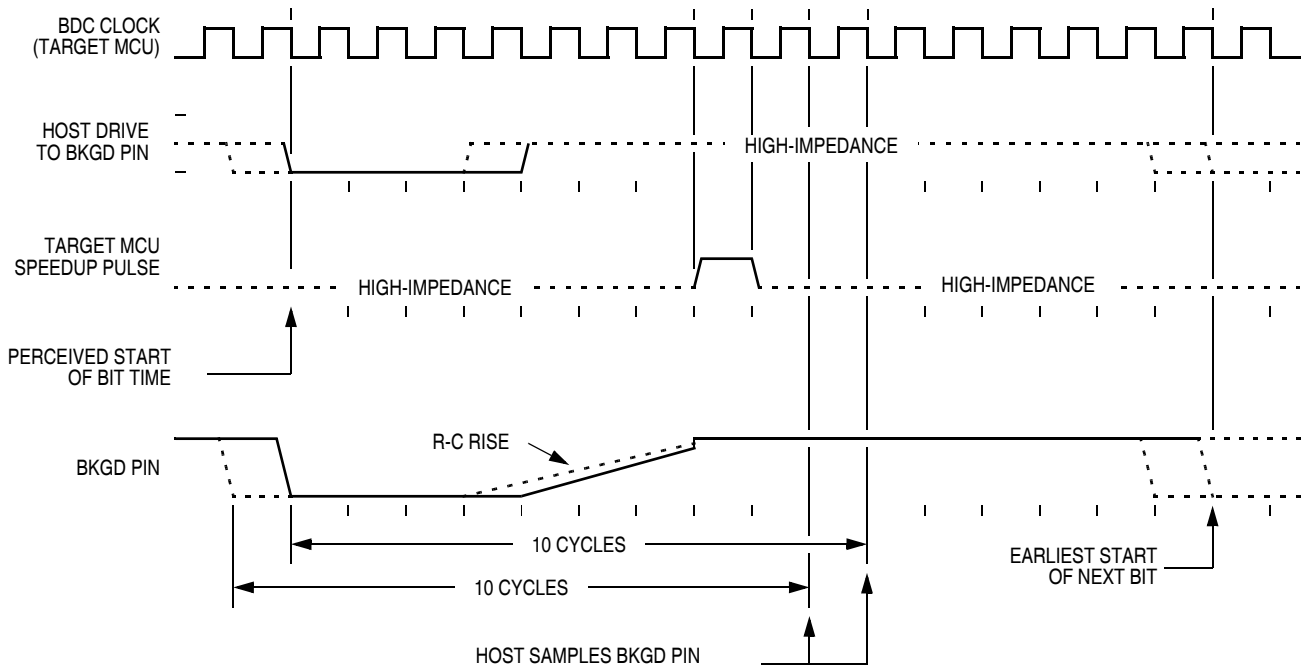
The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

[Figure 12-2](#) 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 12-2. BDC Host-to-Target Serial Bit Timing**

Figure 12-3 shows the host receiving a logic 1 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.



**Figure 12-3. BDC Target-to-Host Serial Bit Timing (Logic 1)**

Figure 12-4 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.

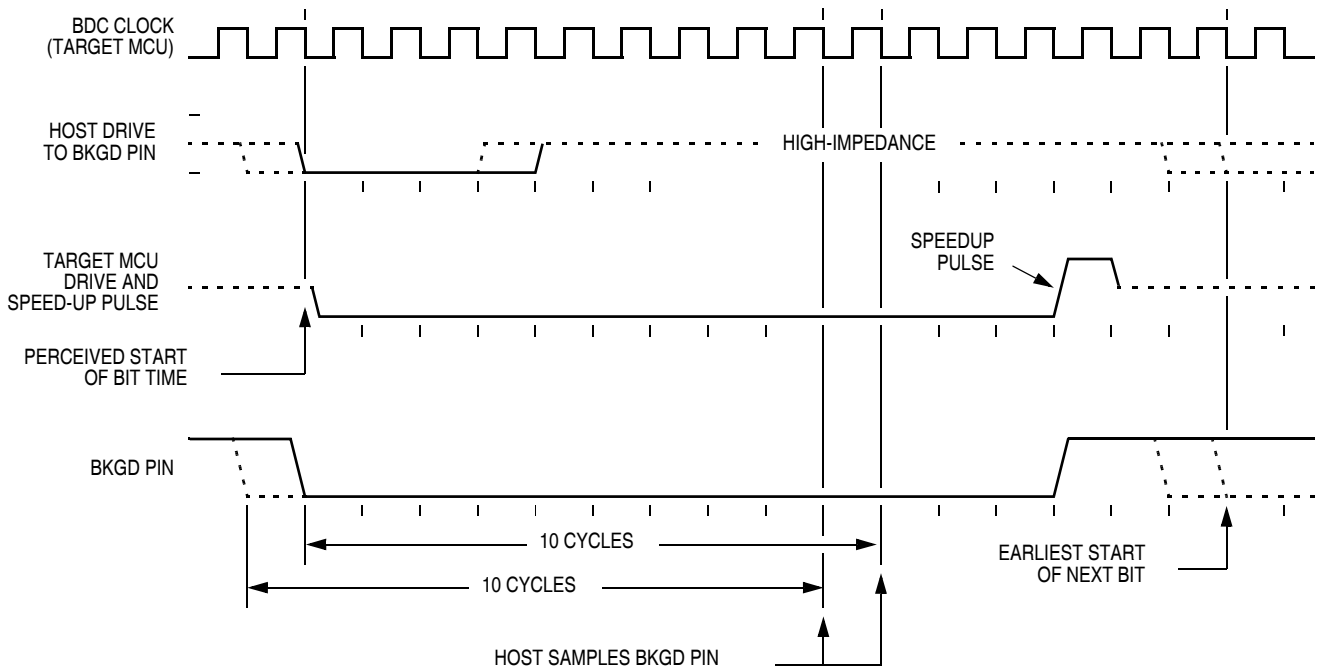


Figure 12-4. BDM Target-to-Host Serial Bit Timing (Logic 0)

### 12.2.3 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 12-1 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 12-1 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 12-1. BDC Command Summary**

Command Mnemonic	Active BDM/ Non-intrusive	Coding Structure	Description
SYNC	Non-intrusive	n/a <sup>1</sup>	Request a timed reference pulse to determine target BDC communication speed
ACK_ENABLE	Non-intrusive	D5/d	Enable acknowledge protocol. Refer to document order no. HCS08RMv1/D.
ACK_DISABLE	Non-intrusive	D6/d	Disable acknowledge protocol. Refer to 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.

<sup>1</sup> 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.

## 12.2.4 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.



## 12.3 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 12.3.6, "Hardware Breakpoints."](#)

### 12.3.1 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)

### 12.3.2 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 12.3.5, “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.

### 12.3.3 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.

### 12.3.4 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 DBGTC 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.

### 12.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGTC register selects one of nine trigger modes. When TRGSEL = 1 in the DBGTC register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGTC 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 DBGTC 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 DBGGEN in DBGTC.

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.

## 12.3.6 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 12.3.5, “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.

## 12.4 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.

### 12.4.1 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.

### 12.4.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ\_STATUS and WRITE\_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

	7	6	5	4	3	2	1	0
R	ENBDM	BDMACT	BKPTEN	FTS	CLKSW	WS	WSF	DVF
W								
Normal Reset	0	0	0	0	0	0	0	0
Reset in Active BDM:	1	1	0	0	1	0	0	0


 = Unimplemented or Reserved

Figure 12-5. BDC Status and Control Register (BDCSCR)

Table 12-2. BDCSCR Register Field Descriptions

Field	Description
7 ENBDM	<b>Enable BDM (Permit Active Background Mode)</b> — 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	<b>Background Mode Active Status</b> — 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	<b>BDC Breakpoint Enable</b> — 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	<b>Force/Tag Select</b> — 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	<b>Select Source for BDC Communications Clock</b> — CLKSW defaults to 0, which selects the alternate BDC clock source. 0 Alternate BDC clock source 1 MCU bus clock

**Table 12-2. BDCSCR Register Field Descriptions (continued)**

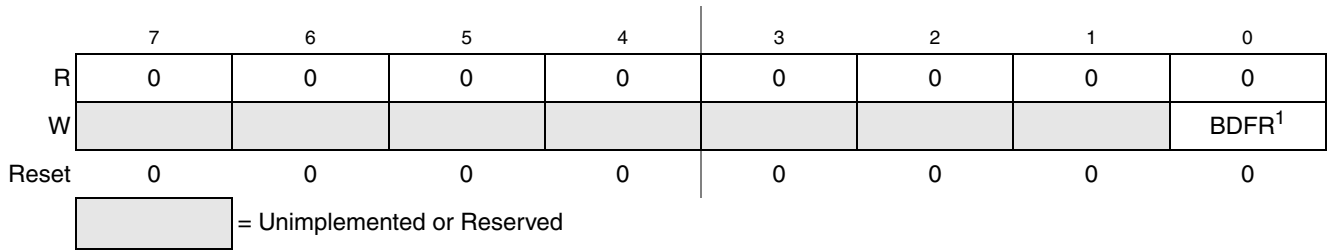
Field	Description
2 WS	<p><b>Wait or Stop Status</b> — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands.</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><b>Wait or Stop Failure Status</b> — 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><b>Data Valid Failure Status</b> — This status bit is not used in the SC9S08MZ16 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>

### 12.4.1.2 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 12.2.4, “BDC Hardware Breakpoint.”](#)

### 12.4.2 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.



<sup>1</sup> BDFR is writable only through serial background mode debug commands, not from user programs.

**Figure 12-6. System Background Debug Force Reset Register (SBDFR)**

**Table 12-3. SBDFR Register Field Description**

Field	Description
0 BDFR	<b>Background Debug Force Reset</b> — 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.

### 12.4.3 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.

#### 12.4.3.1 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.

#### 12.4.3.2 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.

#### 12.4.3.3 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.

#### 12.4.3.4 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.



### 12.4.3.5 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.

### 12.4.3.6 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.

### 12.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

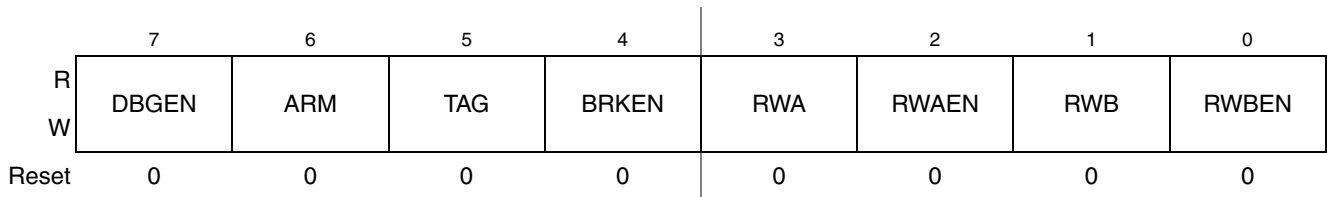


Figure 12-7. Debug Control Register (DBGC)

Table 12-4. DBGC Register Field Descriptions

Field	Description
7 DBGEN	<b>Debug Module Enable</b> — 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	<b>Arm Control</b> — 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	<b>Tag/Force Select</b> — 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	<b>Break Enable</b> — 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	<b>R/W Comparison Value for Comparator A</b> — 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	<b>Enable R/W for Comparator A</b> — 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	<b>R/W Comparison Value for Comparator B</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	<b>Enable R/W for Comparator B</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

### 12.4.3.8 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.

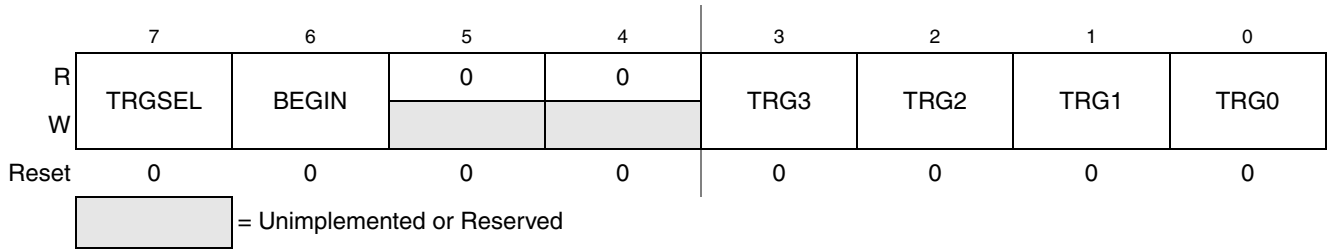


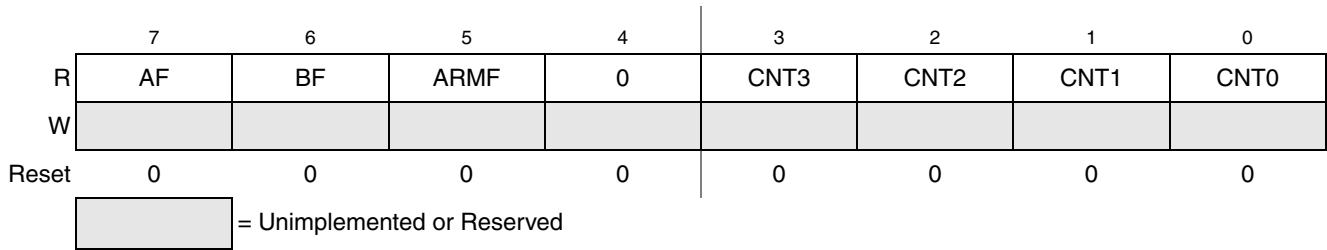
Figure 12-8. Debug Trigger Register (DBGT)

Table 12-5. DBGT Register Field Descriptions

Field	Description
7 TRGSEL	<b>Trigger Type</b> — 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	<b>Begin/End Trigger Select</b> — 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]	<b>Select Trigger Mode</b> — 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)

### 12.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.



**Figure 12-9. Debug Status Register (DBGS)**

**Table 12-6. DBGS Register Field Descriptions**

Field	Description
7 AF	<b>Trigger Match A Flag</b> — 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	<b>Trigger Match B Flag</b> — 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	<b>Arm Flag</b> — 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]	<b>FIFO Valid Count</b> — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO. 0000 Number of valid words in FIFO = No valid data 0001 Number of valid words in FIFO = 1 0010 Number of valid words in FIFO = 2 0011 Number of valid words in FIFO = 3 0100 Number of valid words in FIFO = 4 0101 Number of valid words in FIFO = 5 0110 Number of valid words in FIFO = 6 0111 Number of valid words in FIFO = 7 1000 Number of valid words in FIFO = 8

# Appendix A

## Electrical Characteristics and Timing Specifications

### A.1 Introduction

This section contains electrical and timing specifications.

### A.2 Parameter Classification

The electrical parameters shown in this supplement are guaranteed by various methods. To give the customer a better understanding, the following classification is used and the parameters are tagged accordingly in the tables where appropriate:

**Table A-1. Parameter Classifications**

<b>P</b>	Those parameters are guaranteed during production testing on each individual device.
<b>C</b>	Those parameters are achieved by the design characterization by measuring a statistically relevant sample size across process variations.
<b>T</b>	Those parameters are achieved by design characterization on a small sample size from typical devices under typical conditions unless otherwise noted. All values shown in the typical column are within this category.
<b>D</b>	Those parameters are derived mainly from simulations.

#### NOTE

The classification is shown in the column labeled “C” in the parameter tables where appropriate.

### A.3 Absolute Maximum Ratings

Absolute maximum ratings are stress ratings only, and functional operation at the maxima is not guaranteed. Stress beyond the limits specified in [Table A-2](#) may affect device reliability or cause permanent damage to the device. For functional operating conditions, refer to the remaining tables in this section.

This device contains circuitry protecting against damage due to high static voltage or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (for instance, either  $V_{SS}$  or  $V_{DD}$ ).

Table A-2. Absolute Maximum Ratings

Rating	Symbol	Value	Unit
Supply voltage	$V_{DD}$	-0.3 to 5.8	V
Input voltage	$V_{In}$	-0.3 to $V_{DD} + 0.3$	V
Instantaneous maximum current Single pin limit (applies to all port pins) <sup>1, 2, 3</sup>	$I_D$	±25	mA
Maximum current into $V_{DD}$	$I_{DD}$	120	mA
Storage temperature	$T_{stg}$	-55 to 150	°C
Maximum junction temperature	$T_J$	150	°C

- <sup>1</sup> Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive ( $V_{DD}$ ) and negative ( $V_{SS}$ ) clamp voltages, then use the larger of the two resistance values.
- <sup>2</sup> All functional non-supply pins are internally clamped to  $V_{SS}$  and  $V_{DD}$ .
- <sup>3</sup> Power supply must maintain regulation within operating  $V_{DD}$  range during instantaneous and operating maximum current conditions. If positive injection current ( $V_{In} > V_{DD}$ ) is greater than  $I_{DD}$ , the injection current may flow out of  $V_{DD}$  and could result in external power supply going out of regulation. Ensure external  $V_{DD}$  load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if the clock rate is very low which would reduce overall power consumption.

## A.4 Thermal Characteristics

This section provides information about operating temperature range, power dissipation, and package thermal resistance. Power dissipation on I/O pins is usually small compared to the power dissipation in on-chip logic and it is user-determined rather than being controlled by the MCU design. In order to take  $P_{I/O}$  into account in power calculations, determine the difference between actual pin voltage and  $V_{SS}$  or  $V_{DD}$  and multiply by the pin current for each I/O pin. Except in cases of unusually high pin current (heavy loads), the difference between pin voltage and  $V_{SS}$  or  $V_{DD}$  will be very small.

Table A-3. Thermal Characteristics

Rating	Symbol	Value	Unit
Operating temperature range (packaged)	$T_A$	$T_L$ to $T_H$ -40 to 85	°C
Thermal resistance <sup>1,2,3,4</sup> 44-pin LQFP	$\theta_{JA}$	1s 2s2p	°C/W
		73 56	
32-pin LQFP	$\theta_{JA}$	1s 2s2p	°C/W
		85 56	

- <sup>1</sup> Junction temperature is a function of die size, on-chip power dissipation, package thermal resistance, mounting site (board) temperature, ambient temperature, air flow, power dissipation of other components on the board, and board thermal resistance.
- <sup>2</sup> Junction to Ambient Natural Convection
- <sup>3</sup> 1s — Single layer board, one signal layer

<sup>4</sup> 2s2p — Four layer board, 2 signal and 2 power layers

The average chip-junction temperature ( $T_J$ ) in °C can be obtained from:

$$T_J = T_A + (P_D \times \theta_{JA}) \quad \text{Eqn. A-1}$$

where:

$T_A$  = Ambient temperature, °C

$\theta_{JA}$  = Package thermal resistance, junction-to-ambient, °C/W

$P_D = P_{int} + P_{I/O}$

$P_{int} = I_{DD} \times V_{DD}$ , Watts — chip internal power

$P_{I/O}$  = Power dissipation on input and output pins — user determined

For most applications,  $P_{I/O} \ll P_{int}$  and can be neglected. An approximate relationship between  $P_D$  and  $T_J$  (if  $P_{I/O}$  is neglected) is:

$$P_D = K \div (T_J + 273^\circ\text{C}) \quad \text{Eqn. A-2}$$

Solving equations 1 and 2 for K gives:

$$K = P_D \times (T_A + 273^\circ\text{C}) + \theta_{JA} \times (P_D)^2 \quad \text{Eqn. A-3}$$

where K is a constant pertaining to the particular part. K can be determined from [Equation A-3](#) by measuring  $P_D$  (at equilibrium) for a known  $T_A$ . Using this value of K, the values of  $P_D$  and  $T_J$  can be obtained by solving [Equation A-1](#) and [Equation A-2](#) iteratively for any value of  $T_A$ .

## A.5 DC Characteristics

This section includes information about power supply requirements, I/O pin characteristics, and power supply current in various operating modes.

**Table A-4. DC Characteristics**

Num	C	Parameter	Symbol	Min	Typ <sup>1</sup>	Max	Unit
1	P	Output high voltage — Low Drive (PTxDSn = 0) 5 V, $I_{Load} = -2$ mA	$V_{OH}$	$V_{DD} - 1.5$	—	—	V
		Output high voltage — High Drive (PTxDSn = 1) 5 V, $I_{Load} = -10$ mA		$V_{DD} - 1.5$	—	—	
2	P	Output low voltage — Low Drive (PTxDSn = 0) 5 V, $I_{Load} = -2$ mA	$V_{OL}$	1.5	—	—	V
		Output low voltage — High Drive (PTxDSn = 1) 5 V, $I_{Load} = -10$ mA		1.5	—	—	
3	P	Output high current — Max total $I_{OH}$ for all ports 5 V	$I_{OHT}$	—	—	100	mA
4	P	Output low current — Max total $I_{OL}$ for all ports 5 V	$I_{OLT}$	—	—	100	mA
5	P	Input high voltage; all digital inputs $2.7 \text{ V} \leq V_{DD} \leq 5.5 \text{ V}$	$V_{IH}$	$0.65 \times V_{DD}$	—	—	V
6	P	Input low voltage; all digital inputs	$V_{IL}$	—	—	$0.35 \times V_{DD}$	

Table A-4. DC Characteristics (continued)

Num	C	Parameter	Symbol	Min	Typ <sup>1</sup>	Max	Unit
7	P	Input hysteresis; all digital inputs	$V_{hys}$	$0.06 \times V_{DD}$			V
8	P	Input leakage current; input only pins <sup>2</sup>	$ I_{In} $	—	0.1	—	$\mu A$
9	P	High Impedance (off-state) leakage current <sup>2</sup>	$ I_{OZ} $	—	0.1	—	$\mu A$
10	P	Internal pullup resistors <sup>3</sup>	$R_{PU}$	—	45	—	$k\Omega$
11	P	Internal pulldown resistors <sup>4</sup>	$R_{PD}$	—	45	—	$k\Omega$
12	C	Input Capacitance; all non-supply pins	$C_{In}$	—	—	8	pF
13	P	POR rearm voltage	$V_{POR}$	—	1.4	—	V
14	D	POR rearm time	$t_{POR}$	10	—	—	$\mu s$
15	P	Low-voltage detection threshold — high range $V_{DD}$ falling $V_{DD}$ rising	$V_{LVDH}$	— —	4.3 4.4	—	V
17	P	Low-voltage warning threshold — high range $V_{DD}$ falling $V_{DD}$ rising	$V_{LVWH}$	— —	4.3 4.4	—	V
19	P	Low-voltage inhibit reset/recover hysteresis 5 V	$V_{hys}$	—	100	—	mV

<sup>1</sup> Typical values are based on characterization data at 25 °C unless otherwise stated.

<sup>2</sup> Measured with  $V_{In} = V_{DD}$  or  $V_{SS}$ .

<sup>3</sup> Measured with  $V_{In} = V_{SS}$ .

<sup>4</sup> Measured with  $V_{In} = V_{DD}$ .

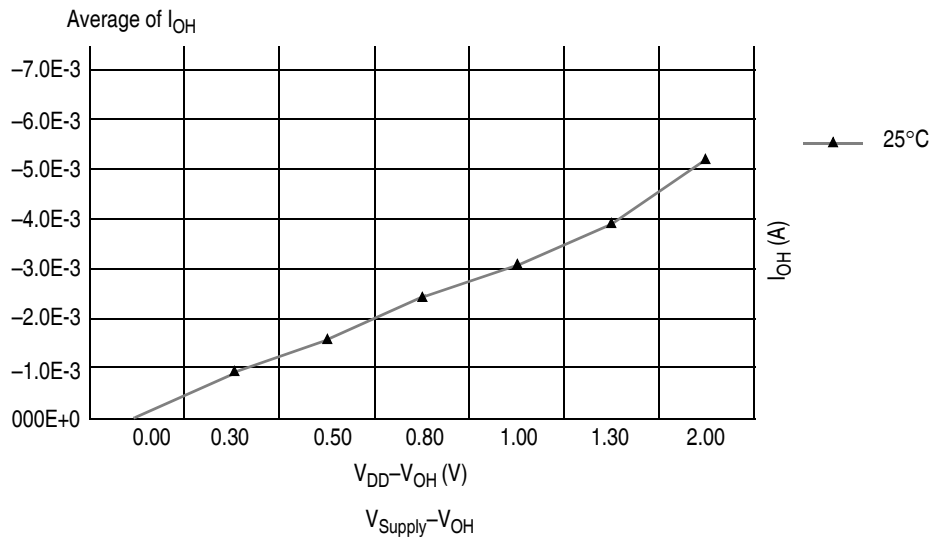
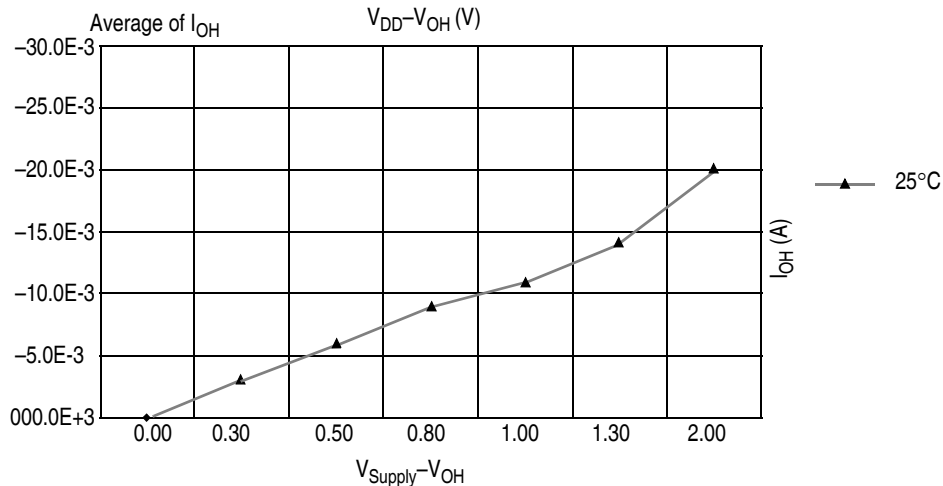


Figure A-1. Typical  $I_{OH}$  (Low Drive) vs  $V_{DD}-V_{OH}$  at  $V_{DD} = 5 V$



Figure A-2. Typical  $I_{OH}$  (High Drive) vs  $V_{DD} - V_{OH}$  at  $V_{DD} = 5$  V

## A.6 Supply Current Characteristics

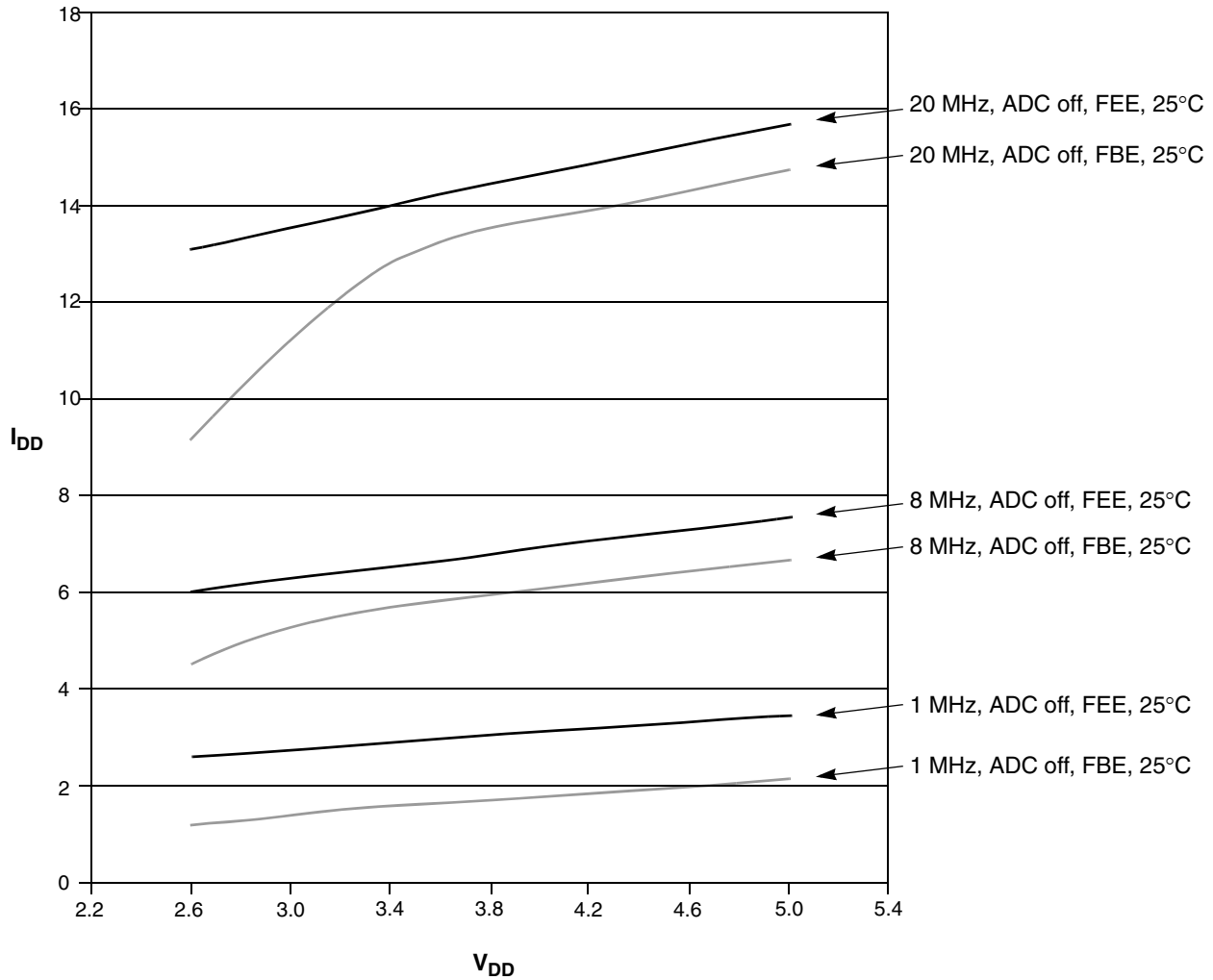
Table A-5. Supply Current Characteristics

Num	C	Parameter	Symbol	$V_{DD}$ (V)	Typical <sup>1</sup>	Max	Unit	Temp (°C)
1	C	Run supply current <sup>2</sup> (measured at CPU clock = 2 MHz, $f_{Bus} = 1$ MHz)	$R I_{DD}$	5	0.988	—	mA	-40 to 85°C
2	C	Run supply current <sup>2</sup> (measured at CPU clock = 16 MHz, $f_{Bus} = 8$ MHz)	$R I_{DD}$	5	6.84	—	mA	-40 to 85°C
3	C	Stop2 mode supply current	$S2 I_{DD}$	5	1.36	—	μA	-40 to 85°C
4	C	Stop3 mode supply current	$S3 I_{DD}$	5	1.49	—	μA	-40 to 85°C
5	C	RTI adder to stop2 or stop3	$S23 I_{DDRTI}$	5	300	—	nA	-40 to 85°C
6	C	LVD adder to stop3 (LVDE = LVDSE = 1)	$S3 I_{DDLVD}$	5	116	—	μA	-40 to 85°C
7	C	Adder to stop3 for oscillator enabled <sup>3</sup> (OSCSTEN = 1)	$S3 I_{DDOSC}$	5	5	—	μA	-40 to 85°C

<sup>1</sup> Typical values are based on characterization data at 25 °C unless otherwise stated. See Figure A-3 through Figure A-5 for typical curves across voltage/temperature.

<sup>2</sup> All modules except ADC active, ICG configured for FBE, and does not include any dc loads on port pins

<sup>3</sup> Values given under the following conditions: low range operation (RANGE = 0) with a 32.768 kHz crystal, low power mode (HGO = 0), clock monitor disabled (LOCD = 1).



Note: External clock is square wave supplied by function generator. For FEE mode, external reference frequency is 4 MHz

**Figure A-3. Typical Run I<sub>DD</sub> for FBE and FEE Modes, I<sub>DD</sub> vs. V<sub>DD</sub>**

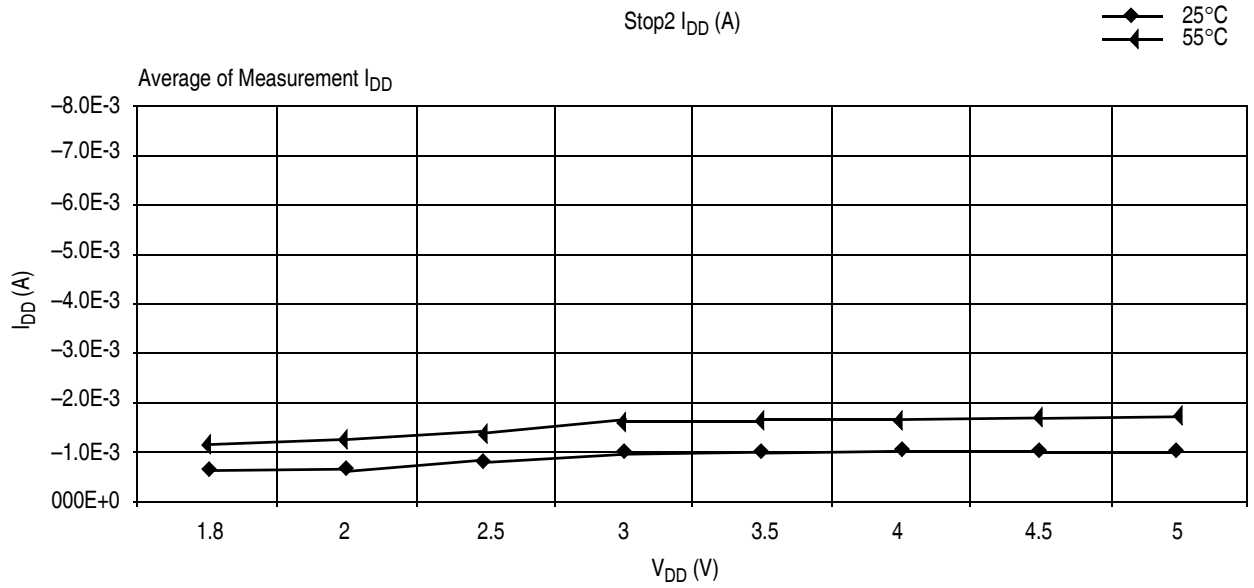


Figure A-4. Typical Stop 2  $I_{DD}$

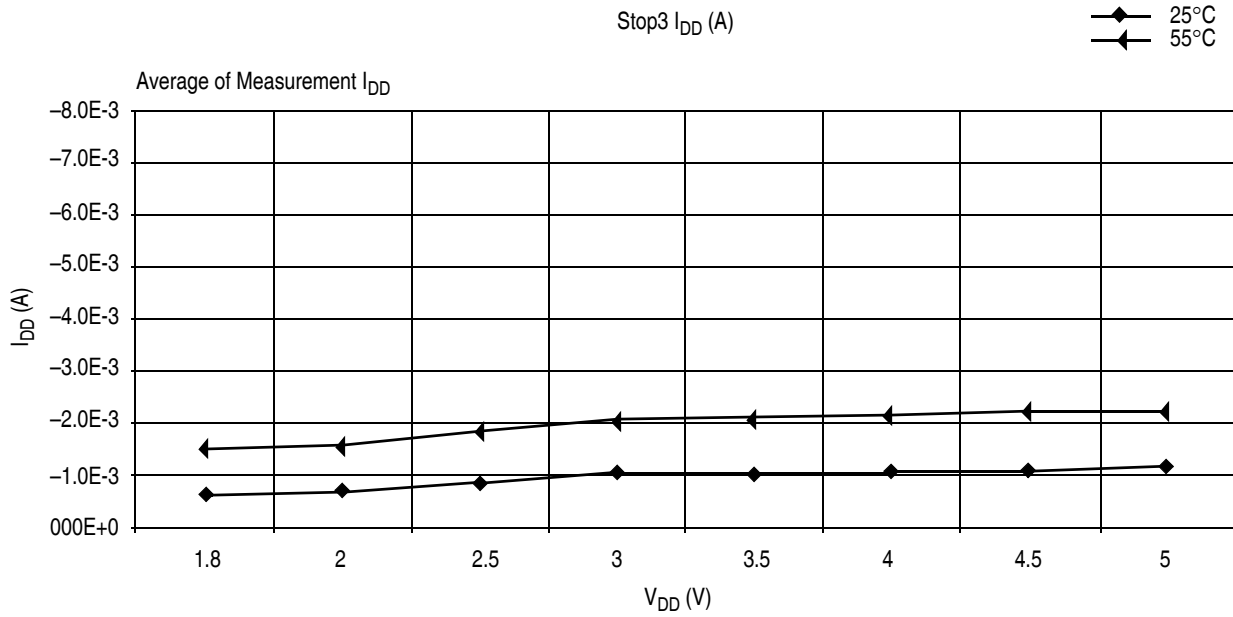


Figure A-5. Typical Stop3  $I_{DD}$

## A.7 ADC Characteristics

Table A-6. 8-bit ADC Operating Conditions

Characteristic	Conditions	Symb	Min	Typical <sup>1</sup>	Max	Unit
Supply voltage	Absolute	$V_{DDAD}$	2.7	—	5.5	V
	Delta to $V_{DD}$ ( $V_{DD} - V_{DDAD}$ ) <sup>2</sup>	$\Delta V_{DDAD}$	-100	0	100	mV
Ground voltage	Delta to $V_{SS}$ ( $V_{SS} - V_{SSAD}$ ) <sup>2</sup>	$\Delta V_{SSAD}$	-100	0	100	mV
Ref voltage high	—	$V_{REFH}$	2.7	$V_{DDAD}$	$V_{DDAD}$	V
Ref voltage low	—	$V_{REFL}$	$V_{SSAD}$	$V_{SSAD}$	$V_{SSAD}$	V
Supply current	Stop, reset, module off	$I_{DDAD}$	—	0.011	1	$\mu$ A
Input voltage	—	$V_{ADIN}$	$V_{REFL}$	—	$V_{REFH}$	V
Input capacitance	—	$C_{ADIN}$	—	4.5	—	pF
Input resistance	—	$R_{ADIN}$	—	3	—	k $\Omega$
Analog source resistance — external to MCU	8-bit mode (all valid $f_{ADCK}$ )	$R_{AS}$	—	—	10	k $\Omega$
ADC conversion clock frequency	High speed (ADLPC = 0)	$f_{ADCK}$	0.4	—	8.0	MHz
	Low power (ADLPC = 1)		0.4	—	4.0	

<sup>1</sup> Typical values assume  $V_{DDAD} = 5.0$  V, Temp = 25 °C,  $f_{ADCK} = 1.0$  MHz unless otherwise stated. Typical values are for reference only and are not tested in production.

<sup>2</sup> DC potential difference.

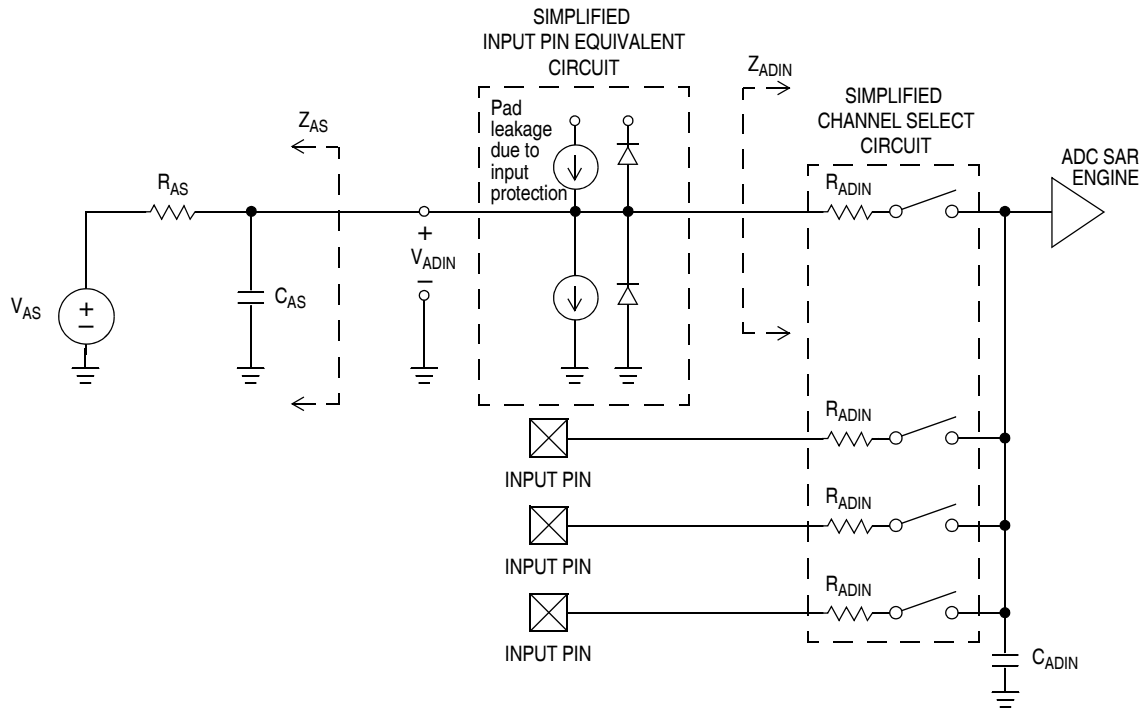


Figure A-6. ADC Input Impedance Equivalency Diagram

Table A-7. 8-bit ADC Characteristics ( $V_{REFH} = V_{DDAD}$ ,  $V_{REFL} = V_{SSAD}$ )

C	Characteristic	Conditions	Symb	Typical <sup>1</sup>	Unit
T	Supply current ADLPC = 1 ADLSMP = 1 ADCO = 1		$I_{DDAD}$	133	$\mu\text{A}$
T	Supply current ADLPC = 1 ADLSMP = 0 ADCO = 1		$I_{DDAD}$	218	$\mu\text{A}$
T	Supply current ADLPC = 0 ADLSMP = 1 ADCO = 1		$I_{DDAD}$	327	$\mu\text{A}$
T	Supply current ADLPC = 0 ADLSMP = 0 ADCO = 1		$I_{DDAD}$	582	$\mu\text{A}$
P	ADC asynchronous clock source $t_{ADACK} = 1/f_{ADACK}$	High speed (ADLPC = 0)	$f_{ADACK}$	3.3	MHz
		Low power (ADLPC = 1)		2	
P	Conversion time (Including sample time)	Short sample (ADLSMP = 0)	$t_{ADC}$	20	ADCK cycles
		Long sample (ADLSMP = 1)		40	

Table A-7. 8-bit ADC Characteristics ( $V_{REFH} = V_{DDAD}$ ,  $V_{REFL} = V_{SSAD}$ ) (continued)

C	Characteristic	Conditions	Symb	Typical <sup>1</sup>	Unit
P	Sample time	Short sample (ADLSMP = 0)	$t_{ADS}$	3.5	ADCK cycles
		Long sample (ADLSMP = 1)		23.5	
P	Total unadjusted error Includes quantization	8-bit mode	$E_{TUE}$	$\pm 0.5$	LSB <sup>2</sup>
P	Differential non-linearity	8-bit mode	DNL	$\pm 0.3$	LSB <sup>2</sup>
		Monotonicity and no-missing-codes guaranteed			
C	Integral non-linearity	8-bit mode	INL	$\pm 0.3$	LSB <sup>2</sup>
P	Zero-scale error $V_{ADIN} = V_{SSA}$	8-bit mode	$E_{ZS}$	$\pm 0.5$	LSB <sup>2</sup>
P	Full-scale error $V_{ADIN} = V_{DDA}$	8-bit mode	$E_{FS}$	$\pm 0.5$	LSB <sup>2</sup>
D	Quantization error	8-bit mode	$E_Q$	$\pm 0.5$	LSB <sup>2</sup>
D	Input leakage error Pad leakage <sup>3</sup> * $R_{AS}$	8-bit mode	$E_{IL}$	$\pm 0.1$	LSB <sup>2</sup>
D	Temp sensor slope	-40 °C – 25 °C	m	3.266	mV/xC
		25 °C – 85 °C		3.636	
D	Temp sensor voltage	25 °C	$V_{TEMP25}$	1.41	mV

<sup>1</sup> Typical values assume  $V_{DDAD} = 5.0$  V, Temp = 25C,  $f_{ADCK} = 1.0$  MHz unless otherwise stated. Typical values are for reference only and are not tested in production.

<sup>2</sup> 1 LSB =  $(V_{REFH} - V_{REFL})/2^N$

<sup>3</sup> Based on input pad leakage current. Refer to pad electricals.

## A.8 Internal Clock Generation Module Characteristics

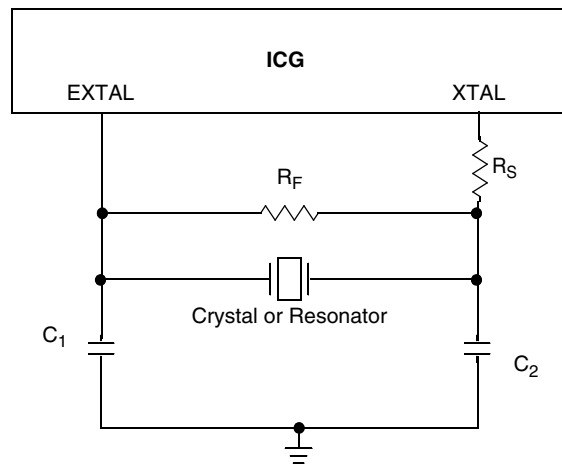


Table A-8. ICG DC Electrical Specifications (Temperature Range = -40 to 85°C Ambient)

Characteristic	Symbol	Typical <sup>1</sup>	Unit
Load capacitors	C <sub>1</sub> C <sub>2</sub>	See Note <sup>2</sup>	
Feedback resistor Low range (32k to 100 kHz) High range (1M – 16 MHz)	R <sub>F</sub>	10 1	MΩ
Series resistor Low range Low Gain (HGO = 0) High Gain (HGO = 1) High range Low Gain (HGO = 0) High Gain (HGO = 1) ≥ 8 MHz 4 MHz 1 MHz	R <sub>S</sub>	0 100 0 0 10 20	kΩ

<sup>1</sup> Typical values are based on characterization data at V<sub>DD</sub> = 5.0 V, 25 °C or is typical recommended value.

<sup>2</sup> See crystal or resonator manufacturer's recommendation.

## A.8.1 ICG Frequency Specifications

Table A-9. ICG Frequency Specifications

(V<sub>DDA</sub> = V<sub>DDA</sub> (min) to V<sub>DDA</sub> (max), Temperature Range = -40 to 85°C Ambient)

Num	C	Characteristic	Symbol	Min	Typical <sup>1</sup>	Max	Unit
1	C	Oscillator crystal or resonator (REFS = 1) (Fundamental mode crystal or ceramic resonator) Low range	f <sub>lo</sub>	32		100	kHz
		High range	f <sub>hi_byp</sub>	1	—	16	MHz
		High Gain, FBE (HGO = 1, CLKS = 10)	f <sub>hi_eng</sub>	2	—	10	MHz
		High Gain, FEE (HGO = 1, CLKS = 11)	f <sub>lp_byp</sub>	1	—	8	MHz
		Low Power, FBE (HGO = 0, CLKS = 10) Low Power, FEE (HGO = 0, CLKS = 11)	f <sub>lp_eng</sub>	2	—	8	MHz
2	C	Input clock frequency (CLKS = 11, REFS = 0) Low range	f <sub>lo</sub>	32	—	100	kHz
		High range	f <sub>hi_eng</sub>	2	—	10	MHz
3	C	Input clock frequency (CLKS = 10, REFS = 0)	f <sub>Extal</sub>	0	—	40	MHz
4	C	Internal reference frequency (untrimmed)	f <sub>ICGIRCLK</sub>	182.25	243	303.75	kHz
5	C	Duty cycle of input clock (REFS = 0)	t <sub>dc</sub>	40	—	60	%
6	C	Output clock ICGOUT frequency CLKS = 10, REFS = 0 All other cases	f <sub>ICGOUT</sub>	f <sub>Extal</sub> (min) f <sub>lo</sub> (min)	— —	f <sub>Extal</sub> (max) f <sub>ICGDCLKmax</sub> (max)	MHz
7	C	Minimum DCO clock (ICGDCLK) frequency	f <sub>ICGDCLKmin</sub>	8	—		MHz
8	C	Maximum DCO clock (ICGDCLK) frequency	f <sub>ICGDCLKmax</sub>		—	40	MHz
9	C	Self-clock mode (ICGOUT) frequency <sup>2</sup>	f <sub>Self</sub>	f <sub>ICGDCLKmin</sub>		f <sub>ICGDCLKmax</sub>	MHz

**Table A-9. ICG Frequency Specifications (continued)**  
**( $V_{DDA} = V_{DDA}(\text{min})$  to  $V_{DDA}(\text{max})$ , Temperature Range =  $-40$  to  $85^{\circ}\text{C}$  Ambient)**

Num	C	Characteristic	Symbol	Min	Typical <sup>1</sup>	Max	Unit
10	C	Self-clock mode reset (ICGOUT) frequency	$f_{\text{Self\_reset}}$	5.5	8	10.5	MHz
11	C	Loss of reference frequency <sup>3</sup> Low range High range	$f_{\text{LOR}}$	5 50		25 500	kHz
12	C	Loss of DCO frequency <sup>4</sup>	$f_{\text{LOD}}$	0.5		1.5	MHz
13	C	Crystal start-up time <sup>5, 6</sup> Low range High range	$t_{\text{CSTL}}$ $t_{\text{CSTH}}$	— —	430 4	— —	ms
14	C	FLL lock time <sup>7</sup> Low range High range	$t_{\text{Lockl}}$ $t_{\text{Lockh}}$	— —		2 2	ms
15	C	FLL frequency unlock range	$n_{\text{Unlock}}$	$-4*N$		$4*N$	counts
16	C	FLL frequency lock range	$n_{\text{Lock}}$	$-2*N$		$2*N$	counts
17	C	ICGOUT period jitter, <sup>8</sup> measured at $f_{\text{ICGOUT Max}}$ Long term jitter (averaged over 2 ms interval)	$C_{\text{Jitter}}$	—		0.2	% $f_{\text{ICG}}$
18	C	Internal oscillator deviation from trimmed frequency <sup>9</sup> $V_{\text{DD}} = 2.7 - 5.5$ V, (constant temperature) $V_{\text{DD}} = 5.0$ V $\pm 10\%$ , $20^{\circ}\text{C}$ to $85^{\circ}\text{C}$	$\text{ACC}_{\text{int}}$	— —	$\pm 0.5$ $\pm 0.5$	— —	%

<sup>1</sup> Typical values are based on characterization data at  $V_{\text{DD}} = 5.0$  V,  $25^{\circ}\text{C}$  unless otherwise stated.

<sup>2</sup> Self-clocked mode frequency is the frequency that the DCO generates when the FLL is open-loop.

<sup>3</sup> Loss of reference frequency is the reference frequency detected internally, which transitions the ICG into self-clocked mode if it is not in the desired range.

<sup>4</sup> Loss of DCO frequency is the DCO frequency detected internally, which transitions the ICG into FLL bypassed external mode (if an external reference exists) if it is not in the desired range.

<sup>5</sup> This parameter is characterized before qualification rather than 100% tested.

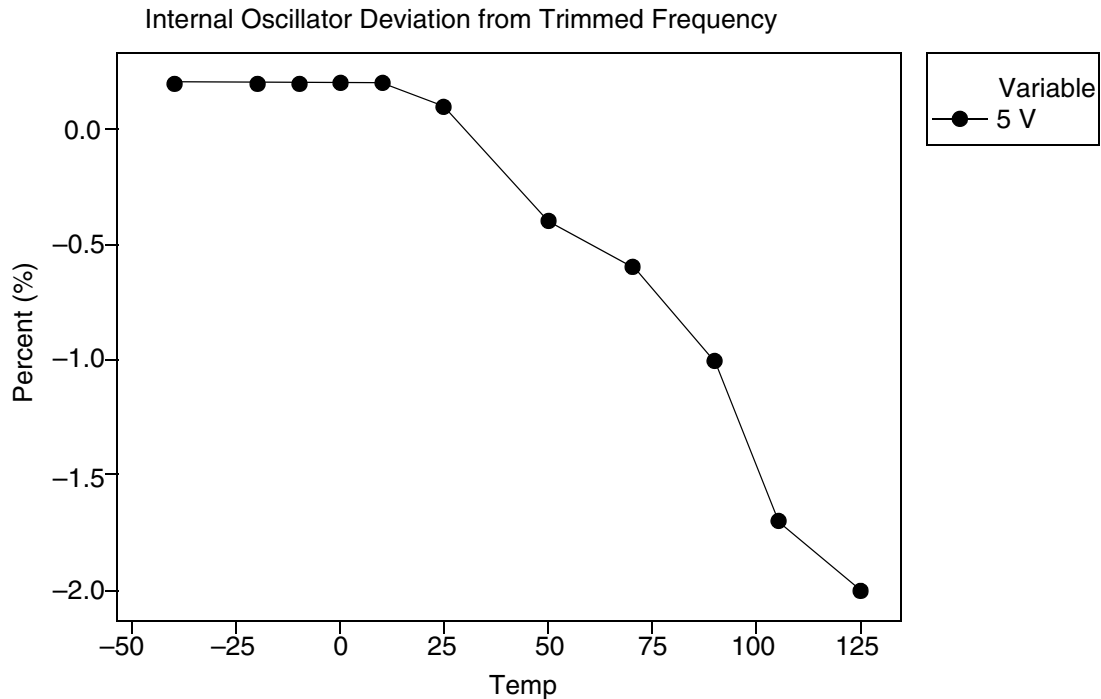
<sup>6</sup> Proper PC board layout procedures must be followed to achieve specifications.

<sup>7</sup> This specification applies to the period of time required for the FLL to lock after entering FLL engaged internal or external modes. If a crystal/resonator is being used as the reference, this specification assumes it is already running.

<sup>8</sup> Jitter is the average deviation from the programmed frequency measured over the specified interval at maximum  $f_{\text{ICGOUT}}$ . Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the FLL circuitry via  $V_{\text{DDA}}$  and  $V_{\text{SSA}}$  and variation in crystal oscillator frequency increase the  $C_{\text{Jitter}}$  percentage for a given interval.

<sup>9</sup> See [Figure A-7](#).





Device trimmed at 25°C at 3.0 V.

**Figure A-7. Internal Oscillator Deviation from Trimmed Frequency**

## A.9 AC Characteristics

This section describes ac timing characteristics for each peripheral system. For detailed information about how clocks for the bus are generated, see [Chapter 8, “Internal Clock Generator \(S08ICGV4\).”](#)

### A.9.1 Control Timing

Table A-10. Control Timing

Num	Parameter	Symbol	Min	Typical <sup>1</sup>	Max	Unit
1	Bus frequency ( $t_{cyc} = 1/f_{Bus}$ )	$f_{Bus}$	DC	—	10	MHz
2	Real-time interrupt internal oscillator period	$t_{RTI}$	700	—	1300	$\mu s$
3	External reset pulse width <sup>2</sup> ( $t_{cyc} = 1/f_{Self\_reset}$ )	$t_{extrst}$	$1.5 \times t_{Self\_reset}$	—	—	ns
4	Reset low drive <sup>3</sup>	$t_{rstdrv}$	$34 \times t_{cyc}$	—	—	ns
5	Active background debug mode latch setup time	$t_{MSSU}$	25	—	—	ns
6	Active background debug mode latch hold time	$t_{MSH}$	25	—	—	ns
7	IRQ pulse width Asynchronous path <sup>2</sup> Synchronous path <sup>4</sup>	$t_{ILIH}, t_{IHIL}$	100 $1.5 \times t_{cyc}$	—	—	ns
9	Port rise and fall time (load = 50 pF) <sup>5</sup> Slew rate control disabled (PTxSE = 0) Slew rate control enabled (PTxSE = 1)	$t_{Rise}, t_{Fall}$	— —	3 30	—	ns

- <sup>1</sup> Typical values are based on characterization data at  $V_{DD} = 5.0 V$ , 25 °C unless otherwise stated.
- <sup>2</sup> This is the shortest pulse that is guaranteed to be recognized as a reset pin request. Shorter pulses are not guaranteed to override reset requests from internal sources.
- <sup>3</sup> When any reset is initiated, internal circuitry drives the reset pin low for about 34 bus cycles and then samples the level on the reset pin about 38 bus cycles later to distinguish external reset requests from internal requests.
- <sup>4</sup> This is the minimum pulse width that is guaranteed to pass through the pin synchronization circuitry. Shorter pulses may or may not be recognized. In stop mode, the synchronizer is bypassed so shorter pulses can be recognized in that case.
- <sup>5</sup> Timing is shown with respect to 20%  $V_{DD}$  and 80%  $V_{DD}$  levels. Temperature range -40°C to 85°C.

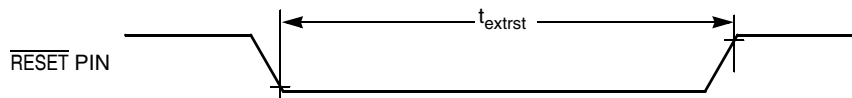


Figure A-8. Reset Timing

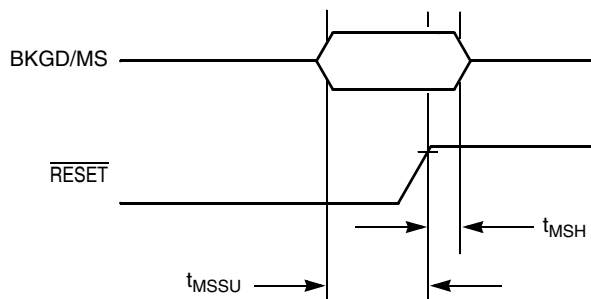


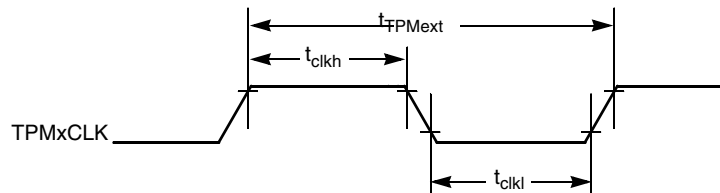
Figure A-9. Active Background Debug Mode Latch Timing

## A.9.2 Timer/PWM (TPM) Module Timing

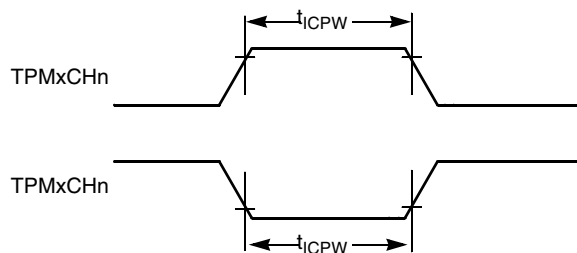
Synchronizer circuits determine the shortest input pulses that can be recognized or the fastest clock that can be used as the optional external source to the timer counter. These synchronizers operate from the current bus rate clock.

**Table A-11. TPM Input Timing**

Function	Symbol	Min	Max	Unit
External clock frequency	$f_{\text{TPMext}}$	dc	$f_{\text{Bus}}/4$	MHz
External clock period	$t_{\text{TPMext}}$	4	—	$t_{\text{cyc}}$
External clock high time	$t_{\text{clkh}}$	1.5	—	$t_{\text{cyc}}$
External clock low time	$t_{\text{clkl}}$	1.5	—	$t_{\text{cyc}}$
Input capture pulse width	$t_{\text{ICPW}}$	1.5	—	$t_{\text{cyc}}$



**Figure A-10. Timer External Clock**



**Figure A-11. Timer Input Capture Pulse**

## A.10 Flash Specifications

This section provides details about program/erase times and program-erase endurance for the flash memory.

Program and erase operations do not require any special power sources other than the normal  $V_{\text{DD}}$  supply. For more detailed information about program/erase operations, see [Chapter 4, “Memory.”](#)

Table A-12. Flash Characteristics

Num	C	Characteristic	Symbol	Min	Typical <sup>1</sup>	Max	Unit
1	D	Supply voltage for program/erase	$V_{\text{prog/erase}}$	2.7	—	5.5	V
2	D	Supply voltage for read operation	$V_{\text{Read}}$	2.7	—	5.5	V
3	D	Internal FCLK frequency <sup>2</sup>	$f_{\text{FCLK}}$	150	—	200	kHz
4	D	Internal FCLK period (1/FCLK)	$t_{\text{FcyC}}$	5	—	6.67	$\mu\text{s}$
5	D	Byte program time (random location) <sup>(2)</sup>	$t_{\text{prog}}$	9			$t_{\text{FcyC}}$
6	D	Byte program time (burst mode) <sup>(2)</sup>	$t_{\text{Burst}}$	4			$t_{\text{FcyC}}$
7	D	Page erase time <sup>3</sup>	$t_{\text{Page}}$	4000			$t_{\text{FcyC}}$
8	D	Mass erase time <sup>(2)</sup>	$t_{\text{Mass}}$	20,000			$t_{\text{FcyC}}$
9	D	Program/erase endurance <sup>4</sup> $T_L$ to $T_H = -40^\circ\text{C}$ to $85^\circ\text{C}$ $T = 25^\circ\text{C}$	—	10,000 —	— 100,000	—	cycles
10	D	Data retention <sup>5</sup>	$t_{\text{D\_ret}}$	15	100	—	years

<sup>1</sup> Typical values are based on characterization data at  $V_{\text{DD}} = 5.0 \text{ V}$ ,  $25^\circ\text{C}$  unless otherwise stated.

<sup>2</sup> The frequency of this clock is controlled by a software setting.

<sup>3</sup> These values are hardware state machine controlled. User code does not need to count cycles. This information supplied for calculating approximate time to program and erase.

<sup>4</sup> Typical endurance for flash was evaluated for this product family on the 9S12Dx64. For additional information on how Freescale Semiconductor defines typical endurance, please refer to Engineering Bulletin EB619, *Typical Endurance for Nonvolatile Memory*.

<sup>5</sup> Typical data retention values are based on intrinsic capability of the technology measured at high temperature and de-rated to  $25^\circ\text{C}$  using the Arrhenius equation. For additional information on how Freescale Semiconductor defines typical data retention, please refer to Engineering Bulletin EB618, *Typical Data Retention for Nonvolatile Memory*.

## A.11 EMC Performance

Electromagnetic compatibility (EMC) performance is highly dependant on the environment in which the MCU resides. Board design and layout, circuit topology choices, location and characteristics of external components as well as MCU software operation all play a significant role in EMC performance. The system designer should consult Freescale applications notes such as AN2321, AN1050, AN1263, AN2764, and AN1259 for advice and guidance specifically targeted at optimizing EMC performance.

# Appendix B

## Ordering Information and Mechanical Drawings

### B.1 Ordering Information

This section contains ordering information for the SC9S08MZ16 devices. See below for an example of the device numbering system.

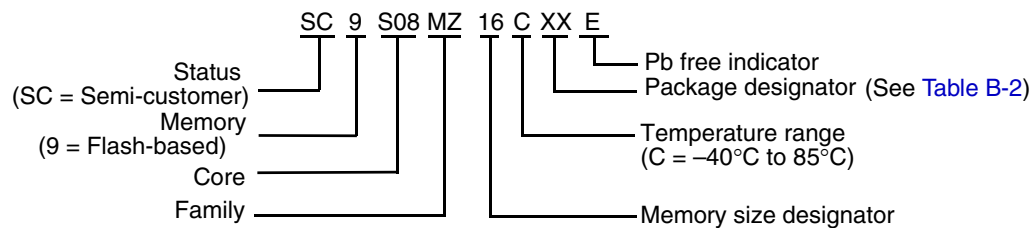
**Table B-1. Device Comparison**

Device Number <sup>1</sup>	Memory		Available Packages <sup>2</sup> Type
	Flash	RAM	
SC9S08MZ16	16,384	1024	44-pin LQFP 32-pin LQFP

<sup>1</sup> See [Table 1-1](#) for a complete description of modules included on each device.

<sup>2</sup> See [Table B-2](#) for package information.

The orderable part numbers for the SC9S08MZ16 device vary according to the device family. Refer to [Figure B-1](#) for examples.



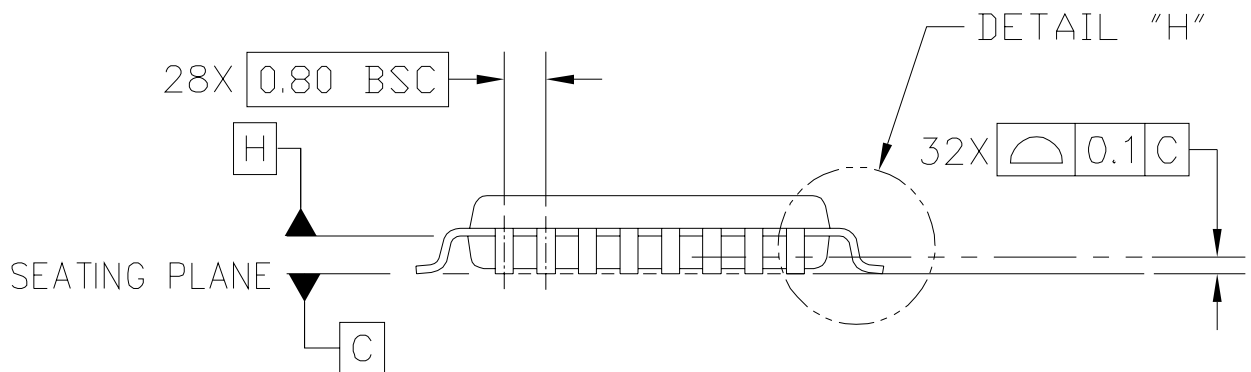
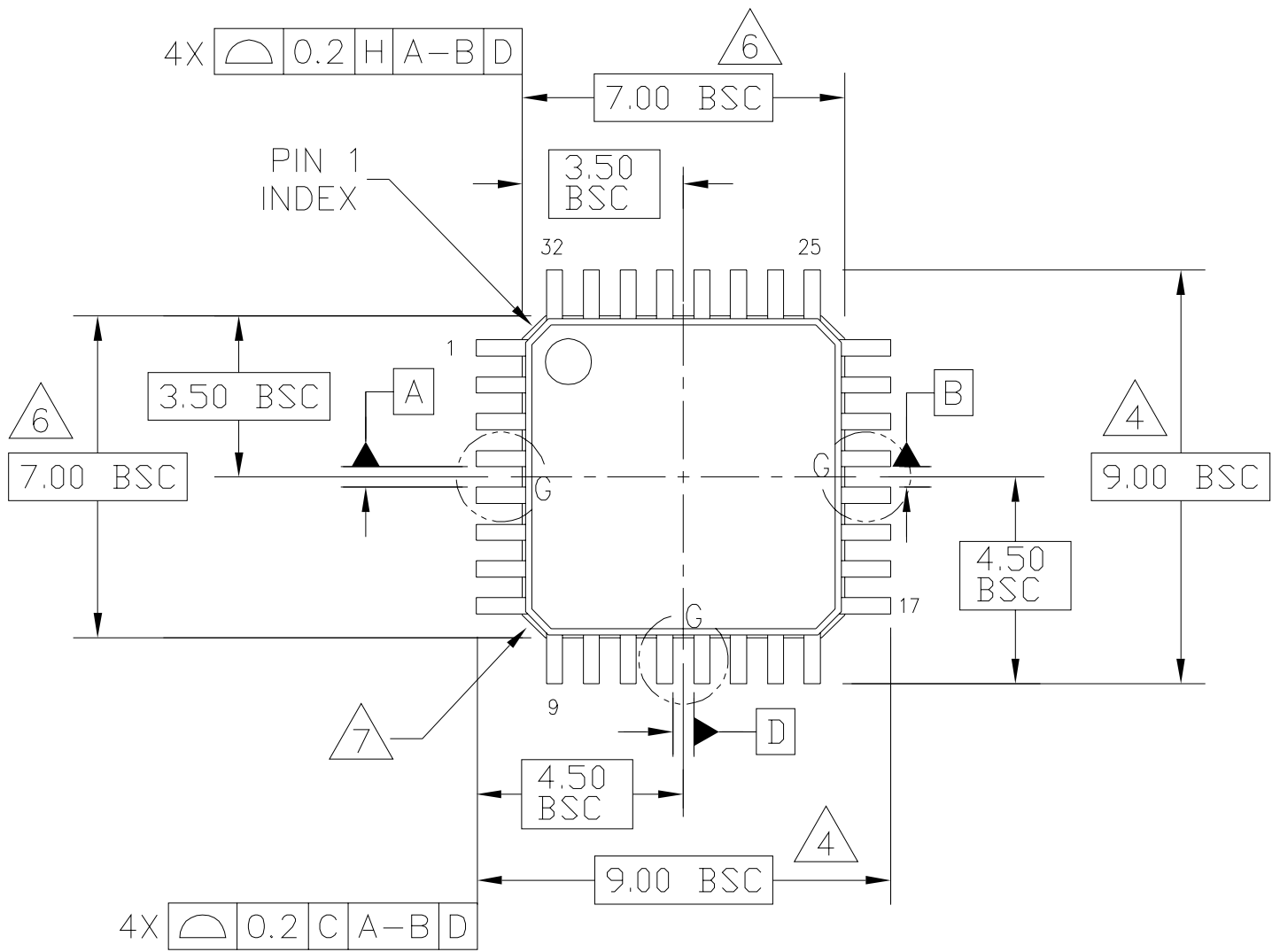
**Figure B-1. SC9S08MZ16 Series Device Numbering System**

### B.2 Mechanical Drawings

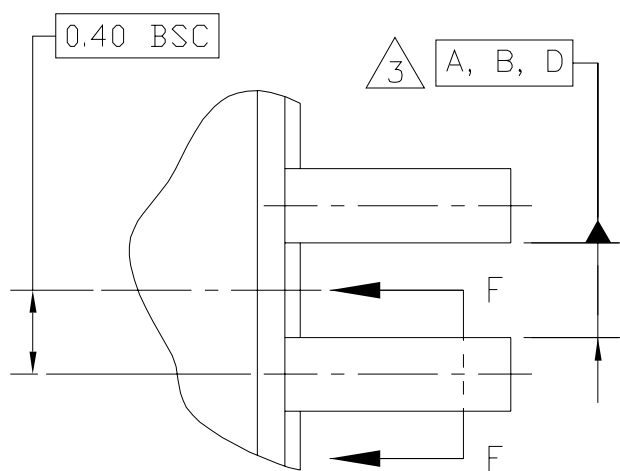
The following pages contain mechanical specifications for SC9S08MZ16 package options. See the following tables for the document numbers that correspond to each package type.

**Table B-2. SC9S08MZ16 Consumer & Industrial Package Information**

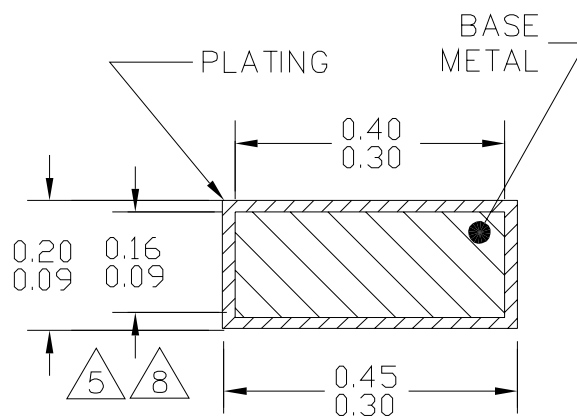
Pin Count	Type	Designator	Document No.
44	LQFP	FG	98ASS23225W
32	LQFP	FJ	98ASH70029A



© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: LOW PROFILE QUAD FLAT PACK (LQFP) 32 LEAD, 0.8 PITCH (7 X 7 X 1.4)	DOCUMENT NO: 98ASH70029A	REV: D	
	CASE NUMBER: 873A-03	19 MAY 2005	
	STANDARD: JEDEC MS-026 BBA		

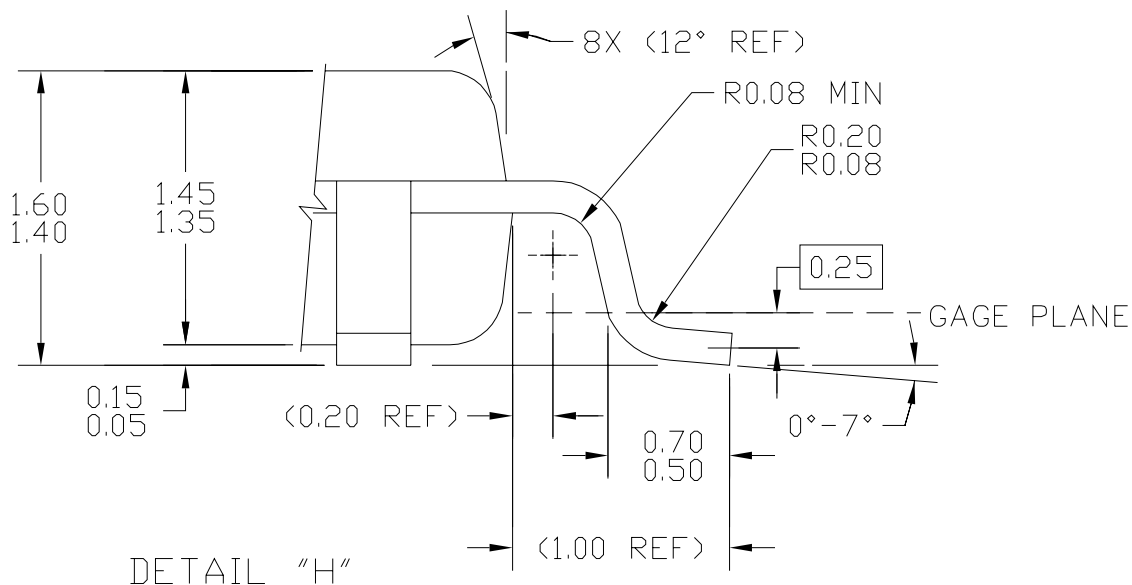


DETAIL G



⊕ 0.2 (M) C A-B D

SECTION F-F  
ROTATED 90°CW  
32 PLACES



DETAIL "H"

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: LOW PROFILE QUAD FLAT PACK (LQFP) 32 LEAD, 0.8 PITCH (7 X 7 X 1.4)	DOCUMENT NO: 98ASH70029A	REV: D	
	CASE NUMBER: 873A-03	19 MAY 2005	
	STANDARD: JEDEC MS-026 BBA		

NOTES:

1. DIMENSIONS ARE IN MILLIMETERS.

2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5-1994.

3. DATUMS A, B, AND D TO BE DETERMINED AT DATUM PLANE H.

4. DIMENSIONS TO BE DETERMINED AT SEATING PLANE DATUM C.

5. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM DIMENSION BY MORE THAN 0.08 MM. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND ADJACENT LEAD OR PROTRUSION: 0.07 MM.

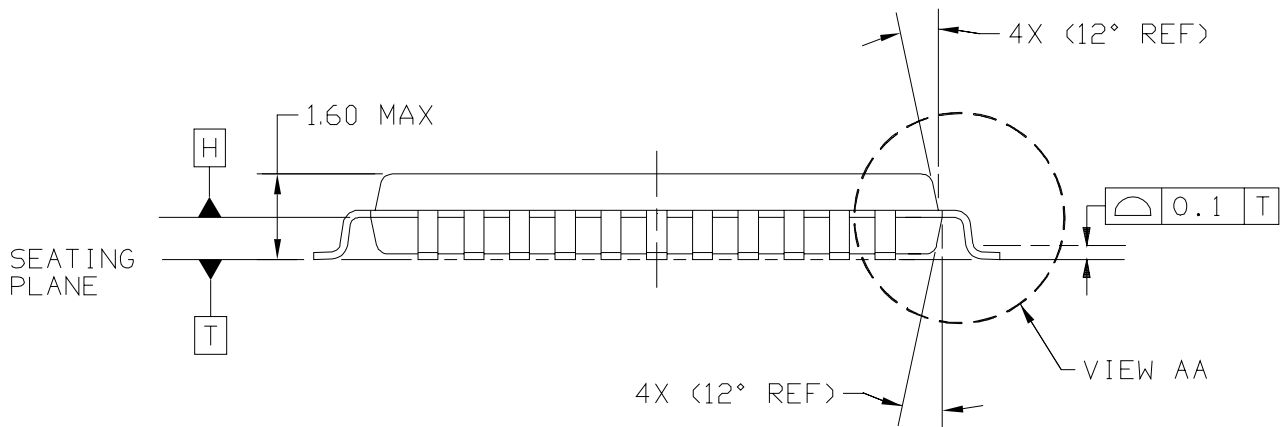
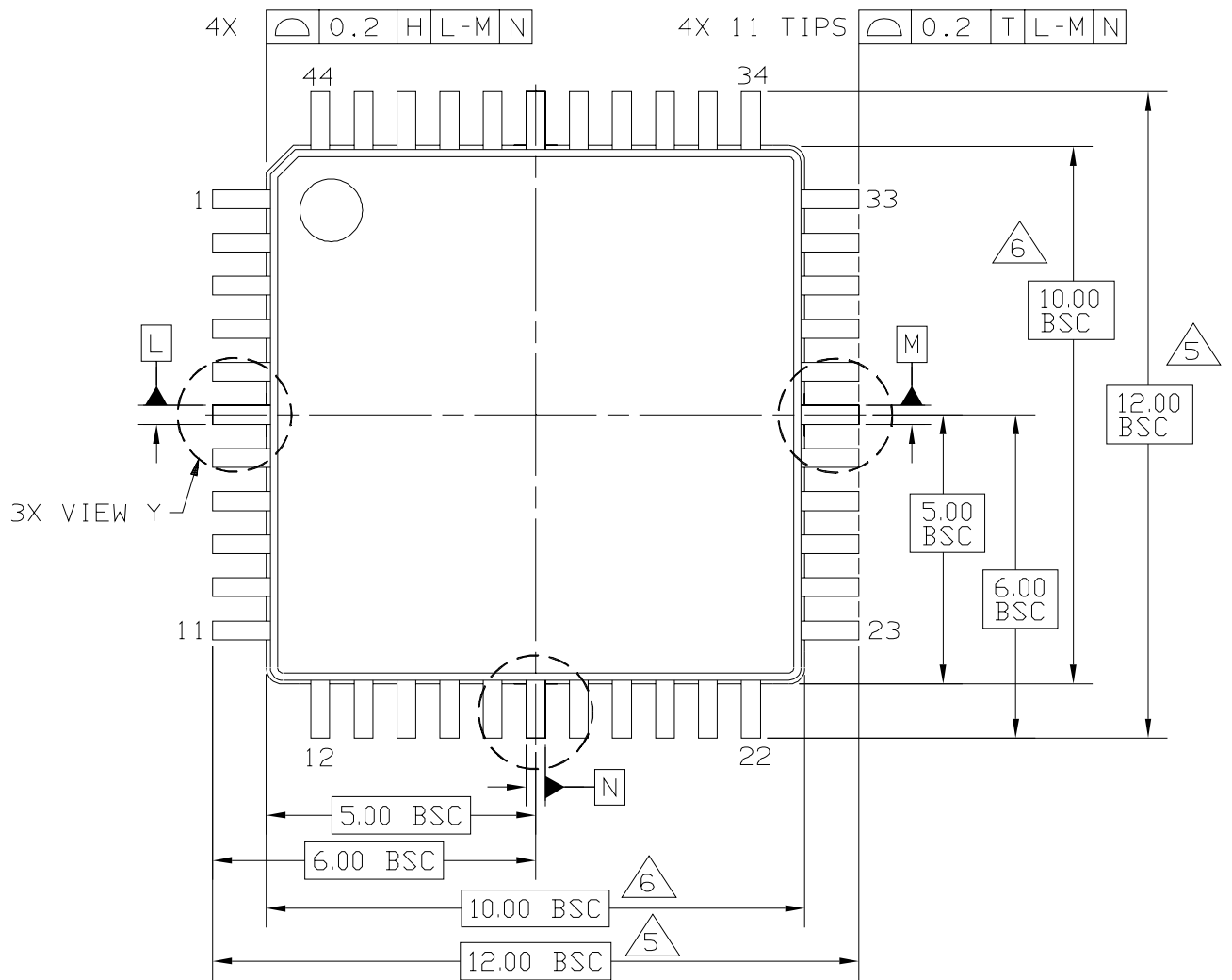
6. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 MM PER SIDE. DIMENSIONS ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.

7. EXACT SHAPE OF EACH CORNER IS OPTIONAL.

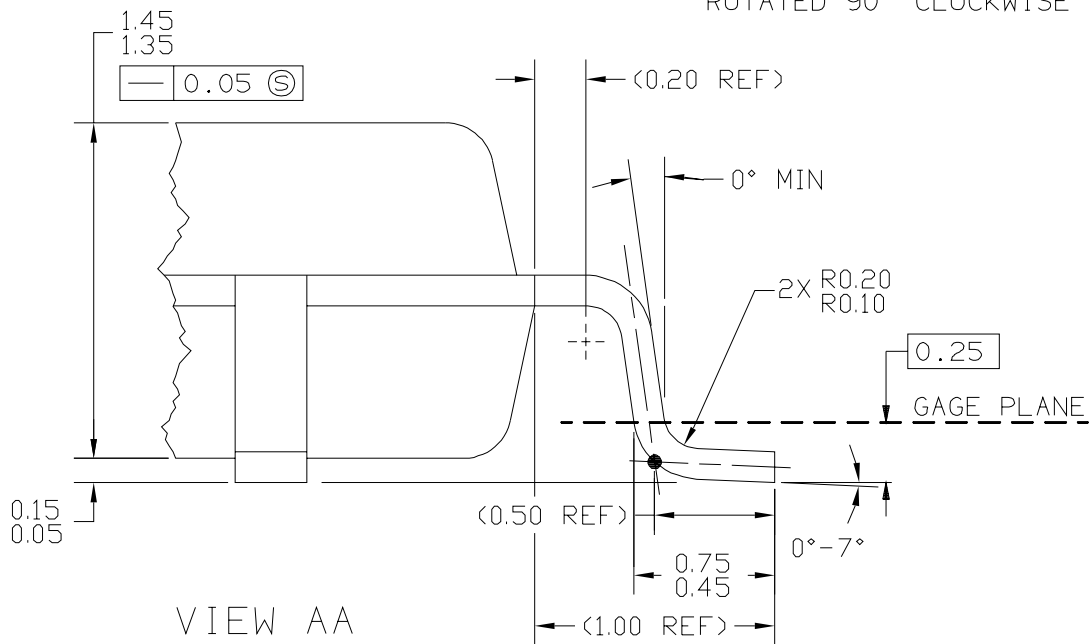
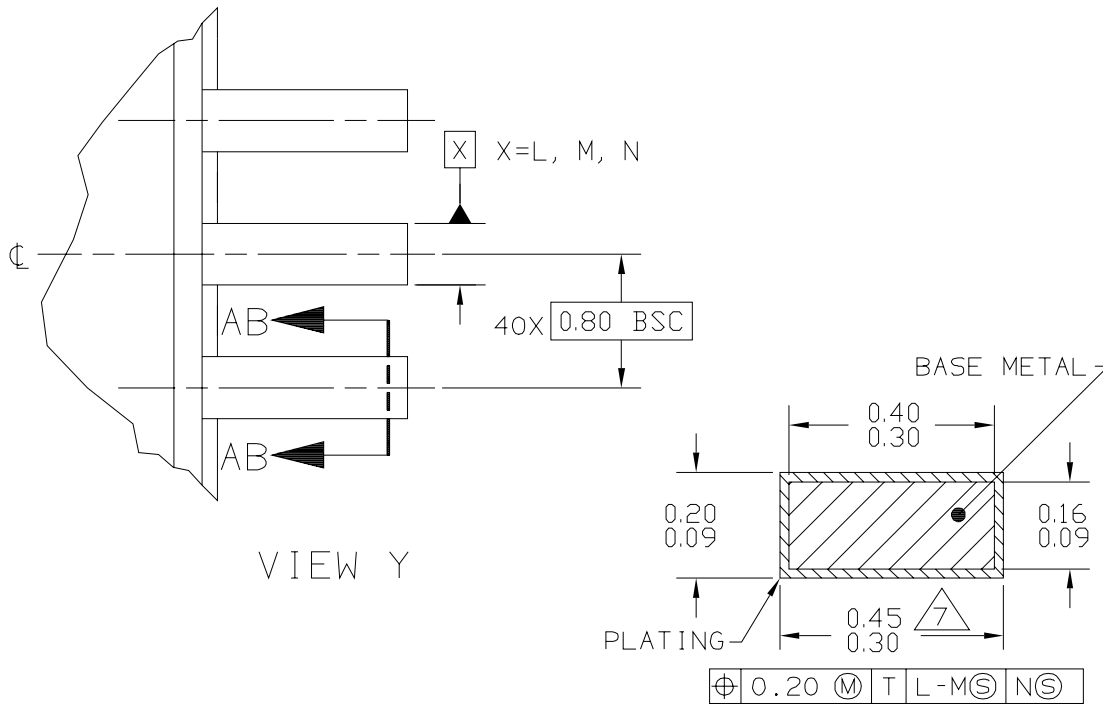
8. THESE DIMENSIONS APPLY TO THE FLAT SECTION OF THE LEAD BETWEEN 0.1 MM AND 0.25 MM FROM THE LEAD TIP.

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: LOW PROFILE QUAD FLAT PACK (LQFP) 32 LEAD, 0.8 PITCH (7 X 7 X 1.4)	DOCUMENT NO: 98ASH70029A	REV: D	
	CASE NUMBER: 873A-03	19 MAY 2005	
	STANDARD: JEDEC MS-026 BBA		





© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: 44 LD LQFP, 10 X 10 PKG, 0.8 PITCH, 1.4 THICK	DOCUMENT NO: 98ASS23225W	REV: D	
	CASE NUMBER: 824D-02	26 FEB 2007	
	STANDARD: JEDEC MS-026-BCB		



© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: 44 LD LQFP, 10 X 10 PKG, 0.8 PITCH, 1.4 THICK	DOCUMENT NO: 98ASS23225W	REV: D	
	CASE NUMBER: 824D-02	26 FEB 2007	
	STANDARD: JEDEC MS-026 BCB		

NOTES:

1. DIMENSIONS AND TOLERANCING PER ASME Y14.5M-1994.
2. CONTROLLING DIMENSION: MILLIMETER
3. DATUM PLANE H IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS L, M AND N TO BE DETERMINED AT DATUM PLANE H.

△5. DIMENSIONS TO BE DETERMINED AT SEATING PLANE T.

△6. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 PER SIDE. DIMENSIONS DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.

△7. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. DAMBAR PROTRUSION SHALL NOT CAUSE THE DIMENSION TO EXCEED 0.53. MINIMUM SPACE BETWEEN PROTRUSION AND ADJACENT LEAD OR PROTRUSION 0.07.

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: 44 LD LQFP, 10 X 10 PKG, 0.8 PITCH, 1.4 THICK	DOCUMENT NO: 98ASS23225W	REV: D	
	CASE NUMBER: 824D-02	26 FEB 2007	
	STANDARD: JEDEC MS-026 BCB		





## **How to Reach Us:**

### **Home Page:**

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

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

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. All rights reserved.