# THE DATASHEET OF

# MC9S08RD32DWE

www.win-source.net 0086-755-83957316

**Freescale Semiconductor**
Addendum

Document Number: QFN_Addendum
Rev. 0, 07/2014

# Addendum for New QFN Package Migration

This addendum provides the changes to the 98A case outline numbers for products covered in this book. Case outlines were changed because of the migration from gold wire to copper wire in some packages. See the table below for the old (gold wire) package versus the new (copper wire) package.

To view the new drawing, go to Freescale.com and search on the new 98A package number for your device.

For more information about QFN package use, see EB806: *Electrical Connection Recommendations for the Exposed Pad on QFN and DFN Packages*.

© Freescale Semiconductor, Inc., 2014. All rights reserved.

| Part Number | Package Description | Original (gold wire) package document number | Current (copper wire) package document number |
|---|---|---|---|
| MC68HC908JW32 | 48 QFN | 98ARH99048A | 98ASA00466D |
| MC9S08AC16 | | | |
| MC9S908AC60 | | | |
| MC9S08AC128 | | | |
| MC9S08AW60 | | | |
| MC9S08GB60A | | | |
| MC9S08GT16A | | | |
| MC9S08JM16 | | | |
| MC9S08JM60 | | | |
| MC9S08LL16 | | | |
| MC9S08QE128 | | | |
| MC9S08QE32 | | | |
| MC9S08RG60 | | | |
| MCF51CN128 | | | |
| MC9RS08LA8 | 48 QFN | 98ARL10606D | 98ASA00466D |
| MC9S08GT16A | 32 QFN | 98ARH99035A | 98ASA00473D |
| MC9S908QE32 | 32 QFN | 98ARE10566D | 98ASA00473D |
| MC9S908QE8 | 32 QFN | 98ASA00071D | 98ASA00736D |
| MC9S08JS16 | 24 QFN | 98ARL10608D | 98ASA00734D |
| MC9S08QB8 | | | |
| MC9S08QG8 | 24 QFN | 98ARL10605D | 98ASA00474D |
| MC9S08SH8 | 24 QFN | 98ARE10714D | 98ASA00474D |
| MC9RS08KB12 | 24 QFN | 98ASA00087D | 98ASA00602D |
| MC9S08QG8 | 16 QFN | 98ARE10614D | 98ASA00671D |
| MC9RS08KB12 | 8 DFN | 98ARL10557D | 98ASA00672D |
| MC9S08QG8 | | | |
| MC9RS08KA2 | 6 DFN | 98ARL10602D | 98ASA00735D |

# MC9S08RC8/16/32/60
# MC9S08RD8/16/32/60
# MC9S08RE8/16/32/60
# MC9S08RG32/60

Data Sheet

***HCS08***
***Microcontrollers***

MC9S08RG60/D
Rev. 1.11
06/2005

freescale.com

freescale™
semiconductor

# MC9S08RG60 Data Sheet

Covers: MC9S08RC8/16/32/60
MC9S08RD8/16/32/60
MC9S08RE8/16/32/60
MC9S08RG32/60

MC9S08RG60/D
Rev. 1.11
06/2005

freescale™
semiconductor

# Revision History

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

http://freescale.com

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

| Version Number | Revision Date | Description of Changes |
|---|---|---|
| 1.11 | 06/2005 | Added 48 QFN package and official mechanical drawings; suppled TBD values for IRO $V_{OL}$; updated $t_{RTI}$ values; re-emphasized that KBI2 will not wake the MCU from stop2 mode. |

This product contains SuperFlash® technology licensed from SST.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.

© Freescale Semiconductor, Inc., 2005. All rights reserved.

# List of Chapters


| | | |
|---|---|---|
| Chapter 1 | Introduction | 15 |
| Chapter 2 | Pins and Connections | 19 |
| Chapter 3 | Modes of Operation | 29 |
| Chapter 4 | Memory | 35 |
| Chapter 5 | Resets, Interrupts, and System Configuration | 57 |
| Chapter 6 | Parallel Input/Output | 73 |
| Chapter 7 | Central Processor Unit (S08CPUV2) | 87 |
| Chapter 8 | Carrier Modulator Timer (S08CMTV1) | 107 |
| Chapter 9 | Keyboard Interrupt (S08KBIV1) | 123 |
| Chapter 10 | Timer/PWM Module (S08TPMV1) | 129 |
| Chapter 11 | Serial Communications Interface (S08SCIV1) | 145 |
| Chapter 12 | Serial Communications Interface (S08SCIV1) | 147 |
| Chapter 13 | Serial Peripheral Interface (S08SPIV3) | 163 |
| Chapter 14 | Analog Comparator (S08ACMPV1) | 179 |
| Chapter 15 | Development Support | 183 |
| Appendix A | Electrical Characteristics | 205 |
| Appendix B | Ordering Information and Mechanical Drawings | 219 |

# Contents

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

## Chapter 1
## Introduction


1.1 Overview ..... 15
1.2 Features ..... 15
1.2.1 Devices in the MC9S08RC/RD/RE/RG Series ..... 16
1.3 MCU Block Diagram ..... 17
1.4 System Clock Distribution ..... 18


## Chapter 2
## Pins and Connections


2.1 Introduction ..... 19
2.2 Device Pin Assignment ..... 19
2.3 Recommended System Connections ..... 21
2.3.1 Power ..... 23
2.3.2 Oscillator ..... 23
2.3.3 PTD1/$\overline{\text{RESET}}$ ..... 23
2.3.4 Background/Mode Select (PTD0/BKGD/MS) ..... 24
2.3.5 IRO Pin Description ..... 24
2.3.6 General-Purpose I/O and Peripheral Ports ..... 24
2.3.7 Signal Properties Summary ..... 25


## Chapter 3
## Modes of Operation


3.1 Introduction ..... 29
3.2 Features ..... 29
3.3 Run Mode ..... 29
3.4 Active Background Mode ..... 29
3.5 Wait Mode ..... 30
3.6 Stop Modes ..... 31
3.6.1 Stop1 Mode ..... 31
3.6.2 Stop2 Mode ..... 31
3.6.3 Stop3 Mode ..... 32
3.6.4 Active BDM Enabled in Stop Mode ..... 33
3.6.5 LVD Reset Enabled ..... 33
3.6.6 On-Chip Peripheral Modules in Stop Mode ..... 33

## Chapter 4
## Memory


4.1 MC9S08RC/RD/RE/RG Memory Map ..... 35
4.1.1 Reset and Interrupt Vector Assignments ..... 36
4.2 Register Addresses and Bit Assignments ..... 38
4.3 RAM ..... 42
4.4 FLASH ..... 42
4.4.1 Features ..... 43
4.4.2 Program and Erase Times ..... 43
4.4.3 Program and Erase Command Execution ..... 44
4.4.4 Burst Program Execution ..... 45
4.4.5 Access Errors ..... 46
4.4.6 FLASH Block Protection ..... 47
4.4.7 Vector Redirection ..... 48
4.5 Security ..... 48
4.6 FLASH Registers and Control Bits ..... 49
4.6.1 FLASH Clock Divider Register (FCDIV) ..... 49
4.6.2 FLASH Options Register (FOPT and NVOPT) ..... 51
4.6.3 FLASH Configuration Register (FCNFG) ..... 51
4.6.4 FLASH Protection Register (FPROT and NVPROT) ..... 52
4.6.5 FLASH Status Register (FSTAT) ..... 54
4.6.6 FLASH Command Register (FCMD) ..... 55


## Chapter 5
## Resets, Interrupts, and System Configuration


5.1 Introduction ..... 57
5.2 Features ..... 57
5.3 MCU Reset ..... 57
5.4 Computer Operating Properly (COP) Watchdog ..... 58
5.5 Interrupts ..... 58
5.5.1 Interrupt Stack Frame ..... 59
5.5.2 External Interrupt Request (IRQ) Pin ..... 60
5.5.2.1 Pin Configuration Options ..... 60
5.5.2.2 Edge and Level Sensitivity ..... 61
5.5.3 Interrupt Vectors, Sources, and Local Masks ..... 61
5.6 Low-Voltage Detect (LVD) System ..... 62
5.6.1 Power-On Reset Operation ..... 63
5.6.2 LVD Reset Operation ..... 63
5.6.3 LVD Interrupt and Safe State Operation ..... 63
5.6.4 Low-Voltage Warning (LVW) ..... 63
5.7 Real-Time Interrupt (RTI) ..... 64

5.8 Reset, Interrupt, and System Control Registers and Control Bits ....64
5.8.1 Interrupt Pin Request Status and Control Register (IRQSC) ....64
5.8.2 System Reset Status Register (SRS) ....65
5.8.3 System Background Debug Force Reset Register (SBDFR) ....67
5.8.4 System Options Register (SOPT) ....68
5.8.5 System Device Identification Register (SDIDH, SDIDL) ....69
5.8.6 System Real-Time Interrupt Status and Control Register (SRTISC) ....70
5.8.7 System Power Management Status and Control 1 Register (SPMSC1) ....71
5.8.8 System Power Management Status and Control 2 Register (SPMSC2) ....72

## Chapter 6
## Parallel Input/Output

6.1 Introduction ....73
6.2 Features ....73
6.3 Pin Descriptions ....74
6.3.1 Port A ....74
6.3.2 Port B ....74
6.3.3 Port C ....75
6.3.4 Port D ....75
6.3.5 Port E ....76
6.4 Parallel I/O Controls ....76
6.4.1 Data Direction Control ....76
6.4.2 Internal Pullup Control ....77
6.5 Stop Modes ....77
6.6 Parallel I/O Registers and Control Bits ....77
6.6.1 Port A Registers (PTAD, PTAPE, and PTADD) ....78
6.6.2 Port B Registers (PTBD, PTBPE, and PTBDD) ....79
6.6.3 Port C Registers (PTCD, PTCPE, and PTCDD) ....81
6.6.4 Port D Registers (PTDD, PTDPE, and PTDDD) ....82
6.6.5 Port E Registers (PTED, PTEPE, and PTEDD) ....84

## Chapter 7
## Central Processor Unit (S08CPUV2)

7.1 Introduction ....87
7.1.1 Features ....87
7.2 Programmer's Model and CPU Registers ....88
7.2.1 Accumulator (A) ....88
7.2.2 Index Register (H:X) ....88
7.2.3 Stack Pointer (SP) ....89
7.2.4 Program Counter (PC) ....89
7.2.5 Condition Code Register (CCR) ....89
7.3 Addressing Modes ....90
7.3.1 Inherent Addressing Mode (INH) ....91
7.3.2 Relative Addressing Mode (REL) ....91
7.3.3 Immediate Addressing Mode (IMM) ....91

7.3.4 Direct Addressing Mode (DIR) ....91
7.3.5 Extended Addressing Mode (EXT) ....91
7.3.6 Indexed Addressing Mode ....91
7.3.6.1 Indexed, No Offset (IX) ....92
7.3.6.2 Indexed, No Offset with Post Increment (IX+) ....92
7.3.6.3 Indexed, 8-Bit Offset (IX1) ....92
7.3.6.4 Indexed, 8-Bit Offset with Post Increment (IX1+) ....92
7.3.6.5 Indexed, 16-Bit Offset (IX2) ....92
7.3.6.6 SP-Relative, 8-Bit Offset (SP1) ....92
7.3.6.7 SP-Relative, 16-Bit Offset (SP2) ....92
7.4 Special Operations ....92
7.4.1 Reset Sequence ....93
7.4.2 Interrupt Sequence ....93
7.4.3 Wait Mode Operation ....94
7.4.4 Stop Mode Operation ....94
7.4.5 BGND Instruction ....94
7.5 HCS08 Instruction Set Summary ....95

## Chapter 8
## Carrier Modulator Timer (S08CMTV1)

8.1 Introduction ....107
8.2 Features ....108
8.3 CMT Block Diagram ....108
8.4 Pin Description ....108
8.5 Functional Description ....109
8.5.1 Carrier Generator ....110
8.5.2 Modulator ....112
8.5.2.1 Time Mode ....113
8.5.2.2 Baseband Mode ....114
8.5.2.3 FSK Mode ....114
8.5.3 Extended Space Operation ....115
8.5.3.1 EXSPC Operation in Time Mode ....115
8.5.3.2 EXSPC Operation in FSK Mode ....116
8.5.4 Transmitter ....116
8.5.5 CMT Interrupts ....117
8.5.6 Wait Mode Operation ....117
8.5.7 Stop Mode Operation ....117
8.5.8 Background Mode Operation ....118
8.6 CMT Registers and Control Bits ....118
8.6.1 Carrier Generator Data Registers (CMTCGH1, CMTCGL1, CMTCGH2, and CMTCGL2) ....118
8.6.2 CMT Output Control Register (CMTOC) ....120
8.6.3 CMT Modulator Status and Control Register (CMTMSC) ....121
8.6.4 CMT Modulator Data Registers (CMTCMD1, CMTCMD2, CMTCMD3, and CMTCMD4) ....122

## Chapter 9
## Keyboard Interrupt (S08KBIV1)

9.1 Introduction ....123
9.2 KBI Block Diagram ....125
9.3 Keyboard Interrupt (KBI) Module ....125
9.3.1 Pin Enables ....125
9.3.2 Edge and Level Sensitivity ....125
9.3.3 KBI Interrupt Controls ....126
9.4 KBI Registers and Control Bits ....126
9.4.1 KBI x Status and Control Register (KBIxSC) ....127
9.4.2 KBI x Pin Enable Register (KBIxPE) ....128

## Chapter 10
## Timer/PWM Module (S08TPMV1)

10.1 Introduction ....129
10.2 Features ....129
10.3 TPM Block Diagram ....131
10.4 Pin Descriptions ....132
10.4.1 External TPM Clock Sources ....132
10.4.2 TPM1CHn — TPM1 Channel n I/O Pins ....132
10.5 Functional Description ....132
10.5.1 Counter ....133
10.5.2 Channel Mode Selection ....134
10.5.2.1 Input Capture Mode ....134
10.5.2.2 Output Compare Mode ....134
10.5.2.3 Edge-Aligned PWM Mode ....134
10.5.3 Center-Aligned PWM Mode ....135
10.6 TPM Interrupts ....137
10.6.1 Clearing Timer Interrupt Flags ....137
10.6.2 Timer Overflow Interrupt Description ....137
10.6.3 Channel Event Interrupt Description ....137
10.6.4 PWM End-of-Duty-Cycle Events ....138
10.7 TPM Registers and Control Bits ....138
10.7.1 Timer Status and Control Register (TPM1SC) ....139
10.7.2 Timer Counter Registers (TPM1CNTH:TPM1CNTL) ....140
10.7.3 Timer Counter Modulo Registers (TPM1MODH:TPM1MODL) ....141
10.7.4 Timer Channel n Status and Control Register (TPM1CnSC) ....142
10.7.5 Timer Channel Value Registers (TPM1CnVH:TPM1CnVL) ....143

## Chapter 11
## Serial Communications Interface (S08SCIV1)

11.1 Introduction ....145

## Chapter 12
## Serial Communications Interface (S08SCIV1)

12.1 Introduction ....147
12.1.1 Features ....147
12.1.2 Modes of Operation ....147
12.1.3 Block Diagram ....148
12.2 Register Definition ....150
12.2.1 SCI Baud Rate Registers (SCI1BDH, SCI1BHL) ....150
12.2.2 SCI Control Register 1 (SCI1C1) ....151
12.2.3 SCI Control Register 2 (SCI1C2) ....152
12.2.4 SCI Status Register 1 (SCI1S1) ....153
12.2.5 SCI Status Register 2 (SCI1S2) ....155
12.2.6 SCI Control Register 3 (SCI1C3) ....155
12.2.7 SCI Data Register (SCI1D) ....156
12.3 Functional Description ....157
12.3.1 Baud Rate Generation ....157
12.3.2 Transmitter Functional Description ....157
12.3.2.1 Send Break and Queued Idle ....158
12.3.3 Receiver Functional Description ....158
12.3.3.1 Data Sampling Technique ....159
12.3.3.2 Receiver Wakeup Operation ....159
12.3.4 Interrupts and Status Flags ....160
12.3.5 Additional SCI Functions ....161
12.3.5.1 8- and 9-Bit Data Modes ....161
12.3.5.2 Stop Mode Operation ....161
12.3.5.3 Loop Mode ....161
12.3.5.4 Single-Wire Operation ....162

## Chapter 13
## Serial Peripheral Interface (S08SPIV3)

13.1 Features ....164
13.2 Block Diagrams ....164
13.2.1 SPI System Block Diagram ....164
13.2.2 SPI Module Block Diagram ....165
13.2.3 SPI Baud Rate Generation ....167
13.3 Functional Description ....167
13.3.1 SPI Clock Formats ....168
13.3.2 SPI Pin Controls ....170
13.3.2.1 SPSCK1 — SPI Serial Clock ....170
13.3.2.2 MOSI1 — Master Data Out, Slave Data In ....170

13.3.2.3 MISO1 — Master Data In, Slave Data Out ....170
13.3.2.4 $\overline{SS1}$ — Slave Select ....170
13.3.3 SPI Interrupts ....171
13.3.4 Mode Fault Detection ....171
13.4 SPI Registers and Control Bits ....171
13.4.1 SPI Control Register 1 (SPI1C1) ....172
13.4.2 SPI Control Register 2 (SPI1C2) ....173
13.4.3 SPI Baud Rate Register (SPI1BR) ....174
13.4.4 SPI Status Register (SPI1S) ....176
13.4.5 SPI Data Register (SPI1D) ....177

## Chapter 14
## Analog Comparator (S08ACMPV1)

14.1 Features ....180
14.2 Block Diagram ....180
14.3 Pin Description ....180
14.4 Functional Description ....181
14.4.1 Interrupts ....181
14.4.2 Wait Mode Operation ....181
14.4.3 Stop Mode Operation ....181
14.4.4 Background Mode Operation ....181
14.5 ACMP Status and Control Register (ACMP1SC) ....182

## Chapter 15
## Development Support

15.1 Introduction ....183
15.1.1 Features ....183
15.2 Background Debug Controller (BDC) ....184
15.2.1 BKGD Pin Description ....184
15.2.2 Communication Details ....185
15.2.3 BDC Commands ....189
15.2.4 BDC Hardware Breakpoint ....191
15.3 On-Chip Debug System (DBG) ....192
15.3.1 Comparators A and B ....192
15.3.2 Bus Capture Information and FIFO Operation ....192
15.3.3 Change-of-Flow Information ....193
15.3.4 Tag vs. Force Breakpoints and Triggers ....193
15.3.5 Trigger Modes ....194
15.3.6 Hardware Breakpoints ....196
15.4 Register Definition ....196
15.4.1 BDC Registers and Control Bits ....196
15.4.1.1 BDC Status and Control Register (BDCSCR) ....197
15.4.1.2 BDC Breakpoint Match Register (BDCBKPT) ....198
15.4.2 System Background Debug Force Reset Register (SBDFR) ....198

15.4.3 DBG Registers and Control Bits ....199
15.4.3.1 Debug Comparator A High Register (DBGCAH) ....199
15.4.3.2 Debug Comparator A Low Register (DBGCAL) ....199
15.4.3.3 Debug Comparator B High Register (DBGCBH) ....199
15.4.3.4 Debug Comparator B Low Register (DBGCBL) ....199
15.4.3.5 Debug FIFO High Register (DBGFH) ....200
15.4.3.6 Debug FIFO Low Register (DBGFL) ....200
15.4.3.7 Debug Control Register (DBGC) ....201
15.4.3.8 Debug Trigger Register (DBGT) ....202
15.4.3.9 Debug Status Register (DBGS) ....203

## Appendix A
## Electrical Characteristics

A.1 Introduction ....205
A.2 Absolute Maximum Ratings ....205
A.3 Thermal Characteristics ....206
A.4 Electrostatic Discharge (ESD) Protection Characteristics ....207
A.5 DC Characteristics ....207
A.6 Supply Current Characteristics ....211
A.7 Analog Comparator (ACMP) Electricals ....211
A.8 Oscillator Characteristics ....212
A.9 AC Characteristics ....212
A.9.1 Control Timing ....212
A.9.2 Timer/PWM (TPM) Module Timing ....213
A.9.3 SPI Timing ....214
A.10 FLASH Specifications ....218

## Appendix B
## Ordering Information and Mechanical Drawings

B.1 Ordering Information ....219
B.2 Mechanical Drawings ....220

# Chapter 1
# Introduction

## 1.1 Overview

The MC9S08RC/RD/RE/RG are members of the low-cost, high-performance HCS08 Family of 8-bit microcontroller units (MCUs). All MCUs in this family use the enhanced HCS08 core and are available with a variety of modules, memory sizes, memory types, and package types.

## 1.2 Features

Features of the MC9S08RC/RD/RE/RG Family of devices are listed here. Please see Table 1-1 for the features that are available on the different family members.

| | |
|---|---|
| **HCS08 CPU (Central Processor Unit)** | • Object code fully upward-compatible with M68HC05 and M68HC08 Families<br>• HC08 instruction set with added BGND instruction<br>• Support for up to 32 interrupt/reset sources<br>• Power-saving modes: wait plus three stops |
| **On-Chip Memory** | • On-chip in-circuit programmable FLASH memory with block protection and security option<br>• On-chip random-access memory (RAM) |
| **Oscillator (OSC)** | • Low power oscillator capable of operating from crystal or resonator from 1 to 16 MHz<br>• 8 MHz internal bus frequency |
| **Analog Comparator (ACMP1)** | • On-chip analog comparator with internal reference (ACMP1)<br>• Full rail-to-rail supply operation<br>• Option to compare to a fixed internal bandgap reference voltage |
| **Serial Communications Interface Module (SCI1)** | • Full-duplex, standard non-return-to-zero (NRZ) format<br>• Double-buffered transmitter and receiver with separate enables<br>• Programmable 8-bit or 9-bit character length<br>• Programmable baud rates (13-bit modulo divider) |
| **Serial Peripheral Interface Module (SPI1)** | • Master or slave mode operation<br>• Full-duplex or single-wire bidirectional option<br>• Programmable transmit bit rate<br>• Double-buffered transmit and receive<br>• Serial clock phase and polarity options<br>• Slave select output<br>• Selectable MSB-first or LSB-first shifting |

| | |
|---|---|
| **Timer/Pulse-Width Modulator (TPM1)** | • 2-channel, 16-bit timer/pulse-width modulator (TPM1) module that can operate as a free-running counter, a modulo counter, or an up-/down-counter when the TPM is configured for center-aligned PWM<br>• Selectable input capture, output compare, and edge-aligned or center-aligned PWM capability on each channel |
| **Keyboard Interrupt Ports (KBI1, KBI2)** | • Providing 12 keyboard interrupts<br>• Eight with falling-edge/low-level plus four with selectable polarity<br>• KBI1 inputs can be configured for edge-only sensitivity or edge-and-level sensitivity |
| **Carrier Modulator Timer (CMT)** | • Dedicated infrared output (IRO) pin<br>• Drives IRO pin for remote control communications<br>• Can be disconnected from IRO pin and used as output compare timer<br>• IRO output pin has high-current sink capability |
| **Development Support** | • Background debugging system (see also the Development Support chapter)<br>• Breakpoint capability to allow single breakpoint setting during in-circuit debugging (plus two more breakpoints in on-chip debug module)<br>• 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. |
| **Port Pins** | • Eight high-current pins (limited by maximum package dissipation)<br>• Software selectable pullups on ports when used as input. Selection is on an individual port bit basis. During output mode, pullups are disengaged.<br>• 39 general-purpose input/output (I/O) pins, depending on package selection |
| **Package Options** | • 28-pin plastic dual in-line package (PDIP)<br>• 28-pin small outline integrated circuit (SOIC)<br>• 32-pin low-profile quad flat package (LQFP)<br>• 44-pin low-profile quad flat package (LQFP)<br>• 48-pin quad flat package (QFN) |
| **System Protection** | • Optional computer operating properly (COP) reset<br>• Low-voltage detection with reset or interrupt<br>• Illegal opcode detection with reset<br>• Illegal address detection with reset (some devices don't have illegal addresses) |

### 1.2.1 Devices in the MC9S08RD/RE/RG Series

Table 1-1 lists the devices available in the MC9S08RD/RE/RG series and summarizes the differences in functions and configuration among them.

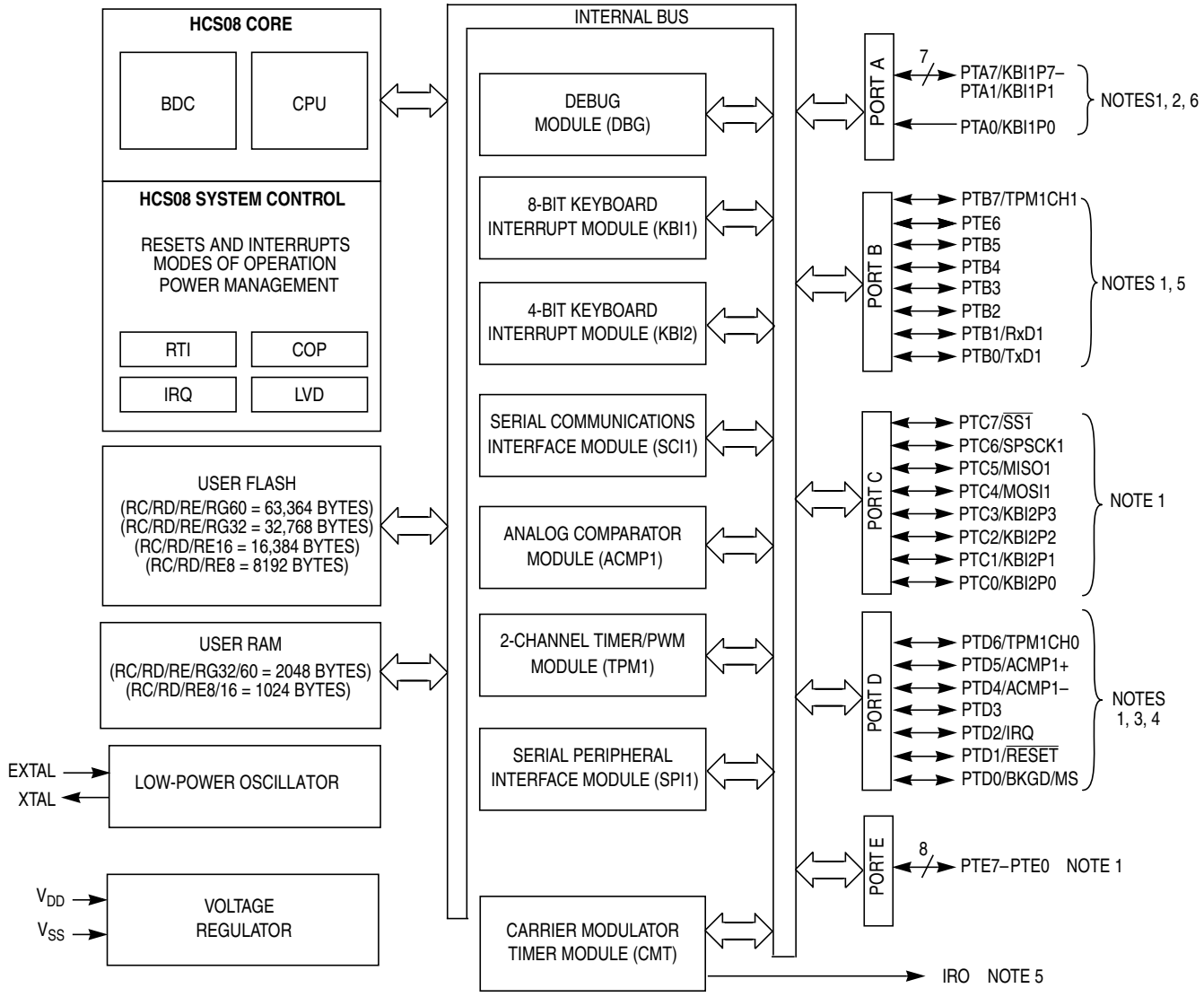
**Table 1-1. Devices in the MC9S08RD/RE/RG Series**

| Device | FLASH | RAM | ACMP(1) | SCI | SPI |
|---|---|---|---|---|---|
| 9S08RG32/60 | 32K/60K | 2K/2K | Yes | Yes | Yes |
| 9S08RE8/16/32/60 | 8/16K/32K/60K | 1K/1K/2K/2K | Yes | Yes | No |
| 9S08RD8/16/32/60 | 8/16K/32K/60K | 1K/1K/2K/2K | No | Yes | No |

1. Available only in 32-, 44-, and 48-pin LQFP packages.

## 1.3 MCU Block Diagram

This block diagram shows the structure of the MC9S08RC/RD/RE/RG MCUs




NOTES:

1. Port pins are software configurable with pullup device if input port
2. PTA0 does not have a clamp diode to $V_{DD}$. PTA0 should not be driven above $V_{DD}$. Also, PTA0 does not pullup to $V_{DD}$ when internal pullup is enabled.
3. IRQ pin contains software configurable pullup/pulldown device if IRQ enabled (IRQPE = 1)
4. The RESET pin contains integrated pullup device enabled if reset enabled (RSTPE = 1)
5. High current drive
6. Pins PTA[7:4] contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

**Figure 1-1. MC9S08RC/RD/RE/RG Block Diagram**

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

**Table 1-2. Block Versions**

| Module | Version |
|---|---|
| Analog Comparator (ACMP) | 1 |
| Carrier Modulator Transmitter (CMT) | 1 |
| Keyboard Interrupt (KBI) | 1 |
| Serial Communications Interface (SCI) | 1 |
| Serial Peripheral Interface (SPI) | 3 |
| Timer Pulse-Width Modulator (TPM) | 1 |
| Central Processing Unit (CPU) | 2 |
| Debug Module (DBG) | 1 |
| FLASH | 1 |
| System Control | 2 |

## 1.4 System Clock Distribution

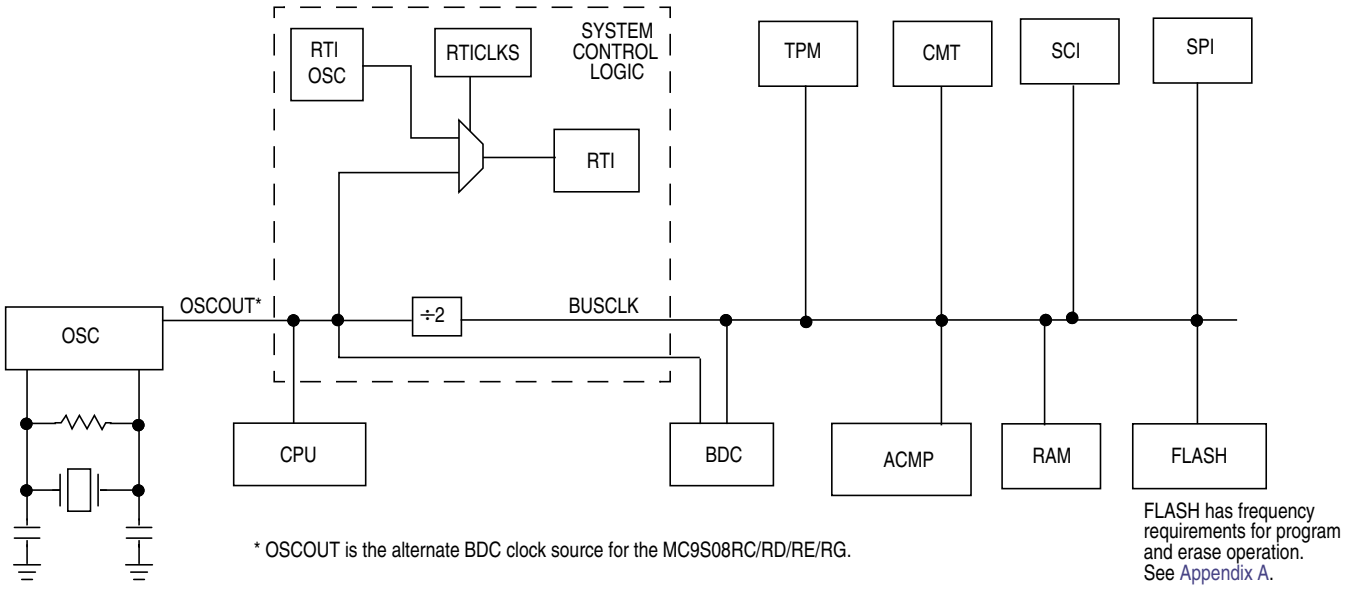



**Figure 1-2. System Clock Distribution Diagram**

Table 1-2 shows a simplified clock connection diagram for the MCU. The CPU operates at the input frequency of the oscillator. The bus clock frequency is half of the oscillator frequency and is used by all of the internal circuits with the exception of the CPU and RTI. The RTI can use either the oscillator input or the internal RTI oscillator as its clock source.

# Chapter 2
# Pins and Connections

## 2.1 Introduction

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

## 2.2 Device Pin Assignment

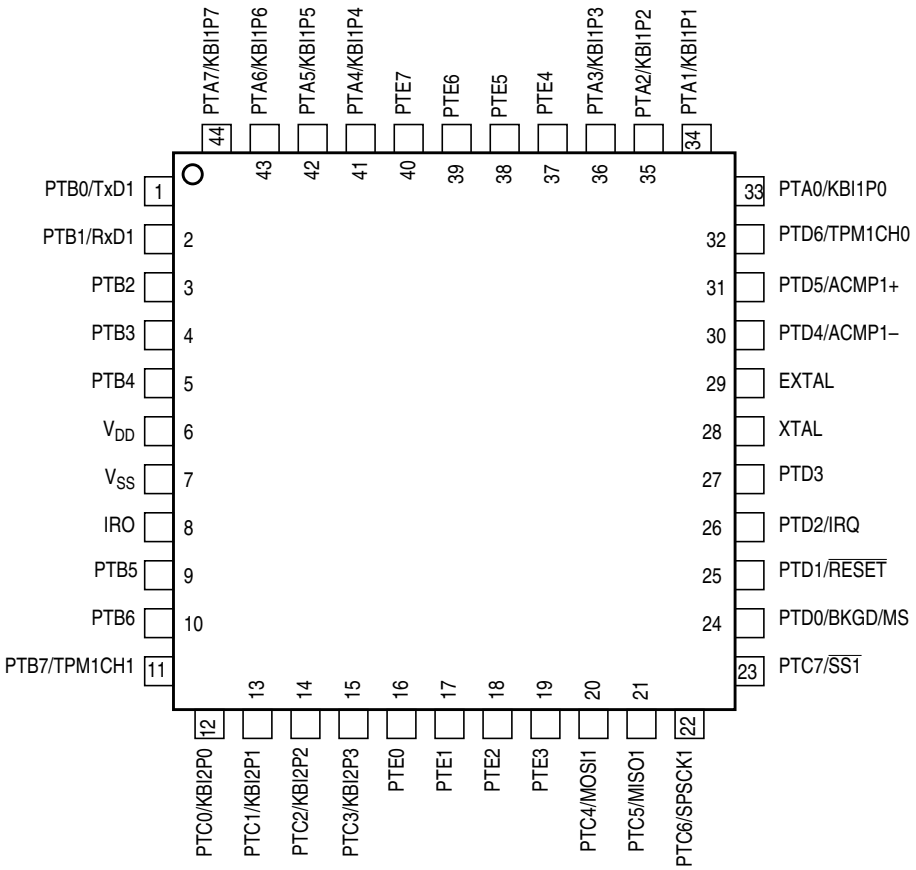



**Figure 2-1. MC9S08RC/RD/RE/RG in 44-Pin LQFP Package**

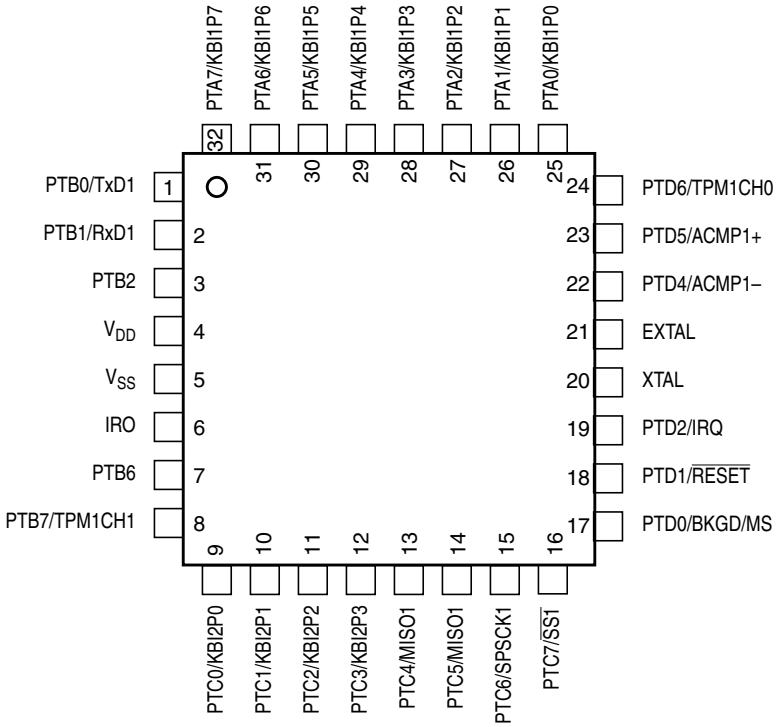



**Figure 2-2. MC9S08RC/RD/RE/RG in 32-Pin LQFP Package**

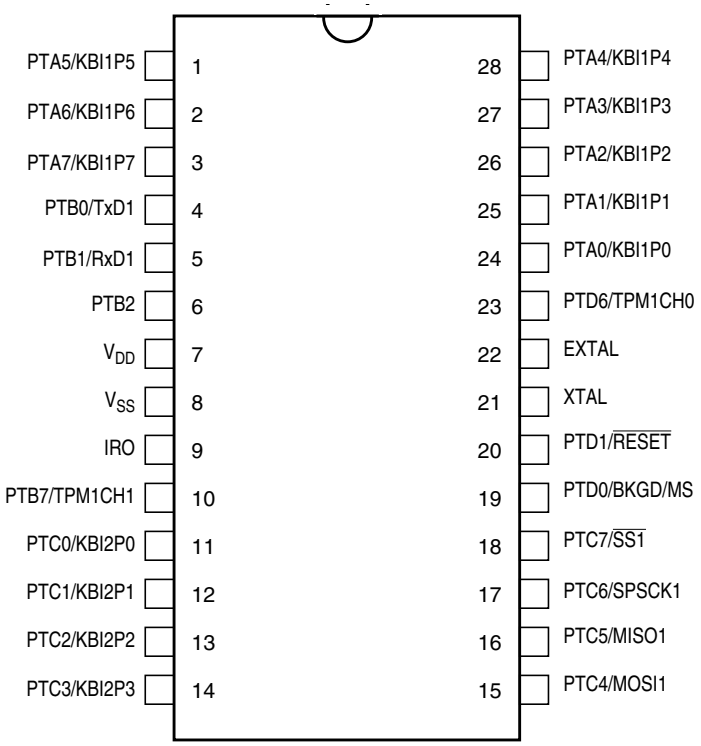



**Figure 2-3. MC9S08RC/RD/RE/RG in 28-Pin SOIC Package and 28-Pin PDIP Package**

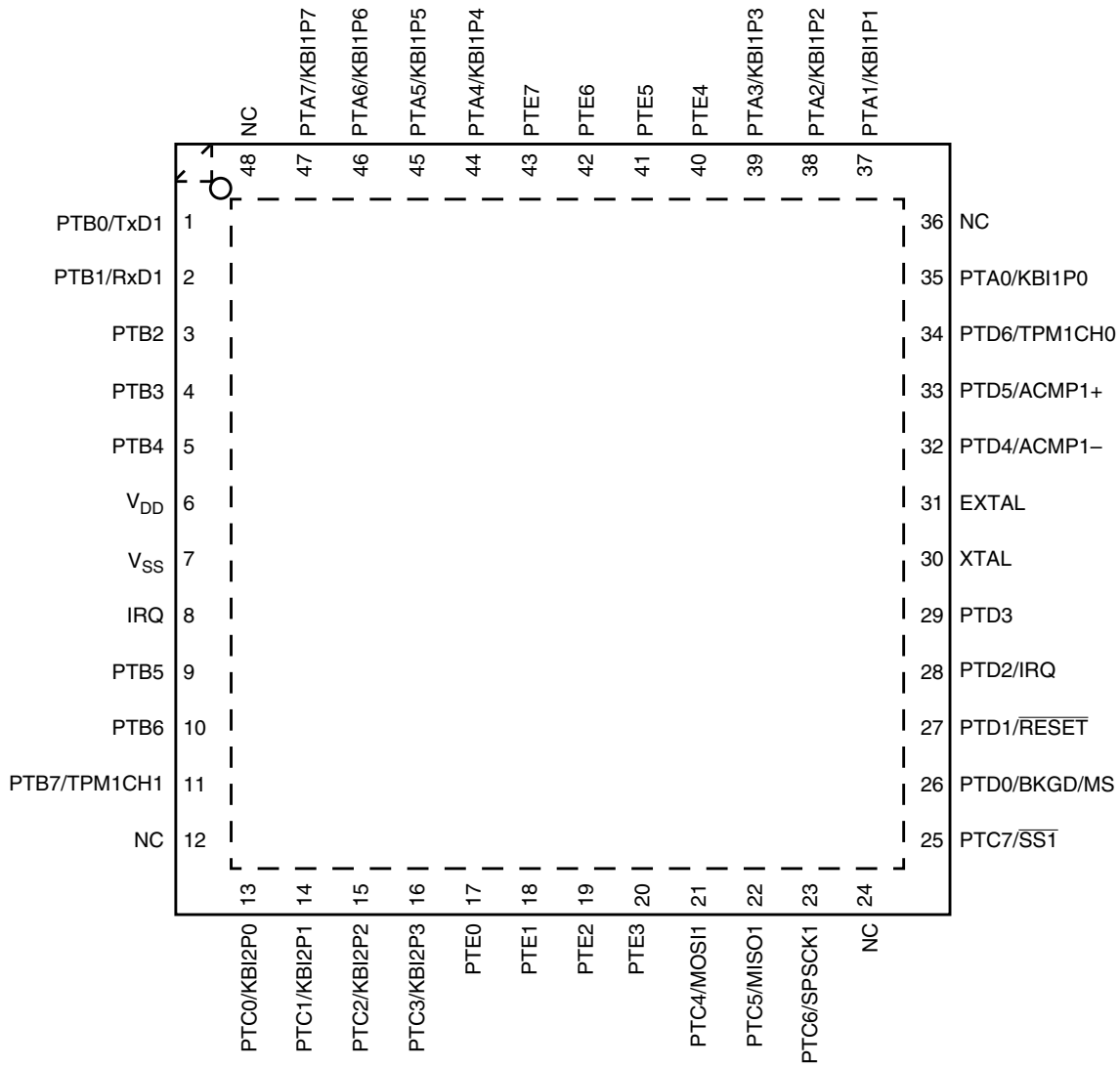
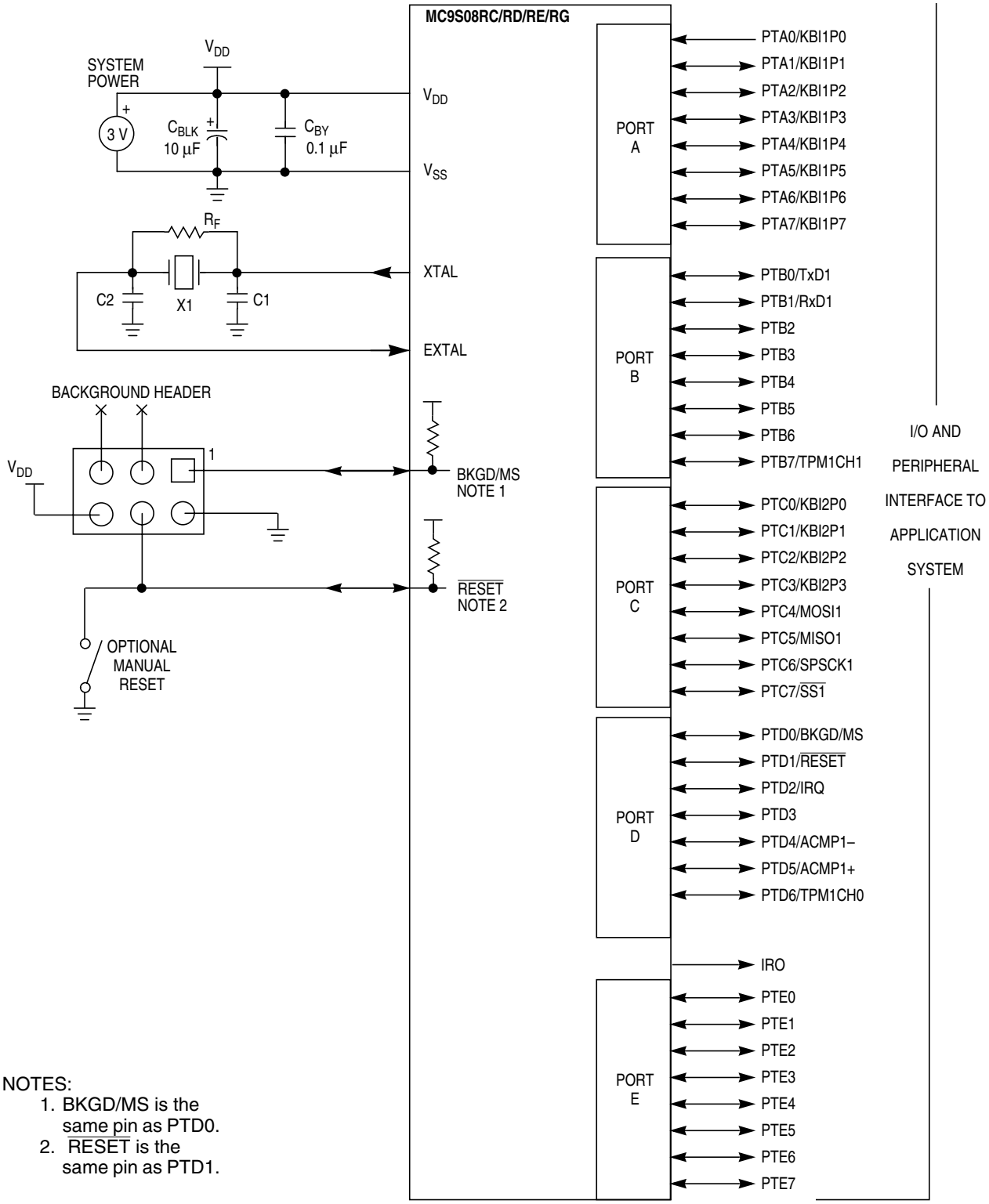



**Figure 2-4. MC9S08RC/RD/RE/RG in 48-Pin QFN Package**

## 2.3 Recommended System Connections

Figure 2-5 shows pin connections that are common to almost all MC9S08RC/RD/RE/RG application systems. A more detailed discussion of system connections follows.




NOTES:
1. BKGD/MS is the same pin as PTD0.
2. RESET is the same pin as PTD1.

**Figure 2-5. Basic System Connections**

### 2.3.1 Power

$V_{DD}$ 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 a 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-µF tantalum capacitor, to provide bulk charge storage for the overall system and a 0.1-µF ceramic bypass capacitor located as near to the MCU power pins as practical to suppress high-frequency noise.

### 2.3.2 Oscillator

The oscillator in the MC9S08RC/RD/RE/RG is a traditional Pierce oscillator that can accommodate a crystal or ceramic resonator in the range of 1 MHz to 16 MHz.

Refer to Figure 2-5 for the following discussion. $R_F$ should be a low-inductance resistor such as a carbon composition resistor. Wire-wound resistors, and some metal film resistors, have too much inductance. C1 and C2 normally should be high-quality ceramic capacitors 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 MΩ. Higher values are sensitive to humidity and lower values reduce gain and (in extreme cases) could prevent startup.

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

### 2.3.3 PTD1/$\overline{\text{RESET}}$

The external pin reset function is shared with an output-only port function on the PTD1/$\overline{\text{RESET}}$ pin. The reset function is enabled when RSTPE in SOPT is set. RSTPE is set following any reset of the MCU and must be cleared in order to use this pin as an output-only port.

Whenever any reset is initiated (whether from an external signal or from an internal system), the reset pin is driven low for about 34 cycles of $f_{Self_reset}$, released, and sampled again about 38 cycles of $f_{Self_reset}$ later. If reset was caused by an internal source such as low-voltage reset or watchdog timeout, the circuitry expects the reset pin sample to return a logic 1. If the pin is still low at this sample point, the reset is assumed to be from an external source. The reset circuitry decodes the cause of reset and records it by setting a corresponding bit in the system control reset status register (SRS).

Never connect any significant capacitance to the reset pin because that would interfere with the circuit and sequence that detects the source of reset. If an external capacitance prevents the reset pin from rising to a valid logic 1 before the reset sample point, all resets will appear to be external resets.

### 2.3.4 Background/Mode Select (PTD0/BKGD/MS)

The background/mode select function is shared with an output-only port function on the PTD0/BKDG/MS pin. While in reset, the 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, this pin has an internal pullup device enabled. To use as an output-only port, BKGDPE in SOPT must be cleared.

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 IRO Pin Description

The IRO pin is the output of the CMT. See the Carrier Modulator Timer (CMT) Module Chapter for a detailed description of this pin function.

### 2.3.6 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. (Not all pins are available in all packages. See Table 2-2.) Immediately after reset, all 37 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 the 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 2-1.

**Table 2-1. Pin Sharing References**

| Port Pins | Alternate Function | Reference[1] |
|---|---|---|
| PTA7–PTA0 | KBI1P7–KBI1P0 | Chapter 9, “Keyboard Interrupt (S08KBIV1)” |
| PTB7 | TPM1CH1 | Chapter 10, “Timer/PWM Module (S08TPMV1)” |
| PTB6–PTB2 | — | Chapter 6, “Parallel Input/Output” |
| PTB1<br>PTB0 | RxD1<br>TxD1 | Chapter 11, “Serial Communications Interface (S08SCIV1)” |
| PTC7<br>PTC6<br>PTC5<br>PTC4 | SS1<br>SPSCK1<br>MISO1<br>MOSI1 | Chapter 13, “Serial Peripheral Interface (S08SPIV3)” |
| PTC3–PTC0 | KBI2P3–KBI2P0 | Chapter 9, “Keyboard Interrupt (S08KBIV1)” |
| PTD6 | TPM1CH0 | Chapter 10, “Timer/PWM Module (S08TPMV1)” |
| PTD5<br>PTD4 | ACMP1+<br>ACMP1– | Chapter 14, “Analog Comparator (S08ACMPV1)” |
| PTD2 | IRQ | Chapter 5, “Resets, Interrupts, and System Configuration” |
| PTD1 | RESET | |
| PTD0 | BKGD/MS | |
| PTE7–PTE0 | — | Chapter 6, “Parallel Input/Output” |

1. See this chapter for information about modules that share these pins.

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,” 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 the PTA7–PTA4 pins are controlled by the KBI module and are configured for rising-edge/high-level sensitivity, the pullup enable control bits enable pulldown devices rather than pullup devices. Similarly, when PTD2 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.

## 2.3.7 Signal Properties Summary

Table 2-2 summarizes I/O pin characteristics. These characteristics are determined by the way the common pin interfaces are hardwired to internal circuits.

**Table 2-2. Signal Properties**

| Pin Name | Dir[1] | High Current Pin | Pullup[2] | Comments[3] |
|---|---|---|---|---|
| $V_{DD}$ | | — | — | |
| $V_{SS}$ | | — | — | |
| XTAL | O | — | — | Crystal oscillator output |
| EXTAL | I | — | — | Crystal oscillator input |
| IRO | O | Y | — | Infrared output |
| PTA0/KBI1P0 | I | N | SWC | PTA0 does not have a clamp diode to $V_{DD}$. PTA0 should not be driven above $V_{DD}$. |
| PTA1/KBI1P1 | I/O | N | SWC | |
| PTA2/KBI1P2 | I/O | N | SWC | |
| PTA3/KBI1P3 | I/O | N | SWC | |
| PTA4/KBI1P4 | I/O | N | SWC | |
| PTA5/KBI1P5 | I/O | N | SWC | |
| PTA6/KBI1P6 | I/O | N | SWC | |
| PTA7/KBI1P7 | I/O | N | SWC | |
| PTB0/TxD1 | I/O | Y | SWC | |
| PTB1/RxD1 | I/O | Y | SWC | |
| PTB2 | I/O | Y | SWC | |
| PTB3 | I/O | Y | SWC | Available only in 44- and 48-pin packages |
| PTB4 | I/O | Y | SWC | Available only in 44- and 48-pin packages |
| PTB5 | I/O | Y | SWC | Available only in 44- and 48-pin packages |
| PTB6 | I/O | Y | SWC | Available only in 32-, 44-, and 48-pin packagess |
| PTB7/TPM1CH1 | I/O | Y | SWC | |
| PTC0/KBI2P0 | I/O | N | SWC | |
| PTC1/KBI2P1 | I/O | N | SWC | |
| PTC2/KBI2P2 | I/O | N | SWC | |
| PTC3/KBI2P3 | I/O | N | SWC | |
| PTC4/MOSI1 | I/O | N | SWC | |
| PTC5/MISO1 | I/O | N | SWC | |
| PTC6/SPSCK1 | I/O | N | SWC | |
| PTC7/$\overline{SS1}$ | I/O | N | SWC | |
| PTD0/BKGD/MS | I/O | N | SWC[4] | Output-only when configured as PTD0 pin. Pullup enabled. |
| PTD1/$\overline{RESET}$ | I/O | N | SWC[3] | Output-only when configured as PTD1 pin. |

**Table 2-2. Signal Properties (continued)**

| Pin Name | Dir[1] | High Current Pin | Pullup[2] | Comments[3] |
|---|---|---|---|---|
| PTD2/IRQ | I/O | N | SWC[5] | Available only in 32-, 44-, and 48-pin packagess |
| PTD3 | I/O | N | SWC | Available only in 44- and 48-pin packages |
| PTD4/ACMP1– | I/O | N | SWC | Available only in 32-, 44-, and 48-pin packagess |
| PTD5/ACMP1+ | I/O | N | SWC | Available only in 32-, 44-, and 48-pin packagess |
| PTD6/TPM1CH0 | I/O | N | SWC | |
| PTE0 | I/O | N | SWC | Available only in 44- and 48-pin packages |
| PTE1 | I/O | N | SWC | Available only in 44- and 48-pin packages |
| PTE2 | I/O | N | SWC | Available only in 44- and 48-pin packages |
| PTE3 | I/O | N | SWC | Available only in 44- and 48-pin packages |
| PTE4 | I/O | N | SWC | Available only in 44- and 48-pin packages |
| PTE5 | I/O | N | SWC | Available only in 44- and 48-pin packages |
| PTE6 | I/O | N | SWC | Available only in 44- and 48-pin packages |
| PTE7 | I/O | N | SWC | Available only in 44- and 48-pin packages |

1. Unless otherwise indicated, all digital inputs have input hysteresis.
2. SWC is software-controlled pullup resistor, the register is associated with the respective port.
3. 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.
4. When these pins are configured as $\overline{\text{RESET}}$ or BKGD/MS pullup device is enabled.
5. When configured for the IRQ function, this pin will have a pullup device enabled when the IRQ is set for falling edge detection and a pulldown device enabled when the IRQ is set for rising edge detection.

# Chapter 3
# Modes of Operation

## 3.1 Introduction

The operating modes of the MC9S08RC/RD/RE/RG are described in this section. Entry into each mode, exit from each mode, and functionality while in each of the modes are described.

## 3.2 Features

- Active background mode for code development
- Wait mode:
  - CPU shuts down to conserve power
  - System clocks running
  - Full voltage regulation maintained
- Stop modes:
  - System clocks stopped; voltage regulator in standby
  - Stop1 — Full power down of internal circuits for maximum power savings
  - Stop2 — Partial power down of internal circuits, RAM remains operational
  - Stop3 — All internal circuits powered for fast recovery

## 3.3 Run Mode

This is the normal operating mode for the MC9S08RC/RD/RE/RG. This mode is selected when the BKGD/MS pin is high at the rising edge of reset. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at $FFFE:$FFFF after reset.

## 3.4 Active Background Mode

The active background mode functions are managed through the background debug controller (BDC) in the HCS08 core. The BDC, together with the on-chip 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 active background mode is entered, 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
  - BACKGROUND command
- Active background commands, which can only be executed while the MCU is in active background mode, 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 MC9S08RC/RD/RE/RG is shipped from the Freescale Semiconductor factory, the FLASH program memory is usually erased 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 the Development Support chapter.

## 3.5 Wait Mode

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

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

## 3.6 Stop Modes

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

Table 3-1 summarizes the behavior of the MCU in each of the stop modes.

**Table 3-1. Stop Mode Behavior**

| Mode | PDC | PPDC | CPU, Digital Peripherals, FLASH | RAM | OSC | ACMP | Regulator | I/O Pins | RTI |
|---|---|---|---|---|---|---|---|---|---|
| Stop1 | 1 | 0 | Off | Off | Off | Standby | Standby | Reset | Off |
| Stop2 | 1 | 1 | Off | Standby | Off | Standby | Standby | States held | Optionally on |
| Stop3 | 0 | Don't care | Standby | Standby | Off | Standby | Standby | States held | Optionally on |

### 3.6.1 Stop1 Mode

Stop1 mode provides the lowest possible standby power consumption by causing the internal circuitry of the MCU to be powered down. To enter stop1, the user must execute a STOP instruction with the PDC bit in SPMSC2 set and the PPDC bit clear. Stop1 can be entered only if the LVD reset is disabled (LVDRE = 0).

When the MCU is in stop1 mode, all internal circuits that are powered from the voltage regulator are turned off. The voltage regulator is in a low-power standby state, as are the OSC and ACMP.

Exit from stop1 is done by asserting any of the wakeup pins on the MCU: $\overline{\text{RESET}}$, IRQ, or KBI1, which have been enabled. IRQ and KBI pins are always active-low when used as wakeup pins in stop1 regardless of how they were configured before entering stop1.

Upon wakeup from stop1 mode, the MCU will start up as from a power-on reset (POR). The CPU will take the reset vector.

### 3.6.2 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 select entry into stop2 upon execution of a STOP instruction, the user must execute a STOP instruction with the PPDC and PDC bits in SPMSC2 set. Stop2 can be entered only if LVDRE = 0.

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

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 ACMP. 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 1 is written to PPDACK in SPMSC2.

Exit from stop2 is done by asserting any of the wakeup pins: $\overline{\text{RESET}}$, IRQ, or KBI1 that have been enabled, or through the real-time interrupt. IRQ and KBI1 pins are always active-low when used as wakeup pins in stop2 regardless of how they were configured before entering stop2. (KBI2 will not wake the MCU from stop2.)

Upon wakeup 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 1 is written to PPDACK in SPMSC2.

For pins that were configured as general-purpose I/O, the user must copy the contents of the I/O port registers, which have been saved in RAM, back 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 be in 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.

### 3.6.3 Stop3 Mode

Upon entering stop3 mode, all of the clocks in the MCU, including the oscillator itself, are halted. The OSC is turned off, the ACMP is disabled, and the voltage regulator is put in standby. 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 $\overline{\text{RESET}}$, any asynchronous interrupt pin that has been enabled, or through the real-time interrupt. The asynchronous interrupt pins are the IRQ or KBI1 and KBI2 pins.

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 (≈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.

### 3.6.4 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 the Development Support chapter 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. The MCU cannot enter either stop1 mode or stop2 mode if ENBDM is set.

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 either stop or wait mode. The BACKGROUND command can be used to wake the MCU from stop and enter active background mode if the ENBDM bit is set. After active background mode is entered, all background commands are available. Table 3-2 summarizes the behavior of the MCU in stop when entry into the active background mode is enabled.

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

| Mode | PDC | PPDC | CPU, Digital Peripherals, FLASH | RAM | OSC | ACMP | Regulator | I/O Pins | RTI |
|---|---|---|---|---|---|---|---|---|---|
| Stop3 | Don't care | Don't care | Standby | Standby | On | Standby | On | States held | Optionally on |

### 3.6.5 LVD Reset Enabled

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 reset is enabled in stop by setting the LVDRE bit in SPMSC1 when the CPU executes a STOP instruction, then the voltage regulator remains active during stop mode. If the user attempts to enter either stop1 or stop2 with the LVD reset enabled (LVDRE = 1) the MCU will instead enter stop3. Table 3-3 summarizes the behavior of the MCU in stop when LVD reset is enabled.

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

| Mode | PDC | PPDC | CPU, Digital Peripherals, FLASH | RAM | OSC | ACMP | Regulator | I/O Pins | RTI |
|---|---|---|---|---|---|---|---|---|---|
| Stop3 | Don't care | Don't care | Standby | Standby | On | Standby | On | States held | Optionally on |

### 3.6.6 On-Chip Peripheral Modules in Stop Mode

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.

### 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 pin states are latched before entering stop. Pin states remain latched until the PPDACK bit is written.
- If the MCU is configured to go into stop1 mode, all I/O pins are forced to their default reset state upon entry into stop.

### Memory

- All RAM and register contents are preserved while the MCU is in stop3 mode.
- All registers will be reset upon wakeup from stop2, but the contents of RAM are preserved. The user may save any memory-mapped register data into RAM before entering stop2 and restore the data upon exit from stop2.
- All registers will be reset upon wakeup from stop1 and the contents of RAM are not preserved. The MCU must be initialized as upon reset. The contents of the FLASH memory are non-volatile and are preserved in any of the stop modes.

**OSC** — In any of the stop modes, the OSC stops running.

**TPM** — When the MCU enters stop mode, the clock to the TPM module stops. The modules halt operation. If the MCU is configured to go into stop2 or stop1 mode, the TPM module will be reset upon wakeup from stop and must be reinitialized.

**ACMP** — When the MCU enters any stop mode, the ACMP will enter a low-power standby state**.** No compare operation will occur while in stop. If the MCU is configured to go into stop2 or stop1 mode, the ACMP will be reset upon wakeup from stop and must be reinitialized.

**KBI** — During stop3, the KBI pins that are enabled continue to function as interrupt sources. During stop1 or stop2, enabled KBI1 pins function as wakeup inputs. When functioning as a wakeup, a KBI pin is always active low regardless of how it was configured before entering stop1 or stop2.

**SCI** — When the MCU enters stop mode, the clock to the SCI module stops. The module halts operation. If the MCU is configured to go into stop2 or stop1 mode, the SCI module will be reset upon wakeup from stop and must be reinitialized.

**SPI** — When the MCU enters stop mode, the clock to the SPI module stops. The module halts operation. If the MCU is configured to go into stop2 or stop1 mode, the SPI module will be reset upon wakeup from stop and must be reinitialized.

**CMT** — When the MCU enters stop mode, the clock to the CMT module stops. The module halts operation. If the MCU is configured to go into stop2 or stop1 mode, the CMT module will be reset upon wakeup from stop and must be reinitialized.

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

# Chapter 4
# Memory

## 4.1 MC9S08RC/RD/RE/RG Memory Map

As shown in Figure 4-1, on-chip memory in the MC9S08RC/RD/RE/RG series of MCUs consists of RAM, FLASH program memory for nonvolatile data storage, and I/O and control/status registers. The registers are divided into three groups:

- Direct-page registers ($0000 through $0045 for 32K and 60K parts, and $0000 through $003F for 16K and 8K parts)
- High-page registers ($1800 through $182B)
- Nonvolatile registers ($FFB0 through $FFBF)

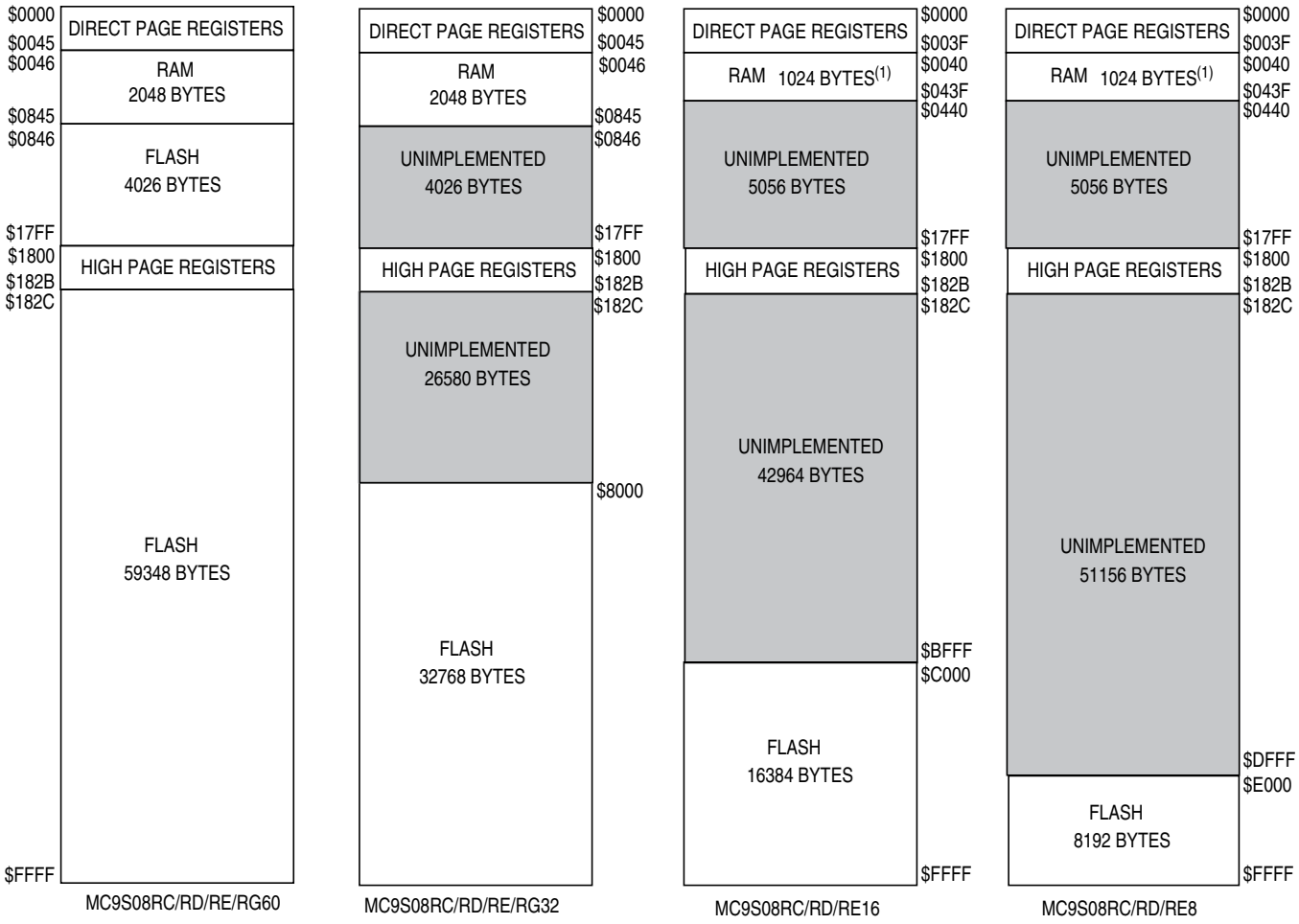



**Figure 4-1. MC9S08RC/RD/RE/RG Memory Map**

## 4.1.1 Reset and Interrupt Vector Assignments

Figure 4-2 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 MC9S08RC/RD/RE/RG. For more details about resets, interrupts, interrupt priority, and local interrupt mask controls, refer to the Chapter 5, “Resets, Interrupts, and System Configuration."

**Figure 4-2. Reset and Interrupt Vectors**

| Vector Number | Address (High/Low) | Vector | Vector Name |
|---|---|---|---|
| 16 through 31 | $FFC0:FFC1 ⇕ $FFDE:FFDF | Unused Vector Space (available for user program) | |
| 15 | $FFE0:FFE1 | SPI[1] | Vspi1 |
| 14 | $FFE2:FFE3 | RTI | Vrti |
| 13 | $FFE4:FFE5 | KBI2 | Vkeyboard2 |
| 12 | $FFE6:FFE7 | KBI1 | Vkeyboard1 |
| 11 | $FFE8:FFE9 | ACMP[2] | Vacmp1 |
| 10 | $FFEA:FFEB | CMT | Vcmt |
| 9 | $FFEC:FFED | SCI Transmit[3] | Vsci1tx |
| 8 | $FFEE:FFEF | SCI Receive[3] | Vsci1rx |
| 7 | $FFF0:FFF1 | SCI Error[3] | Vsci1err |
| 6 | $FFF2:FFF3 | TPM Overflow | Vtpm1ovf |
| 5 | $FFF4:FFF5 | TPM Channel 1 | Vtpm1ch1 |
| 4 | $FFF6:FFF7 | TPM Channel 0 | Vtpm1ch0 |
| 3 | $FFF8:FFF9 | IRQ | Virq |
| 2 | $FFFA:FFFB | Low Voltage Detect | Vlvd |
| 1 | $FFFC:FFFD | SWI | Vswi |
| 0 | $FFFE:FFFF | Reset | Vreset |

1. The SPI module is not included on the MC9S08RC/RD/RE devices. This vector location is unused for those devices.
2. The analog comparator (ACMP) module is not included on the MC9S08RD devices. This vector location is unused for those devices.
3. The SCI module is not included on the MC9S08RC devices. This vector location is unused for those devices.

## 4.2 Register Addresses and Bit Assignments

The registers in the MC9S08RC/RD/RE/RG are divided into these three groups:

- Direct-page registers are located within the first 256 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 $1800 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 $FFB0–$FFBF.

  Nonvolatile register locations include:

  — Three values that are loaded into working registers at reset

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

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

Direct-page registers can be accessed with efficient direct addressing mode instructions. Bit manipulation instructions can be used to access any bit in any direct-page register. Table 4-1 is a summary of all user-accessible direct-page registers and control bits.

The direct page registers in Table 4-1 can use the more efficient direct addressing mode, which requires only the lower byte of the address. Because of this, the lower byte of the address in column one is shown in bold text. In Table 4-2 and Table 4-3, the whole address in column one is shown in bold. In Table 4-1, Table 4-2, and Table 4-3, 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 4-1. Direct-Page Register Summary**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| $0000 | PTAD | PTAD7 | PTAD6 | PTAD5 | PTAD4 | PTAD3 | PTAD2 | PTAD1 | PTAD0 |
| $0001 | PTAPE | PTAPE7 | PTAPE6 | PTAPE5 | PTAPE4 | PTAPE3 | PTAPE2 | PTAPE1 | PTAPE0 |
| $0002 | Reserved | — | — | — | — | — | — | — | — |
| $0003 | PTADD | PTADD7 | PTADD6 | PTADD5 | PTADD4 | PTADD3 | PTADD2 | PTADD1 | PTADD0 |
| $0004 | PTBD | PTBD7 | PTBD6 | PTBD5 | PTBD4 | PTBD3 | PTBD2 | PTBD1 | PTBD0 |
| $0005 | PTBPE | PTBPE7 | PTBPE6 | PTBPE5 | PTBPE4 | PTBPE3 | PTBPE2 | PTBPE1 | PTBPE0 |
| $0006 | Reserved | — | — | — | — | — | — | — | — |
| $0007 | PTBDD | PTBDD7 | PTBDD6 | PTBDD5 | PTBDD4 | PTBDD3 | PTBDD2 | PTBDD1 | PTBDD0 |
| $0008 | PTCD | PTCD7 | PTCD6 | PTCD5 | PTCD4 | PTCD3 | PTCD2 | PTCD1 | PTCD0 |
| $0009 | PTCPE | PTCPE7 | PTCPE6 | PTCPE5 | PTCPE4 | PTCPE3 | PTCPE2 | PTCPE1 | PTCPE0 |
| $000A | Reserved | — | — | — | — | — | — | — | — |
| $000B | PTCDD | PTCDD7 | PTCDD6 | PTCDD5 | PTCDD4 | PTCDD3 | PTCDD2 | PTCDD1 | PTCDD0 |
| $000C | PTDD | 0 | PTDD6 | PTDD5 | PTDD4 | PTDD3 | PTDD2 | PTDD1 | PTDD0 |
| $000D | PTDPE | 0 | PTDPE6 | PTDPE5 | PTDPE4 | PTDPE3 | PTDPE2 | PTDPE1 | PTDPE0 |
| $000E | Reserved | — | — | — | — | — | — | — | — |
| $000F | PTDDD | 0 | PTDDD6 | PTDDD5 | PTDDD4 | PTDDD3 | PTDDD2 | PTDDD1 | PTDDD0 |
| $0010 | PTED | PTED7 | PTED6 | PTED5 | PTED4 | PTED3 | PTED2 | PTED1 | PTED0 |
| $0011 | PTEPE | PTEPE7 | PTEPE6 | PTEPE5 | PTEPE4 | PTEPE3 | PTEPE2 | PTEPE1 | PTEPE0 |
| $0012 | Reserved | — | — | — | — | — | — | — | — |
| $0013 | PTEDD | PTEDD7 | PTEDD6 | PTEDD5 | PTEDD4 | PTEDD3 | PTEDD2 | PTEDD1 | PTEDD0 |
| $0014 | KBI1SC | KBEDG7 | KBEDG6 | KBEDG5 | KBEDG4 | KBF | KBACK | KBIE | KBIMOD |
| $0015 | KBI1PE | KBIPE7 | KBIPE6 | KBIPE5 | KBIPE4 | KBIPE3 | KBIPE2 | KBIPE1 | KBIPE0 |
| $0016 | KBI2SC | 0 | 0 | 0 | 0 | KBF | KBACK | KBIE | KBIMOD |
| $0017 | KBI2PE | 0 | 0 | 0 | 0 | KBIPE3 | KBIPE2 | KBIPE1 | KBIPE0 |
| $0018 | SCI1BDH(1) | 0 | 0 | 0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| $0019 | SCI1BDL(1) | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| $001A | SCI1C1(1) | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| $001B | SCI1C2(1) | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| $001C | SCI1S1(1) | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| $001D | SCI1S2(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RAF |
| $001E | SCI1C3(1) | R8 | T8 | TXDIR | 0 | ORIE | NEIE | FEIE | PEIE |
| $001F | SCI1D(1) | R7/T7 | R6/T6 | R5/T5 | R4/T4 | R3/T3 | R2/T2 | R1/T1 | R0/T0 |
| $0020 | CMTCGH1 | PH7 | PH6 | PH5 | PH4 | PH3 | PH2 | PH1 | PH0 |
| $0021 | CMTCGL1 | PL7 | PL6 | PL5 | PL4 | PL3 | PL2 | PL1 | PL0 |
| $0022 | CMTCGH2 | SH7 | SH6 | SH5 | SH4 | SH3 | SH2 | SH1 | SH0 |
| $0023 | CMTCGL2 | SL7 | SL6 | SL5 | SL4 | SL3 | SL2 | SL1 | SL0 |
| $0024 | CMTOC | IROL | CMTPOL | IROPEN | 0 | 0 | 0 | 0 | 0 |
| $0025 | CMTMSC | EOCF | CMTDIV1 | CMTDIV0 | EXSPC | BASE | FSK | EOCIE | MCGEN |
| $0026 | CMTCMD1 | MB15 | MB14 | MB13 | MB12 | MB11 | MB10 | MB9 | MB8 |
| $0027 | CMTCMD2 | MB7 | MB6 | MB5 | MB4 | MB3 | MB2 | MB1 | MB0 |
| $0028 | CMTCMD3 | SB15 | SB14 | SB13 | SB12 | SB11 | SB10 | SB9 | SB8 |
| $0029 | CMTCMD4 | SB7 | SB6 | SB5 | SB4 | SB3 | SB2 | SB1 | SB0 |

**Table 4-1. Direct-Page Register Summary (continued)**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| $002A | IRQSC | 0 | 0 | IRQEDG | IRQPE | IRQF | IRQACK | IRQIE | IRQMOD |
| $002B | ACMP1SC[2] | ACME | ACBGS | ACF | ACIE | ACO | — | ACMOD1 | ACMOD0 |
| $002C–$002F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| $0030 | TPM1SC | TOF | TOIE | CPWMS | CLKSB | CLKSA | PS2 | PS1 | PS0 |
| $0031 | TPM1CNTH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0032 | TPM1CNTL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0033 | TPM1MODH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0034 | TPM1MODL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0035 | TPM1C0SC | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | 0 | 0 |
| $0036 | TPM1C0VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0037 | TPM1C0VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0038 | TPM1C1SC | CH1F | CH1IE | MS1B | MS1A | ELS1B | ELS1A | 0 | 0 |
| $0039 | TPM1C1VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $003A | TPM1C1VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $003B–$003F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| $0040 | **SPI1C1**[3] | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| $0041 | SPI1C2[3] | 0 | 0 | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| $0042 | **SPI1BR**[3] | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| $0043 | **SPI1S**[3] | SPRF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| $0044 | Reserved | — | — | — | — | — | — | — | — |
| $0045 | **SPI1D**[3] | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

1. The SCI module is not included on the MC9S08RC devices. This is a reserved location for those devices.
2. The analog comparator (ACMP) module is not included on the MC9S08RD devices. This is a reserved location for those devices.
3. The SPI module is not included on the MC9S08RC/RD/RE devices. These are reserved locations on the 32K and 60K versions of these devices. The address range $0040–$004F are RAM locations on the 16K and 8K devices. There are no MC9S08RG8/16 devices.

High-page registers, shown in Table 4-2, 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 $1800.

**Table 4-2. High-Page Register Summary**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| $1800 | SRS | POR | PIN | COP | ILOP | ILAD[1] | 0 | LVD | 0 |
| $1801 | SBDFR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BDFR |
| $1802 | SOPT | 7 | COPT | STOPE | — | 0 | 0 | BKGDPE | RSTPE |
| $1803–$1804 | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| $1805 | Reserved | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $1806 | SDIDH | REV3 | REV2 | REV1 | REV0 | ID11 | ID10 | ID9 | ID8 |
| $1807 | SDIDL | ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |
| $1808 | SRTISC | RTIF | RTIACK | RTICLKS | RTIE | 0 | RTIS2 | RTIS1 | RTIS0 |

**Table 4-2. High-Page Register Summary (continued)**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| $1809 | SPMSC1 | LVDF | LVDACK | LVDIE | SAFE | LVDRE | — | — | — |
| $180A | SPMSC2 | LVWF | LVWACK | 0 | 0 | PPDF | PPDACK | PDC | PPDC |
| $180B–$180F | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |
| $1810 | DBGCAH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $1811 | DBGCAL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $1812 | DBGCBH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $1813 | DBGCBL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $1814 | DBGFH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $1815 | DBGFL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $1816 | DBGC | DBGEN | ARM | TAG | BRKEN | RWA | RWAEN | RWB | RWBEN |
| $1817 | DBGT | TRGSEL | BEGIN | 0 | 0 | TRG3 | TRG2 | TRG1 | TRG0 |
| $1818 | DBGS | AF | BF | ARMF | 0 | CNT3 | CNT2 | CNT1 | CNT0 |
| $1819–$181F | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |
| $1820 | FCDIV | DIVLD | PRDIV8 | DIV5 | DIV4 | DIV3 | DIV2 | DIV1 | DIV0 |
| $1821 | FOPT | KEYEN | FNORED | 0 | 0 | 0 | 0 | SEC01 | SEC00 |
| $1822 | Reserved | — | — | — | — | — | — | — | — |
| $1823 | FCNFG | 0 | 0 | KEYACC | 0 | 0 | 0 | 0 | 0 |
| $1824 | FPROT | FPOPEN | FPDIS | FPS2 | FPS1 | FPS0 | 0 | 0 | 0 |
| $1825 | FSTAT | FCBEF | FCCF | FPVIOL | FACCERR | 0 | FBLANK | 0 | 0 |
| $1826 | FCMD | FCMD7 | FCMD6 | FCMD5 | FCMD4 | FCMD3 | FCMD2 | FCMD1 | FCMD0 |
| $1827–$182B | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |

1. The ILAD bit is only present on 16K and 8K versions of the devices.

Nonvolatile FLASH registers, shown in Table 4-3, are located in the FLASH memory. These registers include an 8-byte backdoor key that optionally can be used to gain access to secure memory resources. During reset events, the contents of NVPROT and NVOPT in the nonvolatile register area of the FLASH memory are transferred into corresponding FPROT and FOPT working registers in the high-page registers to control security and block protection options.

**Table 4-3. Nonvolatile Register Summary**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| $FFB0–$FFB7 | NVBACKKEY | 8-Byte Comparison Key | | | | | | | |
| $FFB8–$FFBC | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |
| $FFBD | NVPROT | FPOPEN | FPDIS | FPS2 | FPS1 | FPS0 | 0 | 0 | 0 |
| $FFBE | Reserved | — | — | — | — | — | — | — | — |
| $FFBF | NVOPT | KEYEN | FNORED | 0 | 0 | 0 | 0 | SEC01 | SEC00 |

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

## 4.3 RAM

The MC9S08RC/RD/RE/RG includes static RAM. The locations in RAM below $0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit-manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

The RAM retains data when the MCU is in low-power wait, stop2, or stop3 mode. At power-on or after wakeup from stop1, 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 $00FF. In the MC9S08RC/RD/RE/RG, it is usually best to reinitialize 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/D.

### 4.4.1 Features

Features of the FLASH memory include:

- FLASH Size
  - MC9S08RC/RD/RE/RG60 — 63374 bytes (124 pages of 512 bytes each)
  - MC9S08RC/RD/RE/RG32 — 32768 bytes (64 pages of 512 bytes each)
  - MC9S08RC/RD/RE16 — 16384 bytes (32 pages of 512 bytes each)
  - MC9S08RC/RD/RE8 — 8192 bytes (16 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for FLASH and RAM
- Auto power-down for low-frequency read accesses

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

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

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

1. Excluding start/end overhead

### 4.4.3 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.

2. Write the command code for the desired command to FCMD. The five valid commands are blank check ($05), byte program ($20), burst program ($25), page erase ($40), and mass erase ($41). The command code is latched into the command buffer.
3. Write a 1 to the FCBEF bit in FSTAT to clear FCBEF and launch the command (including its address and data information).

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

A strictly monitored procedure must be adhered to, 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 4-3 is a flowchart for executing all of the commands except for burst programming. The FCDIV register must be initialized before using any FLASH commands. This must be done only once following a reset.

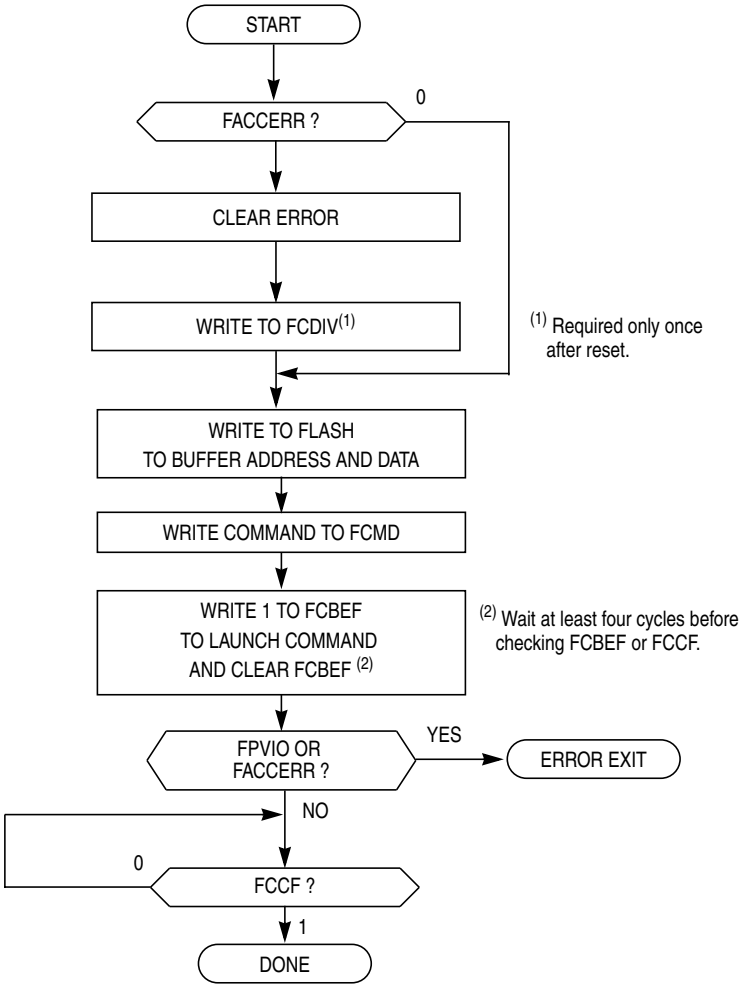



**Figure 4-3. FLASH Program and Erase Flowchart**

### 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 remains enabled after completion of the burst program operation if the following two conditions are met:

- The new burst program command has been queued before the current program operation completes.
- 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 where 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.

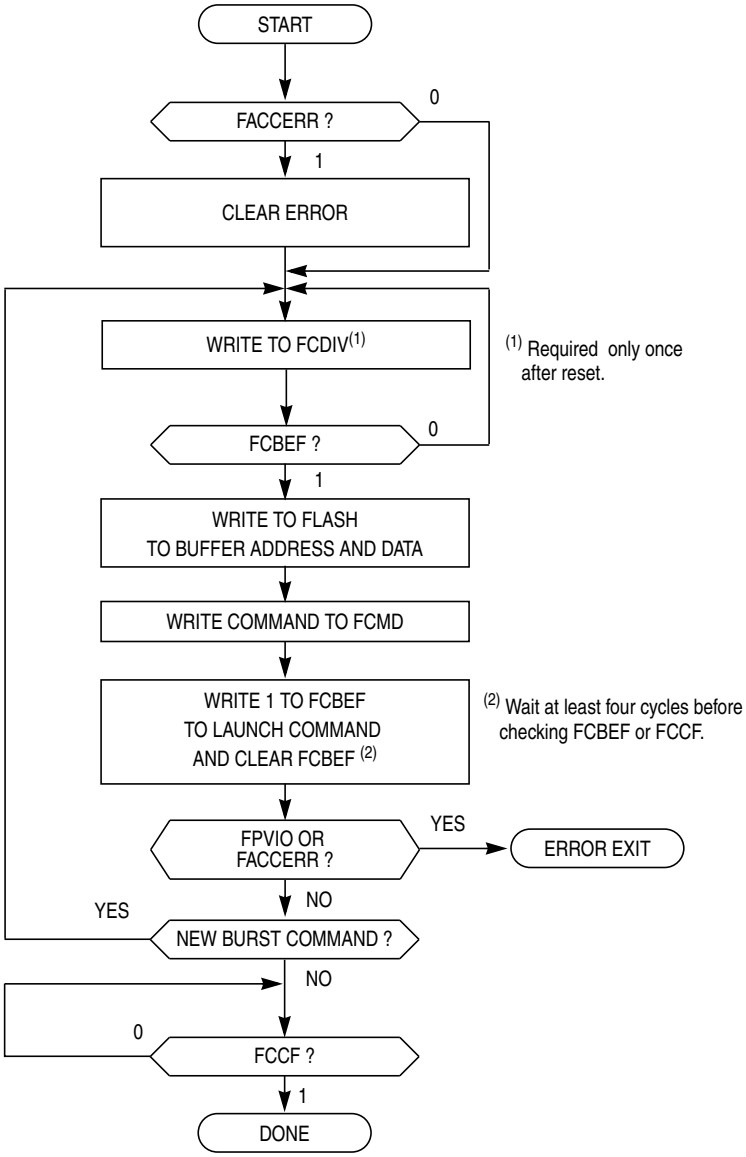



**Figure 4-4. FLASH Burst Program Flowchart**

## 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 ($05, $20, $25, $40, or $41) to FCMD
- Accessing (read or write) any FLASH control register other than the write to FSTAT (to clear FCBEF and launch the command) after writing the command to FCMD
- The MCU enters stop mode while a program or erase command is in progress (The command is aborted.)
- Writing the byte program, burst program, or page erase command code ($20, $25, or $40) 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

Block protection prevents program or erase changes for FLASH memory locations in a designated address range. Mass erase is disabled when any block of FLASH is protected. The MC9S08RC/RD/RE/RG allows a block of memory at the end of FLASH, and/or the entire FLASH memory to be block protected. A disable control bit and a 3-bit control field, for each of the blocks, allows the user to independently set the size of these blocks. A separate control bit allows block protection of the entire FLASH memory array. All seven of these control bits are located in the FPROT register (see Section 4.6.4, “FLASH Protection Register (FPROT and NVPROT)“).

At reset, the high-page register (FPROT) is loaded with the contents of the NVPROT location that is in the nonvolatile register block of the FLASH memory. The value in FPROT cannot be changed directly from application software so a runaway program cannot alter the block protection settings. If the last 512 bytes of FLASH (which includes the NVPROT register) is protected, the application program cannot alter the block protection settings (intentionally or unintentionally). The FPROT control bits can be written by background debug commands to allow a way to erase a protected FLASH memory.

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 during an erase and reprogram operation.

### 4.4.7 Vector Redirection

Whenever any block protection is enabled, the reset and interrupt vectors will be protected. For this reason, a mechanism for redirecting vector reads is provided. Vector redirection allows users to modify interrupt vector information without unprotecting bootloader and reset vector space. 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 $FFBD. All of the interrupt vectors (memory locations $FFC0–$FFFD) are redirected, while the reset vector ($FFFE:FFFF) is not.

For example, if 512 bytes of FLASH are protected, the protected address region is from $FE00 through $FFFF. The interrupt vectors ($FFC0–$FFFD) are redirected to the locations $FDC0–$FDFD. Now, if an SPI interrupt is taken for instance, the values in the locations $FDE0:FDE1 are used for the vector instead of the values in the locations $FFE0:FFE1. This allows the user to reprogram the unprotected portion of the FLASH with new program code including new interrupt vector values while leaving the protected area, which includes the default vector locations, unchanged.

## 4.5 Security

The MC9S08RC/RD/RE/RG 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 while the other three combinations engage security. Notice the erased state (1:1) makes the MCU secure. During development, whenever the FLASH is erased, it is good practice to immediately program the SEC00 bit to 0 in NVOPT so SEC01:SEC00 = 1:0. This would allow the MCU to remain unsecured after a subsequent reset.

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

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

1. Writing 1 to KEYACC in the FCNFG register. This makes the FLASH module interpret writes to the backdoor comparison key locations (NVBACKKEY through NVBACKKEY+7) as values to be compared against the key rather than as the first step in a FLASH program or erase command.

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

The security key can be written only from RAM, so it cannot be entered through background commands without the cooperation of a secure user program. The FLASH memory cannot be accessed by read operations while KEYACC is set.

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 performing 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 that 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 4-2 and Table 4-3 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.

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

= Unimplemented or Reserved

**Figure 4-5. FLASH Clock Divider Register (FCDIV)**

**Table 4-5. FCDIV Field Descriptions**

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

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

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

Table 4-6 shows the appropriate values for PRDIV8 and DIV5:DIV0 for selected bus frequencies.

**Table 4-6. FLASH Clock Divider Settings**

| $f_{Bus}$ | PRDIV8 (Binary) | DIV5:DIV0 (Decimal) | $f_{FCLK}$ | Program/Erase Timing Pulse (5 µs Min, 6.7 µs Max) |
|---|---|---|---|---|
| 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.

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

= Unimplemented or Reserved

**Figure 4-6. FLASH Options Register (FOPT)**

**Table 4-7. FOPT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>KEYEN | **Backdoor Key Mechanism Enable** — When this bit is 0, the backdoor key mechanism cannot be used to disengage security. The backdoor key mechanism is accessible only from user (secured) firmware. BDM commands cannot be used to write key comparison values that would unlock the backdoor key. For more detailed information about the backdoor key mechanism, refer to Section 4.5, “Security."<br>0 No backdoor key access allowed.<br>1 If user firmware writes an 8-byte value that matches the nonvolatile backdoor key (NVBACKKEY through NVBACKKEY+7 in that order), security is temporarily disengaged until the next MCU reset. |
| 6<br>FNORED | **Vector Redirection Disable** — When this bit is 1, then vector redirection is disabled.<br>0 Vector redirection enabled.<br>1 Vector redirection disabled. |
| 1:0<br>SEC0[1:0] | **Security State Code** — This 2-bit field determines the security state of the MCU as shown below. When the MCU is secure, the contents of RAM and FLASH memory cannot be accessed by instructions from any unsecured source including the background debug interface. For more detailed information about security, refer to Section 4.5, “Security.”<br>00 Secure<br>01 Secure<br>10 Unsecured<br>11 Secure<br>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.

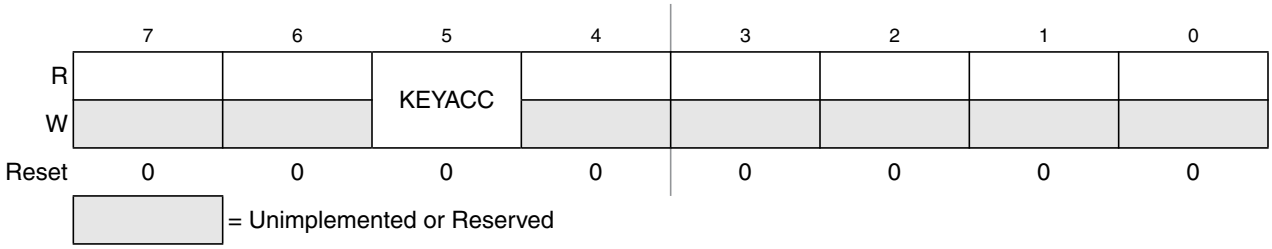



**Figure 4-7. FLASH Configuration Register (FCNFG)**

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

| Field | Description |
|---|---|
| 5<br>KEYACC | **Enable Writing of Access Key** — This bit enables writing of the backdoor comparison key. For more detailed information about the backdoor key mechanism, refer to Section 4.5, “Security."<br>0 Writes to $FFB0–$FFB7 are interpreted as the start of a FLASH programming or erase command.<br>1 Writes to NVBACKKEY ($FFB0–$FFB7) are interpreted as comparison key writes.<br>Reads of the FLASH return invalid data. |

## 4.6.4 FLASH Protection Register (FPROT and NVPROT)

During reset, the contents of the nonvolatile location NVPROT is copied from FLASH into FPROT. Bits 0, 1, and 2 are not used and each always reads as 0. This register may be read at any time, but user program writes have no meaning or effect. Background debug commands can write to FPROT at $1824.

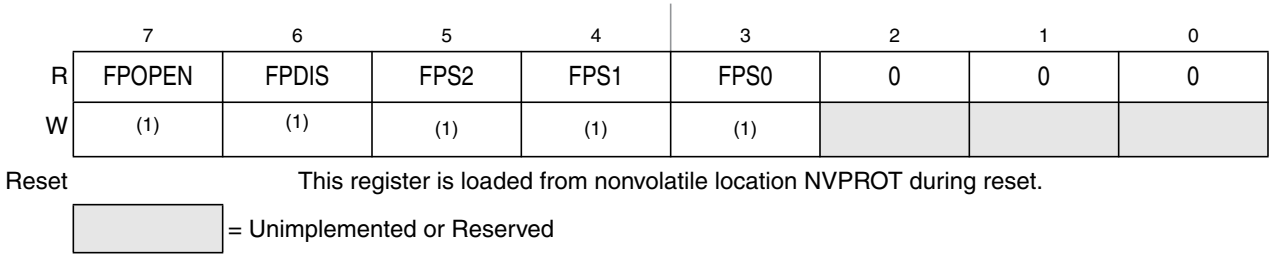



**Figure 4-8. FLASH Protection Register (FPROT)**

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

**Table 4-9. FPROT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FPOPEN | **Open Unprotected FLASH for Program/Erase**<br>0 Entire FLASH memory is block protected (no program or erase allowed).<br>1 Any FLASH location, not otherwise block protected or secured, may be erased or programmed. |
| 6<br>FPDIS | **FLASH Protection Disable**<br>0 FLASH block specified by FPS2:FPS0 is block protected (program and erase not allowed).<br>1 No FLASH block is protected. |
| 5:3<br>FPS[2:0] | **FLASH Protect Size Selects** — When FPDIS = 0, this 3-bit field determines the size of a protected block of FLASH locations at the high address end of the FLASH (see Table 4-10 and Table 4-11). Protected FLASH locations cannot be erased or programmed. |

**Table 4-10. High Address Protected Block for 32K and 60K Versions**

| FPS2:FPS1:FPS0 | Protected Address Range | Protected Block Size | Redirected Vectors[1] |
|---|---|---|---|
| 0:0:0 | $FE00–$FFFF | 512 bytes | $FDC0–$FDFD[2] |
| 0:0:1 | $FC00–$FFFF | 1 Kbytes | $FBC0–$FBFD |
| 0:1:0 | $F800–$FFFF | 2 Kbytes | $F7C0–$F7FD |
| 0:1:1 | $F000–$FFFF | 4 Kbytes | $EFC0–$EFFD |
| 1:0:0 | $E000–$FFFF | 8 Kbytes | $DFC0–$DFFD |
| 1:0:1 | $C000–$FFFF | 16 Kbytes | $BFC0–$BFFD |
| 1:1:0[3] | $8000–$FFFF | 32 Kbytes | $7FC0–$7FFD |
| 1:1:1[3] | $8000–$FFFF | 32 Kbytes | $7FC0–$7FFD |

1. No redirection if FPOPEN = 0, or FNORED = 1.
2. Reset vector is not redirected.
3. Use for 60K version only. When protecting all of 32K version memory, use FPOPEN = 0.

**Table 4-11. High Address Protected Block for 8K and 16K Version**

| FPS2:FPS1:FPS0 | Protected Address Range | Protected Block Size | Redirected Vectors[1] |
|---|---|---|---|
| 0:0:0 | $FE00–$FFFF | 512 bytes | $FDC0–$FDFD[2] |
| 0:0:1 | $FC00–$FFFF | 1 Kbytes | $FBC0–$FBFD |
| 0:1:0 | $F800–$FFFF | 2 Kbytes | $F7C0–$F7FD |
| 0:1:1 | $F000–$FFFF | 4 Kbytes | $EFC0–$EFFD |
| 1:0:0[3] | $E000–$FFFF | 8 Kbytes | $DFC0–$DFFD |
| 1:0:1[3] | $E000–$FFFF | 8 Kbytes | $DFC0–$DFFD |
| 1:1:0[3] | $E000–$FFFF | 8 Kbytes | $DFC0–$DFFD |
| 1:1:1[3] | $E000–$FFFF | 8 Kbytes | $DFC0–$DFFD |

1. No redirection if FPOPEN = 0, or FNORED = 1.
2. Reset vector is not redirected.
3. Use for 60K version only. When protecting all of 32K version memory, use FPOPEN = 0.

## 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.

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

= Unimplemented or Reserved

**Figure 4-9. FLASH Status Register (FSTAT)**

**Table 4-12. FSTAT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>FCBEF | **FLASH Command Buffer Empty Flag** — The FCBEF bit is used to launch commands. It also indicates that the command buffer is empty so that a new command sequence can be executed when performing burst programming. The FCBEF bit is cleared by writing a 1 to it or when a burst program command is transferred to the array for programming. Only burst program commands can be buffered.<br>0 Command buffer is full (not ready for additional commands).<br>1 A new burst program command may be written to the command buffer. |
| 6<br>FCCF | **FLASH Command Complete Flag** — FCCF is set automatically when the command buffer is empty and no command is being processed. FCCF is cleared automatically when a new command is started (by writing 1 to FCBEF to register a command). Writing to FCCF has no meaning or effect.<br>0 Command in progress<br>1 All commands complete |
| 5<br>FPVIOL | **Protection Violation Flag** — FPVIOL is set automatically when FCBEF is cleared to register a command that attempts to erase or program a location in a protected block (the erroneous command is ignored). FPVIOL is cleared by writing a 1 to FPVIOL.<br>0 No protection violation.<br>1 An attempt was made to erase or program a protected location. |
| 4<br>FACCERR | **Access Error Flag** — FACCERR is set automatically when the proper command sequence is not followed exactly (the erroneous command is ignored), if a program or erase operation is attempted before the FCDIV register has been initialized, or if the MCU enters stop while a command was in progress. For a more detailed discussion of the exact actions that are considered access errors, see Section 4.4.5, “Access Errors." FACCERR is cleared by writing a 1 to FACCERR. Writing a 0 to FACCERR has no meaning or effect.<br>0 No access error.<br>1 An access error has occurred. |
| 2<br>FBLANK | **FLASH Verified as All Blank (erased) Flag** — FBLANK is set automatically at the conclusion of a blank check command if the entire FLASH array was verified to be erased. FBLANK is cleared by clearing FCBEF to write a new valid command. Writing to FBLANK has no meaning or effect.<br>0 After a blank check command is completed and FCCF = 1, FBLANK = 0 indicates the FLASH array is not completely erased.<br>1 After a blank check command is completed and FCCF = 1, FBLANK = 1 indicates the FLASH array is completely erased (all $FF). |

### 4.6.6 FLASH Command Register (FCMD)

Only four 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 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>FCMD[7:0] | See Table 4-14 for FLASH commands. |

**Table 4-14. FLASH Commands**

| Command | FCMD | Equate File Label |
|---|---|---|
| Blank check | $05 | mBlank |
| Byte program | $20 | mByteProg |
| Byte program – burst mode | $25 | mBurstProg |
| Page erase (512 bytes/page) | $40 | mPageErase |
| Mass erase (all FLASH) | $41 | 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 section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the MC9S08RC/RD/RE/RG. Some interrupt sources from peripheral modules are discussed in greater detail within other sections of this data sheet. This section 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 having 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 reset pin with enable ($\overline{\text{RESET}}$)
  - COP watchdog with enable and two timeout choices
  - Illegal opcode
  - Illegal address (on 16K and 8K devices)
  - Serial command from a background debug host
- Reset status register (SRS) to indicate source of most recent reset; flag to indicate stop2 (partial power down) mode recovery (PPDF)
- Separate interrupt vectors for each module (reduces polling overhead) (see Table 5-1)
- Safe state for protecting the MCU in low-voltage condition

## 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 ($FFFE:$FFFF). 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 until the user program has a chance to initialize the stack pointer (SP) and system control settings. SP is forced to $00FF at reset.

The MC9S08RC/RD/RE/RG has these sources for reset:

- Power-on reset (POR)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect
- Illegal address (16K and 8K devices only)
- Background debug forced reset
- The reset pin ($\overline{\text{RESET}}$)

Each of these sources, with the exception of the background debug forced reset, has an associated bit in the system reset status register. Whenever the MCU enters reset, the reset pin is driven low for 34 internal bus cycles where the internal bus frequency is one-half the OSC frequency. After the 34 cycles are completed, the pin is released and will be pulled up by the internal pullup resistor, unless it is held low externally. After the pin is released, it is sampled after another 38 cycles to determine whether the reset pin is the cause of the MCU reset.

## 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 timer periodically. If the application program gets lost and fails to reset the COP before it times out, a system reset is generated to force the system back to a known starting point. The COP watchdog is enabled by the COPE bit in SOPT (see Section 5.8.4, “System Options Register (SOPT),” for additional information). The COP timer 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 timer.

After any reset, the COP timer is enabled. This provides a reliable way to detect code that is not executing as intended. If the COP watchdog is not used in an application, it can be disabled by clearing the COPE bit in the write-once SOPT register. Also, the COPT bit can be used to choose one of two timeout periods ($2^{18}$ or $2^{20}$ cycles of the bus rate clock). Even if the application will use the reset default settings in COPE and COPT, the user must write to write-once SOPT during reset initialization to lock in the settings. That way, they cannot be changed accidentally if the application program gets lost.

The write to SRS that services (clears) the COP timer 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.

When the MCU is in active background mode, the COP timer is temporarily disabled.

## 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 was 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 logic 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 uses 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 that 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 Family, 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 just before the RTI that is used to return from the ISR.

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

### 5.5.1 Interrupt Stack Frame

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

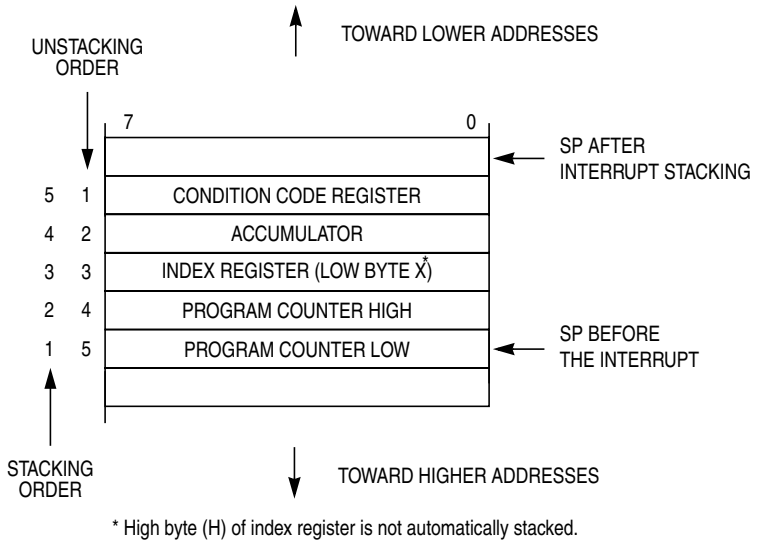



**Figure 5-1. Interrupt Stack Frame**

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

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

## 5.5.2 External Interrupt Request (IRQ) Pin

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

### 5.5.2.1 Pin Configuration Options

The IRQ pin enable (IRQPE) control bit in the IRQSC register must be 1 in order for the IRQ pin to act as the interrupt request (IRQ) input. When the pin is configured 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 merely sets the IRQF flag (which can be polled by software).

When the IRQ pin is configured to detect rising edges, an optional pulldown resistor is available rather than a pullup resistor. 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.

#### 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 this edge detection mode, the IRQF status flag becomes set when an edge is detected (when the IRQ pin changes from the deasserted to the asserted level), but the flag is continuously set (and cannot be cleared) as long as the IRQ pin remains at the asserted level.

### 5.5.3 Interrupt Vectors, Sources, and Local Masks

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

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

**Table 5-1. Vector Summary**

| Vector Priority | Vector Number | Address (High/Low) | Vector Name | Module | Source | Enable | Description |
|---|---|---|---|---|---|---|---|
| Lower | 16 through 31 | $FFC0/FFC1 through $FFDE/FFDF | Unused Vector Space (available for user program) | | | | |
| | 15 | $FFE0/FFE1 | Vspi1 | SPI[1] | SPIF<br>MODF<br>SPTEF | SPIE<br>SPIE<br>SPTIE | SPI |
| | 14 | $FFE2/FFE3 | Vrti | System control | RTIF | RTIE | Real-time interrupt |
| | 13 | $FFE4/FFE5 | Vkeyboard2 | KBI2 | KBF | KBIE | Keyboard 2 pins |
| | 12 | $FFE6/FFE7 | Vkeyboard1 | KBI1 | KBF | KBIE | Keyboard 1 pins |
| | 11 | $FFE8/FFE9 | Vacmp1 | ACMP[2] | ACF | ACIE | ACMP compare |
| | 10 | $FFEA/FFEB | Vcmt | CMT | EOCF | EOCIE | CMT |
| | 9 | $FFEC/FFED | Vsci1tx | SCI[3] | TDRE<br>TC | TIE<br>TCIE | SCI transmit |
| | 8 | $FFEE/FFEF | Vsci1rx | SCI[3] | IDLE<br>RDRF | ILIE<br>RIE | SCI receive |
| | 7 | $FFF0/FFF1 | Vsci1err | SCI[3] | OR<br>NF<br>FE<br>PF | ORIE<br>NFIE<br>FEIE<br>PFIE | SCI error |
| | 6 | $FFF2/FFF3 | Vtpm1ovf | TPM | TOF | TOIE | TPM overflow |
| | 5 | $FFF4/FFF5 | Vtpm1ch1 | TPM | CH1F | CH1IE | TPM channel 1 |
| | 4 | $FFF6/FFF7 | Vtpm1ch0 | TPM | CH0F | CH0IE | TPM channel 0 |
| | 3 | $FFF8/FFF9 | Virq | IRQ | IRQF | IRQIE | IRQ pin |
| | 2 | $FFFA/FFFB | Vlvd | System control | LVDF | LVDIE | Low-voltage detect |
| | 1 | $FFFC/FFFD | Vswi | Core | SWI Instruction | — | Software interrupt |
| Higher | 0 | $FFFE/FFFF | Vreset | System control | COP<br>LVD<br>$\overline{\text{RESET}}$ pin<br>Illegal opcode<br>Illegal address | COPE<br>LVDRE<br>RSTPE<br>—<br>— | Watchdog timer<br>Low-voltage detect<br>External pin<br>Illegal opcode<br>Illegal address |

1. The SPI module is not included on the MC9S08RC/RD/RE devices. This vector location is unused for those devices.
2. The analog comparator (ACMP) module is not included on the MC9S08RD devices. This vector location is unused for those devices.
3. The SCI module is not included on the MC9S08RC devices. This vector location is unused for those devices.

## 5.6 Low-Voltage Detect (LVD) System

The MC9S08RC/RD/RE/RG 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, an LVD circuit with flag bits for warning and detection, and a mechanism for entering a system safe state following an LVD interrupt. The LVD circuit can be configured to generate an interrupt or a reset when low supply voltage has been detected.

### 5.6.1 Power-On Reset Operation

When power is initially applied to the MCU, or when the supply voltage drops below the $V_{POR}$ 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 $V_{LVD}$ 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. This is done by setting LVDRE to 1. LVDRE is a write-once bit that is set following a POR and is unaffected by other resets. When LVDRE = 1, setting the SAFE bit has no effect. After an LVD reset has occurred, the LVD system will hold the MCU in reset until the supply voltage is above the $V_{LVD}$ level. The LVD bit in the SRS register is set following either an LVD reset or POR.

### 5.6.3 LVD Interrupt and Safe State Operation

When the voltage on the supply pin $V_{DD}$ drops below $V_{LVD}$ and the LVD circuit is configured for interrupt operation (LVDIE is set and LVDRE is clear), an LVD interrupt will occur. The LVD trip point is set above the minimum voltage at which the MCU can reliably operate, but the supply voltage may still be dropping. It is recommended that the user place the MCU in the safe state as soon as possible following a LVD interrupt. For systems where the supply voltage may drop so rapidly that the MCU may not have time to service the LVD interrupt and enter the safe state, it is recommended that the LVD be configured to generate a reset. The safe state is entered by executing a STOP instruction with the SAFE bit in the system power management status and control 1 (SPMSC1) register set while in a low voltage condition (LVDF = 1).

After the LVD interrupt has occurred, the user may configure the system to block all interrupts, resets, or wakeups by writing a 1 to the SAFE bit. While SAFE =1 and $V_{DD}$ is below $V_{REARM}$ all interrupts, resets, and wakeups are blocked. After $V_{DD}$ is above $V_{REARM}$, the SAFE bit is ignored (the SAFE bit will still read a  1). After setting the SAFE bit, the MCU must be put into either the stop3 or stop2 mode before the supply voltage drops below the minimum operating voltage of the MCU. The supply voltage may now drop to a level just above the POR trip point and then restored to a level above $V_{REARM}$ and the MCU state (in the case of stop3) and RAM contents will be preserved. When the supply voltage has been restored, interrupts, resets, and wakeups are then unblocked. When the MCU has recovered from stop mode, the SAFE bit should be cleared.

### 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 low-voltage detect voltage. The LVW does not have an interrupt associated with it. However, the FLASH memory cannot be reliably programmed or erased below the $V_{LVW}$ level, so the status of the LVWF bit in the system power management status and control 2 (SPMSC2) register must be checked before initiating any FLASH program or erase operation.

## 5.7 Real-Time Interrupt (RTI)

The real-time interrupt function can be used to generate periodic interrupts based on a multiple of the source clock's period. The RTI has two source clock choices, the external clock input or the RTI's own internal clock. The RTI can be used in run, wait, stop2, and stop3 modes. It is not available in stop1 mode.

In run and wait modes, only the external clock can be used as the RTI's clock source. In stop2 mode, only the internal RTI clock can be used. In stop3, either the external clock or internal RTI clock can be used. When using the external oscillator in stop3 mode, it must be enabled in stop (OSCSTEN = 1) and configured for low bandwidth operation (RANGE = 0).

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 select one of seven RTI periods. The RTI has a local interrupt enable, RTIE, to allow masking of the real-time interrupt. The module can be disabled by writing 0:0:0 to RTIS2:RTIS1:RTIS0 in which case the clock source input is disabled and no interrupts will be generated. See Section 5.8.6, "System Real-Time Interrupt Status and Control Register (SRTISC)," for detailed information about this register.

## 5.8 Reset, Interrupt, and System Control Registers and Control Bits

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

Refer to the direct-page register summary in the Memory chapter 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."

### 5.8.1 Interrupt Pin Request Status and Control Register (IRQSC)

This direct page register includes two unimplemented bits that always read 0, four read/write bits, one read-only status bit, and one write-only bit. These bits are used to configure the IRQ function, report status, and acknowledge IRQ events.

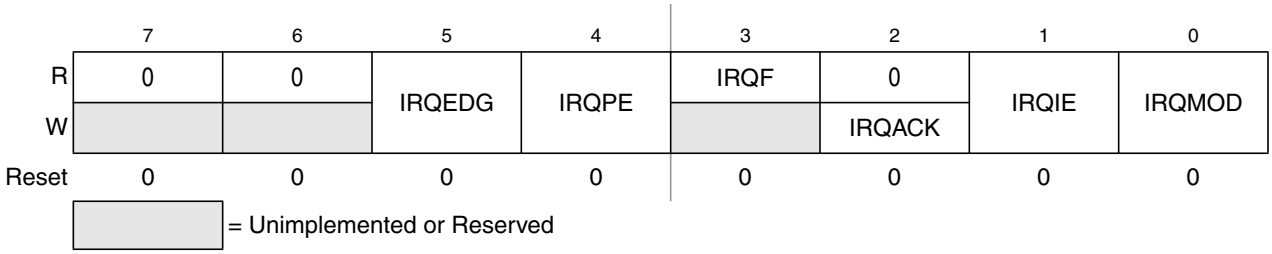



**Figure 5-2. Interrupt Request Status and Control Register (IRQSC)**

**Table 5-2. IRQSC Field Descriptions**

| Field | Description |
|---|---|
| 5<br>IRQEDG | **Interrupt Request (IRQ) Edge Select** — This read/write control bit is used to select the polarity of edges or levels on the IRQ pin that cause IRQF to be set. The IRQMOD control bit determines whether the IRQ pin is sensitive to both edges and levels or only edges. When the IRQ pin is enabled as the IRQ input and is configured to detect rising edges, the optional pullup resistor is reconfigured as an optional pulldown resistor.<br>0 IRQ is falling edge or falling edge/low-level sensitive.<br>1 IRQ is rising edge or rising edge/high-level sensitive. |
| 4<br>IRQPE | **IRQ Pin Enable** — This read/write control bit enables the IRQ pin function. When this bit is set the IRQ pin can be used as an interrupt request. Also, when this bit is set, either an internal pull-up or an internal pull-down resistor is enabled depending on the state of the IRQMOD bit.<br>0 IRQ pin function is disabled.<br>1 IRQ pin function is enabled. |
| 3<br>IRQF | **IRQ Flag** — This read-only status bit indicates when an interrupt request event has occurred.<br>0 No IRQ request.<br>1 IRQ event detected. |
| 2<br>IRQACK | **IRQ Acknowledge** — This write-only bit is used to acknowledge interrupt request events (write 1 to clear IRQF). Writing 0 has no meaning or effect. Reads always return 0. If edge-and-level detection is selected (IRQMOD = 1), IRQF cannot be cleared while the IRQ pin remains at its asserted level. |
| 1<br>IRQIE | **IRQ Interrupt Enable** — This read/write control bit determines whether IRQ events generate a hardware interrupt request.<br>0 Hardware interrupt requests from IRQF disabled (use polling).<br>1 Hardware interrupt requested whenever IRQF = 1. |
| 0<br>IRQMOD | **IRQ Detection Mode** — 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 Section 5.5.2.2, “Edge and Level Sensitivity,” for more details.<br>0 IRQ event on falling edges or rising edges only.<br>1 IRQ event on falling edges and low levels or on rising edges and high levels. |

## 5.8.2 System Reset Status Register (SRS)

This register includes six read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by writing 1 to BDFR in the SBDFR register, none of the status bits in SRS will be set. Writing any value to this register address 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(1) | 0 | 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 | (2) | (2) | (2) | (2) | 0 | 0 | 0 |

u = Unaffected by reset

**Figure 5-3. System Reset Status (SRS)**

1. The ILAD bit is only present in 16K and 8K versions of the devices.
2. 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.

**Table 5-3. SRS Field Descriptions**

| Field | Description |
|---|---|
| 7<br>POR | **Power-On Reset** — Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVR) status bit is also set to indicate that the reset occurred while the internal supply was below the LVR threshold.<br>0 Reset not caused by POR.<br>1 POR caused reset. |
| 6<br>PIN | **External Reset Pin** — Reset was caused by an active-low level on the external reset pin.<br>0 Reset not caused by external reset pin.<br>1 Reset came from external reset pin. |
| 5<br>COP | **Computer Operating Properly (COP) Watchdog** — Reset was caused by the COP watchdog timer timing out. This reset source may be blocked by COPE = 0.<br>0 Reset not caused by COP timeout.<br>1 Reset caused by COP timeout. |
| 4<br>ILOP | **Illegal Opcode** — Reset was caused by an attempt to execute an unimplemented or illegal opcode. The STOP instruction is considered illegal if stop is disabled by STOPE = 0 in the SOPT register. The BGND instruction is considered illegal if active background mode is disabled by ENBDM = 0 in the BDCSC register.<br>0 Reset not caused by an illegal opcode.<br>1 Reset caused by an illegal opcode. |
| 3<br>ILAD | **Illegal Address Access** — Reset was caused by an attempt to access a designated illegal address.<br>0 Reset not caused by an illegal address access<br>1 Reset caused by an illegal address access<br>Illegal address areas only exist in the 16K and 8K versions and are defined as:<br>• $0440–$17FF — Gap from end of RAM to start of high-page registers<br>• $1834–$BFFF — Gap from end of high-page registers to start of FLASH memory<br>Unused and reserved locations in register areas are not considered designated illegal addresses and do not trigger illegal address resets. |
| 1<br>LVD | **Low Voltage Detect** — If the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset occurs. This bit is also set by POR.<br>0 Reset not caused by LVD trip or POR<br>1 Reset caused by LVD trip or POR |

### 5.8.3 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 $00.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | BDFR(1) |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

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

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

**Table 5-4. SBDFR Field Descriptions**

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

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

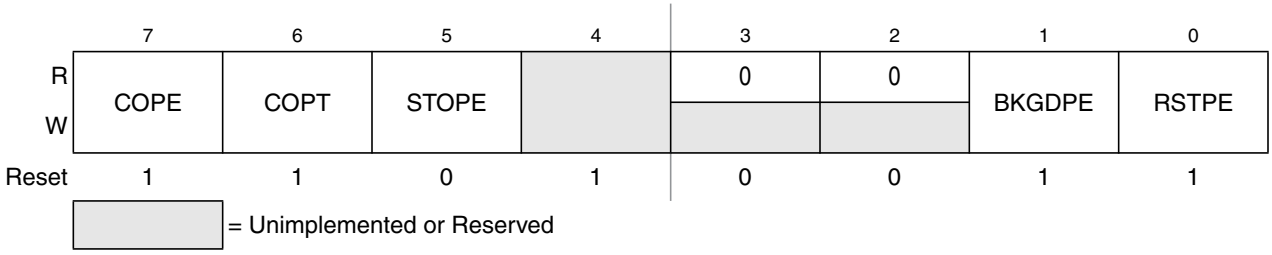



**Figure 5-5. System Options Register (SOPT)**

**Table 5-5. SOPT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>COPE | **COP Watchdog Enable** — This write-once bit defaults to 1 after reset.<br>0 COP watchdog timer disabled.<br>1 COP watchdog timer enabled (force reset on timeout). |
| 6<br>COPT | **COP Watchdog Timeout** — This write-once bit defaults to 1 after reset.<br>0 Short timeout period selected ($2^{18}$ cycles of BUSCLK).<br>1 Long timeout period selected ($2^{20}$ cycles of BUSCLK). |
| 5<br>STOPE | **Stop Mode Enable** — This write-once bit defaults to 0 after reset, which disables stop mode. If stop mode is disabled and a user program attempts to execute a STOP instruction, an illegal opcode reset is forced.<br>0 Stop mode disabled.<br>1 Stop mode enabled. |
| 1<br>BKGDPE | **Background Debug Mode Pin Enable** — The BKGDPE bit enables the PTD0/BKGD/MS pin to function as BKGD/MS. When the bit is clear, the pin will function as PTD0, which is an output only general purpose I/O. This pin always defaults to BKGD/MS function after any reset.<br>0 BKGD pin disabled.<br>1 BKGD pin enabled. |
| 0<br>RSTPE | **$\overline{\text{RESET}}$ Pin Enable** — The RSTPE bit enables the PTD1/$\overline{\text{RESET}}$ pin to function as $\overline{\text{RESET}}$. When the bit is clear, the pin will function as PTD1, which is an output only general purpose I/O. This pin always defaults to $\overline{\text{RESET}}$ function after any reset.<br>0 $\overline{\text{RESET}}$ pin disabled.<br>1 $\overline{\text{RESET}}$ pin enabled. |

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

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | REV3 | REV2 | REV1 | REV0 | ID11 | ID10 | ID9 | ID8 |
| W | | | | | | | | |
| Reset | 0(1) | 0(1) | 0(1) | 0(1) | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 5-6. System Device Identification Register — High (SDIDH)**

1. The revision number that is hard coded into these bits reflects the current silicon revision level.

**Table 5-6. SDIDH Field Descriptions**

| Field | Description |
|---|---|
| 7:4<br>REV[3:0] | **Revision Number** — The high-order 4 bits of address $1806 are hard coded to reflect the current mask set revision number (0–F). |
| 3:0<br>ID[11:8] | **Part Identification Number** — Each derivative in the HCS08 Family has a unique identification number. The MC9S08RC/RD/RE/RG32/60 is hard coded to the value $004 and the MC9S08RC/RD/RE8/16 is hard coded to the value $003. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |
| W | | | | | | | | |
| Reset, 8/16K: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Reset, 32/60K: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

= Unimplemented or Reserved

**Figure 5-7. System Device Identification Register — Low (SDIDL)**

**Table 5-7. SDIDL Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>ID[7:0] | **Part Identification Number** — Each derivative in the HCS08 Family has a unique identification number. The MC9S08RC/RD/RE/RG32/60 is hard coded to the value $004 and the MC9S08RC/RD/RE8/16 is hard coded to the value $003. |

## 5.8.6 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.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RTIF | 0 | RTICLKS | RTIE | 0 | RTIS2 | RTIS1 | RTIS0 |
| W | | RTIACK | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 5-8. System RTI Status and Control Register (SRTISC)**

**Table 5-8. SRTISC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>RTIF | **Real-Time Interrupt Flag** — This read-only status bit indicates the periodic wakeup timer has timed out.<br>0 Periodic wakeup timer not timed out.<br>1 Periodic wakeup timer timed out. |
| 6<br>RTIACK | **Real-Time Interrupt Acknowledge** — 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 0. |
| 5<br>RTICLKS | **Real-Time Interrupt Clock Select** — This read/write bit selects the clock source for the real-time interrupt.<br>0 Real-time interrupt request clock source is internal 1-kHz oscillator.<br>1 Real-time interrupt request clock source is external clock. |
| 4<br>RTIE | **Real-Time Interrupt Enable** — This read-write bit enables real-time interrupts.<br>0 Real-time interrupts disabled.<br>1 Real-time interrupts enabled. |
| 2:0<br>RTIS[2:0] | **Real-Time Interrupt Period Selects** — These read/write bits select the wakeup period for the RTI. One clock source for the real-time interrupt is its own internal clock source, which oscillates with a period of approximately $t_{RTI}$ and is independent of other MCU clock sources. Using an external clock source the delays will be crystal frequency divided by value in RTIS2:RTIS1:RTIS0. See Table 5-9. |

**Table 5-9. Real-Time Interrupt Period**

| RTIS2:RTIS1:RTIS0 | Internal Clock Source (1) ($t_{RTI}$ = 1 ms, Nominal) | External Clock Source (2) Period = $t_{ext}$ |
|---|---|---|
| 0:0:0 | Disable periodic wakeup timer | Disable periodic wakeup timer |
| 0:0:1 | 8 ms | $t_{ext}$ x 256 |
| 0:1:0 | 32 ms | $t_{ex}$ x 1024 |
| 0:1:1 | 64 ms | $t_{ex}$ x 2048 |
| 1:0:0 | 128 ms | $t_{ex}$ x 4096 |
| 1:0:1 | 256 ms | $t_{ext}$ x 8192 |
| 1:1:0 | 512 ms | $t_{ext}$ x 16384 |
| 1:1:1 | 1.024 s | $t_{ex}$ x 32768 |

1. See Table A-9 $t_{RTI}$ in Appendix A, "Electrical Characteristics," for the tolerance on these values.
2. $t_{ext}$ is based on the external clock source, resonator, or crystal selected by the ICG configuration. See Table A-9 for details.

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LVDF | 0 | LVDIE | SAFE | LVDRE(1) | 0 | 0 | 0 |
| W | | LVDACK | | | | | | |
| POR: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Any other reset: | 0 | 0 | 0 | 0 | u | 0 | 0 | 0 |

= Unimplemented or Reserved  u = Unaffected by reset

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

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

**Table 5-10. SPMSC1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LVDF | **Low-Voltage Detect Flag** — Provided LVDE = 1, this read-only status bit indicates a low-voltage detect error. |
| 6<br>LVDACK | **Low-Voltage Detect Acknowledge** — This write-only bit is used to acknowledge low voltage detection errors (write 1 to clear LVDF). Reads always return logic 0. |
| 5<br>LVDIE | **Low-Voltage Detect Interrupt Enable** — This read/write bit enables hardware interrupt requests for LVDF.<br>0 Hardware interrupt disabled (use polling).<br>1 Request a hardware interrupt when LVDF = 1. |
| 4<br>SAFE | **SAFE System from interrupts** — This read/write bit enables hardware to block interrupts and resets from waking the MCU from stop mode while the supply voltage $V_{DD}$ is below the $V_{REARM}$ voltage. For a more detailed description see Section 5.6.3, “LVD Interrupt and Safe State Operation”<br>0 Enable pending interrupts and resets<br>1 Interrupts and resets are blocked while supply voltage is below re-arm voltage |
| 3<br>LVDRE | **Low-Voltage Detect Reset Enable** — This bit enables the LVD reset function. This bit can be written only once after a reset and additional writes have no meaning or effect. It is set following a POR and is unaffected by any other resets, including an LVD reset.<br>0 LVDF does not generate hardware resets.<br>1 Force an MCU reset when LVDF = 1. |

## 5.8.8 System Power Management Status and Control 2 Register (SPMSC2)

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LVWF | 0 | 0 | 0 | PPDF | 0 | PDC | PPDC |
| W | | LVWACK | | | | PPDACK | | |
| Reset | 0(1) | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

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

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

**Table 5-11. SPMSC2 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LVWF | **Low-Voltage Warning Flag** — The LVWF bit indicates the low voltage warning status.<br>0 Low voltage warning **not** present.<br>1 Low voltage warning is present or was present. |
| 6<br>LVWACK | **Low-Voltage Warning Acknowledge** — The LVWF bit indicates the low voltage warning status. Writing a 1 to LVWACK clears LVWF to a 0 if a low voltage warning is not present. |
| 3<br>PPDF | **Partial Power Down Flag** — The PPDF bit indicates that the MCU has exited the stop2 mode.<br>0 Not stop2 mode recovery.<br>1 Stop2 mode recovery. |
| 2<br>PPDACK | **Partial Power Down Acknowledge** — Writing a logic 1 to PPDACK clears the PPDF bit. |
| 1<br>PDC | **Power Down Control** — The write-once PDC bit controls entry into the power down (stop2 and stop1) modes.<br>0 Power down modes are disabled.<br>1 Power down modes are enabled. |
| 0<br>PPDC | **Partial Power Down Control** — The write-once PPDC bit controls which power down mode, stop1 or stop2, is selected.<br>0 Stop1, full power down, mode enabled if PDC set.<br>1 Stop2, partial power down, mode enabled if PDC set. |

# Chapter 6
# Parallel Input/Output

## 6.1 Introduction

This section explains software controls related to parallel input/output (I/O). The MC9S08RC/RD/RE/RG has five I/O ports that include a total of 39 general-purpose I/O pins (two of these pins are output only and one pin is input only). Not all of the ports are available in all packages. 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, external interrupts, or keyboard interrupts. When these other modules are not controlling the port pins, they revert to general-purpose I/O control. For each I/O pin, a port data bit provides access to input (read) and output (write) data. A data direction bit controls the direction of the pin and a pullup enable bit enables an internal pullup device (if the pin is configured as an input).

**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.

## 6.2 Features

Parallel I/O features for the MC9S08RC/RD/RE/RG MCUs, depending on specific device and package choice, include:

- A total of 39 general-purpose I/O pins in five ports (two pins are output only, one is input only)
- High-current drivers on port B pins
- Hysteresis input buffers on all inputs
- Software-controlled pullups on each input pin
- Eight port A pins shared with KBI1
- Eight port B pins shared with SCI and TPMCH1
- Eight port C pins shared with KBI2 and SPI
- Seven port D pins shared with TPMCH0, ACMP, IRQ, $\overline{\text{RESET}}$, and BKGD/MS
- Eight port E pins

## 6.3 Pin Descriptions

The MC9S08RC/RD/RE/RG has a total of 39 parallel I/O pins distributed between four 8-bit ports and one 7-bit port. 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.

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: | PTA7/ KBI1P7 | PTA6/ KBI1P6 | PTA5/ KBI1P5 | PTA4/ KBI1P4 | PTA3/ KBI1P3 | PTA2/ KBI1P2 | PTA1/ KBI1P1 | PTA0/ KBI1P0 |

**Figure 6-1. Port A Pin Names**

Port A is an 8-bit general-purpose I/O port shared with the KBI1 keyboard interrupt inputs. Bit 0 of port A is an input-only pin.

Port A pins are available as general-purpose I/O pins controlled by the port A data (PTAD), data direction (PTADD), and pullup enable (PTAPE) registers. Refer to Section 6.4, "Parallel I/O Controls," for more information about general-purpose I/O control.

Any of the port A pins can be configured as a KBI1 keyboard interrupt pin. Refer to the Keyboard Interrupt (KBI) Module chapter for more information about using port A pins as keyboard interrupt pins.

### 6.3.2 Port B

| Port B | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| | MCU Pin: | PTB7/ TPM1CH1 | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1/ RxD1 | PTB0/ TxD1 |

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

Port B is an 8-bit general-purpose I/O port with two pins shared with the SCI and one pin shared with the TPM. The port B output drivers are capable of high current drive.

Port B pins are available as general-purpose I/O pins controlled by the port B data (PTBD), data direction (PTBDD), and pullup enable (PTBPE) registers. Refer to Section 6.4, "Parallel I/O Controls," for more information about general-purpose I/O control.

When the SCI module is enabled, PTB0 and PTB1 function as the transmit (TxD1) and receive (RxD1) pins of the SCI. Refer to the Serial Communications Interface (SCI) Module chapter for more information about using PTB0 and PTB1 as SCI pins.

The TPM can be configured to use PTB7 as either an input capture, output compare, or PWM pin. Refer to the Timer/PWM Module (TPM) Module chapter for more information about using PTB7 as a timer pin.

### 6.3.3 Port C

| Port C | | Bit 7 | 6 | 5 | 3 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| | MCU Pin: | PTC7/ $\overline{SS1}$ | PTC6/ SPSCK1 | PTC5/ MISO1 | PTC4/ MOSI1 | PTC3/ KBI2P3 | PTC2/ KBI2P2 | PTC1/ KBI2P1 | PTC0/ KBI2P0 |

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

Port C is an 8-bit general-purpose I/O port with four pins shared with the KBI2 keyboard interrupt inputs and four pins shared with the SPI.

Port C pins are available as general-purpose I/O pins controlled by the port C data (PTCD), data direction (PTCDD), and pullup enable (PTCPE) registers. Refer to Section 6.4, “Parallel I/O Controls,” for more information about general-purpose I/O control.

When the SPI module is enabled, PTC7 serves as the SPI module’s slave select pin ($\overline{SS1}$), PTC6 serves as the SPI clock pin (SPSCK1), PTC5 serves as the master-in slave-out pin (MISO1), and PTC4 serves as the master-out slave-in pin (MOSI1). Refer to the Serial Peripheral Interface (SPI) Module chapter for more information about using PTC7–PTC4 as SPI pins.

Any of the port C pins PTC3-PTC0 can be configured as a KBI2 keyboard interrupt pin. Refer to the Keyboard Interrupt (KBI) Module chapter for more information about using port C pins as keyboard interrupt pins.

### 6.3.4 Port D

| Port D | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| | MCU Pin: | | PTD6/ TPM1CH0 | PTD5 | PTD4 | PTD3 | PTD2 | PTD1/ $\overline{RESET}$ | PTD0/ BKGD/MS |

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

Port D is an 7-bit general-purpose I/O port with one pin shared with the BKGD/MS function, one pin shared with the $\overline{RESET}$ function, one pin shared with the IRQ function, and one pin shared with the TPM.

Port D pins are available as general-purpose I/O pins controlled by the port D data (PTDD), data direction (PTDDD), and pullup enable (PTDPE) registers. Refer to Section 6.4, “Parallel I/O Controls,” for more information about general-purpose I/O control.

The PTD0/BKGD/MS pin is configured for the BKGD/MS function during reset and following reset. The internal pullup for this pin is enabled when the BKGD/MS function is enabled, regardless of the PTDPE0 bit. During reset, the BKGD/MS pin functions as a mode select pin. After the MCU is out of reset, the BKGD/MS pin becomes the background communications input/output pin. PTD0 can be configured to be a general-purpose output pin through software control. Refer to Chapter 3, “Modes of Operation,” Chapter 5, “Resets, Interrupts, and System Configuration,” and the Development Support chapter for more information about using this pin.

The PTD1/$\overline{RESET}$ pin is configured for the RESET function during reset and following reset.

The TPM can be configured to use PTD6 as either an input capture, output compare, PWM, or external clock input pin. Refer to the Chapter 6, “Parallel Input/Output,” for more information about using PTD6 as a timer pin.

### 6.3.5 Port E

| Port E | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| | MCU Pin: | PTE7 | PTE6 | PTE5 | PTE4 | PTE3 | PTE2 | PTE1 | PTE0 |

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

Port E is an 8-bit general-purpose I/O port.

Port E pins are available as general-purpose I/O pins controlled by the port E data (PTED), data direction (PTEDD), and pullup enable (PTEPE) registers. Refer to Section 6.4, “Parallel I/O Controls,” for more information about general-purpose I/O control.

## 6.4 Parallel I/O Controls

Provided no on-chip peripheral is controlling a port pin, the pins operate as general-purpose I/O pins that are accessed and controlled by a data register (PTxD), a data direction register (PTxDD), and a pullup enable register (PTxPE) where x is A, B, C, D, or E.

Reads of the data register return the pin value (if PTxDDn = 0) or the contents of the port data register (if PTxDDn = 1). Writes to the port data register are latched into the port register whether the pin is controlled by an on-chip peripheral or the pin is configured as an input. If the corresponding pin is not controlled by a peripheral and is configured as an output, this level will be driven out the port pin.

### 6.4.1 Data Direction Control

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 control 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 control 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.

For the MC9S08RC/RD/RE/RG MCU, reads of PTD0/BKGD/MS and PTD1/$\overline{\text{RESET}}$ will return the value on the output pin.

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

### 6.4.2 Internal Pullup Control

An internal pullup device can be enabled for each port pin that is configured as an input (PTxDDn = 0). The pullup device is available for a peripheral module to use, provided the peripheral is enabled and is an input function as long as the PTxDDn = 0.

**NOTE**

The voltage measured on the pulled up PTA0 pin will be less than $V_{DD}$. The internal gates connected to this pin are pulled all the way to $V_{DD}$. All other pins with enabled pullup resistors will have an unloaded measurement of $V_{DD}$.

## 6.5 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:

- When the MCU enters stop1 mode, all internal registers, including general-purpose I/O control and data registers, are powered down. All of the general-purpose I/O pins assume their reset state: output buffers and pullups turned off. Upon exit from stop1, all I/O must be initialized as if the MCU had been reset.
- When the MCU enters stop2 mode, the internal registers are powered down as in stop1 but the I/O pin states are latched and held. For example, a port pin that is an output driving low continues to function as an output driving low even though its associated data direction and output data registers are powered down internally. Upon exit from stop2, the pins continue to hold their states until a 1 is written to the PPDACK bit. To avoid discontinuity in the pin state following exit from stop2, the user must restore the port control and data registers to the values they held befor4e entering stop2. These values can be stored in RAM before entering stop2 because the RAM is maintained during stop2.
- In stop3 mode, all I/O is maintained because internal logic circuity stays powered up. Upon recovery, normal I/O function is available to the user.

## 6.6 Parallel I/O Registers and Control Bits

This section provides information about all registers and control bits associated with the parallel I/O ports.

Refer to tables in the Memory chapter for the absolute address assignments for all parallel I/O 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.

## 6.6.1 Port A Registers (PTAD, PTAPE, and PTADD)

Port A pins used as general-purpose I/O pins are controlled by the port A data (PTAD), data direction (PTADD), and pullup enable (PTAPE) registers.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTAD7 | PTAD6 | PTAD5 | PTAD4 | PTAD3 | PTAD2 | PTAD1 | PTAD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

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

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTAPE7 | PTAPE6 | PTAPE5 | PTAPE4 | PTAPE3 | PTAPE2 | PTAPE1 | PTAPE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-7. Pullup Enable for Port A (PTAPE)**

**Table 6-2. PTAPE Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTAPE[7:0] | **Pullup Enable for Port A Bits** — For port A pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled provided the corresponding PTADDn is a logic 0. For port A pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled. When any of bits 7 through 4 of port A are enabled as KBI inputs and are configured to detect rising edges/high levels, the pullup enable bits enable pulldown rather than pullup devices.<br>0 Internal pullup device disabled.<br>1 Internal pullup device enabled. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTADD7 | PTADD6 | PTADD5 | PTADD4 | PTADD3 | PTADD2 | PTADD1 | PTADD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-8. Data Direction for Port A (PTADD)**

**Table 6-3. PTADD Field Descriptions**

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

## 6.6.2 Port B Registers (PTBD, PTBPE, and PTBDD)

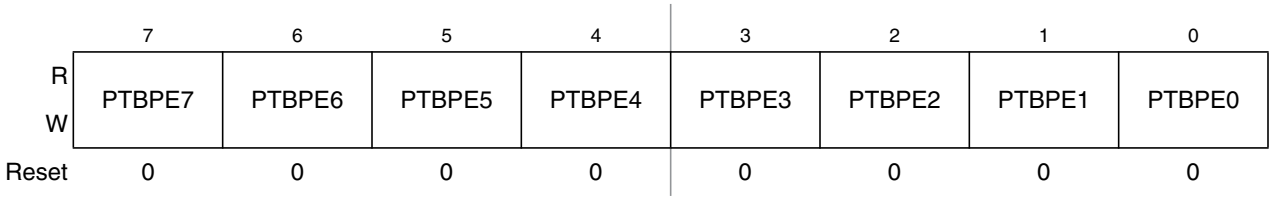
Port B pins used as general-purpose I/O pins are controlled by the port B data (PTBD), data direction (PTBDD), and pullup enable (PTBPE) registers.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBD7 | PTBD6 | PTBD5 | PTBD4 | PTBD3 | PTBD2 | PTBD1 | PTBD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-9. Port B Data Register (PTBD)**

**Table 6-4. PTBD Field Descriptions**

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

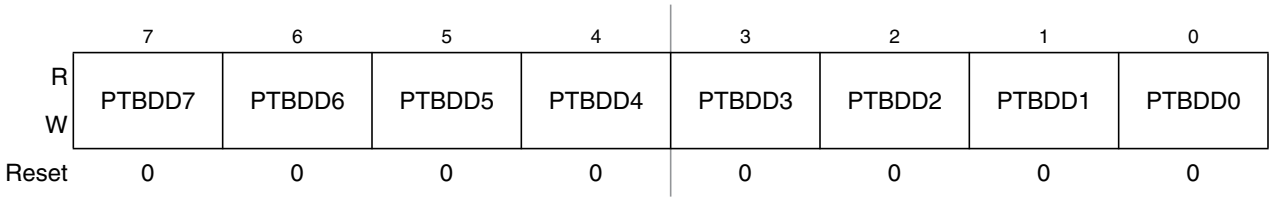


| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBPE7 | PTBPE6 | PTBPE5 | PTBPE4 | PTBPE3 | PTBPE2 | PTBPE1 | PTBPE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-10. Pullup Enable for Port B (PTBPE)**

**Table 6-5. PTBPE Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTBPE[7:0] | **Pullup Enable for Port B Bits** — For port B pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port B pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.<br>0 Internal pullup device disabled.<br>1 Internal pullup device enabled. |



| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBDD7 | PTBDD6 | PTBDD5 | PTBDD4 | PTBDD3 | PTBDD2 | PTBDD1 | PTBDD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-11. Data Direction for Port B (PTBDD)**

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

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

### 6.6.3 Port C Registers (PTCD, PTCPE, and PTCDD)

Port C pins used as general-purpose I/O pins are controlled by the port C data (PTCD), data direction (PTCDD), and pullup enable (PTCPE) registers.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTCD7 | PTCD6 | PTCD5 | PTCD4 | PTCD3 | PTCD2 | PTCD1 | PTCD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Table 6-7. PTCD Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCD[7:0] | **Port C Data Register Bits** — For port C pins that are inputs, reads return the logic level on the pin. For port C pins that are configured as outputs, reads return the last value written to this register.<br>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.<br>Reset forces PTCD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTCPE7 | PTCPE6 | PTCPE5 | PTCPE4 | PTCPE3 | PTCPE2 | PTCPE1 | PTCPE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-13. Pullup Enable for Port C (PTCPE)**

**Table 6-8. PTCPE Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCPE[7:0] | **Pullup Enable for Port C Bits** — For port C pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port C pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.<br>0 Internal pullup device disabled.<br>1 Internal pullup device enabled. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTCDD7 | PTCDD6 | PTCDD5 | PTCDD4 | PTCDD3 | PTCDD2 | PTCDD1 | PTCDD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-14. Data Direction for Port C (PTCDD)**

**Table 6-9. PTCDD Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTCDD[7:0] | **Data Direction for Port C Bits** — These read/write bits control the direction of port C pins and what is read for PTCD reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for port C bit n and PTCD reads return the contents of PTCDn. |

## 6.6.4 Port D Registers (PTDD, PTDPE, and PTDDD)

Port D pins used as general-purpose I/O pins are controlled by the port D data (PTDD), data direction (PTDDD), and pullup enable (PTDPE) registers.

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

= Unimplemented or Reserved

**Figure 6-15. Port D Data Register (PTDD)**

**Table 6-10. PTDD Field Descriptions**

| Field | Description |
|---|---|
| 6:0<br>PTDD[6:0] | **Port D Data Register Bits** — For port D pins that are inputs, reads return the logic level on the pin. For port D pins that are configured as outputs, reads return the last value written to this register.<br>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.<br>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. |

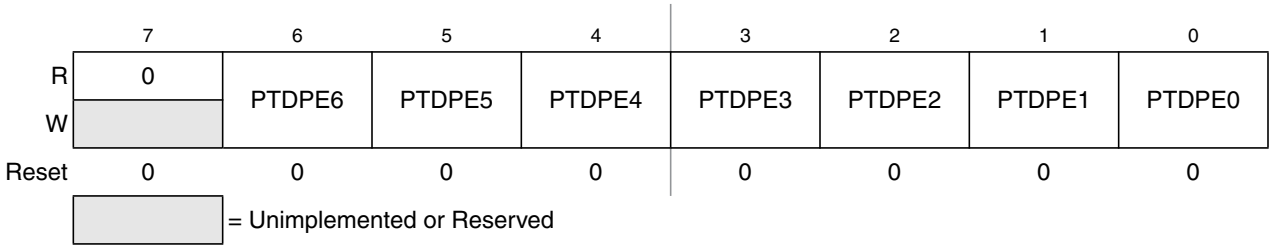



**Figure 6-16. Pullup Enable for Port D (PTDPE)**

**Table 6-11. PTDPE Field Descriptions**

| Field | Description |
|---|---|
| 6:0<br>PTDPE[6:0] | **Pullup Enable for Port D Bits** — For port D pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port D pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.<br>0 Internal pullup device disabled.<br>1 Internal pullup device enabled. |

**Figure 6-17. Data Direction for Port D (PTDDD)**

**Table 6-12. PTDDD Field Descriptions**

| Field | Description |
|---|---|
| 6:0<br>PTDDD[6:0] | **Data Direction for Port D Bits** — These read/write bits control the direction of port D pins and what is read for PTDD reads.<br>0 Input (output driver disabled) and reads return the pin value.<br>1 Output driver enabled for port D bit n and PTDD reads return the contents of PTDDn. |

## 6.6.5 Port E Registers (PTED, PTEPE, and PTEDD)

Port E pins used as general-purpose I/O pins are controlled by the port E data (PTED), data direction (PTEDD), and pullup enable (PTEPE) registers.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTED7 | PTED6 | PTED5 | PTED4 | PTED3 | PTED2 | PTED1 | PTED0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-18. Port E Data Register (PTED)**

**Table 6-13. PTED Field Descriptions**

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTEPE7 | PTEPE6 | PTEPE5 | PTEPE4 | PTEPE3 | PTEPE2 | PTEPE1 | PTEPE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-19. Pullup Enable for Port E (PTED)**

**Table 6-14. PTED Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTED[7:0] | **Pullup Enable for Port E Bits** — For port E pins that are inputs, these read/write control bits determine whether internal pullup devices are enabled. For port E pins that are configured as outputs, these bits are ignored and the internal pullup devices are disabled.<br>0 Internal pullup device disabled.<br>1 Internal pullup device enabled. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTEDD7 | PTEDD6 | PTEDD5 | PTEDD4 | PTEDD3 | PTEDD2 | PTEDD1 | PTEDD0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-20. Data Direction for Port E (PTEDD)**

**Table 6-15. PTEDD Field Descriptions**

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

# Chapter 7
# Central Processor Unit (S08CPUV2)

## 7.1 Introduction

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

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

### 7.1.1 Features

Features of the HCS08 CPU include:

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

## 7.2 Programmer's Model and CPU Registers

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

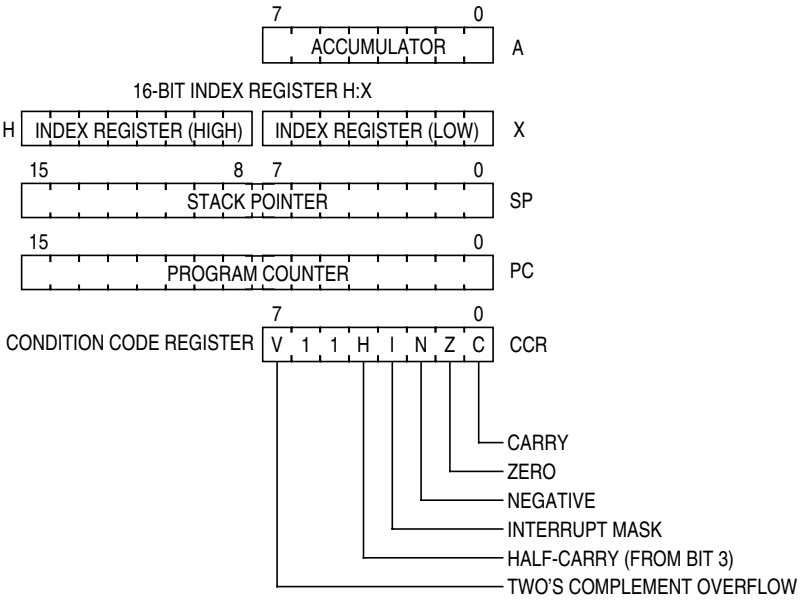



**Figure 7-1. CPU Registers**

### 7.2.1 Accumulator (A)

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

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

### 7.2.2 Index Register (H:X)

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

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

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

### 7.2.3 Stack Pointer (SP)

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

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

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

### 7.2.4 Program Counter (PC)

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

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

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

### 7.2.5 Condition Code Register (CCR)

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

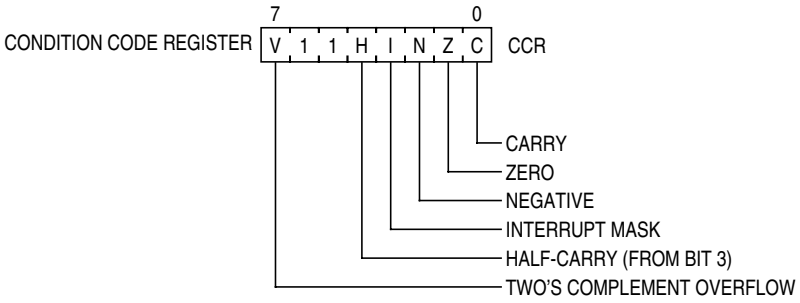



**Figure 7-2. Condition Code Register**

**Table 7-1. CCR Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>V | **Two's Complement Overflow Flag** — The CPU sets the overflow flag when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow flag.<br>0 No overflow<br>1 Overflow |
| 4<br>H | **Half-Carry Flag** — The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an add-without-carry (ADD) or add-with-carry (ADC) operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C condition code bits to automatically add a correction value to the result from a previous ADD or ADC on BCD operands to correct the result to a valid BCD value.<br>0 No carry between bits 3 and 4<br>1 Carry between bits 3 and 4 |
| 3<br>I | **Interrupt Mask Bit** — When the interrupt mask is set, all maskable CPU interrupts are disabled. CPU interrupts are enabled when the interrupt mask is cleared. When a CPU interrupt occurs, the interrupt mask is set automatically after the CPU registers are saved on the stack, but before the first instruction of the interrupt service routine is executed.<br>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.<br>0 Interrupts enabled<br>1 Interrupts disabled |
| 2<br>N | **Negative Flag** — The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result. Simply loading or storing an 8-bit or 16-bit value causes N to be set if the most significant bit of the loaded or stored value was 1.<br>0 Non-negative result<br>1 Negative result |
| 1<br>Z | **Zero Flag** — The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of 0x00 or 0x0000. Simply loading or storing an 8-bit or 16-bit value causes Z to be set if the loaded or stored value was all 0s.<br>0 Non-zero result<br>1 Zero result |
| 0<br>C | **Carry/Borrow Flag** — The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions — such as bit test and branch, shift, and rotate — also clear or set the carry/borrow flag.<br>0 No carry out of bit 7<br>1 Carry out of bit 7 |

## 7.3 Addressing Modes

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

Some instructions use more than one addressing mode. For instance, move instructions use one addressing mode to specify the source operand and a second addressing mode to specify the destination address. Instructions such as BRCLR, BRSET, CBEQ, and DBNZ use one addressing mode to specify the location

of an operand for a test and then use relative addressing mode to specify the branch destination address when the tested condition is true. For BRCLR, BRSET, CBEQ, and DBNZ, the addressing mode listed in the instruction set tables is the addressing mode needed to access the operand to be tested, and relative addressing mode is implied for the branch destination.

### 7.3.1 Inherent Addressing Mode (INH)

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

### 7.3.2 Relative Addressing Mode (REL)

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

### 7.3.3 Immediate Addressing Mode (IMM)

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

### 7.3.4 Direct Addressing Mode (DIR)

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

### 7.3.5 Extended Addressing Mode (EXT)

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

### 7.3.6 Indexed Addressing Mode

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

#### 7.3.6.1 Indexed, No Offset (IX)

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

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

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

#### 7.3.6.3 Indexed, 8-Bit Offset (IX1)

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

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

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

#### 7.3.6.5 Indexed, 16-Bit Offset (IX2)

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

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

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

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

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

## 7.4 Special Operations

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

### 7.4.1 Reset Sequence

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

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

### 7.4.2 Interrupt Sequence

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

The CPU sequence for an interrupt is:

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

After the CCR contents are pushed onto the stack, the I bit in the CCR is set to prevent other interrupts while in the interrupt service routine. Although it is possible to clear the I bit with an instruction in the interrupt service routine, this would allow nesting of interrupts (which is not recommended because it leads to programs that are difficult to debug and maintain).

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

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

### 7.4.3 Wait Mode Operation

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

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

### 7.4.4 Stop Mode Operation

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

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

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

### 7.4.5 BGND Instruction

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

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

## 7.5 HCS08 Instruction Set Summary

### Instruction Set Summary Nomenclature

The nomenclature listed here is used in the instruction descriptions in Table 7-2.

### Operators

| | | |
|---|---|---|
| ( ) | = | Contents of register or memory location shown inside parentheses |
| ← | = | Is loaded with (read: “gets”) |
| & | = | Boolean AND |
| \| | = | Boolean OR |
| ⊕ | = | Boolean exclusive-OR |
| × | = | Multiply |
| ÷ | = | Divide |
| : | = | Concatenate |
| + | = | Add |
| – | = | Negate (two’s complement) |

### CPU registers

| | | |
|---|---|---|
| A | = | Accumulator |
| CCR | = | Condition code register |
| H | = | Index register, higher order (most significant) 8 bits |
| X | = | Index register, lower order (least significant) 8 bits |
| PC | = | Program counter |
| PCH | = | Program counter, higher order (most significant) 8 bits |
| PCL | = | Program counter, lower order (least significant) 8 bits |
| SP | = | Stack pointer |

### Memory and addressing

| | | |
|---|---|---|
| M | = | A memory location or absolute data, depending on addressing mode |
| M:M + 0x0001 | = | A 16-bit value in two consecutive memory locations. The higher-order (most significant) 8 bits are located at the address of M, and the lower-order (least significant) 8 bits are located at the next higher sequential address. |

### Condition code register (CCR) bits

| | | |
|---|---|---|
| V | = | Two’s complement overflow indicator, bit 7 |
| H | = | Half carry, bit 4 |
| I | = | Interrupt mask, bit 3 |
| N | = | Negative indicator, bit 2 |
| Z | = | Zero indicator, bit 1 |
| C | = | Carry/borrow, bit 0 (carry out of bit 7) |

### CCR activity notation

| | | |
|---|---|---|
| – | = | Bit not affected |

| | | |
|---|---|---|
| 0 | = | Bit forced to 0 |
| 1 | = | Bit forced to 1 |
| | = | Bit set or cleared according to results of operation |
| U | = | Undefined after the operation |

### Machine coding notation

| | | |
|---|---|---|
| dd | = | Low-order 8 bits of a direct address 0x0000–0x00FF (high byte assumed to be 0x00) |
| ee | = | Upper 8 bits of 16-bit offset |
| ff | = | Lower 8 bits of 16-bit offset or 8-bit offset |
| ii | = | One byte of immediate data |
| jj | = | High-order byte of a 16-bit immediate data value |
| kk | = | Low-order byte of a 16-bit immediate data value |
| hh | = | High-order byte of 16-bit extended address |
| ll | = | Low-order byte of 16-bit extended address |
| rr | = | Relative offset |

### Source form

Everything in the source forms columns, *except expressions in italic characters,* is literal information that must appear in the assembly source file exactly as shown. The initial 3- to 5-letter mnemonic is always a literal expression. All commas, pound signs (#), parentheses, and plus signs (+) are literal characters.

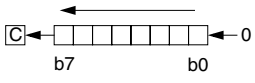
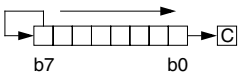
| | | |
|---|---|---|
| *n* | — | Any label or expression that evaluates to a single integer in the range 0–7 |
| *opr8i* | — | Any label or expression that evaluates to an 8-bit immediate value |
| *opr16i* | — | Any label or expression that evaluates to a 16-bit immediate value |
| *opr8a* | — | Any label or expression that evaluates to an 8-bit value. The instruction treats this 8-bit value as the low order 8 bits of an address in the direct page of the 64-Kbyte address space (0x00xx). |
| *opr16a* | — | Any label or expression that evaluates to a 16-bit value. The instruction treats this value as an address in the 64-Kbyte address space. |
| *oprx8* | — | Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing |
| *oprx16* | — | Any label or expression that evaluates to a 16-bit value. Because the HCS08 has a 16-bit address bus, this can be either a signed or an unsigned value. |
| *rel* | — | Any label or expression that refers to an address that is within –128 to +127 locations from the next address after the last byte of object code for the current instruction. The assembler will calculate the 8-bit signed offset and include it in the object code for this instruction. |

### Address modes

| | | |
|---|---|---|
| INH | = | Inherent (no operands) |
| IMM | = | 8-bit or 16-bit immediate |
| DIR | = | 8-bit direct |
| EXT | = | 16-bit extended |

IX = 16-bit indexed no offset
IX+ = 16-bit indexed no offset, post increment (CBEQ and MOV only)
IX1 = 16-bit indexed with 8-bit offset from H:X
IX1+ = 16-bit indexed with 8-bit offset, post increment (CBEQ only)
IX2 = 16-bit indexed with 16-bit offset from H:X
REL = 8-bit relative offset
SP1 = Stack pointer with 8-bit offset
SP2 = Stack pointer with 16-bit offset

**Table 7-2. HCS08 Instruction Set Summary (Sheet 1 of 7)**

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| ADC #*opr8i*<br>ADC *opr8a*<br>ADC *opr16a*<br>ADC *oprx16*,X<br>ADC *oprx8*,X<br>ADC ,X<br>ADC *oprx16*,SP<br>ADC *oprx8*,SP | Add with Carry | A ← (A) + (M) + (C) | | | – | | | | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A9<br>B9<br>C9<br>D9<br>E9<br>F9<br>9ED9<br>9EE9 | ii<br>dd<br>hh ll<br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| ADD #*opr8i*<br>ADD *opr8a*<br>ADD *opr16a*<br>ADD *oprx16*,X<br>ADD *oprx8*,X<br>ADD ,X<br>ADD *oprx16*,SP<br>ADD *oprx8*,SP | Add without Carry | A ← (A) + (M) | | | – | | | | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AB<br>BB<br>CB<br>DB<br>EB<br>FB<br>9EDB<br>9EEB | ii<br>dd<br>hh ll<br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| AIS #*opr8i* | Add Immediate Value (Signed) to Stack Pointer | SP ← (SP) + (M)<br>M is sign extended to a 16-bit value | – | – | – | – | – | – | IMM | A7 | ii | 2 |
| AIX #*opr8i* | Add Immediate Value (Signed) to Index Register (H:X) | H:X ← (H:X) + (M)<br>M is sign extended to a 16-bit value | – | – | – | – | – | – | IMM | AF | ii | 2 |
| AND #*opr8i*<br>AND *opr8a*<br>AND *opr16a*<br>AND *oprx16*,X<br>AND *oprx8*,X<br>AND ,X<br>AND *oprx16*,SP<br>AND *oprx8*,SP | Logical AND | A ← (A) & (M) | 0 | – | – | | | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A4<br>B4<br>C4<br>D4<br>E4<br>F4<br>9ED4<br>9EE4 | ii<br>dd<br>hh ll<br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| ASL *opr8a*<br>ASLA<br>ASLX<br>ASL *oprx8*,X<br>ASL ,X<br>ASL *oprx8*,SP | Arithmetic Shift Left (Same as LSL) |  | | | – | – | | | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38<br>48<br>58<br>68<br>78<br>9E68 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| ASR *opr8a*<br>ASRA<br>ASRX<br>ASR *oprx8*,X<br>ASR ,X<br>ASR *oprx8*,SP | Arithmetic Shift Right |  | | | – | – | | | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 37<br>47<br>57<br>67<br>77<br>9E67 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| BCC *rel* | Branch if Carry Bit Clear | Branch if (C) = 0 | – | – | – | – | – | – | REL | 24 | rr | 3 |

## Table 7-2. HCS08 Instruction Set Summary (Sheet 2 of 7)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| BCLR *n,opr8a* | Clear Bit n in Memory | Mn ← 0 | – | – | – | – | – | – | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 11<br>13<br>15<br>17<br>19<br>1B<br>1D<br>1F | dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BCS *rel* | Branch if Carry Bit Set (Same as BLO) | Branch if (C) = 1 | – | – | – | – | – | – | REL | 25 | rr | 3 |
| BEQ *rel* | Branch if Equal | Branch if (Z) = 1 | – | – | – | – | – | – | REL | 27 | rr | 3 |
| BGE *rel* | Branch if Greater Than or Equal To (Signed Operands) | Branch if (N ⊕ V) = 0 | – | – | – | – | – | – | REL | 90 | rr | 3 |
| BGND | Enter Active Background if ENBDM = 1 | Waits For and Processes BDM Commands Until GO, TRACE1, or TAGGO | – | – | – | – | – | – | INH | 82 | | 5+ |
| BGT *rel* | Branch if Greater Than (Signed Operands) | Branch if (Z) ǀ (N ⊕ V) = 0 | – | – | – | – | – | – | REL | 92 | rr | 3 |
| BHCC *rel* | Branch if Half Carry Bit Clear | Branch if (H) = 0 | – | – | – | – | – | – | REL | 28 | rr | 3 |
| BHCS *rel* | Branch if Half Carry Bit Set | Branch if (H) = 1 | – | – | – | – | – | – | REL | 29 | rr | 3 |
| BHI *rel* | Branch if Higher | Branch if (C) ǀ (Z) = 0 | – | – | – | – | – | – | REL | 22 | rr | 3 |
| BHS *rel* | Branch if Higher or Same (Same as BCC) | Branch if (C) = 0 | – | – | – | – | – | – | REL | 24 | rr | 3 |
| BIH *rel* | Branch if IRQ Pin High | Branch if IRQ pin = 1 | – | – | – | – | – | – | REL | 2F | rr | 3 |
| BIL *rel* | Branch if IRQ Pin Low | Branch if IRQ pin = 0 | – | – | – | – | – | – | REL | 2E | rr | 3 |
| BIT #*opr8i*<br>BIT *opr8a*<br>BIT *opr16a*<br>BIT *oprx16*,X<br>BIT *oprx8*,X<br>BIT ,X<br>BIT *oprx16*,SP<br>BIT *oprx8*,SP | Bit Test | (A) & (M)<br>(CCR Updated but Operands Not Changed) | 0 | – | – | | | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A5<br>B5<br>C5<br>D5<br>E5<br>F5<br>9ED5<br>9EE5 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| BLE *rel* | Branch if Less Than or Equal To (Signed Operands) | Branch if (Z) ǀ (N ⊕ V) = 1 | – | – | – | – | – | – | REL | 93 | rr | 3 |
| BLO *rel* | Branch if Lower (Same as BCS) | Branch if (C) = 1 | – | – | – | – | – | – | REL | 25 | rr | 3 |
| BLS *rel* | Branch if Lower or Same | Branch if (C) ǀ (Z) = 1 | – | – | – | – | – | – | REL | 23 | rr | 3 |
| BLT *rel* | Branch if Less Than (Signed Operands) | Branch if (N ⊕ V ) = 1 | – | – | – | – | – | – | REL | 91 | rr | 3 |
| BMC *rel* | Branch if Interrupt Mask Clear | Branch if (I) = 0 | – | – | – | – | – | – | REL | 2C | rr | 3 |
| BMI *rel* | Branch if Minus | Branch if (N) = 1 | – | – | – | – | – | – | REL | 2B | rr | 3 |
| BMS *rel* | Branch if Interrupt Mask Set | Branch if (I) = 1 | – | – | – | – | – | – | REL | 2D | rr | 3 |
| BNE *rel* | Branch if Not Equal | Branch if (Z) = 0 | – | – | – | – | – | – | REL | 26 | rr | 3 |
| BPL *rel* | Branch if Plus | Branch if (N) = 0 | – | – | – | – | – | – | REL | 2A | rr | 3 |
| BRA *rel* | Branch Always | No Test | – | – | – | – | – | – | REL | 20 | rr | 3 |

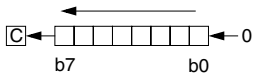
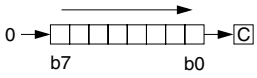
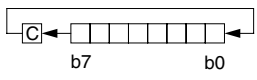
## Table 7-2. HCS08 Instruction Set Summary (Sheet 3 of 7)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| BRCLR *n,opr8a,rel* | Branch if Bit *n* in Memory Clear | Branch if (Mn) = 0 | – | – | – | – | – | | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 01<br>03<br>05<br>07<br>09<br>0B<br>0D<br>0F | dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BRN *rel* | Branch Never | Uses 3 Bus Cycles | – | – | – | – | – | – | REL | 21 | rr | 3 |
| BRSET *n,opr8a,rel* | Branch if Bit *n* in Memory Set | Branch if (Mn) = 1 | – | – | – | – | – | | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 00<br>02<br>04<br>06<br>08<br>0A<br>0C<br>0E | dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BSET *n,opr8a* | Set Bit *n* in Memory | Mn ← 1 | – | – | – | – | – | – | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 10<br>12<br>14<br>16<br>18<br>1A<br>1C<br>1E | dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BSR *rel* | Branch to Subroutine | PC ← (PC) + 0x0002<br>push (PCL); SP ← (SP) – 0x0001<br>push (PCH); SP ← (SP) – 0x0001<br>PC ← (PC) + *rel* | – | – | – | – | – | – | REL | AD | rr | 5 |
| CBEQ *oprx8,rel*<br>CBEQA *#opr8i,rel*<br>CBEQX *#opr8i,rel*<br>CBEQ *oprx8*,X+,*rel*<br>CBEQ ,X+,*rel*<br>CBEQ *oprx8*,SP,*rel* | Compare and Branch if Equal | Branch if (A) = (M)<br>Branch if (A) = (M)<br>Branch if (X) = (M)<br>Branch if (A) = (M)<br>Branch if (A) = (M)<br>Branch if (A) = (M) | – | – | – | – | – | – | DIR<br>IMM<br>IMM<br>IX1+<br>IX+<br>SP1 | 31<br>41<br>51<br>61<br>71<br>9E61 | dd rr<br>ii rr<br>ii rr<br>ff rr<br>rr<br>ff rr | 5<br>4<br>4<br>5<br>5<br>6 |
| CLC | Clear Carry Bit | C ← 0 | – | – | – | – | – | 0 | INH | 98 | | 1 |
| CLI | Clear Interrupt Mask Bit | I ← 0 | – | – | 0 | – | – | – | INH | 9A | | 1 |
| CLR *oprx8*,X<br>CLR ,X<br>CLR *oprx8*,SP | Clear | M ← 0x00<br>A ← 0x00<br>X ← 0x00<br>H ← 0x00<br>M ← 0x00<br>M ← 0x00<br>M ← 0x00 | 0 | – | – | 0 | 1 | – | DIR<br>INH<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3F<br>4F<br>5F<br>8C<br>6F<br>7F<br>9E6F | dd<br><br><br><br>ff<br><br>ff | 5<br>1<br>1<br>1<br>5<br>4<br>6 |
| CMP *#opr8i*<br>CMP *opr8a*<br>CMP *opr16a*<br>CMP *oprx16*,X<br>CMP *oprx8*,X<br>CMP ,X<br>CMP *oprx16*,SP<br>CMP *oprx8*,SP | Compare Accumulator with Memory | (A) – (M)<br>(CCR Updated But Operands Not Changed) | | – | – | | | | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A1<br>B1<br>C1<br>D1<br>E1<br>F1<br>9ED1<br>9EE1 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| COM *oprx8*,X<br>COM ,X<br>COM *oprx8*,SP | Complement (One's Complement) | M ← (M)= 0xFF – (M)<br>A ← (A) = 0xFF – (A)<br>X ← (X) = 0xFF – (X)<br>M ← (M) = 0xFF – (M)<br>M ← (M) = 0xFF – (M)<br>M ← (M) = 0xFF – (M) | 0 | – | – | | | 1 | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 33<br>43<br>53<br>63<br>73<br>9E63 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| CPHX *opr16a*<br>CPHX *#opr16i*<br>CPHX *opr8a*<br>CPHX *oprx8*,SP | Compare Index Register (H:X) with Memory | (H:X) – (M:M + 0x0001)<br>(CCR Updated But Operands Not Changed) | | – | – | | | | EXT<br>IMM<br>DIR<br>SP1 | 3E<br>65<br>75<br>9EF3 | hh ll<br>jj kk<br>dd<br>ff | 6<br>3<br>5<br>6 |

## Table 7-2. HCS08 Instruction Set Summary (Sheet 4 of 7)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| CPX #*opr8i*<br>CPX *opr8a*<br>CPX *opr16a*<br>CPX *oprx16*,X<br>CPX *oprx8*,X<br>CPX ,X<br>CPX *oprx16*,SP<br>CPX *oprx8*,SP | Compare X (Index Register Low) with Memory | (X) – (M)<br>(CCR Updated But Operands Not Changed) | | – | – | | | | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A3<br>B3<br>C3<br>D3<br>E3<br>F3<br>9ED3<br>9EE3 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| DAA | Decimal Adjust Accumulator After ADD or ADC of BCD Values | $(A)_{10}$ | U | – | – | | | | INH | 72 | | 1 |
| DBNZ *opr8a*,*rel*<br>DBNZA *rel*<br>DBNZX *rel*<br>DBNZ *oprx8*,X,*rel*<br>DBNZ ,X,*rel*<br>DBNZ *oprx8*,SP,*rel* | Decrement and Branch if Not Zero | Decrement A, X, or M<br>Branch if (result) ≠ 0<br>DBNZX Affects X Not H | – | – | – | – | – | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3B<br>4B<br>5B<br>6B<br>7B<br>9E6B | dd rr<br>rr<br>rr<br>ff rr<br>rr<br>ff rr | 7<br>4<br>4<br>7<br>6<br>8 |
| DEC *opr8a*<br>DECA<br>DECX<br>DEC *oprx8*,X<br>DEC ,X<br>DEC *oprx8*,SP | Decrement | M ← (M) – 0x01<br>A ← (A) – 0x01<br>X ← (X) – 0x01<br>M ← (M) – 0x01<br>M ← (M) – 0x01<br>M ← (M) – 0x01 | | – | – | | | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3A<br>4A<br>5A<br>6A<br>7A<br>9E6A | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| DIV | Divide | A ← (H:A)÷(X)<br>H ← Remainder | – | – | – | – | | | INH | 52 | | 6 |
| EOR #*opr8i*<br>EOR *opr8a*<br>EOR *opr16a*<br>EOR *oprx16*,X<br>EOR *oprx8*,X<br>EOR ,X<br>EOR *oprx16*,SP<br>EOR *oprx8*,SP | Exclusive OR Memory with Accumulator | A ← (A ⊕ M) | 0 | – | – | | | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A8<br>B8<br>C8<br>D8<br>E8<br>F8<br>9ED8<br>9EE8 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| INC *opr8a*<br>INCA<br>INCX<br>INC *oprx8*,X<br>INC ,X<br>INC *oprx8*,SP | Increment | M ← (M) + 0x01<br>A ← (A) + 0x01<br>X ← (X) + 0x01<br>M ← (M) + 0x01<br>M ← (M) + 0x01<br>M ← (M) + 0x01 | | – | – | | | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3C<br>4C<br>5C<br>6C<br>7C<br>9E6C | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| JMP *opr8a*<br>JMP *opr16a*<br>JMP *oprx16*,X<br>JMP *oprx8*,X<br>JMP ,X | Jump | PC ← Jump Address | – | – | – | – | – | – | DIR<br>EXT<br>IX2<br>IX1<br>IX | BC<br>CC<br>DC<br>EC<br>FC | dd<br>hh ll<br>ee ff<br>ff<br> | 3<br>4<br>4<br>3<br>3 |
| JSR *opr8a*<br>JSR *opr16a*<br>JSR *oprx16*,X<br>JSR *oprx8*,X<br>JSR ,X | Jump to Subroutine | PC ← (PC) + *n* (*n* = 1, 2, or 3)<br>Push (PCL); SP ← (SP) – 0x0001<br>Push (PCH); SP ← (SP) – 0x0001<br>PC ← Unconditional Address | – | – | – | – | – | – | DIR<br>EXT<br>IX2<br>IX1<br>IX | BD<br>CD<br>DD<br>ED<br>FD | dd<br>hh ll<br>ee ff<br>ff<br> | 5<br>6<br>6<br>5<br>5 |
| LDA #*opr8i*<br>LDA *opr8a*<br>LDA *opr16a*<br>LDA *oprx16*,X<br>LDA *oprx8*,X<br>LDA ,X<br>LDA *oprx16*,SP<br>LDA *oprx8*,SP | Load Accumulator from Memory | A ← (M) | 0 | – | – | | | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A6<br>B6<br>C6<br>D6<br>E6<br>F6<br>9ED6<br>9EE6 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| LDHX #*opr16i*<br>LDHX *opr8a*<br>LDHX *opr16a*<br>LDHX ,X<br>LDHX *oprx16*,X<br>LDHX *oprx8*,X<br>LDHX *oprx8*,SP | Load Index Register (H:X) from Memory | H:X ← (M:M + 0x0001) | 0 | – | – | | | – | IMM<br>DIR<br>EXT<br>IX<br>IX2<br>IX1<br>SP1 | 45<br>55<br>32<br>9EAE<br>9EBE<br>9ECE<br>9EFE | jj kk<br>dd<br>hh ll<br><br>ee ff<br>ff<br>ff | 3<br>4<br>5<br>5<br>6<br>5<br>5 |

## Table 7-2. HCS08 Instruction Set Summary (Sheet 5 of 7)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| LDX #*oprx8i*<br>LDX *oprx8*<br>LDX *oprx16*<br>LDX *oprx16*,X<br>LDX *oprx8*,X<br>LDX ,X<br>LDX *oprx16*,SP<br>LDX *oprx8*,SP | Load X (Index Register Low) from Memory | X ← (M) | 0 | – | – | | | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AE<br>BE<br>CE<br>DE<br>EE<br>FE<br>9EDE<br>9EEE | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| LSL *oprx8*<br>LSLA<br>LSLX<br>LSL *oprx8*,X<br>LSL ,X<br>LSL *oprx8*,SP | Logical Shift Left (Same as ASL) |  | | – | – | | | | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38<br>48<br>58<br>68<br>78<br>9E68 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| LSR *oprx8*<br>LSRA<br>LSR*X*<br>LSR *oprx8*,X<br>LSR ,X<br>LSR *oprx8*,SP | Logical Shift Right |  | | – | – | 0 | | | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 34<br>44<br>54<br>64<br>74<br>9E64 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| MOV *oprx8*,*oprx8*<br>MOV *oprx8*,X+<br>MOV #*oprx8*,*oprx8*<br>MOV ,X+,*oprx8* | Move | $(M)_{destination} \leftarrow (M)_{source}$<br>H:X ← (H:X) + 0x0001 in IX+/DIR and DIR/IX+ Modes | 0 | – | – | | | – | DIR/DIR<br>DIR/IX+<br>IMM/DIR<br>IX+/DIR | 4E<br>5E<br>6E<br>7E | dd dd<br>dd<br>ii dd<br>dd | 5<br>5<br>4<br>5 |
| MUL | Unsigned multiply | X:A ← (X) × (A) | – | 0 | – | – | – | 0 | INH | 42 | | 5 |
| NEG *oprx8*<br>NEGA<br>NEGX<br>NEG *oprx8*,X<br>NEG ,X<br>NEG *oprx8*,SP | Negate (Two's Complement) | M ← – (M) = 0x00 – (M)<br>A ← – (A) = 0x00 – (A)<br>X ← – (X) = 0x00 – (X)<br>M ← – (M) = 0x00 – (M)<br>M ← – (M) = 0x00 – (M)<br>M ← – (M) = 0x00 – (M) | | – | – | | | | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 30<br>40<br>50<br>60<br>70<br>9E60 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| NOP | No Operation | Uses 1 Bus Cycle | – | – | – | – | – | – | INH | 9D | | 1 |
| NSA | Nibble Swap Accumulator | A ← (A[3:0]:A[7:4]) | – | – | – | – | – | – | INH | 62 | | 1 |
| ORA #*oprx8i*<br>ORA *oprx8*<br>ORA *oprx16*<br>ORA *oprx16*,X<br>ORA *oprx8*,X<br>ORA ,X<br>ORA *oprx16*,SP<br>ORA *oprx8*,SP | Inclusive OR Accumulator and Memory | A ← (A) \| (M) | 0 | – | – | | | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AA<br>BA<br>CA<br>DA<br>EA<br>FA<br>9EDA<br>9EEA | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| PSHA | Push Accumulator onto Stack | Push (A); SP ← (SP) – 0x0001 | – | – | – | – | – | – | INH | 87 | | 2 |
| PSHH | Push H (Index Register High) onto Stack | Push (H); SP ← (SP) – 0x0001 | – | – | – | – | – | – | INH | 8B | | 2 |
| PSHX | Push X (Index Register Low) onto Stack | Push (X); SP ← (SP) – 0x0001 | – | – | – | – | – | – | INH | 89 | | 2 |
| PULA | Pull Accumulator from Stack | SP ← (SP + 0x0001); Pull (A) | – | – | – | – | – | – | INH | 86 | | 3 |
| PULH | Pull H (Index Register High) from Stack | SP ← (SP + 0x0001); Pull (H) | – | – | – | – | – | – | INH | 8A | | 3 |
| PULX | Pull X (Index Register Low) from Stack | SP ← (SP + 0x0001); Pull (X) | – | – | – | – | – | – | INH | 88 | | 3 |
| ROL *oprx8*<br>ROLA<br>ROLX<br>ROL *oprx8*,X<br>ROL ,X<br>ROL *oprx8*,SP | Rotate Left through Carry |  | | – | – | | | | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 39<br>49<br>59<br>69<br>79<br>9E69 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |

## Table 7-2. HCS08 Instruction Set Summary (Sheet 6 of 7)

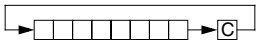
| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| ROR *oprx8*,SP | Rotate Right through Carry |  | | – | – | | | | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 36<br>46<br>56<br>66<br>76<br>9E66 | dd<br><br><br>ff<br><br>ff | 5<br>1<br>1<br>5<br>4<br>6 |
| RSP | Reset Stack Pointer | SP ← 0xFF<br>(High Byte Not Affected) | – | – | – | – | – | – | INH | 9C | | 1 |
| RTI | Return from Interrupt | SP ← (SP) + 0x0001; Pull (CCR)<br>SP ← (SP) + 0x0001; Pull (A)<br>SP ← (SP) + 0x0001; Pull (X)<br>SP ← (SP) + 0x0001; Pull (PCH)<br>SP ← (SP) + 0x0001; Pull (PCL) | | | | | | | INH | 80 | | 9 |
| RTS | Return from Subroutine | SP ← SP + 0x0001; Pull (PCH)<br>SP ← SP + 0x0001; Pull (PCL) | – | – | – | – | – | – | INH | 81 | | 6 |
| SBC #*opr8i*<br>SBC *opr8a*<br>SBC *opr16a*<br>SBC *oprx16*,X<br>SBC *oprx8*,X<br>SBC ,X<br>SBC *oprx16*,SP<br>SBC *oprx8*,SP | Subtract with Carry | A ← (A) – (M) – (C) | | – | – | | | | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A2<br>B2<br>C2<br>D2<br>E2<br>F2<br>9ED2<br>9EE2 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| SEC | Set Carry Bit | C ← 1 | – | – | – | – | – | 1 | INH | 99 | | 1 |
| SEI | Set Interrupt Mask Bit | I ← 1 | – | – | 1 | – | – | – | INH | 9B | | 1 |
| STA *opr8a*<br>STA *opr16a*<br>STA *oprx16*,X<br>STA *oprx8*,X<br>STA ,X<br>STA *oprx16*,SP<br>STA *oprx8*,SP | Store Accumulator in Memory | M ← (A) | 0 | – | – | | | – | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | B7<br>C7<br>D7<br>E7<br>F7<br>9ED7<br>9EE7 | dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 3<br>4<br>4<br>3<br>2<br>5<br>4 |
| STHX *opr8a*<br>STHX *opr16a*<br>STHX *oprx8*,SP | Store H:X (Index Reg.) | (M:M + 0x0001) ← (H:X) | 0 | – | – | | | – | DIR<br>EXT<br>SP1 | 35<br>96<br>9EFF | dd<br>hh ll<br>ff | 4<br>5<br>5 |
| STOP | Enable Interrupts: Stop Processing Refer to MCU Documentation | I bit ← 0; Stop Processing | – | – | 0 | – | – | – | INH | 8E | | 2+ |
| STX *opr8a*<br>STX *opr16a*<br>STX *oprx16*,X<br>STX *oprx8*,X<br>STX ,X<br>STX *oprx16*,SP<br>STX *oprx8*,SP | Store X (Low 8 Bits of Index Register) in Memory | M ← (X) | 0 | – | – | | | – | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | BF<br>CF<br>DF<br>EF<br>FF<br>9EDF<br>9EEF | dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 3<br>4<br>4<br>3<br>2<br>5<br>4 |
| SUB #*opr8i*<br>SUB *opr8a*<br>SUB *opr16a*<br>SUB *oprx16*,X<br>SUB *oprx8*,X<br>SUB ,X<br>SUB *oprx16*,SP<br>SUB *oprx8*,SP | Subtract | A ← (A) – (M) | | – | – | | | | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A0<br>B0<br>C0<br>D0<br>E0<br>F0<br>9ED0<br>9EE0 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ee ff<br>ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 |
| SWI | Software Interrupt | PC ← (PC) + 0x0001<br>Push (PCL); SP ← (SP) – 0x0001<br>Push (PCH); SP ← (SP) – 0x0001<br>Push (X); SP ← (SP) – 0x0001<br>Push (A); SP ← (SP) – 0x0001<br>Push (CCR); SP ← (SP) – 0x0001<br>I ← 1;<br>PCH ← Interrupt Vector High Byte<br>PCL ← Interrupt Vector Low Byte | – | – | 1 | – | – | – | INH | 83 | | 11 |

**Table 7-2. HCS08 Instruction Set Summary (Sheet 7 of 7)**

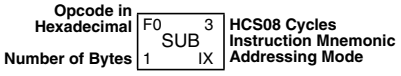
| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Bus Cycles[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| TAP | Transfer Accumulator to CCR | CCR ← (A) | | | | | | | INH | 84 | | 1 |
| TAX | Transfer Accumulator to X (Index Register Low) | X ← (A) | – | – | – | – | – | – | INH | 97 | | 1 |
| TPA | Transfer CCR to Accumulator | A ← (CCR) | – | – | – | – | – | – | INH | 85 | | 1 |
| TST *oprx8*,X<br>TST ,X<br>TST *oprx8*,SP | Test for Negative or Zero | (M) – 0x00<br>(A) – 0x00<br>(X) – 0x00<br>(M) – 0x00<br>(M) – 0x00<br>(M) – 0x00 | 0 | – | – | | | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3D<br>4D<br>5D<br>6D<br>7D<br>9E6D | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| TSX | Transfer SP to Index Reg. | H:X ← (SP) + 0x0001 | – | – | – | – | – | – | INH | 95 | | 2 |
| TXA | Transfer X (Index Reg. Low) to Accumulator | A ← (X) | – | – | – | – | – | – | INH | 9F | | 1 |
| TXS | Transfer Index Reg. to SP | SP ← (H:X) – 0x0001 | – | – | – | – | – | – | INH | 94 | | 2 |
| WAIT | Enable Interrupts; Wait for Interrupt | I bit ← 0; Halt CPU | – | – | 0 | – | – | – | INH | 8F | | 2+ |

[1] Bus clock frequency is one-half of the CPU clock frequency.

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

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

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

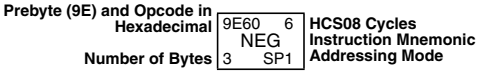


## Table 7-3. Opcode Map (Sheet 2 of 2)

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

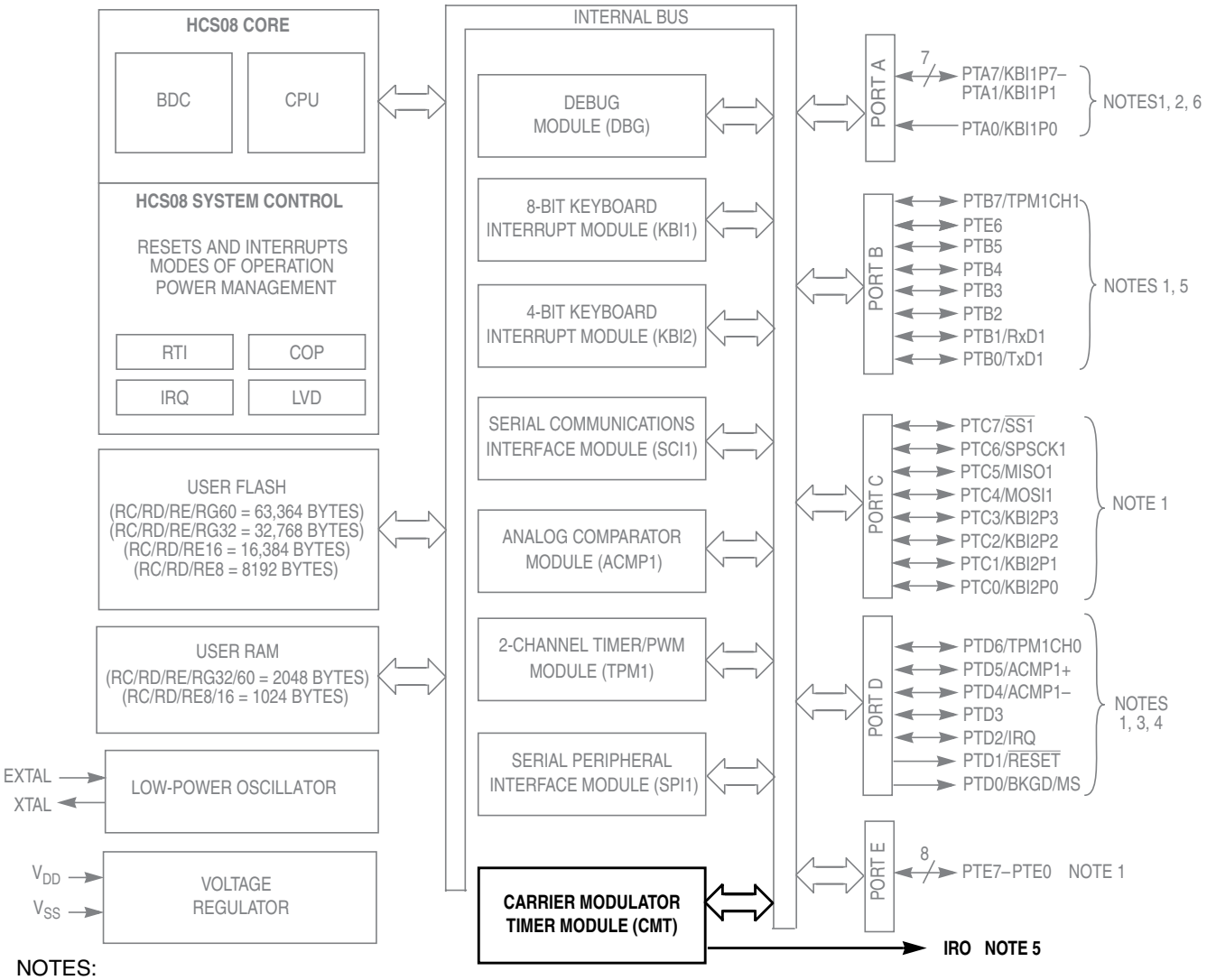
| | | | | | |
|---|---|---|---|---|---|
| **INH** | **Inherent** | **REL** | **Relative** | **SP1** | **Stack Pointer, 8-Bit Offset** |
| **IMM** | **Immediate** | **IX** | **Indexed, No Offset** | **SP2** | **Stack Pointer, 16-Bit Offset** |
| **DIR** | **Direct** | **IX1** | **Indexed, 8-Bit Offset**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

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



# Chapter 8
# Carrier Modulator Timer (S08CMTV1)

## 8.1 Introduction




NOTES:
1. Port pins are software configurable with pullup device if input port
2. PTA0 does not have a clamp diode to VDD. PTA0 should not be driven above VDD. Also, PTA0 does not pullup to VDD when internal pullup is enabled.
3. IRQ pin contains software configurable pullup/pulldown device if IRQ enabled (IRQPE = 1)
4. The RESET pin contains integrated pullup device enabled if reset enabled (RSTPE = 1)
5. High current drive
6. Pins PTA[7:4] contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

**Figure 8-1. MC9S08RC/RD/RE/RG Block Diagram**

## 8.2 Features

The CMT consists of a carrier generator, modulator, and transmitter that drives the infrared out (IRO) pin. The features of this module include:

- Four modes of operation
  - Time with independent control of high and low times
  - Baseband
  - Frequency shift key (FSK)
  - Direct software control of IRO pin
- Extended space operation in time, baseband, and FSK modes
- Selectable input clock divide: 1, 2, 4, or 8
- Interrupt on end of cycle
  - Ability to disable IRO pin and use as timer interrupt

## 8.3 CMT Block Diagram

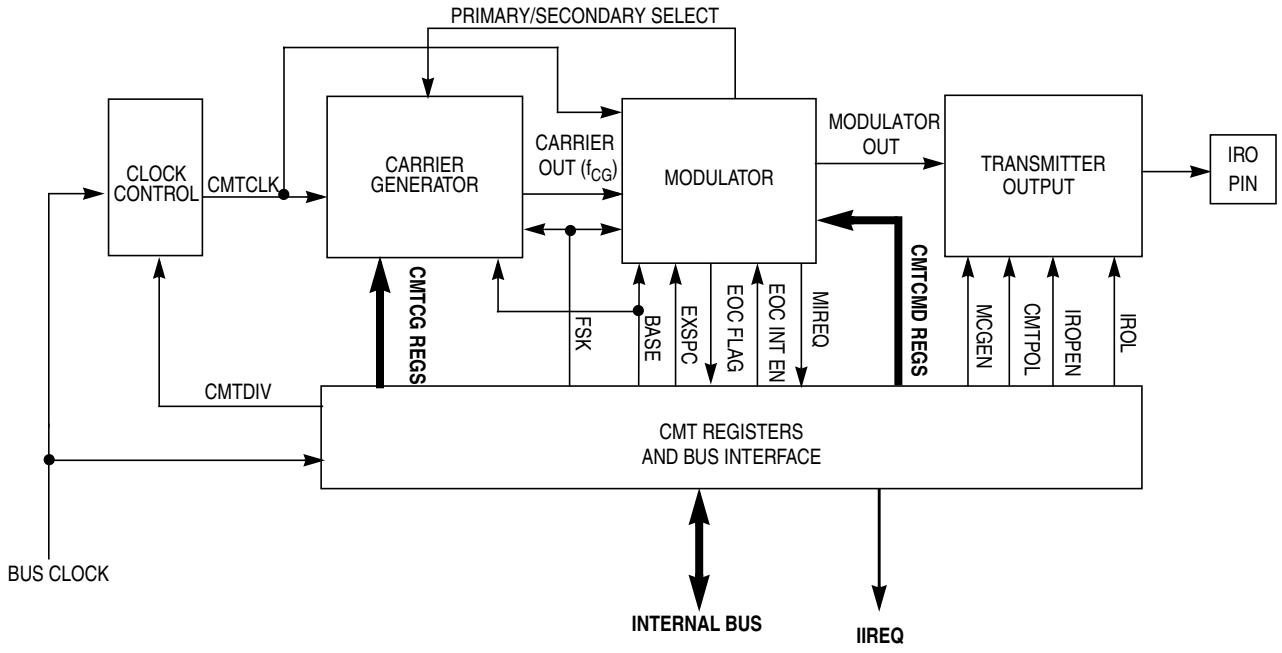



**Figure 8-2. Carrier Modulator Transmitter Module Block Diagram**

## 8.4 Pin Description

The IRO pin is the only pin associated with the CMT. The pin is driven by the transmitter output when the MCGEN bit in the CMTMSC register and the IROPEN bit in the CMTOC register are set. If the MCGEN bit is clear and the IROPEN bit is set, the pin is driven by the IROL bit in the CMTOC register. This enables user software to directly control the state of the IRO pin by writing to the IROL bit. If the IROPEN bit is clear, the pin is disabled and is not driven by the CMT module. This is so the CMT can be configured as a modulo timer for generating periodic interrupts without causing pin activity.

## 8.5 Functional Description

The CMT module consists of a carrier generator, a modulator, a transmitter output, and control registers. The block diagram is shown in Figure 8-2. When operating in time mode, the user independently defines the high and low times of the carrier signal to determine both period and duty cycle. The carrier generator resolution is 125 ns when operating with an 8 MHz internal bus frequency and the CMTDIV1 and CMTDIV0 bits in the CMTMSC register are both equal to 0. The carrier generator can generate signals with periods between 250 ns (4 MHz) and 127.5 µs (7.84 kHz) in steps of 125 ns. See Table 8-1.

**Table 8-1. Clock Divide**

| Bus Clock (MHz) | CMTDIV1:CMTDIV0 | Carrier Generator Resolution (µs) | Min Carrier Generator Period (µs) | Min Modulator Period (µs) |
|---|---|---|---|---|
| 8 | 0:0 | 0.125 | 0.25 | 1.0 |
| 8 | 0:1 | 0.25 | 0.5 | 2.0 |
| 8 | 1:0 | 0.5 | 1.0 | 4.0 |
| 8 | 1:1 | 1.0 | 2.0 | 8.0 |

The possible duty cycle options will depend upon the number of counts required to complete the carrier period. For example, a 1.6 MHz signal has a period of 625 ns and will therefore require 5 × 125 ns counts to generate. These counts may be split between high and low times, so the duty cycles available will be 20 percent (one high, four low), 40 percent (two high, three low), 60 percent (three high, two low) and 80 percent (four high, one low).

For lower frequency signals with larger periods, higher resolution (as a percentage of the total period) duty cycles are possible.

When the BASE bit in the CMT modulator status and control register (CMTMSC) is set, the carrier output ($f_{CG}$) to the modulator is held high continuously to allow for the generation of baseband protocols.

A third mode allows the carrier generator to alternate between two sets of high and low times. When operating in FSK mode, the generator will toggle between the two sets when instructed by the modulator, allowing the user to dynamically switch between two carrier frequencies without CPU intervention.

The modulator provides a simple method to control protocol timing. The modulator has a minimum resolution of 1.0 µs with an 8 MHz internal bus clock. It can count bus clocks (to provide real-time control) or it can count carrier clocks (for self-clocked protocols). See Section 8.5.2, “Modulator,” for more details.

The transmitter output block controls the state of the infrared out pin (IRO). The modulator output is gated on to the IRO pin when the modulator/carrier generator is enabled.

A summary of the possible modes is shown in Table 8-2.

**Table 8-2. CMT Modes of Operation**

| Mode | MCGEN Bit[1] | BASE Bit[2] | FSK Bit[2] | EXSPC Bit | Comment |
|---|---|---|---|---|---|
| Time | 1 | 0 | 0 | 0 | $f_{CG}$ controlled by primary high and low registers.<br>$f_{CG}$ transmitted to IRO pin when modulator gate is open. |
| Baseband | 1 | 1 | x | 0 | $f_{CG}$ is always high. IRO pin high when modulator gate is open. |
| FSK | 1 | 0 | 1 | 0 | $f_{CG}$ control alternates between primary high/low registers and secondary high/low registers.<br>$f_{CG}$ transmitted to IRO pin when modulator gate is open. |
| Extended Space | 1 | x | x | 1 | Setting the EXSPC bit causes subsequent modulator cycles to be spaces (modulator out not asserted) for the duration of the modulator period (mark and space times). |
| IRO Latch | 0 | x | x | x | IROL bit controls state of IRO pin. |

1. To prevent spurious operation, initialize all data and control registers before beginning a transmission (MCGEN=1).
2. These bits are not double buffered and should not be changed during a transmission (while MCGEN=1).

## 8.5.1 Carrier Generator

The carrier signal is generated by counting a register-selected number of input clocks (125 ns for an 8 MHz bus) for both the carrier high time and the carrier low time. The period is determined by the total number of clocks counted. The duty cycle is determined by the ratio of high time clocks to total clocks counted. The high and low time values are user programmable and are held in two registers.

An alternate set of high/low count values is held in another set of registers to allow the generation of dual frequency FSK (frequency shift keying) protocols without CPU intervention.

**NOTE**

Only non-zero data values are allowed. The carrier generator will not work if any of the count values are equal to zero.

The MCGEN bit in the CMTMSC register must be set and the BASE bit must be cleared to enable carrier generator clocks. When the BASE bit is set, the carrier output to the modulator is held high continuously. The block diagram is shown in Figure 8-3.

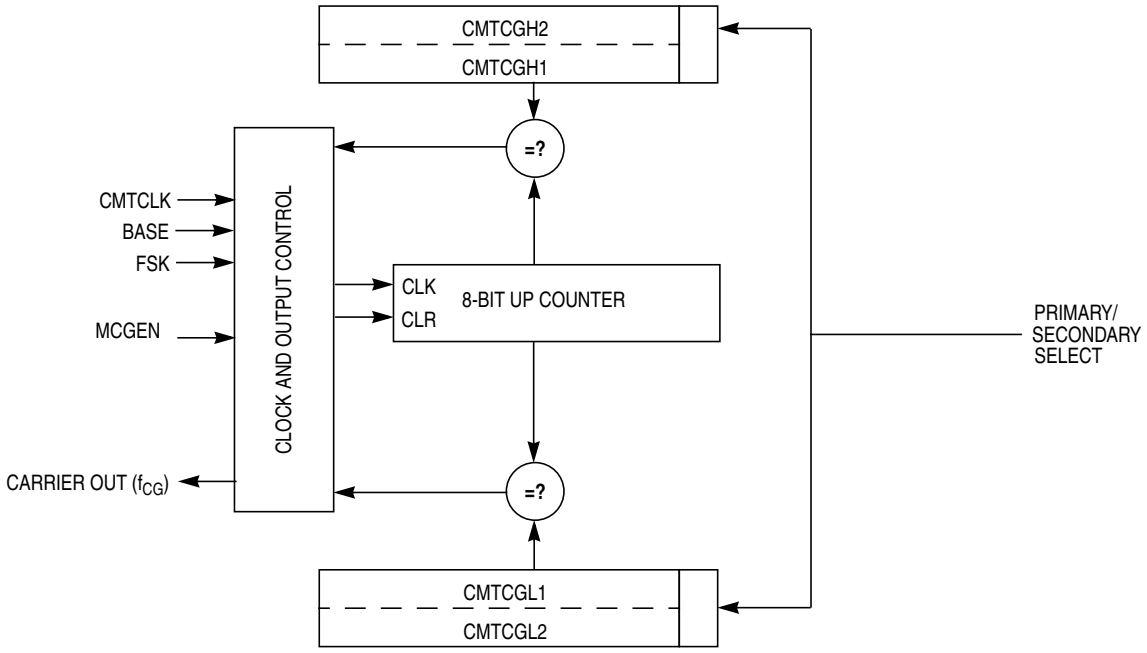



**Figure 8-3. Carrier Generator Block Diagram**

The high/low time counter is an 8-bit up counter. After each increment, the contents of the counter are compared with the appropriate high or low count value register. When the compare value is reached, the counter is reset to a value of $01, and the compare is redirected to the other count value register.

Assuming that the high time count compare register is currently active, a valid compare will cause the carrier output to be driven low. The counter will continue to increment (starting at reset value of $01). When the value stored in the selected low count value register is reached, the counter will again be reset and the carrier output will be driven high.

The cycle repeats, automatically generating a periodic signal that is directed to the modulator. The lowest frequency (maximum period) and highest frequency (minimum period) that can be generated are defined as:

$$f_{max} = f_{CMTCLK} \div (2 \times 1) \text{ Hz} \qquad \textit{Eqn. 8-1}$$

$$f_{min} = f_{CMTCLK} \div (2 \times (2^8 - 1)) \text{ Hz} \qquad \textit{Eqn. 8-2}$$

In the general case, the carrier generator output frequency is:

$$f_{CG} = f_{CMTCLK} \div (\text{Highcount} + \text{Lowcount}) \text{ Hz} \qquad \textit{Eqn. 8-3}$$

Where: $0 < \textit{Highcount} < 256$ and
$0 < \textit{Lowcount} < 256$

The duty cycle of the carrier signal is controlled by varying the ratio of high time to low + high time. As the input clock period is fixed, the duty cycle resolution will be proportional to the number of counts required to generate the desired carrier period.

$$\text{Duty Cycle} = \frac{\text{Highcount}}{\text{Highcount} + \text{Lowcount}} \qquad \textit{Eqn. 8-4}$$

## 8.5.2 Modulator

The modulator has three main modes of operation:

- Gate the carrier onto the modulator output (time mode)
- Control the logic level of the modulator output (baseband mode)
- Count carrier periods and instruct the carrier generator to alternate between two carrier frequencies whenever a modulation period (mark + space counts) expires (FSK mode)

The modulator includes a 17-bit down counter with underflow detection. The counter is loaded from the 16-bit modulation mark period buffer registers, CMTCMD1 and CMTCMD2. The most significant bit is loaded with a logic zero and serves as a sign bit. When the counter holds a positive value, the modulator gate is open and the carrier signal is driven to the transmitter block.

When the counter underflows, the modulator gate is closed and a 16-bit comparator is enabled that compares the logical complement of the value of the down-counter with the contents of the modulation space period register (which has been loaded from the registers CMTCMD3 and CMTCMD4).

When a match is obtained the cycle repeats by opening the modulator gate, reloading the counter with the contents of CMTCMD1 and CMTCMD2, and reloading the modulation space period register with the contents of CMTCMD3 and CMTCMD4.

If the contents of the modulation space period register are all zeroes, the match will be immediate and no space period will be generated (for instance, for FSK protocols that require successive bursts of different frequencies).

The MCGEN bit in the CMTMSC register must be set to enable the modulator timer.

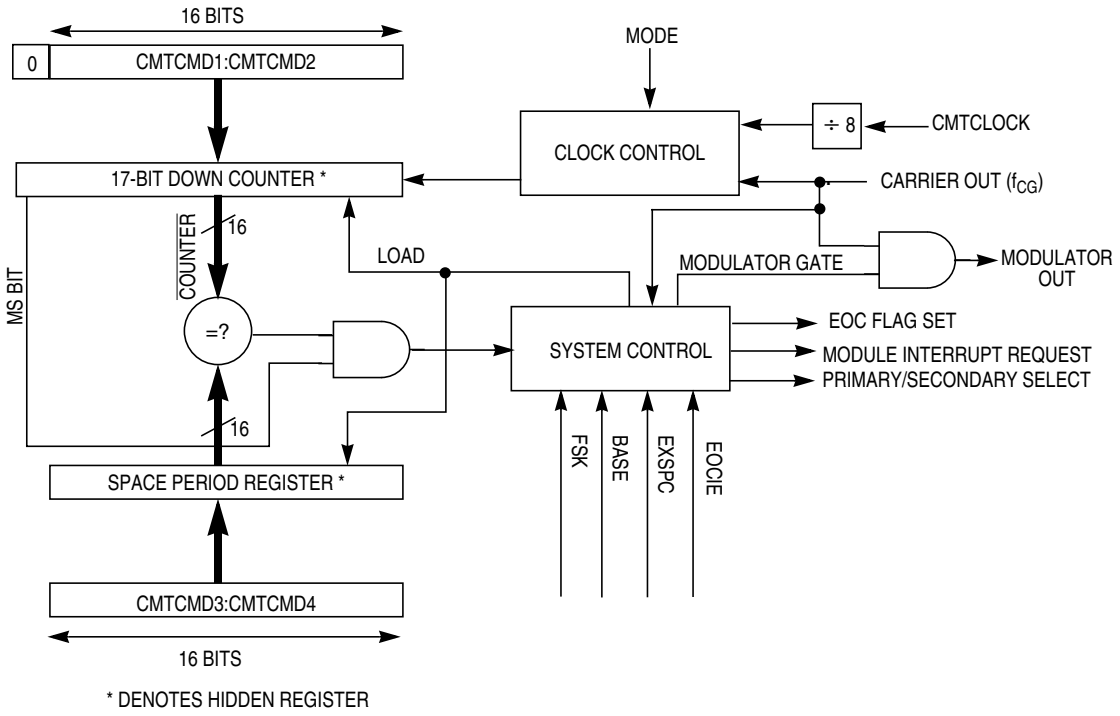



**Figure 8-4. Modulator Block Diagram**

### 8.5.2.1 Time Mode

When the modulator operates in time mode (MCGEN bit is set, BASE bit is clear, and FSK bit is clear), the modulation mark period consists of an integer number of CMTCLK ÷ 8 clock periods. The modulation space period consists of zero or an integer number of CMTCLK ÷ 8 clock periods. With an 8 MHz bus and CMTDIV1:CMTDIV0 = 00, the modulator resolution is 1 µs and has a maximum mark and space period of about 65.535 ms each. See Figure 8-5 for an example of the time mode and baseband mode outputs.

The mark and space time equations for time and baseband mode are:

$$t_{mark} = (CMTCMD1{:}CMTCMD2 + 1) \div (f_{CMTCLK} \div 8) \qquad \textit{Eqn. 8-5}$$

$$t_{space} = CMTCMD3{:}CMTCMD4 \div (f_{CMTCLK} \div 8) \qquad \textit{Eqn. 8-6}$$

where CMTCMD1:CMTCMD2 and CMTCMD3:CMTCMD4 are the decimal values of the concatenated registers.

**NOTE**

If the modulator is disabled while the $t_{mark}$ time is less than the programmed carrier high time ($t_{mark} < CMTCGH1/f_{CMTCLK}$), the modulator can enter into an illegal state and end the curent cycle before the programmed value. Make sure to program $t_{mark}$ greater than the carrier high time to avoid this illegal state.

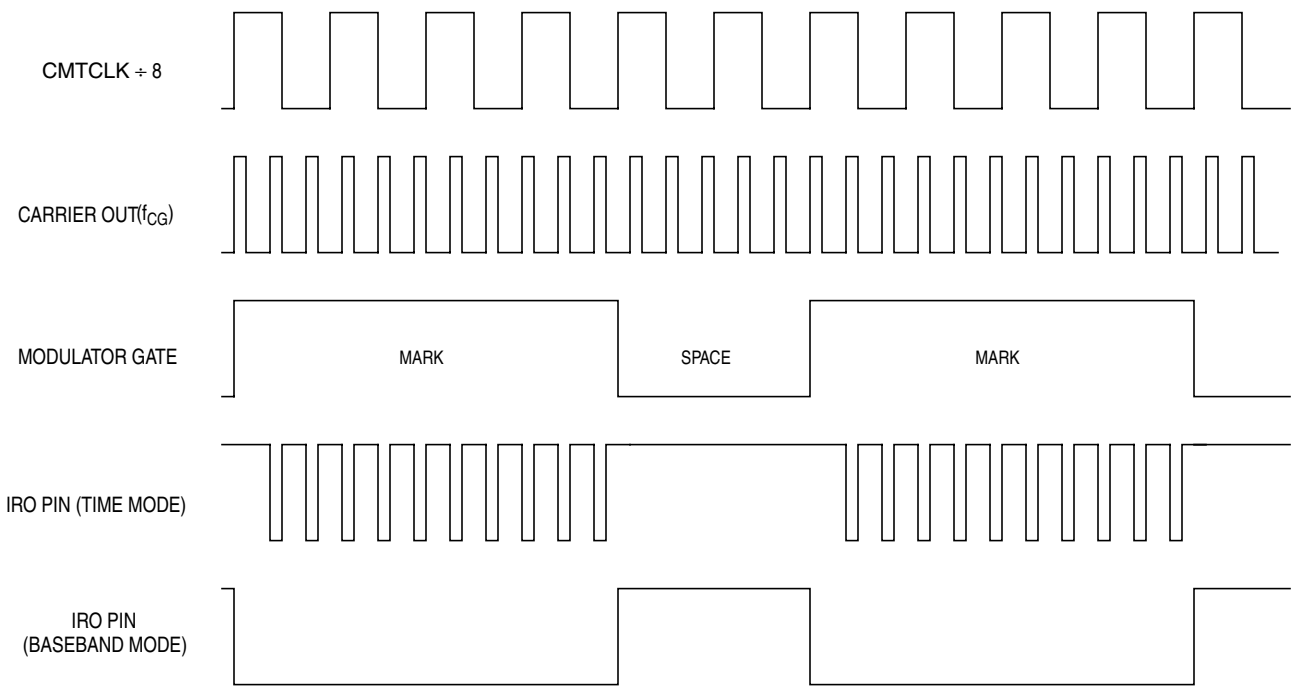



**Figure 8-5. Example CMT Output in Time and Baseband Modes**

### 8.5.2.2 Baseband Mode

Baseband mode (MCGEN bit is set and BASE bit is set) is a derivative of time mode, where the mark and space period is based on (CMTCLK ÷ 8) counts. The mark and space calculations are the same as in time mode. In this mode the modulator output will be at a logic 1 for the duration of the mark period and at a logic 0 for the duration of a space period. See Figure 8-5 for an example of the output for both baseband and time modes. In the example, the carrier out frequency ($f_{CG}$) is generated with a high count of $01 and a low count of $02, which results in a divide of 3 of CMTCLK with a 33 percent duty cycle. The modulator down-counter was loaded with the value $0003 and the space period register with $0002.

**NOTE**

The waveforms in Figure 8-5 and Figure 8-6 are for the purpose of conceptual illustration and are not meant to represent precise timing relationships between the signals shown.

### 8.5.2.3 FSK Mode

When the modulator operates in FSK mode (MCGEN bit is set, FSK bit is set, and BASE bit is clear), the modulation mark and space periods consist of an integer number of carrier clocks (space period can be 0). When the mark period expires, the space period is transparently started (as in time mode). The carrier generator toggles between primary and secondary data register values whenever the modulator space period expires.

The space period provides an interpulse gap (no carrier). If CMTCMD3:CMTCMD4 = $0000, then the modulator and carrier generator will switch between carrier frequencies without a gap or any carrier glitches (zero space).

Using timing data for carrier burst and interpulse gap length calculated by the CPU, FSK mode can automatically generate a phase-coherent, dual-frequency FSK signal with programmable burst and interburst gaps.

The mark and space time equations for FSK mode are:

$$t_{mark} = (CMTCMD1:CMTCMD2 + 1) \div f_{CG} \quad \textit{Eqn. 8-7}$$

$$t_{space} = CMTCMD3:CMTCMD4 \div f_{CG} \quad \textit{Eqn. 8-8}$$

Where $f_{CG}$ is the frequency output from the carrier generator. The example in Figure 8-6 shows what the IRO pin output looks like in FSK mode with the following values: CMTCMD1:CMTCMD2 = $0003, CMTCMD3:CMTCMD4 = $0002, primary carrier high count = $01, primary carrier low count = $02, secondary carrier high count = $03, and secondary carrier low count = $01.

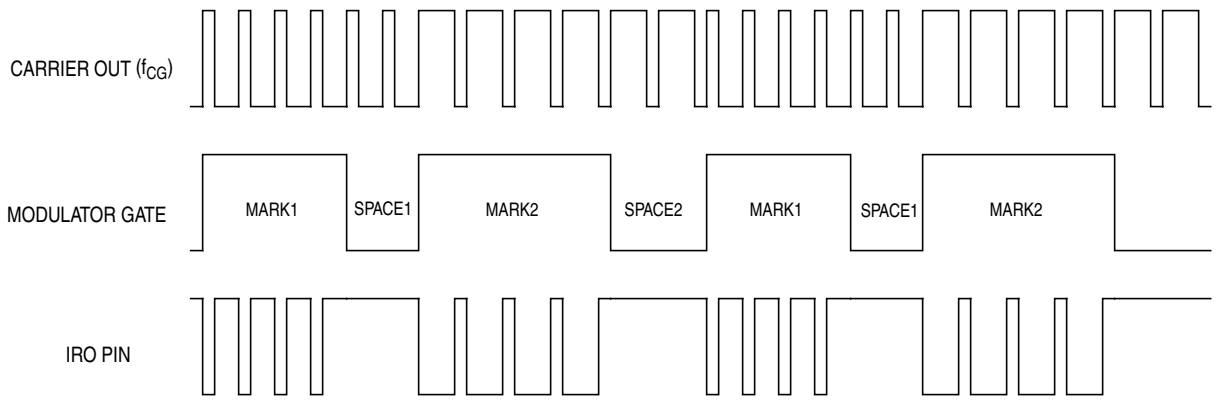



**Figure 8-6. Example CMT Output in FSK Mode**

## 8.5.3 Extended Space Operation

In time, baseband, or FSK mode, the space period can be made longer than the maximum possible value of the space period register. Setting the EXSPC bit in the CMTMSC register will force the modulator to treat the next modulation period (beginning with the next load of the counter and space period register) as a space period equal in length to the mark and space counts combined. Subsequent modulation periods will consist entirely of these extended space periods with no mark periods. Clearing EXSPC will return the modulator to standard operation at the beginning of the next modulation period.

### 8.5.3.1 EXSPC Operation in Time Mode

To calculate the length of an extended space in time or baseband modes, add the mark and space times and multiply by the number of modulation periods that EXSPC is set.

$$t_{exspace} = t_{space} + (t_{mark} + t_{space}) \times (\text{number of modulation periods}) \quad \textit{Eqn. 8-9}$$

For an example of extended space operation, see Figure 8-7.

**NOTE**

The EXSPC feature can be used to emulate a zero mark event.

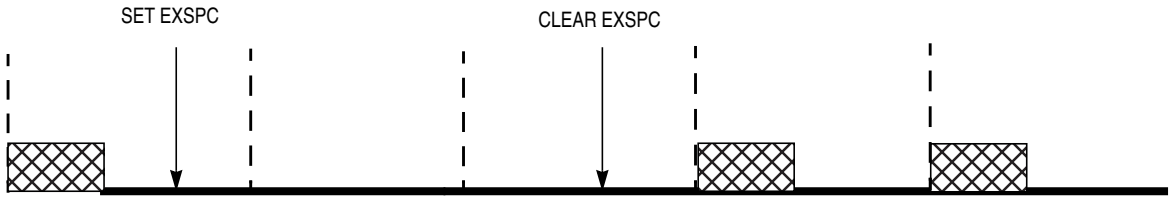



**Figure 8-7. Extended Space Operation**

### 8.5.3.2 EXSPC Operation in FSK Mode

In FSK mode, the modulator continues to count carrier out clocks, alternating between the primary and secondary registers at the end of each modulation period.

To calculate the length of an extended space in FSK mode, the user must know whether the EXSPC bit was set on a primary or secondary modulation period, as well as the total number of both primary and secondary modulation periods completed while the EXSPC bit is high. A status bit for the current modulation is not accessible to the CPU. If necessary, software should maintain tracking of the current modulation cycle (primary or secondary). The extended space period ends at the completion of the space period time of the modulation period during which the EXSPC bit is cleared.

If the EXSPC bit was set during a primary modulation cycle, use the equation:

$$t_{exspace} = (t_{space})_p + (t_{mark} + t_{space})_s + (t_{mark} + t_{space})_p + \ldots \quad \textit{Eqn. 8-10}$$

Where the subscripts p and s refer to mark and space times for the primary and secondary modulation cycles.

If the EXSPC bit was set during a secondary modulation cycle, use the equation:

$$t_{exspace} = (t_{space})_s + (t_{mark} + t_{space})_p + (t_{mark} + t_{space})_s + \ldots \quad \textit{Eqn. 8-11}$$

## 8.5.4 Transmitter

The transmitter output block controls the state of the infrared out pin (IRO). The modulator output is gated on to the IRO pin when the modulator/carrier generator is enabled. When the modulator/carrier generator is disabled, the IRO pin is controlled by the state of the IRO latch.

A polarity bit in the CMTOC register enables the IRO pin to be high true or low true.

### 8.5.5 CMT Interrupts

The end of cycle flag (EOCF) is set when:

- The modulator is not currently active and the MCGEN bit is set to begin the initial CMT transmission
- At the end of each modulation cycle (when the counter is reloaded from CMTCMD1:CMTCMD2) while the MCGEN bit is set

In the case where the MCGEN bit is cleared and then set before the end of the modulation cycle, the EOCF bit will not be set when the MCGEN is set, but will become set at the end of the current modulation cycle.

When the MCGEN becomes disabled, the CMT module does not set the EOC flag at the end of the last modulation cycle.

The EOCF bit is cleared by reading the CMT modulator status and control register (CMTMSC) followed by an access of CMTCMD2 or CMTCMD4.

If the EOC interrupt enable (EOCIE) bit is high when the EOCF bit is set, the CMT module will generate an interrupt request. The EOCF bit must be cleared within the interrupt service routine to prevent another interrupt from being generated after exiting the interrupt service routine.

The EOC interrupt is coincident with loading the down-counter with the contents of CMTCMD1:CMTCMD2 and loading the space period register with the contents of CMTCMD3:CMTCMD4. The EOC interrupt provides a means for the user to reload new mark/space values into the modulator data registers. Modulator data register updates will take effect at the end of the current modulation cycle. Note that the down-counter and space period register are updated at the end of every modulation cycle, regardless of interrupt handling and the state of the EOCF flag.

### 8.5.6 Wait Mode Operation

During wait mode the CMT, if enabled, will continue to operate normally. However, there will be no new codes or changes of pattern mode while in wait mode, because the CPU is not operating.

### 8.5.7 Stop Mode Operation

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

In stop1 and stop2 modes, all CMT register data is lost and must be re-initialized upon recovery from these two stop modes.

No CMT module registers are affected in stop3 mode.

Note, because the clocks are halted, the CMT will resume upon exit from stop (only in stop3 mode). Software should ensure stop2 or stop3 mode is not entered while the modulator is in operation to prevent the IRO pin from being asserted while in stop mode. This may require a time-out period from the time that the MCGEN bit is cleared to allow the last modulator cycle to complete.

### 8.5.8 Background Mode Operation

When the microcontroller is in active background mode, the CMT temporarily suspends all counting until the microcontroller returns to normal user mode.

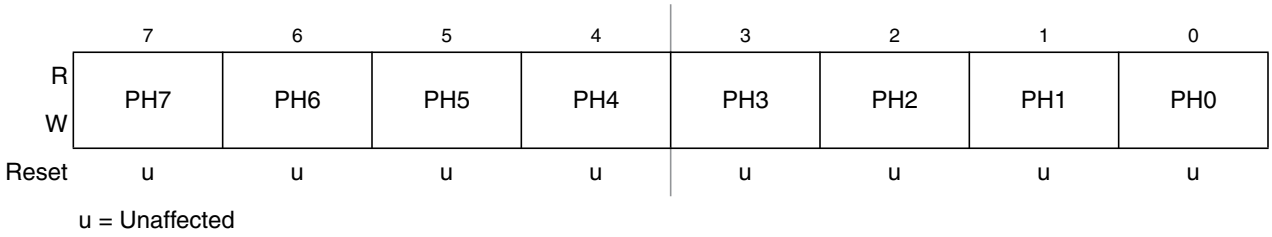
## 8.6 CMT Registers and Control Bits

The following registers control and monitor CMT operation:

- CMT carrier generator data registers (CMTCGH1, CMTCGL1, CMTCGH2, CMTCGL2)
- CMT output control register (CMTOC)
- CMT modulator status and control register (CMTMSC)
- CMT modulator period data registers (CMTCMD1, CMTCMD2, CMTCMD3, CMTCMD4)

### 8.6.1 Carrier Generator Data Registers (CMTCGH1, CMTCGL1, CMTCGH2, and CMTCGL2)

The carrier generator data registers contain the primary and secondary high and low values for generating the carrier output.



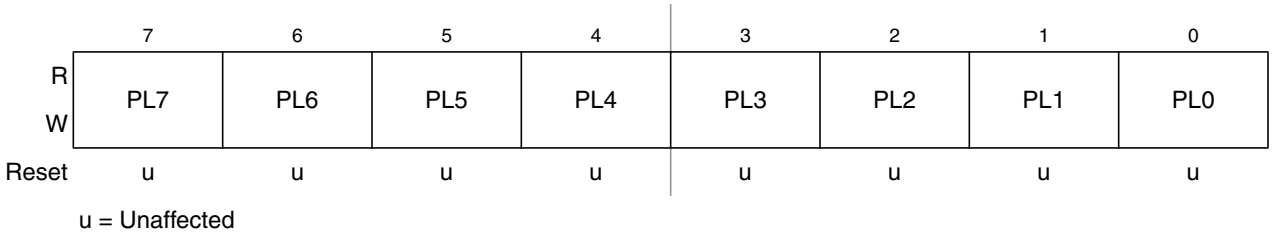
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PH7 | PH6 | PH5 | PH4 | PH3 | PH2 | PH1 | PH0 |
| Reset | u | u | u | u | u | u | u | u |

u = Unaffected

**Figure 8-8. Carrier Generator Data Register High 1(CMTCGH1)**

**Table 8-3. CMTCGH1 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PH[7:0] | **Primary Carrier High Time Data Values** — When selected, these bits contain the number of input clocks required to generate the carrier high and low time periods. When operating in time mode (see Section 8.5.2.1, “Time Mode”), this register pair is always selected. When operating in FSK mode (see Section 8.5.2.3, “FSK Mode”), this register pair and the secondary register pair are alternatively selected under control of the modulator. The primary carrier high and low time values are unaffected out of reset. These bits must be written to nonzero values before the carrier generator is enabled to avoid spurious results. |



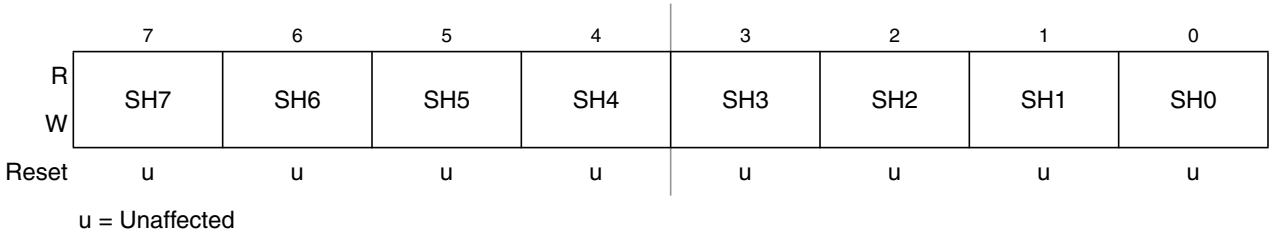
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PL7 | PL6 | PL5 | PL4 | PL3 | PL2 | PL1 | PL0 |
| Reset | u | u | u | u | u | u | u | u |

u = Unaffected

**Figure 8-9. Carrier Generator Data Register Low 1 (CMTCGL1)**

**Table 8-4. CMTCGL1 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PL[7:0] | **Primary Carrier Low Time Data Values** — When selected, these bits contain the number of input clocks required to generate the carrier high and low time periods. When operating in time mode (see Section 8.5.2.1, “Time Mode”), this register pair is always selected. When operating in FSK mode (see Section 8.5.2.3, “FSK Mode”), this register pair and the secondary register pair are alternatively selected under control of the modulator. The primary carrier high and low time values are unaffected out of reset. These bits must be written to nonzero values before the carrier generator is enabled to avoid spurious results. |



| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SH7 | SH6 | SH5 | SH4 | SH3 | SH2 | SH1 | SH0 |
| Reset | u | u | u | u | u | u | u | u |

u = Unaffected

**Figure 8-10. Carrier Generator Data Register High 2 (CMTCGH2)**

**Table 8-5. CMTCGH2 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>SH[7:0] | **Secondary Carrier High Time Data Values** — When selected, these bits contain the number of input clocks required to generate the carrier high and low time periods. When operating in time mode (see Section 8.5.2.1, “Time Mode"), this register pair is never selected. When operating in FSK mode (see Section 8.5.2.3, “FSK Mode"), this register pair and the primary register pair are alternatively selected under control of the modulator. The secondary carrier high and low time values are unaffected out of reset. These bits must be written to nonzero values before the carrier generator is enabled when operating in FSK mode. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SL7 | SL6 | SL5 | SL4 | SL3 | SL2 | SL1 | SL0 |
| Reset | u | u | u | u | u | u | u | u |

u = Unaffected

**Figure 8-11. Carrier Generator Data Register Low 2 (CMTCGL2)**

**Table 8-6. CMTCGL2 Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>SL[7:0] | **Secondary Carrier Low Time Data Values** — When selected, these bits contain the number of input clocks required to generate the carrier high and low time periods. When operating in time mode (see Section 8.5.2.1, “Time Mode"), this register pair is never selected. When operating in FSK mode (see Section 8.5.2.3, “FSK Mode"), this register pair and the primary register pair are alternatively selected under control of the modulator. The secondary carrier high and low time values are unaffected out of reset. These bits must be written to nonzero values before the carrier generator is enabled when operating in FSK mode. |

## 8.6.2 CMT Output Control Register (CMTOC)

This register is used to control the IRO output of the CMT module.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IROL | CMTPOL | IROPEN | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-12. CMT Output Control Register (CMTOC)**

**Table 8-7. CMTOC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IROL | **IRO Latch Control** — Reading IROL reads the state of the IRO latch. Writing IROL changes the state of the IRO pin when the MCGEN bit is clear in the CMTMSC register and the IROPEN bit is set. |
| 6<br>CMTPOL | **CMT Output Polarity** — The CMTPOL bit controls the polarity of the IRO pin output of the CMT.<br>0 IRO pin is active low<br>1 IRO pin is active high |
| 5<br>IROPEN | **IRO Pin Enable** — The IROPEN bit is used to enable and disable the IRO pin. When the pin is enabled, it is an output that drives out either the CMT transmitter output or the state of the IROL bit depending on whether the MCGEN bit in the CMTMSC register is set. Also, the state of the output is either inverted or not depending on the state of the CMTPOL bit. When the pin is disabled, it is in a high impedance state so it doesn’t draw any current. The pin is disabled during reset.<br>0 IRO pin disabled<br>1 IRO pin enabled as output |

### 8.6.3 CMT Modulator Status and Control Register (CMTMSC)

The CMT modulator status and control register (CMTMSC) contains the modulator and carrier generator enable (MCGEN), end of cycle interrupt enable (EOCIE), FSK mode select (FSK), baseband enable (BASE), extended space (EXSPC), prescaler (CMTDIV1:CMTDIV0) bits, and the end of cycle (EOCF) status bit.

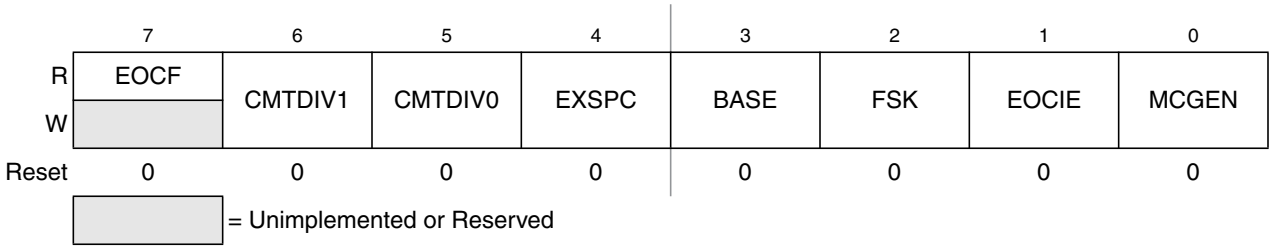



**Figure 8-13. CMT Modulator Status and Control Register (CMTMSC)**

**Table 8-8. CMTMSC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>EOCF | **End of Cycle Status Flag** — The EOCF bit is set when:<br>• The modulator is not currently active and the MCGEN bit is set to begin the initial CMT transmission.<br>• At the end of each modulation cycle while the MCGEN bit is set. This is recognized when a match occurs between the contents of the space period register and the down-counter. At this time, the counter is initialized with the (possibly new) contents of the mark period buffer, CMTCMD1 and CMTCMD2. The space period register is loaded with the (possibly new) contents of the space period buffer, CMTCMD3 and CMTCMD4.<br>This flag is cleared by a read of the CMTMSC register followed by an access of CMTCMD2 or CMTCMD4.<br>In the case where the MCGEN bit is cleared and then set before the end of the modulation cycle, EOCF will not be set when MCGEN is set, but will be set at the end of the current modulation cycle.<br>0 No end of modulation cycle occurrence since flag last cleared<br>1 End of modulator cycle has occurred |
| 6:5<br>CMTDIV[1:0] | **CMT Clock Divide Prescaler** — The CMT clock divide prescaler causes the CMT to be clocked at the BUS CLOCK frequency, or the BUS CLOCK frequency divided by 1, 2, 4, or 8. Because these bits are not double buffered, they should not be changed during a transmission.<br>00 Bus clock ÷ 1<br>01 Bus clock ÷ 2<br>10 Bus clock ÷ 4<br>11 Bus clock ÷ 8 |
| 4<br>EXSPC | **Extended Space Enable** — The EXSPC bit enables extended space operation.<br>0 Extended space disabled<br>1 Extended space enabled |
| 3<br>BASE | **Baseband Enable** — When set, the BASE bit disables the carrier generator and forces the carrier output high for generation of baseband protocols. When BASE is clear, the carrier generator is enabled and the carrier output toggles at the frequency determined by values stored in the carrier data registers. See Section 8.5.2.2, “Baseband Mode." This bit is cleared by reset. This bit is not double buffered and should not be written to during a transmission.<br>0 Baseband mode disabled<br>1 Baseband mode enabled |
| 2<br>FSK | **FSK Mode Select** — The FSK bit enables FSK operation.<br>0 CMT operates in time or baseband mode<br>1 CMT operates in FSK mode |

**Table 8-8. CMTMSC Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>EOCIE | **End of Cycle Interrupt Enable** — A CPU interrupt will be requested when EOCF is set if EOCIE is high.<br>0 CPU interrupt disabled<br>1 CPU interrupt enabled |
| 0<br>MCGEN | **Modulator and Carrier Generator Enable** — Setting MCGEN will initialize the carrier generator and modulator and enable all clocks. After it is enabled, the carrier generator and modulator will function continuously. When MCGEN is cleared, the current modulator cycle will be allowed to expire before all carrier and modulator clocks are disabled (to save power) and the modulator output is forced low. To prevent spurious operation, the user should initialize all data and control registers before enabling the system.<br>0 Modulator and carrier generator disabled<br>1 Modulator and carrier generator enabled |

## 8.6.4 CMT Modulator Data Registers (CMTCMD1, CMTCMD2, CMTCMD3, and CMTCMD4)

The modulator data registers control the mark and space periods of the modulator for all modes. The contents of these registers are transferred to the modulator down counter and space period register upon the completion of a modulation period.

**Table 8-9. Sample Register Summary**

| Name | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| CMTCMD1 | R | MB15 | MB14 | MB13 | MB12 | MB11 | MB10 | MB9 | MB8 |
| | W | | | | | | | | |
| CMTCMD2 | R | MB7 | MB6 | MB5 | MB4 | MB3 | MB2 | MB1 | MB0 |
| | W | | | | | | | | |
| CMTCMD3 | R | SB15 | SB14 | SB13 | SB12 | SB11 | SB10 | SB9 | SB8 |
| | W | | | | | | | | |
| CMTCMD4 | R | SB7 | SB6 | SB5 | SB4 | SB3 | SB2 | SB1 | SB0 |
| | W | | | | | | | | |

# Chapter 9
# Keyboard Interrupt (S08KBIV1)

## 9.1 Introduction

The MC9S08RC/RD/RE/RG has two KBI modules. One has eight keyboard interrupt inputs that share port A pins. The other KBI module has four inputs that are shared on the upper four pins of port C. See the Pins and Connections chapter for more information about the logic and hardware aspects of these pins.

Port A is an 8-bit port that is shared between the KBI1 keyboard interrupt inputs and general-purpose I/O. The eight KBI1PEn control bits in the KBI1PE register allow selection of any combination of port A pins to be assigned as KBI1 inputs. Any pins that are enabled as KBI1 inputs will be forced to act as inputs and the remaining port A pins are available as general-purpose I/O pins controlled by the port A data (PTAD), data direction (PTADD) and pullup enable (PTAPE) registers. The eight PTAPEn control bits in the PTAPE register allow the user to select whether an internal pullup device is enabled on each port A pin that is configured as a port input or a KBI1 input.

KBI1 inputs can be configured for edge-only sensitivity or edge-and-level sensitivity. Bits 3 through 0 of port A are falling-edge/low-level sensitive and bits 7 through 4 can be configured for rising-edge/high-level or for falling-edge/low-level sensitivity.

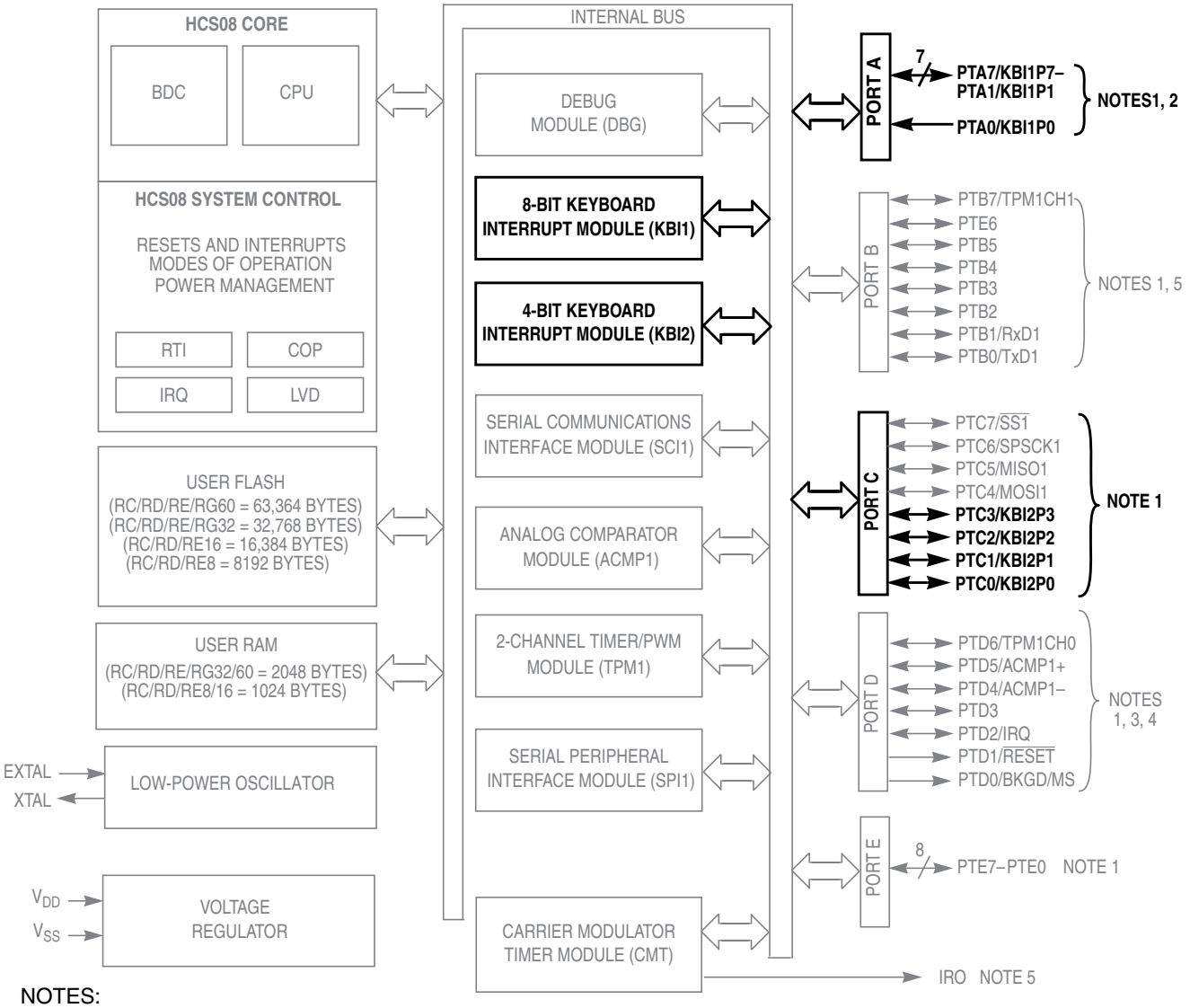
Port C is an 8-bit port with its lower four pins shared between the KBI2 keyboard interrupt inputs and general-purpose I/O. The four KBI2PEn control bits in the KBI2PE register allow selection of any combination of the lower four port C pins to be assigned as KBI2 inputs. Any pins that are enabled as KBI2 inputs will be forced to act as inputs and the remaining port C pins are available as general-purpose I/O pins controlled by the port C data (PTCD), data direction (PTCDD) and pullup enable (PTCPE) registers. The eight PTCPEn control bits in the PTCPE register allow the user to select whether an internal pullup device is enabled on each port C pin that is configured as a port input or a KBI2 input.

Any enabled keyboard interrupt can be used to wake the MCU from wait, standby (stop3), partial power-down (stop2) or power-down modes (stop1). In either stop1 or stop2 mode, an input functions as a falling edge/low-level wakeup, therefore it should be configured to use falling-edge sensing if the MCU will be used in stop1 or stop2 modes.

Either KBI1 or KBI2 can be used to wake the MCU from wait or standby (stop3). Only KBI1 can be used to wake the MCU from partial power down (stop2) or power down (stop1). When using KBI1 to wake up from stop2 or stop1, the pins must be configured to use falling-edge/low-level sensing (KBEDG = 0). The KBF bits for both KBI modules must be cleared before entering stop mode, regardless of whether the interrupt is enabled.

**NOTE**

The voltage measured on the pulled up PTA0 pin will be less than $V_{DD}$. The internal gates connected to this pin are pulled all the way to $V_{DD}$. All other pins with enabled pullup resistors will have an unloaded measurement of $V_{DD}$.




NOTES:

7. Port pins are software configurable with pullup device if input port
8. PTA0 does not have a clamp diode to $V_{DD}$. PTA0 should not be driven above $V_{DD}$. Also, PTA0 does not pullup to $V_{DD}$ when internal pullup is enabled.
9. IRQ pin contains software configurable pullup/pulldown device if IRQ enabled (IRQPE = 1)
10. The RESET pin contains integrated pullup device enabled if reset enabled (RSTPE = 1)
11. High current drive
12. Pins PTA[7:4] contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

**Figure 9-1. MC9S08RC/RD/RE/RG Block Diagram**

## 9.2 KBI Block Diagram

Figure 9-2 shows the block diagram for a KBI module.

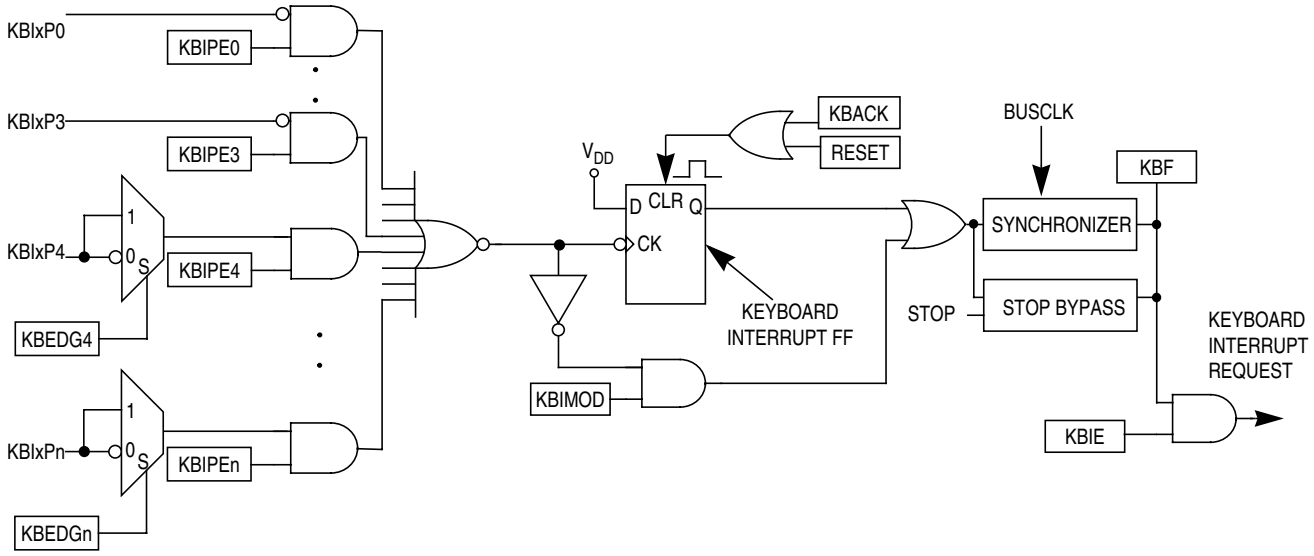



**Figure 9-2. KBI Block Diagram**

The KBI module allows up to eight pins to act as additional interrupt sources. Four of these pins allow falling-edge sensing while the other four can be configured for either rising-edge sensing or falling-edge sensing. The sensing mode for all eight pins can also be modified to detect edges and levels instead of only edges.

## 9.3 Keyboard Interrupt (KBI) Module

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

### 9.3.1 Pin Enables

The KBIPEn control bits in the KBIxPE register allow a user to enable (KBIPEn = 1) any combination of KBI-related port pins to be connected to the KBI module. Pins corresponding to 0s in KBIxPE are general-purpose I/O pins that are not associated with the KBI module.

### 9.3.2 Edge and Level Sensitivity

Synchronous logic is used to detect edges. Prior to detecting an edge, enabled keyboard inputs in a KBI module must be at the deasserted logic level.

A falling edge is detected when an enabled keyboard input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle.

A rising edge is detected when the input signal is seen as a logic 0 during one bus cycle and then a logic 1 during the next cycle.

The KBIMOD control bit can be set to reconfigure the detection logic so that it detects edges and levels. In KBIMOD = 1 mode, the KBF status flag becomes set when an edge is detected (when one or more enabled pins change from the deasserted to the asserted level while all other enabled pins remain at their deasserted levels), but the flag is continuously set (and cannot be cleared) as long as any enabled keyboard input pin remains at the asserted level. When the MCU enters stop mode, the synchronous edge-detection logic is bypassed (because clocks are stopped). In stop mode, KBI inputs act as asynchronous level-sensitive inputs so they can wake the MCU from stop mode.

### 9.3.3 KBI Interrupt Controls

The KBF status flag becomes set (1) when an edge event has been detected on any KBI input pin. If KBIE = 1 in the KBIxSC register, a hardware interrupt will be requested whenever KBF = 1. The KBF flag is cleared by writing a 1 to the keyboard acknowledge (KBACK) bit.

When KBIMOD = 0 (selecting edge-only operation), KBF is always cleared by writing 1 to KBACK. When KBIMOD = 1 (selecting edge-and-level operation), KBF cannot be cleared as long as any keyboard input is at its asserted level.

## 9.4 KBI Registers and Control Bits

This section provides information about all registers and control bits associated with the KBI modules.

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all KBI 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 MCU systems have more than one KBI, so register names include placeholder characters to identify which KBI is being referenced. For example, KBIxSC refers to the KBIx status and control register and KBI2SC is the status and control register for KBI2.

## 9.4.1 KBI x Status and Control Register (KBIxSC)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | KBEDG7 | KBEDG6 | KBEDG5 | KBEDG4 | KBF | 0 | KBIE | KBIMOD |
| W | | | | | | KBACK | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 9-3. KBI x Status and Control Register (KBIxSC)**

**Table 9-1. KBIxSC Field Descriptions**

| Field | Description |
|---|---|
| 7:4<br>KBEDG[7:4] | **Keyboard Edge Select for KBI Port Bits** — Each of these read/write bits selects the polarity of the edges and/or levels that are recognized as trigger events on the corresponding KBI port pin when it is configured as a keyboard interrupt input (KBIPEn = 1). Also see the KBIMOD control bit, which determines whether the pin is sensitive to edges-only or edges and levels.<br>0 Falling edges/low levels.<br>1 Rising edges/high levels. |
| 3<br>KBF | **Keyboard Interrupt Flag** — This read-only status flag is set whenever the selected edge event has been detected on any of the enabled KBI port pins. This flag is cleared by writing a logic 1 to the KBACK control bit. The flag will remain set if KBIMOD = 1 to select edge-and-level operation and any enabled KBI port pin remains at the asserted level.<br>0 No KBI interrupt pending.<br>1 KBI interrupt pending.<br>KBF can be used as a software pollable flag (KBIE = 0) or it can generate a hardware interrupt request to the CPU (KBIE = 1). KBF must be cleared before entering stop mode. |
| 2<br>KBACK | **Keyboard Interrupt Acknowledge** — This write-only bit (reads always return 0) is used to clear the KBF status flag by writing a logic 1 to KBACK. When KBIMOD = 1 to select edge-and-level operation and any enabled KBI port pin remains at the asserted level, KBF is being continuously set so writing 1 to KBACK does not clear the KBF flag. |
| 1<br>KBIE | **Keyboard Interrupt Enable** — This read/write control bit determines whether hardware interrupts are generated when the KBF status flag equals 1. When KBIE = 0, no hardware interrupts are generated, but KBF can still be used for software polling.<br>0 KBF does not generate hardware interrupts (use polling).<br>1 KBI hardware interrupt requested when KBF = 1. |
| 0<br>KBIMOD | **Keyboard Detection Mode** — This read/write control bit selects either edge-only detection or edge-and-level detection. KBI port bits 3 through 0 can detect falling edges-only or falling edges and low levels.<br>KBI port bits 7 through 4 can be configured to detect either:<br>• Rising edges-only or rising edges and high levels (KBEDGn = 1)<br>• Falling edges-only or falling edges and low levels (KBEDGn = 0)<br>0 Edge-only detection.<br>1 Edge-and-level detection. |

## 9.4.2 KBI x Pin Enable Register (KBIxPE)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | KBIPE7 | KBIPE6 | KBIPE5 | KBIPE4 | KBIPE3 | KBIPE2 | KBIPE1 | KBIPE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-4. KBI x Pin Enable Register (KBIxPE)**

**Table 9-2. KBIxPE Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>KBIPE[7:0] | **Keyboard Pin Enable for KBI Port Bits** — Each of these read/write bits selects whether the associated KBI port pin is enabled as a keyboard interrupt input or functions as a general-purpose I/O pin.<br>0 Bit n of KBI port is a general-purpose I/O pin not associated with the KBI.<br>1 Bit n of KBI port enabled as a keyboard interrupt input |

# Chapter 10
# Timer/PWM Module (S08TPMV1)

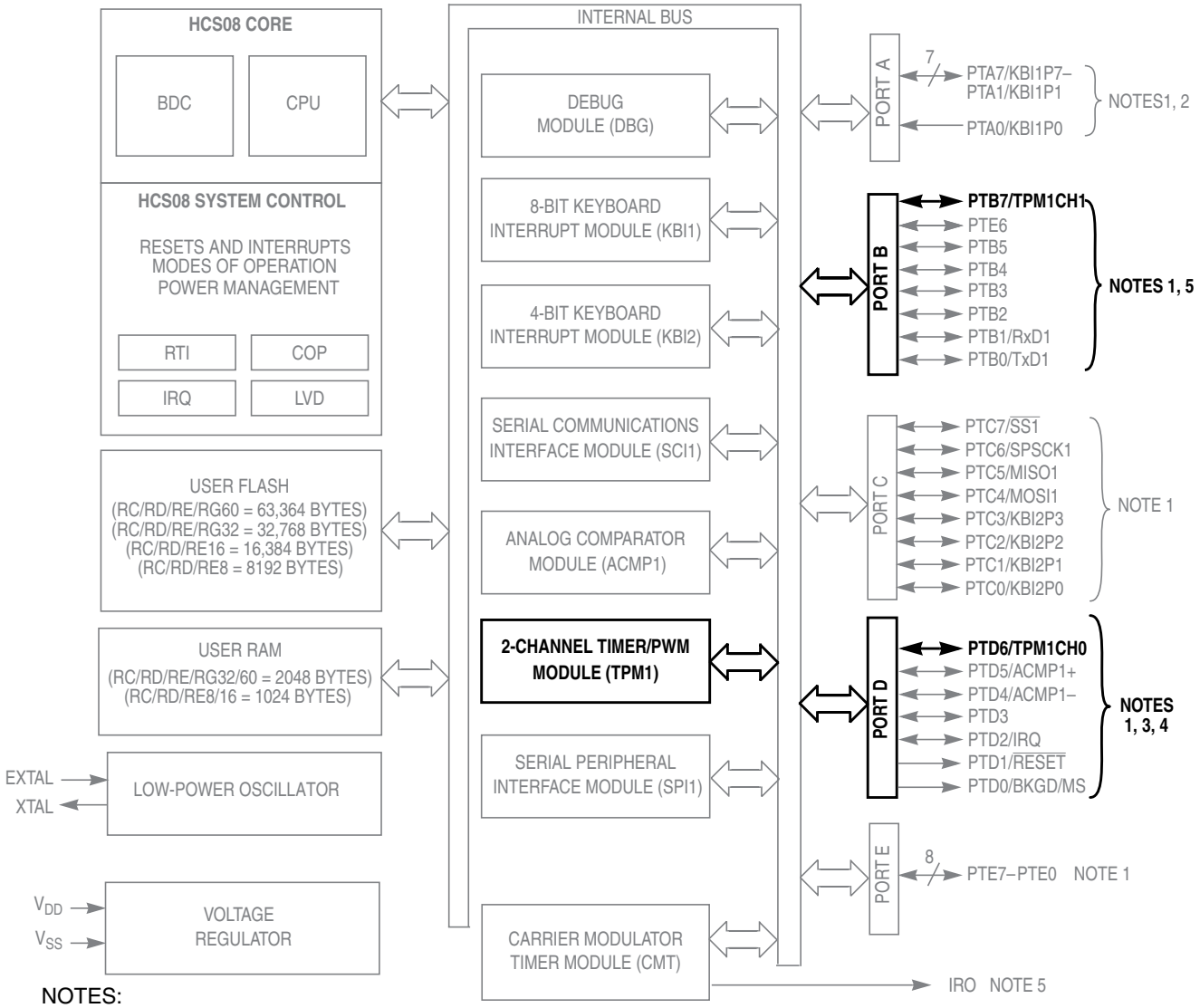
## 10.1 Introduction

The MC9S08RC/RD/RE/RG includes a timer/PWM (TPM) module that supports traditional input capture, output compare, or buffered edge-aligned pulse-width modulation (PWM) on each channel. A control bit in the TPM configures both channels in the timer to operate as center-aligned PWM functions. Timing functions in the TPM are based on a 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. The MC9S08RC/RD/RE/RG devices do not have a separate fixed internal clock source (XCLK). If the XCLK source is selected using the CLKSA and CLKSB control bits (see Table 10-2), the TPM will use the BUSCLK.

## 10.2 Features

Timer system features include:

- Two separate 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
- The TPM may be configured for buffered, center-aligned pulse-width modulation (CPWM) on both channels
- Clock source to prescaler for the TPM is selectable between the bus clock or an external pin:
  - Prescale taps for divide by 1, 2, 4, 8, 16, 32, 64, or 128
  - External clock input shared with TPM1CH0 timer channel pin
- 16-bit modulus register to control counter range
- Timer system enable
- One interrupt per channel plus terminal count interrupt




NOTES:

13. Port pins are software configurable with pullup device if input port
14. PTA0 does not have a clamp diode to VDD. PTA0 should not be driven above VDD. Also, PTA0 does not pullup to VDD when interna pullup is enabled.
15. IRQ pin contains software configurable pullup/pulldown device if IRQ enabled (IRQPE = 1)
16. The RESET pin contains integrated pullup device enabled if reset enabled (RSTPE = 1)
17. High current drive
18. Pins PTA[7:4] contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

**Figure 10-1. MC9S08RC/RD/RE/RG Block Diagram Highlighting TPM Block and Pins**

## 10.3 TPM Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPM1CHn where n is the channel number (for example, 0–4). The TPM shares its I/O pins with general-purpose I/O port pins (refer to the Pins and Connections chapter for more information). Figure 10-2 shows the structure of a TPM. Some MCUs include more than one TPM, with various numbers of channels.

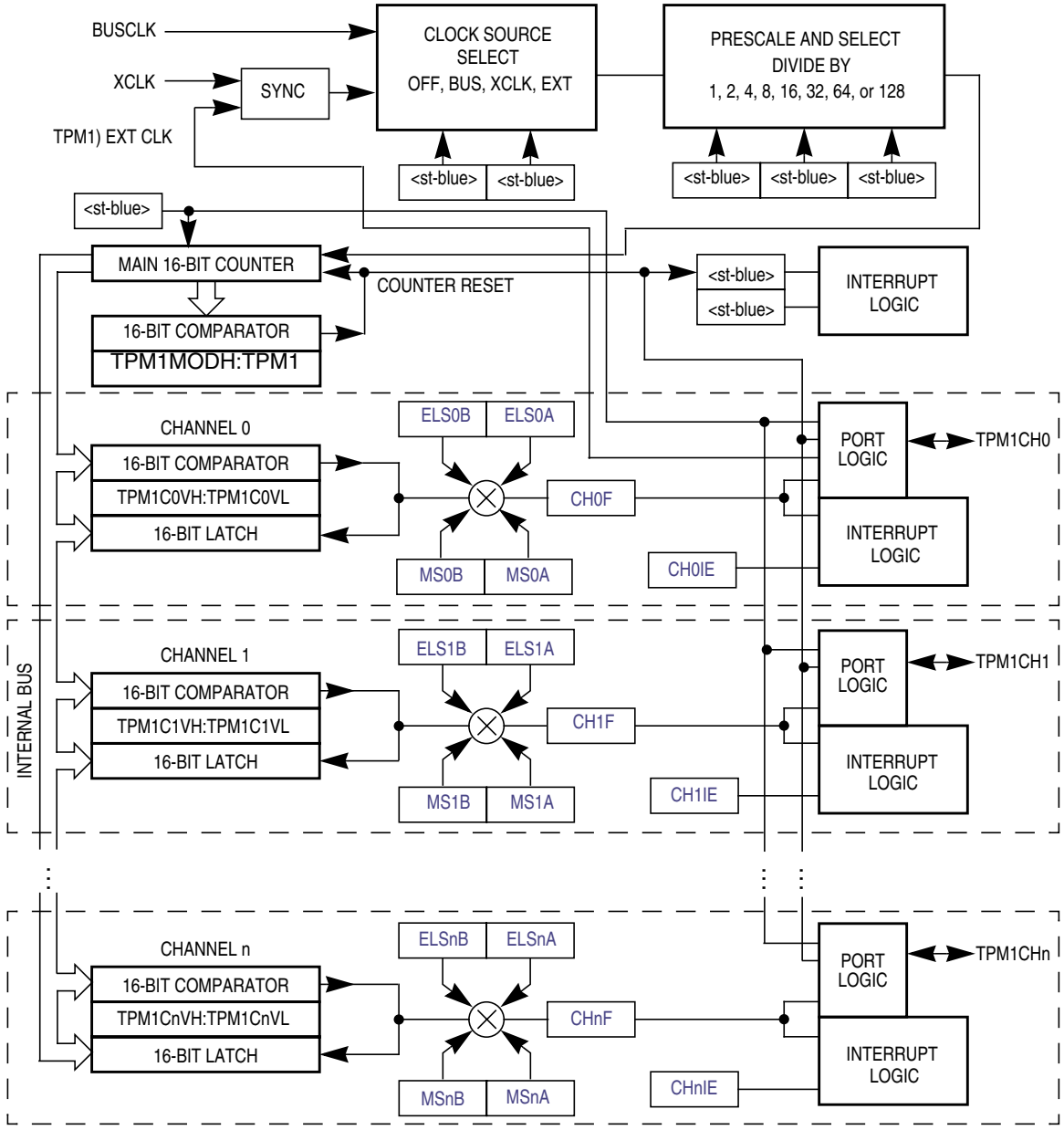



**Figure 10-2. TPM Block Diagram**

The central component of the TPM is the 16-bit counter that can operate as a free-running counter, a modulo counter, or an up-/down-counter when the TPM is configured for center-aligned PWM. 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, TPM1MODH:TPM1MODL, control the modulo value of the counter. (The values $0000 or $FFFF effectively make the counter free running.) Software can read the counter value at any time without affecting the counting sequence. Any write to either byte of the TPM1CNT counter resets the counter regardless of the data value written.

All TPM channels are programmable independently as input capture, output compare, or buffered edge-aligned PWM channels.

## 10.4 Pin Descriptions

Table 10-2 shows the MCU pins related to the TPM module. When TPM1CH0 is used as an external clock input, the associated TPM channel 0 can not use the pin. (Channel 0 can still be used in output compare mode as a software timer.) When any of the pins associated with the timer is configured as a timer input, a passive pullup can be enabled. After reset, the TPM modules are disabled and all pins default to general-purpose inputs with the passive pullups disabled.

### 10.4.1 External TPM Clock Sources

When control bits CLKSB:CLKSA in the timer status and control register are set to 1:1, the prescaler and consequently the 16-bit counter for TPM1 are driven by an external clock source connected to the TPM1CH0 pin. A synchronizer is needed between the external clock and the rest of the TPM. This synchronizer is clocked by the bus clock so the frequency of the external source must be less than one-half the frequency of the bus rate clock. The upper frequency limit for this external clock source is specified to be one-fourth the bus frequency to conservatively accommodate duty cycle and phase-locked loop (PLL) or frequency-locked loop (FLL) frequency jitter effects.

When the TPM is using the channel 0 pin for an external clock, the corresponding ELS0B:ELS0A control bits should be set to 0:0 so channel 0 is not trying to use the same pin.

### 10.4.2 TPM1CHn — TPM1 Channel n I/O Pins

Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the configuration of the channel. In some cases, no pin function is needed so the pin reverts to being controlled by general-purpose I/O controls. When a timer has control of a port pin, the port data and data direction registers do not affect the related pin(s). See the Pins and Connections chapter for additional information about shared pin functions.

## 10.5 Functional Description

All TPM functions are associated with a main 16-bit counter that allows flexible selection of the clock source and prescale divisor. A 16-bit modulo register also is associated with the main 16-bit counter in the TPM. Each TPM channel is optionally associated with an MCU pin and a maskable interrupt function.

The TPM has center-aligned PWM capabilities controlled by the CPWMS control bit in TPM1SC. When CPWMS is set to 1, timer counter TPM1CNT changes to an up-/down-counter and all channels in the associated TPM act as center-aligned PWM channels. When CPWMS = 0, each channel can

independently be configured to operate in input capture, output compare, or buffered edge-aligned PWM mode.

The following sections describe the main 16-bit 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 on the operating mode, these topics are covered in the associated mode sections.

### 10.5.1 Counter

All timer functions are based on the main 16-bit counter (TPM1CNTH:TPM1CNTL). This section discusses selection of the clock source, up-counting vs. up-/down-counting, end-of-count overflow, and manual counter reset.

After any MCU reset, CLKSB:CLKSA = 0:0 so no clock source is selected and the TPM is inactive. Normally, CLKSB:CLKSA would be set to 0:1 so the bus clock drives the timer counter. The clock source for the TPM can be selected to be off, the bus clock (BUSCLK), the fixed system clock (XCLK), or an external input through the TPM1CH0 pin. The maximum frequency allowed for the external clock option is one-fourth the bus rate. Refer to Section 10.7.1, “Timer Status and Control Register (TPM1SC),” and Table 10-2 for more information about clock source selection.

When the microcontroller is in active background mode, the TPM temporarily suspends all counting until the microcontroller returns to normal user operating mode. During stop mode, all TPM clocks are stopped; therefore, the TPM is effectively disabled until clocks resume. During wait mode, the TPM continues to operate normally.

The main 16-bit 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 main 16-bit counter counts from $0000 through its terminal count and then continues with $0000. The terminal count is $FFFF or a modulus value in TPM1MODH:TPM1MODL.

When center-aligned PWM operation is specified, the counter counts upward from $0000 through its terminal count and then counts downward to $0000 where it returns to up-counting. Both $0000 and the terminal count value (value in TPM1MODH:TPM1MODL) are normal length counts (one timer clock period long).

An interrupt flag and enable are associated with the main 16-bit counter. The timer overflow 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 automatically generated whenever the TOF flag is 1.

The conditions that cause TOF to become set depend on the counting mode (up or up/down). In up-counting mode, the main 16-bit counter counts from $0000 through $FFFF and overflows to $0000 on the next counting clock. TOF becomes set at the transition from $FFFF to $0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to $0000. When the main 16-bit counter is operating in up-/down-counting mode, the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The $0000 count value corresponds to the center of a period.)

Because the HCS08 MCU is an 8-bit architecture, a coherency mechanism is built into the timer counter for read operations. Whenever either byte of the counter is read (TPM1CNTH or TPM1CNTL), both bytes are captured into a buffer so when the other byte is read, the value will represent the other byte of the count at the time the first byte was read. The counter continues to count normally, but no new value can be read from either byte until both bytes of the old count have been read.

The main timer counter can be reset manually at any time by writing any value to either byte of the timer count TPM1CNTH or TPM1CNTL. Resetting the counter in this manner also resets the coherency mechanism in case only one byte of the counter was read before resetting the count.

## 10.5.2 Channel Mode Selection

Provided CPWMS = 0 (center-aligned PWM operation is not specified), 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 buffered edge-aligned PWM.

### 10.5.2.1 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 (TPM1CnVH:TPM1CnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

When either byte of the 16-bit capture register is read, both bytes are latched into a buffer to support coherent 16-bit accesses regardless of order. The coherency sequence can be manually reset by writing to the channel status/control register (TPM1CnSC).

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

### 10.5.2.2 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 value registers only after both 8-bit bytes of a 16-bit register have been written. This coherency sequence can be manually reset by writing to the channel status/control register (TPM1CnSC).

An output compare event sets a flag bit (CHnF) that can optionally generate a CPU interrupt request.

### 10.5.2.3 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 setting in the modulus register (TPM1MODH:TPM1MODL). The duty cycle is determined by the setting in the timer channel value

register (TPM1CnVH:TPM1CnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. Duty cycle cases of 0 percent and 100 percent are possible.

As Figure 10-3 shows, the output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal. 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.

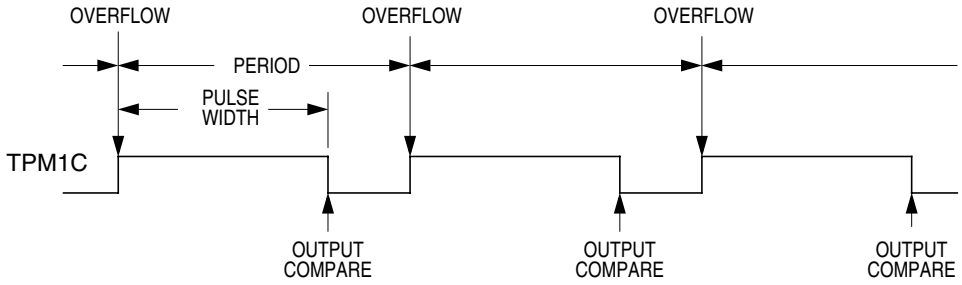



**Figure 10-3. PWM Period and Pulse Width (ELSnA = 0)**

When the channel value register is set to $0000, the duty cycle is 0 percent. By setting the timer channel value register (TPM1CnVH:TPM1CnVL) to a value greater than the modulus setting, 100 percent duty cycle can be achieved. This implies that the modulus setting must be less than $FFFF to get 100 percent duty cycle.

Because the HCS08 is a family of 8-bit MCUs, the settings in the timer channel registers are buffered to ensure coherent 16-bit updates and to avoid unexpected PWM pulse widths. Writes to either register, TPM1CnVH or TPM1CnVL, write to buffer registers. In edge-PWM mode, values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the value in the TPM1CNTH:TPM1CNTL counter is $0000. (The new duty cycle does not take effect until the next full period.)

## 10.5.3 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 TPM1CnVH:TPM1CnVL determines the pulse width (duty cycle) of the PWM signal and the period is determined by the value in TPM1MODH:TPM1MODL. TPM1MODH:TPM1MODL should be kept in the range of $0001 to $7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

$$\text{pulse width} = 2 \times (\text{TPM1CnVH:TPM1CnVL}) \qquad \textit{Eqn. 10-1}$$

$$\text{period} = 2 \times (\text{TPM1MODH:TPM1MODL}); \quad \text{for TPM1MODH:TPM1MODL} = \$0001\text{–}\$7FFF \qquad \textit{Eqn. 10-2}$$

If the channel value register TPM1CnVH:TPM1CnVL is zero or negative (bit 15 set), the duty cycle will be 0 percent. If TPM1CnVH:TPM1CnVL is a positive value (bit 15 clear) and is greater than the (nonzero) modulus setting, the duty cycle will be 100 percent because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is $0001 through $7FFE ($7FFF if

generation of 100 percent duty cycle is not necessary). This is not a significant limitation because the resulting period is much longer than required for normal applications.

TPM1MODH:TPM1MODL = $0000 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 $0000 through $FFFF, but when CPWMS = 1 the counter needs a valid match to the modulus register somewhere other than at $0000 in order to change directions from up-counting to down-counting.

Figure 10-4 shows the output compare value in the TPM channel registers (multiplied by 2), which determines the pulse width (duty cycle) of the CPWM signal. If ELSnA = 0, the compare match while counting up forces the CPWM output signal low and a compare match while counting down forces the output high. The counter counts up until it reaches the modulo setting in TPM1MODH:TPM1MODL, then counts down until it reaches zero. This sets the period equal to two times TPM1MODH:TPM1MODL.

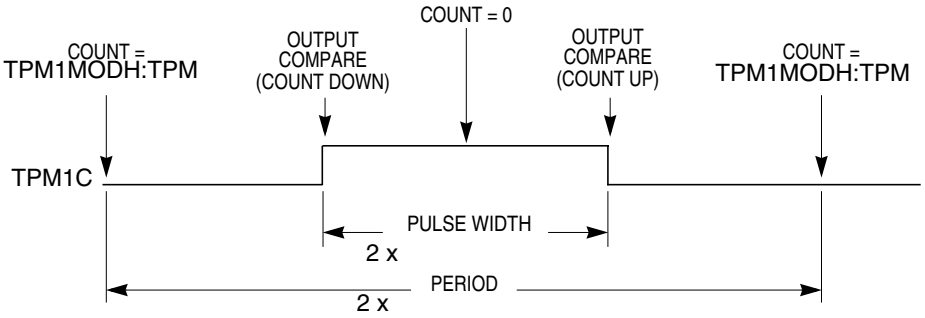



**Figure 10-4. CPWM Period and Pulse Width (ELSnA = 0)**

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.

Because the HCS08 is a family of 8-bit MCUs, 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, TPM1MODH, TPM1MODL, TPM1CnVH, and TPM1CnVL, actually write to buffer registers. Values are transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the timer counter overflows (reverses direction from up-counting to down-counting at the end of the terminal count in the modulus register). This TPM1CNT overflow requirement only applies to PWM channels, not output compares.

Optionally, when TPM1CNTH:TPM1CNTL = TPM1MODH:TPM1MODL, the TPM can generate a TOF interrupt at the end of this count. The user can choose to reload any number of the PWM buffers, and they will all update simultaneously at the start of a new period.

Writing to TPM1SC cancels any values written to TPM1MODH and/or TPM1MODL and resets the coherency mechanism for the modulo registers. Writing to TPM1CnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPM1CnVH:TPM1CnVL.

## 10.6 TPM Interrupts

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on the mode of operation for each channel. 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. See the Resets, Interrupts, and System Configuration chapter for absolute interrupt vector addresses, priority, and local interrupt mask control bits.

For each interrupt source in the TPM, a flag bit is set on 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 verify 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 be generated whenever the associated interrupt flag equals 1. It is the responsibility of user software to perform a sequence of steps to clear the interrupt flag before returning from the interrupt service routine.

### 10.6.1 Clearing Timer Interrupt Flags

TPM interrupt flags are cleared by a 2-step process that includes a read of the flag bit while it is set (1) followed by a write of 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.

### 10.6.2 Timer Overflow Interrupt Description

The conditions that cause TOF to become set depend on the counting mode (up or up/down). In up-counting mode, the 16-bit timer counter counts from $0000 through $FFFF and overflows to $0000 on the next counting clock. TOF becomes set at the transition from $FFFF to $0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to $0000. When the counter is operating in up-/down-counting mode, the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The $0000 count value corresponds to the center of a period.)

### 10.6.3 Channel Event Interrupt Description

The meaning of channel interrupts depends on the current mode of the channel (input capture, output compare, edge-aligned PWM, or center-aligned PWM).

When a channel is configured as an input capture channel, the ELSnB:ELSnA control bits select rising edges, falling edges, any edge, or no edge (off) as the edge that triggers an input capture event. When the selected edge is detected, the interrupt flag is set. The flag is cleared by the 2-step sequence described in Section 10.6.1, “Clearing Timer Interrupt Flags.”

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 2-step sequence described in Section 10.6.1, “Clearing Timer Interrupt Flags.”

### 10.6.4 PWM End-of-Duty-Cycle Events

For channels that are configured for PWM operation, there are two possibilities:

- When the channel is configured for edge-aligned PWM, the channel flag is set when the timer counter matches the channel value register that marks the end of the active duty cycle period.
- When the channel is configured for center-aligned PWM, the timer count matches the channel value register twice during each PWM cycle. In this CPWM case, the channel flag is set at the start and at the end of the active duty cycle, which are the times when the timer counter matches the channel value register.

The flag is cleared by the 2-step sequence described in Section 10.6.1, “Clearing Timer Interrupt Flags.”

## 10.7 TPM Registers and Control Bits

The TPM includes:

- An 8-bit status and control register (TPM1SC)
- A 16-bit counter (TPM1CNTH:TPM1CNTL)
- A 16-bit modulo register (TPM1MODH:TPM1MODL)

Each timer channel has:

- An 8-bit status and control register (TPM1CnSC)
- A 16-bit channel value register (TPM1CnVH:TPM1CnVL)

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all TPM 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.

### 10.7.1 Timer Status and Control Register (TPM1SC)

TPM1SC contains the overflow status flag and control bits that are used to configure the interrupt enable, TPM configuration, clock source, and prescale divisor. These controls relate to all channels within this timer module.

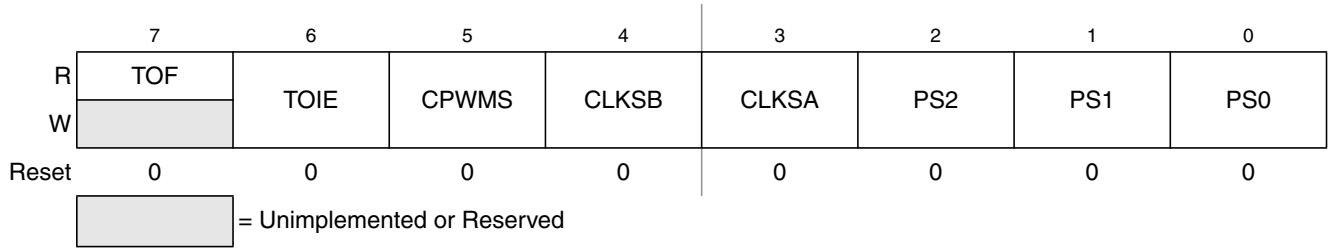



**Figure 10-5. Timer Status and Control Register (TPM1SC)**

**Table 10-1. TPM1SC Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOF | **Timer Overflow Flag** — This flag is set when the TPM counter changes to $0000 after reaching the modulo value programmed in the TPM counter modulo registers. When the TPM is configured for CPWM, TOF is set after the counter has reached the value in the modulo register, at the transition to the next lower count value. Clear TOF by reading the TPM status and control register when TOF is set and then writing a 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. Reset clears TOF. Writing a 1 to TOF has no effect.<br>0 TPM counter has not reached modulo value or overflow<br>1 TPM counter has overflowed |
| 6<br>TOIE | **Timer Overflow Interrupt Enable** — This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals 1. Reset clears TOIE.<br>0 TOF interrupts inhibited (use software polling)<br>1 TOF interrupts enabled |
| 5<br>CPWMS | **Center-Aligned PWM Select** — This read/write bit selects CPWM operating mode. Reset clears this bit so 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.<br>0 All TPM1 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<br>1 All TPM1 channels operate in center-aligned PWM mode |
| 4:3<br>CLKS[B:A] | **Clock Source Select** — As shown in Table 10-2, this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The external source and the XCLK are synchronized to the bus clock by an on-chip synchronization circuit. |
| 2:0<br>PS[2:0] | **Prescale Divisor Select** — This 3-bit field selects one of eight divisors for the TPM clock input as shown in Table 10-3. This prescaler is located after any clock source synchronization or clock source selection, so it affects whatever clock source is selected to drive the TPM system. |

**Table 10-2. TPM Clock Source Selection**

| CLKSB:CLKSA | TPM Clock Source to Prescaler Input |
|---|---|
| 0:0 | No clock selected (TPM disabled) |
| 0:1 | Bus rate clock (BUSCLK) |
| 1:0 | Fixed system clock (XCLK) |
| 1:1 | External source (TPM1 Ext Clk)[1,2] |

1. The maximum frequency that is allowed as an external clock is one-fourth of the bus frequency.
2. When the TPM1CH0 pin is selected as the TPM clock source, the corresponding ELS0B:ELS0A control bits should be set to 0:0 so channel 0 does not try to use the same pin for a conflicting function.

**Table 10-3. Prescale Divisor Selection**

| PS2:PS1:PS0 | TPM Clock Source Divided-By |
|---|---|
| 0:0:0 | 1 |
| 0:0:1 | 2 |
| 0:1:0 | 4 |
| 0:1:1 | 8 |
| 1:0:0 | 16 |
| 1:0:1 | 32 |
| 1:1:0 | 64 |
| 1:1:1 | 128 |

## 10.7.2 Timer Counter Registers (TPM1CNTH:TPM1CNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPM1CNTH or TPM1CNTL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This allows coherent 16-bit reads in either order. The coherency mechanism is automatically restarted by an MCU reset, a write of any value to TPM1CNTH or TPM1CNTL, or any write to the timer status/control register (TPM1SC).

Reset clears the TPM counter registers.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| W | Any write to TPM1CNTH clears the 16-bit counter. | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-6. Timer Counter Register High (TPM1CNTH)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| W | Any write to TPM1CNTL clears the 16-bit counter. | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-7. Timer Counter Register Low (TPM1CNTL)**

When background mode is active, the timer counter and the coherency mechanism are frozen such that the buffer latches remain in the state they were in when the background mode became active even if one or both bytes of the counter are read while background mode is active.

### 10.7.3 Timer Counter Modulo Registers (TPM1MODH:TPM1MODL)

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 $0000 at the next clock (CPWMS = 0) or starts counting down (CPWMS = 1), and the overflow flag (TOF) becomes set. Writing to TPM1MODH or TPM1MODL inhibits the TOF bit and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to $0000, which results in a free-running timer counter (modulo disabled).

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

**Figure 10-8. Timer Counter Modulo Register High (TPM1MODH)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-9. Timer Counter Modulo Register Low (TPM1MODL)**

It is good practice to wait for an overflow interrupt so both bytes of the modulo register can be written well before a new overflow. An alternative approach is to reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

## 10.7.4 Timer Channel n Status and Control Register (TPM1CnSC)

TPM1CnSC contains the channel interrupt status flag and control bits that are used to configure the interrupt enable, channel configuration, and pin function.

**Figure 10-10. Timer Channel n Status and Control Register (TPM1CnSC)**

**Table 10-4. TPM1CnSC Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CHnF | **Channel n Flag** — When channel n is configured for input capture, this flag bit is set when an active edge occurs on the channel n pin. When channel n is an output compare or edge-aligned PWM channel, CHnF is set when the value in the TPM counter registers matches the value in the TPM channel n value registers. This flag is seldom used with center-aligned PWMs because it is set every time the counter matches the channel value register, which correspond to both edges of the active duty cycle period.<br>A corresponding interrupt is requested when CHnF is set and interrupts are enabled (CHnIE = 1). Clear CHnF by reading TPM1CnSC while CHnF is set and then writing a 0 to CHnF. If another interrupt request occurs before the clearing sequence is complete, the sequence is reset so CHnF would remain set after the clear sequence was completed for the earlier CHnF. This is done so a CHnF interrupt request cannot be lost by clearing a previous CHnF. Reset clears CHnF. Writing a 1 to CHnF has no effect.<br>0 No input capture or output compare event occurred on channel n<br>1 Input capture or output compare event occurred on channel n |
| 6<br>CHnIE | **Channel n Interrupt Enable** — This read/write bit enables interrupts from channel n. Reset clears CHnIE.<br>0 Channel n interrupt requests disabled (use software polling)<br>1 Channel n interrupt requests enabled |
| 5<br>MSnB | **Mode Select B for TPM Channel n** — When CPWMS = 0, MSnB = 1 configures TPM channel n for edge-aligned PWM mode. For a summary of channel mode and setup controls, refer to Table 10-5. |
| 4<br>MSnA | **Mode Select A for TPM Channel n** — When CPWMS = 0 and MSnB = 0, MSnA configures TPM channel n for input capture mode or output compare mode. Refer to Table 10-5 for a summary of channel mode and setup controls. |
| 3:2<br>ELSn[B:A] | **Edge/Level Select Bits** — Depending on the operating mode for the timer channel as set by CPWMS:MSnB:MSnA and shown in Table 10-5, 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.<br>Setting ELSnB:ELSnA to 0:0 configures the related timer pin as a general-purpose I/O pin unrelated to any timer channel 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. This is also the setting required for channel 0 when the TPM1CH0 pin is used as an external clock input. |

**Table 10-5. Mode, Edge, and Level Selection**

| CPWMS | MSnB:MSnA | ELSnB:ELSnA | Mode | Configuration |
|---|---|---|---|---|
| X | XX | 00 | Pin not used for TPM channel; use as an external clock for the TPM or revert to general-purpose I/O | |
| 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) |

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. Typically, a program would clear status flags after changing channel configuration bits and before enabling channel interrupts or using the status flags to avoid any unexpected behavior.

## 10.7.5 Timer Channel Value Registers (TPM1CnVH:TPM1CnVL)

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 value registers are cleared by reset.

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

**Figure 10-11. Timer Channel Value Register High (TPM1CnVH)**

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 10-12. Timer Channel Value Register Low (TPM1CnVL)**

In input capture mode, reading either byte (TPM1CnVH or TPM1CnVL) latches the contents of both bytes into a buffer where they remain latched until the other byte is read. This latching mechanism also resets (becomes unlatched) when the TPM1CnSC register is written.

In output compare or PWM modes, writing to either byte (TPM1CnVH or TPM1CnVL) latches the value into a buffer. When both bytes have been written, they are transferred as a coherent 16-bit value into the timer channel value registers. This latching mechanism may be manually reset by writing to the TPM1CnSC register.

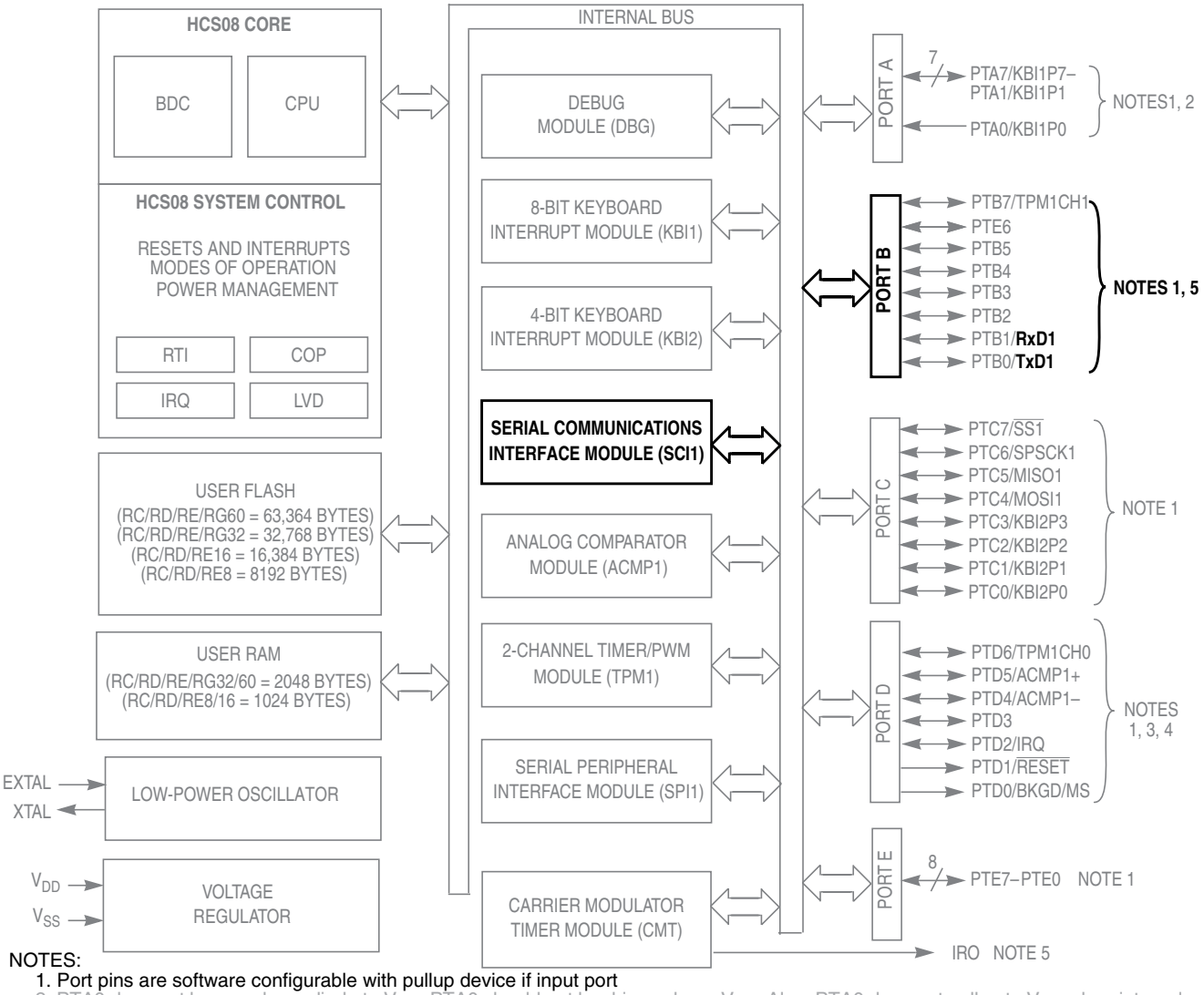
This latching mechanism allows coherent 16-bit writes in either order, which is friendly to various compiler implementations.

# Chapter 11
# Serial Communications Interface (S08SCIV1)

## 11.1 Introduction

The MC9S08RDxx, MC9S08RExx, and MC9S08RGxx devices include a serial communications interface (SCI) module, which is sometimes called a universal asynchronous receiver/transmitters (UART). The SCI module shares pins with PTB0 and PTB1 port pins. When the SCI is enabled, the pins are controlled by the SCI module.

Figure 11-1 is a device-level block diagram with the SCI highlighted.




NOTES:
1. Port pins are software configurable with pullup device if input port
2. PTA0 does not have a clamp diode to $V_{DD}$. PTA0 should not be driven above $V_{DD}$. Also, PTA0 does not pullup to $V_{DD}$ when internal pullup is enabled.
3. IRQ pin contains software configurable pullup/pulldown device if IRQ enabled (IRQPE = 1)
4. The RESET pin contains integrated pullup device enabled if reset enabled (RSTPE = 1)
5. High current drive
6. Pins PTA[7:4] contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

**Figure 11-1. MC9S08RC/RD/RE/RG Block Diagram Highlighting SCI Block and Pins**

# Chapter 12
# Serial Communications Interface (S08SCIV1)

## 12.1 Introduction

### 12.1.1 Features

Features of SCI module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Double-buffered transmitter and receiver with separate enables
- Programmable baud rates (13-bit modulo divider)
- Interrupt-driven or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
- Hardware parity generation and checking
- Programmable 8-bit or 9-bit character length
- Receiver wakeup by idle-line or address-mark

### 12.1.2 Modes of Operation

See Section 12.3, “Functional Description,” for a detailed description of SCI operation in the different modes.

- 8- and 9- bit data modes
- Stop modes — SCI is halted during all stop modes
- Loop modes

### 12.1.3 Block Diagram

Figure 12-1 shows the transmitter portion of the SCI. (Figure 12-2 shows the receiver portion of the SCI.)

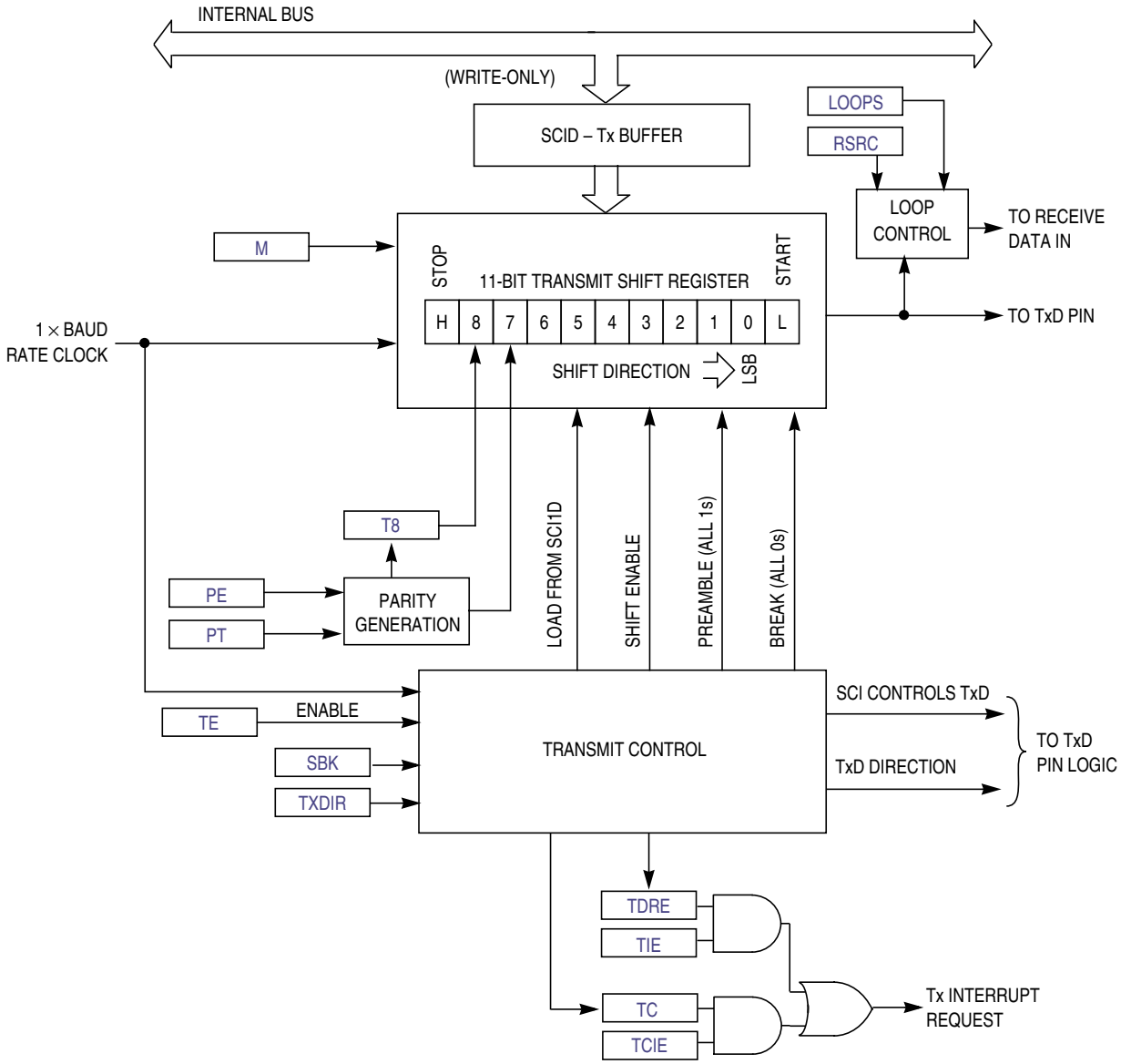



**Figure 12-1. SCI Transmitter Block Diagram**

Figure 12-2 shows the receiver portion of the SCI.

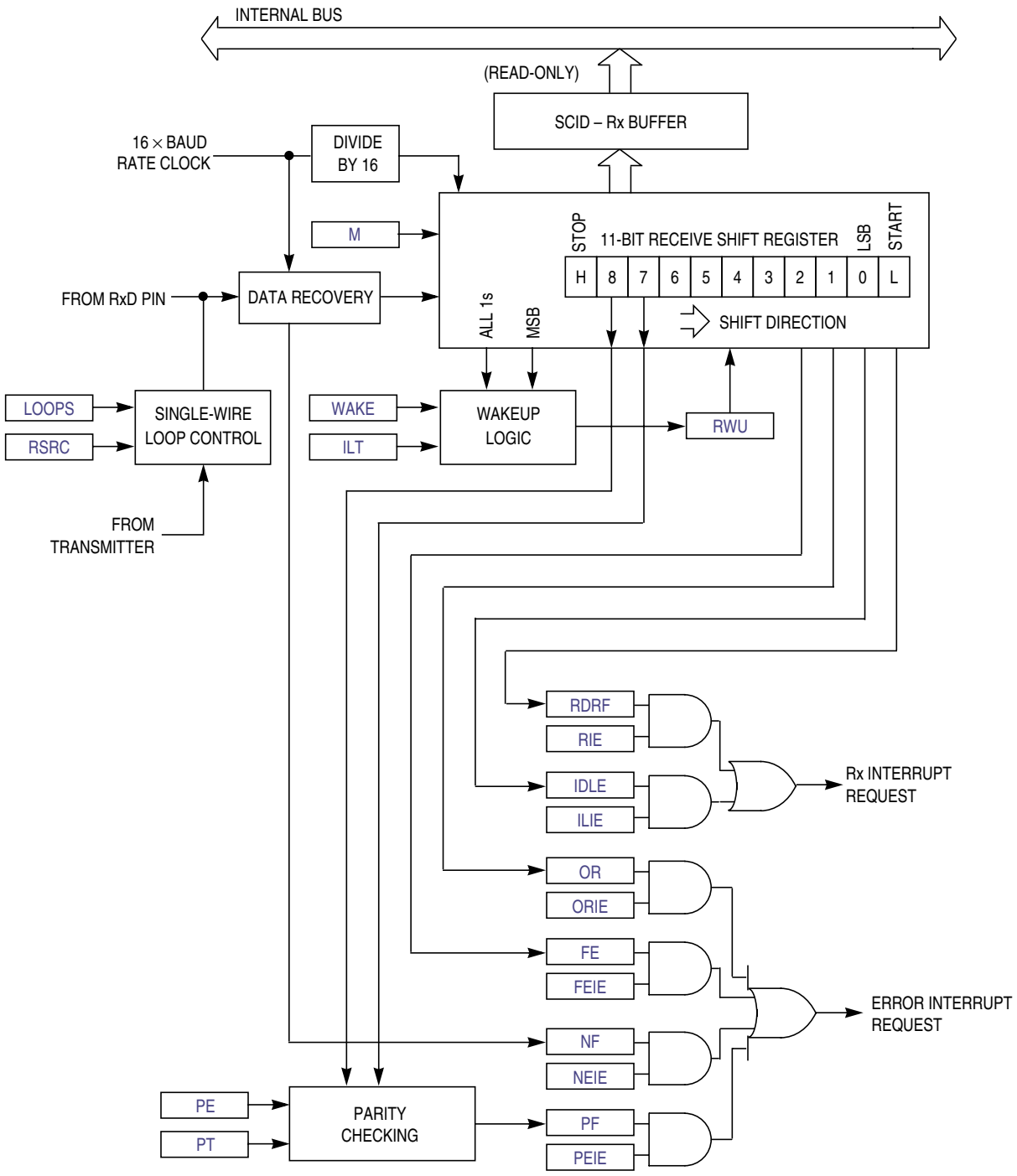



**Figure 12-2. SCI Receiver Block Diagram**

## 12.2 Register Definition

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

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

### 12.2.1 SCI Baud Rate Registers (SCI1BDH, SCI1BHL)

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 SCI1BDH to buffer the high half of the new value and then write to SCI1BDL. The working value in SCI1BDH does not change until SCI1BDL is written.

SCI1BDL 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 SCI1C2 are written to 1).

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 12-3. SCI Baud Rate Register (SCI1BDH)**

**Table 12-1. SCI1BDH Register Field Descriptions**

| Field | Description |
|---|---|
| 4:0<br>SBR[12:8] | **Baud Rate Modulo Divisor** — These 13 bits are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 12-2. |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

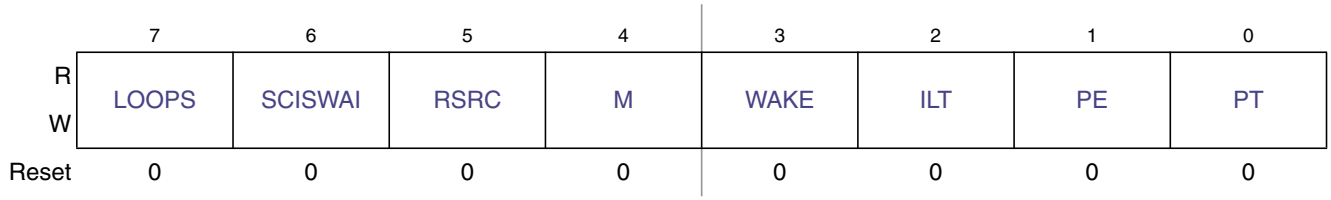
**Figure 12-4. SCI Baud Rate Register (SCI1BDL)**

**Table 12-2. SCI1BDL Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>SBR[7:0] | **Baud Rate Modulo Divisor** — These 13 bits are referred to collectively as BR, and they set the modulo divide rate for the SCI baud rate generator. When BR = 0, the SCI baud rate generator is disabled to reduce supply current. When BR = 1 to 8191, the SCI baud rate = BUSCLK/(16×BR). See also BR bits in Table 12-1. |

## 12.2.2 SCI Control Register 1 (SCI1C1)

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



| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-5. SCI Control Register 1 (SCI1C1)**

**Table 12-3. SCI1C1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>LOOPS | **Loop Mode Select** — Selects between loop back modes and normal 2-pin full-duplex modes. When LOOPS = 1, the transmitter output is internally connected to the receiver input.<br>0 Normal operation — RxD and TxD use separate pins.<br>1 Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input. (See RSRC bit.) RxD pin is not used by SCI. |
| 6<br>SCISWAI | **SCI Stops in Wait Mode**<br>0 SCI clocks continue to run in wait mode so the SCI can be the source of an interrupt that wakes up the CPU.<br>1 SCI clocks freeze while CPU is in wait mode. |
| 5<br>RSRC | **Receiver Source Select** — This bit has no meaning or effect unless the LOOPS bit is set to 1. When LOOPS = 1, the receiver input is internally connected to the TxD pin and RSRC determines whether this connection is also connected to the transmitter output.<br>0 Provided LOOPS = 1, RSRC = 0 selects internal loop back mode and the SCI does not use the RxD pins.<br>1 Single-wire SCI mode where the TxD pin is connected to the transmitter output and receiver input. |
| 4<br>M | **9-Bit or 8-Bit Mode Select**<br>0 Normal — start + 8 data bits (LSB first) + stop.<br>1 Receiver and transmitter use 9-bit data characters<br>start + 8 data bits (LSB first) + 9th data bit + stop. |
| 3<br>WAKE | **Receiver Wakeup Method Select** — Refer to Section 12.3.3.2, “Receiver Wakeup Operation” for more information.<br>0 Idle-line wakeup.<br>1 Address-mark wakeup. |
| 2<br>ILT | **Idle Line Type Select** — Setting this bit to 1 ensures that the stop bit and logic 1 bits at the end of a character do not count toward the 10 or 11 bit times of the logic high level by the idle line detection logic. Refer to Section 12.3.3.2.1, “Idle-Line Wakeup” for more information.<br>0 Idle character bit count starts after start bit.<br>1 Idle character bit count starts after stop bit. |
| 1<br>PE | **Parity Enable** — Enables hardware parity generation and checking. When parity is enabled, the most significant bit (MSB) of the data character (eighth or ninth data bit) is treated as the parity bit.<br>0 No hardware parity generation or checking.<br>1 Parity enabled. |
| 0<br>PT | **Parity Type** — Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of 1s in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even.<br>0 Even parity.<br>1 Odd parity. |

### 12.2.3 SCI Control Register 2 (SCI1C2)

This register can be read or written at any time.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-6. SCI Control Register 2 (SCI1C2)**

**Table 12-4. SCI1C2 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TIE | **Transmit Interrupt Enable (for TDRE)**<br>0 Hardware interrupts from TDRE disabled (use polling).<br>1 Hardware interrupt requested when TDRE flag is 1. |
| 6<br>TCIE | **Transmission Complete Interrupt Enable (for TC)**<br>0 Hardware interrupt requested when TC flag is 1.<br>1 Hardware interrupts from TC disabled (use polling). |
| 5<br>RIE | **Receiver Interrupt Enable (for RDRF)**<br>0 Hardware interrupts from RDRF disabled (use polling).<br>1 Hardware interrupt requested when RDRF flag is 1. |
| 4<br>ILIE | **Idle Line Interrupt Enable (for IDLE)**<br>0 Hardware interrupts from IDLE disabled (use polling).<br>1 Hardware interrupt requested when IDLE flag is 1. |
| 3<br>TE | **Transmitter Enable**<br>0 Transmitter off.<br>1 Transmitter on.<br>TE must be 1 in order to use the SCI transmitter. Normally, when TE = 1, the SCI forces the TxD pin to act as an output for the SCI system. If LOOPS = 1 and RSRC = 0, the TxD pin reverts to being a port B general-purpose I/O pin even if TE = 1.<br>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).<br>TE also can be used to queue an idle character by writing TE = 0 then TE = 1 while a transmission is in progress. Refer to Section 12.3.2.1, “Send Break and Queued Idle,” for more details.<br>When TE is written to 0, the transmitter keeps control of the port TxD pin until any data, queued idle, or queued break character finishes transmitting before allowing the pin to revert to a general-purpose I/O pin. |
| 2<br>RE | **Receiver Enable** — When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin.<br>0 Receiver off.<br>1 Receiver on. |

**Table 12-4. SCI1C2 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>RWU | **Receiver Wakeup Control** — This bit can be written to 1 to place the SCI receiver in a standby state where it waits for automatic hardware detection of a selected wakeup condition. The wakeup condition is either an idle line between messages (WAKE = 0, idle-line wakeup), or a logic 1 in the most significant data bit in a character (WAKE = 1, address-mark wakeup). Application software sets RWU and (normally) a selected hardware condition automatically clears RWU. Refer to Section 12.3.3.2, "Receiver Wakeup Operation," for more details.<br>0 Normal SCI receiver operation.<br>1 SCI receiver in standby waiting for wakeup condition. |
| 0<br>SBK | **Send Break** — Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 10 or 11 bit times of logic 0 are queued as long as SBK = 1. Depending on the timing of the set and clear of SBK relative to the information currently being transmitted, a second break character may be queued before software clears SBK. Refer to Section 12.3.2.1, "Send Break and Queued Idle," for more details.<br>0 Normal transmitter operation.<br>1 Queue break character(s) to be sent. |

## 12.2.4 SCI Status Register 1 (SCI1S1)

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| W | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 12-7. SCI Status Register 1 (SCI1S1)**

**Table 12-5. SCI1S1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TDRE | **Transmit Data Register Empty Flag** — TDRE is set immediately after 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 SCI1S1 with TDRE = 1 and then write to the SCI data register (SCI1D).<br>0 Transmit data register (buffer) full.<br>1 Transmit data register (buffer) empty. |
| 6<br>TC | **Transmission Complete Flag** — TC is set immediately after reset and when TDRE = 1 and no data, preamble, or break character is being transmitted.<br>0 Transmitter active (sending data, a preamble, or a break).<br>1 Transmitter idle (transmission activity complete).<br>TC is cleared automatically by reading SCI1S1 with TC = 1 and then doing one of the following three things:<br>• Write to the SCI data register (SCI1D) to transmit new data<br>• Queue a preamble by changing TE from 0 to 1<br>• Queue a break character by writing 1 to SBK in SCI1C2 |

**Table 12-5. SCI1S1 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>RDRF | **Receive Data Register Full Flag** — RDRF becomes set when a character transfers from the receive shifter into the receive data register (SCI1D). To clear RDRF, read SCI1S1 with RDRF = 1 and then read the SCI data register (SCI1D).<br>0 Receive data register empty.<br>1 Receive data register full. |
| 4<br>IDLE | **Idle Line Flag** — IDLE is set when the SCI receive line becomes idle for a full character time after a period of activity. When ILT = 0, the receiver starts counting idle bit times after the start bit. So if the receive character is all 1s, these bit times and the stop bit time count toward the full character time of logic high (10 or 11 bit times depending on the M control bit) needed for the receiver to detect an idle line. When ILT = 1, the receiver doesn't start counting idle bit times until after the stop bit. So the stop bit and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line.<br>To clear IDLE, read SCI1S1 with IDLE = 1 and then read the SCI data register (SCI1D). 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.<br>0 No idle line detected.<br>1 Idle line was detected. |
| 3<br>OR | **Receiver Overrun Flag** — OR is set when a new serial character is ready to be transferred to the receive data register (buffer), but the previously received character has not been read from SCI1D yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCI1D. To clear OR, read SCI1S1 with OR = 1 and then read the SCI data register (SCI1D).<br>0 No overrun.<br>1 Receive overrun (new SCI data lost). |
| 2<br>NF | **Noise Flag** — The advanced sampling technique used in the receiver takes seven samples during the start bit and three samples in each data bit and the stop bit. If any of these samples disagrees with the rest of the samples within any bit time in the frame, the flag NF will be set at the same time as the flag RDRF gets set for the character. To clear NF, read SCI1S1 and then read the SCI data register (SCI1D).<br>0 No noise detected.<br>1 Noise detected in the received character in SCI1D. |
| 1<br>FE | **Framing Error Flag** — FE is set at the same time as RDRF when the receiver detects a logic 0 where the stop bit was expected. This suggests the receiver was not properly aligned to a character frame. To clear FE, read SCI1S1 with FE = 1 and then read the SCI data register (SCI1D).<br>0 No framing error detected. This does not guarantee the framing is correct.<br>1 Framing error. |
| 0<br>PF | **Parity Error Flag** — PF is set at the same time as RDRF when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, read SCI1S1 and then read the SCI data register (SCI1D).<br>0 No parity error.<br>1 Parity error. |

## 12.2.5 SCI Status Register 2 (SCI1S2)

This register has one read-only status flag. Writes have no effect.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RAF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 12-8. SCI Status Register 2 (SCI1S2)**

**Table 12-6. SCI1S2 Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>RAF | **Receiver Active Flag** — RAF is set when the SCI receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. This status flag can be used to check whether an SCI character is being received before instructing the MCU to go to stop mode.<br>0 SCI receiver idle waiting for a start bit.<br>1 SCI receiver active (RxD input not idle). |

## 12.2.6 SCI Control Register 3 (SCI1C3)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R8 | T8 | TXDIR | 0 | ORIE | NEIE | FEIE | PEIE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 12-9. SCI Control Register 3 (SCI1C3)**

**Table 12-7. SCI1C3 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>R8 | **Ninth Data Bit for Receiver** — When the SCI is configured for 9-bit data (M = 1), R8 can be thought of as a ninth receive data bit to the left of the MSB of the buffered data in the SCI1D register. When reading 9-bit data, read R8 before reading SCI1D because reading SCI1D completes automatic flag clearing sequences which could allow R8 and SCI1D to be overwritten with new data. |
| 6<br>T8 | **Ninth Data Bit for Transmitter** — When the SCI is configured for 9-bit data (M = 1), T8 may be thought of as a ninth transmit data bit to the left of the MSB of the data in the SCI1D register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCI1D is written so T8 should be written (if it needs to change from its previous value) before SCI1D 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 SCI1D is written. |
| 5<br>TXDIR | **TxD Pin Direction in Single-Wire Mode** — When the SCI is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TxD pin.<br>0 TxD pin is an input in single-wire mode.<br>1 TxD pin is an output in single-wire mode. |

**Table 12-7. SCI1C3 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 3<br>ORIE | **Overrun Interrupt Enable** — This bit enables the overrun flag (OR) to generate hardware interrupt requests.<br>0 OR interrupts disabled (use polling).<br>1 Hardware interrupt requested when OR = 1. |
| 2<br>NEIE | **Noise Error Interrupt Enable** — This bit enables the noise flag (NF) to generate hardware interrupt requests.<br>0 NF interrupts disabled (use polling).<br>1 Hardware interrupt requested when NF = 1. |
| 1<br>FEIE | **Framing Error Interrupt Enable** — This bit enables the framing error flag (FE) to generate hardware interrupt requests.<br>0 FE interrupts disabled (use polling).<br>1 Hardware interrupt requested when FE = 1. |
| 0<br>PEIE | **Parity Error Interrupt Enable** — This bit enables the parity error flag (PF) to generate hardware interrupt requests.<br>0 PF interrupts disabled (use polling).<br>1 Hardware interrupt requested when PF = 1. |

## 12.2.7 SCI Data Register (SCI1D)

This register is actually two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer. Reads and writes of this register are also involved in the automatic flag clearing mechanisms for the SCI status flags.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-10. SCI Data Register (SCI1D)**

## 12.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

### 12.3.1 Baud Rate Generation

As shown in Figure 12-11, the clock source for the SCI baud rate generator is the bus-rate clock.

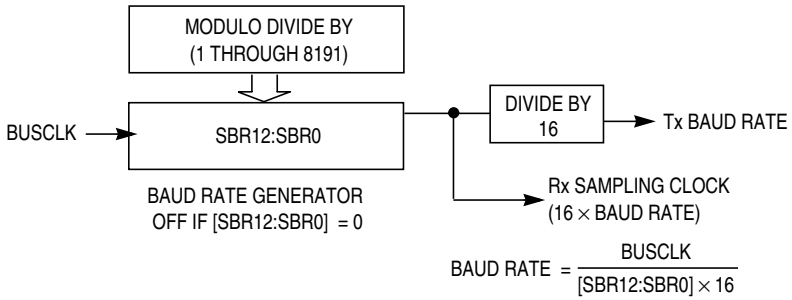



**Figure 12-11. SCI Baud Rate Generation**

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

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

### 12.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter (Figure 12-1), as well as specialized functions for sending break and idle characters.

The transmitter is enabled by setting the TE bit in SCI1C2. 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 (SCI1D).

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 SCI1D.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the TxD1 pin, the transmitter sets the transmit complete flag and enters an idle mode, with TxD1 high, waiting for more characters to transmit.

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

#### 12.3.2.1 Send Break and Queued Idle

The SBK control bit in SCI1C2 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). 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 TxD1 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 TxD1 is an output driving a logic 1. This ensures that the TxD1 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.

### 12.3.3 Receiver Functional Description

In this section, the data sampling technique used to reconstruct receiver data is described in more detail; two variations of the receiver wakeup function are explained. (The receiver block diagram is shown in Figure 12-2.)

The receiver is enabled by setting the RE bit in SCI1C2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section 12.3.5.1, “8- and 9-Bit Data Modes.” For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF) status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program

has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCI1D. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to Section 12.3.4, "Interrupts and Status Flags," for more details about flag clearing.

#### 12.3.3.1 Data Sampling Technique

The SCI receiver uses a 16× 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 RxD1 serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The 16× baud rate clock is used to divide the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

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

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

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

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

#### 12.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCI1C2. When RWU = 1, it inhibits setting of the status flags associated with the receiver, 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.

#### 12.3.3.2.1 Idle-Line Wakeup

When WAKE = 0, the receiver is configured for idle-line wakeup. In this mode, RWU is cleared automatically when the receiver detects a full character time of the idle-line level. The M control bit selects 8-bit or 9-bit data mode that determines how many bit times of idle are needed to constitute a full character time (10 or 11 bit times because of the start and stop bits). 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 doesn't start until after a stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

#### 12.3.3.2.2 Address-Mark Wakeup

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

### 12.3.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF and IDLE events, and a third vector is used for OR, NF, FE, and PF error conditions. Each of these eight 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 SCI1D. 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 TxD1 high. 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 SCI1D. The RDRF flag is cleared by reading SCI1S1 while RDRF = 1 and then reading SCI1D.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCI1S1 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 RxD1 line remains idle for an extended period of time. IDLE is cleared by reading SCI1S1 while IDLE = 1 and then reading

SCI1D. 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 and the data and any associated NF, FE, or PF condition is lost.

### 12.3.5 Additional SCI Functions

The following sections describe additional SCI functions.

#### 12.3.5.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCI1C1. 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 SCI1C3. For the receiver, the ninth bit is held in R8 in SCI1C3.

For coherent writes to the transmit data buffer, write to the T8 bit before writing to SCI1D.

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 SCI1D to the shifter.

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

#### 12.3.5.2 Stop Mode Operation

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

In stop1 and stop2 modes, all SCI register data is lost and must be re-initialized upon recovery from these two stop modes.

No SCI module registers are affected in stop3 mode.

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

#### 12.3.5.3 Loop Mode

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is

internally connected to the receiver input and the RxD1 pin is not used by the SCI, so it reverts to a general-purpose port I/O pin.

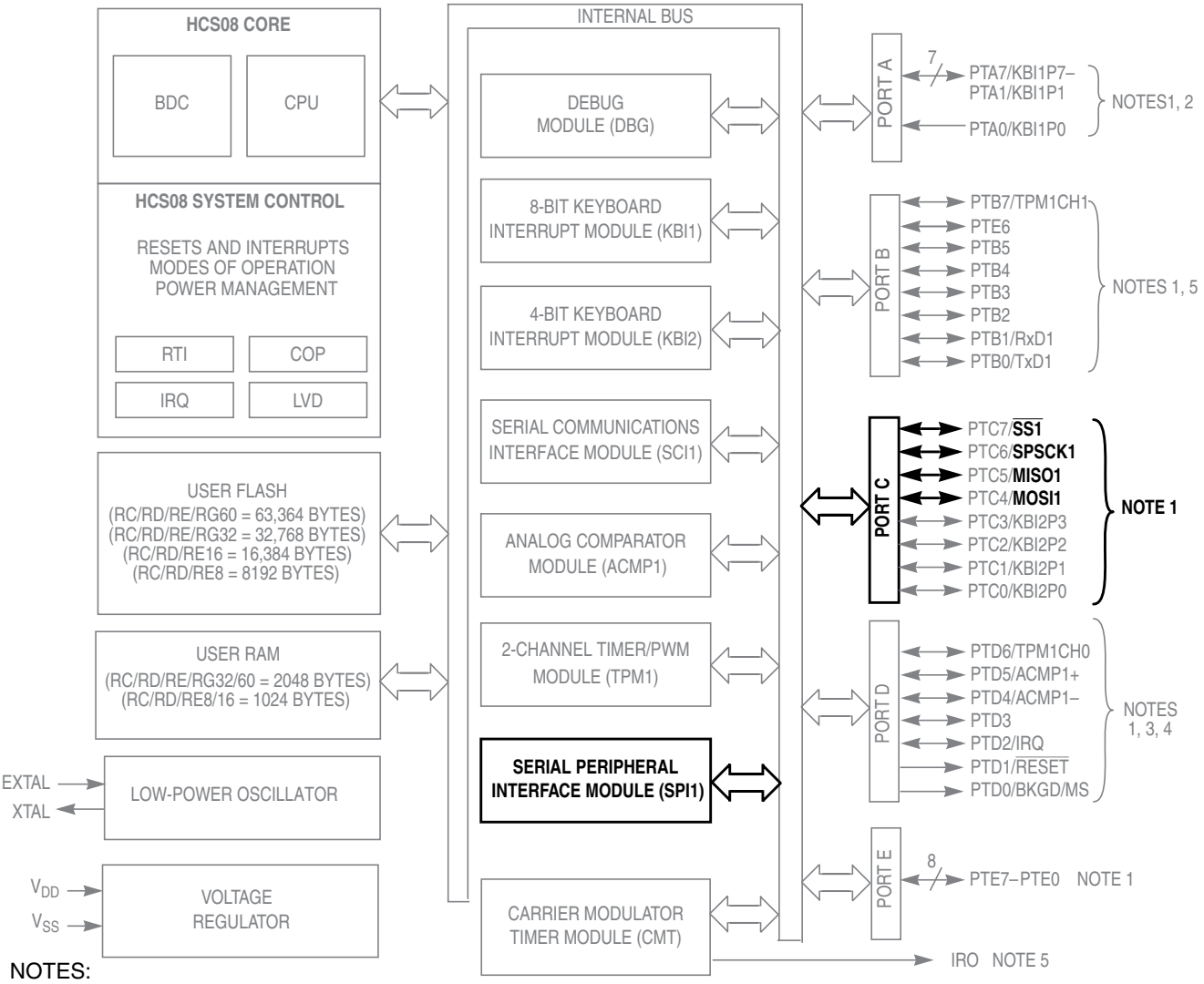
#### 12.3.5.4 Single-Wire Operation

When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD1 pin. The RxD1 pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCI1C3 controls the direction of serial data on the TxD1 pin. When TXDIR = 0, the TxD1 pin is an input to the SCI receiver and the transmitter is temporarily disconnected from the TxD1 pin so an external device can send serial data to the receiver. When TXDIR = 1, the TxD1 pin is an output driven by the transmitter. In single-wire mode, the internal loop back connection from the transmitter to the receiver causes the receiver to receive characters that are sent out by the transmitter.

# Chapter 13
# Serial Peripheral Interface (S08SPIV3)

The SPI is only available on the MC9S08RGxx versions of this family of microcontrollers. The SPI pins are shared with PTC4-PTC7 port pins. When the SPI is enabled these pins are controlled by the SPI module.




NOTES:
1. Port pins are software configurable with pullup device if input port
2. PTA0 does not have a clamp diode to VDD. PTA0 should not be driven above VDD. Also, PTA0 does not pullup to VDD when internal pullup is enabled.
3. IRQ pin contains software configurable pullup/pulldown device if IRQ enabled (IRQPE = 1)
4. The RESET pin contains integrated pullup device enabled if reset enabled (RSTPE = 1)
5. High current drive
6. Pins PTA[7:4] contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

**Figure 13-1. MC9S08RC/RD/RE/RG Block Diagram Highlighting SPI Block and Pins**

## 13.1 Features

Features of the SPI module include:

- Master or slave mode operation
- Full-duplex or single-wire bidirectional option
- Programmable transmit bit rate
- Double-buffered transmit and receive
- Serial clock phase and polarity options
- Slave select output
- Selectable MSB-first or LSB-first shifting

## 13.2 Block Diagrams

This section includes block diagrams showing SPI system connections, the internal organization of the SPI module, and the SPI clock dividers that control the master mode bit rate.

### 13.2.1 SPI System Block Diagram

Figure 13-2 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI1 pin) to the slave while simultaneously shifting data in (on the MISO1 pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK1 signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ($\overline{SS1}$ pin). In this system, the master device has configured its $\overline{SS1}$ pin as an optional slave select output.

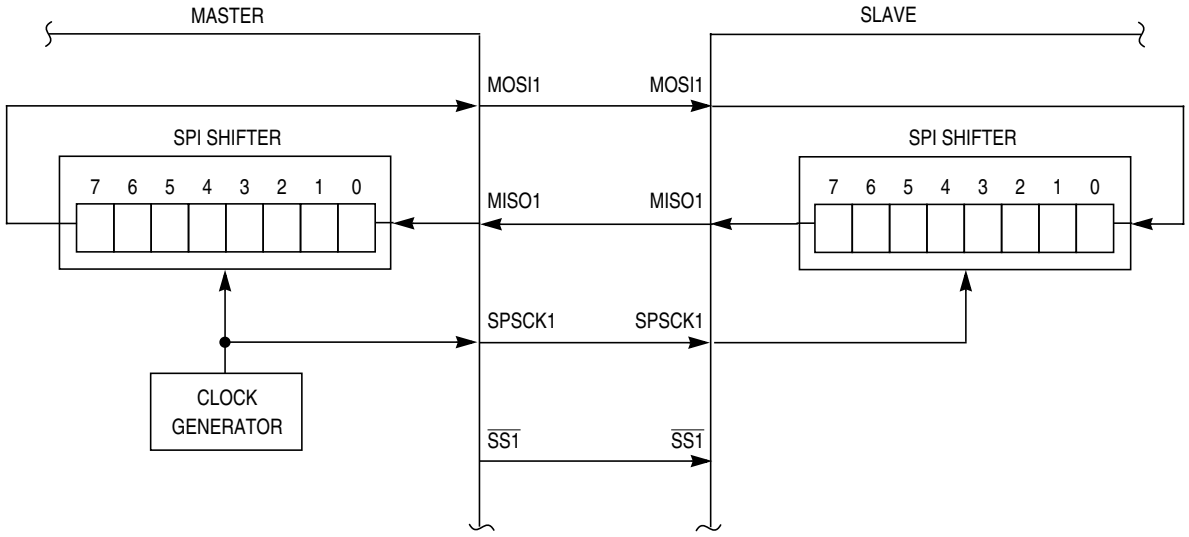



**Figure 13-2. SPI System Connections**

The most common uses of the SPI system include connecting simple shift registers for adding input or output ports or connecting small peripheral devices such as serial A/D or D/A converters. Although Figure 13-2 shows a system where data is exchanged between two MCUs, many practical systems involve simpler connections where data is unidirectionally transferred from the master MCU to a slave or from a slave to the master MCU.

### 13.2.2 SPI Module Block Diagram

Figure 13-3 is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPI1D) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in a byte of data, the data is transferred into the double-buffered receiver where it can be read (read from SPI1D). Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCK1 pin, the shifter output is routed to MOSI1, and the shifter input is routed from the MISO1 pin.

When the SPI is configured as a slave, the SPSCK1 pin is routed to the clock input of the SPI, the shifter output is routed to MISO1, and the shifter input is routed from the MOSI1 pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

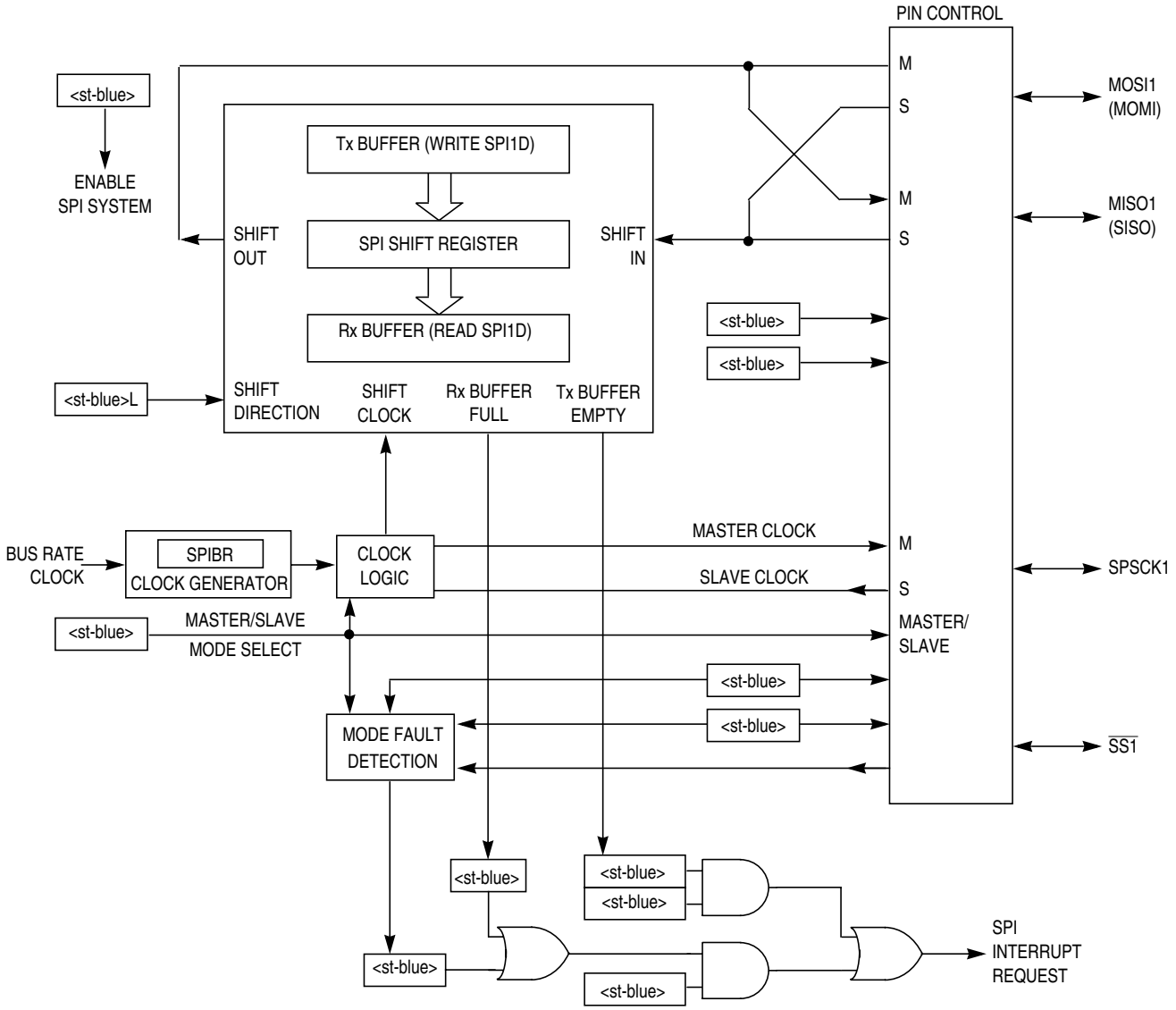



**Figure 13-3. SPI Module Block Diagram**

### 13.2.3 SPI Baud Rate Generation

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

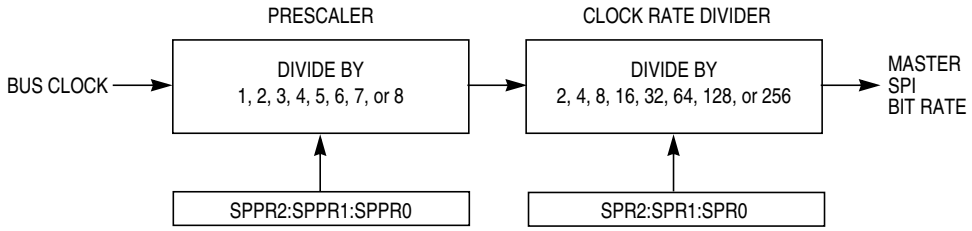



**Figure 13-4. SPI Baud Rate Generation**

## 13.3 Functional Description

An SPI transfer is initiated by checking for the SPI transmit buffer empty flag (SPTEF = 1) and then writing a byte of data to the SPI data register (SPI1D) in the master SPI device. When the SPI shift register is available, this byte of data is moved from the transmit data buffer to the shifter, SPTEF is set to indicate there is room in the buffer to queue another transmit character if desired, and the SPI serial transfer starts.

During the SPI transfer, data is sampled (read) on the MISO1 pin at one SPSCK edge and shifted, changing the bit value on the MOSI1 pin, one-half SPSCK cycle later. After eight SPSCK cycles, the data that was in the shift register of the master has been shifted out the MOSI1 pin to the slave while eight bits of data were shifted in the MISO1 pin into the master's shift register. At the end of this transfer, the received data byte is moved from the shifter into the receive data buffer and SPRF is set to indicate the data can be read by reading SPI1D. If another byte of data is waiting in the transmit buffer at the end of a transfer, it is moved into the shifter, SPTEF is set, and a new transfer is started.

Normally, SPI data is transferred most significant bit (MSB) first. If the least significant bit first enable (LSBFE) bit is set, SPI data is shifted LSB first.

When the SPI is configured as a slave, its $\overline{SS1}$ pin must be driven low before a transfer starts and $\overline{SS1}$ must stay low throughout the transfer. If a clock format where CPHA = 0 is selected, $\overline{SS1}$ must be driven to a logic 1 between successive transfers. If CPHA = 1, $\overline{SS1}$ may remain low between successive transfers. See Section 13.3.1, "SPI Clock Formats," for more details.

Because the transmitter and receiver are double buffered, a second byte, in addition to the byte currently being shifted out, can be queued into the transmit data buffer, and a previously received character can be in the receive data buffer while a new character is being shifted in. The SPTEF flag indicates when the transmit buffer has room for a new character. The SPRF flag indicates when a received character is available in the receive data buffer. The received character must be read out of the receive buffer (read SPI1D) before the next transfer is finished or a receive overrun error results.

In the case of a receive overrun, the new data is lost because the receive buffer still held the previous character and was not ready to accept the new data. There is no indication for such an overrun condition so the application system designer must ensure that previous data has been read from the receive buffer before a new transfer is initiated.

### 13.3.1 SPI Clock Formats

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

Figure 13-5 shows the clock formats when CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the sixteenth SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

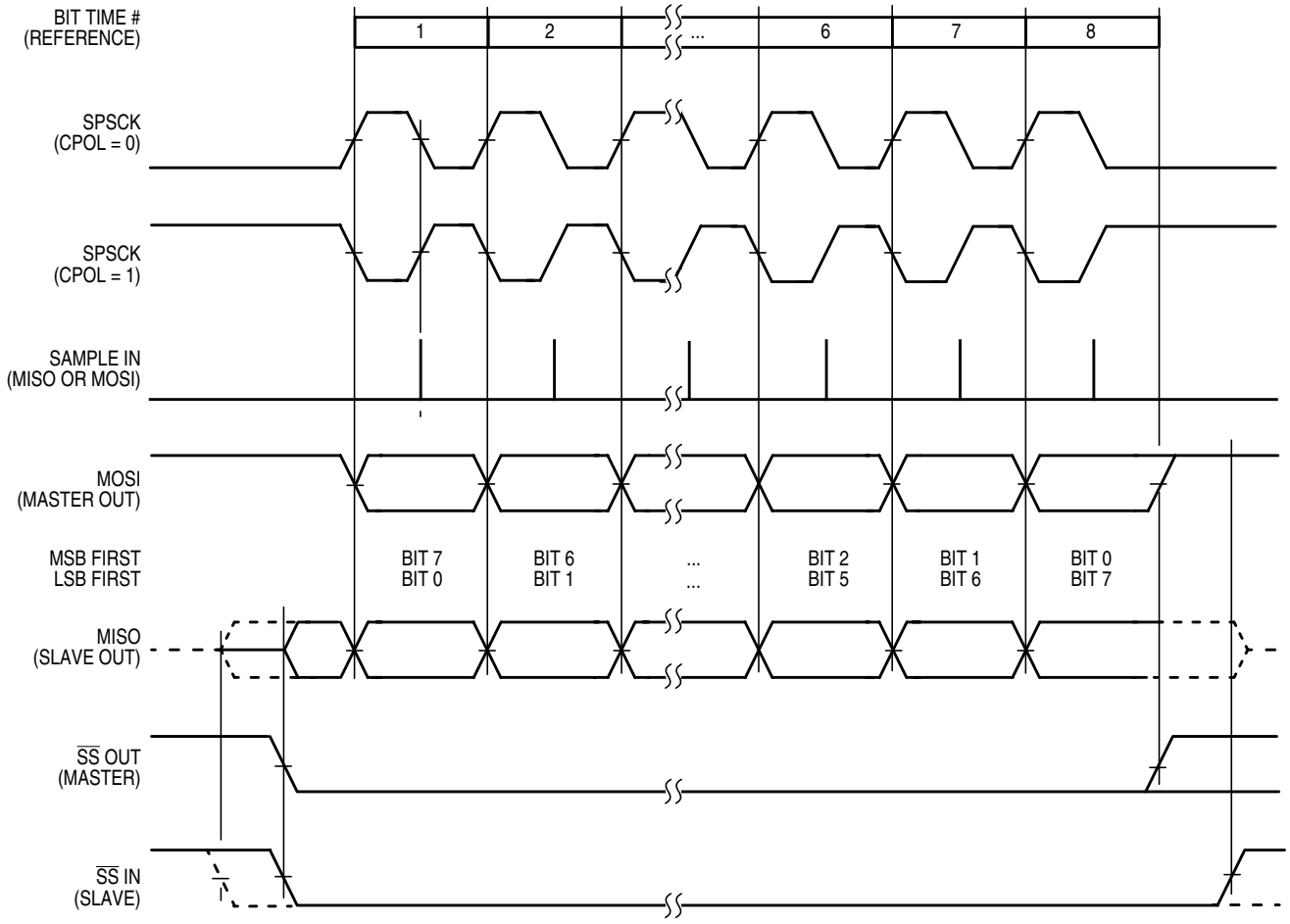



**Figure 13-5. SPI Clock Formats (CPHA = 1)**

When CPHA = 1, the slave begins to drive its MISO output when $\overline{SS1}$ goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CHPA = 1, the slave's $\overline{SS}$ input is not required to go to its inactive high level between transfers.

Figure 13-6 shows the clock formats when CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ($\overline{SS}$ IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCK cycle after the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

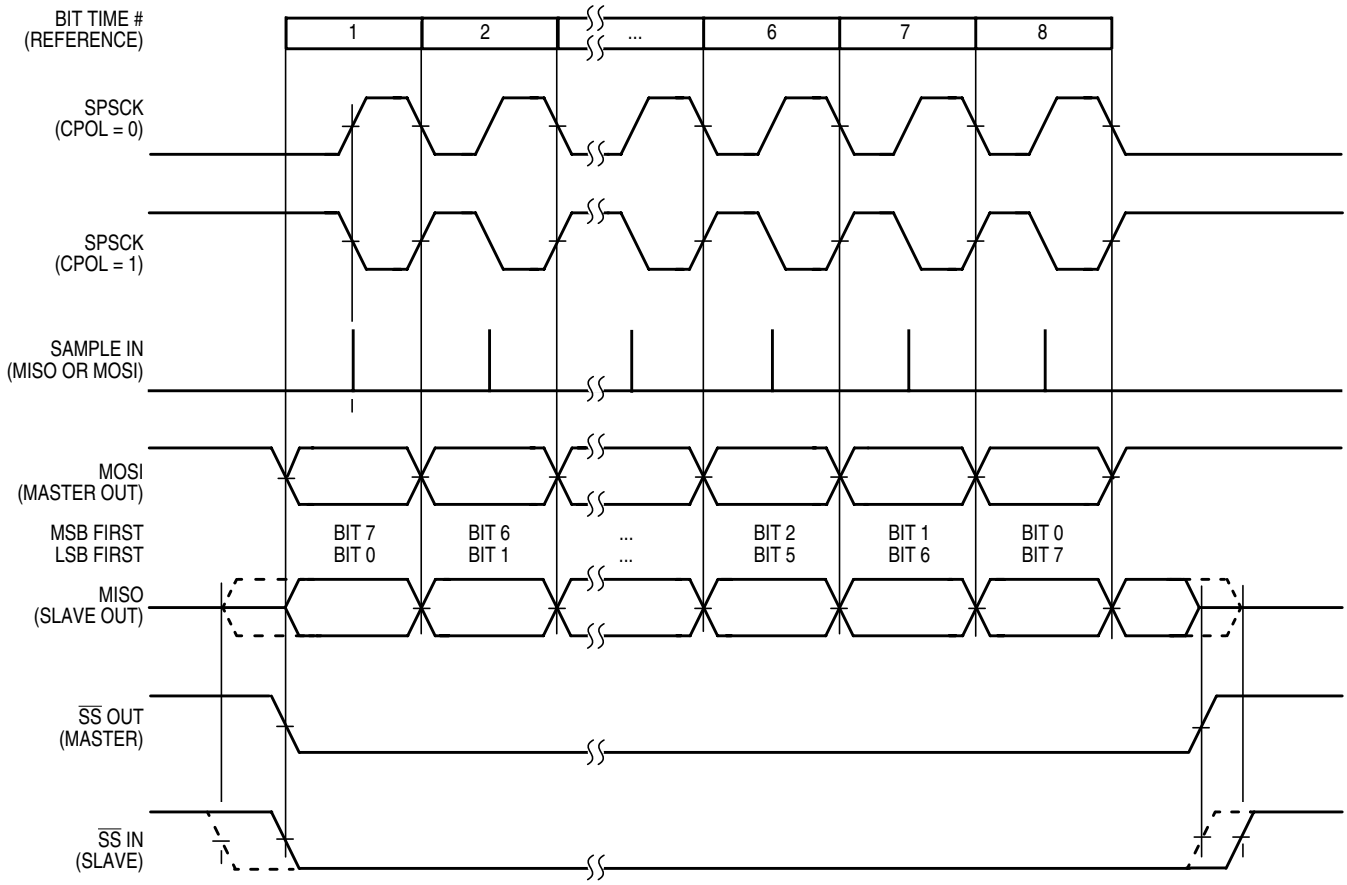



**Figure 13-6. SPI Clock Formats (CPHA = 0)**

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when $\overline{SS1}$ goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's $\overline{SS}$ input must go to its inactive high level between transfers.

## 13.3.2 SPI Pin Controls

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled (SPE = 0), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

### 13.3.2.1 SPSCK1 — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

### 13.3.2.2 MOSI1 — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data input. If SPC0 = 1 to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and slave mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 13.3.2.3 MISO1 — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data output. If SPC0 = 1 to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 13.3.2.4 $\overline{SS1}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (MODFEN = 0), this pin is not used by the SPI and reverts to being a general-purpose port I/O pin. When the SPI is enabled as a master and MODFEN = 1, the slave select output enable bit determines whether this pin acts as the mode fault input (SSOE = 0) or as the slave select output (SSOE = 1).

### 13.3.3 SPI Interrupts

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

### 13.3.4 Mode Fault Detection

A mode fault occurs and the mode fault flag (MODF) becomes set when a master SPI device detects an error on the $\overline{SS1}$ pin (provided the $\overline{SS1}$ pin is configured as the mode fault input signal). The $\overline{SS1}$ pin is configured to be the mode fault input signal when MSTR = 1, mode fault enable is set (MODFEN = 1), and slave select output enable is clear (SSOE = 0).

The mode fault detection feature can be used in a system where more than one SPI device might become a master at the same time. The error is detected when a master's $\overline{SS1}$ pin is low, indicating that some other SPI device is trying to address this master as if it were a slave. This could indicate a harmful output driver conflict, so the mode fault logic is designed to disable all SPI output drivers when such an error is detected.

When a mode fault is detected, MODF is set and MSTR is cleared to change the SPI configuration back to slave mode. The output drivers on the SPSCK1, MOSI1, and MISO1 (if not bidirectional mode) are disabled.

MODF is cleared by reading it while it is set, then writing to the SPI control register 1 (SPI1C1). User software should verify the error condition has been corrected before changing the SPI back to master mode.

## 13.4 SPI Registers and Control Bits

The SPI has five 8-bit registers to select SPI options, control baud rate, report SPI 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 SPI registers. This section refers to registers and control bits only by their names, and a Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

### 13.4.1 SPI Control Register 1 (SPI1C1)

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

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Figure 13-7. SPI Control Register 1 (SPI1C1)**

SPIE — SPI Interrupt Enable (for SPRF and MODF)

This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events.
1 = When SPRF or MODF is 1, request a hardware interrupt.
0 = Interrupts from SPRF and MODF inhibited (use polling).

SPE — SPI System Enable

Disabling the SPI halts any transfer that is in progress, clears data buffers, and initializes internal state machines. SPRF is cleared and SPTEF is set to indicate the SPI transmit data buffer is empty.
1 = SPI system enabled.
0 = SPI system inactive.

SPTIE — SPI Transmit Interrupt Enable

This is the interrupt enable bit for SPI transmit buffer empty (SPTEF).
1 = When SPTEF is 1, hardware interrupt requested.
0 = Interrupts from SPTEF inhibited (use polling).

MSTR — Master/Slave Mode Select
1 = SPI module configured as a master SPI device.
0 = SPI module configured as a slave SPI device.

CPOL — Clock Polarity

This bit effectively places an inverter in series with the clock signal from a master SPI or to a slave SPI device. Refer to Section 13.3.1, “SPI Clock Formats,” for more details.
1 = Active-low SPI clock (idles high).
0 = Active-high SPI clock (idles low).

CPHA — Clock Phase

This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to Section 13.3.1, “SPI Clock Formats,” for more details.
1 = First edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer.
0 = First edge on SPSCK occurs at the middle of the first cycle of an 8-cycle data transfer.

SSOE — Slave Select Output Enable

This bit is used in combination with the mode fault enable (MODFEN) bit in SPCR2 and the master/slave (MSTR) control bit to determine the function of the $\overline{SS1}$ pin as shown in Table 13-1.

**Table 13-1. $\overline{SS1}$ Pin Function**

| MODFEN | SSOE | Master Mode | Slave Mode |
|---|---|---|---|
| 0 | 0 | General-purpose I/O (not SPI) | Slave select input |
| 0 | 1 | General-purpose I/O (not SPI) | Slave select input |
| 1 | 0 | $\overline{SS}$ input for mode fault | Slave select input |
| 1 | 1 | Automatic $\overline{SS}$ output | Slave select input |

LSBFE — LSB First (Shifter Direction)

1 = SPI serial data transfers start with least significant bit.
0 = SPI serial data transfers start with most significant bit.

## 13.4.2 SPI Control Register 2 (SPI1C2)

This read/write register is used to control optional features of the SPI system. Bits 7, 6, 5, and 2 are not implemented and always read 0.

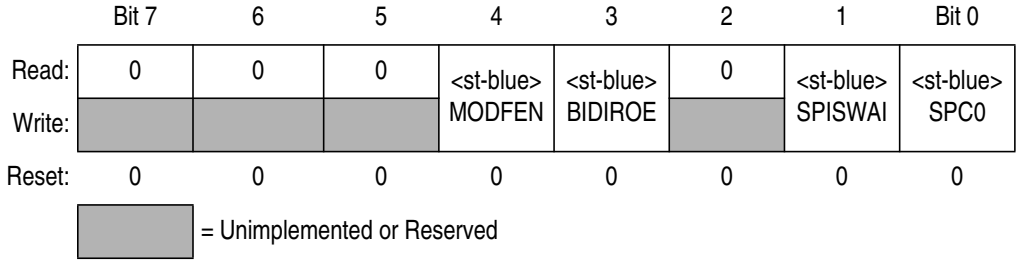



**Figure 13-8. SPI Control Register 2 (SPI1C2)**

MODFEN — Master Mode-Fault Function Enable

When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS1}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS1}$ pin is used (refer to Table 13-1 for more details).

1 = Mode fault function enabled, master $\overline{SS1}$ pin acts as the mode fault input or the slave select output.
0 = Mode fault function disabled, master $\overline{SS1}$ pin reverts to general-purpose I/O not controlled by SPI.

BIDIROE — Bidirectional Mode Output Enable

When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI1 (MOMI) or MISO1 (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect.

1 = SPI I/O pin enabled as an output.
0 = Output driver disabled so SPI data I/O pin acts as an input.

SPISWAI — SPI Stop in Wait Mode

1 = SPI clocks stop when the MCU enters wait mode.
0 = SPI clocks continue to operate in wait mode.

SPC0 — SPI Pin Control 0

The SPC0 bit chooses single-wire bidirectional mode. If MSTR = 0 (slave mode), the SPI uses the MISO1 (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (master mode), the SPI uses the MOSI1 (MOMI) pin for bidirectional SPI data transfers. When SPC0 = 1, BIDIROE is used to enable or disable the output driver for the single bidirectional SPI I/O pin.

1 = SPI configured for single-wire bidirectional operation.
0 = SPI uses separate pins for data input and data output.

## 13.4.3 SPI Baud Rate Register (SPI1BR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 13-9. SPI Baud Rate Register (SPI1BR)**

SPPR2:SPPR1:SPPR0 — SPI Baud Rate Prescale Divisor

This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in Table 13-2. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see Figure 13-4).

**Table 13-2. SPI Baud Rate Prescaler Divisor**

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

SPR2:SPR1:SPR0 — SPI Baud Rate Divisor

This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown in Figure 13-3. The input to this divider comes from the SPI baud rate prescaler (see Figure 13-4). The output of this divider is the SPI bit rate clock for master mode.

**Table 13-3. SPI Baud Rate Divisor**

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

### 13.4.4 SPI Status Register (SPI1S)

This register has three read-only status bits. Bits 6, 3, 2, 1, and 0 are not implemented and always read 0s. Writes have no meaning or effect.

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | SPRF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 13-10. SPI Status Register (SPI1S)**

SPRF — SPI Read Buffer Full Flag

SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data register (SPI1D). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register.

1 = Data available in the receive data buffer.
0 = No data available in the receive data buffer.

SPTEF — SPI Transmit Buffer Empty Flag

This bit is set when there is room in the transmit data buffer. It is cleared by reading SPI1S with SPTEF set, followed by writing a data value to the transmit buffer at SPI1D. SPI1S must be read with SPTEF = 1 before writing data to SPI1D or the SPI1D write will be ignored. SPTEF generates an SPTEF CPU interrupt request if the SPTIE bit in the SPI1C1 is also set. SPTEF is automatically set when a data byte transfers from the transmit buffer into the transmit shift register. For an idle SPI (no data in the transmit buffer or the shift register and no transfer in progress), data written to SPI1D is transferred to the shifter almost immediately so SPTEF is set within two bus cycles allowing a second 8-bit data value to be queued into the transmit buffer. After completion of the transfer of the value in the shift register, the queued value from the transmit buffer will automatically move to the shifter and SPTEF will be set to indicate there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter.

1 = SPI transmit buffer empty.
0 = SPI transmit buffer not empty.

MODF — Master Mode Fault Flag

MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The $\overline{SS1}$ pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPI1C1).

1 = Mode fault error detected.
0 = No mode fault error.

### 13.4.5 SPI Data Register (SPI1D)

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-11. SPI Data Register (SPI1D)**

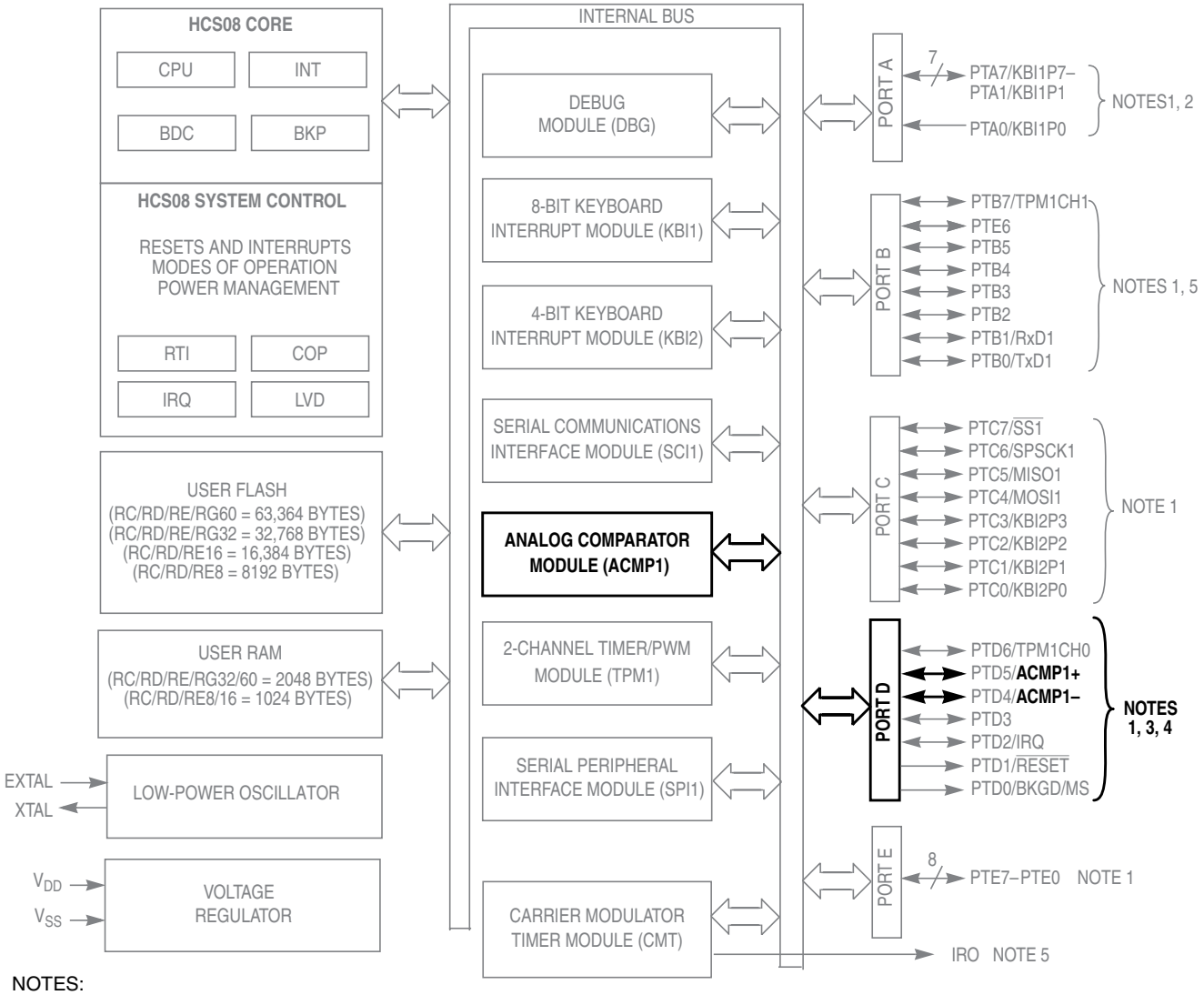
Reads of this register return the data read from the receive data buffer. Writes to this register write data to the transmit data buffer. When the SPI is configured as a master, writing data to the transmit data buffer initiates an SPI transfer.

Data should not be written to the transmit data buffer unless the SPI transmit buffer empty flag (SPTEF) is set, indicating there is room in the transmit buffer to queue a new transmit byte.

Data may be read from SPI1D any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

# Chapter 14
# Analog Comparator (S08ACMPV1)

The 32-, 44-, and 46-pin  packages of the MC9S08RCxx, MC9S08RExx, and MC9S08RGxx devices include an analog comparator. This comparator has two inputs or can optionally use an internal bandgap reference. The comparator inputs are shared with PTD4 and PTD5 port I/O pins.




NOTES:

19.Port pins are software configurable with pullup device if input port

20.PTA0 does not have a clamp diode to $V_{DD}$. PTA0 should not be driven above $V_{DD}$. Also, PTA0 does not pullup to $V_{DD}$ when internal pullup is enabled.

21.IRQ pin contains software configurable pullup/pulldown device if IRQ enabled (IRQPE = 1)

22.The RESET pin contains integrated pullup device enabled if reset enabled (RSTPE = 1)

23.High current drive

24.Pins PTA[7:4] contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn =  1) and rising edge is selected (KBEDGn = 1).

**Figure 14-1. MC9S08RC/RD/RE/RG Block Diagram Highlighting ACMP Block and Pins**

## 14.1 Features

The ACMP has the following features:

- Full rail-to-rail supply operation.
- Less than 40 mV of input offset.
- Selectable interrupt on rising edge, falling edge, or either rising or falling edge of comparator output.
- Option to compare to fixed internal bandgap reference voltage.

## 14.2 Block Diagram

The block diagram for the analog comparator module is shown below.

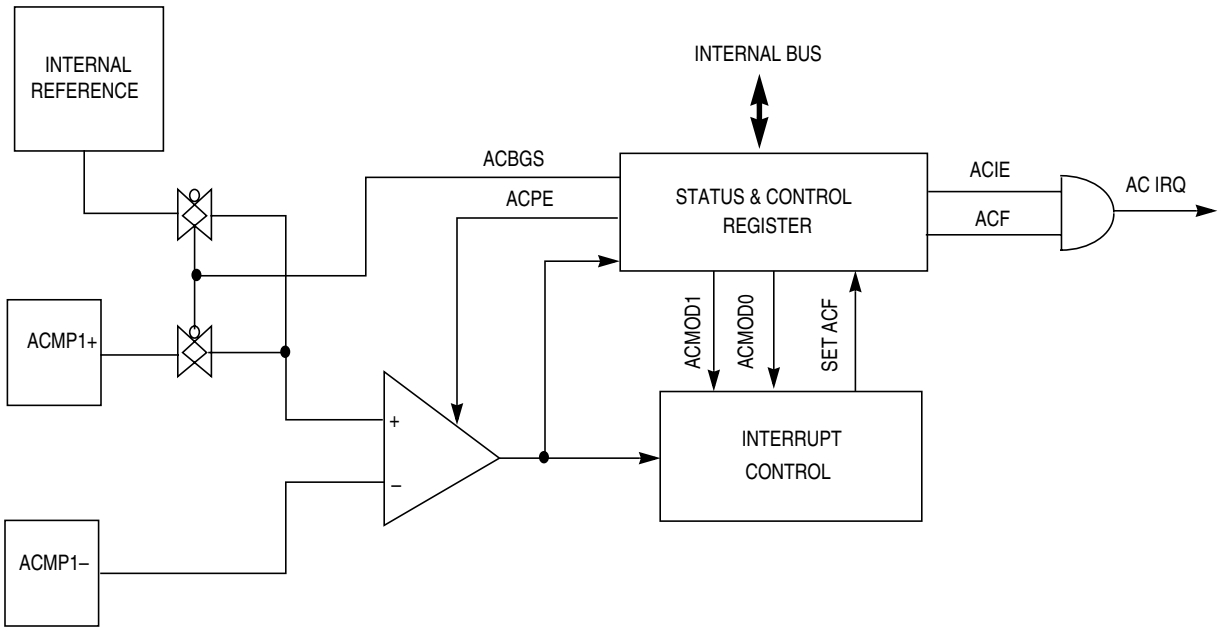



**Figure 14-2. Analog Comparator Module Block Diagram**

## 14.3 Pin Description

The ACMP has two analog input pins: ACMP1– and ACMP1+. Each of these pins can accept an input voltage that varies across the full operating voltage range of the MCU. When the ACMP1+ is configured to use the internal bandgap reference, ACMP1+ is available to be as general-purpose I/O. As shown in the block diagram, the ACMP1+ pin is connected to the comparator non-inverting input if ACBGS is equal to logic 0, and the ACMP1– pin is connected to the inverting input of the comparator.

## 14.4 Functional Description

The analog comparator can be used to compare two analog input voltages applied to ACMP1– and ACMP1+; or it can be used to compare an analog input voltage applied to ACMP1– with an internal bandgap reference voltage. The ACBGS bit is used to select the mode of operation. The comparator output is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. The ACMOD0 and ACMOD1 bits are used to select the condition that will cause the ACF bit to be set. The ACF bit can be set on a rising edge of the comparator output, a falling edge of the comparator output, or either a rising or a falling edge. The comparator output can be read directly through the ACO bit.

### 14.4.1 Interrupts

The ACMP module is capable of generating an interrupt on a compare event. The interrupt request is asserted when both the ACIE bit and the ACF bit are set. The interrupt is deasserted by clearing either the ACIE bit or the ACF bit. The ACIE bit is cleared by writing a 0 and the ACF bit is cleared by writing a 1.

### 14.4.2 Wait Mode Operation

During wait mode the ACMP, if enabled, will continue to operate normally. Also, if enabled, the interrupt can wake up the MCU.

### 14.4.3 Stop Mode Operation

During stop mode, clocks to the ACMP module are halted. The ACMP comparator circuit will enter a low power state. No compare operation will occur while in stop mode.

In stop1 and stop2 modes, the ACMP module will be in its reset state when the MCU recovers from the stop condition and must be re-initialized.

In stop3 mode, control and status register information is maintained and upon recovery normal ACMP function is available to the user.

### 14.4.4 Background Mode Operation

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

## 14.5 ACMP Status and Control Register (ACMP1SC)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | ACME | ACBGS | ACF | ACIE | ACO | 0 | ACMOD1 | ACMOD0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-3. ACMP Status and Control Register (ACMP1SC)**

**Table 14-1. ACMP1SC Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ACME | **Analog Comparator Module Enable** — The ACME bit enables the analog comparator module. When the module is not enabled, it remains in a low-power state.<br>0 Analog comparator disabled<br>1 Analog comparator enabled |
| 6<br>ACBGS | **Analog Comparator Bandgap Select** — The ACBGS bit is used to select the internal bandgap as the comparator reference.<br>0 External pin ACMP1+ selected as comparator non-inverting input<br>1 Internal bandgap reference selected as comparator non-inverting input |
| 5<br>ACF | **Analog Comparator Flag** — The ACF bit is set when a compare event occurs. Compare events are defined by the ACMOD0 and ACMOD1 bits. The ACF bit is cleared by writing a 1 to the bit.<br>0 Compare event has not occurred.<br>1 Compare event has occurred. |
| 4<br>ACIE | **Analog Comparator Interrupt Enable** — The ACIE bit enables the interrupt from the ACMP. When this bit is set, an interrupt will be asserted when the ACF bit is set.<br>0 Interrupt disabled<br>1 Interrupt enabled |
| 3<br>ACO | **Analog Comparator Output** — Reading the ACO bit will return the current value of the analog comparator output. The register bit is reset to 0 and will read as 0 when the analog comparator module is disabled (ACME = 0). |
| 1:0<br>ACMOD[1:0] | **Analog Comparator Modes** — The ACMOD1 and ACMOD0 bits select the flag setting mode that controls the type of compare event that sets the ACF bit.<br>00 Comparator output falling edge<br>01 Comparator output rising edge<br>10 Comparator output falling edge<br>11 Comparator output either rising or falling edge |

# Chapter 15
# Development Support

## 15.1 Introduction

### 15.1.1 Features

Features of the background debug controller (BDC) 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 debug module (DBG) 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:
  - A-only
  - A OR B
  - A then B
  - A AND B data (full mode)
  - A AND NOT B data (full mode)
  - Event-only B (store data)
  - A then event-only B (store data)

— Inside range (A ≤ address ≤ B)
— Outside range (address < A or address > B)

## 15.2 Background Debug Controller (BDC)

All MCUs in the HCS08 Family contain a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this system does not interfere with normal application resources. It does not use any user memory or locations in the memory map and does not share any on-chip peripherals.

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.
- Non-intrusive commands can be executed at any time even while the user's program is running. Non-intrusive commands allow a user to read or write MCU memory locations or access status and control registers within the background debug controller.

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, $\overline{\text{RESET}}$, and sometimes $V_{DD}$. An open-drain connection to reset allows the host to force a target system reset, which is useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes $V_{DD}$ can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

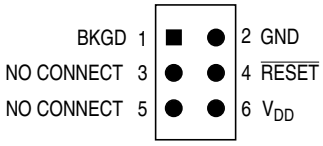



**Figure 15-1. BDM Tool Connector**

### 15.2.1 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 15.2.2, "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 15.2.2, "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 development system is connected, it can pull both BKGD and $\overline{\text{RESET}}$ low, release $\overline{\text{RESET}}$ to select active background mode rather than normal operating mode, then release BKGD. It is not necessary to reset the target MCU to communicate with it through the background debug interface.

### 15.2.2 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 15-2 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.

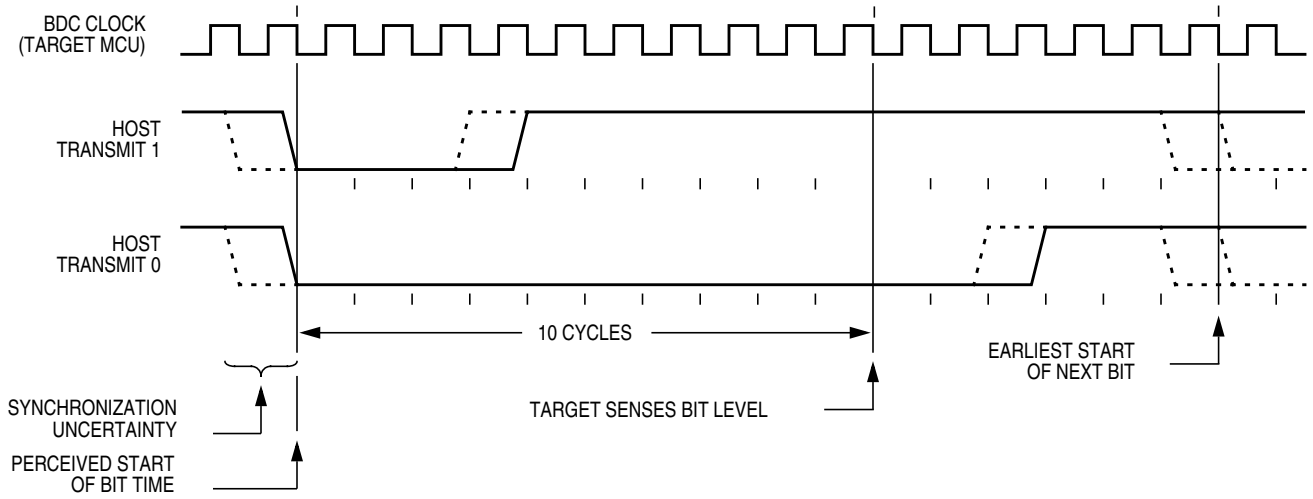



**Figure 15-2. BDC Host-to-Target Serial Bit Timing**

Figure 15-3 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.

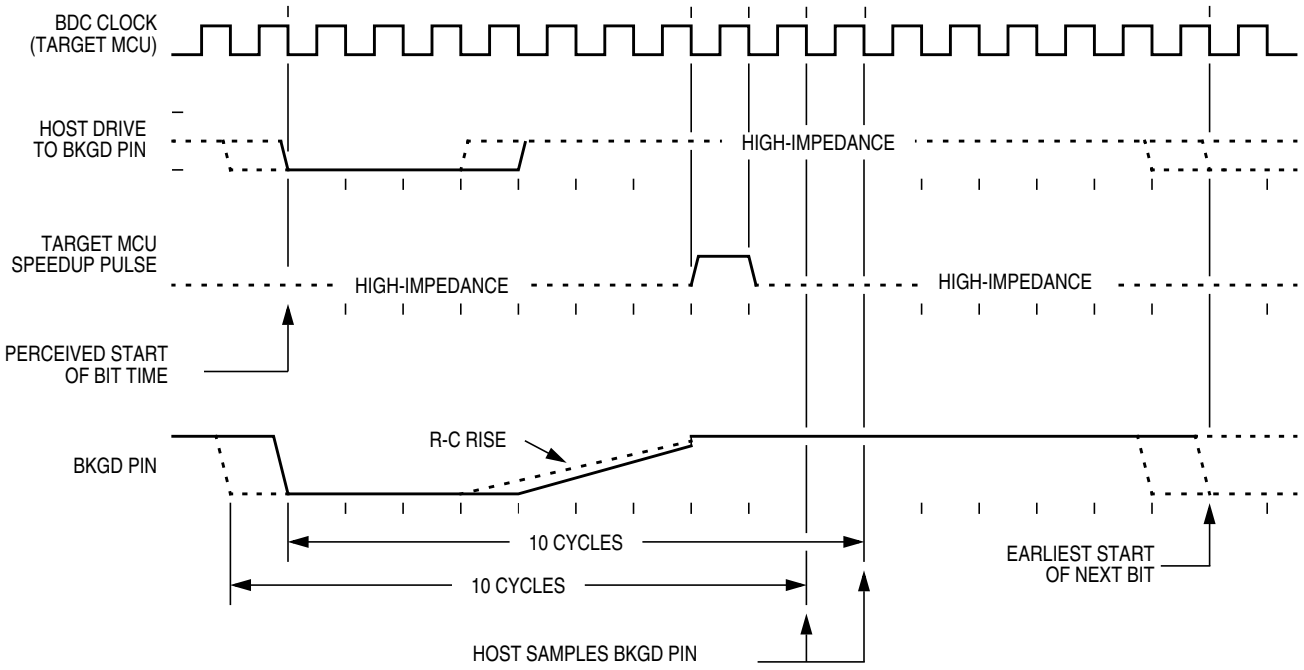



**Figure 15-3. BDC Target-to-Host Serial Bit Timing (Logic 1)**

Figure 15-4 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.

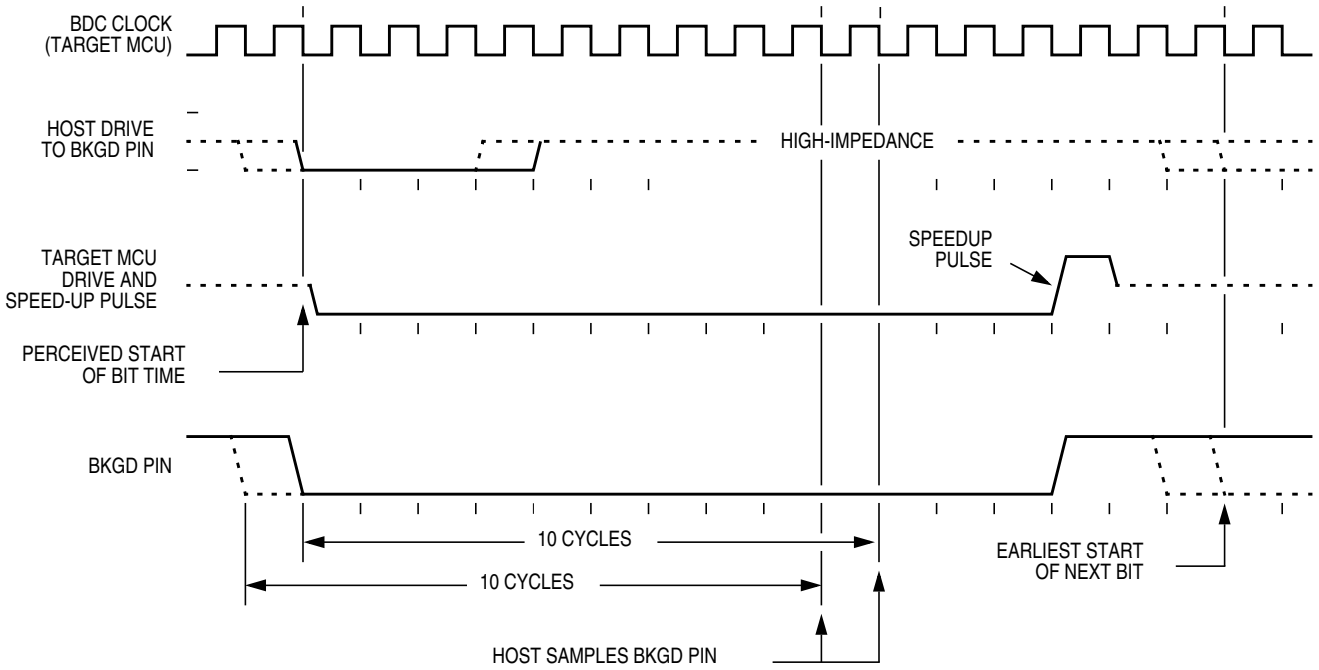



**Figure 15-4. BDM Target-to-Host Serial Bit Timing (Logic 0)**

### 15.2.3 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 15-1 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 15-1 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 15-1. BDC Command Summary**

| Command Mnemonic | Active BDM/ Non-intrusive | Coding Structure | Description |
|---|---|---|---|
| SYNC | Non-intrusive | n/a[1] | Request a timed reference pulse to determine target BDC communication speed |
| ACK_ENABLE | Non-intrusive | D5/d | Enable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D. |
| ACK_DISABLE | Non-intrusive | D6/d | Disable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D. |
| BACKGROUND | Non-intrusive | 90/d | Enter active background mode if enabled (ignore if ENBDM bit equals 0) |
| READ_STATUS | Non-intrusive | E4/SS | Read BDC status from BDCSCR |
| WRITE_CONTROL | Non-intrusive | C4/CC | Write BDC controls in BDCSCR |
| READ_BYTE | Non-intrusive | E0/AAAA/d/RD | Read a byte from target memory |
| READ_BYTE_WS | Non-intrusive | E1/AAAA/d/SS/RD | Read a byte and report status |
| READ_LAST | Non-intrusive | E8/SS/RD | Re-read byte from address just read and report status |
| WRITE_BYTE | Non-intrusive | C0/AAAA/WD/d | Write a byte to target memory |
| WRITE_BYTE_WS | Non-intrusive | C1/AAAA/WD/d/SS | Write a byte and report status |
| READ_BKPT | Non-intrusive | E2/RBKP | Read BDCBKPT breakpoint register |
| WRITE_BKPT | Non-intrusive | C2/WBKP | Write BDCBKPT breakpoint register |
| GO | Active BDM | 08/d | Go to execute the user application program starting at the address currently in the PC |
| TRACE1 | Active BDM | 10/d | Trace 1 user instruction at the address in the PC, then return to active background mode |
| TAGGO | Active BDM | 18/d | Same as GO but enable external tagging (HCS08 devices have no external tagging pin) |
| READ_A | Active BDM | 68/d/RD | Read accumulator (A) |
| READ_CCR | Active BDM | 69/d/RD | Read condition code register (CCR) |
| READ_PC | Active BDM | 6B/d/RD16 | Read program counter (PC) |
| READ_HX | Active BDM | 6C/d/RD16 | Read H and X register pair (H:X) |
| READ_SP | Active BDM | 6F/d/RD16 | Read stack pointer (SP) |
| READ_NEXT | Active BDM | 70/d/RD | Increment H:X by one then read memory byte located at H:X |
| READ_NEXT_WS | Active BDM | 71/d/SS/RD | Increment H:X by one then read memory byte located at H:X. Report status and data. |
| WRITE_A | Active BDM | 48/WD/d | Write accumulator (A) |
| WRITE_CCR | Active BDM | 49/WD/d | Write condition code register (CCR) |
| WRITE_PC | Active BDM | 4B/WD16/d | Write program counter (PC) |
| WRITE_HX | Active BDM | 4C/WD16/d | Write H and X register pair (H:X) |
| WRITE_SP | Active BDM | 4F/WD16/d | Write stack pointer (SP) |
| WRITE_NEXT | Active BDM | 50/WD/d | Increment H:X by one, then write memory byte located at H:X |
| WRITE_NEXT_WS | Active BDM | 51/WD/d/SS | Increment H:X by one, then write memory byte located at H:X. Also report status. |

[1] The SYNC command is a special operation that does not have a command code.

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct communications speed to use for BDC communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

- Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (The slowest clock is normally the reference oscillator/64 or the self-clocked rate/64.)
- Drives BKGD high for a brief speedup pulse to get a fast rise time (This speedup pulse is typically one cycle of the fastest clock in the system.)
- Removes all drive to the BKGD pin so it reverts to high impedance
- Monitors the BKGD pin for the sync response pulse

The target, upon detecting the SYNC request from the host (which is a much longer low time than would ever occur during normal BDC communications):

- Waits for BKGD to return to a logic high
- Delays 16 cycles to allow the host to stop driving the high speedup pulse
- Drives BKGD low for 128 BDC clock cycles
- Drives a 1-cycle high speedup pulse to force a fast rise time on BKGD
- Removes all drive to the BKGD pin so it reverts to high impedance

The host measures the low time of this 128-cycle sync response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the communication protocol can easily tolerate speed errors of several percent.

### 15.2.4 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.

## 15.3 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 15.3.6, "Hardware Breakpoints."

### 15.3.1 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)

### 15.3.2 Bus Capture Information and FIFO Operation

The usual way to use the FIFO is to setup the trigger mode and other control options, then arm the debugger. When the FIFO has filled or the debugger has stopped storing data into the FIFO, you would read the information out of it in the order it was stored into the FIFO. Status bits indicate the number of words of valid information that are in the FIFO as data is stored into it. If a trace run is manually halted by writing 0 to ARM before the FIFO is full (CNT = 1:0:0:0), the information is shifted by one position and

the host must perform ((8 – CNT) – 1) dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see Section 15.3.5, “Trigger Modes”), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

In trigger modes where the FIFO is storing change-of-flow addresses, there is a delay between CPU addresses and the input side of the FIFO. Because of this delay, if the trigger event itself is a change-of-flow address or a change-of-flow address appears during the next two bus cycles after a trigger event starts the FIFO, it will not be saved into the FIFO. In the case of an end-trace, if the trigger event is a change-of-flow, it will be saved as the last change-of-flow entry for that debug run.

The FIFO can also be used to generate a profile of executed instruction addresses when the debugger is not armed. When ARM = 0, reading DBGFL causes the address of the most-recently fetched opcode to be saved in the FIFO. To use the profiling feature, a host debugger would read addresses out of the FIFO by reading DBGFH then DBGFL at regular periodic intervals. The first eight values would be discarded because they correspond to the eight DBGFL reads needed to initially fill the FIFO. Additional periodic reads of DBGFH and DBGFL return delayed information about executed instructions so the host debugger can develop a profile of executed instruction addresses.

### 15.3.3 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.

### 15.3.4 Tag vs. Force Breakpoints and Triggers

Tagging is a term that refers to identifying an instruction opcode as it is fetched into the instruction queue, but not taking any other action until and unless that instruction is actually executed by the CPU. This distinction is important because any change-of-flow from a jump, branch, subroutine call, or interrupt causes some instructions that have been fetched into the instruction queue to be thrown away without being executed.

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the DBGT register is set to select tag-type operation, the output from comparator A or B is qualified by a block of logic in the debug module that tracks opcodes and only produces a trigger to the debugger if the opcode at the compare address is actually executed. There is separate opcode tracking logic for each comparator so more than one compare event can be tracked through the instruction queue at a time.

### 15.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGT register selects one of nine trigger modes. When TRGSEL = 1 in the DBGT register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGT chooses whether the FIFO begins storing data when the qualified trigger is detected (begin trace), or the FIFO stores data in a circular fashion from the time it is armed until the qualified trigger is detected (end trigger).

A debug run is started by writing a 1 to the ARM bit in the DBGC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGS. A begin-trace debug run ends when the FIFO gets full. An end-trace run ends when the selected trigger event occurs. Any debug run can be stopped manually by writing a 0 to ARM or DBGEN in DBGC.

In all trigger modes except event-only modes, the FIFO stores change-of-flow addresses. In event-only trigger modes, the FIFO stores data in the low-order eight bits of the FIFO.

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following trigger mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

**A-Only** — Trigger when the address matches the value in comparator A

**A OR B** — Trigger when the address matches either the value in comparator A or the value in comparator B

**A Then B** — Trigger when the address matches the value in comparator B but only after the address for another cycle matched the value in comparator A. There can be any number of cycles after the A match and before the B match.

**A AND B Data (Full Mode)** — This is called a full mode because address, data, and R/W (optionally) must match within the same bus cycle to cause a trigger event. Comparator A checks address, the low byte of comparator B checks data, and R/W is checked against RWA if RWAEN = 1. The high-order half of comparator B is not used.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

**A AND NOT B Data (Full Mode)** — Address must match comparator A, data must not match the low half of comparator B, and R/W must match RWA if RWAEN = 1. All three conditions must be met within the same bus cycle to cause a trigger.

In full trigger modes it is not useful to specify a tag-type CPU breakpoint (BRKEN = TAG = 1), but if you do, the comparator B data match is ignored for the purpose of issuing the tag request to the CPU and the CPU breakpoint is issued when the comparator A address matches.

**Event-Only B (Store Data)** — Trigger events occur each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

**A Then Event-Only B (Store Data)** — After the address has matched the value in comparator A, a trigger event occurs each time the address matches the value in comparator B. Trigger events cause the data to be captured into the FIFO. The debug run ends when the FIFO becomes full.

**Inside Range (A ≤ Address ≤ B)** — A trigger occurs when the address is greater than or equal to the value in comparator A and less than or equal to the value in comparator B at the same time.

**Outside Range (Address < A or Address > B)** — A trigger occurs when the address is either less than the value in comparator A or greater than the value in comparator B.

### 15.3.6 Hardware Breakpoints

The BRKEN control bit in the DBGC register may be set to 1 to allow any of the trigger conditions described in Section 15.3.5, “Trigger Modes,” to be used to generate a hardware breakpoint request to the CPU. TAG in DBGC controls whether the breakpoint request will be treated as a tag-type breakpoint or a force-type breakpoint. A tag breakpoint causes the current opcode to be marked as it enters the instruction queue. If a tagged opcode reaches the end of the pipe, the CPU executes a BGND instruction to go to active background mode rather than executing the tagged opcode. A force-type breakpoint causes the CPU to finish the current instruction and then go to active background mode.

If the background mode has not been enabled (ENBDM = 1) by a serial WRITE_CONTROL command through the BKGD pin, the CPU will execute an SWI instruction instead of going to active background mode.

## 15.4 Register Definition

This section contains the descriptions of the BDC and DBG registers and control bits.

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

### 15.4.1 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.

### 15.4.1.1 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 15-5. BDC Status and Control Register (BDCSCR)**

**Table 15-2. BDCSCR Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ENBDM | **Enable BDM (Permit Active Background Mode)** — Typically, this bit is written to 1 by the debug host shortly after the beginning of a debug session or whenever the debug host resets the target and remains 1 until a normal reset clears it.<br>0 BDM cannot be made active (non-intrusive commands still allowed)<br>1 BDM can be made active to allow active background mode commands |
| 6<br>BDMACT | **Background Mode Active Status** — This is a read-only status bit.<br>0 BDM not active (user application program running)<br>1 BDM active and waiting for serial commands |
| 5<br>BKPTEN | **BDC Breakpoint Enable** — If this bit is clear, the BDC breakpoint is disabled and the FTS (force tag select) control bit and BDCBKPT match register are ignored.<br>0 BDC breakpoint disabled<br>1 BDC breakpoint enabled |
| 4<br>FTS | **Force/Tag Select** — When FTS = 1, a breakpoint is requested whenever the CPU address bus matches the BDCBKPT match register. When FTS = 0, a match between the CPU address bus and the BDCBKPT register causes the fetched opcode to be tagged. If this tagged opcode ever reaches the end of the instruction queue, the CPU enters active background mode rather than executing the tagged opcode.<br>0 Tag opcode at breakpoint address and enter active background mode if CPU attempts to execute that instruction<br>1 Breakpoint match forces active background mode at next instruction boundary (address need not be an opcode) |
| 3<br>CLKSW | **Select Source for BDC Communications Clock** — CLKSW defaults to 0, which selects the alternate BDC clock source.<br>0 Alternate BDC clock source<br>1 MCU bus clock |

**Table 15-2. BDCSCR Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>WS | **Wait or Stop Status** — When the target CPU is in wait or stop mode, most BDC commands cannot function. However, the BACKGROUND command can be used to force the target CPU out of wait or stop and into active background mode where all BDC commands work. Whenever the host forces the target MCU into active background mode, the host should issue a READ_STATUS command to check that BDMACT = 1 before attempting other BDC commands.<br>0 Target CPU is running user application code or in active background mode (was not in wait or stop mode when background became active)<br>1 Target CPU is in wait or stop mode, or a BACKGROUND command was used to change from wait or stop to active background mode |
| 1<br>WSF | **Wait or Stop Failure Status** — This status bit is set if a memory access command failed due to the target CPU executing a wait or stop instruction at or about the same time. The usual recovery strategy is to issue a BACKGROUND command to get out of wait or stop mode into active background mode, repeat the command that failed, then return to the user program. (Typically, the host would restore CPU registers and stack values and re-execute the wait or stop instruction.)<br>0 Memory access did not conflict with a wait or stop instruction<br>1 Memory access command failed because the CPU entered wait or stop mode |
| 0<br>DVF | **Data Valid Failure Status** — This status bit is not used in the MC9S08RC/RD/RE/RG because it does not have any slow access memory.<br>0 Memory access did not conflict with a slow memory access<br>1 Memory access command failed because CPU was not finished with a slow memory access |

#### 15.4.1.2 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 15.2.4, “BDC Hardware Breakpoint.”

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

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

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

= Unimplemented or Reserved

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

**Figure 15-6. System Background Debug Force Reset Register (SBDFR)**

**Table 15-3. SBDFR Register Field Description**

| Field | Description |
|---|---|
| 0<br>BDFR | **Background Debug Force Reset** — A serial active background mode command such as WRITE_BYTE allows an external debug host to force a target system reset. Writing 1 to this bit forces an MCU reset. This bit cannot be written from a user program. |

## 15.4.3 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.

### 15.4.3.1 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.

### 15.4.3.2 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.

### 15.4.3.3 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.

### 15.4.3.4 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.

#### 15.4.3.5 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.

#### 15.4.3.6 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.

### 15.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | DBGEN | ARM | TAG | BRKEN | RWA | RWAEN | RWB | RWBEN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 15-7. Debug Control Register (DBGC)**

**Table 15-4. DBGC Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DBGEN | **Debug Module Enable** — Used to enable the debug module. DBGEN cannot be set to 1 if the MCU is secure.<br>0 DBG disabled<br>1 DBG enabled |
| 6<br>ARM | **Arm Control** — Controls whether the debugger is comparing and storing information in the FIFO. A write is used to set this bit (and ARMF) and completion of a debug run automatically clears it. Any debug run can be manually stopped by writing 0 to ARM or to DBGEN.<br>0 Debugger not armed<br>1 Debugger armed |
| 5<br>TAG | **Tag/Force Select** — Controls whether break requests to the CPU will be tag or force type requests. If BRKEN = 0, this bit has no meaning or effect.<br>0 CPU breaks requested as force type requests<br>1 CPU breaks requested as tag type requests |
| 4<br>BRKEN | **Break Enable** — Controls whether a trigger event will generate a break request to the CPU. Trigger events can cause information to be stored in the FIFO without generating a break request to the CPU. For an end trace, CPU break requests are issued to the CPU when the comparator(s) and R/W meet the trigger requirements. For a begin trace, CPU break requests are issued when the FIFO becomes full. TRGSEL does not affect the timing of CPU break requests.<br>0 CPU break requests not enabled<br>1 Triggers cause a break request to the CPU |
| 3<br>RWA | **R/W Comparison Value for Comparator A** — When RWAEN = 1, this bit determines whether a read or a write access qualifies comparator A. When RWAEN = 0, RWA and the R/W signal do not affect comparator A.<br>0 Comparator A can only match on a write cycle<br>1 Comparator A can only match on a read cycle |
| 2<br>RWAEN | **Enable R/W for Comparator A** — Controls whether the level of R/W is considered for a comparator A match.<br>0 R/W is not used in comparison A<br>1 R/W is used in comparison A |
| 1<br>RWB | **R/W Comparison Value for Comparator B** — When RWBEN = 1, this bit determines whether a read or a write access qualifies comparator B. When RWBEN = 0, RWB and the R/W signal do not affect comparator B.<br>0 Comparator B can match only on a write cycle<br>1 Comparator B can match only on a read cycle |
| 0<br>RWBEN | **Enable R/W for Comparator B** — Controls whether the level of R/W is considered for a comparator B match.<br>0 R/W is not used in comparison B<br>1 R/W is used in comparison B |

### 15.4.3.8 Debug Trigger Register (DBGT)

This register can be read any time, but may be written only if ARM = 0, except bits 4 and 5 are hard-wired to 0s.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TRGSEL | BEGIN | 0 | 0 | TRG3 | TRG2 | TRG1 | TRG0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 15-8. Debug Trigger Register (DBGT)**

**Table 15-5. DBGT Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TRGSEL | **Trigger Type** — Controls whether the match outputs from comparators A and B are qualified with the opcode tracking logic in the debug module. If TRGSEL is set, a match signal from comparator A or B must propagate through the opcode tracking logic and a trigger event is only signalled to the FIFO logic if the opcode at the match address is actually executed.<br>0 Trigger on access to compare address (force)<br>1 Trigger if opcode at compare address is executed (tag) |
| 6<br>BEGIN | **Begin/End Trigger Select** — Controls whether the FIFO starts filling at a trigger or fills in a circular manner until a trigger ends the capture of information. In event-only trigger modes, this bit is ignored and all debug runs are assumed to be begin traces.<br>0 Data stored in FIFO until trigger (end trace)<br>1 Trigger initiates data storage (begin trace) |
| 3:0<br>TRG[3:0] | **Select Trigger Mode** — Selects one of nine triggering modes, as described below.<br>0000 A-only<br>0001 A OR B<br>0010 A Then B<br>0011 Event-only B (store data)<br>0100 A then event-only B (store data)<br>0101 A AND B data (full mode)<br>0110 A AND NOT B data (full mode)<br>0111 Inside range: A ≤ address ≤ B<br>1000 Outside range: address < A or address > B<br>1001 – 1111 (No trigger) |

### 15.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | AF | BF | ARMF | 0 | CNT3 | CNT2 | CNT1 | CNT0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 15-9. Debug Status Register (DBGS)**

**Table 15-6. DBGS Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>AF | **Trigger Match A Flag** — AF is cleared at the start of a debug run and indicates whether a trigger match A condition was met since arming.<br>0 Comparator A has not matched<br>1 Comparator A match |
| 6<br>BF | **Trigger Match B Flag** — BF is cleared at the start of a debug run and indicates whether a trigger match B condition was met since arming.<br>0 Comparator B has not matched<br>1 Comparator B match |
| 5<br>ARMF | **Arm Flag** — While DBGEN = 1, this status bit is a read-only image of ARM in DBGC. This bit is set by writing 1 to the ARM control bit in DBGC (while DBGEN = 1) and is automatically cleared at the end of a debug run. A debug run is completed when the FIFO is full (begin trace) or when a trigger event is detected (end trace). A debug run can also be ended manually by writing 0 to ARM or DBGEN in DBGC.<br>0 Debugger not armed<br>1 Debugger armed |
| 3:0<br>CNT[3:0] | **FIFO Valid Count** — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO.<br>0000 Number of valid words in FIFO = No valid data<br>0001 Number of valid words in FIFO = 1<br>0010 Number of valid words in FIFO = 2<br>0011 Number of valid words in FIFO = 3<br>0100 Number of valid words in FIFO = 4<br>0101 Number of valid words in FIFO = 5<br>0110 Number of valid words in FIFO = 6<br>0111 Number of valid words in FIFO = 7<br>1000 Number of valid words in FIFO = 8 |

# Appendix A
# Electrical Characteristics

## A.1 Introduction

This section contains electrical and timing specifications.

## A.2 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-1 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 $V_{DD}$) or the programmable pull-up resistor associated with the pin is enabled.

**Table A-1. Absolute Maximum Ratings**

| Rating | Symbol | Value | Unit |
|---|---|---|---|
| Supply voltage | $V_{DD}$ | –0.3 to +3.8 | V |
| Maximum current into $V_{DD}$ | $I_{DD}$ | 120 | mA |
| Digital input voltage | $V_{In}$ | –0.3 to $V_{DD}$ + 0.3 | V |
| Instantaneous maximum current<br>Single pin limit (applies to all port pins)(1), (2), (3) | $I_D$ | ± 25 | mA |
| Storage temperature range | $T_{stg}$ | –55 to 150 | °C |

1. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive ($V_{DD}$) and negative ($V_{SS}$) clamp voltages, then use the larger of the two resistance values.
2. All functional non-supply pins are internally clamped to $V_{SS}$ and $V_{DD}$.
3. Power supply must maintain regulation within operating $V_{DD}$ range during instantaneous and operating maximum current conditions. If positive injection current ($V_{In} > V_{DD}$) is greater than $I_{DD}$, the injection current may flow out of $V_{DD}$ and could result in external power supply going out of regulation. Ensure external $V_{DD}$ load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if the clock rate is very low (which would reduce overall power consumption).

## A.3 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 voltage regulator circuits 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-2. Thermal Characteristics**

| Rating | Symbol | Value | Unit |
|---|---|---|---|
| Operating temperature range (packaged) | $T_A$ | $T_L$ to $T_H$<br>–40 to 85 | °C |
| Thermal resistance<br>28-pin PDIP<br>28-pin SOIC<br>32-pin LQFP<br>44-pin LQFP<br>48-pin QFN | $\theta_{JA}$ | <br>75<br>70<br>72<br>70<br>82 | °C/W |

The average chip-junction temperature ($T_J$) in °C can be obtained from:

$$T_J = T_A + (P_D \times \theta_{JA}) \qquad \textit{Eqn. A-1}$$

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, $P_{I/O} << P_{int}$ and can be neglected. An approximate relationship between $P_D$ and $T_J$ (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \qquad \textit{Eqn. A-2}$$

Solving equations 1 and 2 for K gives:

$$K = P_D \times (T_A + 273°C) + \theta_{JA} \times (P_D)^2 \qquad \textit{Eqn. A-3}$$

where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring $P_D$ (at equilibrium) for a known $T_A$. Using this value of K, the values of $P_D$ and $T_J$ can be obtained by solving equations 1 and 2 iteratively for any value of $T_A$.

## A.4 Electrostatic Discharge (ESD) Protection Characteristics

Although damage from static discharge is much less common on these devices than on early CMOS circuits, normal handling precautions should be used to avoid exposure to static discharge. Qualification tests are performed to ensure that these devices can withstand exposure to reasonable levels of static without suffering any permanent damage. All ESD testing is in conformity with CDF-AEC-Q00 Stress Test Qualification for Automotive Grade Integrated Circuits. (http://www.aecouncil.com/) A device is considered to have failed if, after exposure to ESD pulses, the device no longer meets the device specification requirements. Complete dc parametric and functional testing is performed per the applicable device data sheet at room temperature followed by hot temperature, unless specified otherwise in the device data sheet.

**Table A-3. ESD Protection Characteristics**

| Parameter | Symbol | Value | Unit |
|---|---|---|---|
| ESD Target for Machine Model (MM)<br>MM circuit description | $V_{THMM}$ | 200 | V |
| ESD Target for Human Body Model (HBM)<br>HBM circuit description | $V_{THHBM}$ | 2000 | V |

## 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 (Temperature Range = 0 to 70°C Ambient)**

| Parameter | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Low-voltage detection threshold | $V_{LVD}$ | 1.82 | 1.875 | 1.90 | V |
| Power on reset (POR) voltage | $V_{POR}$ | 0.8 | 0.9 | 1.1 | V |
| Maximum low-voltage safe state re-arm[1] | $V_{REARM}$ | 1.90 | 2.24 | 2.60 | V |

1. If SAFE bit is set, $V_{DD}$ must be above re-arm voltage to allow MCU to accept interrupts, refer to Section 5.6, "Low-Voltage Detect (LVD) System."

**Table A-5. DC Characteristics (Temperature Range = –40 to 85°C Ambient)**

| Parameter | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Supply voltage (run, wait and stop modes.)<br>$0 < f_{Bus} < 8$ MHz | $V_{DD}$ | 1.8 | | 3.6 | V |
| Minimum RAM retention supply voltage applied to $V_{DD}$ | $V_{RAM}$ | $V_{POR}$[1], [2] | | — | V |
| Low-voltage detection threshold<br>($V_{DD}$ falling)<br>($V_{DD}$ rising) | $V_{LVD}$ | <br>1.82<br>1.92 | <br>1.88<br>1.96 | <br>1.93<br>2.01 | V |
| Low-voltage warning threshold<br>($V_{DD}$ falling)<br>($V_{DD}$ rising) | $V_{LVW}$ | <br>2.07<br>2.16 | <br>2.13<br>2.21 | <br>2.18<br>2.26 | V |
| Power on reset (POR) voltage | $V_{POR}$ | 0.85 | 1.0 | 1.2 | V |

**Table A-5. DC Characteristics (Temperature Range = –40 to 85°C Ambient) (continued)**

| Parameter | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Maximum low-voltage safe state re-arm[3] | $V_{REARM}$ | — | — | 3.0 | V |
| Input high voltage ($V_{DD}$ > 2.3 V) (all digital inputs) | $V_{IH}$ | 0.70 × $V_{DD}$ | | — | V |
| Input high voltage (1.8 V ≤ $V_{DD}$ ≤ 2.3 V) (all digital inputs) | $V_{IH}$ | 0.85 × $V_{DD}$ | | — | V |
| Input low voltage ($V_{DD}$ > 2.3 V) (all digital inputs) | $V_{IL}$ | — | | 0.35 × $V_{DD}$ | V |
| Input low voltage (1.8 V ≤ $V_{DD}$ ≤ 2.3 V) (all digital inputs) | $V_{IL}$ | — | | 0.30 × $V_{DD}$ | V |
| Input hysteresis (all digital inputs) | $V_{hys}$ | 0.06 × $V_{DD}$ | | — | V |
| Input leakage current (Per pin) $V_{In}$ = $V_{DD}$ or $V_{SS}$, all input only pins | $\lvert I_{In}\rvert$ | — | 0.025 | 1.0 | µA |
| High impedance (off-state) leakage current (per pin) $V_{In}$ = $V_{DD}$ or $V_{SS}$, all input/output | $\lvert I_{OZ}\rvert$ | — | 0.025 | 1.0 | µA |
| Internal pullup resistors[4] [5] | $R_{PU}$ | 17.5 | | 52.5 | κW |
| Internal pulldown resistor (IRQ) | $R_{PD}$ | 17.5 | | 52.5 | κW |
| Output high voltage ($V_{DD}$ ≥ 1.8 V) $I_{OH}$ = –2 mA (ports A, C, D and E) | $V_{OH}$ | $V_{DD}$ – 0.5 | | — | V |
| Output high voltage (port B and IRO)<br>$I_{OH}$ = –10 mA ($V_{DD}$ ≥ 2.7 V)<br>$I_{OH}$ = –6 mA ($V_{DD}$ ≥ 2.3 V)<br>$I_{OH}$ = –3 mA ($V_{DD}$ ≥ 1.8 V) | | $V_{DD}$ – 0.5 | | —<br>—<br>— | |
| Maximum total $I_{OH}$ for all port pins | $\lvert I_{OHT}\rvert$ | — | | 60 | mA |
| Output low voltage ($V_{DD}$ ≥ 1.8 V) $I_{OL}$ = 2.0 mA (ports A, C, D and E) | $V_{OL}$ | — | | 0.5 | V |
| Output low voltage (port B)<br>$I_{OL}$ = 10.0 mA ($V_{DD}$ ≥ 2.7 V)<br>$I_{OL}$ = 6 mA ($V_{DD}$ ≥ 2.3 V)<br>$I_{OL}$ = 3 mA ($V_{DD}$ ≥ 1.8 V) | | —<br>—<br>— | | 0.5<br>0.5<br>0.5 | |
| Output low voltage (IRO)<br>$I_{OL}$ = 16 mA ($V_{DD}$ ≥ 2.7 V)<br>$I_{OL}$ = 6 mA ($V_{DD}$ ≥ 2.3 V)<br>$I_{OL}$ = 3 mA ($V_{DD}$ ≥ 1.8 V) | | —<br>—<br>— | | 1.2<br>1.2<br>1.2 | |
| Maximum total $I_{OL}$ for all port pins | $I_{OLT}$ | — | | 60 | mA |
| dc injection current[2], [6], [7], [8],, [9]<br>$V_{IN}$ < $V_{SS}$, $V_{IN}$ > $V_{DD}$<br>Single pin limit<br>Total MCU limit, includes sum of all stressed pins | $\lvert I_{IC}\rvert$ | <br><br>—<br>— | | <br><br>0.2<br>5 | <br><br>mA<br>mA |
| Input capacitance (all non-supply pins) | $C_{In}$ | — | | 7 | pF |

1. RAM will retain data down to POR voltage. RAM data not guaranteed to be valid following a POR.
2. This parameter is characterized and not tested on each device.
3. If SAFE bit is set, $V_{DD}$ must be above re-arm voltage to allow MCU to accept interrupts, refer to Section 5.6, "Low-Voltage Detect (LVD) System."
4. Measurement condition for pull resistors: $V_{In}$ = $V_{SS}$ for pullup and $V_{In}$ = $V_{DD}$ for pulldown.
5. The PTA0 pullup resistor may not pull up to the specified minimum $V_{IH}$. However, all ports are functionally tested to guarantee that a logic 1 will be read on any port input when the pullup is enabled and no dc load is present on the pin.
6. All functional non-supply pins are internally clamped to $V_{SS}$ and $V_{DD}$.

7. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.
8. Power supply must maintain regulation within operating $V_{DD}$ range during instantaneous and operating maximum current conditions. If positive injection current ($V_{In} > V_{DD}$) is greater than $I_{DD}$, the injection current may flow out of $V_{DD}$ and could result in external power supply going out of regulation. Ensure external $V_{DD}$ load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if clock rate is very low (which would reduce overall power consumption).
9. PTA0 does not have a clamp diode to $V_{DD}$. Do not drive PTA0 above $V_{DD}$.

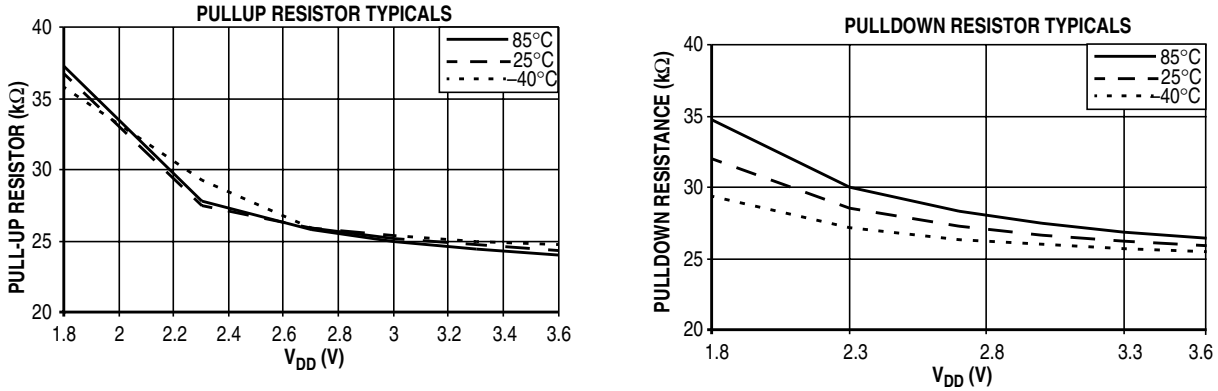



**Figure A-1. Pullup and Pulldown Typical Resistor Values ($V_{DD}$ = 3.0 V)**

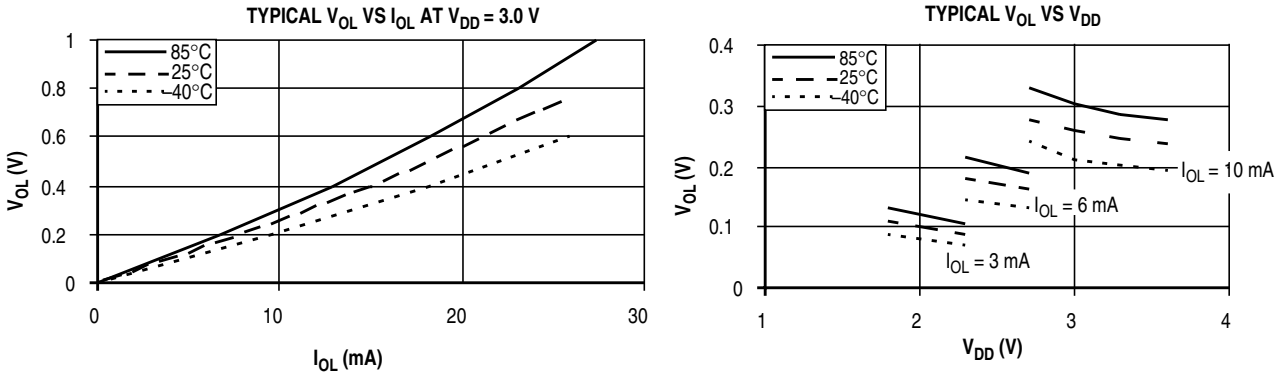



**Figure A-2. Typical Low-Side Driver (Sink) Characteristics (Port B and IRO)**

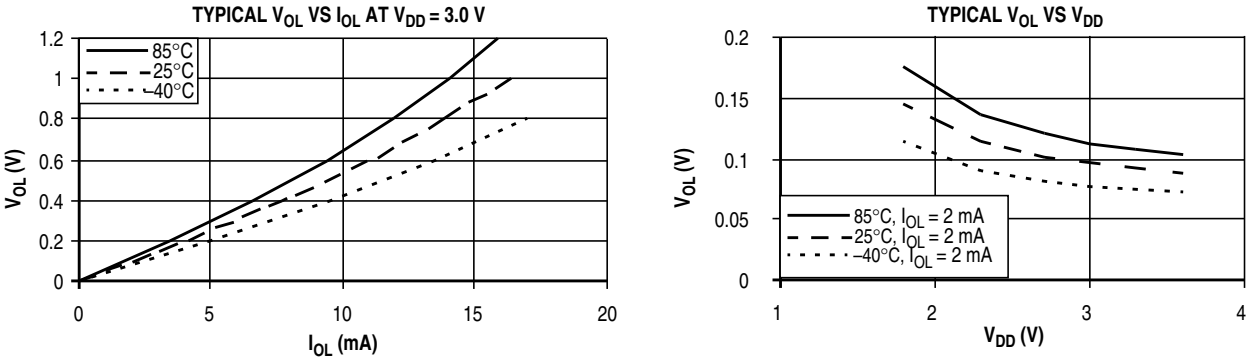



**Figure A-3. Typical Low-Side Driver (Sink) Characteristics (Ports A, C, D and E)**

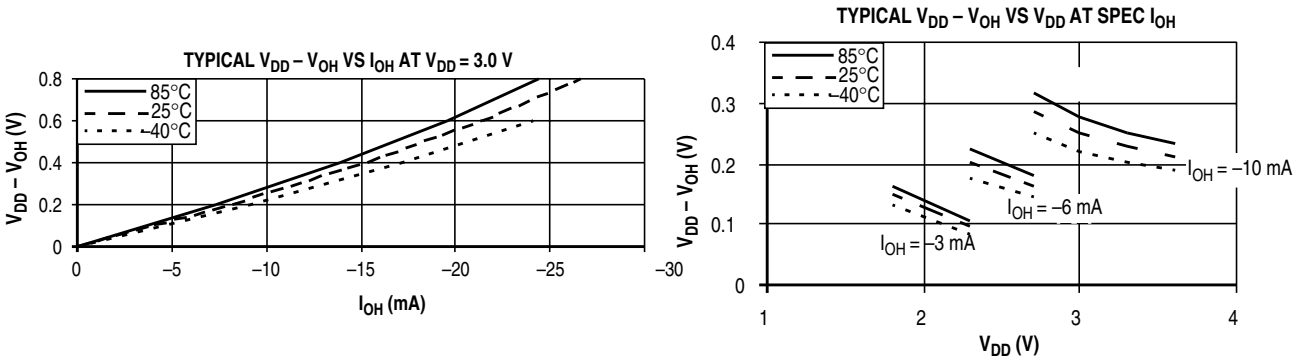



**Figure A-4. Typical High-Side Driver (Source) Characteristics (Port B and IRO)**

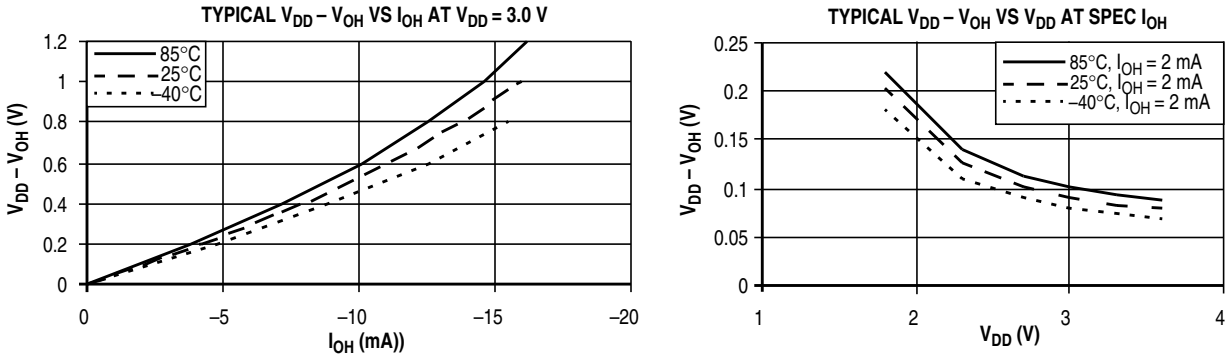



**Figure A-5. Typical High-Side (Source) Characteristics (Ports A, C, D and E)**

# A.6 Supply Current Characteristics

**Table A-6. Supply Current Characteristics**

| Parameter | Symbol | $V_{DD}$ (V)[1] | Typical[2] | Max | Temp. (°C) |
|---|---|---|---|---|---|
| Run supply current[3] measured at (CPU clock = 2 MHz, $f_{Bus}$ = 1 MHz) | $RI_{DD}$ | 3 | 500 µA | 1.525 mA<br>1.525 mA | 70<br>85 |
| | | 2 | 450 µA | 1.475 mA<br>1.475 mA | 70<br>85 |
| Run supply current[3] measured at (CPU clock = 16 MHz, $f_{Bus}$ = 8 MHz) | $RI_{DD}$ | 3 | 3.8 mA | 4.8 mA<br>4.8 mA | 70<br>85 |
| | | 2 | 2.6 mA | 3.6 mA<br>3.6 mA | 70<br>85 |
| Stop1 mode supply current | $S1I_{DD}$ | 3 | 100 nA | 350 nA<br>736 nA | 70<br>85 |
| | | 2 | 100 nA | 150 nA<br>450 nA | 70<br>85 |
| Stop2 mode supply current | $S2I_{DD}$ | 3 | 500 nA | 1.20 µA<br>1.90 µA | 70<br>85 |
| | | 2 | 500 nA | 1.00 µA<br>1.70 µA | 70<br>85 |
| Stop3 mode supply current | $S3I_{DD}$ | 3 | 600 nA | 2.65 µA<br>4.65 µA | 70<br>85 |
| | | 2 | 500 nA | 2.30 µA<br>4.30 µA | 70<br>85 |
| RTI adder from stop2 or stop3 | | 3 | 300 nA | | |
| | | 2 | 300 nA | | |
| Adder for LVD reset enabled in stop3 | | 3 | 70 µA | | |
| | | 2 | 60 µA | | |

1. 3 V values are 100% tested; 2 V values are characterized but not tested.
2. Typicals are measured at 25°C.
3. Does not include any dc loads on port pins

# A.7 Analog Comparator (ACMP) Electricals

**Table A-7. ACMP Electrical Specifications (Temp Range = -40 to 85° C Ambient)**

| Characteristic | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Analog input voltage | VAIN | $V_{SS}$ – 0.3 | — | $V_{DD}$ | V |
| Analog input offset voltage | VAIO | | — | 40 | mV |
| Analog Comparator initialization delay | $t_{AINIT}$ | | — | 1 | µs |
| Analog Comparator bandgap reference voltage | $V_{BG}$ | 1.208 | 1.218 | 1.228 | V |

# A.8 Oscillator Characteristics

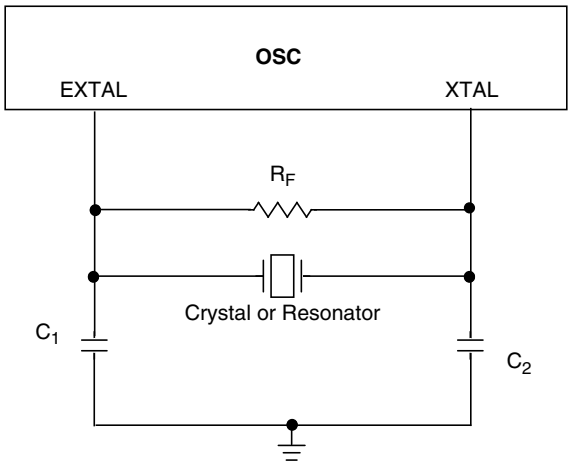



**Table A-8. OSC Electrical Specifications (Temperature Range = -40 to 85°C Ambient)**

| Characteristic | Symbol | Min | Typ(1) | Max | Unit |
|---|---|---|---|---|---|
| Frequency | $f_{OSC}$ | 1 | — | 16 | MHz |
| Load Capacitors | $C_1$<br>$C_2$ | Note(2) | | | |
| Feedback resistor | $R_F$ | | 1 | | MΩ |

1. Data in typical column was characterized at 3.0 V, 25°C or is typical recommended value.
2. See crystal or resonator manufacturer's recommendation.

# A.9 AC Characteristics

This section describes ac timing characteristics for each peripheral system.

## A.9.1 Control Timing

**Table A-9. Control Timing**

| Parameter | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Bus frequency ($t_{cyc}$ = 1/$f_{Bus}$) | $f_{Bus}$ | dc | — | 8 | MHz |
| Real time interrupt internal oscillator period | $t_{RTI}$ | 400 | | 1600 | µs |
| External reset pulse width(1) | $t_{extrst}$ | 1.5 $t_{cyc}$ | | — | ns |
| Reset low drive(2) | $t_{rstdrv}$ | 34 $t_{cyc}$ | | — | ns |
| Active background debug mode latch setup time | $t_{MSSU}$ | 25 | | — | ns |
| Active background debug mode latch hold time | $t_{MSH}$ | 25 | | — | ns |
| IRQ pulse width(3) | $t_{ILIH}$ | 1.5 $t_{cyc}$ | | — | ns |
| Port rise and fall time (load = 50 pF)(4) | $t_{Rise}$, $t_{Fall}$ | — | 3 | | ns |

1. 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.

2. When any reset is initiated, internal circuitry drives the reset pin low for about 34 cycles of $f_{Bus}$ and then samples the level on the reset pin about 38 cycles later to distinguish external reset requests from internal requests.
3. This is the minimum pulse width that is guaranteed to pass through the pin synchronization circuitry. Shorter pulses may or may not be recognized. In stop mode, the synchronizer is bypassed so shorter pulses can be recognized in that case.
4. Timing is shown with respect to 20% $V_{DD}$ and 80% $V_{DD}$ levels. Temperature range –40°C to 85°C.

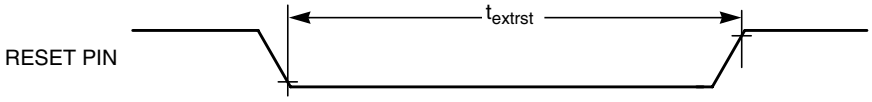



**Figure A-6. Reset Timing**

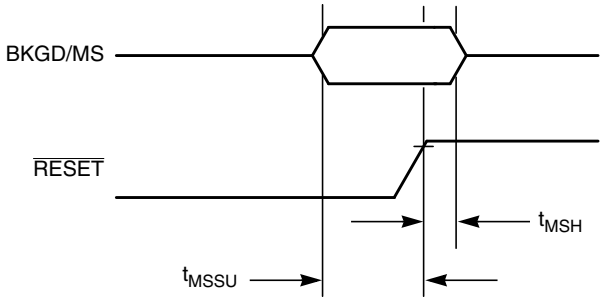



**Figure A-7. Active Background Debug Mode Latch Timing**

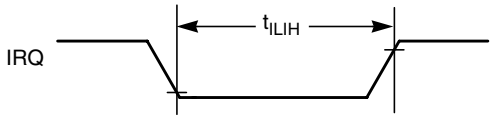



**Figure A-8. IRQ Timing**

## 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-10. TPM Input Timing**

| Function | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| External clock frequency | $f_{TPMext}$ | dc | $f_{Bus}/4$ | MHz |
| External clock period | $t_{TPMext}$ | 4 | — | $t_{cyc}$ |
| External clock high time | $t_{clkh}$ | 1.5 | — | $t_{cyc}$ |
| External clock low time | $t_{clkl}$ | 1.5 | — | $t_{cyc}$ |
| Input capture pulse width | $t_{ICPW}$ | 1.5 | — | $t_{cyc}$ |

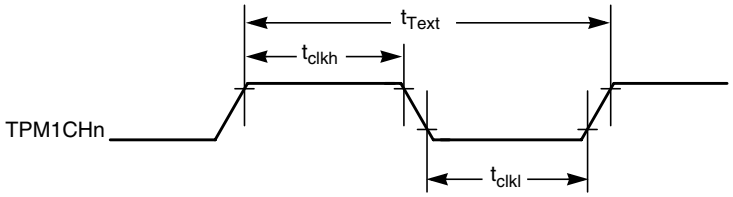



**Figure A-9. Timer External Clock**

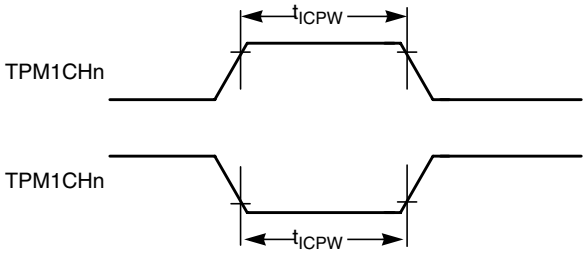



**Figure A-10. Timer Input Capture Pulse**

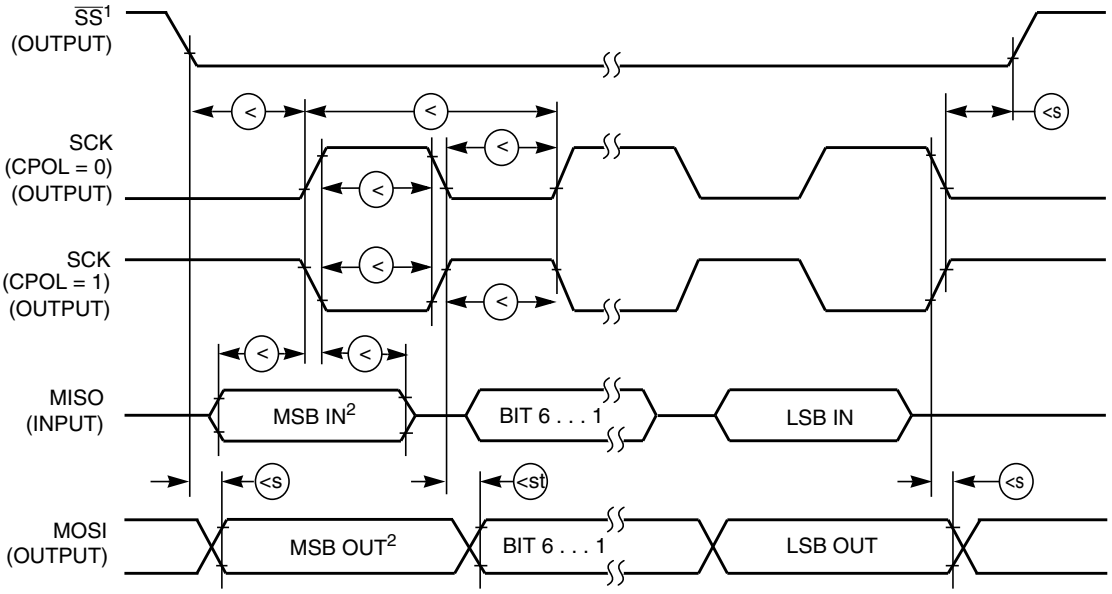
## A.9.3 SPI Timing

Table A-11 and Figure A-11 through Figure A-14 describe the timing requirements for the SPI system.

**Table A-11. SPI Electrical Characteristic**

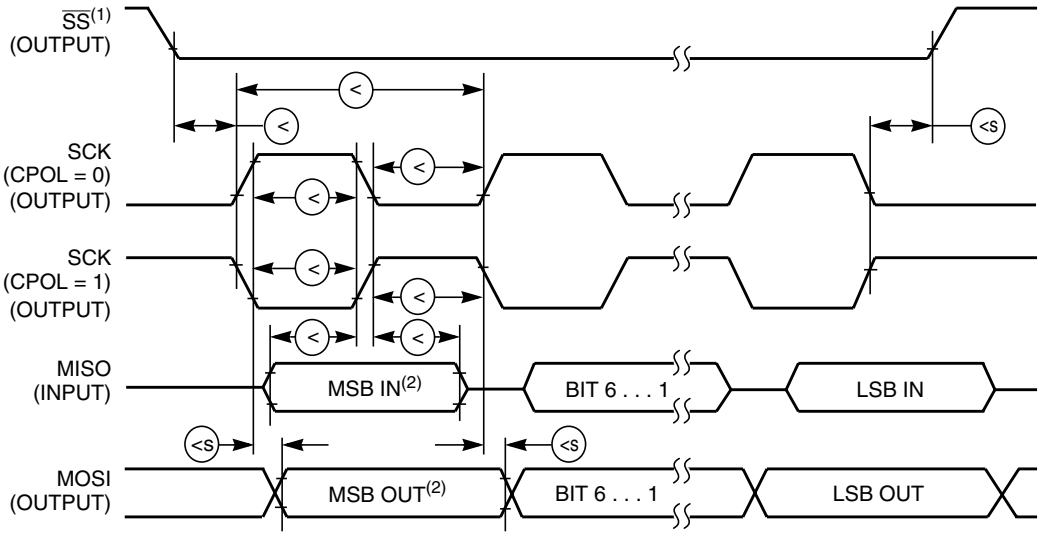
| Number[1] | Characteristic[2] | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| | Operating frequency[3]<br>Master<br>Slave | <br>$f_{op}$<br>$f_{op}$ | <br>$f_{Bus}/2048$<br>dc | <br>$f_{Bus}/2$<br>$f_{Bus}/4$ | Hz |
| 1 | Cycle time<br>Master<br>Slave | <br>$t_{SCK}$<br>$t_{SCK}$ | <br>2<br>4 | <br>2048<br>— | <br>$t_{cyc}$<br>$t_{cyc}$ |
| 2 | Enable lead time<br>Master<br>Slave | <br>$t_{Lead}$<br>$t_{Lead}$ | <br>—<br>1/2 | <br>1/2<br>— | <br>$t_{SCK}$<br>$t_{SCK}$ |
| 3 | Enable lag time<br>Master<br>Slave | <br>$t_{Lag}$<br>$t_{Lag}$ | <br>—<br>1/2 | <br>1/2<br>— | <br>$t_{SCK}$<br>$t_{SCK}$ |
| 4 | Clock (SPSCK) high time<br>Master and Slave | $t_{SCKH}$ | 1/2 $t_{SCK}$ – 25 | — | ns |
| 5 | Clock (SPSCK) low time<br>Master and Slave | $t_{SCKL}$ | 1/2 $t_{SCK}$ – 25 | — | ns |
| 6 | Data setup time (inputs)<br>Master<br>Slave | <br>$t_{SI(M)}$<br>$t_{SI(S)}$ | <br>30<br>30 | <br>—<br>— | <br>ns<br>ns |
| 7 | Data hold time (inputs)<br>Master<br>Slave | <br>$t_{HI(M)}$<br>$t_{HI(S)}$ | <br>30<br>30 | <br>—<br>— | <br>ns<br>ns |
| 8 | Access time, slave[4] | $t_A$ | 0 | 40 | ns |
| 9 | Disable time, slave[5] | $t_{dis}$ | — | 40 | ns |
| 10 | Data setup time (outputs)<br>Master<br>Slave | <br>$t_{SO}$<br>$t_{SO}$ | <br>25<br>25 | <br>—<br>— | <br>ns<br>ns |
| 11 | Data hold time (outputs)<br>Master<br>Slave | <br>$t_{HO}$<br>$t_{HO}$ | <br>–10<br>–10 | <br>—<br>— | <br>ns<br>ns |

1. Refer to Figure A-11 through Figure A-14.
2. All timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$, unless noted; 100 pF load on all SPI pins. All timing assumes slew rate control disabled and high drive strength enabled for SPI output pins.
3. Maximum baud rate must be limited to 5 MHz due to pad input characteristics.
4. Time to data active from high-impedance state.
5. Hold time to high-impedance state.



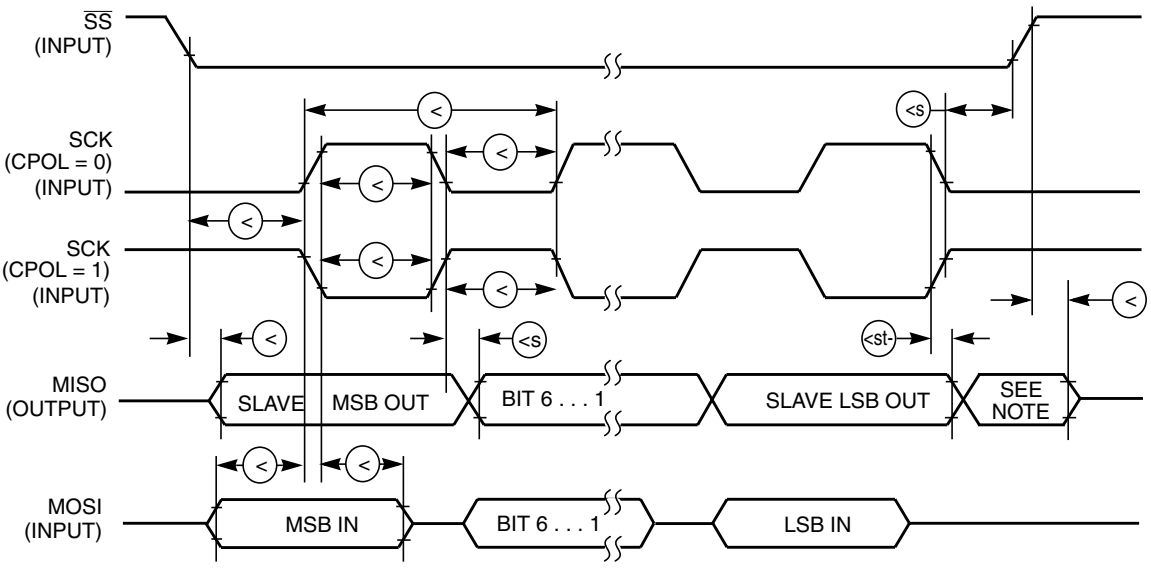

NOTES:
1. $\overline{SS}$ output mode (MODFEN = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

**Figure A-11. SPI Master Timing (CPHA = 0)**




NOTES:
1. $\overline{SS}$ output mode (MODFEN = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

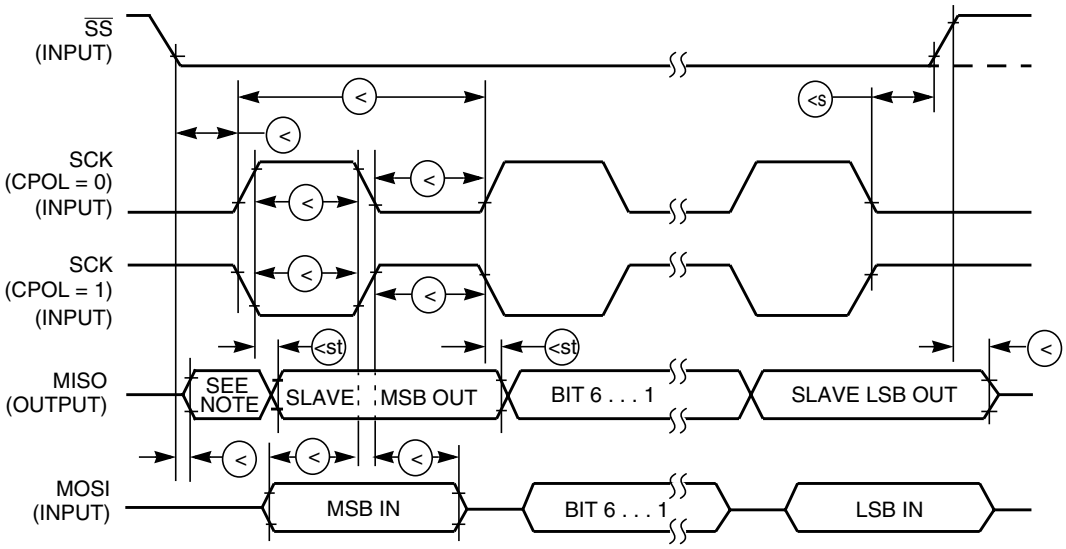
**Figure A-12. SPI Master Timing (CPHA = 1)**




NOTE:

1. Not defined but normally MSB of character just received

**Figure A-13. SPI Slave Timing (CPHA = 0)**




NOTE:

1. Not defined but normally LSB of character just received

**Figure A-14. SPI Slave Timing (CPHA = 1)**

## 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 $V_{DD}$ supply. For more detailed information about program/erase operations, see the Memory chapter.

**Table A-12. FLASH Characteristics**

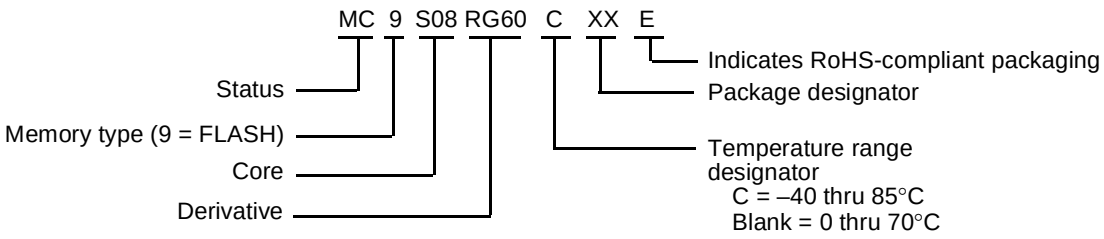
| Characteristic | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Supply voltage for program/erase | $V_{prog/erase}$ | 2.05 | | 3.6 | V |
| Supply voltage for read operation<br>$0 < f_{Bus} < 8$ MHz | $V_{Read}$ | 1.8 | | 3.6 | V |
| Internal FCLK frequency(1) | $f_{FCLK}$ | 150 | | 200 | kHz |
| Internal FCLK period (1/FCLK) | $t_{Fcyc}$ | 5 | | 6.67 | µs |
| Byte program time (random location)(2) | $t_{prog}$ | 9 | | | $t_{Fcyc}$ |
| Byte program time (burst mode)(2) | $t_{Burst}$ | 4 | | | $t_{Fcyc}$ |
| Page erase time(2) | $t_{Page}$ | 4000 | | | $t_{Fcyc}$ |
| Mass erase time(2) | $t_{Mass}$ | 20,000 | | | $t_{Fcyc}$ |
| Program/erase endurance(3)<br>$T_L$ to $T_H$ = –40°C to + 85°C<br>T = 25°C | | 10,000 | <br>100,000 | <br>— | cycles |
| Data retention(4) | $t_{D_ret}$ | 15 | 100 | — | years |

1. The frequency of this clock is controlled by a software setting.
2. 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.
3. **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/D, *Typical Endurance for Nonvolatile Memory.*
4. **Typical data retention** values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25°C using the Arrhenius equation. For additional information on how Freescale Semiconductor defines typical data retention, please refer to Engineering Bulletin EB618/D, *Typical Data Retention for Nonvolatile Memory.*

# Appendix B
# Ordering Information and Mechanical Drawings

## B.1 Ordering Information

This section contains generic ordering information for MC9S08RD/RE/RG devices and an example of the device numbering system.




See Table B-1 for package availability for each MC9S08RD/RE/RG device.

**Table B-1. Package Descriptions**

| Pin Count | Type | Abbreviation | Designator | Document No. |
|---|---|---|---|---|
| 28 | Plastic Dual In-Line Package | PDIP | P | 98ASB42390B |
| 28 | Small Outline Integrated Circuit | SOIC | DW | 98ASB42345B |
| 32 | Low Quad Flat Package | LQFP | FJ | 98ASH70029A |
| 44 | Low Quad Flat Package | LQFP | FG | 98ASB42885B |
| 48 | Quad Flat Package | QFN | FD | 98ARH99048A |

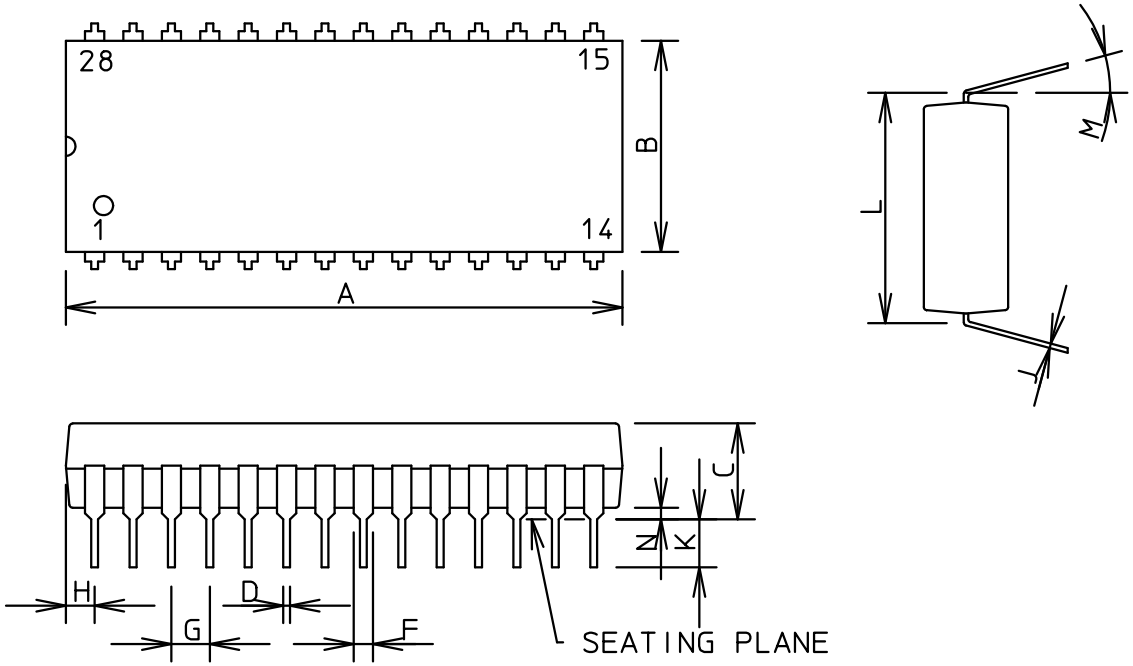
## B.2 Mechanical Drawings

The following pages are mechanical drawings for these packages:

- 28-Pin PDIP
- 28-Pin SOIC
- 32-Pin LQFP
- 44-Pin LQFP
- 48-Pin QFN

COPYRIGHT 1978-95 MOTOROLA ALL RIGHTS RESERVED

DO NOT SCALE THIS DRAWING | ISSUE B | DATE 08/18/78

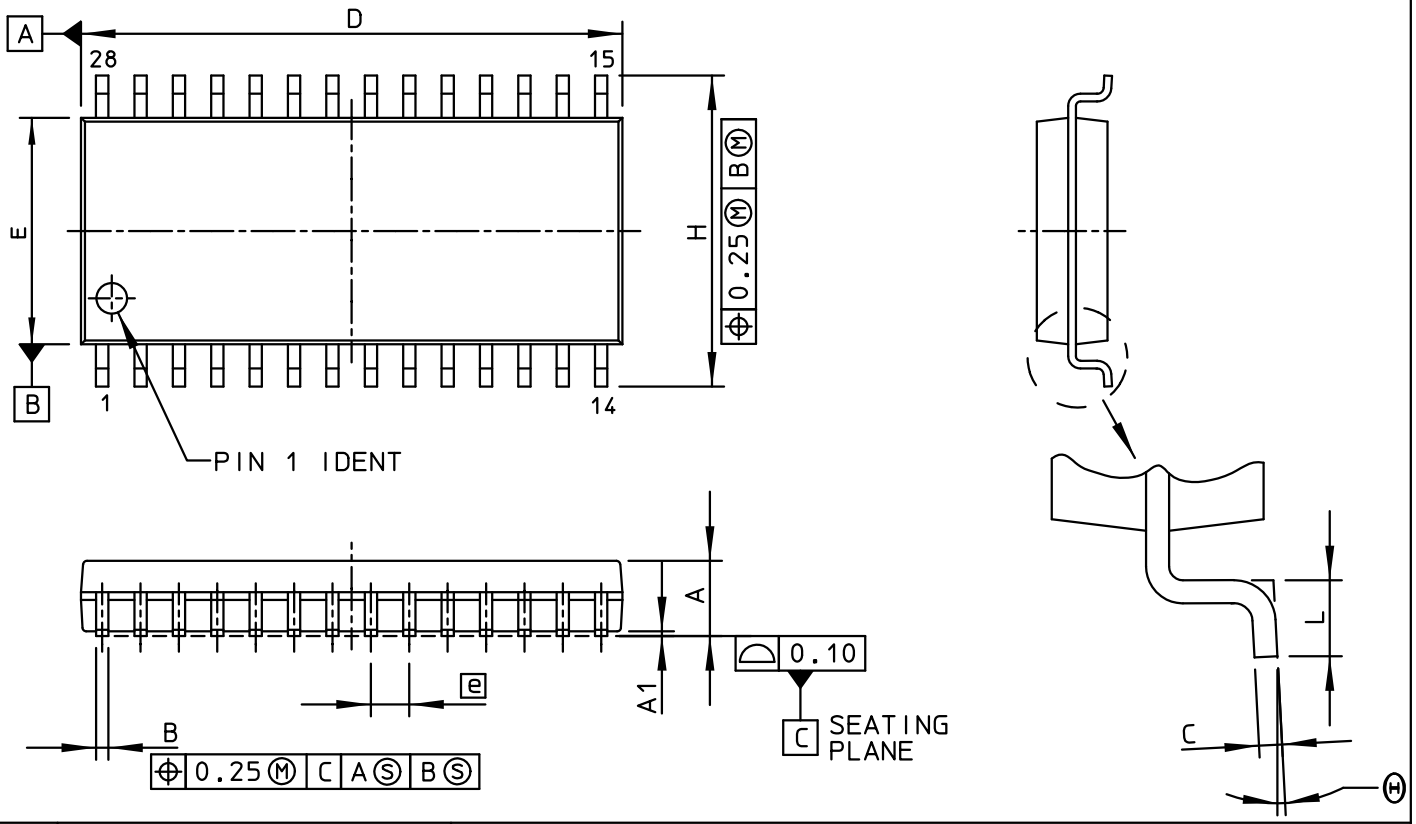



| DIM | MILLIMETERS MIN | MILLIMETERS MAX | INCHES MIN | INCHES MAX |
|---|---|---|---|---|
| A | 36.45 | 37.21 | 1.435 | 1.465 |
| B | 13.72 | 14.22 | 0.540 | 0.560 |
| C | 3.94 | 5.08 | 0.155 | 0.200 |
| D | 0.36 | 0.56 | 0.014 | 0.022 |
| F | 1.02 | 1.52 | 0.040 | 0.060 |
| G | 2.54 BSC | | 0.100BSC | |
| H | 1.65 | 2.16 | 0.065 | 0.085 |
| J | 0.20 | 0.38 | 0.008 | 0.015 |
| K | 2.92 | 3.43 | 0.115 | 0.135 |
| L | 15.24 BSC | | 0.600 BSC | |
| M | 0° | 15° | 0° | 15° |
| N | 0.51 | 1.02 | 0.020 | 0.040 |

NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.
4. 710-01 OBSOLETE, NEW STD 710-02.
5. CONTROLLING DIMENSION: INCH

COPYRIGHT 1997 MOTOROLA ALL RIGHTS RESERVED

| MECHANICAL OUTLINES DICTIONARY | 98ASB42345B |
|---|---|
| | PAGE 751F |
| DO NOT SCALE THIS DRAWING | ISSUE F \| DATE 27MAY97 |



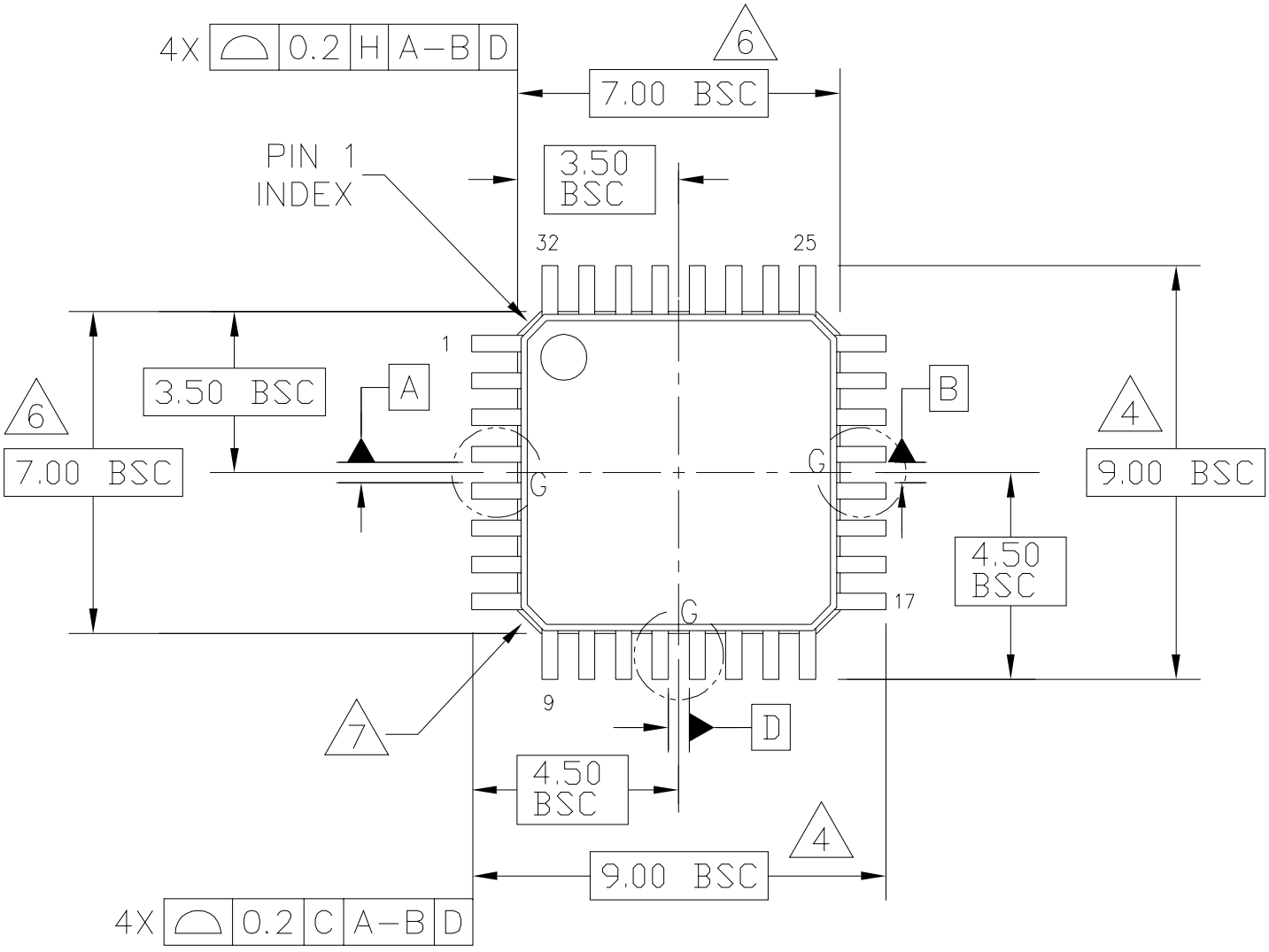

| DIM | MIN | MAX |
|---|---|---|
| A | 2.35 | 2.65 |
| A1 | 0.13 | 0.29 |
| B | 0.35 | 0.49 |
| C | 0.23 | 0.32 |
| D | 17.80 | 18.05 |
| E | 7.40 | 7.60 |
| e | 1.27 BSC | |
| H | 10.05 | 10.55 |
| L | 0.41 | 0.90 |

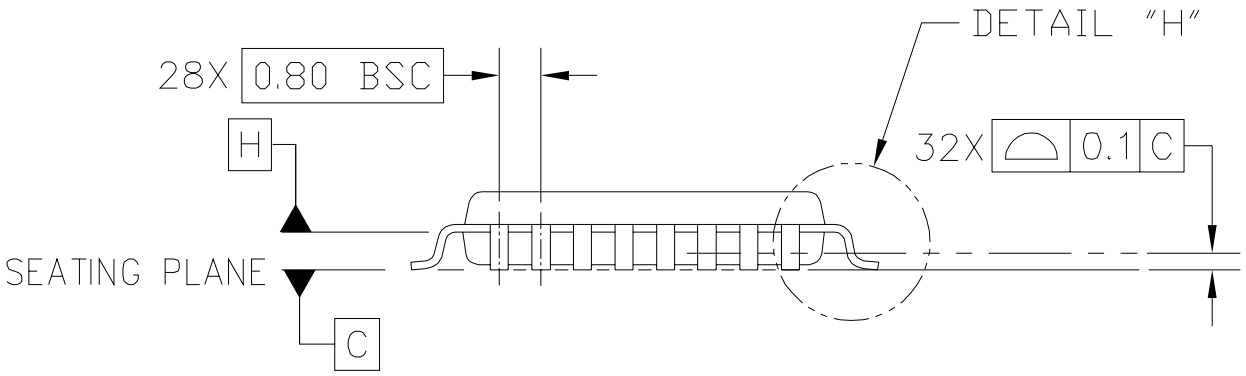
NOTES

1. DIMENSIONS ARE IN MILLIMETERS.
2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5M-1994.
3. DIMENSIONS "D" AND "E" DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 PER SIDE.
5. 751F-01 THRU -04 OBSOLETE.
   NEW STANDARD: 751F-05
6. DIMENSION "B" DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.13 TOTAL IN EXCESS OF "B" DIMENSION AT MAXIMUM MATERIAL CONDITION.

freescale semiconductor
© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.






TITLE:

LOW PROFILE QUAD FLAT PACK (LQFP)
32 LEAD, 0.8 PITCH (7 X 7 X 1.4)

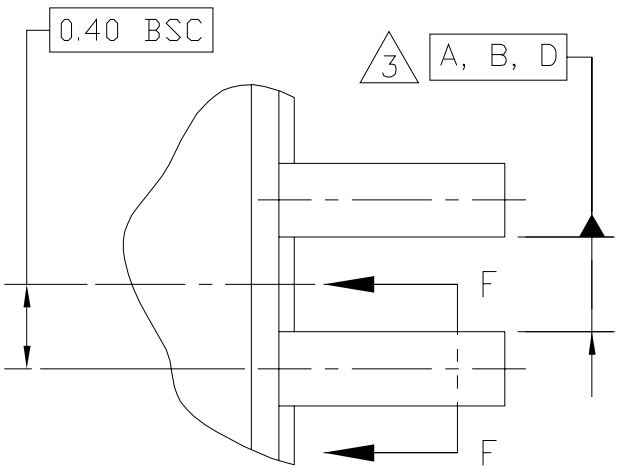
CASE NUMBER: 873A-04

STANDARD: JEDEC MS-026 BBA

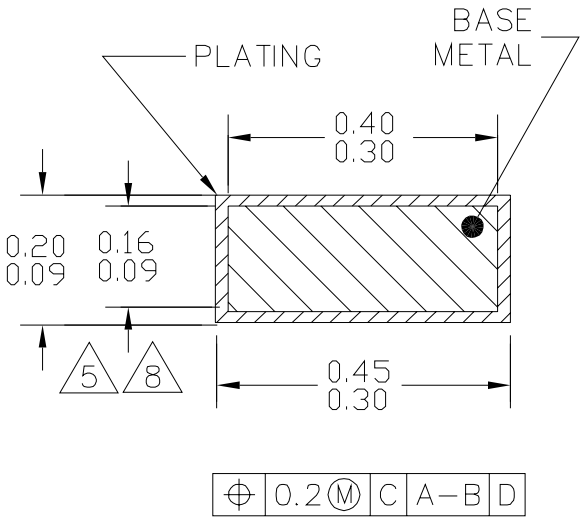
PACKAGE CODE: 6300

freescale semiconductor
© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.

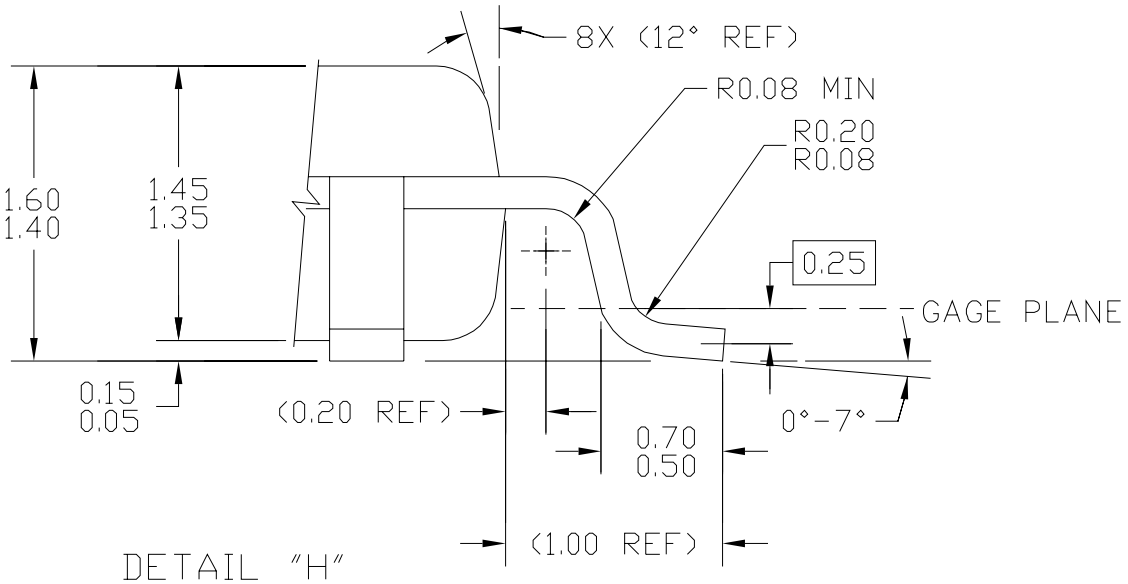





DETAIL G




SECTION F–F
ROTATED 90°CW
32 PLACES




DETAIL "H"

TITLE:
LOW PROFILE QUAD FLAT PACK (LQFP)
32 LEAD, 0.8 PITCH (7 X 7 X 1.4)

CASE NUMBER: 873A-04
STANDARD: JEDEC MS-026 BBA
PACKAGE CODE: 6300

freescale semiconductor
© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.



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.

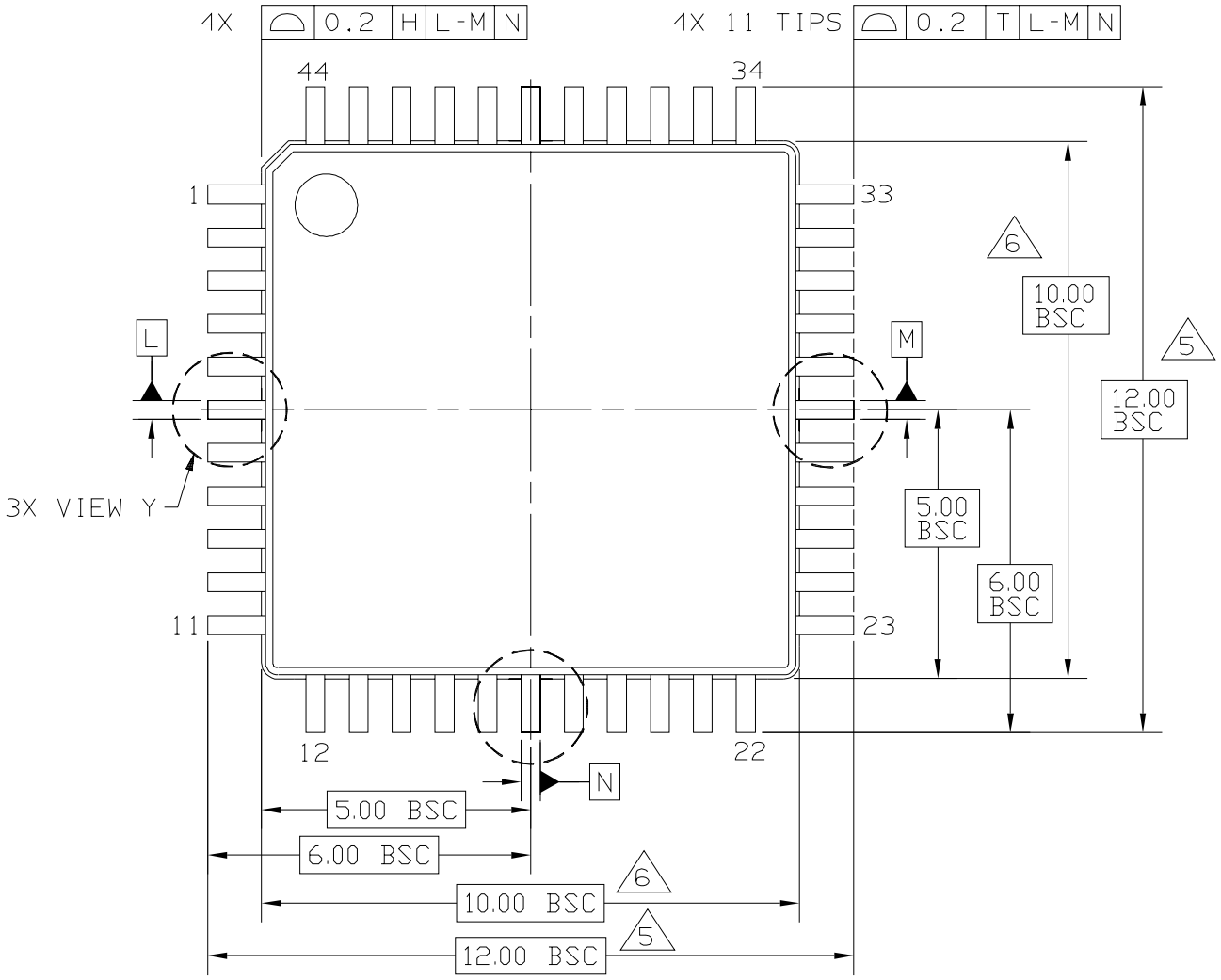
6. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 MM PER SIDE. DIMENSIONS ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.

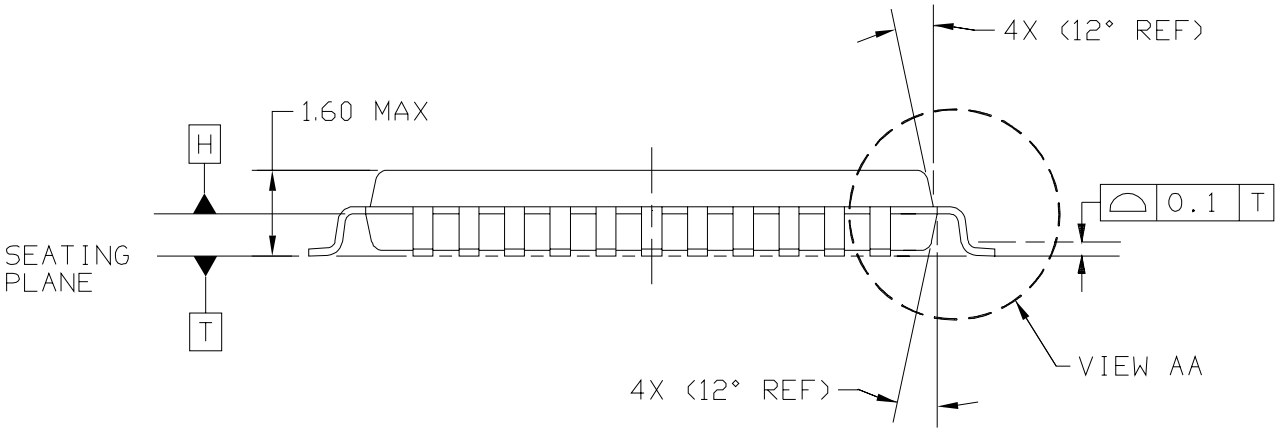
7. EXACT SHAPE OF EACH CORNER IS OPTIONAL.

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
© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.

| MECHANICAL OUTLINES DICTIONARY | DOCUMENT NO: 98ASS23225W |
|---|---|
| | PAGE: 824D |
| DO NOT SCALE THIS DRAWING | REV: B |

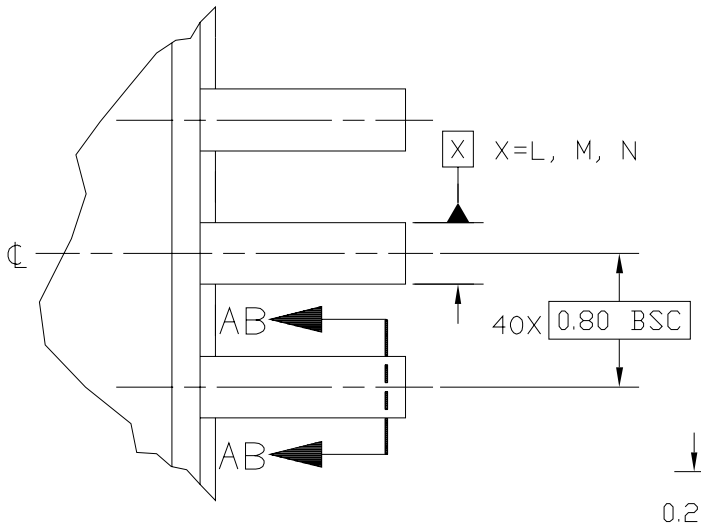


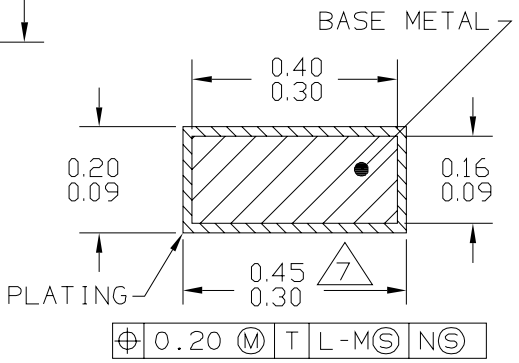

| TITLE: | CASE NUMBER: 824D-03 | |
|---|---|---|
| 44 LD TQFP, 10 X 10 PKG, 0.8 PITCH, 1.4 THICK | STANDARD: JEDEC MS-026-BCB | |
| | PACKAGE CODE: 8256 | SHEET: 1 OF 4 |

freescale semiconductor
© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.

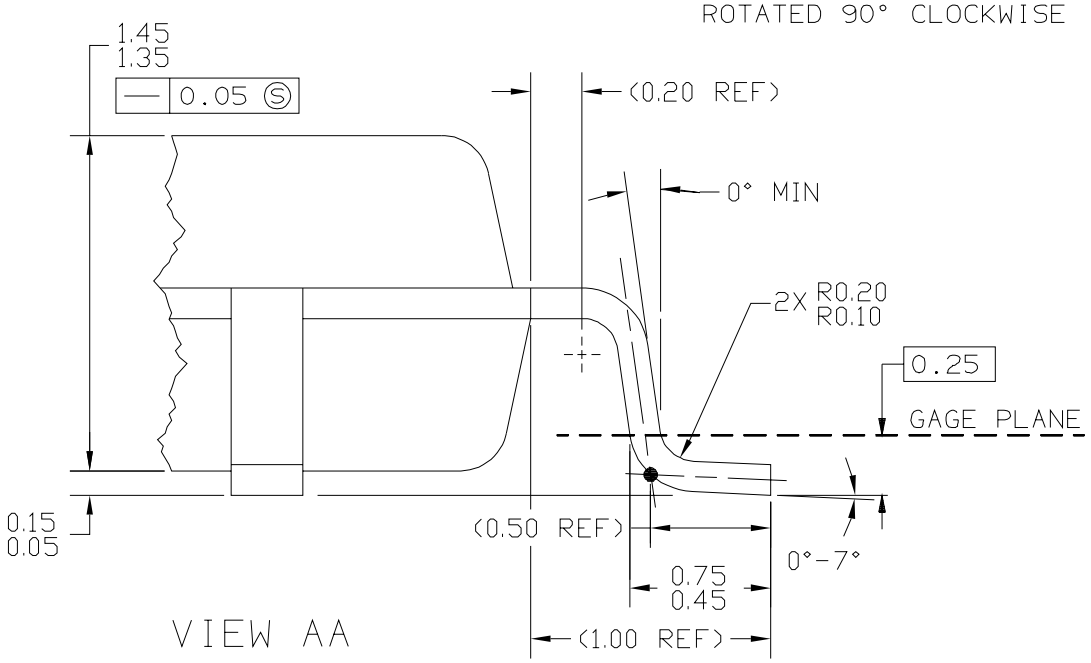





VIEW Y




SECTION AB-AB
ROTATED 90° CLOCKWISE




VIEW AA

TITLE:

44 LD TQFP,
10 X 10 PKG, 0.8 PITCH, 1.4 THICK

CASE NUMBER: 824D-03

STANDARD: JEDEC MS-026 BCB

PACKAGE CODE: 8256

freescale semiconductor
© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.



NOTES

1. DIMENSIONS AND TOLERANCING PER ANSI Y14.5M, 1982.

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- .

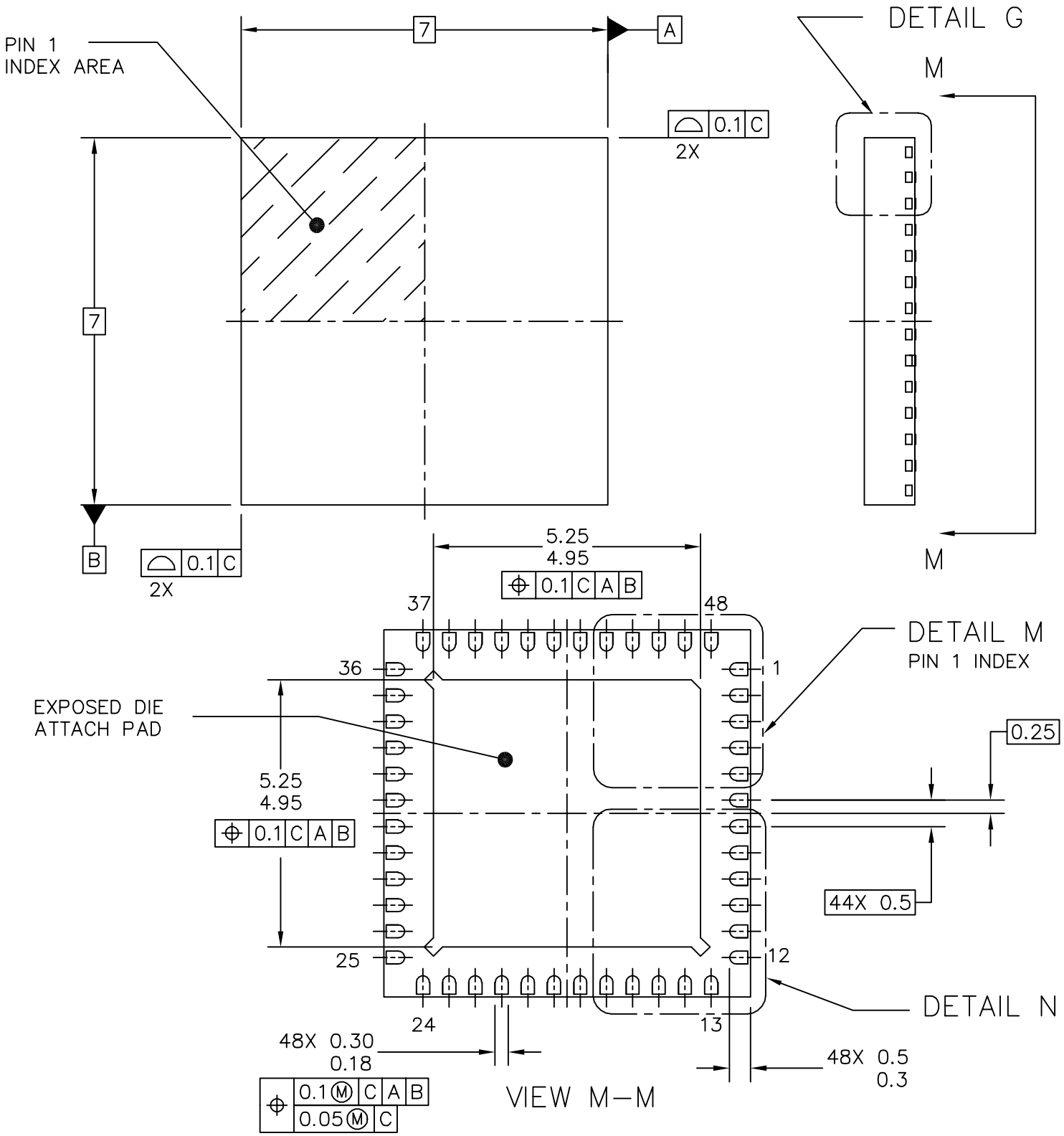
5. DIMENSIONS TO BE DETERMINED AT SEATING PLANE -T- .

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
© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.

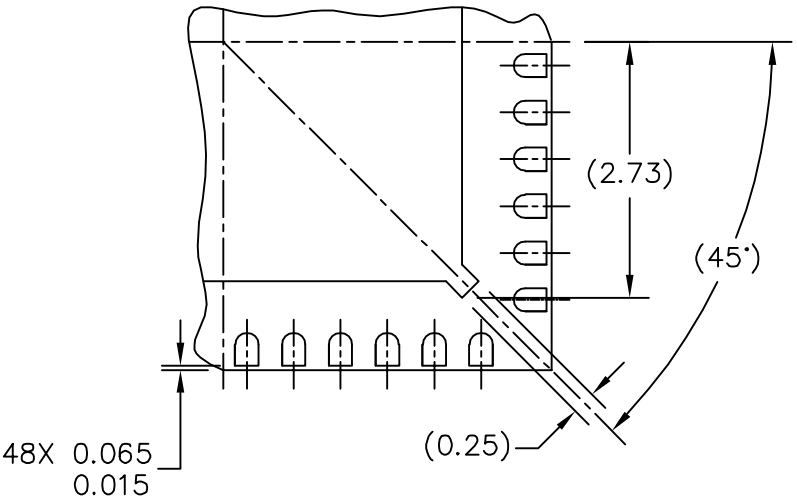
| MECHANICAL OUTLINES DICTIONARY | DOCUMENT NO: 98ARH99048A |
|---|---|
| | PAGE: 1314 |
| DO NOT SCALE THIS DRAWING | REV: D |



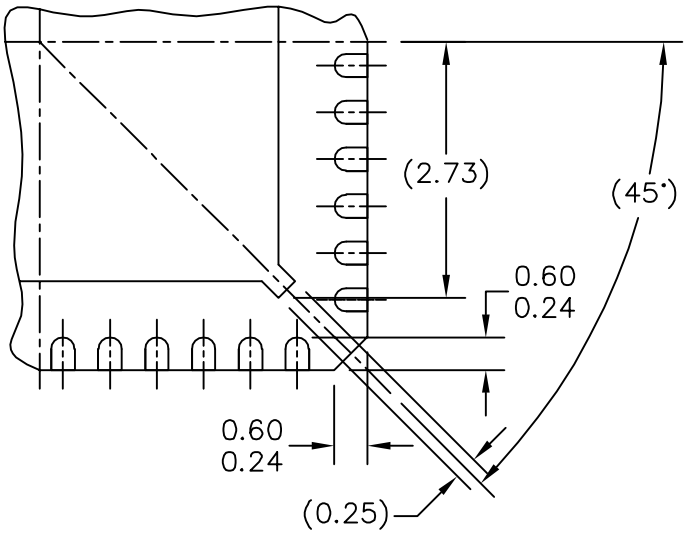

| TITLE: THERMALLY ENHANCED QUAD FLAT NON–LEADED PACKAGE (QFN) 48 TERMINAL, 0.5 PITCH (7 X 7 X 1) | CASE NUMBER: 1314–03 | |
|---|---|---|
| | STANDARD: JEDEC–MO–220 VKKD–2 | |
| | PACKAGE CODE: 6152 | SHEET: 1 OF 6 |

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.

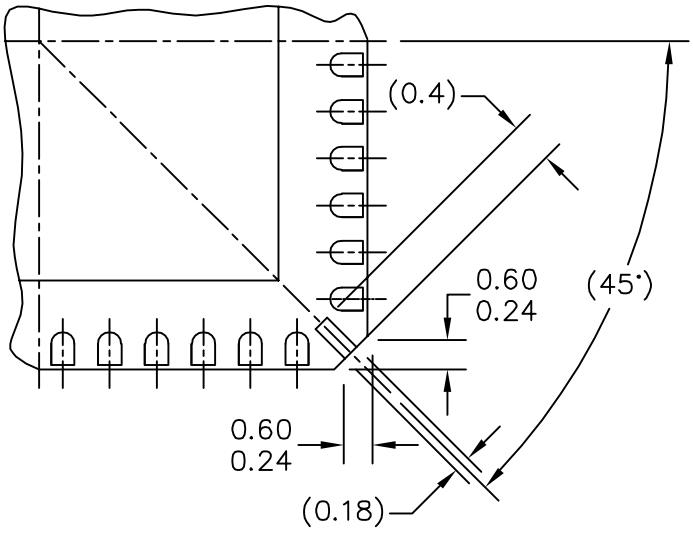





DETAIL N
PREFERRED CORNER CONFIGURATION
4




DETAIL N
CORNER CONFIGURATION OPTION1
4 5



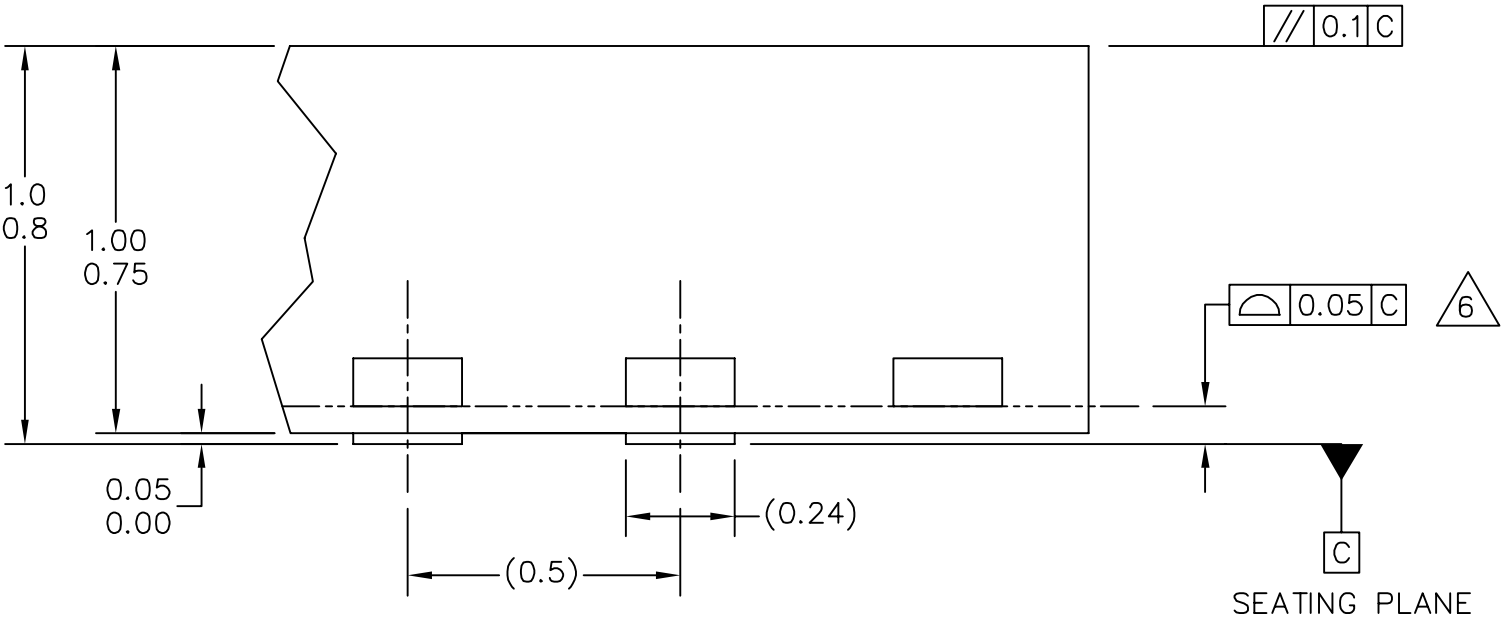

DETAIL N
CORNER CONFIGURATION OPTION2
4 5

TITLE: THERMALLY ENHANCED QUAD FLAT NON-LEADED PACKAGE (QFN) 48 TERMINAL, 0.5 PITCH (7 X 7 X 1)

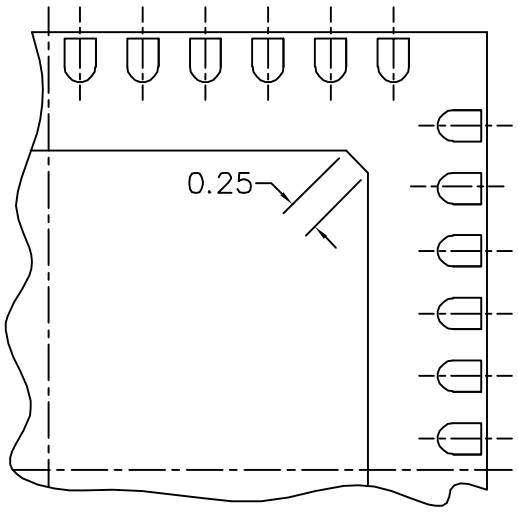
CASE NUMBER: 1314-03
STANDARD: JEDEC-MO-220 VKKD-2
PACKAGE CODE: 6152
SHEET: 2 OF 6

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.

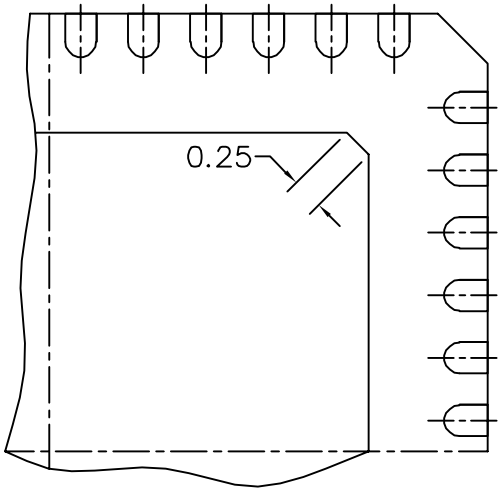





DETAIL G
VIEW ROTATED 90° CW




DETAIL M
PREFERED PIN 1 BACKSIDE IDENTIFIER



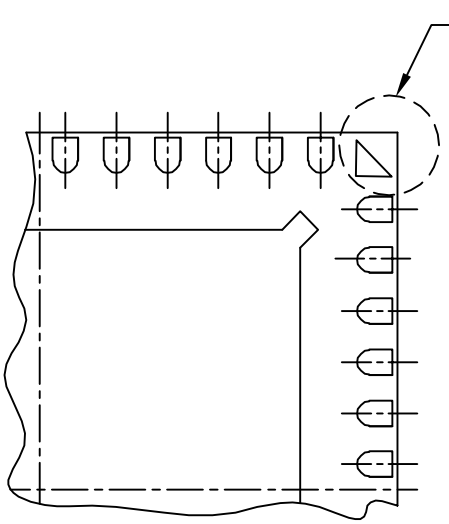

DETAIL M
PIN 1 BACKSIDE IDENTIFIER OPTION1

TITLE: THERMALLY ENHANCED QUAD FLAT NON-LEADED PACKAGE (QFN) 48 TERMINAL, 0.5 PITCH (7 X 7 X 1)

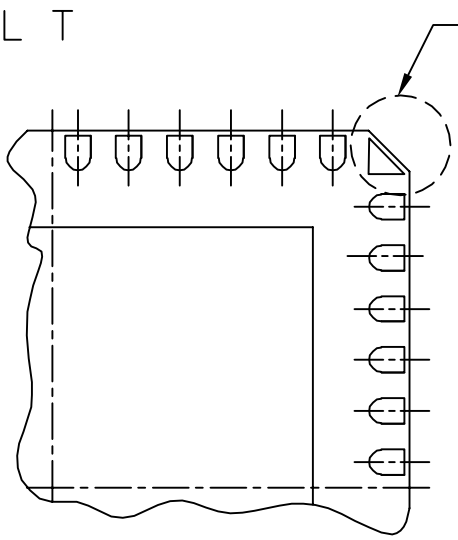
CASE NUMBER: 1314-03
STANDARD: JEDEC-MO-220 VKKD-2
PACKAGE CODE: 6152
SHEET: 3 OF 6

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.

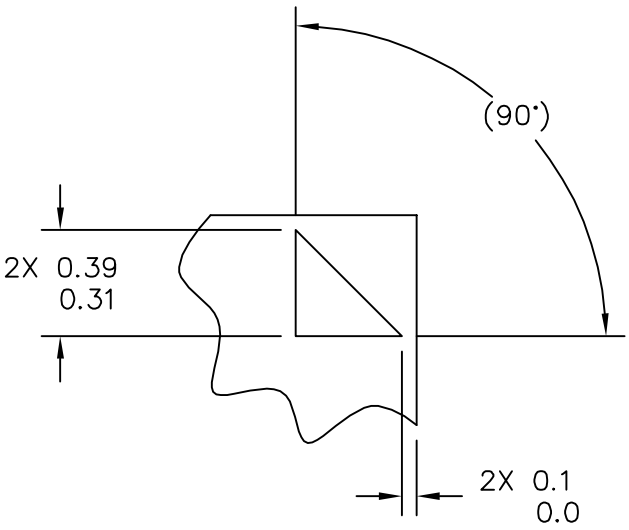




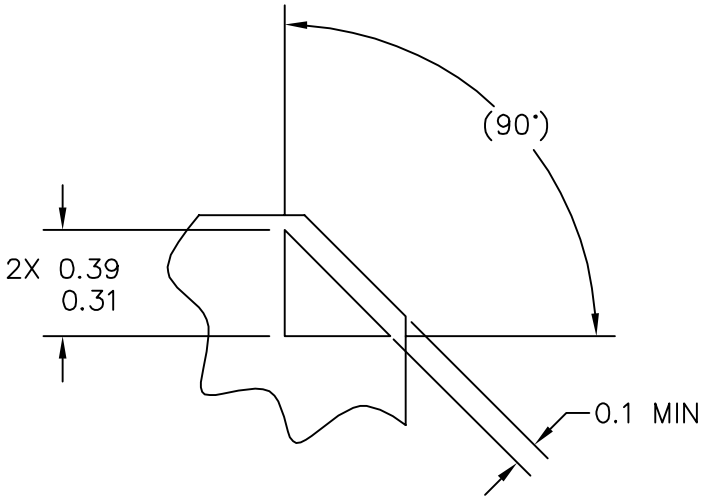

DETAIL M
PIN 1 BACKSIDE IDENTIFIER OPTION2




DETAIL M
PIN 1 BACKSIDE IDENTIFIER OPTION3




DETAIL T
PIN 1 BACKSIDE IDENTIFIER OPTION2




DETAIL S
PIN 1 BACKSIDE IDENTIFIER OPTION3

TITLE: THERMALLY ENHANCED QUAD FLAT NON-LEADED PACKAGE (QFN) 48 TERMINAL, 0.5 PITCH (7 X 7 X 1)

CASE NUMBER: 1314-03

STANDARD: JEDEC-MO-220 VKKD-2

PACKAGE CODE: 6152

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.
ELECTRONIC VERSIONS ARE UNCONTROLLED EXCEPT WHEN ACCESSED DIRECTLY FROM THE DOCUMENT CONTROL REPOSITORY. PRINTED VERSIONS ARE UNCONTROLLED EXCEPT WHEN STAMPED "CONTROLLED COPY" IN RED.



NOTES:

1. DIMENSIONS ARE IN MILLIMETERS.
2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5M–1994.
3. THE COMPLETE JEDEC DESIGNATOR FOR THIS PACKAGE IS: HF–PQFP–N.
4. CORNER CHAMFER MAY NOT BE PRESENT. DIMENSIONS OF OPTIONAL FEATURES ARE FOR REFERENCE ONLY.
5. CORNER LEADS CAN BE USED FOR THERMAL OR GROUND AND ARE TIED TO THE DIE ATTACH PAD. THESE LEADS ARE NOT INCLUDED IN THE LEAD COUNT.
6. COPLANARITY APPLIES TO LEADS, CORNER LEADS, AND DIE ATTACH PAD.
7. FOR ANVIL SINGULATED QFN PACKAGES, MAXIMUM DRAFT ANGLE IS 12°.
8. MIN METAL GAP SHOULD BE 0.2MM.

## *How to Reach Us:*

**USA/Europe/Locations not listed:**
Freescale Semiconductor Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

**Japan:**
Freescale Semiconductor Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu
Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

**Asia/Pacific:**
Freescale Semiconductor H.K. Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

*Learn More:*
For more information about Freescale Semiconductor products, please visit
**http://www.freescale.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 which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.
© Freescale Semiconductor, Inc. 2004. All rights reserved.

MC9S08RG60/D
Rev. 1.11

freescale™
semiconductor

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View MC9S08RD32DWE on WIN SOURCE
- Freescale Semiconductor - NXP Information

## Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management