# THE DATASHEET OF

# MC9S08QG8CFFE

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 |

# MC9S08QG8
# MC9S08QG4

Data Sheet

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

MC9S08QG8
Rev. 5
11/2009

freescale.com

freescale™
semiconductor

# MC9S08QG8/4 Features

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

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

## Memory Options

- FLASH read/program/erase over full operating voltage and temperature
- MC9S08QG8 — 8 Kbytes FLASH, 512 bytes RAM MC9S08QG4 — 4 Kbytes FLASH, 256 bytes RAM

## Power-Saving Modes

- Wait plus three stops

## Clock Source Options

- **ICS** — Internal clock source module containing a frequency-locked-loop (FLL) controlled by internal or external reference; precision trimming of internal reference allows 0.2% resolution and 2% deviation over temperature and voltage; supports bus frequencies from 1 MHz to 10 MHz
- **XOSC** — Low-power oscillator module with software selectable crystal or ceramic resonator range, 31.25 kHz to 38.4 kHz or 1 MHz to 16 MHz, and supports external clock source input up to 20 MHz

## System Protection

- Watchdog computer operating properly (COP) reset with option to run from dedicated 1-kHz internal clock source or bus clock
- Low-voltage detection with reset or interrupt
- Illegal opcode detection with reset
- Illegal address detection with reset
- FLASH block protect

## Peripherals

- **ADC** — 8-channel, 10-bit analog-to-digital converter with automatic compare function, asynchronous clock source, temperature sensor, and internal bandgap reference channel; ADC is hardware triggerable using the RTI counter
- **ACMP** — Analog comparator module with option to compare to internal reference; output can be optionally routed to TPM module
- **SCI** — Serial communications interface module with option for 13-bit break capabilities
- **SPI** — Serial peripheral interface module
- **IIC** — Inter-integrated circuit bus module
- **TPM**— 2-channel timer/pulse-width modulator; each channel can be used for input capture, output compare, buffered edge-aligned PWM, or buffered center-aligned PWM
- **MTIM** — 8-bit modulo timer module with 8-bit prescaler
- **KBI** — 8-pin keyboard interrupt module with software selectable polarity on edge or edge/level modes

## Input/Output

- 12 general-purpose input/output (I/O) pins, one input-only pin and one output-only pin; outputs 10 mA each, 60 mA max for package
- Software selectable pullups on ports when used as input
- Software selectable slew rate control and drive strength on ports when used as output
- Internal pullup on $\overline{\text{RESET}}$ and $\overline{\text{IRQ}}$ pins to reduce customer system cost

## Development Support

- Single-wire background debug interface
- On-chip, in-circuit emulation (ICE) with real-time bus capture

## Package Options

- 24-pin quad flat no lead (QFN) package
- 16-pin plastic dual in-line package (PDIP) — MC9S08QG8 only
- 16-pin quad flat no lead (QFN) package
- 16-pin thin shrink small outline package (TSSOP)
- 8-pin dual flat no lead (DFN) package
- 8-pin PDIP — *MC9S08QG4 only*
- 8-pin narrow body small outline integrated circuit (SOIC) package

# MC9S08QG8 Data Sheet

Covers MC9S08QG8
MC9S08QG4

MC9S08QG8
Rev. 5
11/2009

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

© Freescale Semiconductor, Inc., 2007-2009. All rights reserved.

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.

| Rev No. | Revision Date | Description of Changes |
|---|---|---|
| 2 Draft A | 06/08/2006 | Previous version was 1.01; revision numbering will increment by integers from now on.<br><br>Clarified PTA5 pullup behavior note; clarified that FCDIV is write once after reset; expanded FPROT/NVPROT register description added note for servicing the COP if the COP is enabled during an erase function; added requirements for using ACMP0 in ACMP introduction; added factory trim value section to ICS introduction; debug section added to Development Support chapter; updated RTI period and added RTI graph to control timing section; other minor grammar edits. |
| 3 | 10/2007 | Added 24-pin QFN package and updated the A-5. DC Characteristics table Supply Voltage row. |
| 4 | 2/2008 | Incorporated core team markups from shared review. See Project Sync issue #3313 for archive. |
| 5 | 11/2009 | Added new part number information for the maskset revision 4.<br>Corrected bit 0 of KBISC register in the Table 4-2. |

© Freescale Semiconductor, Inc., 2007-2008. All rights reserved.

This product incorporates SuperFlash® Technology licensed from SST.

PRELIMINARY

# List of Chapters


| Chapter | Title | Page |
|---|---|---|
| Chapter 1 | Device Overview | 19 |
| Chapter 2 | External Signal Description | 23 |
| Chapter 3 | Modes of Operation | 33 |
| Chapter 4 | Memory Map and Register Definition | 39 |
| Chapter 5 | Resets, Interrupts, and General System Control | 59 |
| Chapter 6 | Parallel Input/Output Control | 77 |
| Chapter 7 | Central Processor Unit (S08CPUV2) | 87 |
| Chapter 8 | Analog Comparator (S08ACMPV2) | 107 |
| Chapter 9 | Analog-to-Digital Converter (S08ADC10V1) | 115 |
| Chapter 10 | Internal Clock Source (S08ICSV1) | 143 |
| Chapter 11 | Inter-Integrated Circuit (S08IICV1) | 155 |
| Chapter 12 | Keyboard Interrupt (S08KBIV2) | 173 |
| Chapter 13 | Modulo Timer (S08MTIMV1) | 181 |
| Chapter 14 | Serial Communications Interface (S08SCIV3) | 191 |
| Chapter 15 | Serial Peripheral Interface (S08SPIV3) | 211 |
| Chapter 16 | Timer/Pulse-Width Modulator (S08TPMV2) | 227 |
| Chapter 17 | Development Support | 243 |
| Appendix A | Electrical Characteristics | 265 |
| Appendix B | Ordering Information and Mechanical Drawings | 289 |

# Contents

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

## Chapter 1
## Device Overview


1.1 Introduction ..... 19
1.1.1 Devices in the MC9S08QG8/4 Series ..... 19
1.1.2 MCU Block Diagram ..... 20


## Chapter 2
## External Signal Description


2.1 Device Pin Assignment ..... 23
2.2 Recommended System Connections ..... 25
2.2.1 Power ..... 26
2.2.2 Oscillator (XOSC) ..... 27
2.2.3 Reset (Input Only) ..... 27
2.2.4 Background / Mode Select (BKGD/MS) ..... 28
2.2.5 General-Purpose I/O and Peripheral Ports ..... 28


## Chapter 3
## Modes of Operation


3.1 Introduction ..... 33
3.2 Features ..... 33
3.3 Run Mode ..... 33
3.4 Active Background Mode ..... 33
3.5 Wait Mode ..... 34
3.6 Stop Modes ..... 35
3.6.1 Stop3 Mode ..... 35
3.6.2 Stop2 Mode ..... 36
3.6.3 Stop1 Mode ..... 37
3.6.4 On-Chip Peripheral Modules in Stop Modes ..... 37


## Chapter 4
## Memory Map and Register Definition


4.1 MC9S08QG8/4 Memory Map ..... 39
4.2 Reset and Interrupt Vector Assignments ..... 40
4.3 Register Addresses and Bit Assignments ..... 41
4.4 RAM ..... 45
4.5 FLASH ..... 46
4.5.1 Features ..... 47

4.5.2 Program and Erase Times ..... 47
4.5.3 Program and Erase Command Execution ..... 48
4.5.4 Burst Program Execution ..... 49
4.5.5 Access Errors ..... 51
4.5.6 FLASH Block Protection ..... 51
4.5.7 Vector Redirection ..... 52
4.6 Security ..... 52
4.7 FLASH Registers and Control Bits ..... 54
4.7.1 FLASH Clock Divider Register (FCDIV) ..... 54
4.7.2 FLASH Options Register (FOPT and NVOPT) ..... 55
4.7.3 FLASH Configuration Register (FCNFG) ..... 56
4.7.4 FLASH Protection Register (FPROT and NVPROT) ..... 56
4.7.5 FLASH Status Register (FSTAT) ..... 57
4.7.6 FLASH Command Register (FCMD) ..... 58

## Chapter 5
## Resets, Interrupts, and General System Control

5.1 Introduction ..... 59
5.2 Features ..... 59
5.3 MCU Reset ..... 59
5.4 Computer Operating Properly (COP) Watchdog ..... 60
5.5 Interrupts ..... 61
5.5.1 Interrupt Stack Frame ..... 62
5.5.2 External Interrupt Request Pin ($\overline{\text{IRQ}}$) ..... 62
5.5.3 Interrupt Vectors, Sources, and Local Masks ..... 63
5.6 Low-Voltage Detect (LVD) System ..... 65
5.6.1 Power-On Reset Operation ..... 65
5.6.2 LVD Reset Operation ..... 65
5.6.3 LVD Interrupt Operation ..... 65
5.6.4 Low-Voltage Warning (LVW) ..... 65
5.7 Real-Time Interrupt (RTI) ..... 65
5.8 Reset, Interrupt, and System Control Registers and Control Bits ..... 66
5.8.1 Interrupt Pin Request Status and Control Register (IRQSC) ..... 67
5.8.2 System Reset Status Register (SRS) ..... 68
5.8.3 System Background Debug Force Reset Register (SBDFR) ..... 69
5.8.4 System Options Register 1 (SOPT1) ..... 70
5.8.5 System Options Register 2 (SOPT2) ..... 71
5.8.6 System Device Identification Register (SDIDH, SDIDL) ..... 72
5.8.7 System Real-Time Interrupt Status and Control Register (SRTISC) ..... 73
5.8.8 System Power Management Status and Control 1 Register (SPMSC1) ..... 74
5.8.9 System Power Management Status and Control 2 Register (SPMSC2) ..... 75
5.8.10 System Power Management Status and Control 3 Register (SPMSC3) ..... 76

## Chapter 6
## Parallel Input/Output Control

6.1 Port Data and Data Direction ..... 77
6.2 Pin Control — Pullup, Slew Rate, and Drive Strength ..... 78
6.3 Pin Behavior in Stop Modes ..... 79
6.4 Parallel I/O Registers ..... 79
6.4.1 Port A Registers ..... 79
6.4.2 Port A Control Registers ..... 80
6.4.3 Port B Registers ..... 83
6.4.4 Port B Control Registers ..... 84

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

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

## Chapter 8
## Analog Comparator (S08ACMPV2)

8.1 Introduction ..... 107
8.1.1 ACMP Configuration Information ..... 107
8.1.2 ACMP/TPM Configuration Information ..... 107
8.1.3 Features ..... 109

8.1.4 Modes of Operation ..... 109
8.1.5 Block Diagram ..... 109
8.2 External Signal Description ..... 111
8.3 Register Definition ..... 111
8.3.1 ACMP Status and Control Register (ACMPSC) ..... 112
8.4 Functional Description ..... 113

## Chapter 9
## Analog-to-Digital Converter (S08ADC10V1)

9.1 Introduction ..... 115
9.1.1 Module Configurations ..... 117
9.1.2 Features ..... 119
9.1.3 Block Diagram ..... 119
9.2 External Signal Description ..... 120
9.2.1 Analog Power ($V_{DDAD}$) ..... 121
9.2.2 Analog Ground ($V_{SSAD}$) ..... 121
9.2.3 Voltage Reference High ($V_{REFH}$) ..... 121
9.2.4 Voltage Reference Low ($V_{REFL}$) ..... 121
9.2.5 Analog Channel Inputs (ADx) ..... 121
9.3 Register Definition ..... 121
9.3.1 Status and Control Register 1 (ADCSC1) ..... 121
9.3.2 Status and Control Register 2 (ADCSC2) ..... 123
9.3.3 Data Result High Register (ADCRH) ..... 124
9.3.4 Data Result Low Register (ADCRL) ..... 124
9.3.5 Compare Value High Register (ADCCVH) ..... 125
9.3.6 Compare Value Low Register (ADCCVL) ..... 125
9.3.7 Configuration Register (ADCCFG) ..... 125
9.3.8 Pin Control 1 Register (APCTL1) ..... 127
9.3.9 Pin Control 2 Register (APCTL2) ..... 128
9.3.10 Pin Control 3 Register (APCTL3) ..... 129
9.4 Functional Description ..... 130
9.4.1 Clock Select and Divide Control ..... 130
9.4.2 Input Select and Pin Control ..... 131
9.4.3 Hardware Trigger ..... 131
9.4.4 Conversion Control ..... 131
9.4.5 Automatic Compare Function ..... 134
9.4.6 MCU Wait Mode Operation ..... 134
9.4.7 MCU Stop3 Mode Operation ..... 134
9.4.8 MCU Stop1 and Stop2 Mode Operation ..... 135
9.5 Initialization Information ..... 135
9.5.1 ADC Module Initialization Example ..... 135
9.6 Application Information ..... 137

9.6.1 External Pins and Routing ..... 137
9.6.2 Sources of Error ..... 139

## Chapter 10
## Internal Clock Source (S08ICSV1)

10.1 Introduction ..... 143
10.1.1 Module Configuration ..... 143
10.1.2 Factory Trim Value ..... 143
10.1.3 Features ..... 145
10.1.4 Modes of Operation ..... 145
10.1.5 Block Diagram ..... 146
10.2 External Signal Description ..... 147
10.3 Register Definition ..... 147
10.3.1 ICS Control Register 1 (ICSC1) ..... 147
10.3.2 ICS Control Register 2 (ICSC2) ..... 148
10.3.3 ICS Trim Register (ICSTRM) ..... 149
10.3.4 ICS Status and Control (ICSSC) ..... 149
10.4 Functional Description ..... 150
10.4.1 Operational Modes ..... 150
10.4.2 Mode Switching ..... 152
10.4.3 Bus Frequency Divider ..... 152
10.4.4 Low Power Bit Usage ..... 153
10.4.5 Internal Reference Clock ..... 153
10.4.6 Optional External Reference Clock ..... 153
10.4.7 Fixed Frequency Clock ..... 153

## Chapter 11
## Inter-Integrated Circuit (S08IICV1)

11.1 Introduction ..... 155
11.1.1 Module Configuration ..... 155
11.1.2 Features ..... 157
11.1.3 Modes of Operation ..... 157
11.1.4 Block Diagram ..... 158
11.2 External Signal Description ..... 158
11.2.1 SCL — Serial Clock Line ..... 158
11.2.2 SDA — Serial Data Line ..... 158
11.3 Register Definition ..... 158
11.3.1 IIC Address Register (IICA) ..... 159
11.3.2 IIC Frequency Divider Register (IICF) ..... 159
11.3.3 IIC Control Register (IICC) ..... 162
11.3.4 IIC Status Register (IICS) ..... 163
11.3.5 IIC Data I/O Register (IICD) ..... 164

11.4 Functional Description ..... 165
11.4.1 IIC Protocol ..... 165
11.5 Resets ..... 168
11.6 Interrupts ..... 168
11.6.1 Byte Transfer Interrupt ..... 169
11.6.2 Address Detect Interrupt ..... 169
11.6.3 Arbitration Lost Interrupt ..... 169
11.7 Initialization/Application Information ..... 170

## Chapter 12
## Keyboard Interrupt (S08KBIV2)

12.1 Introduction ..... 173
12.1.1 Features ..... 175
12.1.2 Modes of Operation ..... 175
12.1.3 Block Diagram ..... 175
12.2 External Signal Description ..... 176
12.3 Register Definition ..... 176
12.3.1 KBI Status and Control Register (KBISC) ..... 176
12.3.2 KBI Pin Enable Register (KBIPE) ..... 177
12.3.3 KBI Edge Select Register (KBIES) ..... 177
12.4 Functional Description ..... 178
12.4.1 Edge Only Sensitivity ..... 178
12.4.2 Edge and Level Sensitivity ..... 178
12.4.3 KBI Pullup/Pulldown Resistors ..... 179
12.4.4 KBI Initialization ..... 179

## Chapter 13
## Modulo Timer (S08MTIMV1)

13.1 Introduction ..... 181
13.1.1 MTIM/TPM Configuration Information ..... 181
13.1.2 Features ..... 183
13.1.3 Modes of Operation ..... 183
13.1.4 Block Diagram ..... 184
13.2 External Signal Description ..... 184
13.3 Register Definition ..... 184
13.3.1 MTIM Status and Control Register (MTIMSC) ..... 186
13.3.2 MTIM Clock Configuration Register (MTIMCLK) ..... 187
13.3.3 MTIM Counter Register (MTIMCNT) ..... 188
13.3.4 MTIM Modulo Register (MTIMMOD) ..... 188
13.4 Functional Description ..... 189
13.4.1 MTIM Operation Example ..... 190

## Chapter 14
## Serial Communications Interface (S08SCIV3)

14.1 Introduction ..... 191
14.1.1 Features ..... 194
14.1.2 Modes of Operation ..... 194
14.1.3 Block Diagram ..... 195
14.2 Register Definition ..... 197
14.2.1 SCI Baud Rate Registers (SCIBDH, SCIBHL) ..... 197
14.2.2 SCI Control Register 1 (SCIC1) ..... 198
14.2.3 SCI Control Register 2 (SCIC2) ..... 199
14.2.4 SCI Status Register 1 (SCIS1) ..... 200
14.2.5 SCI Status Register 2 (SCIS2) ..... 202
14.2.6 SCI Control Register 3 (SCIC3) ..... 202
14.2.7 SCI Data Register (SCID) ..... 203
14.3 Functional Description ..... 204
14.3.1 Baud Rate Generation ..... 204
14.3.2 Transmitter Functional Description ..... 204
14.3.3 Receiver Functional Description ..... 206
14.3.4 Interrupts and Status Flags ..... 207
14.4 Additional SCI Functions ..... 208
14.4.1 8- and 9-Bit Data Modes ..... 208
14.4.2 Stop Mode Operation ..... 209
14.4.3 Loop Mode ..... 209
14.4.4 Single-Wire Operation ..... 209

## Chapter 15
## Serial Peripheral Interface (S08SPIV3)

15.1 Introduction ..... 211
15.1.1 Features ..... 213
15.1.2 Block Diagrams ..... 213
15.1.3 SPI Baud Rate Generation ..... 215
15.2 External Signal Description ..... 216
15.2.1 SPSCK — SPI Serial Clock ..... 216
15.2.2 MOSI — Master Data Out, Slave Data In ..... 216
15.2.3 MISO — Master Data In, Slave Data Out ..... 216
15.2.4 $\overline{SS}$ — Slave Select ..... 216
15.3 Modes of Operation ..... 217
15.3.1 SPI in Stop Modes ..... 217
15.4 Register Definition ..... 217
15.4.1 SPI Control Register 1 (SPIC1) ..... 217
15.4.2 SPI Control Register 2 (SPIC2) ..... 218
15.4.3 SPI Baud Rate Register (SPIBR) ..... 219

15.4.4 SPI Status Register (SPIS) ..... 220
15.4.5 SPI Data Register (SPID) ..... 221
15.5 Functional Description ..... 222
15.5.1 SPI Clock Formats ..... 222
15.5.2 SPI Interrupts ..... 225
15.5.3 Mode Fault Detection ..... 225

## Chapter 16
## Timer/Pulse-Width Modulator (S08TPMV2)

16.1 Introduction ..... 227
16.1.1 ACMP/TPM Configuration Information ..... 227
16.1.2 MTIM/TPM Configuration Information ..... 227
16.1.3 Features ..... 229
16.1.4 Block Diagram ..... 229
16.2 External Signal Description ..... 231
16.2.1 External TPM Clock Sources ..... 231
16.2.2 TPMCHn — TPM Channel n I/O Pins ..... 231
16.3 Register Definition ..... 231
16.3.1 Timer Status and Control Register (TPMSC) ..... 232
16.3.2 Timer Counter Registers (TPMCNTH:TPMCNTL) ..... 233
16.3.3 Timer Counter Modulo Registers (TPMMODH:TPMMODL) ..... 234
16.3.4 Timer Channel n Status and Control Register (TPMCnSC) ..... 235
16.3.5 Timer Channel Value Registers (TPMCnVH:TPMCnVL) ..... 236
16.4 Functional Description ..... 237
16.4.1 Counter ..... 237
16.4.2 Channel Mode Selection ..... 238
16.4.3 Center-Aligned PWM Mode ..... 240
16.5 TPM Interrupts ..... 241
16.5.1 Clearing Timer Interrupt Flags ..... 241
16.5.2 Timer Overflow Interrupt Description ..... 241
16.5.3 Channel Event Interrupt Description ..... 242
16.5.4 PWM End-of-Duty-Cycle Events ..... 242

## Chapter 17
## Development Support

17.1 Introduction ..... 243
17.1.1 Module Configuration ..... 243
17.1.2 Features ..... 244
17.2 Background Debug Controller (BDC) ..... 244
17.2.1 BKGD Pin Description ..... 245
17.2.2 Communication Details ..... 246
17.2.3 BDC Commands ..... 248

17.2.4 BDC Hardware Breakpoint ..... 251
17.3 On-Chip Debug System (DBG) ..... 252
17.3.1 Comparators A and B ..... 252
17.3.2 Bus Capture Information and FIFO Operation ..... 252
17.3.3 Change-of-Flow Information ..... 253
17.3.4 Tag vs. Force Breakpoints and Triggers ..... 253
17.3.5 Trigger Modes ..... 254
17.3.6 Hardware Breakpoints ..... 256
17.4 Register Definition ..... 256
17.4.1 BDC Registers and Control Bits ..... 256
17.4.2 System Background Debug Force Reset Register (SBDFR) ..... 258
17.4.3 DBG Registers and Control Bits ..... 259

## Appendix A
## Electrical Characteristics

A.1 Introduction ..... 265
A.2 Absolute Maximum Ratings ..... 265
A.3 Thermal Characteristics ..... 266
A.4 ESD Protection and Latch-Up Immunity ..... 268
A.5 DC Characteristics ..... 269
A.6 Supply Current Characteristics ..... 272
A.7 External Oscillator (XOSC) and Internal Clock Source (ICS) Characteristics ..... 274
A.8 AC Characteristics ..... 276
A.8.1 Control Timing ..... 276
A.8.2 TPM/MTIM Module Timing ..... 277
A.8.3 SPI Timing ..... 278
A.9 Analog Comparator (ACMP) Electricals ..... 282
A.10 ADC Characteristics ..... 282
A.11 FLASH Specifications ..... 285
A.12 EMC Performance ..... 286
A.12.1 Radiated Emissions ..... 286
A.12.2 Conducted Transient Susceptibility ..... 286

## Appendix B
## Ordering Information and Mechanical Drawings

B.1 Ordering Information ..... 289
B.1.1 Device Numbering Scheme ..... 289
B.2 Mechanical Drawings ..... 289

| Section Number | Title | Page |
|---|---|---|

# Chapter 1
# Device Overview

## 1.1 Introduction

The MC9S08QG8 is a member of the low-cost, high-performance HCS08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced HCS08 core and are available with a variety of modules, memory sizes, memory types, and package types. Refer to Table 1-1 for features associated with each device in this series.

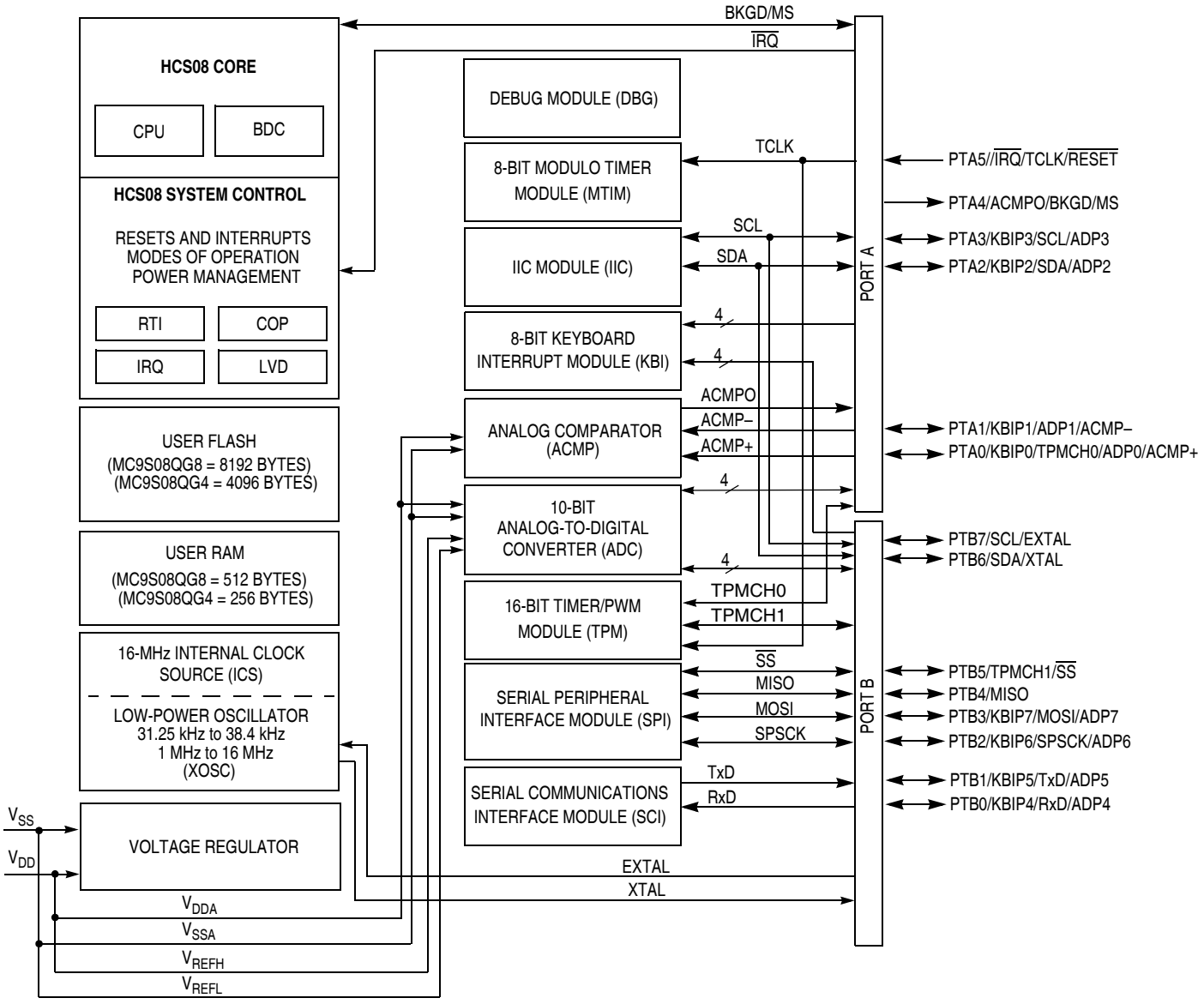
### 1.1.1 Devices in the MC9S08QG8/4 Series

Table 1-1 summarizes the features available in the MC9S08QG8/4 series of MCUs.

**Table 1-1. Devices in the MC9S08QG8/4 Series**

| Feature | Device | | | | | |
|---|---|---|---|---|---|---|
| | MC9S08QG8 | | | MC9S08QG4 | | |
| Package | 24-Pin | 16-Pin | 8-Pin | 24-Pin | 16-Pin | 8-Pin |
| FLASH | 8K | | | 4K | | |
| RAM | 512 | | | 256 | | |
| XOSC | yes | yes | no | yes | yes | no |
| ICS | yes | | | yes | | |
| ACMP | yes | | | yes | | |
| ADC | 8-ch | 8-ch | 4-ch | 8-ch | 8-ch | 4-ch |
| DBG | yes | | | yes | yes | yes |
| IIC | yes | | | yes | | |
| IRQ | yes | | | yes | | |
| KBI | 8-pin | 8-pin | 4-pin | 8-pin | 8-pin | 4-pin |
| MTIM | yes | | | yes | | |
| SCI | yes | yes | no | yes | yes | no |
| SPI | yes | yes | no | yes | yes | no |
| TPM | 2-ch | 2-ch | 1-ch | 2-ch | 2-ch | 1-ch |
| I/O pins | 12 I/O<br>1 Output only<br>1 Input only | 12 I/O<br>1 Output only<br>1 Input only | 4 I/O<br>1 Output only<br>1 Input only | 12 I/O<br>1 Output only<br>1 Input only | 12 I/O<br>1 Output only<br>1 Input only | 4 I/O<br>1 Output only<br>1 Input only |
| Package Types | 24 QFN | 16 PDIP<br>16 QFN<br>16 TSSOP | 8 DFN<br>8 SOIC | 24 QFN | 16 QFN<br>16 TSSOP | 8 DFN<br>8 PDIP<br>8 SOIC |

## 1.1.2 MCU Block Diagram




NOTES:
1. Not all pins or pin functions are available on all devices; see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

**Figure 1-1. MC9S08QG8/4 Block Diagram**

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

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

| Module | | Version |
|---|---|---|
| Analog Comparator | (ACMP) | 2 |
| Analog-to-Digital Converter | (ADC) | 1 |
| Central Processing Unit | (CPU) | 2 |
| IIC Module | (IIC) | 1 |
| Internal Clock Source | (ICS) | 1 |
| Keyboard Interrupt | (KBI) | 2 |
| Modulo Timer | (MTIM) | 1 |
| Serial Communications Interface | (SCI) | 3 |
| Serial Peripheral Interface | (SPI) | 3 |
| Timer Pulse-Width Modulator | (TPM) | 2 |
| Low-Power Oscillator | (XOSC) | 1 |
| Debug Module | (DBG) | 2 |

## System Clock Distribution

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

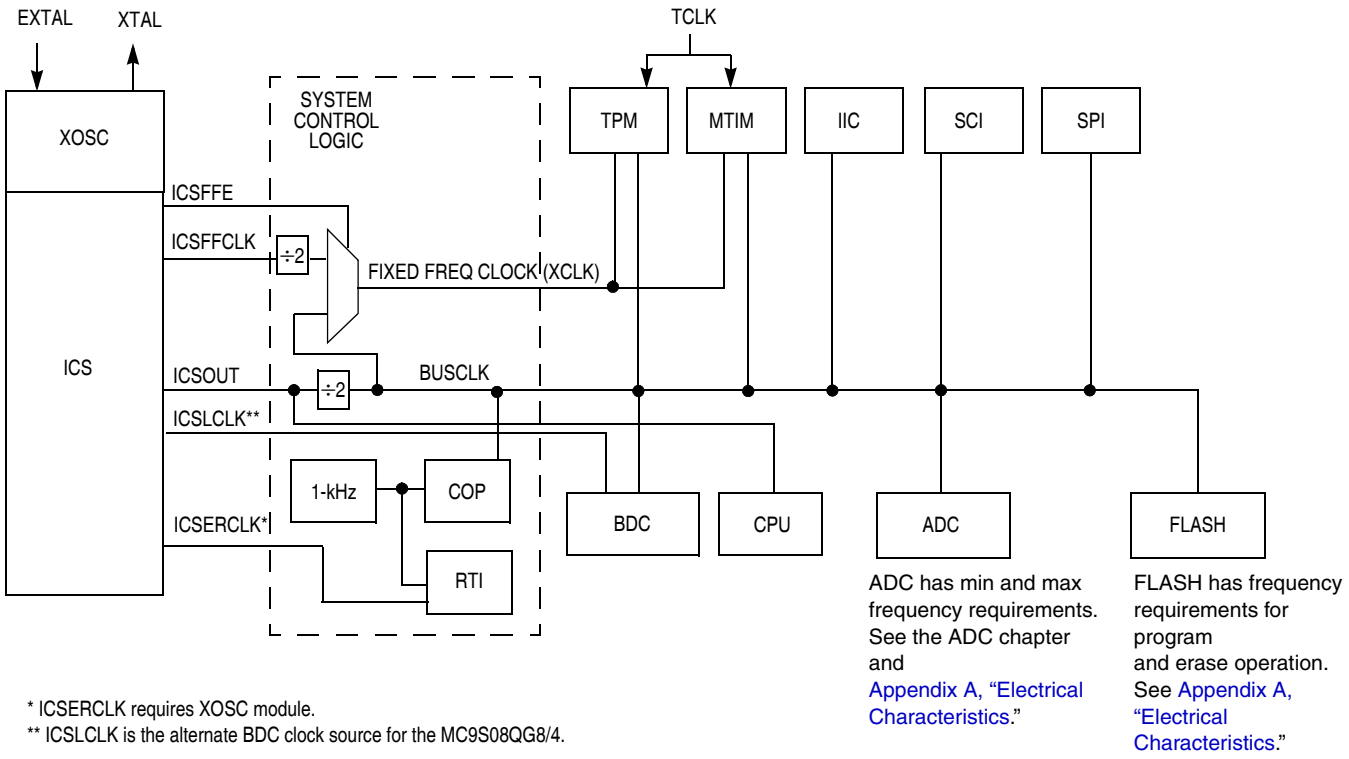



**Figure 1-2. System Clock Distribution Diagram**

# Chapter 2
# External Signal Description

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

## 2.1 Device Pin Assignment

The following figures show the pin assignments for the available packages. Refer to Table 1-1 to see which package types are available for each device in the series.

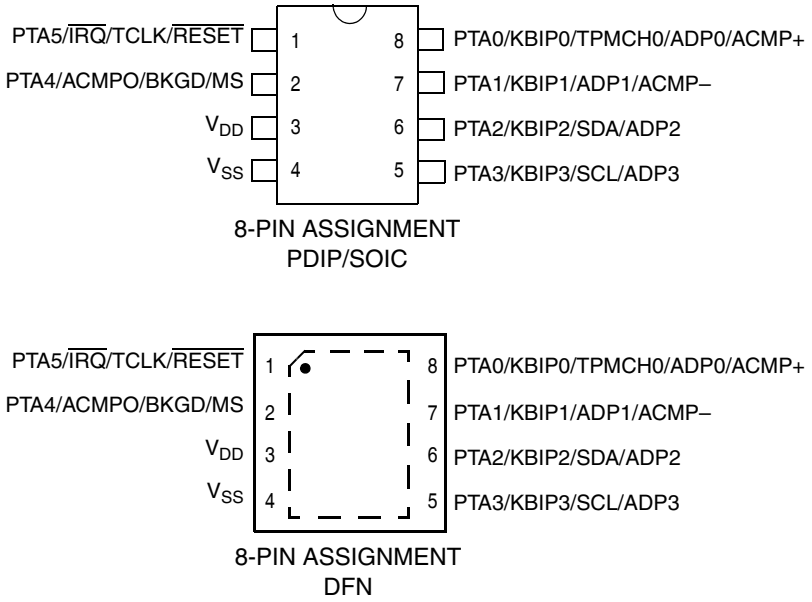



**Figure 2-1. 8-Pin Packages**

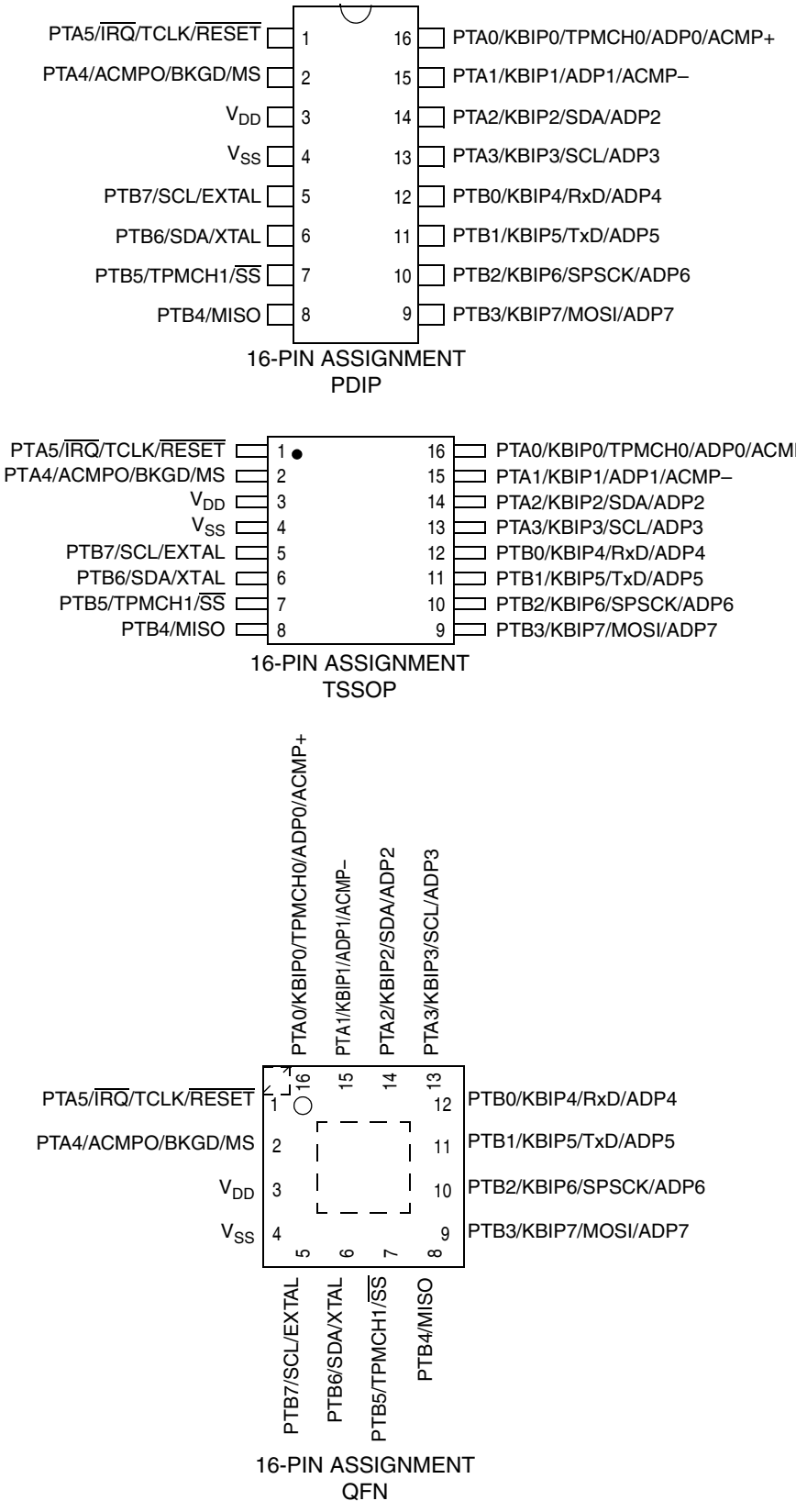



**Figure 2-2. 16-Pin Packages**

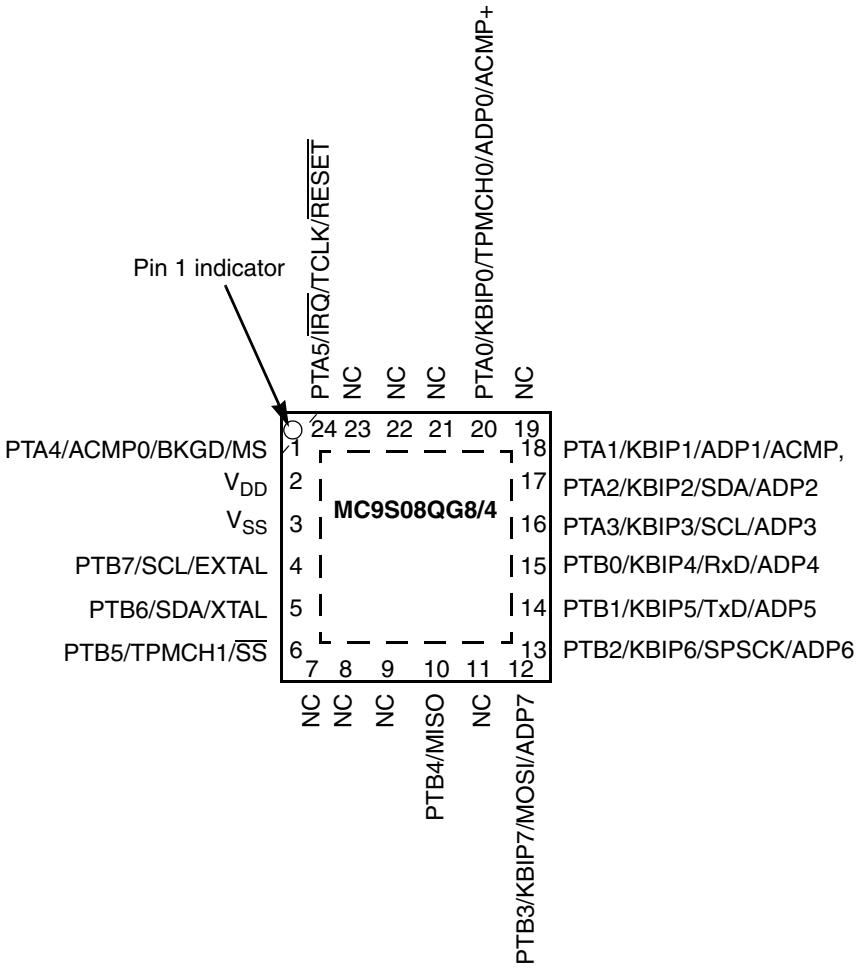
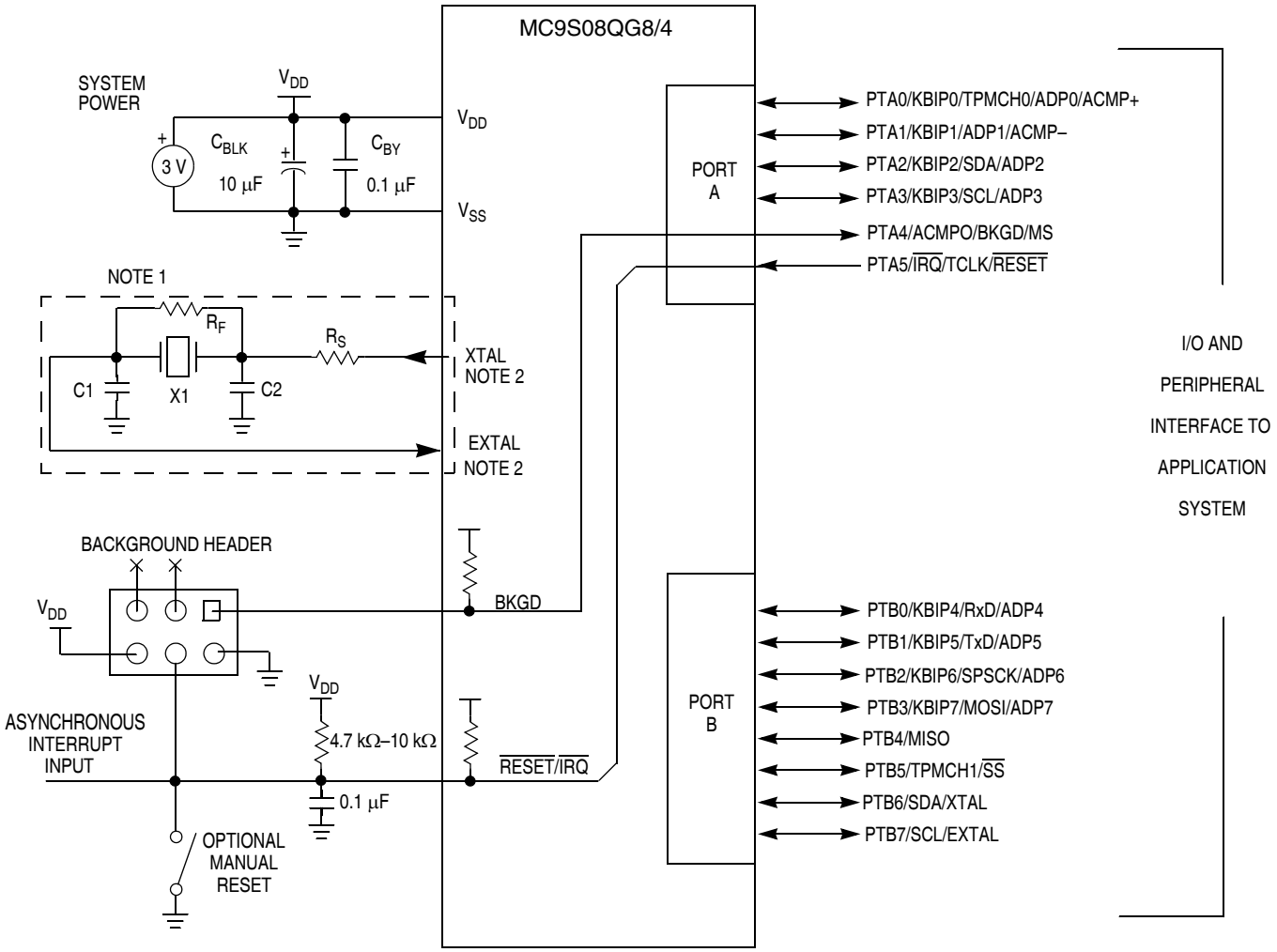



**Figure 2-3. 24-Pin Packages**

## 2.2 Recommended System Connections

Figure 2-4 shows pin connections that are common to almost all MC9S08QG8/4 application systems.




NOTES:
1. Not required if using the internal clock option.
2. XTAL is the same pin as PTB6; EXTAL the same pin as PTB7.
3. The RESET pin can only be used to reset into user mode; you can not enter BDM using the RESET pin. BDM can be entered by holding MS low during POR or writing a 1 to BDFR in SBDFR with MS low after issuing the BDM command.
4. IRQ feature has optional internal pullup device.
5. RC filter on RESET/IRQ pin recommended for noisy environments.

**Figure 2-4. Basic System Connections**

## 2.2.1 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, ACMP and ADC modules, 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: a bulk electrolytic capacitor, such as a 10-µF tantalum capacitor, to provide bulk charge storage for the overall system, and a bypass capacitor, such as a 0.1-µF ceramic capacitor, located as near to the MCU power pins as practical to suppress high-frequency noise.

### 2.2.2 Oscillator (XOSC)

Out of reset, the MCU uses an internally generated clock provided by the internal clock source (ICS) module. The internal frequency is nominally 16-MHz and the default ICS settings will provide for a 8-MHz bus out of reset. For more information on the ICS, see Chapter 10, "Internal Clock Source (S08ICSV1)."

The oscillator module (XOSC) in this MCU is a Pierce oscillator that can accommodate a crystal or ceramic resonator in either of two frequency ranges selected by the RANGE bit in ICSC2. Rather than a crystal or ceramic resonator, an external clock source can be connected to the EXTAL input pin.

Refer to Figure 2-4 for the following discussion. $R_S$ (when used) and $R_F$ should be low-inductance resistors such as carbon composition resistors. Wire-wound resistors, and some metal film resistors, have too much inductance. C1 and C2 normally should be high-quality ceramic capacitors that are specifically designed for high-frequency applications.

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

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

### 2.2.3 Reset (Input Only)

After a power-on reset (POR), the PTA5/$\overline{\text{IRQ}}$/TCLK/$\overline{\text{RESET}}$ pin defaults to a general-purpose input port pin, PTA5. Setting RSTPE in SOPT1 configures the pin to be the $\overline{\text{RESET}}$ input pin. After configured as $\overline{\text{RESET}}$, the pin will remain $\overline{\text{RESET}}$ until the next POR. The $\overline{\text{RESET}}$ pin can be used to reset the MCU from an external source when the pin is driven low. When enabled as the $\overline{\text{RESET}}$ pin (RSTPE = 1), an internal pullup device is automatically enabled.

**NOTE**

This pin does not contain a clamp diode to $V_{DD}$ and should not be driven above $V_{DD}$.

The voltage measured on the internally pulled-up $\overline{\text{RESET}}$ pin will not be pulled to $V_{DD}$. The internal gates connected to this pin are pulled to $V_{DD}$. The $\overline{\text{RESET}}$ pullup should not be used to pull up components external to the MCU.

**NOTE**

In EMC-sensitive applications, an external RC filter is recommended on the $\overline{\text{RESET}}$ pin, if enabled. See Figure 2-4 for an example.

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

During a power-on-reset (POR) or background debug force reset (see 5.8.3, “System Background Debug Force Reset Register (SBDFR),” for more information), the PTA4/ACMPO/BKGD/MS pin functions as a mode select pin. Immediately after any reset, the pin functions as the background pin and can be used for background debug communication. When enabled as the BKGD/MS pin (BKGDPE = 1), an internal pullup device is automatically enabled.

The background debug communication function is enabled when BKGDPE in SOPT1 is set. BKGDPE is set following any reset of the MCU and must be cleared to use the PTA4/ACMPO/BKGD/MS pin’s alternative pin functions.

If nothing is connected to this pin, the MCU will enter normal operating mode at the rising edge of the internal reset after a POR or force BDC reset. If a debug system is connected to the 6-pin standard background debug header, it can hold BKGD/MS low during a POR or immediately after issuing a background debug force reset, which will force 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 maximum bus clock rate, so there must never be any significant capacitance connected to the BKGD/MS pin that could interfere with background serial communications.

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

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

The MC9S08QG8/4 series of MCUs support up to 12 general-purpose I/O pins, 1 input-only pin, and 1 output-only pin, which are shared with on-chip peripheral functions (timers, serial I/O, ADC, keyboard interrupts, etc.). On each MC9S08QG8/4 device, there is one input-only and one output-only port pin.

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

For information about controlling these pins as general-purpose I/O pins, see the Chapter 6, “Parallel Input/Output Control.” For information about how and when on-chip peripheral systems use these pins, see the appropriate chapter referenced in Table 2-2.

Immediately after reset, all pins that are not output-only are configured as high-impedance general-purpose inputs with internal pullup devices disabled. After reset, the output-only port function is not enabled but is configured for low output drive strength with slew rate control enabled. The PTA4 pin defaults to BKGD/MS on any reset.

**NOTE**

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

When using the 8-pin devices, the user must either enable on-chip pullup devices or change the direction of non-bonded out port B pins to outputs so the pins do not float.

#### 2.2.5.1 Pin Control Registers

To select drive strength or enable slew rate control or pullup devices, the user writes to the appropriate pin control register located in the high page register block of the memory map. The pin control registers operate independently of the parallel I/O registers and allow control of a port on an individual pin basis.

##### 2.2.5.1.1 Internal Pullup Enable

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

The KBI module, when enabled for rising edge detection, causes an enabled internal pull device to be configured as a pulldown.

#### 2.2.5.2 Output Slew Rate Control

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

#### 2.2.5.3 Output Drive Strength Select

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

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

| Pin Number | | | Priority (Lowest → Highest) | | | | |
|---|---|---|---|---|---|---|---|
| 24-pin | 16-pin | 8-pin | Port Pin | Alt 1 | Alt 2 | Alt 3 | Alt 4 |
| 24 | 1 | 1 | PTA5[1] | $\overline{IRQ}$ | TCLK | | $\overline{RESET}$ |
| 1 | 2 | 2 | PTA4 | | ACMPO | BKGD | MS |
| 2 | 3 | 3 | | | | | $V_{DD}$ |
| 3 | 4 | 4 | | | | | $V_{SS}$ |
| 4 | 5 | — | PTB7 | | SCL[2] | EXTAL | |
| 5 | 6 | — | PTB6 | | SDA[2] | XTAL | |
| 6 | 7 | — | PTB5 | | TPMCH1 | $\overline{SS}$ | |
| 10 | 8 | — | PTB4 | | MISO | | |
| 12 | 9 | — | PTB3 | KBIP7 | MOSI | ADP7 | |
| 13 | 10 | — | PTB2 | KBIP6 | SPSCK | ADP6 | |
| 14 | 11 | — | PTB1 | KBIP5 | TxD | ADP5 | |
| 15 | 12 | — | PTB0 | KBIP4 | RxD | ADP4 | |
| 16 | 13 | 5 | PTA3 | KBIP3 | SCL[2] | ADP3 | |
| 17 | 14 | 6 | PTA2 | KBIP2 | SDA[2] | ADP2 | |
| 18 | 15 | 7 | PTA1 | KBIP1 | | ADP1[3] | ACMP–[3] |
| 20 | 16 | 8 | PTA0 | KBIP0 | TPMCH0 | ADP0[3] | ACMP+[3] |

1 Pin does not contain a clamp diode to $V_{DD}$ and should not be driven above $V_{DD}$. The voltage measured on the internally pulled-up $\overline{RESET}$ pin will not be pulled to $V_{DD}$. The internal gates connected to this pin are pulled to $V_{DD}$.

2 IIC pins can be repositioned using IICPS in SOPT2; default reset locations are on PTA2 and PTA3.

3 If ACMP and ADC are both enabled, both will have access to the pin.

**Table 2-2. Pin Function Reference**

| Signal Function | Example(s) | Reference |
|---|---|---|
| Port Pins | PTAx, PTBx | Chapter 6, “Parallel Input/Output Control” |
| Analog comparator | ACMPO, ACMP–, ACMP+ | Chapter 8, “Analog Comparator (S08ACMPV2)” |
| Serial peripheral interface | $\overline{SS}$, MISO, MOSI, SPSCK | Chapter 15, “Serial Peripheral Interface (S08SPIV3) |
| Keyboard interrupts | KBIPx | Chapter 12, “Keyboard Interrupt (S08KBIV2)” |
| Timer/PWM | TCLK, TPMCHx | Chapter 16, “Timer/Pulse-Width Modulator (S08TPMV2)” |
| Inter-integrated circuit | SCL, SDA | Chapter 11, “Inter-Integrated Circuit (S08IICV1)” |
| Serial communications interface | TxD, RxD | Chapter 14, “Serial Communications Interface (S08SCIV3) |
| Oscillator/clocking | EXTAL, XTAL | Chapter 10, “Internal Clock Source (S08ICSV1)” |
| Analog-to-digital | ADPx | Chapter 9, “Analog-to-Digital Converter (S08ADC10V1)” |
| Power/core | BKGD/MS, $V_{DD}$, $V_{SS}$ | Chapter 2, “External Signal Description” |
| Reset and interrupts | $\overline{RESET}$, $\overline{IRQ}$ | Chapter 5, “Resets, Interrupts, and General System Control” |

**NOTE**

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

# Chapter 3
# Modes of Operation

## 3.1 Introduction

The operating modes of the MC9S08QG8/4 are described in this section. Entry into each mode, exit from each mode, and functionality while in each mode are described.

## 3.2 Features

- Active background mode for code development
- Wait mode:
  - CPU halts operation to conserve power
  - System clocks running
  - Full voltage regulation is maintained
- Stop modes: CPU and bus clocks stopped
  - Stop1: Full powerdown of internal circuits for maximum power savings
  - Stop2: Partial powerdown of internal circuits; RAM contents retained
  - Stop3: All internal circuits powered for fast recovery; RAM and register contents are retained

## 3.3 Run Mode

Run is the normal operating mode for the MC9S08QG8/4. This mode is selected upon the MCU exiting reset if the BKGD/MS pin is high. In this mode, the CPU executes code from internal memory with execution beginning at the address fetched from memory at 0xFFFE:0xFFFF after reset.

## 3.4 Active Background Mode

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

Active background mode is entered in any of five ways:

- When the BKGD/MS pin is low during POR or immediately after issuing a background debug force reset (see 5.8.3, “System Background Debug Force Reset Register (SBDFR)”)
- When a BACKGROUND command is received through the BKGD pin
- When a BGND instruction is executed
- When encountering a BDC breakpoint
- When encountering a DBG breakpoint

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

Background commands are of two types:

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

The active background mode is used to program a bootloader or user application program into the FLASH program memory before the MCU is operated in run mode for the first time. When the MC9S08QG8/4 is shipped from the Freescale factory, the FLASH program memory is erased by default unless specifically noted, so there is no program that could be executed in run mode until the FLASH memory is initially programmed. The active background mode can also be used to erase and reprogram the FLASH memory after it has been previously programmed.

For additional information about the active background mode, refer to 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 the condition code register (CCR) is cleared when the CPU enters wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits wait mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

While the MCU is in wait mode, there are some restrictions on which background debug commands can be used. Only the BACKGROUND command and memory-access-with-status commands are available while the MCU is in wait mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in 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 STOPE in SOPT1 is set. In any stop mode, the bus and CPU clocks are halted. The ICS module can be configured to leave the reference clocks running. See Chapter 10, "Internal Clock Source (S08ICSV1)," for more information.

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

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

| STOPE | ENBDM [1] | LVDE | LVDSE | PDC | PPDC | Stop Mode |
|---|---|---|---|---|---|---|
| 0 | x | x | | x | x | Stop modes disabled; illegal opcode reset if STOP instruction executed |
| 1 | 1 | x | | x | x | Stop3 with BDM enabled [2] |
| 1 | 0 | Both bits must be 1 | | x | x | Stop3 with voltage regulator active |
| 1 | 0 | Either bit a 0 | | 0 | x | Stop3 |
| 1 | 0 | Either bit a 0 | | 1 | 1 | Stop2 |
| 1 | 0 | Either bit a 0 | | 1 | 0 | Stop1 |

1 ENBDM is located in the BDCSCR which is only accessible through BDC commands; see Section 17.4.1.1, "BDC Status and Control Register (BDCSCR)".

2 When in Stop3 mode with BDM enabled, the $S_{IDD}$ will be near $R_{IDD}$ levels because internal clocks are enabled.

### 3.6.1 Stop3 Mode

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

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

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

#### 3.6.1.1 LVD Enabled in Stop Mode

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

For the ADC to operate the LVD must be left enabled when entering stop3.

#### 3.6.1.2 Active BDM Enabled in Stop Mode

Entry into the active background mode from run mode is enabled if ENBDM in BDCSCR is set. This register is described in Chapter 17, "Development Support." If ENBDM is set when the CPU executes a

STOP instruction, the system clocks to the background debug logic remain active when the MCU enters stop mode. Because of this, background debug communication remains possible. In addition, the voltage regulator does not enter its low-power standby state but maintains full internal regulation.

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 entering background debug mode, all background commands are available.

### 3.6.2 Stop2 Mode

Stop2 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1. Most of the internal circuitry of the MCU is powered off in stop2 as in stop1 with the exception of the RAM. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their states during stop2.

Exit from stop2 is performed by asserting the wake-up pin (PTA5) on the MCU.

**NOTE**

PTA5/$\overline{\text{IRQ}}$/TCLK/$\overline{\text{RESET}}$ always functions as an active-low wakeup input when the MCU is in stop2, regardless of how the pin is configured before entering stop2. The pullup is not automatically enabled. To use the internal pullup, set the PTAPE5 bit in the PTAPE register

In addition, the real-time interrupt (RTI) can wake the MCU from stop2, if enabled.

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

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

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

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

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

### 3.6.3 Stop1 Mode

Stop1 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1. Most of the internal circuitry of the MCU is powered off in stop1, providing the lowest possible standby current. Upon entering stop1, all I/O pins automatically transition to their default reset states.

Exit from stop1 is performed by asserting the wake-up pin (PTA5) on the MCU.

**NOTE**

PTA5/$\overline{\text{IRQ}}$/TCLK/$\overline{\text{RESET}}$ always functions as an active-low wakeup input when the MCU is in stop2, regardless of how the pin is configured before entering stop2. The pullup is not automatically enabled. To use the internal pullup, set the PTAPE5 bit in the PTAPE register

In addition, the real-time interrupt (RTI) can wake the MCU from stop1 if enabled.

Upon wake-up from stop1 mode, the MCU starts up as from a power-on reset (POR):

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

In addition to the above, upon waking up from stop1, the PDF bit in SPMSC2 is set. This flag is used to direct user code to go to a stop1 recovery routine. PDF remains set until a 1 is written to PPDACK in SPMSC2.

### 3.6.4 On-Chip Peripheral Modules in Stop Modes

When the MCU enters any stop mode, system clocks to the internal peripheral modules are stopped. Even in the exception case (ENBDM = 1), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. Refer to Section 3.6.3, “Stop1 Mode,” Section 3.6.2, “Stop2 Mode,” and Section 3.6.1, “Stop3 Mode,” for specific information on system behavior in stop modes.

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

| Peripheral | Mode | | |
|---|---|---|---|
| | Stop1 | Stop2 | Stop3 |
| CPU | Off | Off | Standby |
| RAM | Off | Standby | Standby |
| FLASH | Off | Off | Standby |
| Parallel Port Registers | Off | Off | Standby |
| ADC | Off | Off | Optionally On[1] |
| ACMP | Off | Off | Standby |
| ICS | Off | Off | Optionally On[2] |
| IIC | Off | Off | Standby |

**Table 3-2. Stop Mode Behavior (continued)**

| Peripheral | Mode | | |
|---|---|---|---|
| | Stop1 | Stop2 | Stop3 |
| MTIM | Off | Off | Standby |
| SCI | Off | Off | Standby |
| SPI | Off | Off | Standby |
| TPM | Off | Off | Standby |
| Voltage Regulator | Off | Standby | Standby |
| XOSC | Off | Off | Optionally On[3] |
| I/O Pins | Hi-Z | States Held | States Held |

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

2 IRCLKEN and IREFSTEN set in ICSC1, else in standby.

3 ERCLKEN and EREFSTEN set in ICSC2, else in standby. For high frequency range (RANGE in ICSC2 set) requires the LVD to also be enabled in stop3.

# Chapter 4
# Memory Map and Register Definition

## 4.1 MC9S08QG8/4 Memory Map

As shown in Figure 4-1, on-chip memory in the MC9S08QG8/4 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 these groups:

- Direct-page registers (0x0000 through 0x005F)
- High-page registers (0x1800 through 0x184F)
- Nonvolatile registers (0xFFB0 through 0xFFBF)

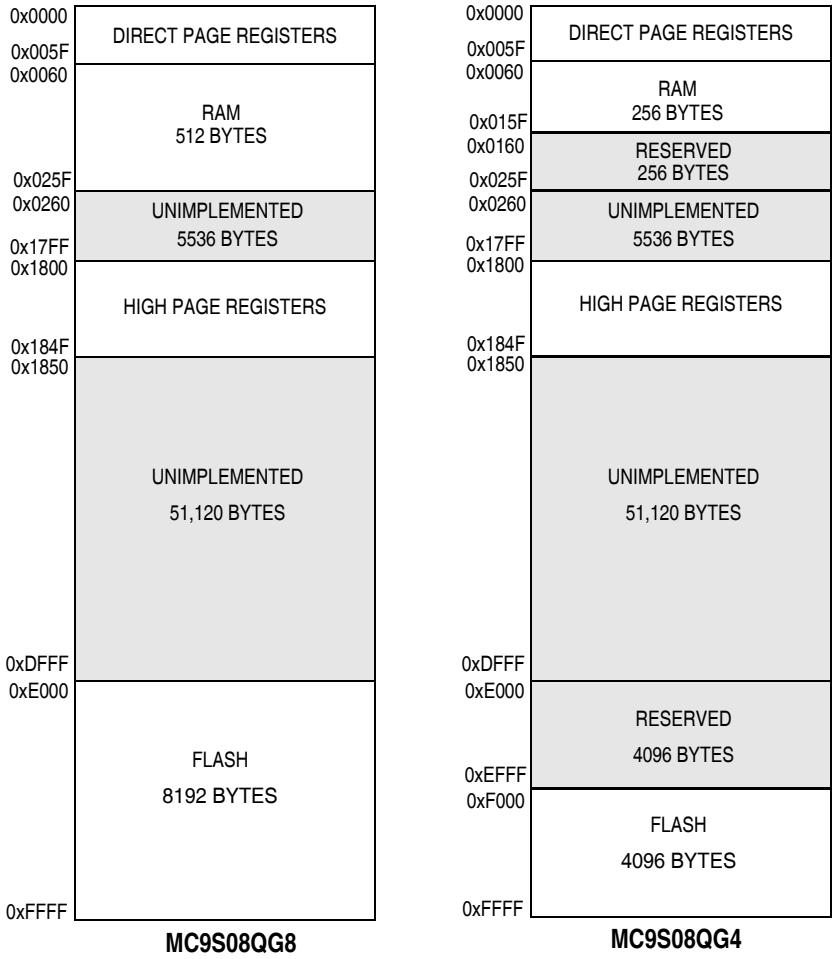



**Figure 4-1. MC9S08QG8/4 Memory Map**

## 4.2 Reset and Interrupt Vector Assignments

Table 4-1 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the Freescale Semiconductor-provided equate file for the MC9S08QG8/4.

**Table 4-1. Reset and Interrupt Vectors**

| Address (High:Low) | Vector | Vector Name |
|---|---|---|
| 0xFFC0:FFC1<br>⇕<br>0xFFCE:FFCF | Unused Vector Space<br>(available for user program) | |
| 0xFFD0:FFD1 | RTI | Vrti |
| 0xFFD2:FFD3 | Reserved | — |
| 0xFFD4:FFD5 | Reserved | — |
| 0xFFD6:FFD7 | ACMP | Vacmp |
| 0xFFD8:FFD9 | ADC Conversion | Vadc |
| 0xFFDA:FFDB | KBI Interrupt | Vkeyboard |
| 0xFFDC:FFDD | IIC | Viic |
| 0xFFDE:FFDF | SCI Transmit | Vscitx |
| 0xFFE0:FFE1 | SCI Receive | Vscirx |
| 0xFFE2:FFE3 | SCI Error | Vscierr |
| 0xFFE4:FFE5 | SPI | Vspi |
| 0xFFE6:FFE7 | MTIM Overflow | Vmtim |
| 0xFFE8:FFE9 | Reserved | — |
| 0xFFEA:FFEB | Reserved | — |
| 0xFFEC:FFED | Reserved | — |
| 0xFFEE:FFEF | Reserved | — |
| 0xFFF0:FFF1 | TPM Overflow | Vtpmovf |
| 0xFFF2:FFF3 | TPM Channel 1 | Vtpmch1 |
| 0xFFF4:FFF5 | TPM Channel 0 | Vtpmch0 |
| 0xFFF6:FFF7 | Reserved | — |
| 0xFFF8:FFF9 | Low Voltage Detect | Vlvd |
| 0xFFFA:FFFB | IRQ | Virq |
| 0xFFFC:FFFD | SWI | Vswi |
| 0xFFFE:FFFF | Reset | Vreset |

## 4.3 Register Addresses and Bit Assignments

The registers in the MC9S08QG8/4 are divided into these groups:

- Direct-page registers are located in the first 96 locations in the memory map; these are accessible with efficient direct addressing mode instructions.
- High-page registers are used much less often, so they are located from 0x1800 and above in the memory map. This leaves more room in the direct page for more frequently used registers and RAM.
- The nonvolatile register area consists of a block of 16 locations in FLASH memory at 0xFFB0–0xFFBF. Nonvolatile register locations include:
  - NVPROT and NVOPT are loaded into working registers at reset.
  - An 8-byte backdoor comparison key that optionally allows a user to gain controlled access to secure memory.

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

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

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

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 | **PTAD** | 0 | 0 | PTAD5 | PTAD4 | PTAD3 | PTAD2 | PTAD1 | PTAD0 |
| 0x0001 | **PTADD** | 0 | 0 | PTADD5 | PTADD4 | PTADD3 | PTADD2 | PTADD1 | PTADD0 |
| 0x0002 | **PTBD** | PTBD7 | PTBD6 | PTBD5 | PTBD4 | PTBD3 | PTBD2 | PTBD1 | PTBD0 |
| 0x0003 | **PTBDD** | PTBDD7 | PTBDD6 | PTBDD5 | PTBDD4 | PTBDD3 | PTBDD2 | PTBDD1 | PTBDD0 |
| 0x0004–0x000B | Reserved | — — | — — | — — | — — | — — | — — | — — | — — |
| 0x000C | **KBISC** | 0 | 0 | 0 | 0 | KBF | KBACK | KBIE | KBMOD |
| 0x000D | **KBIPE** | KBIPE7 | KBIPE6 | KBIPE5 | KBIPE4 | KBIPE3 | KBIPE2 | KBIPE1 | KBIPE0 |
| 0x000E | **KBIES** | KBEDG7 | KBEDG6 | KBEDG5 | KBEDG4 | KBEDG3 | KBEDG2 | KBEDG1 | KBEDG0 |
| 0x000F | **IRQSC** | 0 | IRQPDD | 0 | IRQPE | IRQF | IRQACK | IRQIE | IRQMOD |
| 0x0010 | **ADCSC1** | COCO | AIEN | ADCO | ADCH | | | | |
| 0x0011 | **ADCSC2** | ADACT | ADTRG | ACFE | ACFGT | — | — | — | — |
| 0x0012 | **ADCRH** | 0 | 0 | 0 | 0 | 0 | 0 | ADR9 | ADR8 |
| 0x0013 | **ADCRL** | ADR7 | ADR6 | ADR5 | ADR4 | ADR3 | ADR2 | ADR1 | ADR0 |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0014 | **ADCCVH** | 0 | 0 | 0 | 0 | 0 | 0 | ADCV9 | ADCV8 |
| 0x0015 | **ADCCVL** | ADCV7 | ADCV6 | ADCV5 | ADCV4 | ADCV3 | ADCV2 | ADCV1 | ADCV0 |
| 0x0016 | **ADCCFG** | ADLPC | ADIV | | ADLSMP | MODE | | ADICLK | |
| 0x0017 | **APCTL1** | ADPC7 | ADPC6 | ADPC5 | ADPC4 | ADPC3 | ADPC2 | ADPC1 | ADPC0 |
| 0x0018 | Reserved | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x0019 | Reserved | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x001A | **ACMPSC** | ACME | ACBGS | ACF | ACIE | ACO | ACOPE | ACMOD | |
| 0x001B–<br>0x001F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x0020 | **SCIBDH** | 0 | 0 | 0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| 0x0021 | **SCIBDL** | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| 0x0022 | **SCIC1** | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT |
| 0x0023 | **SCIC2** | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| 0x0024 | **SCIS1** | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| 0x0025 | **SCIS2** | 0 | 0 | 0 | 0 | 0 | BRK13 | 0 | RAF |
| 0x0026 | **SCIC3** | R8 | T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE |
| 0x0027 | **SCID** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0028 | **SPIC1** | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| 0x0029 | **SPIC2** | 0 | 0 | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| 0x002A | **SPIBR** | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| 0x002B | **SPIS** | SPRF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| 0x002C | Reserved | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x002D | **SPID** | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x002E | Reserved | — | — | — | — | — | — | — | — |
| 0x002F | Reserved | — | — | — | — | — | — | — | — |
| 0x0030 | **IICA** | ADDR | | | | | | | 0 |
| 0x0031 | **IICF** | MULT | | ICR | | | | | |
| 0x0032 | **IICC** | IICEN | IICIE | MST | TX | TXAK | RSTA | 0 | 0 |
| 0x0033 | **IICS** | TCF | IAAS | BUSY | ARBL | 0 | SRW | IICIF | RXAK |
| 0x0034 | **IICD** | DATA | | | | | | | |
| 0x0035 | Reserved | — | — | — | — | — | — | — | — |
| 0x0036 | Reserved | — | — | — | — | — | — | — | — |
| 0x0037 | Reserved | — | — | — | — | — | — | — | — |
| 0x0038 | **ICSC1** | CLKS | | RDIV | | | IREFS | IRCLKEN | IREFSTEN |
| 0x0039 | **ICSC2** | BDIV | | RANGE | HGO | LP | EREFS | ERCLKEN | EREFSTEN |
| 0x003A | **ICSTRM** | TRIM | | | | | | | |
| 0x003B | **ICSSC** | 0 | 0 | 0 | 0 | CLKST | | OSCINIT | FTRIM |
| 0x003C | **MTIMSC** | TOF | TOIE | TRST | TSTP | 0 | 0 | 0 | 0 |
| 0x003D | **MTIMCLK** | 0 | 0 | CLKS | | PS | | | |
| 0x003E | **MTIMCNT** | COUNT | | | | | | | |

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x003F | MTIMMOD | MOD | | | | | | | |
| 0x0040 | TPMSC | TOF | TOIE | CPWMS | CLKSB | CLKSA | PS2 | PS1 | PS0 |
| 0x0041 | TPMCNTH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0042 | TPMCNTL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0043 | TPMMODH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0044 | TPMMODL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0045 | TPMC0SC | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | 0 | 0 |
| 0x0046 | TPMC0VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x0047 | TPMC0VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x0048 | TPMC1SC | CH1F | CH1IE | MS1B | MS1A | ELS1B | ELS1A | 0 | 0 |
| 0x0049 | TPMC1VH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x004A | TPMC1VL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x004B–<br>0x005F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |

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

**Table 4-3. High-Page Register Summary**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x1800 | SRS | POR | PIN | COP | ILOP | ILAD | 0 | LVD | 0 |
| 0x1801 | SBDFR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BDFR |
| 0x1802 | SOPT1 | COPE | COPT | STOPE | — | 0 | 0 | BKGDPE | RSTPE |
| 0x1803 | SOPT2 | COPCLKS | 0 | 0 | 0 | 0 | 0 | IICPS | ACIC |
| 0x1804 | Reserved | — | — | — | — | — | — | — | — |
| 0x1805 | Reserved | — | — | — | — | — | — | — | — |
| 0x1806 | SDIDH | — | — | — | — | ID11 | ID10 | ID9 | ID8 |
| 0x1807 | SDIDL | ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |
| 0x1808 | SRTISC | RTIF | RTIACK | RTICLKS | RTIE | 0 | RTIS | | |
| 0x1809 | SPMSC1 | LVDF | LVDACK | LVDIE | LVDRE | LVDSE | LVDE | 0 | BGBE |
| 0x180A | SPMSC2 | 0 | 0 | 0 | PDF | PPDF | PPDACK | PDC | PPDC |
| 0x180B | Reserved | — | — | — | — | — | — | — | — |
| 0x180C | SPMSC3 | LVWF | LVWACK | LVDV | LVWV | — | — | — | — |
| 0x180D–<br>0x180F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x1810 | DBGCAH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1811 | DBGCAL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1812 | DBGCBH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1813 | DBGCBL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 0x1814 | DBGFH | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| 0x1815 | DBGFL | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

**Table 4-3. High-Page Register Summary (continued)**

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x1816 | DBGC | DBGEN | ARM | TAG | BRKEN | RWA | RWAEN | RWB | RWBEN |
| 0x1817 | DBGT | TRGSEL | BEGIN | 0 | 0 | TRG3 | TRG2 | TRG1 | TRG0 |
| 0x1818 | DBGS | AF | BF | ARMF | 0 | CNT3 | CNT2 | CNT1 | CNT0 |
| 0x1819–0x181F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x1820 | FCDIV | DIVLD | PRDIV8 | DIV | | | | | |
| 0x1821 | FOPT | KEYEN | FNORED | 0 | 0 | 0 | 0 | SEC01 | SEC00 |
| 0x1822 | Reserved | — | — | — | — | — | — | — | — |
| 0x1823 | FCNFG | 0 | 0 | KEYACC | 0 | 0 | 0 | 0 | 0 |
| 0x1824 | FPROT | FPS | | | | | | | FPDIS |
| 0x1825 | FSTAT | FCBEF | FCCF | FPVIOL | FACCERR | 0 | FBLANK | 0 | 0 |
| 0x1826 | FCMD | FCMD | | | | | | | |
| 0x1827–0x183F | Reserved | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— | —<br>— |
| 0x1840 | PTAPE | 0 | 0 | PTAPE5 | PTAPE4 | PTAPE3 | PTAPE2 | PTAPE1 | PTAPE0 |
| 0x1841 | PTASE | 0 | 0 | PTASE5 | PTASE4 | PTASE3 | PTASE2 | PTASE1 | PTASE0 |
| 0x1842 | PTADS | 0 | 0 | PTADS5 | PTADS4 | PTADS3 | PTADS2 | PTADS1 | PTADS0 |
| 0x1843 | Reserved | — | — | — | — | — | — | — | — |
| 0x1844 | PTBPE | PTBPE7 | PTBPE6 | PTBPE5 | PTBPE4 | PTBPE3 | PTBPE2 | PTBPE1 | PTBPE0 |
| 0x1845 | PTBSE | PTBSE7 | PTBSE6 | PTBSE5 | PTBSE4 | PTBSE3 | PTBSE2 | PTBSE1 | PTBSE0 |
| 0x1846 | PTBDS | PTBDS7 | PTBDS6 | PTBDS5 | PTBDS4 | PTBDS3 | PTBDS2 | PTBDS1 | PTBDS0 |
| 0x1847 | Reserved | — | — | — | — | — | — | — | — |

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

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

| Address | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0xFFAE | Reserved for Storage of FTRIM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FTRIM |
| 0xFFAF | Reserved for Storage of ICSTRM | TRIM | | | | | | | |
| 0xFFB0 – 0xFFB7 | NVBACKKEY | 8-Byte Comparison Key | | | | | | | |
| 0xFFB8 – 0xFFBC | Unused | — — | — — | — — | — — | — — | — — | — — | — — |
| 0xFFBD | NVPROT | FPS | | | | | | | FPDIS |
| 0xFFBE | Unused | — | — | — | — | — | — | — | — |
| 0xFFBF | NVOPT | KEYEN | FNORED | 0 | 0 | 0 | 0 | SEC01 | SEC00 |

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

## 4.4 RAM

The MC9S08QG8/4 includes static RAM. The locations in RAM below 0x0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

The RAM retains data when the MCU is in low-power wait, stop2, or stop3 mode. At power-on 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 ($V_{RAM}$).

For compatibility with M68HC05 MCUs, the HCS08 resets the stack pointer to 0x00FF. In the MC9S08QG8/4, 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 Semiconductor-provided equate file).

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

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

The RAM array is not automatically initialized out of reset.

## 4.5 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.

### 4.5.1 Features

Features of the FLASH memory include:

- FLASH size
  - MC9S08QG8: 8,192 bytes (16 pages of 512 bytes each)
  - MC9S08QG4: 4,096 bytes (8 pages of 512 bytes each)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for FLASH and RAM
- Auto power-down for low-frequency read accesses

### 4.5.2 Program and Erase Times

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

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

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

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

1 Excluding start/end overhead

**NOTE**

If the COP is enabled during an erase function, make sure the COP is serviced during the erase command execution.

### 4.5.3 Program and Erase Command Execution

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

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

**NOTE**

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

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

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

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

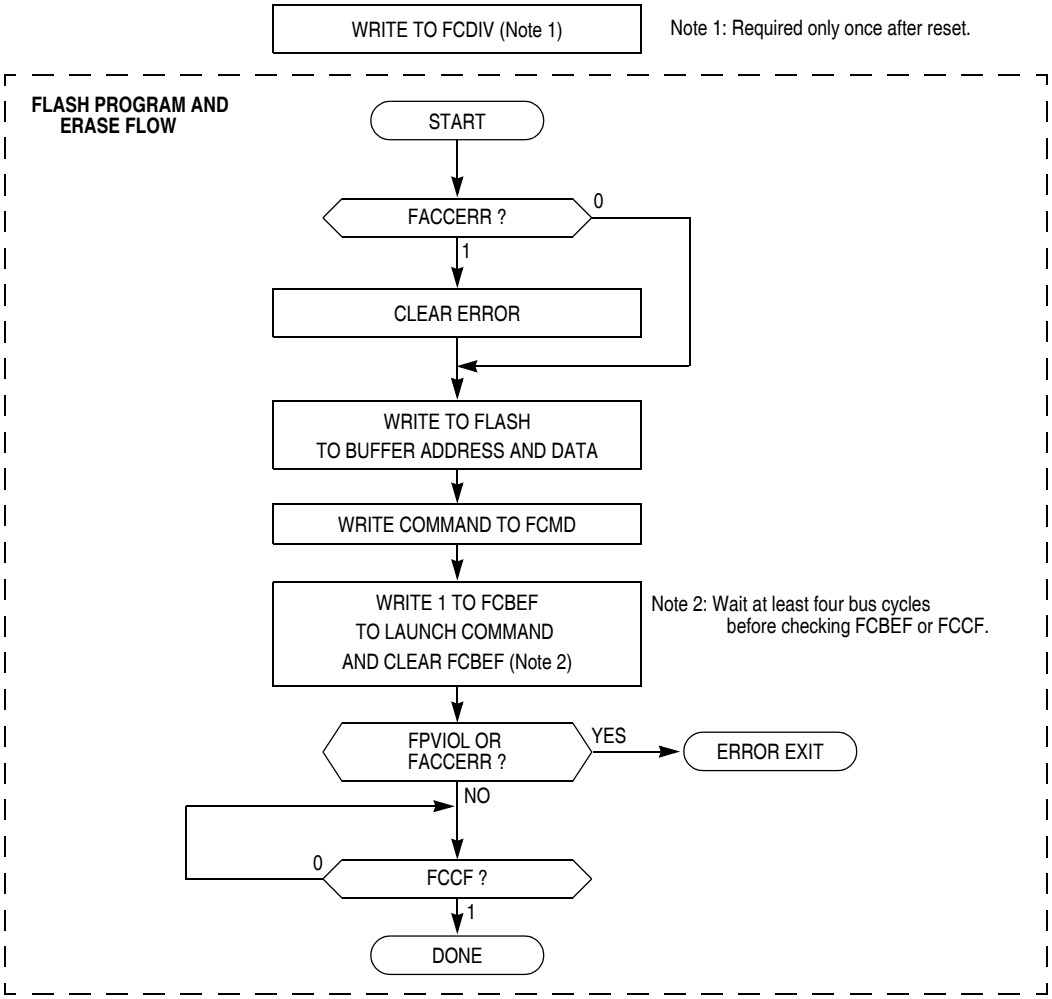



**Figure 4-2. FLASH Program and Erase Flowchart**

## 4.5.4 Burst Program Execution

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

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

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

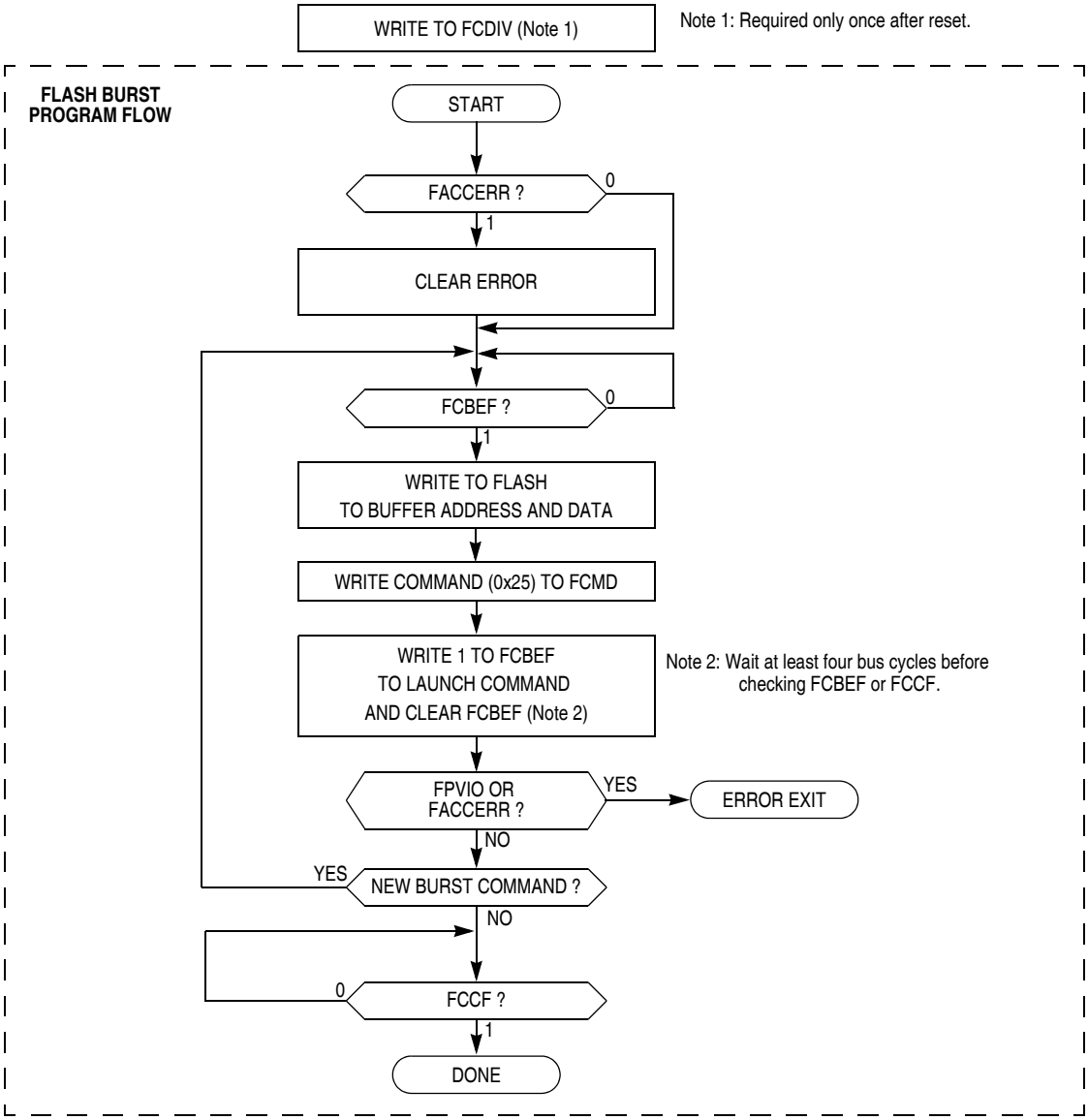



**Figure 4-3. FLASH Burst Program Flowchart**

### 4.5.5 Access Errors

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

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

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

### 4.5.6 FLASH Block Protection

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

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

The block protection mechanism is illustrated in Figure 4-4. The FPS bits are used as the upper bits of the last address of unprotected memory. This address is formed by concatenating FPS7:FPS1 with logic 1 bits as shown. For example, to protect the last 1536 bytes of memory (addresses 0xFA00 through 0xFFFF), the FPS bits must be set to 1111 100, which results in the value 0xF9FF as the last address of unprotected memory. In addition to programming the FPS bits to the appropriate value, FPDIS (bit 0 of NVPROT)

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

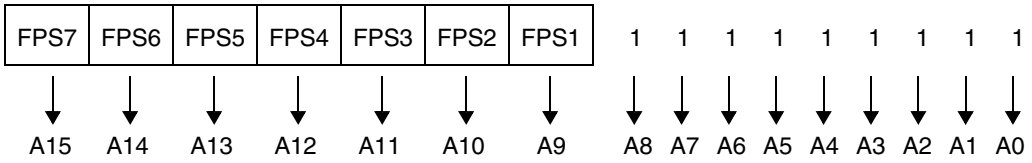



**Figure 4-4. Block Protection Mechanism**

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

### 4.5.7 Vector Redirection

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

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

## 4.6 Security

The MC9S08QG8/4 includes circuitry to prevent unauthorized access to the contents of FLASH and RAM memory. When security is engaged, FLASH and RAM are considered secure resources. Direct-page registers, high-page registers, and the background debug controller are considered unsecured resources. Programs executing within secure memory have normal access to any MCU memory locations and resources. Attempts to access a secure memory location with a program executing from an unsecured memory space or through the background debug interface are blocked (writes are ignored and reads return all 0s).

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

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

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

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

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

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

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

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

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

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

## 4.7 FLASH Registers and Control Bits

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

### 4.7.1 FLASH Clock Divider Register (FCDIV)

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

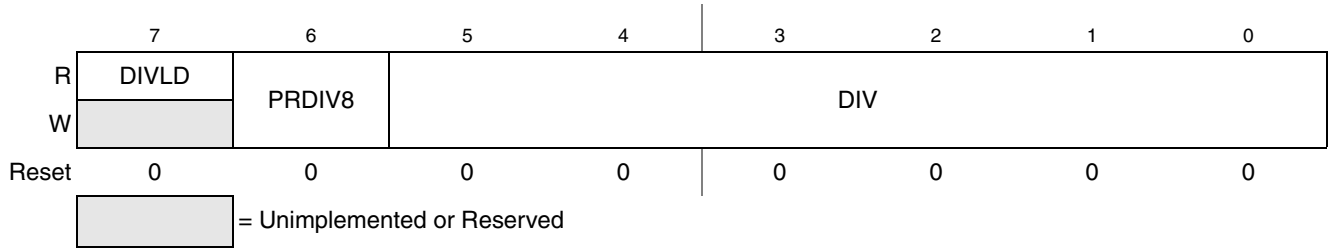



**Figure 4-5. FLASH Clock Divider Register (FCDIV)**

**Table 4-6. FCDIV Register Field Descriptions**

| Field | Description |
|---|---|
| 7<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 | **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 DIV 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. |

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

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

Table 4-7 shows the appropriate values for PRDIV8 and DIV for selected bus frequencies.

**Table 4-7. FLASH Clock Divider Settings**

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

## 4.7.2 FLASH Options Register (FOPT and NVOPT)

During reset, the contents of the nonvolatile location NVOPT are copied from FLASH into FOPT. To change the value in this register, erase and reprogram the NVOPT location in FLASH memory as usual and then issue a new MCU reset.

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

= Unimplemented or Reserved

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

**Table 4-8. FOPT Register Field Descriptions**

| Field | Description |
|---|---|
| 7<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 4.6, “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 in Table 4-9. When the MCU is secure, the contents of RAM and FLASH memory cannot be accessed by instructions from any unsecured source including the background debug interface. SEC01:SEC00 changes to 1:0 after successful backdoor key entry or a successful blank check of FLASH.<br>For more detailed information about security, refer to Section 4.6, “Security.” |

**Table 4-9. Security States[1]**

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

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

## 4.7.3 FLASH Configuration Register (FCNFG)

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

= Unimplemented or Reserved

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

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

| Field | Description |
|---|---|
| 5<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 4.6, “Security.”<br>0 Writes to 0xFFB0–0xFFB7 are interpreted as the start of a FLASH programming or erase command.<br>1 Writes to NVBACKKEY (0xFFB0–0xFFB7) are interpreted as comparison key writes. |

## 4.7.4 FLASH Protection Register (FPROT and NVPROT)

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | FPS(1) | | | | | | | FPDIS(1) |
| Reset | This register is loaded from nonvolatile location NVPROT during reset. | | | | | | | |

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

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

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

| Field | Description |
|---|---|
| 7:1<br>FPS | **FLASH Protect Select Bits** — When FPDIS = 0, this 7-bit field determines the ending address of unprotected FLASH locations at the high address end of the FLASH. Protected FLASH locations cannot be erased or programmed. |
| 0<br>FPDIS | **FLASH Protection Disable**<br>0 FLASH block specified by FPS7:FPS1 is block protected (program or erase not allowed).<br>1 No FLASH block is protected. |

## 4.7.5 FLASH Status Register (FSTAT)

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

= Unimplemented or Reserved

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

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

| Field | Description |
|---|---|
| 7<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 can 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. |

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

| Field | Description |
|---|---|
| 4<br>FACCERR | **Access Error Flag** — FACCERR is set automatically when the proper command sequence is not obeyed exactly (the erroneous command is ignored), if a program or erase operation is attempted before the FCDIV register has been initialized, or if the MCU enters stop while a command was in progress. For a more detailed discussion of the exact actions that are considered access errors, see Section 4.5.5, "Access Errors." FACCERR is cleared by writing a 1 to FACCERR. Writing a 0 to FACCERR has no meaning or effect.<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 0xFF). |

## 4.7.6 FLASH Command Register (FCMD)

Only five command codes are recognized in normal user modes as shown in Table 4-13. Refer to Section 4.5.3, "Program and Erase Command Execution," for a detailed discussion of FLASH programming and erase operations.

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

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

**Table 4-13. FLASH Commands**

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

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

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

# Chapter 5
# Resets, Interrupts, and General System Control

## 5.1 Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupts in the MC9S08QG8/4. 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 with their own chapters but are part of the system control logic.

## 5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vectors for each module (reduces polling overhead) (see Table 5-2)

## 5.3 MCU Reset

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

The MC9S08QG8/4 has the following sources for reset:

- External pin reset (PIN) — enabled using RSTPE in SOPT1
- Power-on reset (POR)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- Background debug force reset

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

## 5.4 Computer Operating Properly (COP) Watchdog

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

After any reset, the COPE becomes set in SOPT1 enabling the COP watchdog (see Section 5.8.4, "System Options Register 1 (SOPT1)," for additional information). If the COP watchdog is not used in an application, it can be disabled by clearing COPE. The COP counter is reset by writing any value to the address of SRS. This write does not affect the data in the read-only SRS. Instead, the act of writing to this address is decoded and sends a reset signal to the COP counter.

The COPCLKS bit in SOPT2 (see Section 5.8.5, "System Options Register 2 (SOPT2)," for additional information) selects the clock source used for the COP timer. The clock source options are either the bus clock or an internal 1-kHz clock source. With each clock source, there is an associated short and long time-out controlled by COPT in SOPT1. Table 5-1 summaries the control functions of the COPCLKS and COPT bits. The COP watchdog defaults to operation from the 1-kHz clock source and the associated long time-out ($2^8$ cycles).

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

| Control Bits | | Clock Source | COP Overflow Count |
|---|---|---|---|
| COPCLKS | COPT | | |
| 0 | 0 | ~1 kHz | $2^5$ cycles (32 ms)[1] |
| 0 | 1 | ~1 kHz | $2^8$ cycles (256 ms)[1] |
| 1 | 0 | Bus | $2^{13}$ cycles |
| 1 | 1 | Bus | $2^{18}$ cycles |

[1] Values are shown in this column based on $t_{RTI}$ = 1 ms. See $t_{RTI}$ in the appendix Section A.8.1, "Control Timing," for the tolerance of this value.

Even if the application will use the reset default settings of COPE, COPCLKS, and COPT, the user must write to the write-once SOPT1 and SOPT2 registers during reset initialization to lock in the settings. That way, they cannot be changed accidentally if the application program gets lost. The initial writes to SOPT1 and SOPT2 will reset the COP counter.

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

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

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

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

## 5.5 Interrupts

Interrupts provide a way to save the current CPU status and registers, execute an interrupt service routine (ISR), and then restore the CPU status so processing resumes where it 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 $\overline{\text{IRQ}}$ 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 1 to enable the interrupt. The I bit in the CCR is 0 to allow interrupts. The global interrupt mask (I bit) in the CCR is initially set after reset, which masks (prevents) all maskable interrupt sources. The user program initializes the stack pointer and performs other system setup before clearing the I bit to allow the CPU to respond to interrupts.

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

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

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

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

**NOTE**

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

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

### 5.5.1 Interrupt Stack Frame

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

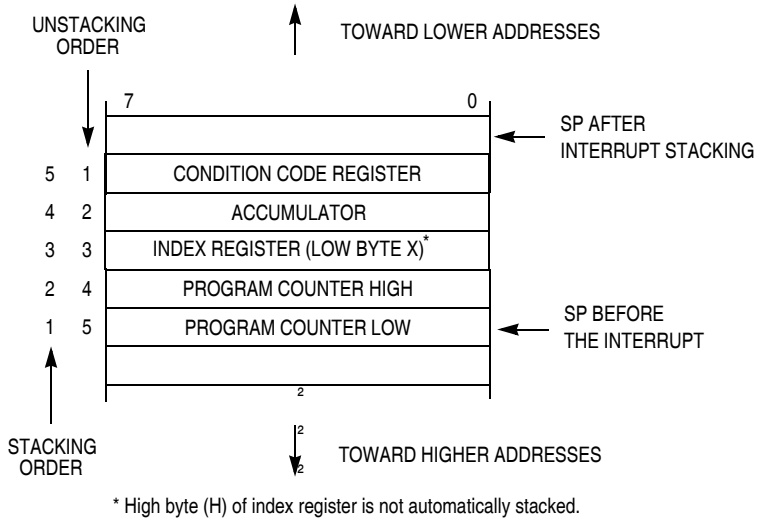



**Figure 5-1. Interrupt Stack Frame**

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

The status flag causing the interrupt must be acknowledged (cleared) before returning from the ISR. Typically, the flag is 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 Pin ($\overline{\text{IRQ}}$)

External interrupts are managed by the IRQ status and control register, IRQSC. When the $\overline{\text{IRQ}}$ 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 $\overline{\text{IRQ}}$ (if enabled) can wake the MCU.

#### 5.5.2.1 Pin Configuration Options

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

The $\overline{\text{IRQ}}$ pin, when enabled, defaults to use an internal pullup device (IRQPDD = 0). If the user desires to use an external pullup, the IRQPDD can be written to a 1 to turn off the internal device.

BIH and BIL instructions may be used to detect the level on the $\overline{\text{IRQ}}$ pin when the pin is configured to act as the IRQ input.

**NOTE**

This pin does not contain a clamp diode to $V_{DD}$ and should not be driven above $V_{DD}$.

The voltage measured on the internally pulled-up $\overline{\text{IRQ}}$ pin will not be pulled to $V_{DD}$. The internal gates connected to this pin are pulled to $V_{DD}$. The $\overline{\text{IRQ}}$ pullup should not be used to pull up components external to the MCU. The internal gates connected to this pin are pulled all the way to $V_{DD}$.

#### 5.5.2.2 Edge and Level Sensitivity

The IRQMOD control bit 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 $\overline{\text{IRQ}}$ pin changes from the deasserted to the asserted level), but the flag is continuously set (and cannot be cleared) as long as the $\overline{\text{IRQ}}$ pin remains at the asserted level.

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

Table 5-2 provides a summary of all interrupt sources. Higher-priority sources are located toward the bottom of the table. The high-order byte of the address for the interrupt service routine is located at the first address in the vector address column, and the low-order byte of the address for the interrupt service routine is located at the next higher address.

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

**Table 5-2. Vector Summary**

| Vector Priority | Vector Number | Address (High:Low) | Vector Name | Module | Source | Enable | Description |
|---|---|---|---|---|---|---|---|
| Lower | 31 through 24 | 0xFFC0:FFC1 through 0xFFCE:FFCF | Unused Vector Space (available for user program) | | | | |
| | 23 | 0xFFD0:FFD1 | Vrti | System control | RTIF | RTIE | Real-time interrupt |
| | 22 | 0xFFD2:FFD3 | — | — | — | — | — |
| | 21 | 0xFFD4:FFD5 | — | — | — | — | — |
| | 20 | 0xFFD6:FFD7 | Vacmp | ACMP | ACF | ACIE | ACMP |
| | 19 | 0xFFD8:FFD9 | Vadc | ADC | COCO | AIEN | ADC |
| | 18 | 0xFFDA:FFDB | Vkeyboard | KBI | KBF | KBIE | Keyboard pins |
| | 17 | 0xFFDC:FFDD | Viic | IIC | IICIF | IICIE | IIC control |
| | 16 | 0xFFDE:FFDF | Vscitx | SCI | TDRE<br>TC | TIE<br>TCIE | SCI transmit |
| | 15 | 0xFFE0:FFE1 | Vscirx | SCI | IDLE<br>RDRF | ILIE<br>RIE | SCI receive |
| | 14 | 0xFFE2:FFE3 | Vscierr | SCI | OR<br>NF<br>FE<br>PF | ORIE<br>NFIE<br>FEIE<br>PFIE | SCI error |
| | 13 | 0xFFE4:FFE5 | Vspi | SPI | SPIF<br>MODF<br>SPTEF | SPIE<br>SPIE<br>SPTIE | SPI |
| | 12 | 0xFFE6:FFE7 | Vmtim | MTIM | TOF | TOIE | MTIM |
| | 11 | 0xFFE8:FFE9 | — | — | — | — | — |
| | 10 | 0xFFEA:FFEB | — | — | — | — | — |
| | 9 | 0xFFEC:FFED | — | — | — | — | — |
| | 8 | 0xFFEE:FFEF | — | — | — | — | — |
| | 7 | 0xFFF0:FFF1 | Vtpmovf | TPM | TOF | TOIE | TPM overflow |
| | 6 | 0xFFF2:FFF3 | Vtpmch1 | TPM | CH1F | CH1IE | TPM channel 1 |
| | 5 | 0xFFF4:FFF5 | Vtpmch0 | TPM | CH0F | CH0IE | TPM channel 0 |
| | 4 | 0xFFF6:FFF7 | — | — | — | — | — |
| | 3 | 0xFFF8:FFF9 | Vlvd | System control | LVDF | LVDIE | Low-voltage detect |
| | 2 | 0xFFFA:FFFB | Virq | IRQ | IRQF | IRQIE | $\overline{\text{IRQ}}$ pin |
| | 1 | 0xFFFC:FFFD | Vswi | CPU | SWI Instruction | — | Software interrupt |
| Higher | 0 | 0xFFFE:FFFF | Vreset | System control | COP<br>LVD<br>$\overline{\text{RESET}}$ pin<br>Illegal opcode<br>Illegal address<br>POR | COPE<br>LVDRE<br>RSTPE<br>—<br>—<br>— | Watchdog timer<br>Low-voltage detect<br>External pin<br>Illegal opcode<br>Illegal address<br>power-on-reset |

## 5.6 Low-Voltage Detect (LVD) System

The MC9S08QG8/4 includes a system to protect against low voltage conditions to protect memory contents and control MCU system states during supply voltage variations. The system is comprised of a power-on reset (POR) circuit and a LVD circuit with a user selectable trip voltage, either high ($V_{LVDH}$) or low ($V_{LVDL}$). The LVD circuit is enabled when LVDE in SPMSC1 is high and the trip voltage is selected by LVDV in SPMSC3. The LVD is disabled upon entering any of the stop modes unless LVDSE is set in SPMSC1. If LVDSE and LVDE are both set, then the MCU cannot enter stop1 or stop2, and the current consumption in stop3 with the LVD enabled will be greater.

### 5.6.1 Power-On Reset Operation

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

### 5.6.2 LVD Reset Operation

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

### 5.6.3 LVD Interrupt Operation

When a low voltage condition is detected and the LVD circuit is configured using SPMSC1 for interrupt operation (LVDE set, LVDIE set, and LVDRE clear), then LVDF in SPMSC1 will be set and an LVD interrupt request will occur.

### 5.6.4 Low-Voltage Warning (LVW)

The LVD system has a low voltage warning flag to indicate to the user that the supply voltage is approaching, but is above, the LVD voltage. The LVW does not have an interrupt associated with it. There are two user selectable trip voltages for the LVW, one high ($V_{LVWH}$) and one low ($V_{LVWL}$). The trip voltage is selected by LVWV in SPMSC3.

## 5.7 Real-Time Interrupt (RTI)

The real-time interrupt function can be used to generate periodic interrupts. The RTI can accept two sources of clocks, the 1-kHz internal clock or an external clock if available. External clock input requires the XOSC module; consult Table 1-1 to see if your MCU contains this module. The RTICLKS bit in SRTISC is used to select the RTI clock source.

Either RTI clock source can be used when the MCU is in run, wait or stop3 mode. When using the external oscillator in stop3, it must be enabled in stop (EREFSTEN = 1) and configured for low frequency operation

(RANGE = 0). Only the internal 1-kHz clock source can be selected to wake the MCU from stop1 or stop2 modes.

The SRTISC register includes a read-only status flag, a write-only acknowledge bit, and a 3-bit control value (RTIS) used to select one of seven wakeup periods. The RTI has a local interrupt enable, RTIE, to allow masking of the real-time interrupt. The RTI can be disabled by writing each bit of RTIS to 0s, and no interrupts will be generated. See Section 5.8.7, "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 eight 8-bit registers in the high-page register space are related to reset and interrupt systems.

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

Some control bits in the SOPT1, SOPT2, 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 status and control bits, which are used to configure the IRQ function, report status, and acknowledge IRQ events.

| | 7 | 6 | 5[1] | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | IRQPDD | 0 | IRQPE | IRQF | 0 | IRQIE | IRQMOD |
| W | | | | | | IRQACK | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

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

1 Bit 5 is a reserved bit that must always be written to 0.

**Table 5-3. IRQSC Register Field Descriptions**

| Field | Description |
|---|---|
| 6<br>IRQPDD | **Interrupt Request ($\overline{\text{IRQ}}$) Pull Device Disable** — This read/write control bit is used to disable the internal pullup device when the $\overline{\text{IRQ}}$ pin is enabled (IRQPE = 1) allowing for an external device to be used.<br>0 $\overline{\text{IRQ}}$ pull device enabled if IRQPE = 1.<br>1 $\overline{\text{IRQ}}$ pull device disabled if IRQPE = 1. |
| 4<br>IRQPE | **$\overline{\text{IRQ}}$ Pin Enable** — This read/write control bit enables the $\overline{\text{IRQ}}$ pin function. When this bit is set the $\overline{\text{IRQ}}$ pin can be used as an interrupt request.<br>0 $\overline{\text{IRQ}}$ pin function is disabled.<br>1 $\overline{\text{IRQ}}$ 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 $\overline{\text{IRQ}}$ pin remains at its asserted level. |
| 1<br>IRQIE | **IRQ Interrupt Enable** — This read/write control bit determines whether IRQ events generate an interrupt request.<br>0 Interrupt request when IRQF set is disabled (use polling).<br>1 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. See Section 5.5.2.2, “Edge and Level Sensitivity,” for more details.<br>0 IRQ event on falling edges only.<br>1 IRQ event on falling edges and low levels. |

## 5.8.2 System Reset Status Register (SRS)

This high page register includes 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, all of the status bits in SRS will be cleared. 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 | 0 | LVD | 0 |
| W | Writing any value to SRS address clears COP watchdog timer. | | | | | | | |
| POR: | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| LVD: | u(1) | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Any other reset: | 0 | Note (2) | Note (2) | Note (2) | Note (2) | 0 | 0 | 0 |

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

1 u = unaffected

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

**Table 5-4. SRS Register 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 (LVD) status bit is also set to indicate that the reset occurred while the internal supply was below the LVD 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 can 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 SOPT1 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** — Reset was caused by an attempt to access either data or an instruction at an unimplemented memory address.<br>0 Reset not caused by an illegal address<br>1 Reset caused by an illegal address |
| 1<br>LVD | **Low Voltage Detect** — If the LVDRE bit is set and the supply drops below the LVD trip voltage, an LVD reset will occur. 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 high page register contains a single write-only control bit. A serial background command such as WRITE_BYTE must be used to write to SBDFR. Attempts to write this register from a user program are ignored. Reads always return 0x00.

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

= Unimplemented or Reserved

**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 5-5. SBDFR Register Field Descriptions**

| Field | Description |
|---|---|
| 0<br>BDFR | **Background Debug Force Reset** — A serial background command such as WRITE_BYTE can 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. To enter user mode, PTA4/ACMPO/BKGD/MS must be high immediately after issuing WRITE_BYTE command. To enter BDM, PTA4/ACMPO/BKGD/MS must be low immediately after issuing WRITE_BYTE command. See Table A-9., “Control Timing,” for more information. |

## 5.8.4 System Options Register 1 (SOPT1)

This high page register is a write-once register so only the first write after reset is honored. It can be read at any time. Any subsequent attempt to write to SOPT1 (intentionally or unintentionally) is ignored to avoid accidental changes to these sensitive settings. SOPT1 must be written during the user reset initialization program to set the desired controls even if the desired settings are the same as the reset settings.

| | 7 | 6 | 5 | 4[1] | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | COPE | COPT | STOPE | | 0 | 0 | BKGDPE | RSTPE |
| W | | | | | | | | |
| Reset: | 1 | 1 | 0 | 1 | 0 | 0 | 1 | u[2] |
| POR: | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| LVD: | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

= Unimplemented or Reserved

**Figure 5-5. System Options Register 1 (SOPT1)**

1 Bit 4 is reserved; writes will change the value but will have no effect on this MCU.

2 u = unaffected

**Table 5-6. SOPT1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>COPE | **COP Watchdog Enable** — This write-once bit selects whether the COP watchdog is enabled.<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 selects the timeout period of the COP. COPT along with COPCLKS in SOPT2 defines the COP timeout period.<br>0 Short timeout period selected.<br>1 Long timeout period selected. |
| 5<br>STOPE | **Stop Mode Enable** — This write-once bit is used to enable 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** — This write-once bit when set enables the PTA4/ACMPO/BKGD/MS pin to function as BKGD/MS. When clear, the pin functions as one of its output only alternative functions. This pin defaults to the BKGD/MS function following any MCU reset.<br>0 PTA4/ACMPO/BKGD/MS pin functions as PTA4 or ACMPO.<br>1 PTA4/ACMPO/BKGD/MS pin functions as BKGD/MS. |
| 0<br>RSTPE | **$\overline{\text{RESET}}$ Pin Enable** — This write-once bit when set enables the PTA5/$\overline{\text{IRQ}}$/TCLK/$\overline{\text{RESET}}$ pin to function as $\overline{\text{RESET}}$. When clear, the pin functions as one of its input only alternative functions. This pin defaults to its input-only port function following an MCU POR. When RSTPE is set, an internal pullup device is enabled on $\overline{\text{RESET}}$.<br>0 PTA5/$\overline{\text{IRQ}}$/TCLK/$\overline{\text{RESET}}$ pin functions as PTA5, $\overline{\text{IRQ}}$, or TCLK.<br>1 PTA5/$\overline{\text{IRQ}}$/TCLK/$\overline{\text{RESET}}$ pin functions as $\overline{\text{RESET}}$. |

## 5.8.5 System Options Register 2 (SOPT2)

This high page register contains bits to configure MCU specific features on the MC9S08QG8/4 devices.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | COPCLKS[1] | 0 | 0 | 0 | 0 | 0 | IICPS | ACIC |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

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

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

**Table 5-7. SOPT2 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>COPCLKS | **COP Watchdog Clock Select** — This write-once bit selects the clock source of the COP watchdog.<br>0 Internal 1-kHz clock is source to COP.<br>1 Bus clock is source to COP. |
| 1<br>IICPS | **IIC Pin Select**— This bit selects the location of the SDA and SCL pins of the IIC module.<br>0 SDA on PTA2, SCL on PTA3.<br>1 SDA on PTB6, SCL on PTB7. |
| 0<br>ACIC | **Analog Comparator to Input Capture Enable**— This bit connects the output of ACMP to TPM input channel 0.<br>0 ACMP output not connected to TPM input channel 0.<br>1 ACMP output connected to TPM input channel 0. |

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

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

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

= Unimplemented or Reserved

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

**Table 5-8. SDIDH Register Field Descriptions**

| Field | Description |
|---|---|
| 7:4<br>Reserved | **Bits 7:4 are reserved. Reading these bits will result in an indeterminate value; writes have no effect.** |
| 3:0<br>ID[11:8] | **Part Identification Number** — Each derivative in the HCS08 Family has a unique identification number. The MC9S08QG8 is hard coded to the value 0x009. See also ID bits in Table 5-9. |

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

= Unimplemented or Reserved

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

**Table 5-9. SDIDL Register 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 MC9S08QG8 is hard coded to the value 0x009. See also ID bits in Table 5-8. |

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

This high page register contains status and control bits for the RTI.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | RTIF | 0 | RTICLKS | RTIE | 0 | RTIS | | |
| W | | RTIACK | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

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

**Table 5-10. SRTISC Register 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 | **Real-Time Interrupt Delay Selects** — These read/write bits select the period for the RTI. See Table 5-11. |

**Table 5-11. Real-Time Interrupt Period**

| RTIS2:RTIS1:RTIS0 | Using Internal 1-kHz Clock Source[1] [2] | Using External Clock Source Period = $t_{ext}$[3] |
|---|---|---|
| 0:0:0 | Disable RTI | Disable RTI |
| 0:0:1 | 8 ms | $t_{ext}$ x 256 |
| 0:1:0 | 32 ms | $t_{ext}$ x 1024 |
| 0:1:1 | 64 ms | $t_{ext}$ x 2048 |
| 1:0:0 | 128 ms | $t_{ext}$ 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 | $t_{ext}$ x 32768 |

1 Values are shown in this column based on $t_{RTI}$ = 1 ms. See $t_{RTI}$ in the appendix Section A.8.1, “Control Timing,” for the tolerance of this value.

2 The initial RTI timeout period will be up to one 1-kHz clock period less than the time specified.

3 $t_{ext}$ is the period of the external crystal frequency.

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

This high page register contains status and control bits to support the low voltage detect function and to enable the bandgap voltage reference for use by the ADC module. To configure the low voltage detect trip voltage, see Table 5-14 for the LVDV bit description in SPMSC3.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1[1] | 0 |
|---|---|---|---|---|---|---|---|---|
| R | LVDF | 0 | LVDIE | LVDRE[2] | LVDSE | LVDE[2] | 0 | BGBE |
| W | | LVDACK | | | | | | |
| Reset: | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

= Unimplemented or Reserved

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

1 Bit 1 is a reserved bit that must always be written to 0.

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

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

| Field | Description |
|---|---|
| 7<br>LVDF | **Low-Voltage Detect Flag** — Provided LVDE = 1, this read-only status bit indicates a low-voltage detect event. |
| 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 0. |
| 5<br>LVDIE | **Low-Voltage Detect Interrupt Enable** — This 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>LVDRE | **Low-Voltage Detect Reset Enable** — This write-once bit enables LVDF events to generate a hardware reset (provided LVDE = 1).<br>0 LVDF does not generate hardware resets.<br>1 Force an MCU reset when LVDF = 1. |
| 3<br>LVDSE | **Low-Voltage Detect Stop Enable** — Provided LVDE = 1, this read/write bit determines whether the low-voltage detect function operates when the MCU is in stop mode.<br>0 Low-voltage detect disabled during stop mode.<br>1 Low-voltage detect enabled during stop mode. |
| 2<br>LVDE | **Low-Voltage Detect Enable** — This write-once bit enables low-voltage detect logic and qualifies the operation of other bits in this register.<br>0 LVD logic disabled.<br>1 LVD logic enabled. |
| 0<br>BGBE | **Bandgap Buffer Enable** — This bit enables an internal buffer for the bandgap voltage reference for use by the ADC module on one of its internal channels or as a voltage reference for ACMP module.<br>0 Bandgap buffer disabled.<br>1 Bandgap buffer enabled. |

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

This high page register contains status and control bits to configure the stop mode behavior of the MCU. See Section 3.6, "Stop Modes," for more information on stop modes.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | PDF | PPDF | 0 | PDC[1] | PPDC[1] |
| W | | | | | | PPDACK | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

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

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

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

| Field | Description |
|---|---|
| 4<br>PDF | **Power Down Flag** — This read-only status bit indicates the MCU has recovered from stop1 mode.<br>0 MCU has not recovered from stop1 mode.<br>1 MCU recovered from stop1 mode. |
| 3<br>PPDF | **Partial Power Down Flag** — This read-only status bit indicates that the MCU has recovered from stop2 mode.<br>0 MCU has not recovered from stop2 mode.<br>1 MCU recovered from stop2 mode. |
| 2<br>PPDACK | **Partial Power Down Acknowledge** — Writing a 1 to PPDACK clears the PPDF and the PDF bits. |
| 1<br>PDC | **Power Down Control** — The 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 PPDC bit controls which power down mode is selected.<br>0 Stop1 full power down mode enabled if PDC set.<br>1 Stop2 partial power down mode enabled if PDC set. |

## 5.8.10 System Power Management Status and Control 3 Register (SPMSC3)

This high page register is used to report the status of the low voltage warning function and to select the low voltage detect trip voltage.

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

= Unimplemented or Reserved  U= Unaffected by reset

**Figure 5-12. System Power Management Status and Control 3 Register (SPMSC3)**

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 5-14. SPMSC3 Register 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. |
| 5<br>LVDV | **Low-Voltage Detect Voltage Select** — The LVDV bit selects the LVD trip point voltage ($V_{LVD}$).<br>0 Low trip point selected ($V_{LVD} = V_{LVDL}$).<br>1 High trip point selected ($V_{LVD} = V_{LVDH}$). |
| 4<br>LVWV | **Low-Voltage Warning Voltage Select** — The LVWV bit selects the LVW trip point voltage ($V_{LVW}$).<br>0 Low trip point selected ($V_{LVW} = V_{LVWL}$).<br>1 High trip point selected ($V_{LVW} = V_{LVWH}$). |

# Chapter 6
# Parallel Input/Output Control

This section explains software controls related to parallel input/output (I/O) and pin control. The MC9S08QG8 has two parallel I/O ports which include a total of 12 I/O pins, one output-only pin and one input-only pin. See Section Chapter 2, “External Signal Description,” for more information about pin assignments and external hardware considerations of these pins. Not all pins are available on all devices of the MC9S08QG8/4 Family; see Table 1-1 for the number of general-purpose pins available on your device.

All of these I/O pins are shared with on-chip peripheral functions as shown in Table 2-2. The peripheral modules have priority over the I/Os so that when a peripheral is enabled, the I/O functions associated with the shared pins are disabled. After reset, the shared peripheral functions are disabled so that the pins are controlled by the I/O. All of the I/Os are configured as inputs (PTxDDn = 0) with pullup devices disabled (PTxPEn = 0), except for output-only pin PTA4 which defaults to the BKGD/MS pin.

**NOTE**

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

## 6.1 Port Data and Data Direction

Reading and writing of parallel I/Os is performed through the port data registers. The direction, either input or output, is controlled through the port data direction registers. The parallel I/O port function for an individual pin is illustrated in the block diagram shown in Figure 6-1.

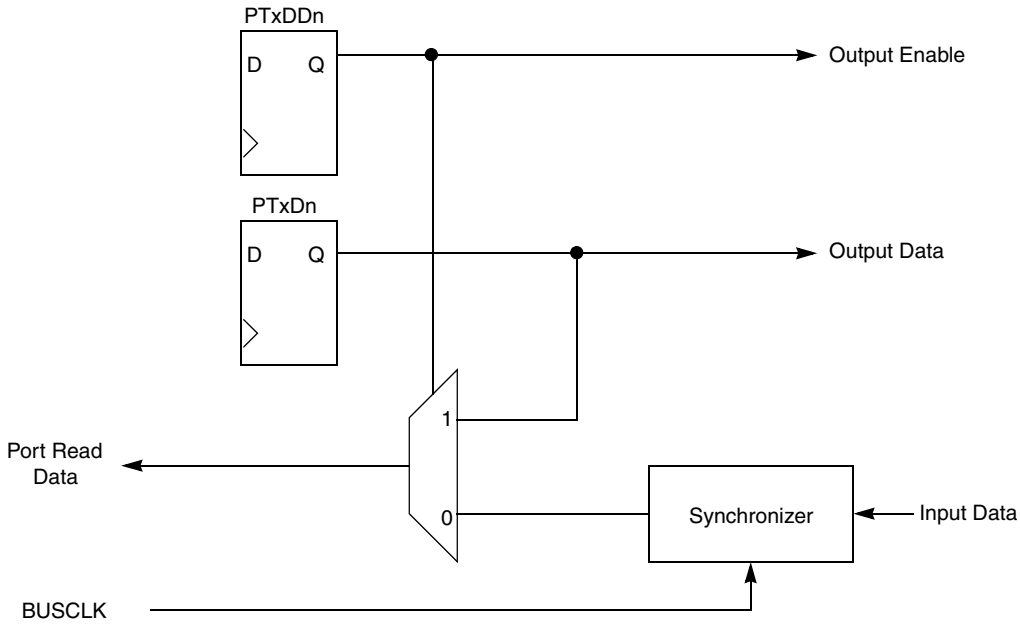



**Figure 6-1. Parallel I/O Block Diagram**

The data direction control bit (PTxDDn) determines whether the output buffer for the associated pin is enabled, and also controls the source for port data register reads. The input buffer for the associated pin is always enabled unless the pin is enabled as an analog function or is an output-only pin.

When a shared digital function is enabled for a pin, the output buffer is controlled by the shared function. However, the data direction register bit will continue to control the source for reads of the port data register.

When a shared analog function is enabled for a pin, both the input and output buffers are disabled. A value of 0 is read for any port data bit where the bit is an input (PTxDDn = 0) and the input buffer is disabled. In general, whenever a pin is shared with both an alternate digital function and an analog function, the analog function has priority such that if both the digital and analog functions are enabled, the analog function controls the pin.

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

## 6.2 Pin Control — Pullup, Slew Rate, and Drive Strength

Associated with the parallel I/O ports is a set of registers located in the high page register space that operate independently of the parallel I/O registers. These registers are used to control pullups, slew rate, and drive strength for the pins.

## 6.3 Pin Behavior in Stop Modes

Pin behavior following execution of a STOP instruction depends on the stop mode that is entered. An explanation of pin behavior for the various stop modes follows:

- In stop1 mode, all internal registers including parallel I/O control and data registers are powered off. Each of the pins assumes its default reset state (output buffer and internal pullup disabled). Upon exit from stop1, all pins must be re-configured the same as if the MCU had been reset by POR.
- Stop2 mode is a partial power-down mode, whereby latches maintain the pin state as before the STOP instruction was executed. CPU register status and the state of I/O registers must be saved in RAM before the STOP instruction is executed to place the MCU in stop2 mode. Upon recovery from stop2 mode, before accessing any I/O, the user must examine the state of the PPDF bit in the SPMSC2 register. If the PPDF bit is 0, I/O must be initialized as if a power on reset had occurred. If the PPDF bit is 1, I/O data previously stored in RAM, before the STOP instruction was executed, and peripherals previously enabled will require being initialized and restored to their pre-stop condition. The user must then write a 1 to the PPDACK bit in the SPMSC2 register. Access of pins is now permitted again in the user application program.
- In stop3 mode, all pin states are maintained because internal logic stays powered up. Upon recovery, all pin functions are the same as before entering stop3.

## 6.4 Parallel I/O Registers

### 6.4.1 Port A Registers

This section provides information about the registers associated with the parallel I/O ports.

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

#### 6.4.1.1 Port A Data (PTAD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTAD5[1] | PTAD4[2] | PTAD3 | PTAD2 | PTAD1 | PTAD0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

1 Reads of bit PTAD5 always return the pin value of PTA5, regardless of the value stored in bit PTADD5.

2 Reads of bit PTAD4 always return the contents of PTAD4, regardless of the value stored in bit PTADD4.

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

| Field | Description |
|---|---|
| 5:0<br>PTAD[5:0] | **Port A Data Register Bits** — For port A pins that are inputs, reads return the logic level on the pin. For port A pins that are configured as outputs, reads return the last value written to this register.<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. |

### 6.4.1.2 Port A Data Direction (PTADD)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTADD5[1] | PTADD4[2] | PTADD3 | PTADD2 | PTADD1 | PTADD0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

1 PTADD5 has no effect on the input-only PTA5 pin.

2 PTADD4 has no effect on the output-only PTA4 pin.

**Table 6-2. PTADD Register Field Descriptions**

| Field | Description |
|---|---|
| 5:0<br>PTADD[5:0] | **Data Direction for Port A Bits** — These read/write bits control the direction of port A pins and what is read for PTAD reads.<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. |

## 6.4.2 Port A Control Registers

The pins associated with port A are controlled by the registers in this section. These registers control the pin pullup, slew rate, and drive strength of the port A pins independent of the parallel I/O register.

### 6.4.2.1 Port A Internal Pullup Enable (PTAPE)

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTAPE5 | PTAPE4[1] | PTAPE3 | PTAPE2 | PTAPE1 | PTAPE0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

1 PTAPE4 has no effect on the output-only PTA4 pin.

**Table 6-3. PTAPE Register Field Descriptions**

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

### 6.4.2.2 Port A Slew Rate Enable (PTASE)

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTASE5[1] | PTASE4 | PTASE3 | PTASE2 | PTASE1 | PTASE0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 6-6. Slew Rate Enable for Port A Register (PTASE)**

1 PTASE5 has no effect on the input-only PTA5 pin.

**Table 6-4. PTASE Register Field Descriptions**

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

### 6.4.2.3 Port A Drive Strength Select (PTADS)

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

### 6.4.2.4 Port A Drive Strength Select (PTADS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | PTADS5[1] | PTADS4 | PTADS3 | PTADS2 | PTADS1 | PTADS0 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-8. Drive Strength Selection for Port A Register (PTADS)**

1 PTADS5 has no effect on the input-only PTA5 pin.

**Table 6-5. PTADS Register Field Descriptions**

| Field | Description |
|---|---|
| 5:0<br>PTADS[5:0] | **Output Drive Strength Selection for Port A Bits** — Each of these control bits selects between low and high output drive for the associated PTA pin. For port A pins that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for port A bit n.<br>1 High output drive strength selected for port A bit n. |

### 6.4.3 Port B Registers

This section provides information about the registers associated with the parallel I/O ports.

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

#### 6.4.3.1 Port B Data (PTBD)

| | 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 6-10. Port B Data Register (PTBD)**

**Table 6-6. PTBD Register 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 the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled. |

#### 6.4.3.2 Port B Data Direction (PTBDD)

| | 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 6-7. PTBDD Register 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. |

## 6.4.4 Port B Control Registers

The pins associated with port B are controlled by the registers in this section. These registers control the pin pullup, slew rate, and drive strength of the port B pins independent of the parallel I/O register.

### 6.4.4.1 Port B Internal Pullup Enable (PTBPE)

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

| | 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 6-12. Internal Pullup Enable for Port B Register (PTBPE)**

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

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

### 6.4.4.2 Port B Slew Rate Enable (PTBSE)

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBSE7 | PTBSE6 | PTBSE5 | PTBSE4 | PTBSE3 | PTBSE2 | PTBSE1 | PTBSE0 |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 6-14. Slew Rate Enable for Port B Register (PTBSE)**

**Table 6-9. PTBSE Register Field Descriptions**

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

### 6.4.4.3 Port B Drive Strength Select (PTBDS)

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | PTBDS7 | PTBDS6 | PTBDS5 | PTBDS4 | PTBDS3 | PTBDS2 | PTBDS1 | PTBDS0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 6-16. Drive Strength Selection for Port B Register (PTBDS)**

**Table 6-10. PTBDS Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>PTBDS[7:0] | **Output Drive Strength Selection for Port B Bits** — Each of these control bits selects between low and high output drive for the associated PTB pin. For port B pins that are configured as inputs, these bits have no effect.<br>0 Low output drive strength selected for port B bit n.<br>1 High output drive strength selected for port B bit n. |

# Chapter 7
# Central Processor Unit (S08CPUV2)

## 7.1 Introduction

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

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

### 7.1.1 Features

Features of the HCS08 CPU include:

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

## 7.2 Programmer's Model and CPU Registers

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

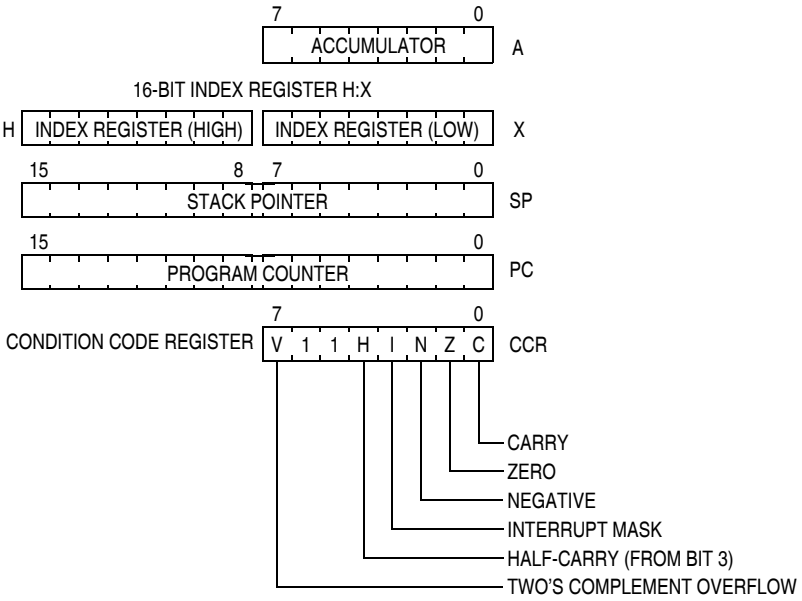



**Figure 7-1. CPU Registers**

### 7.2.1 Accumulator (A)

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

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

### 7.2.2 Index Register (H:X)

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

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

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

### 7.2.3 Stack Pointer (SP)

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

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

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

### 7.2.4 Program Counter (PC)

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

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

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

### 7.2.5 Condition Code Register (CCR)

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

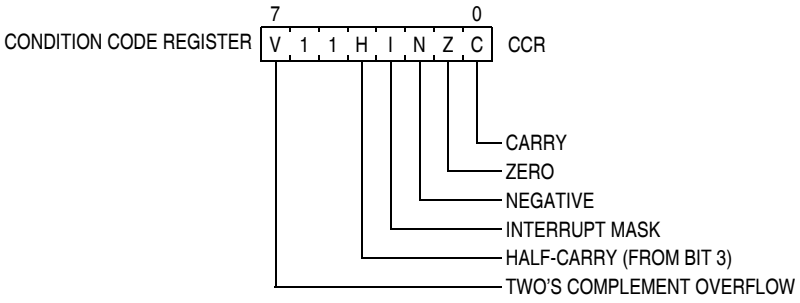



**Figure 7-2. Condition Code Register**

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

| Field | Description |
|---|---|
| 7<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

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

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

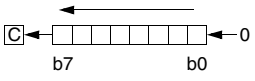
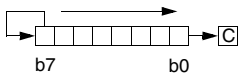
| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | VH | I N Z C |
| ADC #*opr8i*<br>ADC *opr8a*<br>ADC *opr16a*<br>ADC *oprx16*,X<br>ADC *oprx8*,X<br>ADC ,X<br>ADC *oprx16*,SP<br>ADC *oprx8*,SP | Add with Carry<br>A ← (A) + (M) + (C) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A9 ii<br>B9 dd<br>C9 hh ll<br>D9 ee ff<br>E9 ff<br>F9<br>9E D9 ee ff<br>9E E9 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕↕ | – ↕ ↕ ↕ |
| 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<br>A ← (A) + (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AB ii<br>BB dd<br>CB hh ll<br>DB ee ff<br>EB ff<br>FB<br>9E DB ee ff<br>9E EB ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕↕ | – ↕ ↕ ↕ |
| AIS #*opr8i* | Add Immediate Value (Signed) to Stack Pointer<br>SP ← (SP) + (M) | IMM | A7 ii | 2 | pp | – – | – – – – |
| AIX #*opr8i* | Add Immediate Value (Signed) to Index Register (H:X)<br>H:X ← (H:X) + (M) | IMM | AF ii | 2 | pp | – – | – – – – |
| 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<br>A ← (A) & (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A4 ii<br>B4 dd<br>C4 hh ll<br>D4 ee ff<br>E4 ff<br>F4<br>9E D4 ee ff<br>9E E4 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 – | – ↕ ↕ – |
| ASL *opr8a*<br>ASLA<br>ASLX<br>ASL *oprx8*,X<br>ASL ,X<br>ASL *oprx8*,SP | Arithmetic Shift Left<br>  <br><br>(Same as LSL) | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38 dd<br>48<br>58<br>68 ff<br>78<br>9E 68 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ – | – ↕ ↕ ↕ |
| ASR *opr8a*<br>ASRA<br>ASRX<br>ASR *oprx8*,X<br>ASR ,X<br>ASR *oprx8*,SP | Arithmetic Shift Right<br>  <br> | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 37 dd<br>47<br>57<br>67 ff<br>77<br>9E 67 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ – | – ↕ ↕ ↕ |
| BCC *rel* | Branch if Carry Bit Clear<br>(if C = 0) | REL | 24 rr | 3 | ppp | – – | – – – – |

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

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

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

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

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

| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | VH | I N Z C |
| CMP #*oprx8i*<br>CMP *oprx8a*<br>CMP *oprx16a*<br>CMP *oprx16*,X<br>CMP *oprx8*,X<br>CMP ,X<br>CMP *oprx16*,SP<br>CMP *oprx8*,SP | Compare Accumulator with Memory<br>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 ii<br>B1 dd<br>C1 hh ll<br>D1 ee ff<br>E1 ff<br>F1<br>9E D1 ee ff<br>9E E1 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕ – | – ↕ ↕ ↕ |
| COM *oprx8a*<br>COMA<br>COMX<br>COM *oprx8*,X<br>COM ,X<br>COM *oprx8*,SP | Complement (One's Complement)<br>M ← ($\overline{M}$)= $FF – (M)<br>A ← ($\overline{A}$) = $FF – (A)<br>X ← ($\overline{X}$) = $FF – (X)<br>M ← ($\overline{M}$) = $FF – (M)<br>M ← ($\overline{M}$) = $FF – (M)<br>M ← ($\overline{M}$) = $FF – (M) | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 33 dd<br>43<br>53<br>63 ff<br>73<br>9E 63 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | 0 – | – ↕ ↕ 1 |
| CPHX *oprx16a*<br>CPHX #*oprx16i*<br>CPHX *oprx8a*<br>CPHX *oprx8*,SP | Compare Index Register (H:X) with Memory<br>(H:X) – (M:M + $0001)<br>(CCR Updated But Operands Not Changed) | EXT<br>IMM<br>DIR<br>SP1 | 3E hh ll<br>65 jj kk<br>75 dd<br>9E F3 ff | 6<br>3<br>5<br>6 | prrfpp<br>ppp<br>rrfpp<br>prrfpp | ↕ – | – ↕ ↕ ↕ |
| CPX #*oprx8i*<br>CPX *oprx8a*<br>CPX *oprx16a*<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<br>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 ii<br>B3 dd<br>C3 hh ll<br>D3 ee ff<br>E3 ff<br>F3<br>9E D3 ee ff<br>9E E3 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕ – | – ↕ ↕ ↕ |
| DAA | Decimal Adjust Accumulator<br>After ADD or ADC of BCD Values | INH | 72 | 1 | p | U – | – ↕ ↕ ↕ |
| DBNZ *oprx8a*,*rel*<br>DBNZA *rel*<br>DBNZX *rel*<br>DBNZ *oprx8*,X,*rel*<br>DBNZ ,X,*rel*<br>DBNZ *oprx8*,SP,*rel* | Decrement A, X, or M and Branch if Not Zero (if (result) ≠ 0)<br>DBNZX Affects X Not H | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3B dd rr<br>4B rr<br>5B rr<br>6B ff rr<br>7B rr<br>9E 6B ff rr | 7<br>4<br>4<br>7<br>6<br>8 | rfwpppp<br>fppp<br>fppp<br>rfwpppp<br>rfwppp<br>prfwpppp | – – | – – – – |
| DEC *oprx8a*<br>DECA<br>DECX<br>DEC *oprx8*,X<br>DEC ,X<br>DEC *oprx8*,SP | Decrement<br>M ← (M) – $01<br>A ← (A) – $01<br>X ← (X) – $01<br>M ← (M) – $01<br>M ← (M) – $01<br>M ← (M) – $01 | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3A dd<br>4A<br>5A<br>6A ff<br>7A<br>9E 6A ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ – | – ↕ ↕ – |
| DIV | Divide<br>A ← (H:A)÷(X); H ← Remainder | INH | 52 | 6 | fffffp | – – | – – ↕ ↕ |
| EOR #*oprx8i*<br>EOR *oprx8a*<br>EOR *oprx16a*<br>EOR *oprx16*,X<br>EOR *oprx8*,X<br>EOR ,X<br>EOR *oprx16*,SP<br>EOR *oprx8*,SP | Exclusive OR Memory with Accumulator<br>A ← (A ⊕ M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A8 ii<br>B8 dd<br>C8 hh ll<br>D8 ee ff<br>E8 ff<br>F8<br>9E D8 ee ff<br>9E E8 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 – | – ↕ ↕ – |

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

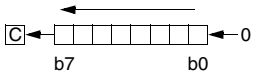
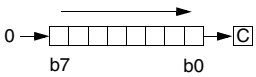
| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | VH | I N Z C |
| INC *opr8a*<br>INCA<br>INCX<br>INC *oprx8*,X<br>INC ,X<br>INC *oprx8*,SP | Increment M ← (M) + $01<br>A ← (A) + $01<br>X ← (X) + $01<br>M ← (M) + $01<br>M ← (M) + $01<br>M ← (M) + $01 | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3C dd<br>4C<br>5C<br>6C ff<br>7C<br>9E 6C ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ – | – ↕ ↕ – |
| JMP *opr8a*<br>JMP *opr16a*<br>JMP *oprx16*,X<br>JMP *oprx8*,X<br>JMP ,X | Jump<br>PC ← Jump Address | DIR<br>EXT<br>IX2<br>IX1<br>IX | BC dd<br>CC hh ll<br>DC ee ff<br>EC ff<br>FC | 3<br>4<br>4<br>3<br>3 | ppp<br>pppp<br>pppp<br>ppp<br>ppp | – – | – – – – |
| JSR *opr8a*<br>JSR *opr16a*<br>JSR *oprx16*,X<br>JSR *oprx8*,X<br>JSR ,X | Jump to Subroutine<br>PC ← (PC) + *n* (*n* = 1, 2, or 3)<br>Push (PCL); SP ← (SP) – $0001<br>Push (PCH); SP ← (SP) – $0001<br>PC ← Unconditional Address | DIR<br>EXT<br>IX2<br>IX1<br>IX | BD dd<br>CD hh ll<br>DD ee ff<br>ED ff<br>FD | 5<br>6<br>6<br>5<br>5 | ssppp<br>pssppp<br>pssppp<br>ssppp<br>ssppp | – – | – – – – |
| 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<br>A ← (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A6 ii<br>B6 dd<br>C6 hh ll<br>D6 ee ff<br>E6 ff<br>F6<br>9E D6 ee ff<br>9E E6 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 – | – ↕ ↕ – |
| 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)<br>H:X ← (M:M + $0001) | IMM<br>DIR<br>EXT<br>IX<br>IX2<br>IX1<br>SP1 | 45 jj kk<br>55 dd<br>32 hh ll<br>9E AE<br>9E BE ee ff<br>9E CE ff<br>9E FE ff | 3<br>4<br>5<br>5<br>6<br>5<br>5 | ppp<br>rrpp<br>prrpp<br>prrfp<br>pprrpp<br>prrpp<br>prrpp | 0 – | – ↕ ↕ – |
| LDX #*opr8i*<br>LDX *opr8a*<br>LDX *opr16a*<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<br>X ← (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AE ii<br>BE dd<br>CE hh ll<br>DE ee ff<br>EE ff<br>FE<br>9E DE ee ff<br>9E EE ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 – | – ↕ ↕ – |
| LSL *opr8a*<br>LSLA<br>LSLX<br>LSL *oprx8*,X<br>LSL ,X<br>LSL *oprx8*,SP | Logical Shift Left<br>  <br><br>(Same as ASL) | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38 dd<br>48<br>58<br>68 ff<br>78<br>9E 68 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ – | – ↕ ↕ ↕ |
| LSR *opr8a*<br>LSRA<br>LSR*X*<br>LSR *oprx8*,X<br>LSR ,X<br>LSR *oprx8*,SP | Logical Shift Right<br>  <br> | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 34 dd<br>44<br>54<br>64 ff<br>74<br>9E 64 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ – | – 0 ↕ ↕ |

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

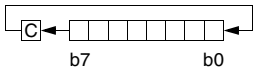
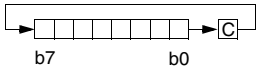
| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | VH | I N Z C |
| MOV *oprx8*,*oprx8*<br>MOV *oprx8*,X+<br>MOV #*oprx8*,*oprx8*<br>MOV ,X+,*oprx8* | Move<br>$(M)_{destination} \leftarrow (M)_{source}$<br>In IX+/DIR and DIR/IX+ Modes,<br>H:X ← (H:X) + $0001 | DIR/DIR<br>DIR/IX+<br>IMM/DIR<br>IX+/DIR | 4E dd dd<br>5E dd<br>6E ii dd<br>7E dd | 5<br>5<br>4<br>5 | rpwpp<br>rfwpp<br>pwpp<br>rfwpp | 0 – | – ↕ ↕ – |
| MUL | Unsigned multiply<br>X:A ← (X) × (A) | INH | 42 | 5 | ffffp | – 0 | – – – 0 |
| NEG *oprx8*<br>NEGA<br>NEGX<br>NEG *oprx8*,X<br>NEG ,X<br>NEG *oprx8*,SP | Negate M ← – (M) = $00 – (M)<br>(Two's Complement) A ← – (A) = $00 – (A)<br>X ← – (X) = $00 – (X)<br>M ← – (M) = $00 – (M)<br>M ← – (M) = $00 – (M)<br>M ← – (M) = $00 – (M) | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 30 dd<br>40<br>50<br>60 ff<br>70<br>9E 60 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ – | – ↕ ↕ ↕ |
| NOP | No Operation — Uses 1 Bus Cycle | INH | 9D | 1 | p | – – | – – – – |
| NSA | Nibble Swap Accumulator<br>A ← (A[3:0]:A[7:4]) | INH | 62 | 1 | p | – – | – – – – |
| ORA #*oprx8*<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<br>A ← (A) \| (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | AA ii<br>BA dd<br>CA hh ll<br>DA ee ff<br>EA ff<br>FA<br>9E DA ee ff<br>9E EA ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | 0 – | – ↕ ↕ – |
| PSHA | Push Accumulator onto Stack<br>Push (A); SP ← (SP) – $0001 | INH | 87 | 2 | sp | – – | – – – – |
| PSHH | Push H (Index Register High) onto Stack<br>Push (H); SP ← (SP) – $0001 | INH | 8B | 2 | sp | – – | – – – – |
| PSHX | Push X (Index Register Low) onto Stack<br>Push (X); SP ← (SP) – $0001 | INH | 89 | 2 | sp | – – | – – – – |
| PULA | Pull Accumulator from Stack<br>SP ← (SP + $0001); Pull (A) | INH | 86 | 3 | ufp | – – | – – – – |
| PULH | Pull H (Index Register High) from Stack<br>SP ← (SP + $0001); Pull (H) | INH | 8A | 3 | ufp | – – | – – – – |
| PULX | Pull X (Index Register Low) from Stack<br>SP ← (SP + $0001); Pull (X) | INH | 88 | 3 | ufp | – – | – – – – |
| ROL *oprx8*<br>ROLA<br>ROLX<br>ROL *oprx8*,X<br>ROL ,X<br>ROL *oprx8*,SP | Rotate Left through Carry<br>  | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 39 dd<br>49<br>59<br>69 ff<br>79<br>9E 69 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ – | – ↕ ↕ ↕ |
| ROR *oprx8*<br>RORA<br>RORX<br>ROR *oprx8*,X<br>ROR ,X<br>ROR *oprx8*,SP | Rotate Right through Carry<br>  | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 36 dd<br>46<br>56<br>66 ff<br>76<br>9E 66 ff | 5<br>1<br>1<br>5<br>4<br>6 | rfwpp<br>p<br>p<br>rfwpp<br>rfwp<br>prfwpp | ↕ – | – ↕ ↕ ↕ |

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

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

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

| Source Form | Operation | Address Mode | Object Code | Cycles | Cyc-by-Cyc Details | Affect on CCR | |
|---|---|---|---|---|---|---|---|
| | | | | | | VH | I N Z C |
| SUB #*opr8i*<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<br>A ← (A) – (M) | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP2<br>SP1 | A0 ii<br>B0 dd<br>C0 hh ll<br>D0 ee ff<br>E0 ff<br>F0<br>9E D0 ee ff<br>9E E0 ff | 2<br>3<br>4<br>4<br>3<br>3<br>5<br>4 | pp<br>rpp<br>prpp<br>prpp<br>rpp<br>rfp<br>pprpp<br>prpp | ↕ – | – ↕ ↕ ↕ |
| SWI | Software Interrupt<br>PC ← (PC) + $0001<br>Push (PCL); SP ← (SP) – $0001<br>Push (PCH); SP ← (SP) – $0001<br>Push (X); SP ← (SP) – $0001<br>Push (A); SP ← (SP) – $0001<br>Push (CCR); SP ← (SP) – $0001<br>I ← 1;<br>PCH ← Interrupt Vector High Byte<br>PCL ← Interrupt Vector Low Byte | INH | 83 | 11 | ssssvvfppp | – – | 1 – – – |
| TAP | Transfer Accumulator to CCR<br>CCR ← (A) | INH | 84 | 1 | p | ↕ ↕ | ↕ ↕ ↕ ↕ |
| TAX | Transfer Accumulator to X (Index Register Low)<br>X ← (A) | INH | 97 | 1 | p | – – | – – – – |
| TPA | Transfer CCR to Accumulator<br>A ← (CCR) | INH | 85 | 1 | p | – – | – – – – |
| TST *opr8a*<br>TSTA<br>TSTX<br>TST *oprx8*,X<br>TST ,X<br>TST *oprx8*,SP | Test for Negative or Zero (M) – $00<br>(A) – $00<br>(X) – $00<br>(M) – $00<br>(M) – $00<br>(M) – $00 | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3D dd<br>4D<br>5D<br>6D ff<br>7D<br>9E 6D ff | 4<br>1<br>1<br>4<br>3<br>5 | rfpp<br>p<br>p<br>rfpp<br>rfp<br>prfpp | 0 – | – ↕ ↕ – |
| TSX | Transfer SP to Index Reg.<br>H:X ← (SP) + $0001 | INH | 95 | 2 | fp | – – | – – – – |
| TXA | Transfer X (Index Reg. Low) to Accumulator<br>A ← (X) | INH | 9F | 1 | p | – – | – – – – |

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

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

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

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

**Operation Symbols:**

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

**Addressing Modes:**

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

**Cycle-by-Cycle Codes:**

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

**CCR Bits:**

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

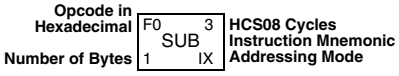
**CCR Effects:**

| | |
|---|---|
| ↕ | Set or cleared |
| – | Not affected |
| U | Undefined |

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

| Bit-Manipulation | | Branch | Read-Modify-Write | | | | | Control | | Register/Memory | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 5<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 Post Increment |
| EXT | Extended | IX2 | Indexed, 16-Bit Offset | | |
| DD | DIR to DIR | IMD | IMM to DIR | IX1+ | Indexed, 1-Byte Offset with Post Increment |
| IX+D | IX+ to DIR | DIX+ | DIR to IX+ | | |

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

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

| 9E60 6 |
|---|
| NEG |
| 3 SP1 |

HCS08 Cycles
Instruction Mnemonic
Addressing Mode

# Chapter 8
# Analog Comparator (S08ACMPV2)

## 8.1 Introduction

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

Figure 8-1 shows the MC9S08QG8/4 block diagram with the ACMP highlighted.

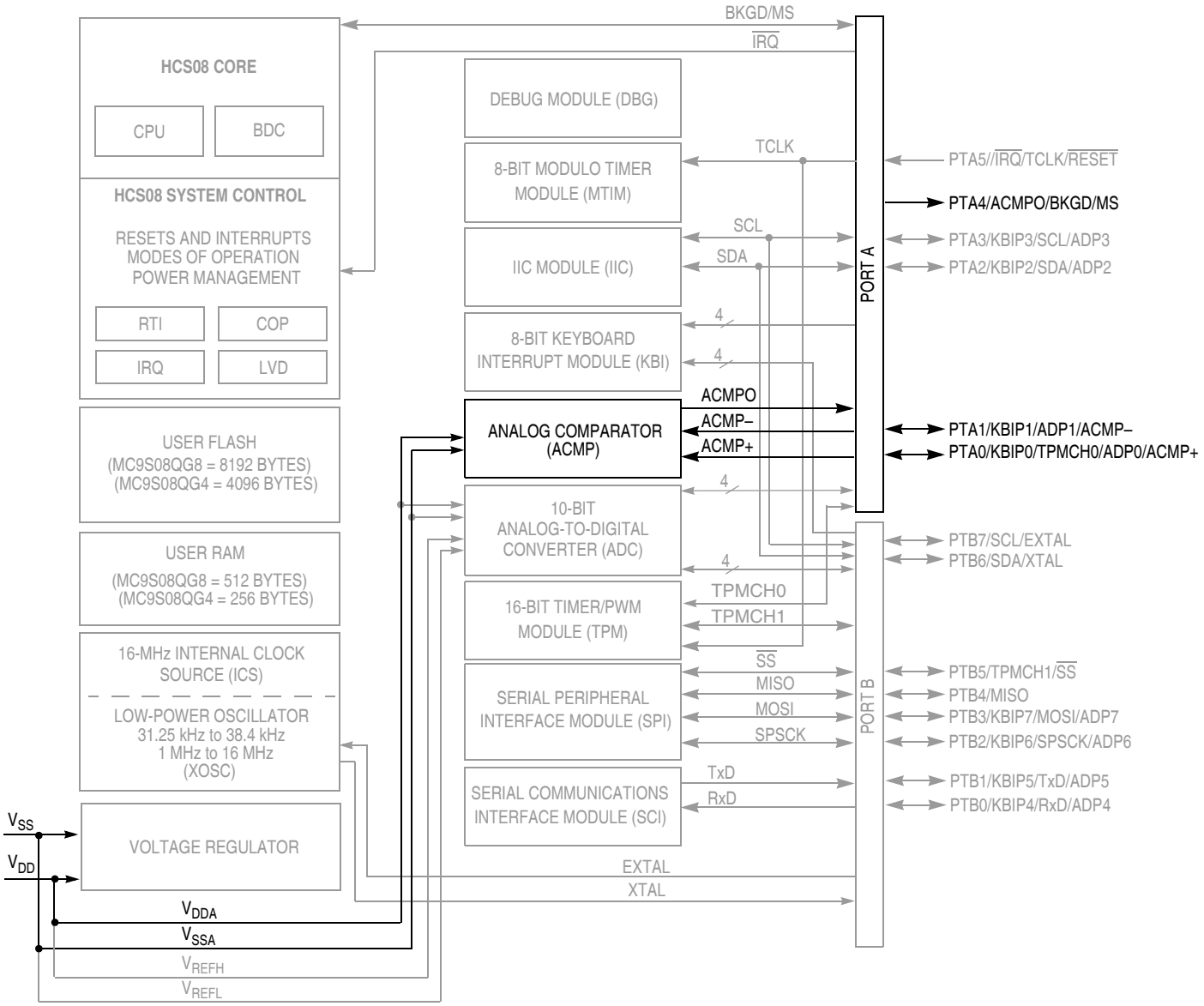
### 8.1.1 ACMP Configuration Information

When using the bandgap reference voltage for input to ACMP+, the user must enable the bandgap buffer by setting BGBE =1 in SPMSC1; see Section 5.8.8, “System Power Management Status and Control 1 Register (SPMSC1)”. For the value of the bandgap voltage reference see Section A.5, “DC Characteristics”.

To use ACMPO, the BKGDPE bit in SOPT1 must be cleared. This will disable the background debug mode and on-chip ICE.

### 8.1.2 ACMP/TPM Configuration Information

The ACMP module can be configured to connect the output of the analog comparator to TPM input capture channel 0 by setting ACIC in SOPT2. With ACIC set, the TPMCH0 pin is not available externally regardless of the configuration of the TPM module.




NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

**Figure 8-1. MC9S08QG8/4 Block Diagram Highlighting ACMP Block and Pins**

### 8.1.3 Features

The ACMP has the following features:

- Full rail-to-rail supply operation.
- Less than 40 mV of input offset.
- Less than 15 mV of hysteresis.
- Selectable interrupt on rising edge, falling edge, or either rising or falling edges of comparator output.
- Option to compare to fixed internal bandgap reference voltage.
- Option to allow comparator output to be visible on a pin, ACMPO.

### 8.1.4 Modes of Operation

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

#### 8.1.4.1 ACMP in Wait Mode

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

#### 8.1.4.2 ACMP in Stop Modes

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

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

During stop3 mode, clocks to the ACMP module are halted. No registers are affected. In addition, the ACMP comparator circuit will enter a low power state. No compare operation will occur while in stop3.

If stop3 is exited with a reset, the ACMP will be put into its reset state. If stop3 is exited with an interrupt, the ACMP continues from the state it was in when stop3 was entered.

#### 8.1.4.3 ACMP in Active Background Mode

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

### 8.1.5 Block Diagram

The block diagram for the analog comparator module is shown Figure 8-2.

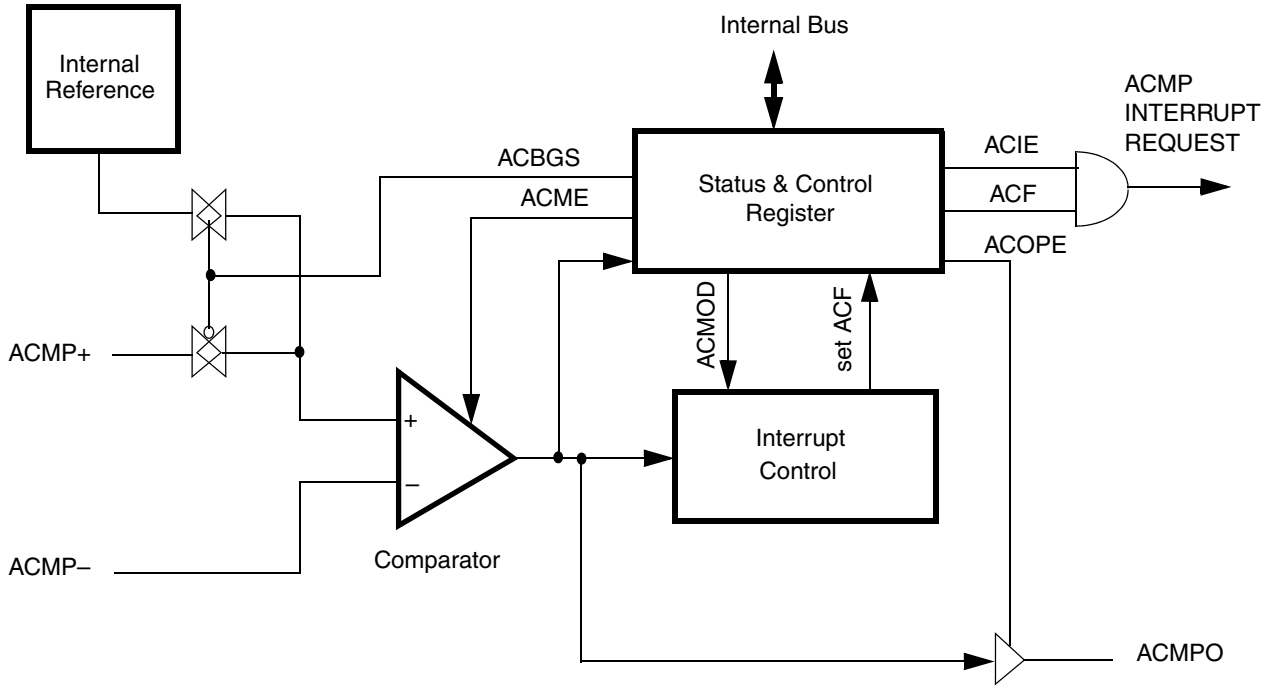



**Figure 8-2. Analog Comparator (ACMP) Block Diagram**

## 8.2 External Signal Description

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

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

**Table 8-1. Signal Properties**

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

## 8.3 Register Definition

The ACMP includes one register:

- An 8-bit status and control register

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

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

### 8.3.1 ACMP Status and Control Register (ACMPSC)

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

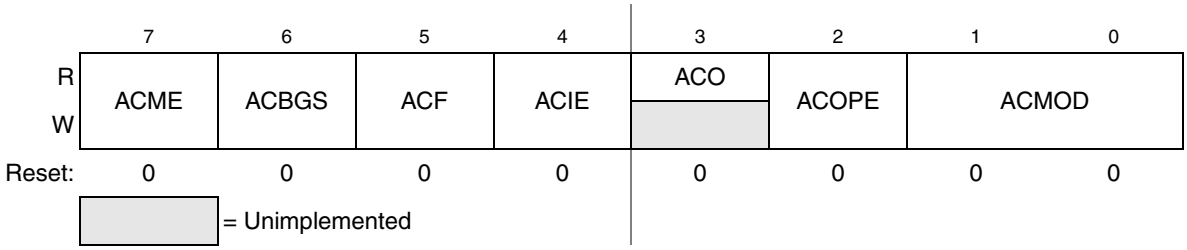



**Figure 8-3. ACMP Status and Control Register**

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

| Field | Description |
|---|---|
| 7<br>ACME | **Analog Comparator Module Enable** — ACME enables the ACMP module.<br>0 ACMP not enabled<br>1 ACMP is enabled |
| 6<br>ACBGS | **Analog Comparator Bandgap Select** — ACBGS is used to select between the bandgap reference voltage or the ACMP+ pin as the input to the non-inverting input of the analog comparatorr.<br>0 External pin ACMP+ selected as non-inverting input to comparator<br>1 Internal reference select as non-inverting input to comparator |
| 5<br>ACF | **Analog Comparator Flag** — ACF is set when a compare event occurs. Compare events are defined by ACMOD. ACF is cleared by writing a one to ACF.<br>0 Compare event has not occurred<br>1 Compare event has occurred |
| 4<br>ACIE | **Analog Comparator Interrupt Enable** — ACIE enables the interrupt from the ACMP. When ACIE is set, an interrupt will be asserted when ACF is set.<br>0 Interrupt disabled<br>1 Interrupt enabled |
| 3<br>ACO | **Analog Comparator Output** — Reading ACO will return the current value of the analog comparator output. ACO is reset to a 0 and will read as a 0 when the ACMP is disabled (ACME = 0). |
| 2<br>ACOPE | **Analog Comparator Output Pin Enable** — ACOPE is used to enable the comparator output to be placed onto the external pin, ACMPO.<br>0 Analog comparator output not available on ACMPO<br>1 Analog comparator output is driven out on ACMPO |
| 1:0<br>ACMOD | **Analog Comparator Mode** — ACMOD selects the type of compare event which sets ACF.<br>00 Encoding 0 — Comparator output falling edge<br>01 Encoding 1 — Comparator output rising edge<br>10 Encoding 2 — Comparator output falling edge<br>11 Encoding 3 — Comparator output rising or falling edge |

## 8.4 Functional Description

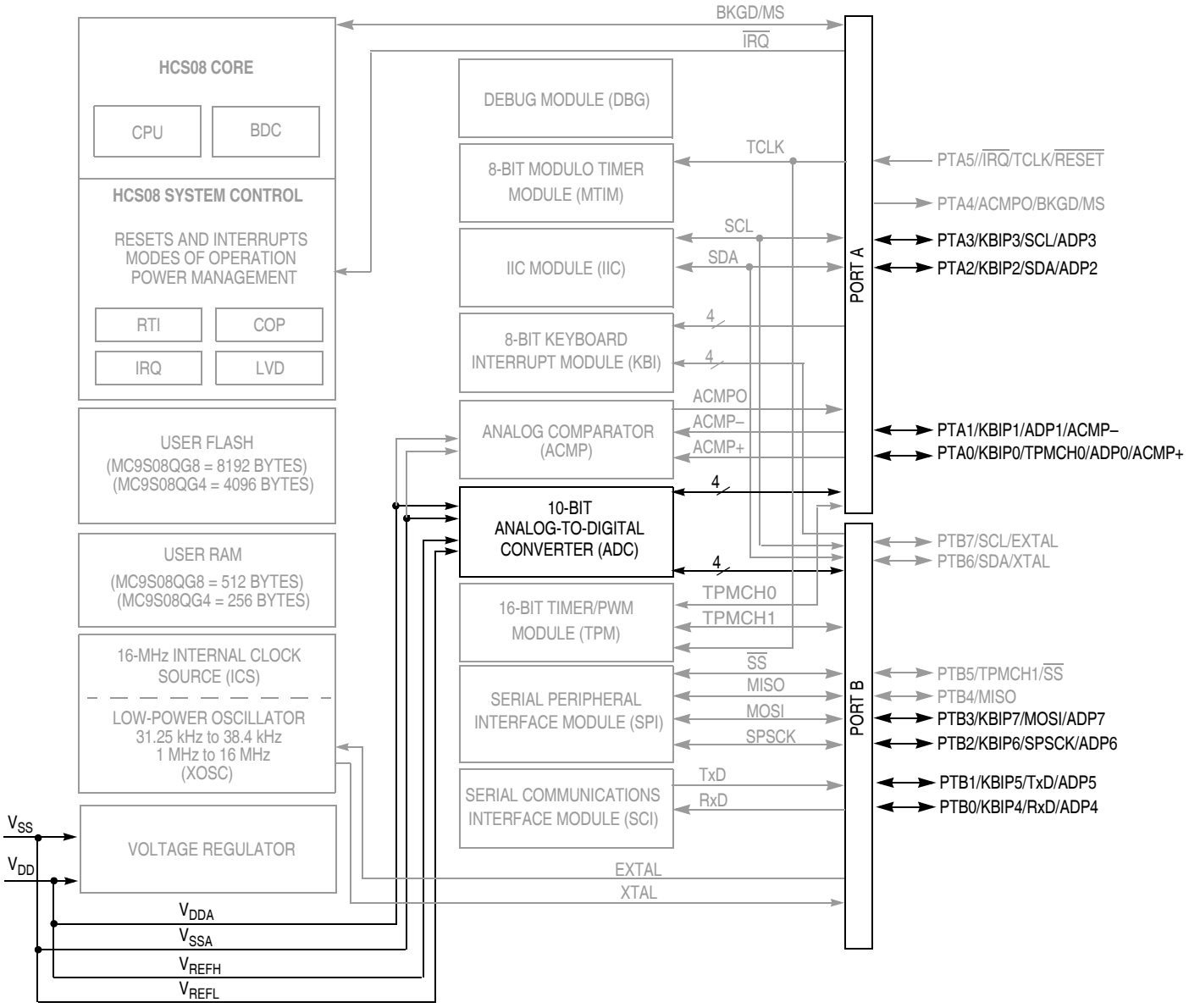
The analog comparator can be used to compare two analog input voltages applied to ACMP+ and ACMP–; or it can be used to compare an analog input voltage applied to ACMP– with an internal bandgap reference voltage. ACBGS is used to select between the bandgap reference voltage or the ACMP+ pin as the input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

# Chapter 9
# Analog-to-Digital Converter (S08ADC10V1)

## 9.1 Introduction

The 10-bit analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

Figure 9-1 shows the MC9S08QG8/4 with the ADC module and pins highlighted.




NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. IRQ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as IRQ pin function (IRQPE = 1).
6. $\overline{\text{RESET}}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

**Figure 9-1. MC9S08QG8/4 Block Diagram Highlighting ADC Block and Pins**

## 9.1.1 Module Configurations

This section provides device-specific information for configuring the ADC on MC9S08QG8/4 devices.

### 9.1.1.1 Analog Supply and Voltage Reference Connections

The $V_{DDAD}$ and $V_{REFH}$ sources for the ADC are internally connected to the $V_{DD}$ pin. The $V_{SSAD}$ and $V_{REFL}$ sources for the ADC are internally connected to the $V_{SS}$ pin.

### 9.1.1.2 Channel Assignments

The ADC channel assignments for the MC9S08QG8/4 devices are shown in Table 9-1. Reserved channels convert to an unknown value.

**Table 9-1. ADC Channel Assignment**

| ADCH | Channel | Input | Pin Control |
|---|---|---|---|
| 00000 | AD0 | PTA0/ADP0 | ADPC0 |
| 00001 | AD1 | PTA1/ADP1 | ADPC1 |
| 00010 | AD2 | PTA2/ADP2 | ADPC2 |
| 00011 | AD3 | PTA3/ADP3 | ADPC3 |
| 00100 | AD4 | PTB0/ADP4 | ADPC4 |
| 00101 | AD5 | PTB1/ADP5 | ADPC5 |
| 00110 | AD6 | PTB2/ADP6 | ADPC6 |
| 00111 | AD7 | PTB3/ADP7 | ADPC7 |
| 01000 | AD8 | $V_{SS}$ | N/A |
| 01001 | AD9 | $V_{SS}$ | N/A |
| 01010 | AD10 | $V_{SS}$ | N/A |
| 01011 | AD11 | $V_{SS}$ | N/A |
| 01100 | AD12 | $V_{SS}$ | N/A |
| 01101 | AD13 | $V_{SS}$ | N/A |
| 01110 | AD14 | $V_{SS}$ | N/A |
| 01111 | AD15 | $V_{SS}$ | N/A |

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

[1] For information, see Section 9.1.1.6, “Temperature Sensor.”

**NOTE**

Selecting the internal bandgap channel requires BGBE =1 in SPMSC1; see Section 5.8.8, “System Power Management Status and Control 1 Register (SPMSC1).” For the value of the bandgap voltage reference see Section A.5, “DC Characteristics.”

#### 9.1.1.3 Alternate Clock

The ADC is capable of performing conversions using the MCU bus clock, the bus clock divided by two, or the local asynchronous clock (ADACK) within the module. The alternate clock, ALTCLK, input for the MC9S08QG8/4 MCU devices is not implemented.

#### 9.1.1.4 Hardware Trigger

The ADC hardware trigger, ADHWT, is output from the real-time interrupt (RTI) counter. The RTI counter can be clocked by either ICSERCLK or a nominal 1-kHz clock source within the RTI block.

The period of the RTI is determined by the input clock frequency and the RTIS bits. The RTI counter is a free running counter that generates an overflow at the RTI rate determined by the RTIS bits. When the ADC hardware trigger is enabled, a conversion is initiated upon an RTI counter overflow.

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

#### 9.1.1.5 Analog Pin Enables

The ADC on MC9S08QG8 devices contains only one analog pin enable register, APCTL1.

#### 9.1.1.6 Temperature Sensor

The ADC module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 9-1 provides an approximate transfer function of the temperature sensor.

$$\text{Temp} = 25 - ((V_{TEMP} - V_{TEMP25}) \div m) \qquad \textit{Eqn. 9-1}$$

where:

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

For temperature calculations, use the $V_{TEMP25}$ and m values from Section A.10, “ADC Characteristics,” in Appendix A, “Electrical Characteristics.”

In application code, the user reads the temperature sensor channel, calculates $V_{TEMP}$, and compares to $V_{TEMP25}$. If $V_{TEMP}$ is greater than $V_{TEMP25}$, the cold slope value is applied in Equation 9-1. If $V_{TEMP}$ is less than $V_{TEMP25}$ the hot slope value is applied in Equation 9-1.

#### 9.1.1.7 Low-Power Mode Operation

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

### 9.1.2 Features

Features of the ADC module include:

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

### 9.1.3 Block Diagram

Figure 9-2 provides a block diagram of the ADC module

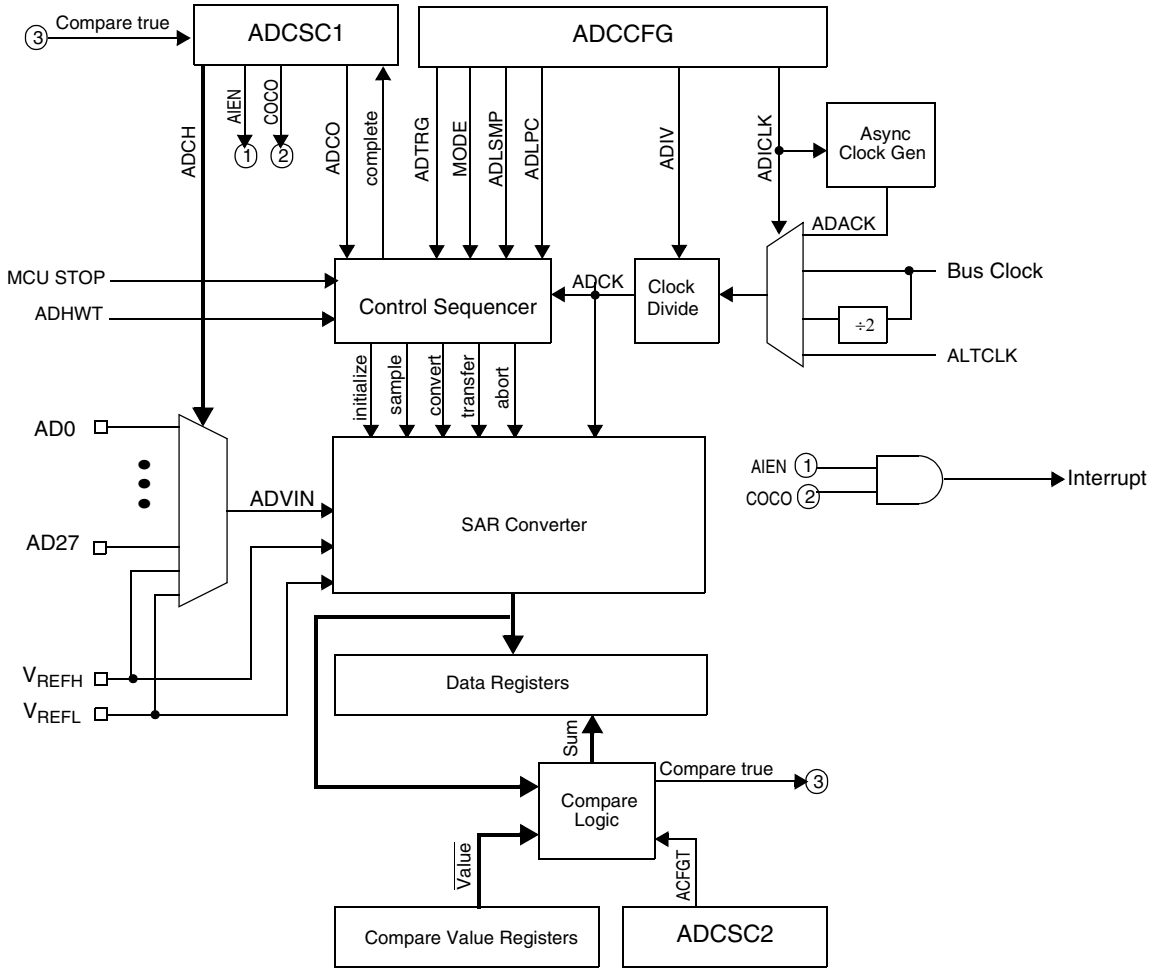



**Figure 9-2. ADC Block Diagram**

## 9.2 External Signal Description

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

**Table 9-2. Signal Properties**

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

### 9.2.1 Analog Power ($V_{DDAD}$)

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

### 9.2.2 Analog Ground ($V_{SSAD}$)

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

### 9.2.3 Voltage Reference High ($V_{REFH}$)

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

### 9.2.4 Voltage Reference Low ($V_{REFL}$)

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

### 9.2.5 Analog Channel Inputs (ADx)

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

## 9.3 Register Definition

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

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

### 9.3.1 Status and Control Register 1 (ADCSC1)

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

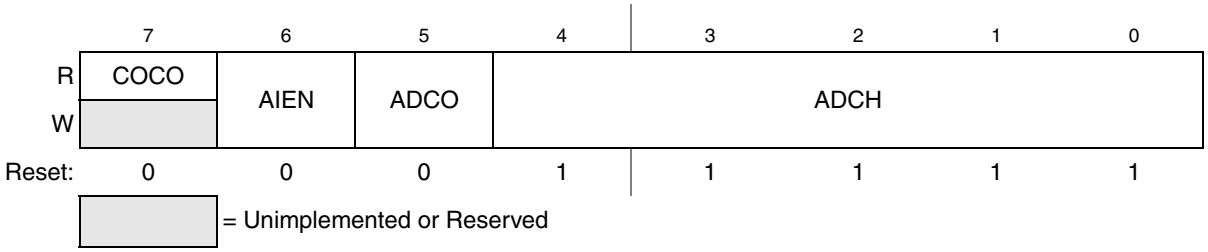



**Figure 9-3. Status and Control Register (ADCSC1)**

**Table 9-3. ADCSC1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>COCO | **Conversion Complete Flag** — The COCO flag is a read-only bit which is set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1) the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared whenever ADCSC1 is written or whenever ADCRL is read.<br>0 Conversion not completed<br>1 Conversion completed |
| 6<br>AIEN | **Interrupt Enable** — AIEN is used to enable conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted.<br>0 Conversion complete interrupt disabled<br>1 Conversion complete interrupt enabled |
| 5<br>ADCO | **Continuous Conversion Enable** — ADCO is used to enable continuous conversions.<br>0 One conversion following a write to the ADCSC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected.<br>1 Continuous conversions initiated following a write to ADCSC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected. |
| 4:0<br>ADCH | **Input Channel Select** — The ADCH bits form a 5-bit field which is used to select one of the input channels. The input channels are detailed in Figure 9-4.<br>The successive approximation converter subsystem is turned off when the channel select bits are all set to 1. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way will prevent an additional, single conversion from being performed. It is not necessary to set the channel select bits to all 1s to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes. |

**Figure 9-4. Input Channel Select**

| ADCH | Input Select |
|---|---|
| 00000 | AD0 |
| 00001 | AD1 |
| 00010 | AD2 |
| 00011 | AD3 |
| 00100 | AD4 |
| 00101 | AD5 |
| 00110 | AD6 |

| ADCH | Input Select |
|---|---|
| 10000 | AD16 |
| 10001 | AD17 |
| 10010 | AD18 |
| 10011 | AD19 |
| 10100 | AD20 |
| 10101 | AD21 |
| 10110 | AD22 |

**Figure 9-4. Input Channel Select (continued)**

| ADCH | Input Select |
|---|---|
| 00111 | AD7 |
| 01000 | AD8 |
| 01001 | AD9 |
| 01010 | AD10 |
| 01011 | AD11 |
| 01100 | AD12 |
| 01101 | AD13 |
| 01110 | AD14 |
| 01111 | AD15 |

| ADCH | Input Select |
|---|---|
| 10111 | AD23 |
| 11000 | AD24 |
| 11001 | AD25 |
| 11010 | AD26 |
| 11011 | AD27 |
| 11100 | Reserved |
| 11101 | $V_{REFH}$ |
| 11110 | $V_{REFL}$ |
| 11111 | Module disabled |

## 9.3.2 Status and Control Register 2 (ADCSC2)

The ADCSC2 register is used to control the compare function, conversion trigger and conversion active of the ADC module.

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

= Unimplemented or Reserved

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

**Figure 9-5. Status and Control Register 2 (ADCSC2)**

**Table 9-4. ADCSC2 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADACT | **Conversion Active** — ADACT indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted.<br>0 Conversion not in progress<br>1 Conversion in progress |
| 6<br>ADTRG | **Conversion Trigger Select** — ADTRG is used to select the type of trigger to be used for initiating a conversion. Two types of trigger are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input.<br>0 Software trigger selected<br>1 Hardware trigger selected |

**Table 9-4. ADCSC2 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 5<br>ACFE | **Compare Function Enable** — ACFE is used to enable the compare function.<br>0 Compare function disabled<br>1 Compare function enabled |
| 4<br>ACFGT | **Compare Function Greater Than Enable** — ACFGT is used to configure the compare function to trigger when the result of the conversion of the input being monitored is greater than or equal to the compare value. The compare function defaults to triggering when the result of the compare of the input being monitored is less than the compare value.<br>0 Compare triggers when input is less than compare level<br>1 Compare triggers when input is greater than or equal to compare level |

## 9.3.3 Data Result High Register (ADCRH)

ADCRH contains the upper two bits of the result of a 10-bit conversion. When configured for 8-bit conversions both ADR8 and ADR9 are equal to zero. ADCRH is updated each time a conversion completes except when automatic compare is enabled and the compare condition is not met. In 10-bit MODE, reading ADCRH prevents the ADC from transferring subsequent conversion results into the result registers until ADCRL is read. If ADCRL is not read until after the next conversion is completed, then the intermediate conversion result will be lost. In 8-bit mode there is no interlocking with ADCRL. In the case that the MODE bits are changed, any data in ADCRH becomes invalid.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | ADR9 | ADR8 |
| W | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 9-6. Data Result High Register (ADCRH)**

## 9.3.4 Data Result Low Register (ADCRL)

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

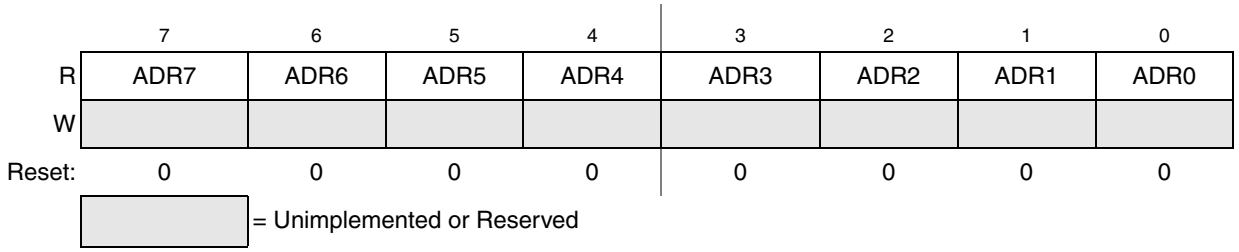



**Figure 9-7. Data Result Low Register (ADCRL)**

## 9.3.5 Compare Value High Register (ADCCVH)

This register holds the upper two bits of the 10-bit compare value. These bits are compared to the upper two bits of the result following a conversion in 10-bit mode when the compare function is enabled.In 8-bit operation, ADCCVH is not used during compare.

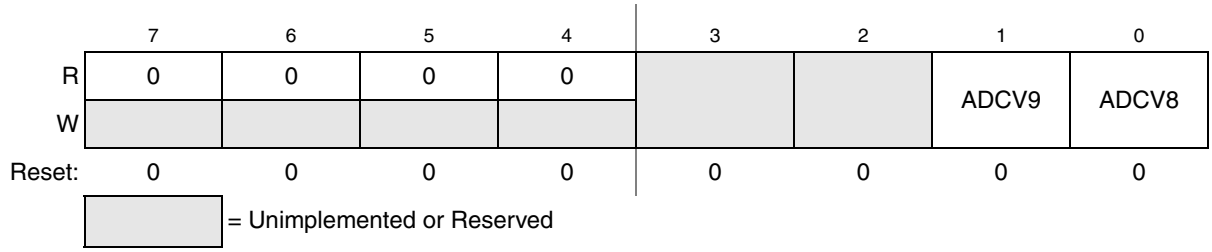



**Figure 9-8. Compare Value High Register (ADCCVH)**

## 9.3.6 Compare Value Low Register (ADCCVL)

This register holds the lower 8 bits of the 10-bit compare value, or all 8 bits of the 8-bit compare value. Bits ADCV7:ADCV0 are compared to the lower 8 bits of the result following a conversion in either 10-bit or 8-bit mode.

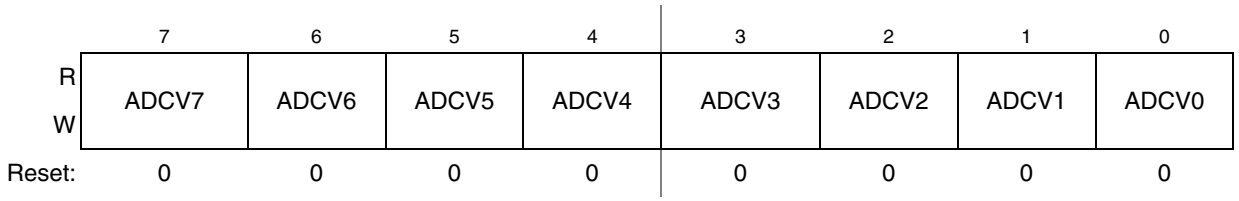



**Figure 9-9. Compare Value Low Register(ADCCVL)**

## 9.3.7 Configuration Register (ADCCFG)

ADCCFG is used to select the mode of operation, clock source, clock divide, and configure for low power or long sample time.

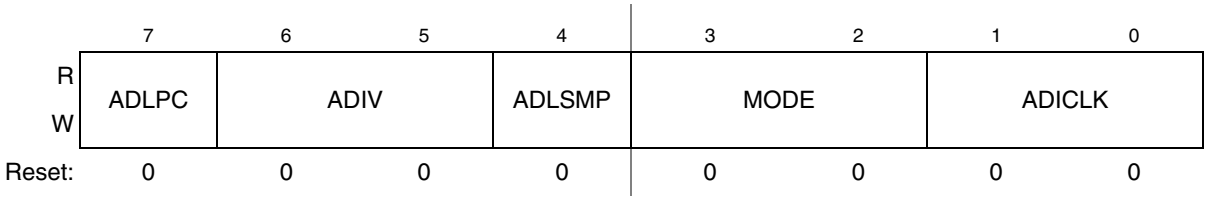



**Figure 9-10. Configuration Register (ADCCFG)**

**Table 9-5. ADCCFG Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADLPC | **Low Power Configuration** — ADLPC controls the speed and power configuration of the successive approximation converter. This is used to optimize power consumption when higher sample rates are not required.<br>0 High speed configuration<br>1 Low power configuration: {FC31}The power is reduced at the expense of maximum clock speed. |
| 6:5<br>ADIV | **Clock Divide Select** — ADIV select the divide ratio used by the ADC to generate the internal clock ADCK. Table 9-6 shows the available clock configurations. |
| 4<br>ADLSMP | **Long Sample Time Configuration** — ADLSMP selects between long and short sample time. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required.<br>0 Short sample time<br>1 Long sample time |
| 3:2<br>MODE | **Conversion Mode Selection** — MODE bits are used to select between 10- or 8-bit operation. See Table 9-7. |
| 1:0<br>ADICLK | **Input Clock Select** — ADICLK bits select the input clock source to generate the internal clock ADCK. See Table 9-8. |

**Table 9-6. Clock Divide Select**

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

**Table 9-7. Conversion Modes**

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

**Table 9-8. Input Clock Select**

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

## 9.3.8 Pin Control 1 Register (APCTL1)

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

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

**Figure 9-11. Pin Control 1 Register (APCTL1)**

**Table 9-9. APCTL1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADPC7 | **ADC Pin Control 7** — ADPC7 is used to control the pin associated with channel AD7.<br>0 AD7 pin I/O control enabled<br>1 AD7 pin I/O control disabled |
| 6<br>ADPC6 | **ADC Pin Control 6** — ADPC6 is used to control the pin associated with channel AD6.<br>0 AD6 pin I/O control enabled<br>1 AD6 pin I/O control disabled |
| 5<br>ADPC5 | **ADC Pin Control 5** — ADPC5 is used to control the pin associated with channel AD5.<br>0 AD5 pin I/O control enabled<br>1 AD5 pin I/O control disabled |
| 4<br>ADPC4 | **ADC Pin Control 4** — ADPC4 is used to control the pin associated with channel AD4.<br>0 AD4 pin I/O control enabled<br>1 AD4 pin I/O control disabled |
| 3<br>ADPC3 | **ADC Pin Control 3** — ADPC3 is used to control the pin associated with channel AD3.<br>0 AD3 pin I/O control enabled<br>1 AD3 pin I/O control disabled |
| 2<br>ADPC2 | **ADC Pin Control 2** — ADPC2 is used to control the pin associated with channel AD2.<br>0 AD2 pin I/O control enabled<br>1 AD2 pin I/O control disabled |

**Table 9-9. APCTL1 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>ADPC1 | **ADC Pin Control 1** — ADPC1 is used to control the pin associated with channel AD1.<br>0 AD1 pin I/O control enabled<br>1 AD1 pin I/O control disabled |
| 0<br>ADPC0 | **ADC Pin Control 0** — ADPC0 is used to control the pin associated with channel AD0.<br>0 AD0 pin I/O control enabled<br>1 AD0 pin I/O control disabled |

## 9.3.9 Pin Control 2 Register (APCTL2)

APCTL2 is used to control channels 8–15 of the ADC module.

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

**Figure 9-12. Pin Control 2 Register (APCTL2)**

**Table 9-10. APCTL2 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADPC15 | **ADC Pin Control 15** — ADPC15 is used to control the pin associated with channel AD15.<br>0 AD15 pin I/O control enabled<br>1 AD15 pin I/O control disabled |
| 6<br>ADPC14 | **ADC Pin Control 14** — ADPC14 is used to control the pin associated with channel AD14.<br>0 AD14 pin I/O control enabled<br>1 AD14 pin I/O control disabled |
| 5<br>ADPC13 | **ADC Pin Control 13** — ADPC13 is used to control the pin associated with channel AD13.<br>0 AD13 pin I/O control enabled<br>1 AD13 pin I/O control disabled |
| 4<br>ADPC12 | **ADC Pin Control 12** — ADPC12 is used to control the pin associated with channel AD12.<br>0 AD12 pin I/O control enabled<br>1 AD12 pin I/O control disabled |
| 3<br>ADPC11 | **ADC Pin Control 11** — ADPC11 is used to control the pin associated with channel AD11.<br>0 AD11 pin I/O control enabled<br>1 AD11 pin I/O control disabled |
| 2<br>ADPC10 | **ADC Pin Control 10** — ADPC10 is used to control the pin associated with channel AD10.<br>0 AD10 pin I/O control enabled<br>1 AD10 pin I/O control disabled |

**Table 9-10. APCTL2 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>ADPC9 | **ADC Pin Control 9** — ADPC9 is used to control the pin associated with channel AD9.<br>0 AD9 pin I/O control enabled<br>1 AD9 pin I/O control disabled |
| 0<br>ADPC8 | **ADC Pin Control 8** — ADPC8 is used to control the pin associated with channel AD8.<br>0 AD8 pin I/O control enabled<br>1 AD8 pin I/O control disabled |

## 9.3.10 Pin Control 3 Register (APCTL3)

APCTL3 is used to control channels 16–23 of the ADC module.

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

**Figure 9-13. Pin Control 3 Register (APCTL3)**

**Table 9-11. APCTL3 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ADPC23 | **ADC Pin Control 23** — ADPC23 is used to control the pin associated with channel AD23.<br>0 AD23 pin I/O control enabled<br>1 AD23 pin I/O control disabled |
| 6<br>ADPC22 | **ADC Pin Control 22** — ADPC22 is used to control the pin associated with channel AD22.<br>0 AD22 pin I/O control enabled<br>1 AD22 pin I/O control disabled |
| 5<br>ADPC21 | **ADC Pin Control 21** — ADPC21 is used to control the pin associated with channel AD21.<br>0 AD21 pin I/O control enabled<br>1 AD21 pin I/O control disabled |
| 4<br>ADPC20 | **ADC Pin Control 20** — ADPC20 is used to control the pin associated with channel AD20.<br>0 AD20 pin I/O control enabled<br>1 AD20 pin I/O control disabled |
| 3<br>ADPC19 | **ADC Pin Control 19** — ADPC19 is used to control the pin associated with channel AD19.<br>0 AD19 pin I/O control enabled<br>1 AD19 pin I/O control disabled |
| 2<br>ADPC18 | **ADC Pin Control 18** — ADPC18 is used to control the pin associated with channel AD18.<br>0 AD18 pin I/O control enabled<br>1 AD18 pin I/O control disabled |

**Table 9-11. APCTL3 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 1<br>ADPC17 | **ADC Pin Control 17** — ADPC17 is used to control the pin associated with channel AD17.<br>0 AD17 pin I/O control enabled<br>1 AD17 pin I/O control disabled |
| 0<br>ADPC16 | **ADC Pin Control 16** — ADPC16 is used to control the pin associated with channel AD16.<br>0 AD16 pin I/O control enabled<br>1 AD16 pin I/O control disabled |

## 9.4 Functional Description

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

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. The selected channel voltage is converted by a successive approximation algorithm into an 11-bit digital result. In 8-bit mode, the selected channel voltage is converted by a successive approximation algorithm into a 9-bit digital result.

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

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

### 9.4.1 Clock Select and Divide Control

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

- The bus clock, which is equal to the frequency at which software is executed. This is the default selection following reset.
- The bus clock divided by 2. For higher bus clock rates, this allows a maximum divide by 16 of the bus clock.
- ALTCLK, as defined for this MCU (See module section introduction).
- The asynchronous clock (ADACK) – This clock is generated from a clock source within the ADC module. When selected as the clock source this clock remains active while the MCU is in wait or stop3 mode and allows conversions in these modes for lower noise operation.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC will not perform according to specifications. If the available clocks

are too fast, then the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

### 9.4.2 Input Select and Pin Control

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

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

### 9.4.3 Hardware Trigger

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

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

### 9.4.4 Conversion Control

Conversions can be performed in either 10-bit mode or 8-bit mode as determined by the MODE bits. Conversions can be initiated by either a software or hardware trigger. In addition, the ADC module can be configured for low power operation, long sample time, continuous conversion, and automatic compare of the conversion result to a software determined compare value.

#### 9.4.4.1 Initiating Conversions

A conversion is initiated:

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

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

#### 9.4.4.2 Completing Conversions

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

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

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

#### 9.4.4.3 Aborting Conversions

Any conversion in progress will be aborted when:

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

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

#### 9.4.4.4 Power Control

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

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

#### 9.4.4.5 Total Conversion Time

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

result of the conversion is transferred to ADCRH and ADCRL upon completion of the conversion algorithm.

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

The maximum total conversion time for different conditions is summarized in Table 9-12.

**Table 9-12. Total Conversion Time vs. Control Conditions**

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

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

$$\text{Conversion time} = \frac{23 \text{ ADCK cyc}}{8 \text{ MHz}/1} + \frac{5 \text{ bus cyc}}{8 \text{ MHz}} = 3.5 \text{ µs}$$

$$\text{Number of bus cycles} = 3.5 \text{ µs} \times 8 \text{ MHz} = 28 \text{ cycles}$$

**NOTE**

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

### 9.4.5 Automatic Compare Function

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

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

**NOTE**

The compare function can be used to monitor the voltage on a channel while the MCU is in either wait or stop3 mode. The ADC interrupt will wake the MCU when the compare condition is met.

### 9.4.6 MCU Wait Mode Operation

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

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

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

### 9.4.7 MCU Stop3 Mode Operation

The STOP instruction is used to put the MCU in a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

#### 9.4.7.1 Stop3 Mode With ADACK Disabled

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

#### 9.4.7.2 Stop3 Mode With ADACK Enabled

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

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

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

**NOTE**

It is possible for the ADC module to wake the system from low power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure that the data transfer blocking mechanism (discussed in Section 9.4.4.2, "Completing Conversions) is cleared when entering stop3 and continuing ADC conversions.

### 9.4.8 MCU Stop1 and Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters either stop1 or stop2 mode. All module registers contain their reset values following exit from stop1 or stop2. Therefore the module must be re-enabled and re-configured following exit from stop1 or stop2.

## 9.5 Initialization Information

This section gives an example which provides some basic direction on how a user would initialize and configure the ADC module. The user has the flexibility of choosing between configuring the module for 8-bit or 10-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to Table 9-6, Table 9-7, and Table 9-8 for information used in this example.

**NOTE**

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

### 9.5.1 ADC Module Initialization Example

#### 9.5.1.1 Initialization Sequence

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

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.

2. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.
3. Update status and control register 1 (ADCSC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

#### 9.5.1.2 Pseudo — Code Example

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

**ADCCFG = 0x98 (%10011000)**

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

**ADCSC2 = 0x00 (%00000000)**

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

**ADCSC1 = 0x41 (%01000001)**

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

**ADCRH/L = 0xxx**

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

**ADCCVH/L = 0xxx**

Holds compare value when compare function enabled

**APCTL1=0x02**

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

**APCTL2=0x00**

All other AD pins remain general purpose I/O pins

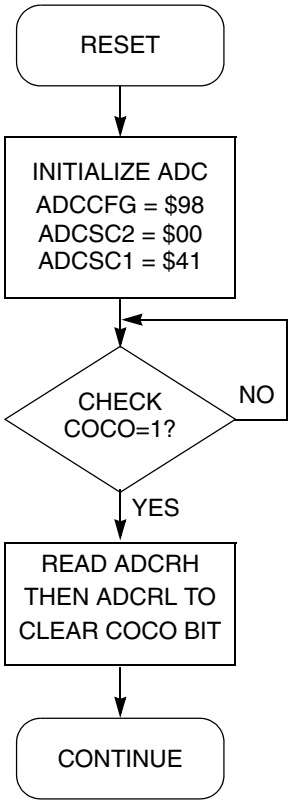



**Figure 9-14. Initialization Flowchart for Example**

## 9.6 Application Information

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

### 9.6.1 External Pins and Routing

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

#### 9.6.1.1 Analog Supply Pins

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

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

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

### 9.6.1.2 Analog Reference Pins

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

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

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

### 9.6.1.3 Analog Input Pins

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

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

For proper conversion, the input voltage must fall between $V_{REFH}$ and $V_{REFL}$. If the input is equal to or exceeds $V_{REFH}$, the converter circuit converts the signal to $3FF (full scale 10-bit representation) or $FF (full scale 8-bit representation). If the input is equal to or less than $V_{REFL}$, the converter circuit converts it to $000. Input voltages between $V_{REFH}$ and $V_{REFL}$ are straight-line linear conversions. There will be a brief current associated with $V_{REFL}$ when the sampling capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

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

### 9.6.2 Sources of Error

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

#### 9.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7kΩ and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 10-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source ($R_{AS}$) is kept below 5 kΩ.

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

#### 9.6.2.2 Pin Leakage Error

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

#### 9.6.2.3 Noise-Induced Errors

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

- There is a 0.1 µF low-ESR capacitor from $V_{REFH}$ to $V_{REFL}$.
- There is a 0.1 µF low-ESR capacitor from $V_{DDAD}$ to $V_{SSAD}$.
- If inductive isolation is used from the primary supply, an additional 1 µF capacitor is placed from $V_{DDAD}$ to $V_{SSAD}$.
- $V_{SSAD}$ (and $V_{REFL}$, if connected) is connected to $V_{SS}$ at a quiet point in the ground plane.
- Operate the MCU in wait or stop3 mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
  - For software triggered conversions, immediately follow the write to the ADCSC1 with a WAIT instruction or STOP instruction.
  - For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces $V_{DD}$ noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

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

- Place a 0.01 µF capacitor ($C_{AS}$) on the selected input channel to $V_{REFL}$ or $V_{SSAD}$ (this will improve noise issues but will affect sample rate based on the external analog source resistance).

- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

#### 9.6.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 1024 steps (in 10-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8 or 10), defined as 1LSB, is:

$$1_{LSB} = (V_{REFH} - V_{REFL}) / 2^N \qquad \textit{Eqn. 9-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code will transition when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2LSB in 8- or 10-bit mode. As a consequence, however, the code width of the first ($000) conversion is only 1/2LSB and the code width of the last ($FF or $3FF) is 1.5LSB.

#### 9.6.2.5 Linearity Errors

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

- Zero-scale error ($E_{ZS}$) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2LSB). Note, if the first conversion is $001, then the difference between the actual $001 code width and its ideal (1LSB) is used.
- Full-scale error ($E_{FS}$) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5LSB). Note, if the last conversion is $3FE, then the difference between the actual $3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function, and therefore includes all forms of error.

#### 9.6.2.6 Code Jitter, Non-Monotonicity and Missing Codes

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

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the

converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 LSB and will increase with noise. This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 9.6.2.3 will reduce this error.

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

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

# Chapter 10
# Internal Clock Source (S08ICSV1)

## 10.1 Introduction

The internal clock source (ICS) module provides clock source choices for the MCU. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock. The module can provide this FLL clock or either of the internal or external reference clocks as a source for the MCU system clock. There are also signals provided to control a low power oscillator (XOSC) module to allow the use of an external crystal/resonator as the external reference clock.

Whichever clock source is chosen, it is passed through a reduced bus divider (BDIV) which allows a lower final output clock frequency to be derived.

The bus frequency will be one-half of the ICSOUT frequency.

**NOTE**

The external reference clock is not available on all packages. See Table 1-1 for external clock availability for each package option.

### 10.1.1 Module Configuration

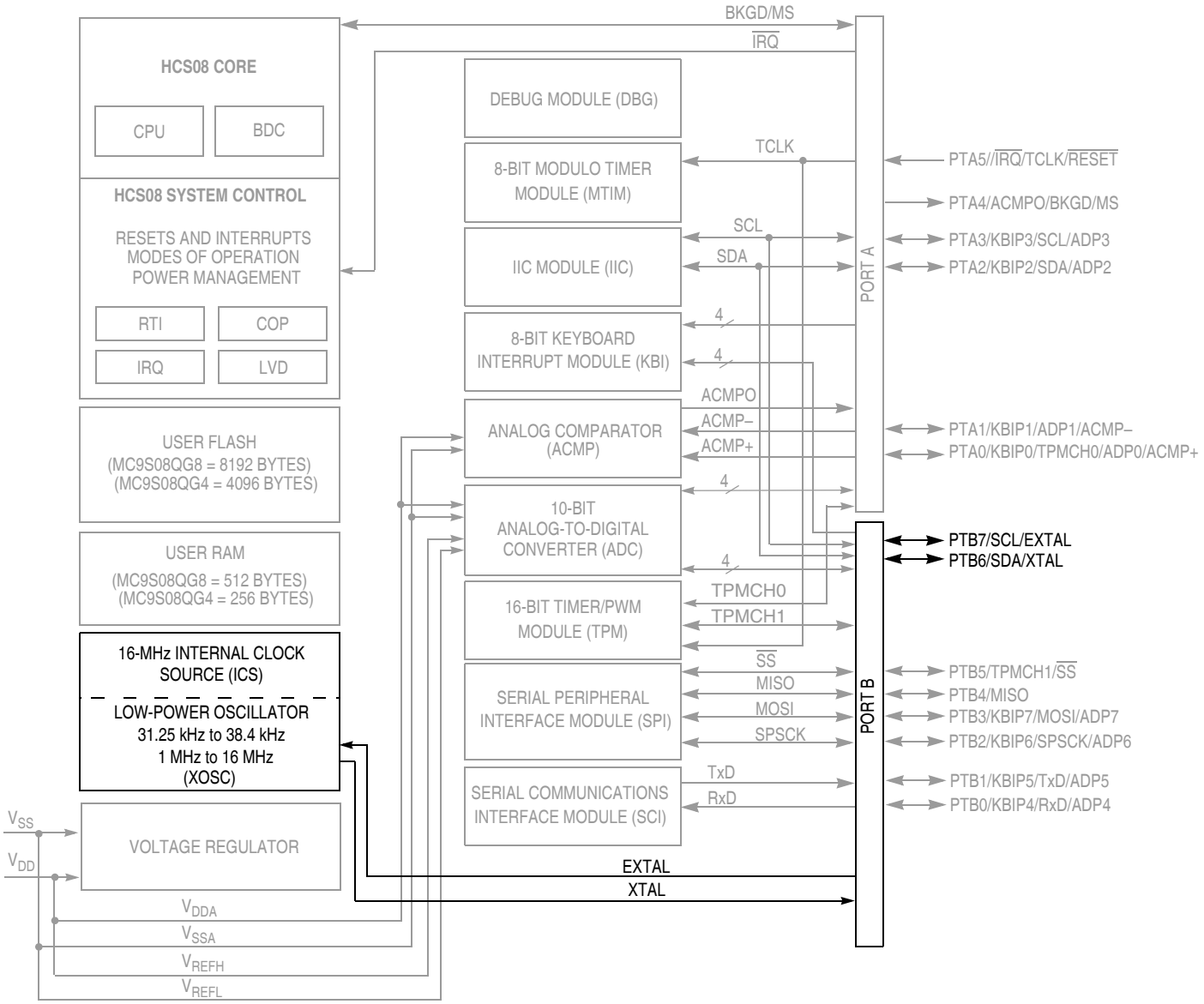
When the internal reference is enabled in stop mode (IREFSTEN = 1), the voltage regulator must also be enabled in stop mode by setting the LVDE and LVDSE bits in the SPMSC1 register.

On this MCU, the internal reference is not connected to any module that is operational in stop mode. Therefore, the IREFSTEN bit in the ICSC1 register should always be cleared.

Figure 10-1 shows the MC9S08QG8/4 block diagram with the ICS highlighted.

### 10.1.2 Factory Trim Value

A factory trim value is stored in FLASH during production testing. To be used, this value must be copied from FLASH memory to the ICSTRM register. A factory value for this FTRIM bit is also stored in FLASH and must be copied into the FTRIM bit in the ICSSC register. See Table 4-4 for the FLASH locations of the factory ICSTRM and FTRIM values.




NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

**Figure 10-1. MC9S08QG8/4 Block Diagram Highlighting ICS Block and Pins**

### 10.1.3 Features

Key features of the ICS module are:

- Frequency-locked loop (FLL) is trimmable for accuracy
  - 0.2% resolution using internal 32 kHz reference
  - 2% deviation over voltage and temperature using internal 32 kHz reference
- External reference clock up to 5 MHz can be used to control the FLL
  - 3 bit select for reference divider is provided
- Internal reference clock has 9 trim bits available
- Internal or external reference clock can be selected as the clock source for the MCU
- Whichever clock is selected as the source can be divided down
  - 2 bit select for clock divider is provided
    - Allowable dividers are: 1, 2, 4, 8
    - BDC clock is provided as a constant divide by 2 of the DCO output
- Control signals for a low power oscillator as the external reference clock are provided
  - HGO, RANGE, EREFS, ERCLKEN, EREFSTEN
- FLL engaged internal mode is automatically selected out of reset

### 10.1.4 Modes of Operation

The ICS features the following modes of operation: FEI, FEE, FBI, FBILP, FBE, FBELP, and stop.

#### 10.1.4.1 FLL Engaged Internal (FEI)

In FLL engaged internal mode, which is the default mode, the ICS supplies a clock derived from the FLL which is controlled by the internal reference clock. The BDC clock is supplied from the FLL.

#### 10.1.4.2 FLL Engaged External (FEE)

In FLL engaged external mode, the ICS supplies a clock derived from the FLL which is controlled by an external reference clock. The BDC clock is supplied from the FLL.

#### 10.1.4.3 FLL Bypassed Internal (FBI)

In FLL bypassed internal mode, the FLL is enabled and controlled by the internal reference clock, but is bypassed. The ICS supplies a clock derived from the internal reference clock. The BDC clock is supplied from the FLL.

#### 10.1.4.4 FLL Bypassed Internal Low Power (FBILP)

In FLL bypassed internal low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the internal reference clock. The BDC clock is not available.

### 10.1.4.5 FLL Bypassed External (FBE)

In FLL bypassed external mode, the FLL is enabled and controlled by an external reference clock, but is bypassed. The ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source. The BDC clock is supplied from the FLL.

### 10.1.4.6 FLL Bypassed External Low Power (FBELP)

In FLL bypassed external low power mode, the FLL is disabled and bypassed, and the ICS supplies a clock derived from the external reference clock. The external reference clock can be an external crystal/resonator supplied by an OSC controlled by the ICS, or it can be another external clock source. The BDC clock is not available.

### 10.1.4.7 Stop (STOP)

In stop mode, the FLL is disabled and the internal or external reference clock can be selected to be enabled or disabled. The BDC clock is not available. ICS does not provide an MCU clock source.

## 10.1.5 Block Diagram

This section contains the ICS block diagram.

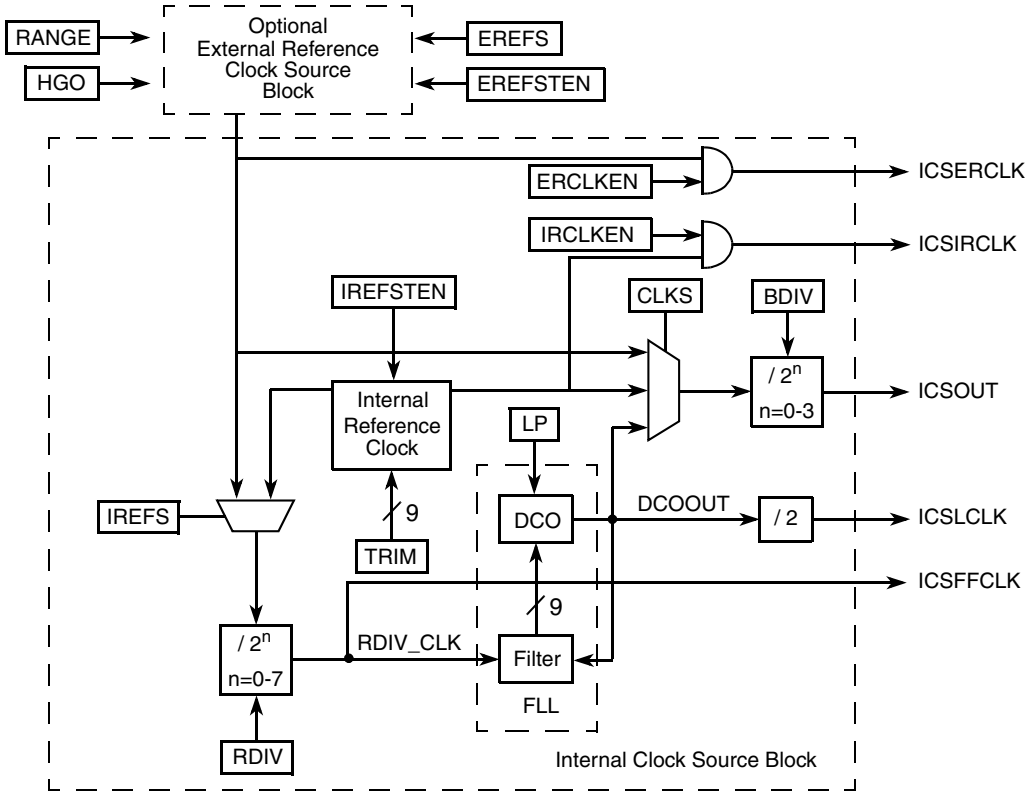



**Figure 10-2. Internal Clock Source (ICS) Block Diagram**

## 10.2 External Signal Description

No ICS signal connects off chip.

## 10.3 Register Definition

### 10.3.1 ICS Control Register 1 (ICSC1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CLKS | | RDIV | | | IREFS | IRCLKEN | IREFSTEN |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Figure 10-3. ICS Control Register 1 (ICSC1)**

**Table 10-1. ICS Control Register 1 Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>CLKS | **Clock Source Select** — Selects the clock source that controls the bus frequency. The actual bus frequency depends on the value of the BDIV bits.<br>00 Output of FLL is selected.<br>01 Internal reference clock is selected.<br>10 External reference clock is selected.<br>11 Reserved, defaults to 00. |
| 5:3<br>RDIV | **Reference Divider** — Selects the amount to divide down the FLL reference clock selected by the IREFS bits. Resulting frequency must be in the range 31.25 kHz to 39.0625 kHz.<br>000 Encoding 0 — Divides reference clock by 1 (reset default)<br>001 Encoding 1 — Divides reference clock by 2<br>010 Encoding 2 — Divides reference clock by 4<br>011 Encoding 3 — Divides reference clock by 8<br>100 Encoding 4 — Divides reference clock by 16<br>101 Encoding 5 — Divides reference clock by 32<br>110 Encoding 6 — Divides reference clock by 64<br>111 Encoding 7 — Divides reference clock by 128 |
| 2<br>IREFS | **Internal Reference Select** — The IREFS bit selects the reference clock source for the FLL.<br>1 Internal reference clock selected<br>0 External reference clock selected |
| 1<br>IRCLKEN | **Internal Reference Clock Enable** — The IRCLKEN bit enables the internal reference clock for use as ICSIRCLK.<br>1 ICSIRCLK active<br>0 ICSIRCLK inactive |
| 0<br>IREFSTEN | **Internal Reference Stop Enable** — The IREFSTEN bit controls whether or not the internal reference clock remains enabled when the ICS enters stop mode.<br>1 Internal reference clock stays enabled in stop if IRCLKEN is set or if ICS is in FEI, FBI, or FBILP mode before entering stop<br>0 Internal reference clock is disabled in stop |

## 10.3.2 ICS Control Register 2 (ICSC2)

**Figure 10-4. ICS Control Register 2 (ICSC2)**

**Table 10-2. ICS Control Register 2 Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>BDIV | **Bus Frequency Divider** — Selects the amount to divide down the clock source selected by the CLKS bits. This controls the bus frequency.<br>00 Encoding 0 — Divides selected clock by 1<br>01 Encoding 1 — Divides selected clock by 2 (reset default)<br>10 Encoding 2 — Divides selected clock by 4<br>11 Encoding 3 — Divides selected clock by 8 |
| 5<br>RANGE | **Frequency Range Select** — Selects the frequency range for the external oscillator.<br>1 High frequency range selected for the external oscillator<br>0 Low frequency range selected for the external oscillator |
| 4<br>HGO | **High Gain Oscillator Select** — The HGO bit controls the external oscillator mode of operation.<br>1 Configure external oscillator for high gain operation<br>0 Configure external oscillator for low power operation |
| 3<br>LP | **Low Power Select** — The LP bit controls whether the FLL is disabled in FLL bypassed modes.<br>1 FLL is disabled in bypass modes unless BDM is active<br>0 FLL is not disabled in bypass mode |
| 2<br>EREFS | **External Reference Select** — The EREFS bit selects the source for the external reference clock.<br>1 Oscillator requested<br>0 External Clock Source requested |
| 1<br>ERCLKEN | **External Reference Enable** — The ERCLKEN bit enables the external reference clock for use as ICSERCLK.<br>1 ICSERCLK active<br>0 ICSERCLK inactive |
| 0<br>EREFSTEN | **External Reference Stop Enable** — The EREFSTEN bit controls whether or not the external reference clock remains enabled when the ICS enters stop mode.<br>1 External reference clock stays enabled in stop if ERCLKEN is set or if ICS is in FEE, FBE, or FBELP mode before entering stop<br>0 External reference clock is disabled in stop |

### 10.3.3 ICS Trim Register (ICSTRM)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | TRIM | | | | | | | |
| POR: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset: | U | U | U | U | U | U | U | U |

**Figure 10-5. ICS Trim Register (ICSTRM)**

**Table 10-3. ICS Trim Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>TRIM | **ICS Trim Setting** — The TRIM bits control the internal reference clock frequency by controlling the internal reference clock period. The bits' effect are binary weighted (i.e., bit 1 will adjust twice as much as bit 0). Increasing the binary value in TRIM will increase the period, and decreasing the value will decrease the period.<br><br>An additional fine trim bit is available in ICSSC as the FTRIM bit. |

### 10.3.4 ICS Status and Control (ICSSC)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | CLKST | | OSCINIT | FTRIM |
| W | | | | | | | | |
| POR: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | U |

**Figure 10-6. ICS Status and Control Register (ICSSC)**

**Table 10-4. ICS Status and Control Register Field Descriptions**

| Field | Description |
|---|---|
| 3:2<br>CLKST | **Clock Mode Status** — The CLKST bits indicate the current clock mode. The CLKST bits don't update immediately after a write to the CLKS bits due to internal synchronization between clock domains.<br>00 Output of FLL is selected.<br>01 FLL Bypassed, Internal reference clock is selected.10FLL Bypassed, External reference clock is selected.<br>11 Reserved. |
| 1 | **OSC Initialization** — If the external reference clock is selected by ERCLKEN or by the ICS being in FEE, FBE, or FBELP mode, and if EREFS is set, then this bit is set after the initialization cycles of the external oscillator clock have completed. This bit is only cleared when either ERCLKEN or EREFS are cleared. |
| FTRIM<br>0 | **ICS Fine Trim** — The FTRIM bit controls the smallest adjustment of the internal reference clock frequency. Setting FTRIM will increase the period and clearing FTRIM will decrease the period by the smallest amount possible. |

## 10.4 Functional Description

### 10.4.1 Operational Modes

The states of the ICS are shown as a state diagram and are described in the following sections. The arrows indicate the allowed movements between the states.

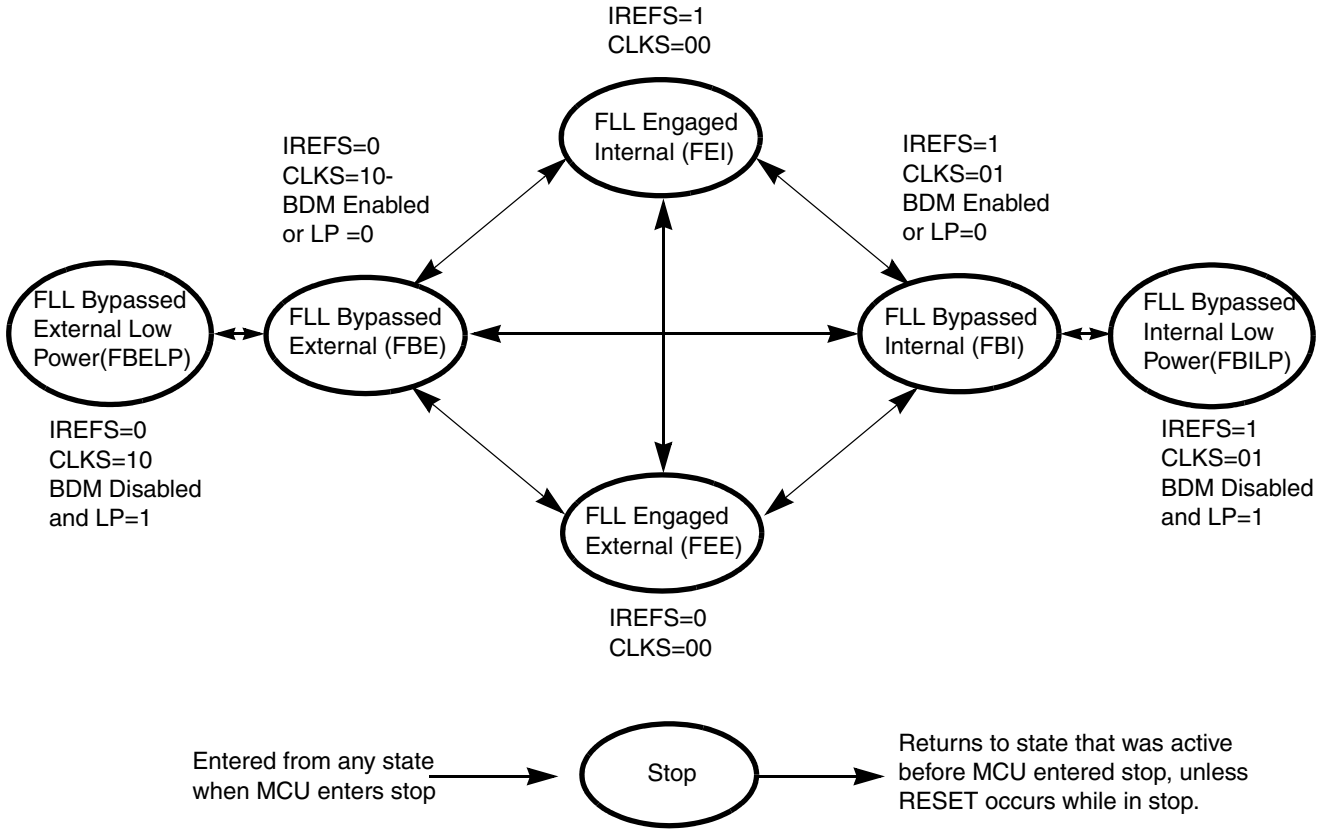



**Figure 10-7. Clock Switching Modes**

#### 10.4.1.1 FLL Engaged Internal (FEI)

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

- CLKS bits are written to 00
- IREFS bit is written to 1
- RDIV bits are written to divide reference clock to be within the range of 31.25 kHz to 39.0625 kHz.

In FLL engaged internal mode, the ICSOUT clock is derived from the FLL clock, which is controlled by the internal reference clock. The FLL loop will lock the frequency to 512 times the filter frequency, as selected by the RDIV bits. The ICSLCLK is available for BDC communications, and the internal reference clock is enabled.

#### 10.4.1.2 FLL Engaged External (FEE)

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

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

In FLL engaged external mode, the ICSOUT clock is derived from the FLL clock which is controlled by the external reference clock.The FLL loop will lock the frequency to 512 times the filter frequency, as selected by the RDIV bits. The ICSLCLK is available for BDC communications, and the external reference clock is enabled.

#### 10.4.1.3 FLL Bypassed Internal (FBI)

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

- CLKS bits are written to 01
- IREFS bit is written to 1
- BDM mode is active or LP bit is written to 0

In FLL bypassed internal mode, the ICSOUT clock is derived from the internal reference clock. The FLL clock is controlled by the internal reference clock, and the FLL loop will lock the FLL frequency to 512 times the filter frequency, as selected by the RDIV bits. The ICSLCLK will be available for BDC communications, and the internal reference clock is enabled.

#### 10.4.1.4 FLL Bypassed Internal Low Power (FBILP)

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

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

In FLL bypassed internal low power mode, the ICSOUT clock is derived from the internal reference clock and the FLL is disabled. The ICSLCLK will be not be available for BDC communications, and the internal reference clock is enabled.

#### 10.4.1.5 FLL Bypassed External (FBE)

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

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- BDM mode is active or LP bit is written to 0.

In FLL bypassed external mode, the ICSOUT clock is derived from the external reference clock. The FLL clock is controlled by the external reference clock, and the FLL loop will lock the FLL frequency to 512

times the filter frequency, as selected by the RDIV bits, so that the ICSLCLK will be available for BDC communications, and the external reference clock is enabled.

#### 10.4.1.6 FLL Bypassed External Low Power (FBELP)

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

- CLKS bits are written to 10.
- IREFS bit is written to 0.
- BDM mode is not active and LP bit is written to 1.

In FLL bypassed external low power mode, the ICSOUT clock is derived from the external reference clock and the FLL is disabled. The ICSLCLK will be not be available for BDC communications. The external reference clock is enabled.

#### 10.4.1.7 Stop

ICS stop mode is entered whenever the MCU enters stop. In this mode, all ICS clock signals are stopped except in the following cases:

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

- IRCLKEN bit is written to 1
- IREFSTEN bit is written to 1

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

- ERCLKEN bit is written to 1
- EREFSTEN bit is written to 1

### 10.4.2 Mode Switching

When switching between FLL engaged internal (FEI) and FLL engaged external (FEE) modes the IREFS bit can be changed at anytime, but the RDIV bits must be changed simultaneously so that the resulting frequency stays in the range of 31.25 kHz to 39.0625 kHz. After a change in the IREFS value the FLL will begin locking again after a few full cycles of the resulting divided reference frequency.

The CLKS bits can also be changed at anytime, but the RDIV bits must be changed simultaneously so that the resulting frequency stays in the range of 31.25 kHz to 39.0625 kHz. The actual switch to the newly selected clock will not occur until after a few full cycles of the new clock. If the newly selected clock is not available, the previous clock will remain selected.

### 10.4.3 Bus Frequency Divider

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

### 10.4.4 Low Power Bit Usage

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

### 10.4.5 Internal Reference Clock

When IRCLKEN is set the internal reference clock signal will be presented as ICSIRCLK, which can be used as an additional clock source. The ICSIRCLK frequency can be re-targeted by trimming the period of the internal reference clock. This can be done by writing a new value to the TRIM bits in the ICSTRM register. Writing a larger value will slow down the ICSIRCLK frequency, and writing a smaller value to the ICSTRM register will speed up the ICSIRCLK frequency. The TRIM bits will effect the ICSOUT frequency if the ICS is in FLL engaged internal (FEI), FLL bypassed internal (FBI), or FLL bypassed internal low power (FBILP) mode. The TRIM and FTRIM value will not be affected by a reset.

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

If IREFSTEN is set and the IRCLKEN bit is written to 1, the internal reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

All MCU devices are factory programmed with a trim value in a reserved memory location. This value can be copied to the ICSTRM register during reset initialization. The factory trim value does not include the FTRIM bit. For finer precision, the user can trim the internal oscillator in the application and set the FTRIM bit accordingly.

### 10.4.6 Optional External Reference Clock

The ICS module can support an external reference clock with frequencies between 31.25 kHz to 5 MHz in all modes. When the ERCLKEN is set, the external reference clock signal will be presented as ICSERCLK, which can be used as an additional clock source. When IREFS = 1, the external reference clock will not be used by the FLL and will only be used as ICSERCLK. In these modes, the frequency can be equal to the maximum frequency the chip-level timing specifications will support (see the Device Overview chapter).

If EREFSTEN is set and the ERCLKEN bit is written to 1, the external reference clock will keep running during stop mode in order to provide a fast recovery upon exiting stop.

### 10.4.7 Fixed Frequency Clock

The ICS provides the divided FLL reference clock as ICSFFCLK for use as an additional clock source for peripheral modules. The ICS provides an output signal (ICSFFE) which indicates when the ICS is providing ICSOUT frequencies four times or greater than the divided FLL reference clock (ICSFFCLK). In FLL engaged mode (FEI and FEE), this is always true and ICSFFE is always high. In ICS Bypass modes, ICSFFE will get asserted for the following combinations of BDIV and RDIV values:

- BDIV=00 (divide by 1), RDIV ≥ 010
- BDIV=01 (divide by 2), RDIV ≥ 011
- BDIV=10 (divide by 4), RDIV ≥ 100
- BDIV=11 (divide by 8), RDIV ≥ 101

# Chapter 11
# Inter-Integrated Circuit (S08IICV1)

## 11.1 Introduction

The inter-integrated circuit (IIC) provides a method of communication between a number of devices. The interface is designed to operate up to 100 kbps with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

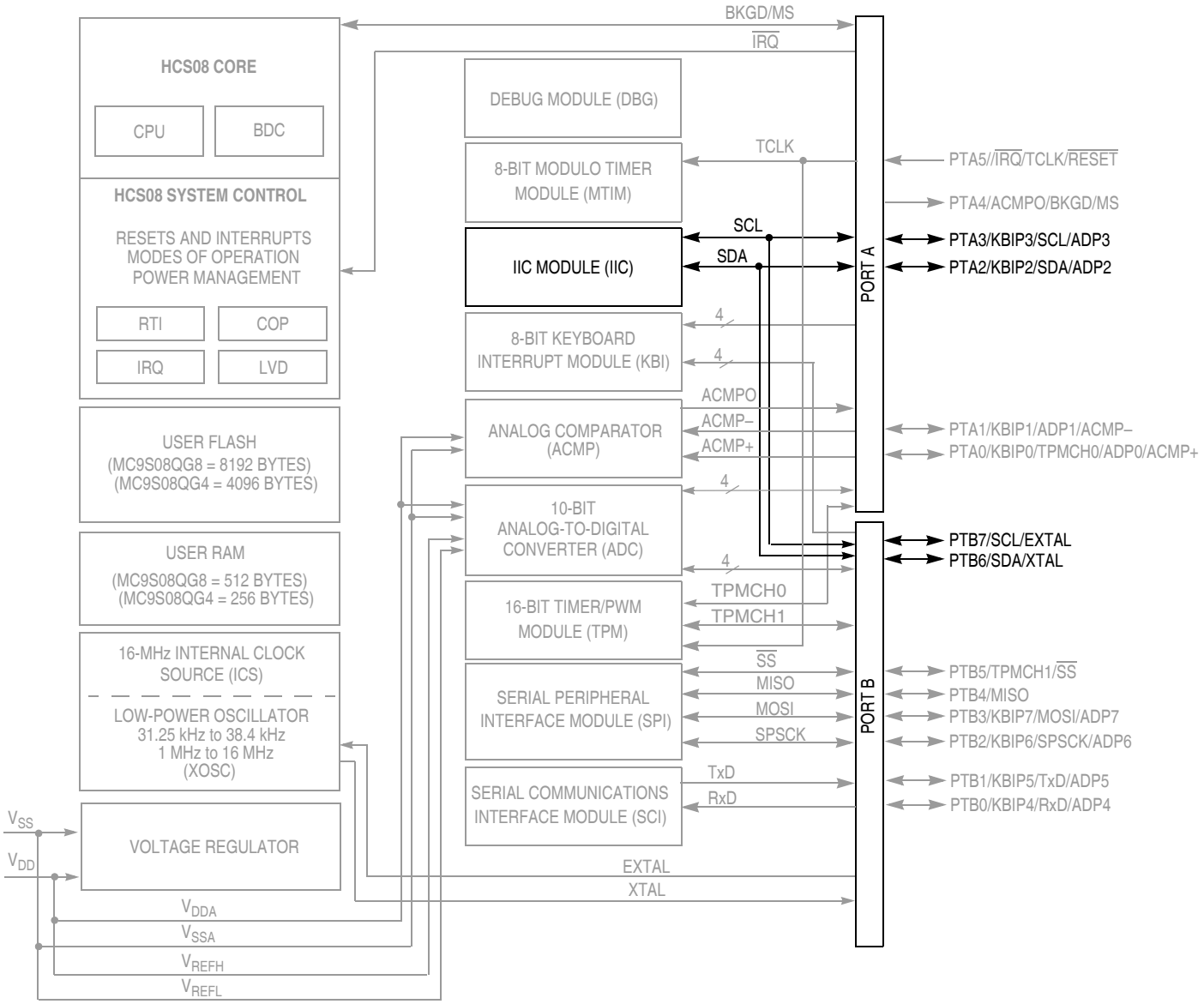
### 11.1.1 Module Configuration

The IIC module pins, SDA and SCL can be repositioned under software control using IICPS in SOPT2 as as shown in Table 11-1. IICPS in SOPT2 selects which general-purpose I/O ports are associated with IIC operation.

**Table 11-1. IIC Position Options**

| IICPS in SOPT2 | Port Pin for SDA | Port Pin for SCL |
|---|---|---|
| 0 (default) | PTA2 | PTA3 |
| 1 | PTB6 | PTB7 |

Figure 11-1 is the MC9S08QG8/4 block diagram with the IIC block highlighted.




NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

**Figure 11-1. MC9S08QG8/4 Block Diagram Highlighting IIC Block and Pins**

### 11.1.2 Features

The IIC includes these distinctive features:

- Compatible with IIC bus standard
- Multi-master operation
- Software programmable for one of 64 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus busy detection

### 11.1.3 Modes of Operation

The IIC functions the same in normal and monitor modes. A brief description of the IIC in the various MCU modes is given here.

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

### 11.1.4 Block Diagram

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

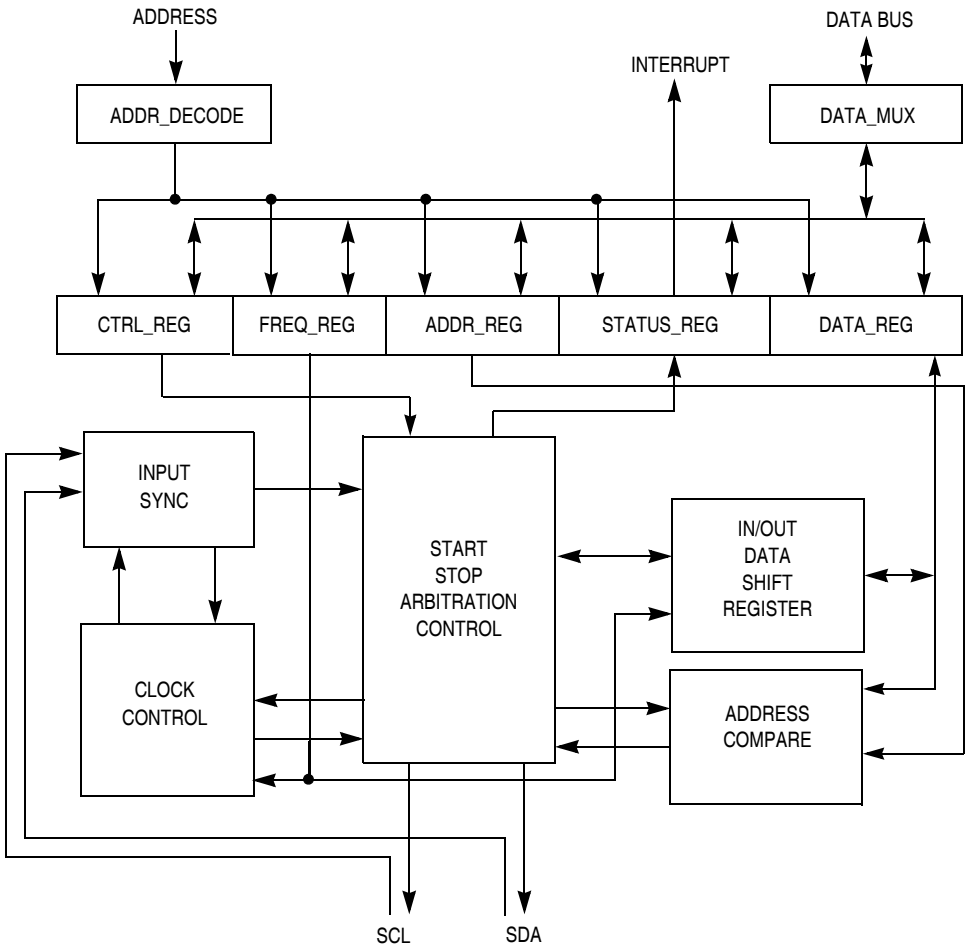



**Figure 11-2. IIC Functional Block Diagram**

## 11.2 External Signal Description

This section describes each user-accessible pin signal.

### 11.2.1 SCL — Serial Clock Line

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

### 11.2.2 SDA — Serial Data Line

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

## 11.3 Register Definition

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

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

## 11.3.1 IIC Address Register (IICA)

**Figure 11-3. IIC Address Register (IICA)**

**Table 11-2. IICA Register Field Descriptions**

| Field | Description |
|---|---|
| 7:1<br>ADDR[7:1] | **IIC Address Register** — The ADDR contains the specific slave address to be used by the IIC module. This is the address the module will respond to when addressed as a slave. |

## 11.3.2 IIC Frequency Divider Register (IICF)

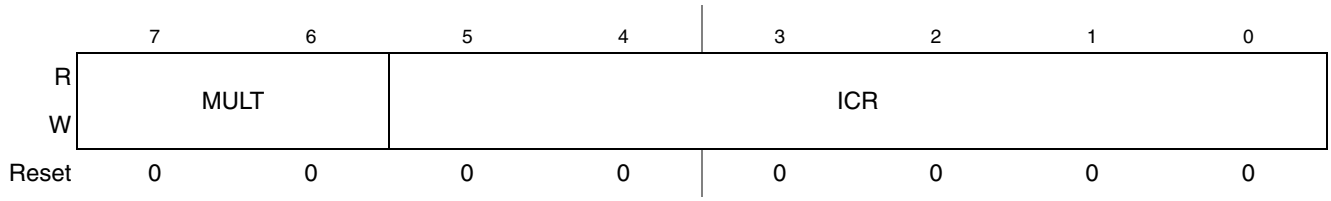



**Figure 11-4. IIC Frequency Divider Register (IICF)**

**Table 11-3. IICF Register Field Descriptions**

| Field | Description |
|---|---|
| 7:6<br>MULT | **IIC Multiplier Factor** — The MULT bits define the multiplier factor mul. This factor is used along with the SCL divider to generate the IIC baud rate. The multiplier factor mul as defined by the MULT bits is provided below.<br>00 mul = 01<br>01 mul = 02<br>10 mul = 04<br>11 Reserved |
| 5:0<br>ICR | **IIC Clock Rate** — The ICR bits are used to prescale the bus clock for bit rate selection. These bits are used to define the SCL divider and the SDA hold value. The SCL divider multiplied by the value provided by the MULT register (multiplier factor mul) is used to generate IIC baud rate.<br>IIC baud rate = bus speed (Hz)/(mul * SCL divider)<br>SDA hold time is the delay from the falling edge of the SCL (IIC clock) to the changing of SDA (IIC data). The ICR is used to determine the SDA hold value.<br>SDA hold time = bus period (s) * SDA hold value<br>Table 11-4 provides the SCL divider and SDA hold values for corresponding values of the ICR. These values can be used to set IIC baud rate and SDA hold time. For example:<br>Bus speed = 8 MHz<br>MULT is set to 01 (mul = 2)<br>Desired IIC baud rate = 100 kbps<br><br>IIC baud rate = bus speed (Hz)/(mul * SCL divider)<br>100000 = 8000000/(2*SCL divider)<br>SCL divider = 40<br>Table 11-4 shows that ICR must be set to 0B to provide an SCL divider of 40 and that this will result in an SDA hold value of 9.<br>SDA hold time = bus period (s) * SDA hold value<br>SDA hold time = 1/8000000 * 9 = 1.125 μs<br>If the generated SDA hold value is not acceptable, the MULT bits can be used to change the ICR. This will result in a different SDA hold value. |

**Table 11-4. IIC Divider and Hold Values**

| ICR (hex) | SCL Divider | SDA Hold Value |
|---|---|---|
| 00 | 20 | 7 |
| 01 | 22 | 7 |
| 02 | 24 | 8 |
| 03 | 26 | 8 |
| 04 | 28 | 9 |
| 05 | 30 | 9 |
| 06 | 34 | 10 |
| 07 | 40 | 10 |
| 08 | 28 | 7 |
| 09 | 32 | 7 |
| 0A | 36 | 9 |
| 0B | 40 | 9 |
| 0C | 44 | 11 |
| 0D | 48 | 11 |
| 0E | 56 | 13 |
| 0F | 68 | 13 |
| 10 | 48 | 9 |
| 11 | 56 | 9 |
| 12 | 64 | 13 |
| 13 | 72 | 13 |
| 14 | 80 | 17 |
| 15 | 88 | 17 |
| 16 | 104 | 21 |
| 17 | 128 | 21 |
| 18 | 80 | 9 |
| 19 | 96 | 9 |
| 1A | 112 | 17 |
| 1B | 128 | 17 |
| 1C | 144 | 25 |
| 1D | 160 | 25 |
| 1E | 192 | 33 |
| 1F | 240 | 33 |

| ICR (hex) | SCL Divider | SDA Hold Value |
|---|---|---|
| 20 | 160 | 17 |
| 21 | 192 | 17 |
| 22 | 224 | 33 |
| 23 | 256 | 33 |
| 24 | 288 | 49 |
| 25 | 320 | 49 |
| 26 | 384 | 65 |
| 27 | 480 | 65 |
| 28 | 320 | 33 |
| 29 | 384 | 33 |
| 2A | 448 | 65 |
| 2B | 512 | 65 |
| 2C | 576 | 97 |
| 2D | 640 | 97 |
| 2E | 768 | 129 |
| 2F | 960 | 129 |
| 30 | 640 | 65 |
| 31 | 768 | 65 |
| 32 | 896 | 129 |
| 33 | 1024 | 129 |
| 34 | 1152 | 193 |
| 35 | 1280 | 193 |
| 36 | 1536 | 257 |
| 37 | 1920 | 257 |
| 38 | 1280 | 129 |
| 39 | 1536 | 129 |
| 3A | 1792 | 257 |
| 3B | 2048 | 257 |
| 3C | 2304 | 385 |
| 3D | 2560 | 385 |
| 3E | 3072 | 513 |
| 3F | 3840 | 513 |

## 11.3.3 IIC Control Register (IICC)

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

= Unimplemented or Reserved

**Figure 11-5. IIC Control Register (IICC)**

**Table 11-5. IICC Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>IICEN | **IIC Enable** — The IICEN bit determines whether the IIC module is enabled.<br>0 IIC is not enabled.<br>1 IIC is enabled. |
| 6<br>IICIE | **IIC Interrupt Enable** — The IICIE bit determines whether an IIC interrupt is requested.<br>0 IIC interrupt request not enabled.<br>1 IIC interrupt request enabled. |
| 5<br>MST | **Master Mode Select** — The MST bit is changed from a 0 to a 1 when a START signal is generated on the bus and master mode is selected. When this bit changes from a 1 to a 0 a STOP signal is generated and the mode of operation changes from master to slave.<br>0 Slave Mode.<br>1 Master Mode. |
| 4<br>TX | **Transmit Mode Select** — The TX bit selects the direction of master and slave transfers. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. When addressed as a slave this bit should be set by software according to the SRW bit in the status register.<br>0 Receive.<br>1 Transmit. |
| 3<br>TXAK | **Transmit Acknowledge Enable** — This bit specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers.<br>0 An acknowledge signal will be sent out to the bus after receiving one data byte.<br>1 No acknowledge signal response is sent. |
| 2<br>RSTA | **Repeat START** — Writing a one to this bit will generate a repeated START condition provided it is the current master. This bit will always be read as a low. Attempting a repeat at the wrong time will result in loss of arbitration. |

## 11.3.4 IIC Status Register (IICS)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TCF | IAAS | BUSY | ARBL | 0 | SRW | IICIF | RXAK |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 11-6. IIC Status Register (IICS)**

**Table 11-6. IICS Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TCF | **Transfer Complete Flag** — This bit is set on the completion of a byte transfer. Note that this bit is only valid during or immediately following a transfer to the IIC module or from the IIC module.The TCF bit is cleared by reading the IICD register in receive mode or writing to the IICD in transmit mode.<br>0 Transfer in progress.<br>1 Transfer complete. |
| 6<br>IAAS | **Addressed as a Slave** — The IAAS bit is set when the calling address matches the programmed slave address. Writing the IICC register clears this bit.<br>0 Not addressed.<br>1 Addressed as a slave. |
| 5<br>BUSY | **Bus Busy** — The BUSY bit indicates the status of the bus regardless of slave or master mode. The BUSY bit is set when a START signal is detected and cleared when a STOP signal is detected.<br>0 Bus is idle.<br>1 Bus is busy. |
| 4<br>ARBL | **Arbitration Lost** — This bit is set by hardware when the arbitration procedure is lost. The ARBL bit must be cleared by software, by writing a one to it.<br>0 Standard bus operation.<br>1 Loss of arbitration. |
| 2<br>SRW | **Slave Read/Write** — When addressed as a slave the SRW bit indicates the value of the R/W command bit of the calling address sent to the master.<br>0 Slave receive, master writing to slave.<br>1 Slave transmit, master reading from slave. |
| 1<br>IICIF | **IIC Interrupt Flag** — The IICIF bit is set when an interrupt is pending. This bit must be cleared by software, by writing a one to it in the interrupt routine. One of the following events can set the IICIF bit:<br>• One byte transfer completes<br>• Match of slave address to calling address<br>• Arbitration lost<br>0 No interrupt pending.<br>1 Interrupt pending. |
| 0<br>RXAK | **Receive Acknowledge** — When the RXAK bit is low, it indicates an acknowledge signal has been received after the completion of one byte of data transmission on the bus. If the RXAK bit is high it means that no acknowledge signal is detected.<br>0 Acknowledge received.<br>1 No acknowledge received. |

### 11.3.5 IIC Data I/O Register (IICD)

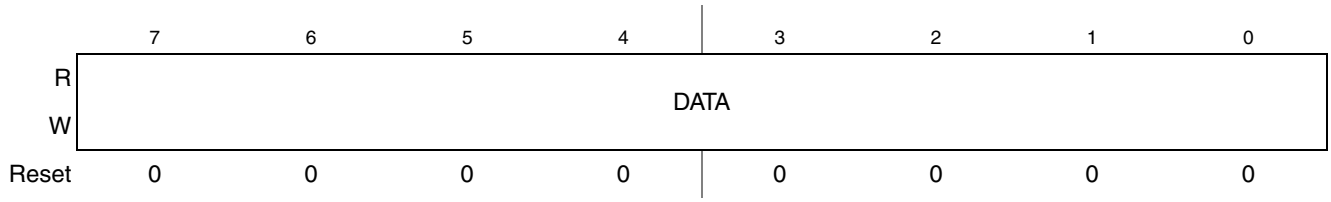



**Figure 11-7. IIC Data I/O Register (IICD)**

**Table 11-7. IICD Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>DATA | **Data** — In master transmit mode, when data is written to the IICD, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates receiving of the next byte of data. |

**NOTE**

When transmitting out of master receive mode, the IIC mode should be switched before reading the IICD register to prevent an inadvertent initiation of a master receive data transfer.

In slave mode, the same functions are available after an address match has occurred.

Note that the TX bit in IICC must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the IIC is configured for master transmit but a master receive is desired, then reading the IICD will not initiate the receive.

Reading the IICD will return the last byte received while the IIC is configured in either master receive or slave receive modes. The IICD does not reflect every byte that is transmitted on the IIC bus, nor can software verify that a byte has been written to the IICD correctly by reading it back.

In master transmit mode, the first byte of data written to IICD following assertion of MST is used for the address transfer and should comprise of the calling address (in bit 7–bit 1) concatenated with the required R/W bit (in position bit 0).

## 11.4 Functional Description

This section provides a complete functional description of the IIC module.

### 11.4.1 IIC Protocol

The IIC bus system uses a serial data line (SDA) and a serial clock line (SCL) for data transfer. All devices connected to it must have open drain or open collector outputs. A logic AND function is exercised on both lines with external pull-up resistors. The value of these resistors is system dependent.

Normally, a standard communication is composed of four parts:

- START signal
- Slave address transmission
- Data transfer
- STOP signal

The STOP signal should not be confused with the CPU STOP instruction. The IIC bus system communication is described briefly in the following sections and illustrated in Figure 11-8.

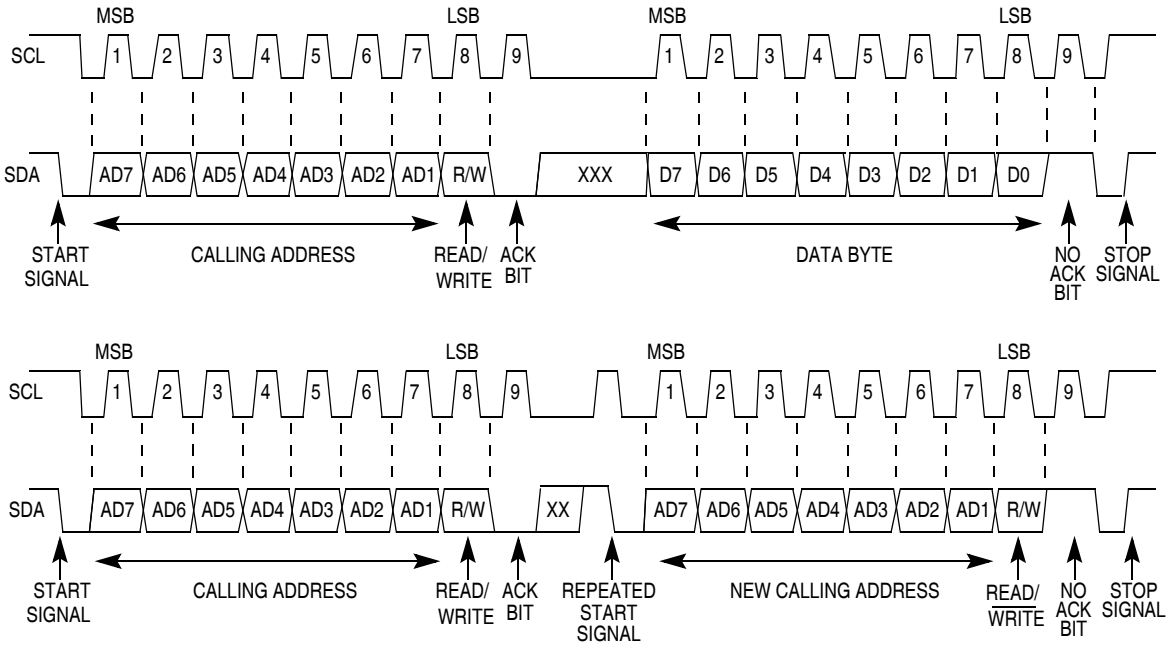



**Figure 11-8. IIC Bus Transmission Signals**

#### 11.4.1.1 START Signal

When the bus is free; i.e., no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in Figure 11-8, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

#### 11.4.1.2 Slave Address Transmission

The first byte of data transferred immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 11-8).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address that is equal to its own slave address. The IIC cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the IIC will revert to slave mode and operate correctly even if it is being addressed by another master.

#### 11.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master.

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 11-8. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the 9th bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets this as an end of data transfer and releases the SDA line.

In either case, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a STOP signal.
- Commences a new calling by generating a repeated START signal.

#### 11.4.1.4 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see Figure 11-8).

The master can generate a STOP even if the slave has generated an acknowledge at which point the slave must release the bus.

#### 11.4.1.5 Repeated START Signal

As shown in Figure 11-8, a repeated START signal is a START signal generated without first generating a STOP signal to terminate the communication. This is used by the master to communicate with another slave or with the same slave in different mode (transmit/receive mode) without releasing the bus.

#### 11.4.1.6 Arbitration Procedure

The IIC bus is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure, a bus master loses arbitration if it transmits logic 1 while another master transmits logic 0. The losing masters immediately switch over to slave receive mode and stop driving SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

#### 11.4.1.7 Clock Synchronization

Because wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and after a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 11-9). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.

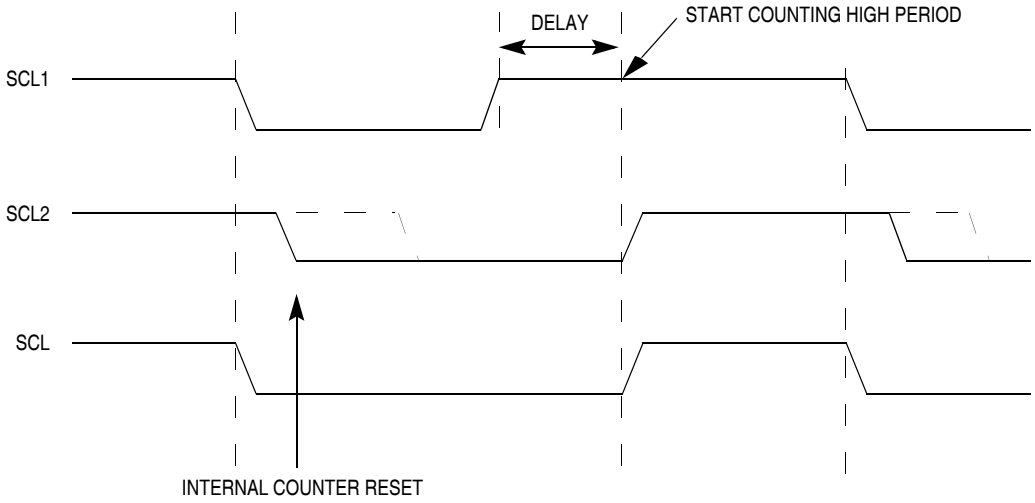



**Figure 11-9. IIC Clock Synchronization**

#### 11.4.1.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

#### 11.4.1.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

## 11.5 Resets

The IIC is disabled after reset. The IIC cannot cause an MCU reset.

## 11.6 Interrupts

The IIC generates a single interrupt.

An interrupt from the IIC is generated when any of the events in Table 11-8 occur provided the IICIE bit is set. The interrupt is driven by bit IICIF (of the IIC status register) and masked with bit IICIE (of the IIC control register). The IICIF bit must be cleared by software by writing a one to it in the interrupt routine. The user can determine the interrupt type by reading the status register.

**Table 11-8. Interrupt Summary**

| Interrupt Source | Status | Flag | Local Enable |
|---|---|---|---|
| Complete 1-byte transfer | TCF | IICIF | IICIE |
| Match of received calling address | IAAS | IICIF | IICIE |
| Arbitration Lost | ARBL | IICIF | IICIE |

### 11.6.1 Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the 9th clock to indicate the completion of byte transfer.

### 11.6.2 Address Detect Interrupt

When the calling address matches the programmed slave address (IIC address register), the IAAS bit in the status register is set. The CPU is interrupted provided the IICIE is set. The CPU must check the SRW bit and set its Tx mode accordingly.

### 11.6.3 Arbitration Lost Interrupt

The IIC is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, the relative priority of the contending masters is determined by a data arbitration procedure. The IIC module asserts this interrupt when it loses the data arbitration process and the ARBL bit in the status register is set.

Arbitration is lost in the following circumstances:

- SDA sampled as a low when the master drives a high during an address or data transmit cycle.
- SDA sampled as a low when the master drives a high during the acknowledge bit of a data receive cycle.
- A START cycle is attempted when the bus is busy.
- A repeated START cycle is requested in slave mode.
- A STOP condition is detected when the master did not request it.

This bit must be cleared by software by writing a one to it.

# 11.7 Initialization/Application Information

**Module Initialization (Slave)**

1. Write: IICA
   — to set the slave address
2. Write: IICC
   — to enable IIC and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in Figure 11-11

**Module Initialization (Master)**

1. Write: IICF
   — to set the IIC baud rate (example provided in this chapter)
2. Write: IICC
   — to enable IIC and interrupts
3. Initialize RAM variables (IICEN = 1 and IICIE = 1) for transmit data
4. Initialize RAM variables used to achieve the routine shown in Figure 11-11
5. Write: IICC
   — to enable TX
6. Write: IICC
   — to enable MST (master mode)
7. Write: IICD
   — with the address of the target slave. (The LSB of this byte will determine whether the communication is master receive or transmit.)

**Module Use**

The routine shown in Figure 11-11 can handle both master and slave IIC operations. For slave operation, an incoming IIC message that contains the proper address will begin IIC communication. For master operation, communication must be initiated by writing to the IICD register.

**Register Model**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| IICA | ADDR | | | | | | | 0 |
| | Address to which the module will respond when addressed as a slave (in slave mode) | | | | | | | |
| IICF | MULT | | | ICR | | | | |
| | Baud rate = BUSCLK / (2 x MULT x (SCL DIVIDER)) | | | | | | | |
| IICC | IICEN | IICIE | MST | TX | TXAK | RSTA | 0 | 0 |
| | Module configuration | | | | | | | |
| IICS | TCF | IAAS | BUSY | ARBL | 0 | SRW | IICIF | RXAK |
| | Module status flags | | | | | | | |
| IICD | DATA | | | | | | | |
| | Data register; Write to transmit IIC data read to read IIC data | | | | | | | |

**Figure 11-10. IIC Module Quick Start**

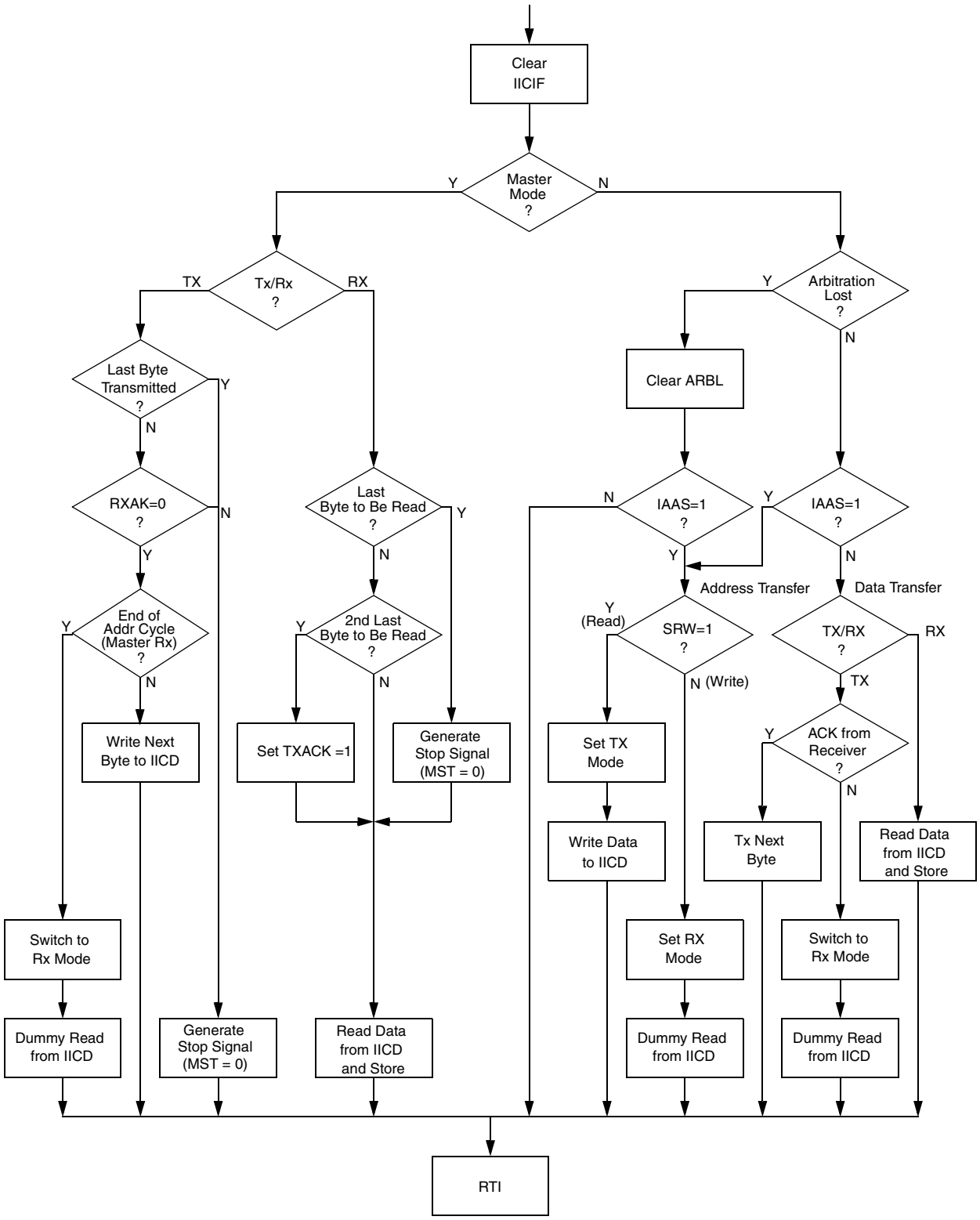


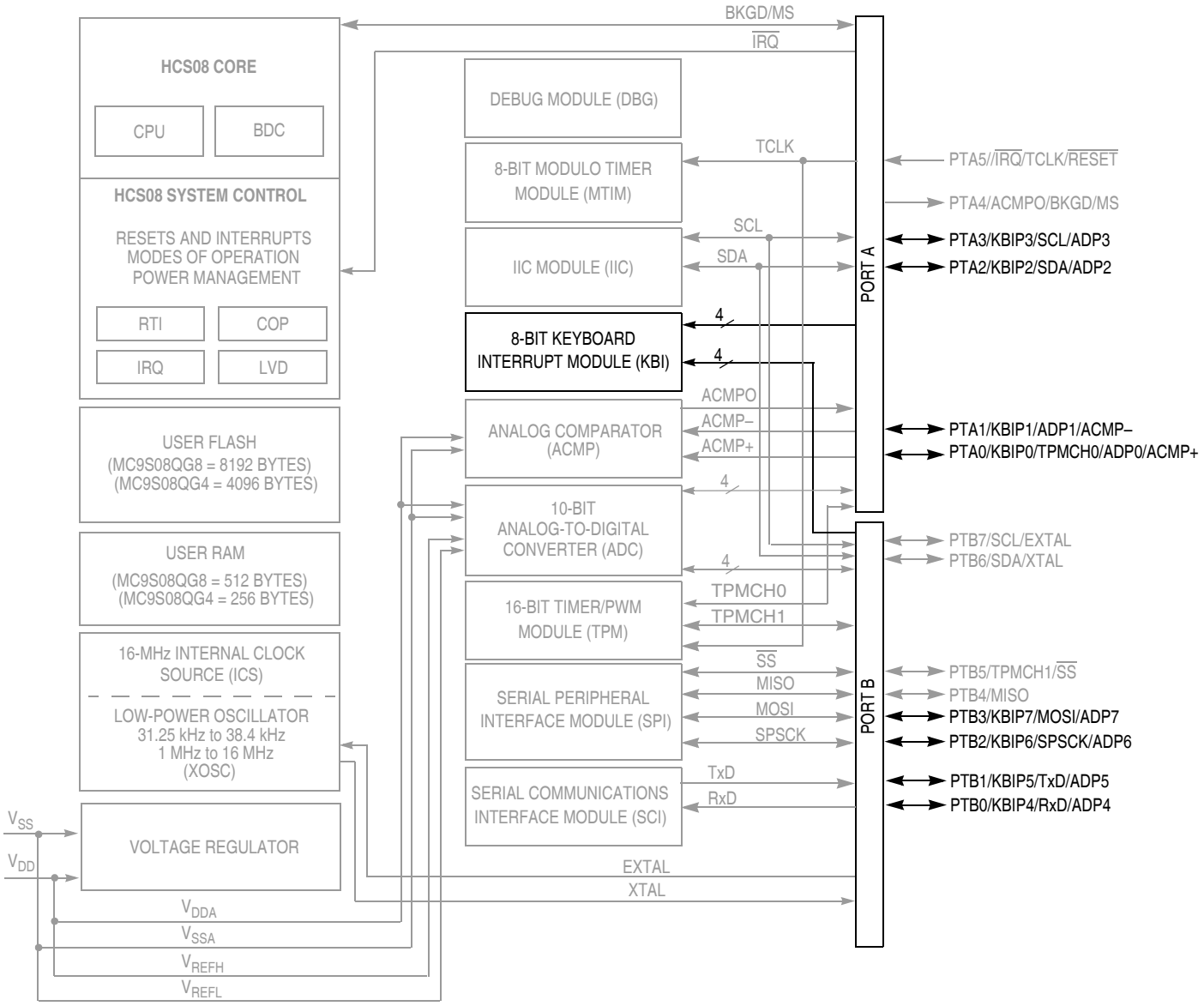

**Figure 11-11. Typical IIC Interrupt Routine**

# Chapter 12
# Keyboard Interrupt (S08KBIV2)

## 12.1 Introduction

The keyboard interrupt KBI module provides up to eight independently enabled external interrupt sources.

Figure 12-1 Shows the MC9S08QG8/4 block guide with the KBI highlighted.




NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

**Figure 12-1. MC9S08QG8/4 Block Diagram Highlighting KBI Block and Pins**

### 12.1.1 Features

The KBI features include:

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

### 12.1.2 Modes of Operation

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

#### 12.1.2.1 KBI in Wait Mode

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

#### 12.1.2.2 KBI in Stop Modes

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

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

#### 12.1.2.3 KBI in Active Background Mode

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

### 12.1.3 Block Diagram

The block diagram for the keyboard interrupt module is shown Figure 12-2.

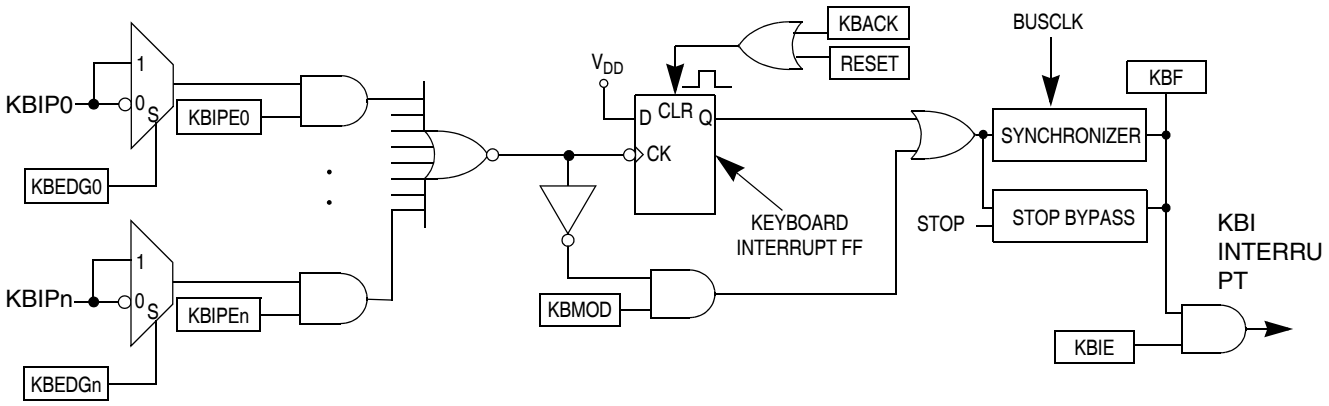



**Figure 12-2. KBI Block Diagram**

## 12.2 External Signal Description

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

The signal properties of KBI are shown in Table 12-1.

**Table 12-1. Signal Properties**

| Signal | Function | I/O |
|---|---|---|
| KBIPn | Keyboard interrupt pins | I |

## 12.3 Register Definition

The KBI includes three registers:

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

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

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

### 12.3.1 KBI Status and Control Register (KBISC)

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

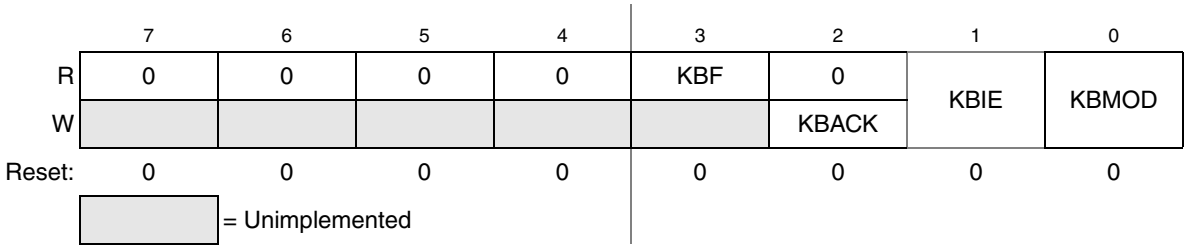



**Figure 12-3. KBI Status and Control Register**

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

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

## 12.3.2 KBI Pin Enable Register (KBIPE)

KBIPE contains the pin enable control bits.

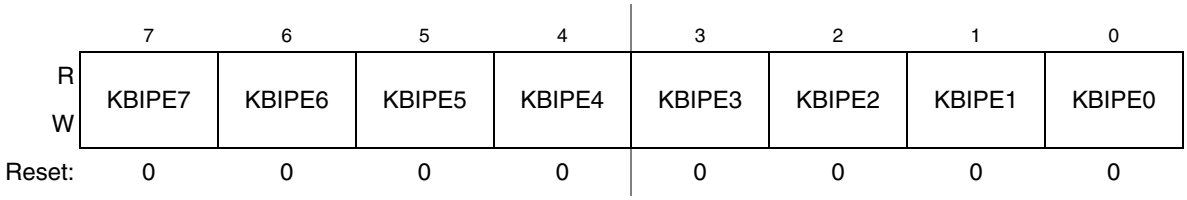



**Figure 12-4. KBI Pin Enable Register**

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

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

## 12.3.3 KBI Edge Select Register (KBIES)

KBIES contains the edge select control bits.

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

**Figure 12-5. KBI Edge Select Register**

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

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

## 12.4 Functional Description

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

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

### 12.4.1 Edge Only Sensitivity

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

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

### 12.4.2 Edge and Level Sensitivity

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

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

### 12.4.3 KBI Pullup/Pulldown Resistors

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

### 12.4.4 KBI Initialization

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

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

# Chapter 13
# Modulo Timer (S08MTIMV1)

## 13.1 Introduction

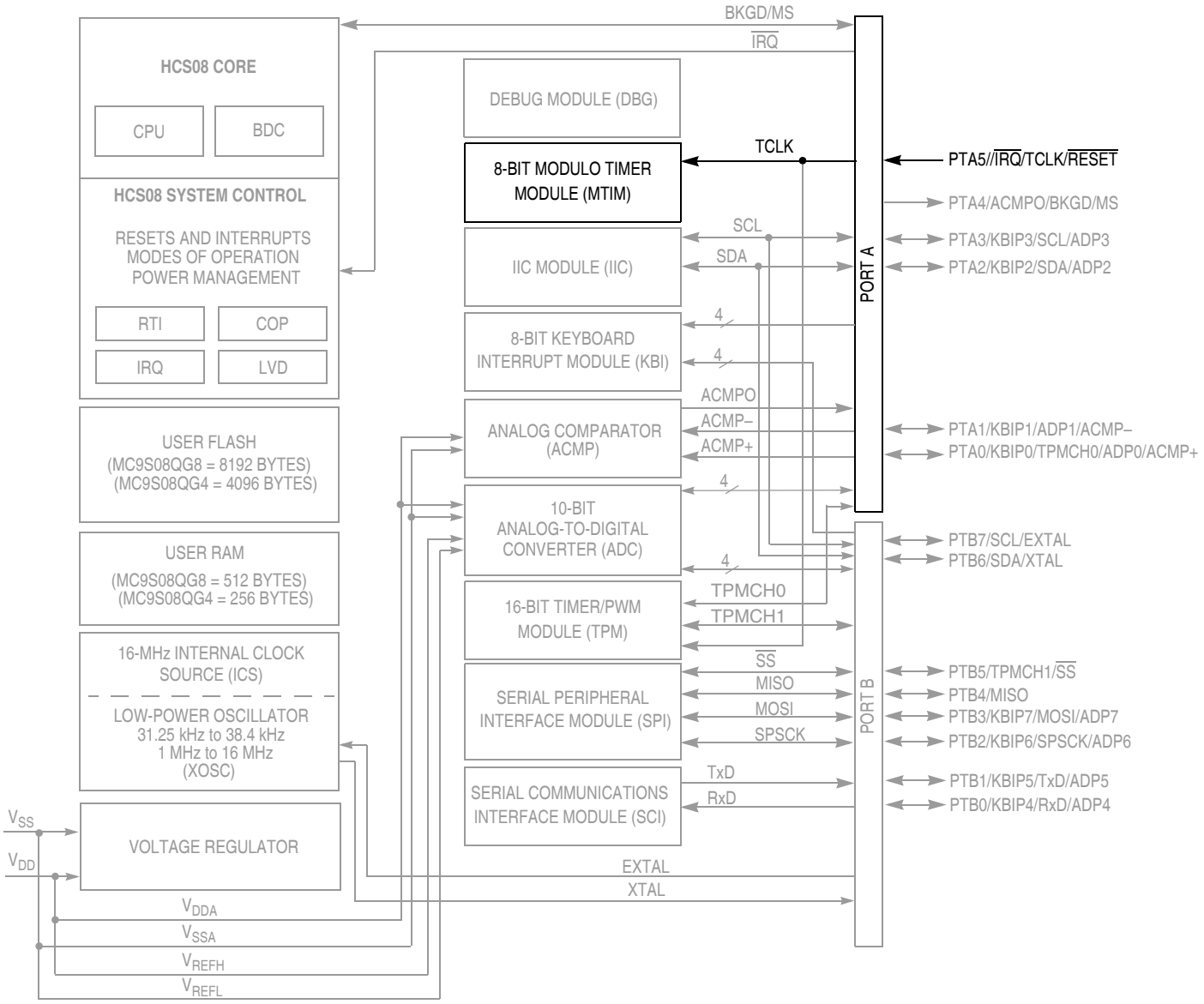
The MTIM is a simple 8-bit timer with several software selectable clock sources and a programmable interrupt.

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

Figure 13-1 shows the MC9S08QG8/4 block diagram with the MTIM highlighted.

### 13.1.1 MTIM/TPM Configuration Information

The external clock for the MTIM module, TCLK, is selected by setting CLKS = 1:1 or 1:0 in MTIMCLK, which selects the TCLK pin input. The TCLK input on PTA5 can be enabled as external clock inputs to both the MTIM and TPM modules simultaneously.




NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

**Figure 13-1. MC9S08QG8/4 Block Diagram Highlighting MTIM Block and Pins**

### 13.1.2 Features

Timer system features include:

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

### 13.1.3 Modes of Operation

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

#### 13.1.3.1 MTIM in Wait Mode

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

#### 13.1.3.2 MTIM in Stop Modes

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

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

#### 13.1.3.3 MTIM in Active Background Mode

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

### 13.1.4 Block Diagram

The block diagram for the modulo timer module is shown Figure 13-2.

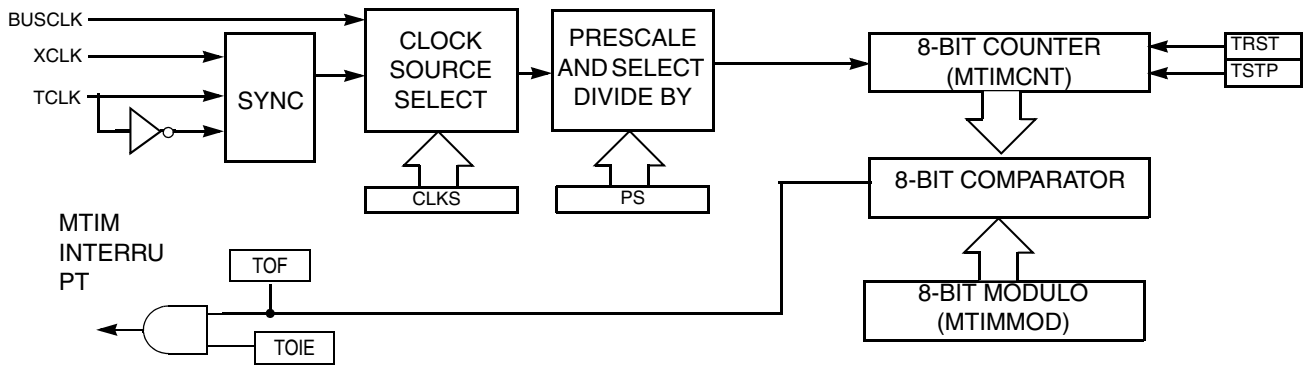



**Figure 13-2. Modulo Timer (MTIM) Block Diagram**

## 13.2 External Signal Description

The MTIM includes one external signal, TCLK, used to input an external clock when selected as the MTIM clock source. The signal properties of TCLK are shown in Table 13-1.

**Table 13-1. Signal Properties**

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

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

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

## 13.3 Register Definition

Figure 13-3 is a summary of MTIM registers.

**Figure 13-3.** MTIM **Register Summary**

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

Each MTIM includes four registers:

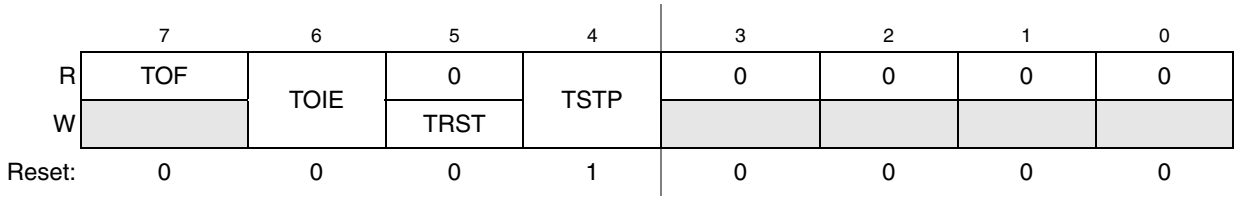
- An 8-bit status and control register
- An 8-bit clock configuration register
- An 8-bit counter register
- An 8-bit modulo register

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

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

### 13.3.1 MTIM Status and Control Register (MTIMSC)

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



| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | TOF | TOIE | 0 | TSTP | 0 | 0 | 0 | 0 |
| W | | | TRST | | | | | |
| Reset: | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Figure 13-4. MTIM Status and Control Register**

**Table 13-2. MTIM Status and Control Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOF | **MTIM Overflow Flag** — This read-only bit is set when the MTIM counter register overflows to $00 after reaching the value in the MTIM modulo register. Clear TOF by reading the MTIMSC register while TOF is set, then writing a 0 to TOF. TOF is also cleared when TRST is written to a 1 or when any value is written to the MTIMMOD register.<br>0 MTIM counter has not reached the overflow value in the MTIM modulo register.<br>1 MTIM counter has reached the overflow value in the MTIM modulo register. |
| 6<br>TOIE | **MTIM Overflow Interrupt Enable** — This read/write bit enables MTIM overflow interrupts. If TOIE is set, then an interrupt is generated when TOF = 1. Reset clears TOIE. Do not set TOIE if TOF = 1. Clear TOF first, then set TOIE.<br>0 TOF interrupts are disabled. Use software polling.<br>1 TOF interrupts are enabled. |
| 5<br>TRST | **MTIM Counter Reset** — When a 1 is written to this write-only bit, the MTIM counter register resets to $00 and TOF is cleared. Reading this bit always returns 0.<br>0 No effect. MTIM counter remains at current state.<br>1 MTIM counter is reset to $00. |
| 4<br>TSTP | **MTIM Counter Stop** — When set, this read/write bit stops the MTIM counter at its current value. Counting resumes from the current value when TSTP is cleared. Reset sets TSTP to prevent the MTIM from counting.<br>0 MTIM counter is active.<br>1 MTIM counter is stopped. |
| 3:0 | Unused register bits, always read 0. |

## 13.3.2 MTIM Clock Configuration Register (MTIMCLK)

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

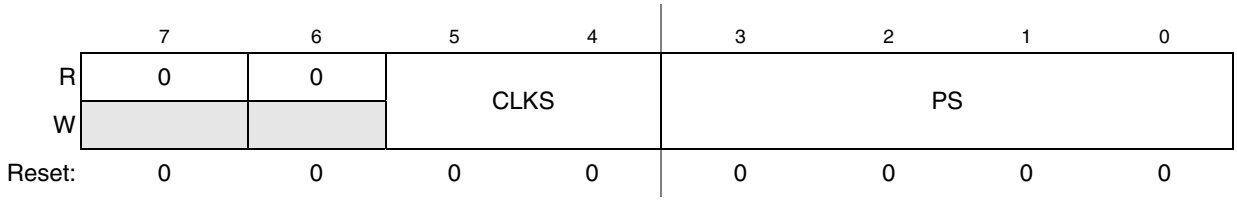



**Figure 13-5. MTIM Clock Configuration Register**

**Table 13-3. MTIM Clock Configuration Register Field Description**

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

### 13.3.3 MTIM Counter Register (MTIMCNT)

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

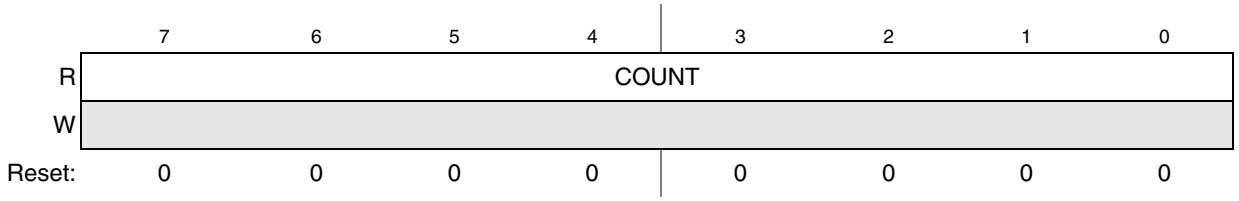



**Figure 13-6. MTIM Counter Register**

**Table 13-4. MTIM Counter Register Field Description**

| Field | Description |
|---|---|
| 7:0<br>COUNT | **MTIM Count** — These eight read-only bits contain the current value of the 8-bit counter. Writes have no effect to this register. Reset clears the count to $00. |

### 13.3.4 MTIM Modulo Register (MTIMMOD)

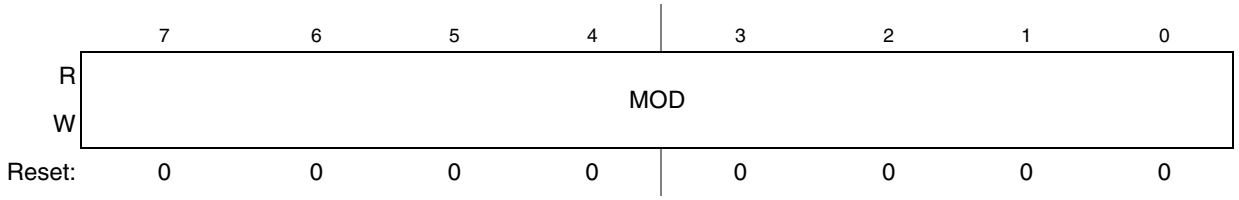



**Figure 13-7. MTIM Modulo Register**

**Table 13-5. MTIM Modulo Register Field Descriptions**

| Field | Description |
|---|---|
| 7:0<br>MOD | **MTIM Modulo** — These eight read/write bits contain the modulo value used to reset the count and set TOF. A value of $00 puts the MTIM in free-running mode. Writing to MTIMMOD resets the COUNT to $00 and clears TOF. Reset sets the modulo to $00. |

## 13.4 Functional Description

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

The MTIM counter (MTIMCNT) has three modes of operation: stopped, free-running, and modulo. Out of reset, the counter is stopped. If the counter is started without writing a new value to the modulo register, then the counter will be in free-running mode. The counter is in modulo mode when a value other than $00 is in the modulo register while the counter is running.

After any MCU reset, the counter is stopped and reset to $00, and the modulus is set to $00. The bus clock is selected as the default clock source and the prescale value is divide by 1. To start the MTIM in free-running mode, simply write to the MTIM status and control register (MTIMSC) and clear the MTIM stop bit (TSTP).

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

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

The MTIM modulo register (MTIMMOD) allows the overflow compare value to be set to any value from $01 to $FF. Reset clears the modulo value to $00, which results in a free running counter.

When the counter is active (TSTP = 0), the counter increments at the selected rate until the count matches the modulo value. When these values match, the counter overflows to $00 and continues counting. The MTIM overflow flag (TOF) is set whenever the counter overflows. The flag sets on the transition from the modulo value to $00. Writing to MTIMMOD while the counter is active resets the counter to $00 and clears TOF.

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

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

### 13.4.1 MTIM Operation Example

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

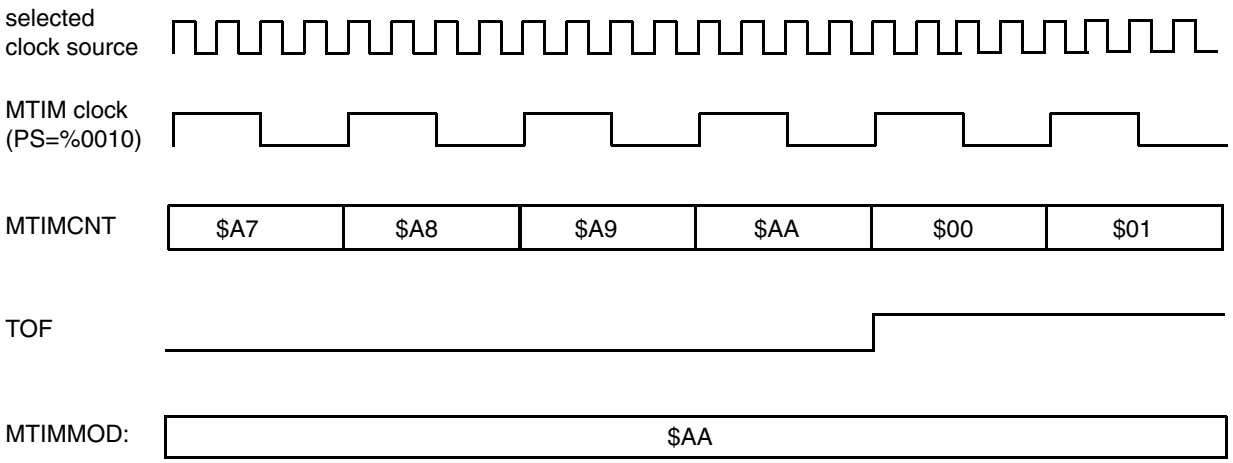


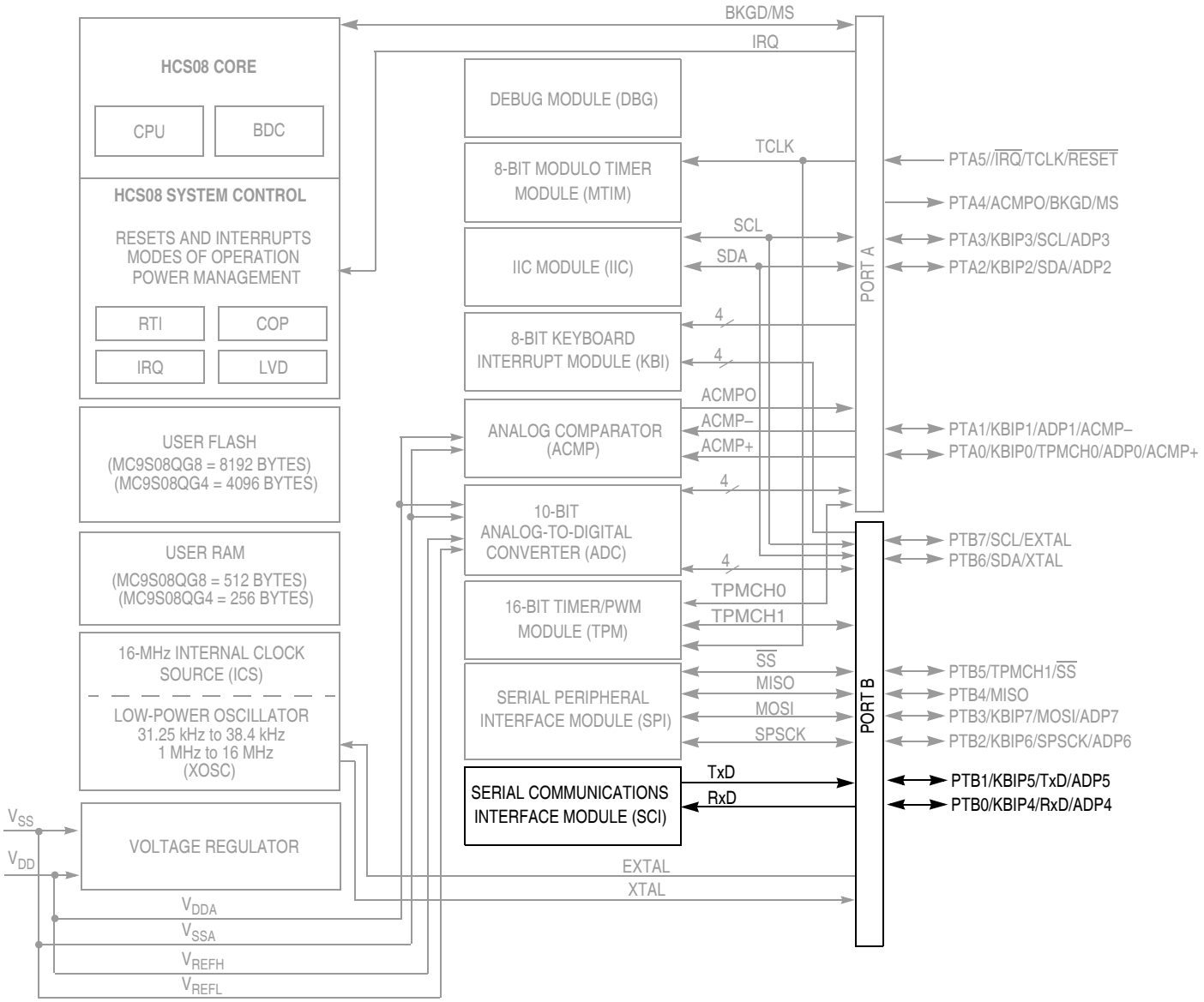

**Figure 13-8. MTIM counter overflow example**

In the example of Figure 13-8, the selected clock source could be any of the five possible choices. The prescaler is set to PS = %0010 or divide-by-4. The modulo value in the MTIMMOD register is set to $AA. When the counter, MTIMCNT, reaches the modulo value of $AA, the counter overflows to $00 and continues counting. The timer overflow flag, TOF, sets when the counter value changes from $AA to $00. An MTIM overflow interrupt is generated when TOF is set, if TOIE = 1.

# Chapter 14
# Serial Communications Interface (S08SCIV3)

## 14.1 Introduction

Figure 14-1 shows the MC9S08QG8/4 block diagram with the SCI highlighted.




NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. IRQ contains a software configurable (IRQPDD) pullup/pulldown device if PTA5 enabled as IRQ pin function (IRQPE = 1).
6. $\overline{\text{RESET}}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

**Figure 14-1. MC9S08QG8/4 Block Diagram Highlighting SCI Block and Pins**

**Module Initialization:**

| | | | |
|---|---|---|---|
| Write: | SCIBDH:SCIBDL | to set | baud rate |
| Write: | SCFC1 | to configure | 1-wire/2-wire, 9/8-bit data, wakeup, and parity, if used. |
| Write; | SCIC2 | to configure | interrupts, enable Rx and Tx, RWU<br>Enable Rx wakeup, SBK sends break character |
| Write: | SCIC3 | to enable | Rx error interrupt sources. Also controls pin direction in 1-wire modes. R8 and T8 only used in 9-bit data modes. |

**Module Use:**

Wait for TDRE, then write data to SCID

Wait for RDRF, then read data from SCID

A small number of applications will use RWU to manage automatic receiver wakeup, SBK to send break characters, and R8 and T8 for 9-bit data.

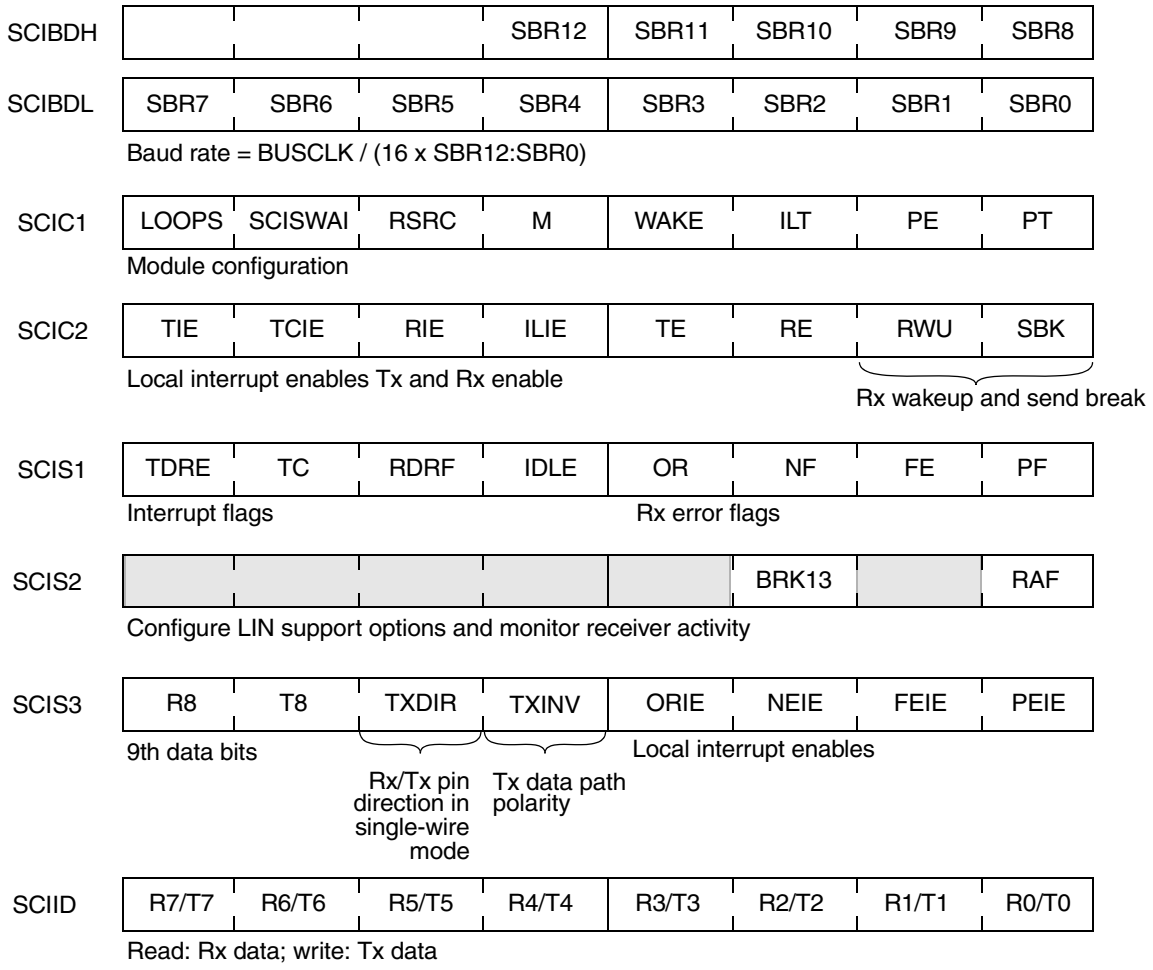



**Figure 14-2. SCI Module Quick Start**

### 14.1.1 Features

Features of SCI module include:

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

### 14.1.2 Modes of Operation

See Section 14.3, “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 mode
- Single-wire mode

### 14.1.3 Block Diagram

Figure 14-3 shows the transmitter portion of the SCI. (Figure 14-4 shows the receiver portion of the SCI.)

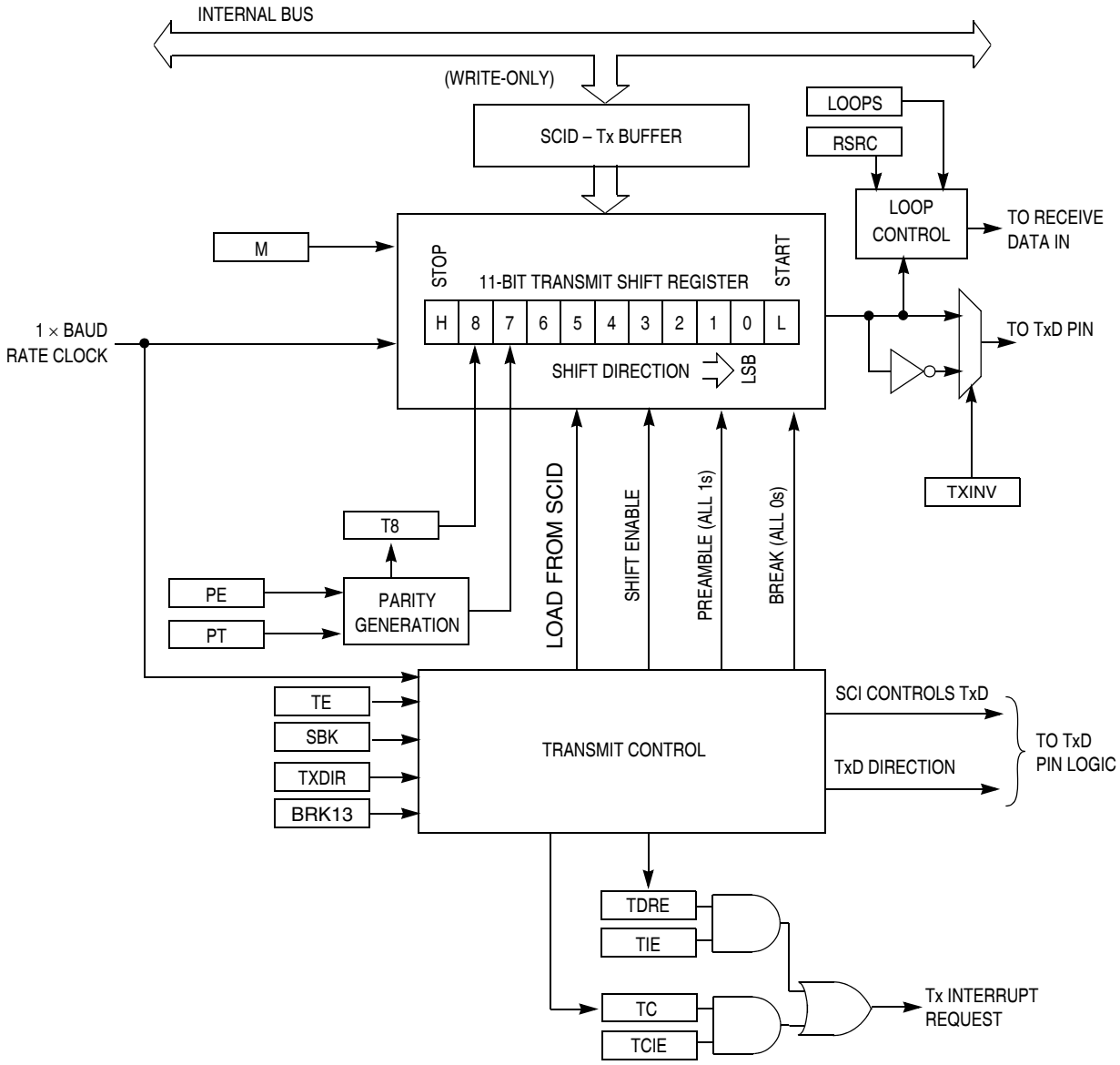



**Figure 14-3. SCI Transmitter Block Diagram**

Figure 14-4 shows the receiver portion of the SCI.

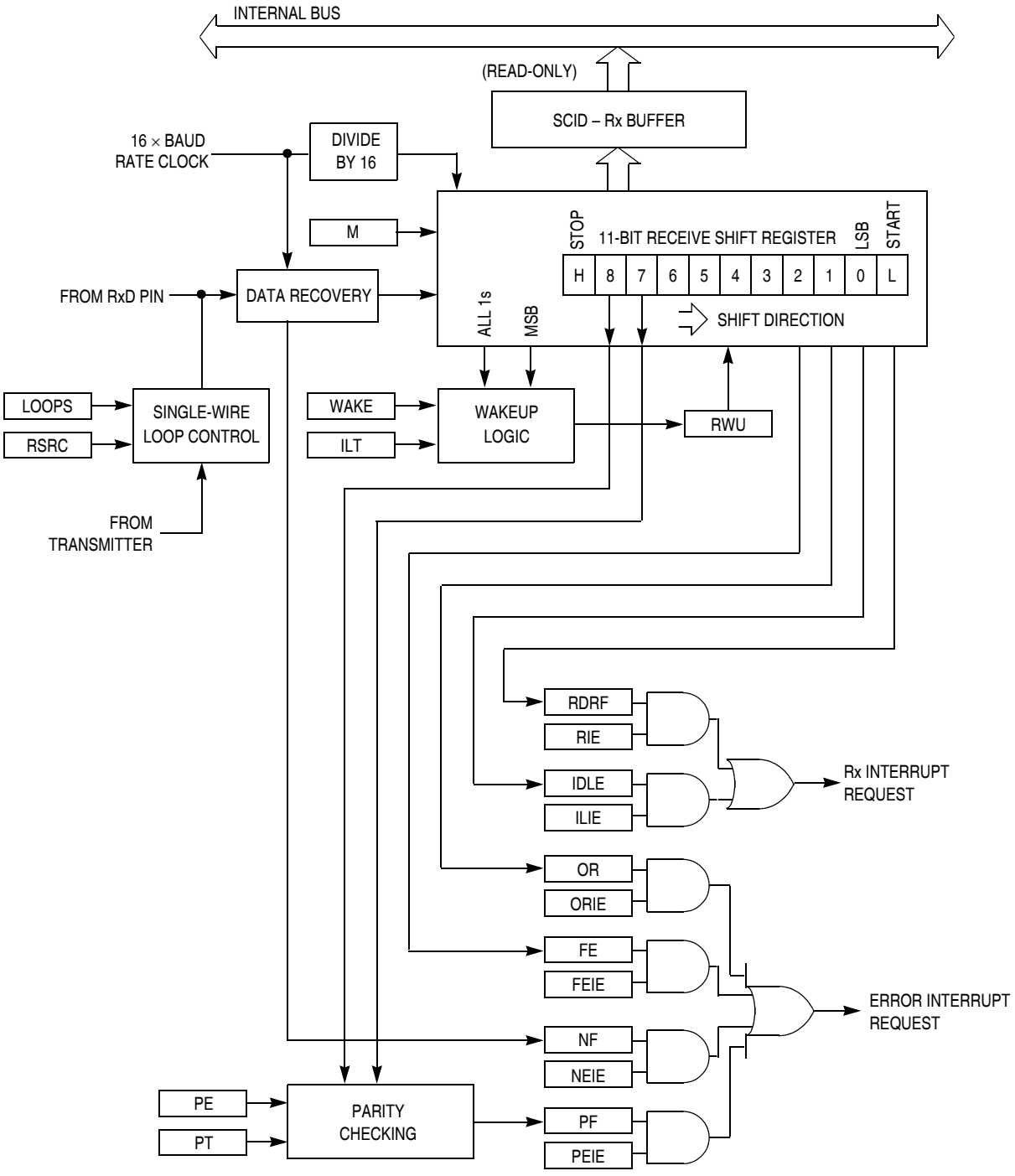



**Figure 14-4. SCI Receiver Block Diagram**

## 14.2 Register Definition

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

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

### 14.2.1 SCI Baud Rate Registers (SCIBDH, SCIBDL)

This pair of registers controls the prescale divisor for SCI baud rate generation. To update the 13-bit baud rate setting [SBR12:SBR0], first write to SCIBDH to buffer the high half of the new value and then write to SCIBDL. The working value in SCIBDH does not change until SCIBDL is written.

SCIBDL is reset to a non-zero value, so after reset the baud rate generator remains disabled until the first time the receiver or transmitter is enabled (RE or TE bits in SCIC2 are written to 1).

| | 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 14-5. SCI Baud Rate Register (SCIBDH)**

**Table 14-1. SCIBDH 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 14-2. |

| | 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 14-6. SCI Baud Rate Register (SCIBDL)**

**Table 14-2. SCIBDL 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 14-1. |

## 14.2.2 SCI Control Register 1 (SCIC1)

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 14-7. SCI Control Register 1 (SCIC1)**

**Table 14-3. SCIC1 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 14.3.3.2, “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 14.3.3.2.1, “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. |

**Table 14-3. SCIC1 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 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. |

## 14.2.3 SCI Control Register 2 (SCIC2)

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 14-8. SCI Control Register 2 (SCIC2)**

**Table 14-4. SCIC2 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 interrupts from TC disabled (use polling).<br>1 Hardware interrupt requested when TC flag is 1. |
| 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. When TE = 1, the SCI forces the TxD pin to act as an output for the SCI system.<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 14.3.2.1, “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. |

**Table 14-4. SCIC2 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>RE | **Receiver Enable** — When the SCI receiver is off, the RxD pin reverts to being a general-purpose port I/O pin. If LOOPS = 1, the RxD pin reverts to being a general-purpose I/O pin even if RE = 1.<br>0 Receiver off.<br>1 Receiver on. |
| 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 14.3.3.2, "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 14.3.2.1, "Send Break and Queued Idle," for more details.<br>0 Normal transmitter operation.<br>1 Queue break character(s) to be sent. |

## 14.2.4 SCI Status Register 1 (SCIS1)

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

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

= Unimplemented or Reserved

**Figure 14-9. SCI Status Register 1 (SCIS1)**

**Table 14-5. SCIS1 Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TDRE | **Transmit Data Register Empty Flag** — TDRE is set out of reset and when a transmit data value transfers from the transmit data buffer to the transmit shifter, leaving room for a new character in the buffer. To clear TDRE, read SCIS1 with TDRE = 1 and then write to the SCI data register (SCID).<br>0 Transmit data register (buffer) full.<br>1 Transmit data register (buffer) empty. |
| 6<br>TC | **Transmission Complete Flag** — TC is set out of 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 SCIS1 with TC = 1 and then doing one of the following three things:<br>• Write to the SCI data register (SCID) 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 SCIC2 |

**Table 14-5. SCIS1 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 (SCID). To clear RDRF, read SCIS1 with RDRF = 1 and then read the SCI data register (SCID).<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 SCIS1 with IDLE = 1 and then read the SCI data register (SCID). After IDLE has been cleared, it cannot become set again until after a new character has been received and RDRF has been set. IDLE will get set only once even if the receive line remains idle for an extended period.<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 SCID yet. In this case, the new character (and all associated error information) is lost because there is no room to move it into SCID. To clear OR, read SCIS1 with OR = 1 and then read the SCI data register (SCID).<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 SCIS1 and then read the SCI data register (SCID).<br>0 No noise detected.<br>1 Noise detected in the received character in SCID. |
| 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 SCIS1 with FE = 1 and then read the SCI data register (SCID).<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 SCIS1 and then read the SCI data register (SCID).<br>0 No parity error.<br>1 Parity error. |

## 14.2.5 SCI Status Register 2 (SCIS2)

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 | BRK13 | 0 | RAF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 14-10. SCI Status Register 2 (SCIS2)**

**Table 14-6. SCIS2 Register Field Descriptions**

| Field | Description |
|---|---|
| 2<br>BRK13 | **Break Character Length** — BRK13 is used to select a longer break character length. Detection of a framing error is not affected by the state of this bit.<br>0 Break character is 10 bit times (11 if M = 1)<br>1 Break character is 13 bit times (14 if M = 1) |
| 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). |

## 14.2.6 SCI Control Register 3 (SCIC3)

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

= Unimplemented or Reserved

**Figure 14-11. SCI Control Register 3 (SCIC3)**

**Table 14-7. SCIC3 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 SCID register. When reading 9-bit data, read R8 before reading SCID because reading SCID completes automatic flag clearing sequences which could allow R8 and SCID to be overwritten with new data. |
| 6<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 SCID register. When writing 9-bit data, the entire 9-bit value is transferred to the SCI shift register after SCID is written so T8 should be written (if it needs to change from its previous value) before SCID is written. If T8 does not need to change in the new value (such as when it is used to generate mark or space parity), it need not be written each time SCID is written. |

**Table 14-7. SCIC3 Register Field Descriptions (continued)**

| Field | Description |
|---|---|
| 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. |
| 4<br>TXINV[1] | **Transmit Data Inversion** — Setting this bit reverses the polarity of the transmitted data output.<br>0 Transmit data not inverted<br>1 Transmit data inverted |
| 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. |

[1] Setting TXINV inverts the TxD output for all cases: data bits, start and stop bits, break, and idle.

## 14.2.7 SCI Data Register (SCID)

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

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

**Figure 14-12. SCI Data Register (SCID)**

## 14.3 Functional Description

The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The SCI comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the SCI, writes the data to be transmitted, and processes received data. The following describes each of the blocks of the SCI.

### 14.3.1 Baud Rate Generation

As shown in Figure 14-13, the clock source for the SCI baud rate generator is the bus-rate clock.

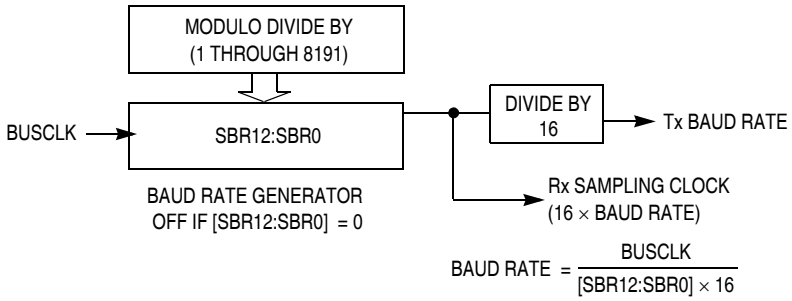



**Figure 14-13. SCI Baud Rate Generation**

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

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

### 14.3.2 Transmitter Functional Description

This section describes the overall block diagram for the SCI transmitter, as well as specialized functions for sending break and idle characters. The transmitter block diagram is shown in Figure 14-3.

The transmitter output (TxD) idle state defaults to logic high (TXINV = 0 following reset). The transmitter output is inverted by setting TXINV = 1. The transmitter is enabled by setting the TE bit in SCIC2. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit data buffer. Programs store data into the transmit data buffer by writing to the SCI data register (SCID).

The central element of the SCI transmitter is the transmit shift register that is either 10 or 11 bits long depending on the setting in the M control bit. For the remainder of this section, we will assume M = 0,

selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new SCI character, the value waiting in the transmit data register is transferred to the shift register (synchronized with the baud rate clock) and the transmit data register empty (TDRE) status flag is set to indicate another character may be written to the transmit data buffer at SCID.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out the 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.

#### 14.3.2.1 Send Break and Queued Idle

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

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake 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.

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

**Table 14-8. Break Character Length**

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

### 14.3.3 Receiver Functional Description

In this section, the receiver block diagram (Figure 14-4) is used as a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, two variations of the receiver wakeup function are explained.

The receiver is enabled by setting the RE bit in SCIC2. Character frames consist of a start bit of logic 0, eight (or nine) data bits (LSB first), and a stop bit of logic 1. For information about 9-bit data mode, refer to Section 14.4.1, "8- and 9-Bit Data Modes." For the remainder of this discussion, we assume the SCI is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (RDRF) status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCID. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to Section 14.3.4, "Interrupts and Status Flags," for more details about flag clearing.

#### 14.3.3.1 Data Sampling Technique

The SCI receiver uses a 16× baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the 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.

#### 14.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIC2. 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.

##### 14.3.3.2.1 Idle-Line Wakeup

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

When the RWU bit is set, the idle character that wakes a receiver does not set the receiver idle bit, IDLE, or the receive data register full flag, RDRF. It therefore will not generate an interrupt when this idle character occurs. The receiver will wake up and wait for the next data transmission which will set RDRF and generate an interrupt if enabled.

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

##### 14.3.3.2.2 Address-Mark Wakeup

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

Address-mark wakeup allows messages to contain idle characters but requires that the MSB be reserved for use in address frames. The logic 1 MSB of an address frame clears the receivers RWU bit before the stop bit is received and sets the RDRF flag.

### 14.3.4 Interrupts and Status Flags

The SCI system has three separate interrupt vectors to reduce the amount of software needed to isolate the cause of the interrupt. One interrupt vector is associated with the transmitter for TDRE and TC events. Another interrupt vector is associated with the receiver for RDRF 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 SCID. If the transmit interrupt enable (TIE) bit is set, a hardware interrupt will be requested whenever TDRE = 1. Transmit complete (TC) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with 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 SCID. The RDRF flag is cleared by reading SCIS1 while RDRF = 1 and then reading SCID.

When polling is used, this sequence is naturally satisfied in the normal course of the user program. If hardware interrupts are used, SCIS1 must be read in the interrupt service routine (ISR). Normally, this is done in the ISR anyway to check for receive errors, so the sequence is automatically satisfied.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RxD1 line remains idle for an extended period of time. IDLE is cleared by reading SCIS1 while IDLE = 1 and then reading SCID. After IDLE has been cleared, it cannot become set again until the receiver has received at least one new character and has set RDRF.

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

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

## 14.4 Additional SCI Functions

The following sections describe additional SCI functions.

### 14.4.1 8- and 9-Bit Data Modes

The SCI system (transmitter and receiver) can be configured to operate in 9-bit data mode by setting the M control bit in SCIC1. In 9-bit mode, there is a ninth data bit to the left of the MSB of the SCI data register. For the transmit data buffer, this bit is stored in T8 in SCIC3. For the receiver, the ninth bit is held in R8 in SCIC3.

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

If the bit value to be transmitted as the ninth bit of a new character is the same as for the previous character, it is not necessary to write to T8 again. When data is transferred from the transmit data buffer to the transmit shifter, the value in T8 is copied at the same time data is transferred from SCID to the shifter.

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

### 14.4.2 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.

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

### 14.4.3 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.

### 14.4.4 Single-Wire Operation

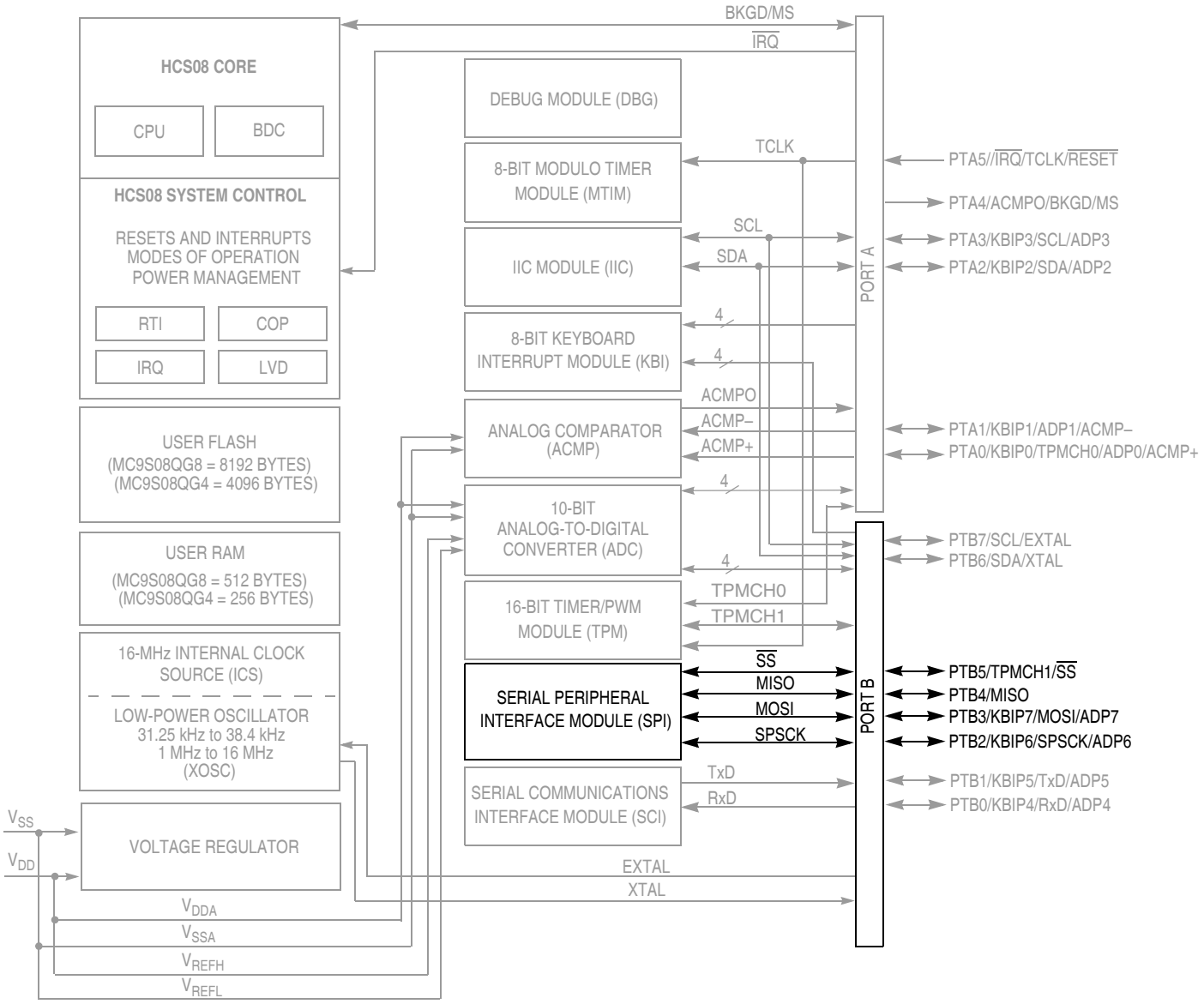
When LOOPS = 1, the RSRC bit in the same register chooses between loop mode (RSRC = 0) or single-wire mode (RSRC = 1). Single-wire mode is used to implement a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TxD1 pin. The RxD1 pin is not used and reverts to a general-purpose port I/O pin.

In single-wire mode, the TXDIR bit in SCIC3 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 15
# Serial Peripheral Interface (S08SPIV3)

## 15.1 Introduction

Figure 15-1 shows the MC9S08QG8/4 block diagram with the SPI highlighted.




NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

**Figure 15-1. MC9S08QG8/4 Block Diagram Highlighting SPI Block and Pins**

### 15.1.1 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

### 15.1.2 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.

#### 15.1.2.1 SPI System Block Diagram

Figure 15-2 shows the SPI modules of two MCUs connected in a master-slave arrangement. The master device initiates all SPI data transfers. During a transfer, the master shifts data out (on the MOSI pin) to the slave while simultaneously shifting data in (on the MISO pin) from the slave. The transfer effectively exchanges the data that was in the SPI shift registers of the two SPI systems. The SPSCK signal is a clock output from the master and an input to the slave. The slave device must be selected by a low level on the slave select input ($\overline{SS}$ pin). In this system, the master device has configured its $\overline{SS}$ pin as an optional slave select output.

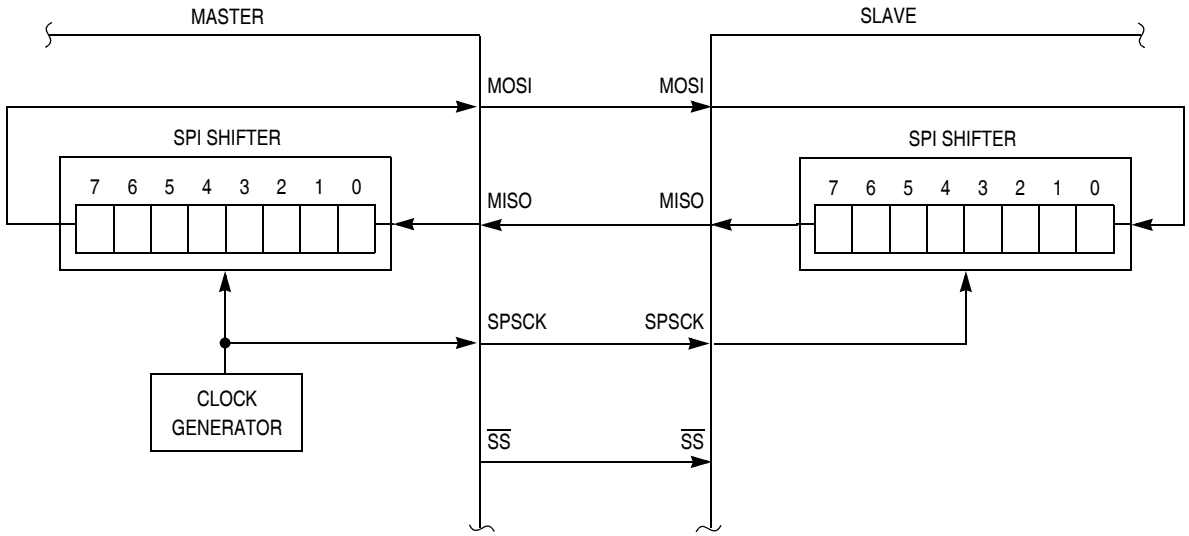



**Figure 15-2. SPI System Connections**

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 15-2 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.

#### 15.1.2.2 SPI Module Block Diagram

Figure 15-3 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 SPID) 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 SPID). Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

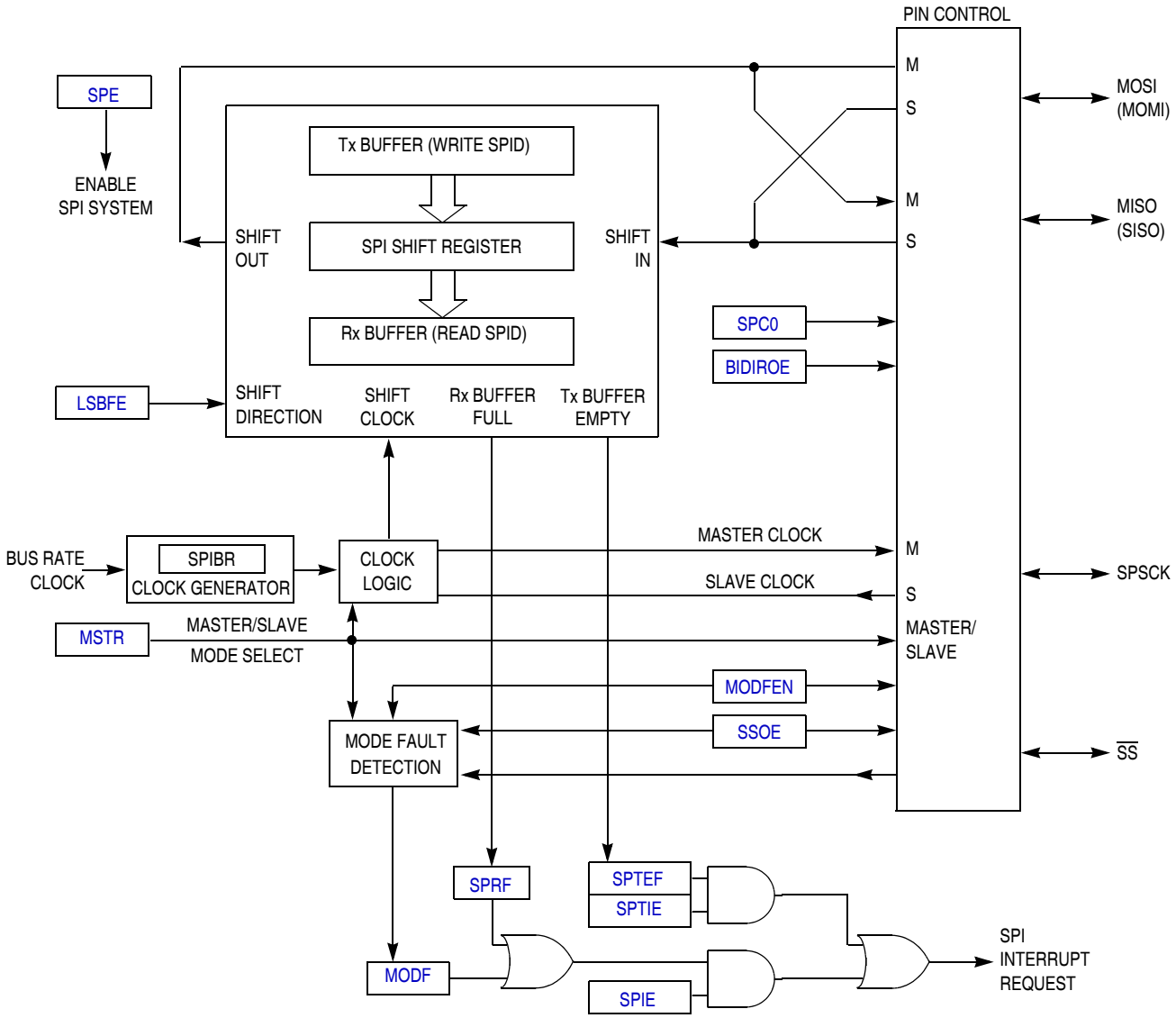



**Figure 15-3. SPI Module Block Diagram**

## 15.1.3 SPI Baud Rate Generation

As shown in Figure 15-4, 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.

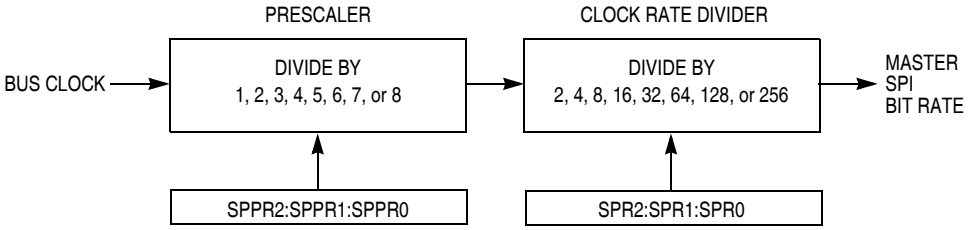



**Figure 15-4. SPI Baud Rate Generation**

## 15.2 External Signal Description

The SPI optionally shares four port pins. The function of these pins depends on the settings of SPI control bits. When the SPI is disabled (SPE = 0), these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI.

### 15.2.1 SPSCK — SPI Serial Clock

When the SPI is enabled as a slave, this pin is the serial clock input. When the SPI is enabled as a master, this pin is the serial clock output.

### 15.2.2 MOSI — Master Data Out, Slave Data In

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data output. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data input. If SPC0 = 1 to select single-wire bidirectional mode, and master mode is selected, this pin becomes the bidirectional data I/O pin (MOMI). Also, the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and slave mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 15.2.3 MISO — Master Data In, Slave Data Out

When the SPI is enabled as a master and SPI pin control zero (SPC0) is 0 (not bidirectional mode), this pin is the serial data input. When the SPI is enabled as a slave and SPC0 = 0, this pin is the serial data output. If SPC0 = 1 to select single-wire bidirectional mode, and slave mode is selected, this pin becomes the bidirectional data I/O pin (SISO) and the bidirectional mode output enable bit determines whether the pin acts as an input (BIDIROE = 0) or an output (BIDIROE = 1). If SPC0 = 1 and master mode is selected, this pin is not used by the SPI and reverts to being a general-purpose port I/O pin.

### 15.2.4 $\overline{SS}$ — Slave Select

When the SPI is enabled as a slave, this pin is the low-true slave select input. When the SPI is enabled as a master and mode fault enable is off (MODFEN = 0), this pin is not used by the SPI and reverts to being a general-purpose port I/O pin. When the SPI is enabled as a master and MODFEN = 1, the slave select output enable bit determines whether this pin acts as the mode fault input (SSOE = 0) or as the slave select output (SSOE = 1).

## 15.3 Modes of Operation

### 15.3.1 SPI in Stop Modes

The SPI is disabled in all stop modes, regardless of the settings before executing the STOP instruction. During either stop1 or stop2 mode, the SPI module will be fully powered down. Upon wake-up from stop1 or stop2 mode, the SPI module will be in the reset state. During stop3 mode, clocks to the SPI module are halted. No registers are affected. If stop3 is exited with a reset, the SPI will be put into its reset state. If stop3 is exited with an interrupt, the SPI continues from the state it was in when stop3 was entered.

## 15.4 Register Definition

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.

### 15.4.1 SPI Control Register 1 (SPIC1)

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | SPIE | SPE | SPTIE | MSTR | CPOL | CPHA | SSOE | LSBFE |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Figure 15-5. SPI Control Register 1 (SPIC1)**

**Table 15-1. SPIC1 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>SPIE | **SPI Interrupt Enable (for SPRF and MODF)** — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events.<br>0 Interrupts from SPRF and MODF inhibited (use polling)<br>1 When SPRF or MODF is 1, request a hardware interrupt |
| 6<br>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.<br>0 SPI system inactive<br>1 SPI system enabled |
| 5<br>SPTIE | **SPI Transmit Interrupt Enable** — This is the interrupt enable bit for SPI transmit buffer empty (SPTEF).<br>0 Interrupts from SPTEF inhibited (use polling)<br>1 When SPTEF is 1, hardware interrupt requested |

**Table 15-1. SPIC1 Field Descriptions (continued)**

| Field | Description |
|---|---|
| 4<br>MSTR | **Master/Slave Mode Select**<br>0 SPI module configured as a slave SPI device<br>1 SPI module configured as a master SPI device |
| 3<br>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 15.5.1, "SPI Clock Formats" for more details.<br>0 Active-high SPI clock (idles low)<br>1 Active-low SPI clock (idles high) |
| 2<br>CPHA | **Clock Phase** — This bit selects one of two clock formats for different kinds of synchronous serial peripheral devices. Refer to Section 15.5.1, "SPI Clock Formats" for more details.<br>0 First edge on SPSCK occurs at the middle of the first cycle of an 8-cycle data transfer<br>1 First edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer |
| 1<br>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 $\overline{SS}$ pin as shown in Table 15-2. |
| 0<br>LSBFE | **LSB First (Shifter Direction)**<br>0 SPI serial data transfers start with most significant bit<br>1 SPI serial data transfers start with least significant bit |

**Table 15-2. $\overline{SS}$ Pin Function**

| MODFEN | SSOE | Master Mode | Slave Mode |
|---|---|---|---|
| 0 | 0 | General-purpose I/O (not SPI) | Slave select input |
| 0 | 1 | General-purpose I/O (not SPI) | Slave select input |
| 1 | 0 | $\overline{SS}$ input for mode fault | Slave select input |
| 1 | 1 | Automatic $\overline{SS}$ output | Slave select input |

**NOTE**

Ensure that the SPI should not be disabled (SPE=0) at the same time as a bit change to the CPHA bit. These changes should be performed as separate operations or unexpected behavior may occur.

## 15.4.2 SPI Control Register 2 (SPIC2)

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.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | MODFEN | BIDIROE | 0 | SPISWAI | SPC0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 15-6. SPI Control Register 2 (SPIC2)**

**Table 15-3. SPIC2 Register Field Descriptions**

| Field | Description |
|---|---|
| 4<br>MODFEN | **Master Mode-Fault Function Enable** — When the SPI is configured for slave mode, this bit has no meaning or effect. (The $\overline{SS}$ pin is the slave select input.) In master mode, this bit determines how the $\overline{SS}$ pin is used (refer to Table 15-2 for more details).<br>0 Mode fault function disabled, master $\overline{SS}$ pin reverts to general-purpose I/O not controlled by SPI<br>1 Mode fault function enabled, master $\overline{SS}$ pin acts as the mode fault input or the slave select output |
| 3<br>BIDIROE | **Bidirectional Mode Output Enable** — When bidirectional mode is enabled by SPI pin control 0 (SPC0) = 1, BIDIROE determines whether the SPI data output driver is enabled to the single bidirectional SPI I/O pin. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (MOMI) or MISO (SISO) pin, respectively, as the single SPI data I/O pin. When SPC0 = 0, BIDIROE has no meaning or effect.<br>0 Output driver disabled so SPI data I/O pin acts as an input<br>1 SPI I/O pin enabled as an output |
| 1<br>SPISWAI | **SPI Stop in Wait Mode**<br>0 SPI clocks continue to operate in wait mode<br>1 SPI clocks stop when the MCU enters wait mode |
| 0<br>SPC0 | **SPI Pin Control 0** — The SPC0 bit chooses single-wire bidirectional mode. If MSTR = 0 (slave mode), the SPI uses the MISO (SISO) pin for bidirectional SPI data transfers. If MSTR = 1 (master mode), the SPI uses the MOSI (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.<br>0 SPI uses separate pins for data input and data output<br>1 SPI configured for single-wire bidirectional operation |

## 15.4.3 SPI Baud Rate Register (SPIBR)

This register is used to set the prescaler and bit rate divisor for an SPI master. This register may be read or written at any time.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | SPPR2 | SPPR1 | SPPR0 | 0 | SPR2 | SPR1 | SPR0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 15-7. SPI Baud Rate Register (SPIBR)**

**Table 15-4. SPIBR Register Field Descriptions**

| Field | Description |
|---|---|
| 6:4<br>SPPR[2:0] | **SPI Baud Rate Prescale Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate prescaler as shown in Table 15-5. The input to this prescaler is the bus rate clock (BUSCLK). The output of this prescaler drives the input of the SPI baud rate divider (see Figure 15-4). |
| 2:0<br>SPR[2:0] | **SPI Baud Rate Divisor** — This 3-bit field selects one of eight divisors for the SPI baud rate divider as shown in Table 15-6. The input to this divider comes from the SPI baud rate prescaler (see Figure 15-4). The output of this divider is the SPI bit rate clock for master mode. |

**Table 15-5. SPI Baud Rate Prescaler Divisor**

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

**Table 15-6. SPI Baud Rate Divisor**

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

## 15.4.4 SPI Status Register (SPIS)

This register has three read-only status bits. Bits 6, 3, 2, 1, and 0 are not implemented and always read 0. Writes have no meaning or effect.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | SPRF | 0 | SPTEF | MODF | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 15-8. SPI Status Register (SPIS)**

**Table 15-7. SPIS Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>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 (SPID). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register.<br>0 No data available in the receive data buffer<br>1 Data available in the receive data buffer |
| 5<br>SPTEF | **SPI Transmit Buffer Empty Flag** — This bit is set when there is room in the transmit data buffer. It is cleared by reading SPIS with SPTEF set, followed by writing a data value to the transmit buffer at SPID. SPIS must be read with SPTEF = 1 before writing data to SPID or the SPID write will be ignored. SPTEF generates an SPTEF CPU interrupt request if the SPTIE bit in the SPIC1 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 SPID 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.<br>0 SPI transmit buffer not empty<br>1 SPI transmit buffer empty |
| 4<br>MODF | **Master Mode Fault Flag** — MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The $\overline{SS}$ pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPIC1).<br>0 No mode fault error<br>1 Mode fault error detected |

## 15.4.5 SPI Data Register (SPID)

| | 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 15-9. SPI Data Register (SPID)**

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 SPID 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.

## 15.5 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 (SPID) 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 MISO pin at one SPSCK edge and shifted, changing the bit value on the MOSI 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 MOSI pin to the slave while eight bits of data were shifted in the MISO 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 SPID. 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 $\overline{SS}$ pin must be driven low before a transfer starts and $\overline{SS}$ must stay low throughout the transfer. If a clock format where CPHA = 0 is selected, $\overline{SS}$ must be driven to a logic 1 between successive transfers. If CPHA = 1, $\overline{SS}$ may remain low between successive transfers. See Section 15.5.1, "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 SPID) 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.

### 15.5.1 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 15-10 shows the clock formats when CPHA = 1. At the top of the figure, the eight bit times are shown for reference with bit 1 starting at the first SPSCK edge and bit 8 ending one-half SPSCK cycle after the sixteenth SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the

MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

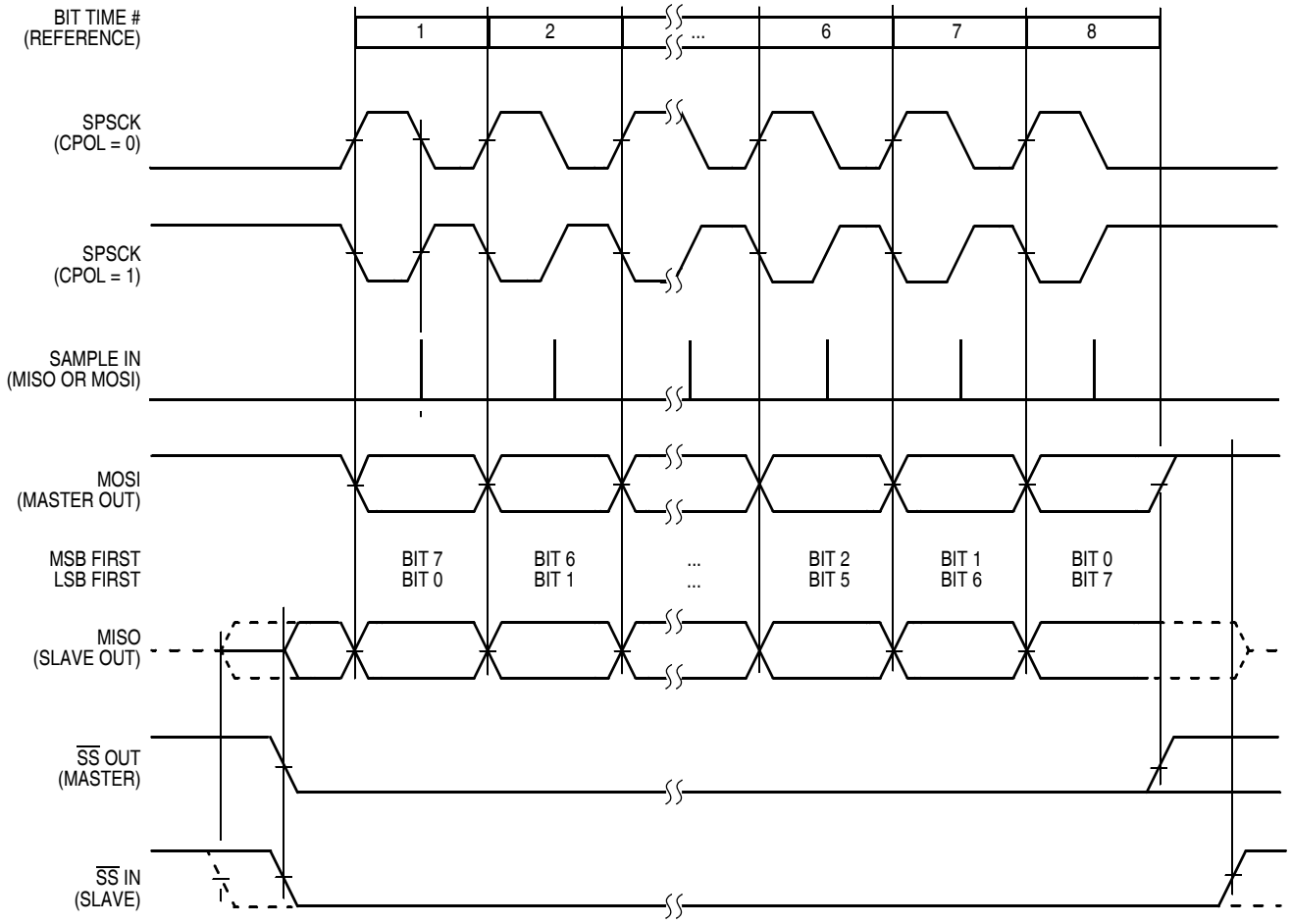



**Figure 15-10. SPI Clock Formats (CPHA = 1)**

When CPHA = 1, the slave begins to drive its MISO output when $\overline{SS}$ goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CHPA = 1, the slave's $\overline{SS}$ input is not required to go to its inactive high level between transfers.

Figure 15-11 shows the clock formats when CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ($\overline{SS}$ IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting

in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL. The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low at the start of the first bit time of the transfer and goes back high one-half SPSCK cycle after the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

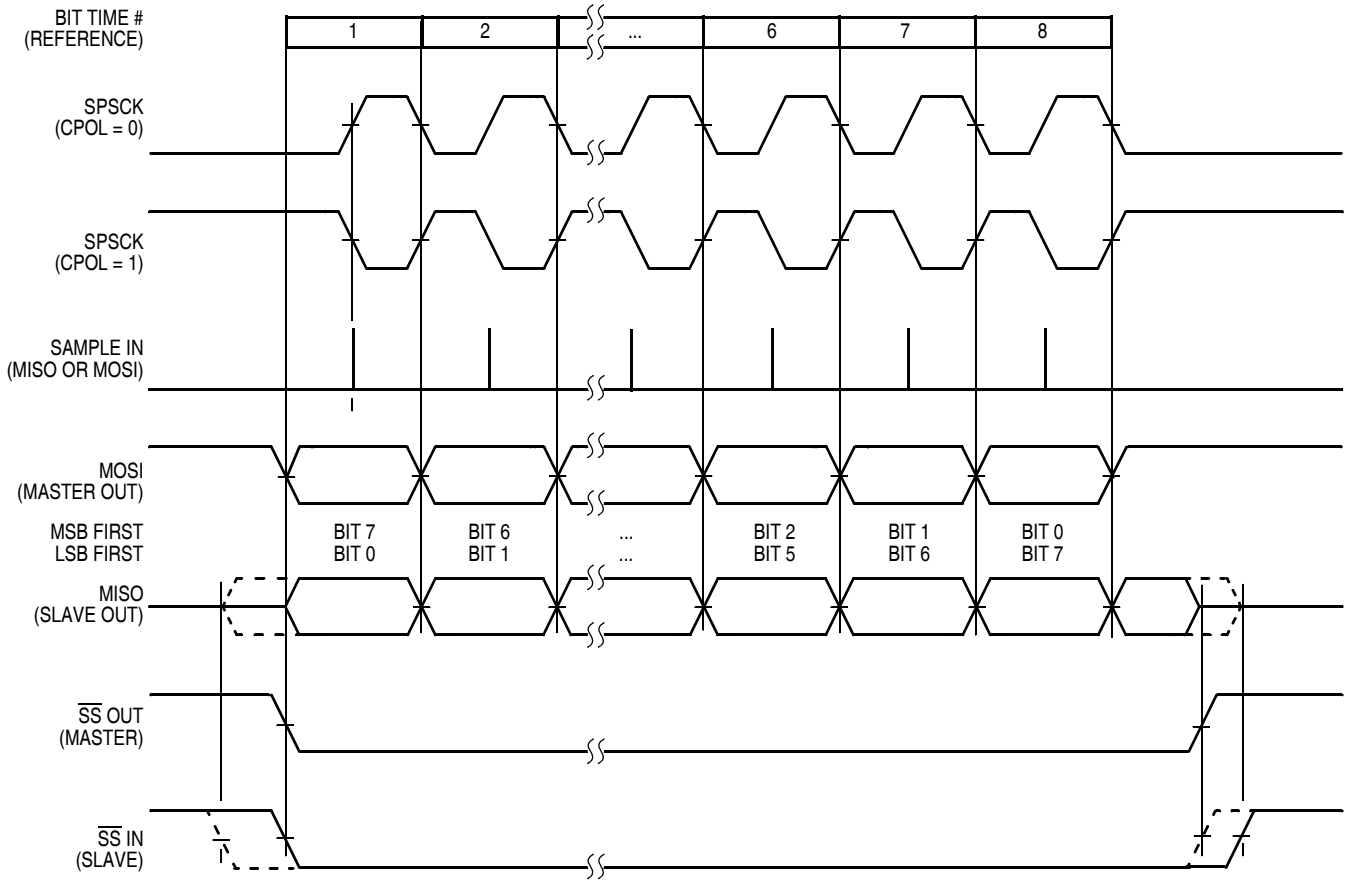



**Figure 15-11. SPI Clock Formats (CPHA = 0)**

When CPHA = 0, the slave begins to drive its MISO output with the first data bit value (MSB or LSB depending on LSBFE) when $\overline{SS}$ goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the second SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CPHA = 0, the slave's $\overline{SS}$ input must go to its inactive high level between transfers.

### 15.5.2 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).

### 15.5.3 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 $\overline{SS}$ pin (provided the $\overline{SS}$ pin is configured as the mode fault input signal). The $\overline{SS}$ 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 $\overline{SS}$ 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 SPSCK, MOSI, and MISO (if not bidirectional mode) are disabled.

MODF is cleared by reading it while it is set, then writing to the SPI control register 1 (SPIC1). User software should verify the error condition has been corrected before changing the SPI back to master mode.

# Chapter 16
# Timer/Pulse-Width Modulator (S08TPMV2)

## 16.1 Introduction

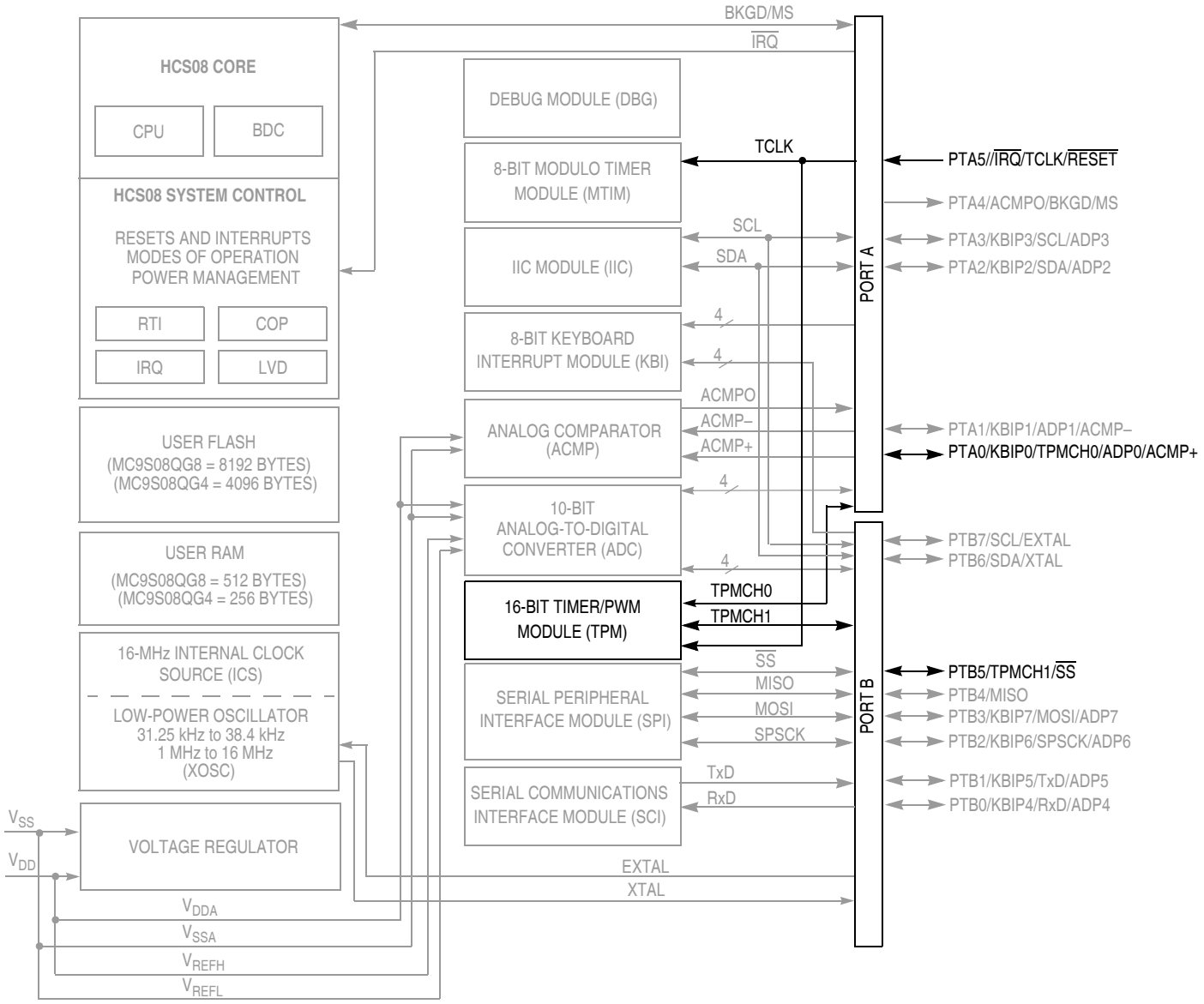
Figure 16-1 shows the MC9S08QG8/4 block diagram with the TPM highlighted.

### 16.1.1 ACMP/TPM Configuration Information

The ACMP module can be configured to connect the output of the analog comparator to TPM input capture channel 0 by setting ACIC in SOPT2. With ACIC set, the TPMCH0 pin is not available externally regardless of the configuration of the TPM module.

### 16.1.2 MTIM/TPM Configuration Information

The external clock for the TPM module, TPMCLK, is selected by setting CLKS[B:A] = 1:1 in TPMSC, which selects the TCLK pin input. The TCLK input on PTA5 can be enabled as external clock inputs to both the MTIM and TPM modules simultaneously.




NOTES:

1. Not all pins or pin functions are available on all devices, see Table 1-1 for available functions on each device.
2. Port pins are software configurable with pullup device if input port.
3. Port pins are software configurable for output drive strength.
4. Port pins are software configurable for output slew rate control.
5. $\overline{IRQ}$ contains a software configurable (IRQPDD) pullup device if PTA5 enabled as $\overline{IRQ}$ pin function (IRQPE = 1).
6. $\overline{RESET}$ contains integrated pullup device if PTA5 enabled as reset pin function (RSTPE = 1).
7. PTA4 contains integrated pullup device if BKGD enabled (BKGDPE = 1).
8. SDA and SCL pin locations can be repositioned under software control (IICPS), defaults on PTA2 and PTA3.
9. When pin functions as KBI (KBIPEn = 1) and associated pin is configured to enable the pullup device, KBEDGn can be used to reconfigure the pullup as a pulldown device.

**Figure 16-1. MC9S08QG8/4 Block Diagram Highlighting TPM Block and Pins**

### 16.1.3 Features

The TPM has the following features:

- Each TPM may be configured for buffered, center-aligned pulse-width modulation (CPWM) on all channels
- Clock sources independently selectable per TPM (multiple TPMs device)
- Selectable clock sources (device dependent): bus clock, fixed system clock, external pin
- Clock prescaler taps for divide by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit free-running or up/down (CPWM) count operation
- 16-bit modulus register to control counter range
- Timer system enable
- One interrupt per channel plus a terminal count interrupt for each TPM module (multiple TPMs device)
- Channel features:
  - 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

### 16.1.4 Block Diagram

Figure 16-2 shows the structure of a TPM. Some MCUs include more than one TPM, with various numbers of channels.

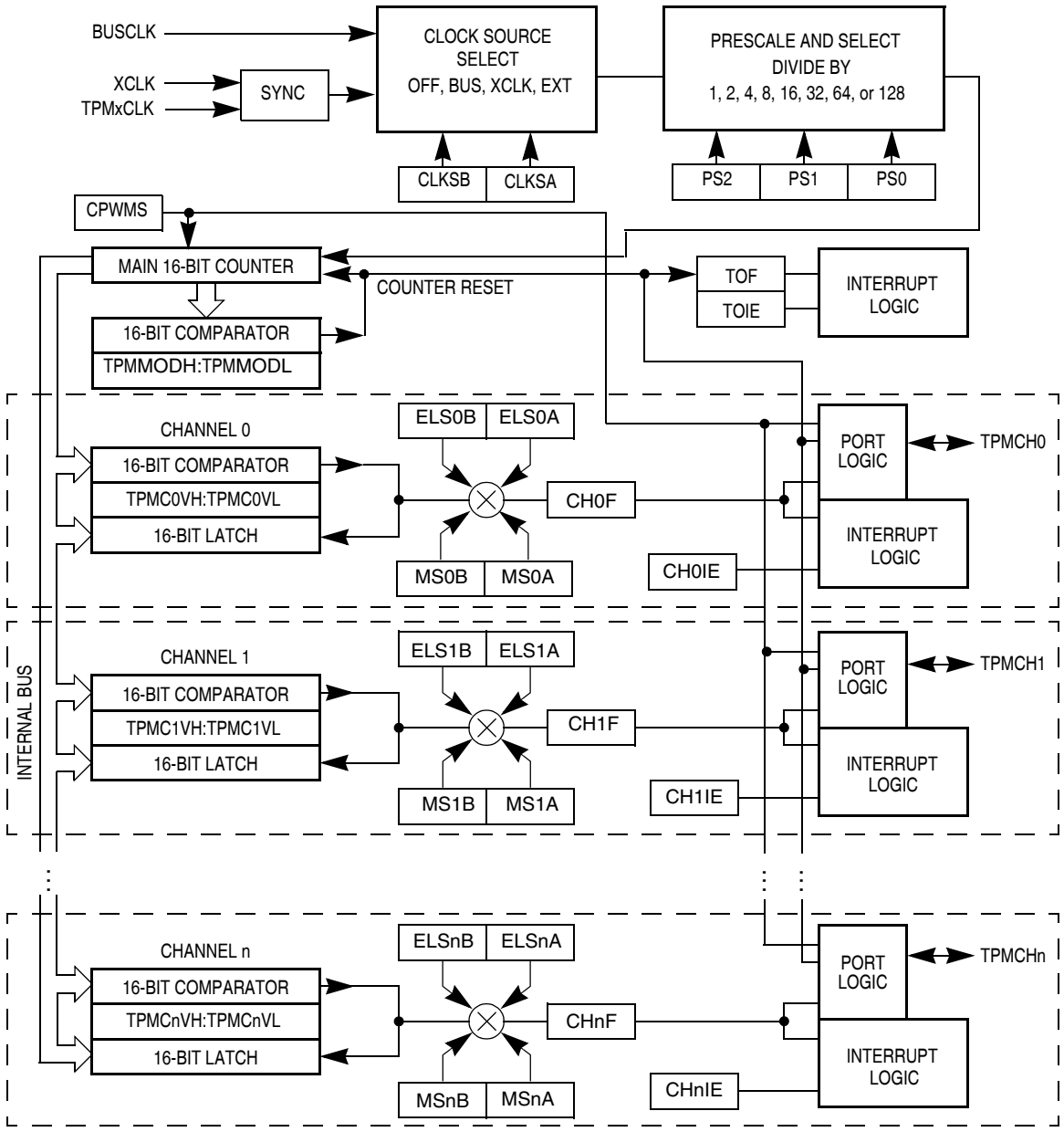



**Figure 16-2. TPM Block Diagram**

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, TPMMODH:TPMMODL, control the modulo value of the counter. (The values 0x0000 or 0xFFFF effectively make the counter free running.) Software can read the counter value at any time without affecting the counting sequence. Any write to either byte of the TPMCNT 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.

## 16.2 External Signal Description

When any pin 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.

### 16.2.1 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 TPM are driven by an external clock source, TPMxCLK, connected to an I/O 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.

On some devices the external clock input is shared with one of the TPM channels. When a TPM channel is shared as the external clock input, the associated TPM channel cannot use the pin. (The channel can still be used in output compare mode as a software timer.) Also, if one of the TPM channels is used as the external clock input, the corresponding ELSnB:ELSnA control bits must be set to 0:0 so the channel is not trying to use the same pin.

### 16.2.2 TPMCHn — TPM 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.

## 16.3 Register Definition

The TPM includes:

- An 8-bit status and control register (TPMSC)
- A 16-bit counter (TPMCNTH:TPMCNTL)
- A 16-bit modulo register (TPMMODH:TPMMODL)

Each timer channel has:

- An 8-bit status and control register (TPMCnSC)
- A 16-bit channel value register (TPMCnVH:TPMCnVL)

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.

### 16.3.1 Timer Status and Control Register (TPMSC)

TPMSC 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.

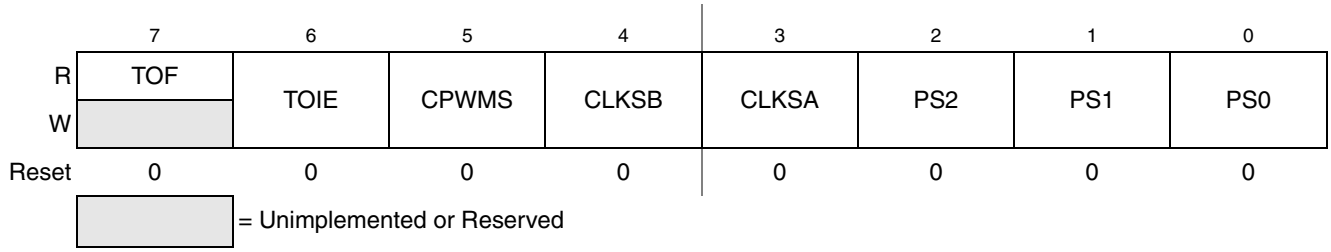



**Figure 16-3. Timer Status and Control Register (TPMSC)**

**Table 16-1. TPMSC Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>TOF | **Timer Overflow Flag** — This flag is set when the TPM counter changes to 0x0000 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 TPM 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 TPM channels operate in center-aligned PWM mode |
| 4:3<br>CLKS[B:A] | **Clock Source Select** — As shown in Table 16-2, 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 16-3. 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 16-2. 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 (TPMCLK)[1,2] |

1 The maximum frequency that is allowed as an external clock is one-fourth of the bus frequency.

2 If the external clock input is shared with channel n and is selected as the TPM clock source, the corresponding ELSnB:ELSnA control bits should be set to 0:0 so channel n does not try to use the same pin for a conflicting function.

**Table 16-3. 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 |

## 16.3.2 Timer Counter Registers (TPMCNTH:TPMCNTL)

The two read-only TPM counter registers contain the high and low bytes of the value in the TPM counter. Reading either byte (TPMCNTH or TPMCNTL) latches the contents of both bytes into a buffer where they remain latched until the other 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 TPMCNTH or TPMCNTL, or any write to the timer status/control register (TPMSC).

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 TPMCNTH clears the 16-bit counter. | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16-4. Timer Counter Register High (TPMCNTH)**

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

**Figure 16-5. Timer Counter Register Low (TPMCNTL)**

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.

## 16.3.3 Timer Counter Modulo Registers (TPMMODH:TPMMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock (CPWMS = 0) or starts counting down (CPWMS = 1), and the overflow flag (TOF) becomes set. Writing to TPMMODH or TPMMODL inhibits TOF and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000, which results in a free-running timer counter (modulo disabled).

| | 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 16-6. Timer Counter Modulo Register High (TPMMODH)**

| | 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 16-7. Timer Counter Modulo Register Low (TPMMODL)**

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.

### 16.3.4 Timer Channel n Status and Control Register (TPMCnSC)

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

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CHnF | CHnIE | MSnB | MSnA | ELSnB | ELSnA | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 16-8. Timer Channel n Status and Control Register (TPMCnSC)**

**Table 16-4. TPMCnSC 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 TPMCnSC 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 16-5. |
| 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 16-5 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 16-5, 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. |

**Table 16-5. 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.

## 16.3.5 Timer Channel Value Registers (TPMCnVH:TPMCnVL)

These read/write registers contain the captured TPM counter value of the input capture function or the output compare value for the output compare or PWM functions. The channel 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 16-9. Timer Channel Value Register High (TPMCnVH)**

| | 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 16-10. Timer Channel Value Register Low (TPMCnVL)**

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

In output compare or PWM modes, writing to either byte (TPMCnVH or TPMCnVL) 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 TPMCnSC register.

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

## 16.4 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 TPMSC. When CPWMS is set to 1, timer counter TPMCNT 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.

### 16.4.1 Counter

All timer functions are based on the main 16-bit counter (TPMCNTH:TPMCNTL). 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. The maximum frequency allowed for the external clock option is one-fourth the bus rate. Refer to Section 16.3.1, “Timer Status and Control Register (TPMSC)” and Table 16-2 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 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMMODH:TPMMODL.

When center-aligned PWM operation is specified, the counter counts upward from 0x0000 through its terminal count and then counts downward to 0x0000 where it returns to up-counting. Both 0x0000 and the terminal count value (value in TPMMODH:TPMMODL) 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 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the 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 0x0000 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 (TPMCNTH or TPMCNTL), 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 TPMCNTH or TPMCNTL. 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.

## 16.4.2 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.

### 16.4.2.1 Input Capture Mode

With the input capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TPM latches the contents of the TPM counter into the channel value registers (TPMCnVH:TPMCnVL). 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 (TPMCnSC).

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

### 16.4.2.2 Output Compare Mode

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

In output compare mode, values are transferred to the corresponding timer channel 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 (TPMCnSC).

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

### 16.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS = 0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the setting in the modulus register (TPMMODH:TPMMODL). The duty cycle is determined by the setting in the timer channel value register (TPMCnVH:TPMCnVL). 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 16-11 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.

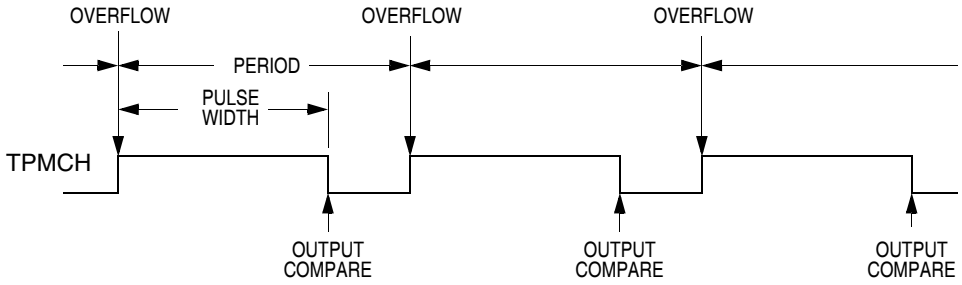



**Figure 16-11. PWM Period and Pulse Width (ELSnA = 0)**

When the channel value register is set to 0x0000, the duty cycle is 0 percent. By setting the timer channel value register (TPMCnVH:TPMCnVL) to a value greater than the modulus setting, 100% duty cycle can be achieved. This implies that the modulus setting must be less than 0xFFFF to get 100% 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, TPMCnVH or TPMCnVL, 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 TPMCNTH:TPMCNTL counter is 0x0000. (The new duty cycle does not take effect until the next full period.)

### 16.4.3 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 TPMCnVH:TPMCnVL determines the pulse width (duty cycle) of the PWM signal and the period is determined by the value in TPMMODH:TPMMODL. TPMMODH:TPMMODL should be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. ELSnA will determine the polarity of the CPWM output.

$$\text{pulse width} = 2 \times (\text{TPMCnVH:TPMCnVL}) \qquad \textit{Eqn. 16-1}$$

$$\text{period} = 2 \times (\text{TPMMODH:TPMMODL}); \quad \text{for TPMMODH:TPMMODL} = 0\text{x}0001\text{–}0\text{x}7\text{FFF} \qquad \textit{Eqn. 16-2}$$

If the channel value register TPMCnVH:TPMCnVL is zero or negative (bit 15 set), the duty cycle will be 0%. If TPMCnVH:TPMCnVL is a positive value (bit 15 clear) and is greater than the (nonzero) modulus setting, the duty cycle will be 100% because the duty cycle compare will never occur. This implies the usable range of periods set by the modulus register is 0x0001 through 0x7FFE (0x7FFF if generation of 100% duty cycle is not necessary). This is not a significant limitation because the resulting period is much longer than required for normal applications.

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

Figure 16-12 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 TPMMODH:TPMMODL, then counts down until it reaches zero. This sets the period equal to two times TPMMODH:TPMMODL.

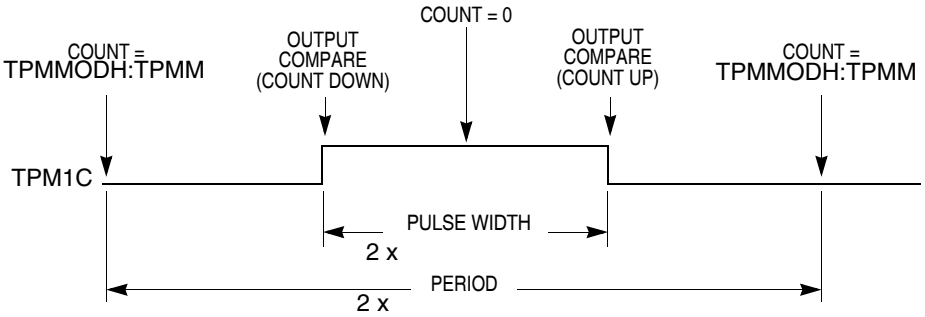



**Figure 16-12. CPWM Period and Pulse Width (ELSnA = 0)**

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, TPMMODH, TPMMODL, TPMCnVH, and TPMCnVL, 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 TPMCNT overflow requirement only applies to PWM channels, not output compares.

Optionally, when TPMCNTH:TPMCNTL = TPMMODH:TPMMODL, 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 TPMSC cancels any values written to TPMMODH and/or TPMMODL and resets the coherency mechanism for the modulo registers. Writing to TPMCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMCnVH:TPMCnVL.

## 16.5 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.

### 16.5.1 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.

### 16.5.2 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 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the 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 0x0000 count value corresponds to the center of a period.)

### 16.5.3 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 16.5.1, “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 16.5.1, “Clearing Timer Interrupt Flags.”

### 16.5.4 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 16.5.1, “Clearing Timer Interrupt Flags.”

# Chapter 17
# Development Support

## 17.1 Introduction

Development support systems in the HCS08 include the background debug controller (BDC) and the on-chip debug module (DBG). The BDC provides a single-wire debug interface to the target MCU that provides a convenient interface for programming the on-chip FLASH and other nonvolatile memories. The BDC is also the primary debug interface for development and allows non-intrusive access to memory data and traditional debug features such as CPU register modify, breakpoints, and single instruction trace commands.

In the HCS08 Family, address and data bus signals are not available on external pins. Debug is done through commands fed into the target MCU via the single-wire background debug interface. The debug module provides a means to selectively trigger and capture bus information so an external development system can reconstruct what happened inside the MCU on a cycle-by-cycle basis without having external access to the address and data signals.

### 17.1.1 Module Configuration

The alternate BDC clock source is the ICSLCLK. This clock source is selected by clearing the CLKSW bit in the BDCSCR register. For details on ICSLCLK, see Section 10.4, “Functional Description” of the ICS chapter.

### 17.1.2 Features

Features of the BDC module include:

- Single pin for mode selection and background communications
- BDC registers are not located in the memory map
- SYNC command to determine target communications rate
- Non-intrusive commands for memory access
- Active background mode commands for CPU register access
- GO and TRACE1 commands
- BACKGROUND command can wake CPU from stop or wait modes
- One hardware address breakpoint built into BDC
- Oscillator runs in stop mode, if BDC enabled
- COP watchdog disabled while in active background mode

Features of the ICE system include:

- Two trigger comparators: Two address + read/write (R/W) or one full address + data + R/W
- Flexible 8-word by 16-bit FIFO (first-in, first-out) buffer for capture information:
  - Change-of-flow addresses or
  - Event-only data
- Two types of breakpoints:
  - Tag breakpoints for instruction opcodes
  - Force breakpoints for any address access
- Nine trigger modes:
  - Basic: A-only, A OR B
  - Sequence: A then B
  - Full: A AND B data, A AND NOT B data
  - Event (store data): Event-only B, A then event-only B
  - Range: Inside range (A ≤ address ≤ B), outside range (address < A or address > B)

## 17.2 Background Debug Controller (BDC)

All MCUs in the HCS08 Family contain a single-wire background debug interface that supports in-circuit programming of on-chip nonvolatile memory and sophisticated non-intrusive debug capabilities. Unlike debug interfaces on earlier 8-bit MCUs, this system does not interfere with normal application resources. It does not use any user memory or locations in the memory map and does not share any on-chip peripherals.

BDC commands are divided into two groups:

- Active background mode commands require that the target MCU is in active background mode (the user program is not running). Active background mode commands allow the CPU registers to be

read or written, and allow the user to trace one user instruction at a time, or GO to the user program from active background mode.

- Non-intrusive commands can be executed at any time even while the user's program is running. Non-intrusive commands allow a user to read or write MCU memory locations or access status and control registers within the background debug controller.

Typically, a relatively simple interface pod is used to translate commands from a host computer into commands for the custom serial interface to the single-wire background debug system. Depending on the development tool vendor, this interface pod may use a standard RS-232 serial port, a parallel printer port, or some other type of communications such as a universal serial bus (USB) to communicate between the host PC and the pod. The pod typically connects to the target system with ground, the BKGD pin, $\overline{\text{RESET}}$, and sometimes $V_{DD}$. An open-drain connection to reset allows the host to force a target system reset, which is useful to regain control of a lost target system or to control startup of a target system before the on-chip nonvolatile memory has been programmed. Sometimes $V_{DD}$ can be used to allow the pod to use power from the target system to avoid the need for a separate power supply. However, if the pod is powered separately, it can be connected to a running target system without forcing a target system reset or otherwise disturbing the running application program.

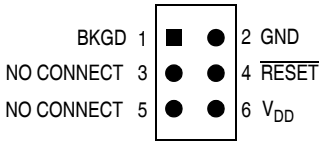



**Figure 17-1. BDM Tool Connector**

## 17.2.1 BKGD Pin Description

BKGD is the single-wire background debug interface pin. The primary function of this pin is for bidirectional serial communication of active background mode commands and data. During reset, this pin is used to select between starting in active background mode or starting the user's application program. This pin is also used to request a timed sync response pulse to allow a host development tool to determine the correct clock frequency for background debug serial communications.

BDC serial communications use a custom serial protocol first introduced on the M68HC12 Family of microcontrollers. This protocol assumes the host knows the communication clock rate that is determined by the target BDC clock rate. All communication is initiated and controlled by the host that drives a high-to-low edge to signal the beginning of each bit time. Commands and data are sent most significant bit first (MSB first). For a detailed description of the communications protocol, refer to Section 17.2.2, "Communication Details."

If a host is attempting to communicate with a target MCU that has an unknown BDC clock rate, a SYNC command may be sent to the target MCU to request a timed sync response signal from which the host can determine the correct communication speed.

BKGD is a pseudo-open-drain pin and there is an on-chip pullup so no external pullup resistor is required. Unlike typical open-drain pins, the external RC time constant on this pin, which is influenced by external capacitance, plays almost no role in signal rise time. The custom protocol provides for brief, actively

driven speedup pulses to force rapid rise times on this pin without risking harmful drive level conflicts. Refer to Section 17.2.2, “Communication Details,” for more detail.

When no debugger pod is connected to the 6-pin BDM interface connector, the internal pullup on BKGD chooses normal operating mode. When a debug pod is connected to BKGD it is possible to force the MCU into active background mode after reset. The specific conditions for forcing active background depend upon the HCS08 derivative (refer to the introduction to this Development Support section). It is not necessary to reset the target MCU to communicate with it through the background debug interface.

### 17.2.2 Communication Details

The BDC serial interface requires the external controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 BDC clock cycles per bit (nominal speed). The interface times out if 512 BDC clock cycles occur between falling edges from the host. Any BDC command that was in progress when this timeout occurs is aborted without affecting the memory or operating mode of the target MCU system.

The custom serial protocol requires the debug pod to know the target BDC communication clock speed.

The clock switch (CLKSW) control bit in the BDC status and control register allows the user to select the BDC clock source. The BDC clock source can either be the bus or the alternate BDC clock source.

The BKGD pin can receive a high or low level or transmit a high or low level. The following diagrams show timing for each of these cases. Interface timing is synchronous to clocks in the target BDC, but asynchronous to the external host. The internal BDC clock signal is shown for reference in counting cycles.

Figure 17-2 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target HCS08 MCU. The host is asynchronous to the target so there is a 0-to-1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target BDC clock cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Because the target does not drive the BKGD pin during the host-to-target transmission period, there is no need to treat the line as an open-drain signal during this period.

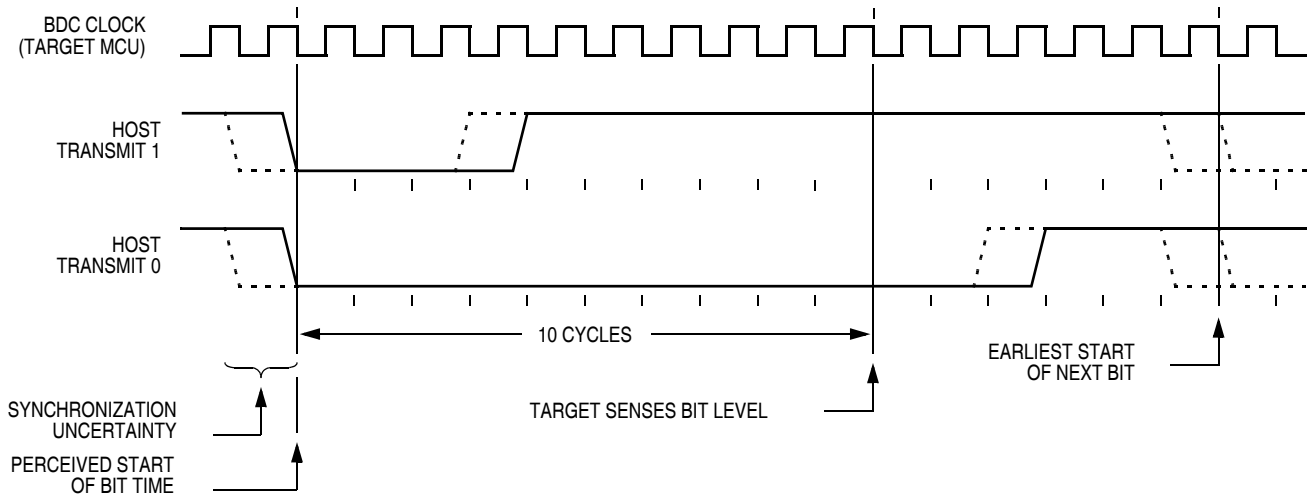



**Figure 17-2. BDC Host-to-Target Serial Bit Timing**

Figure 17-3 shows the host receiving a logic 1 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target BDC cycles). The host must release the low drive before the target MCU drives a brief active-high speedup pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

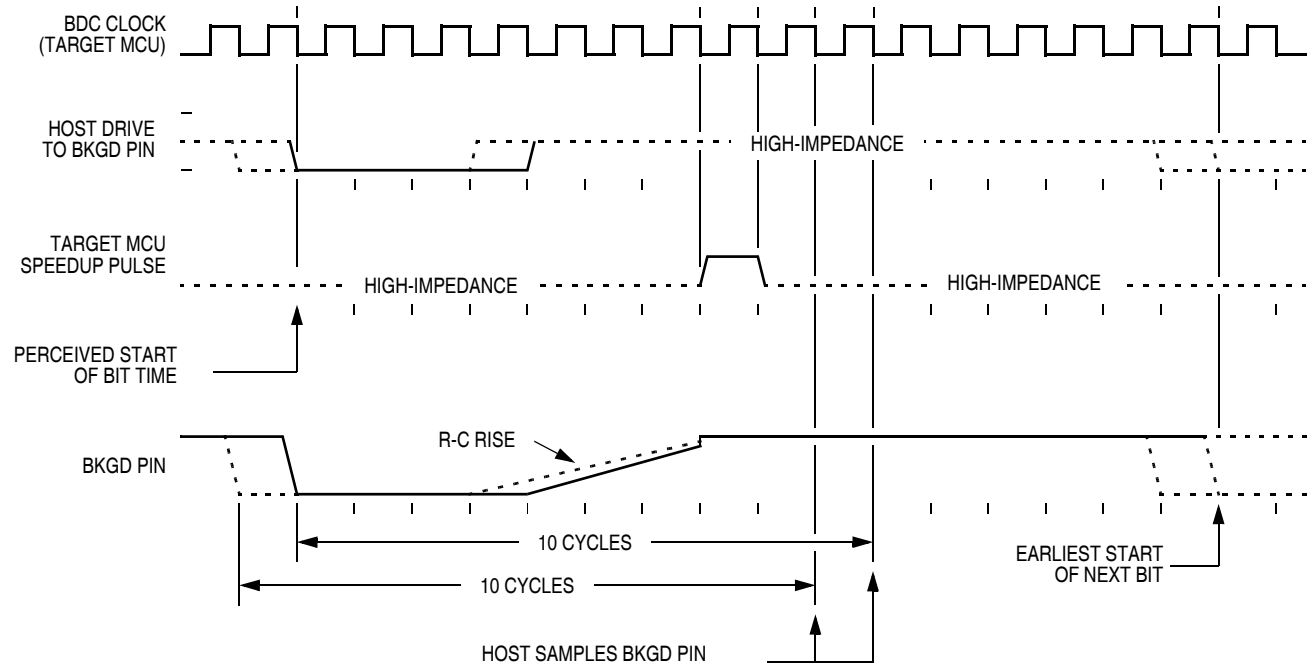



**Figure 17-3. BDC Target-to-Host Serial Bit Timing (Logic 1)**

Figure 17-4 shows the host receiving a logic 0 from the target HCS08 MCU. Because the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time but the target HCS08 finishes it. Because the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 BDC clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.

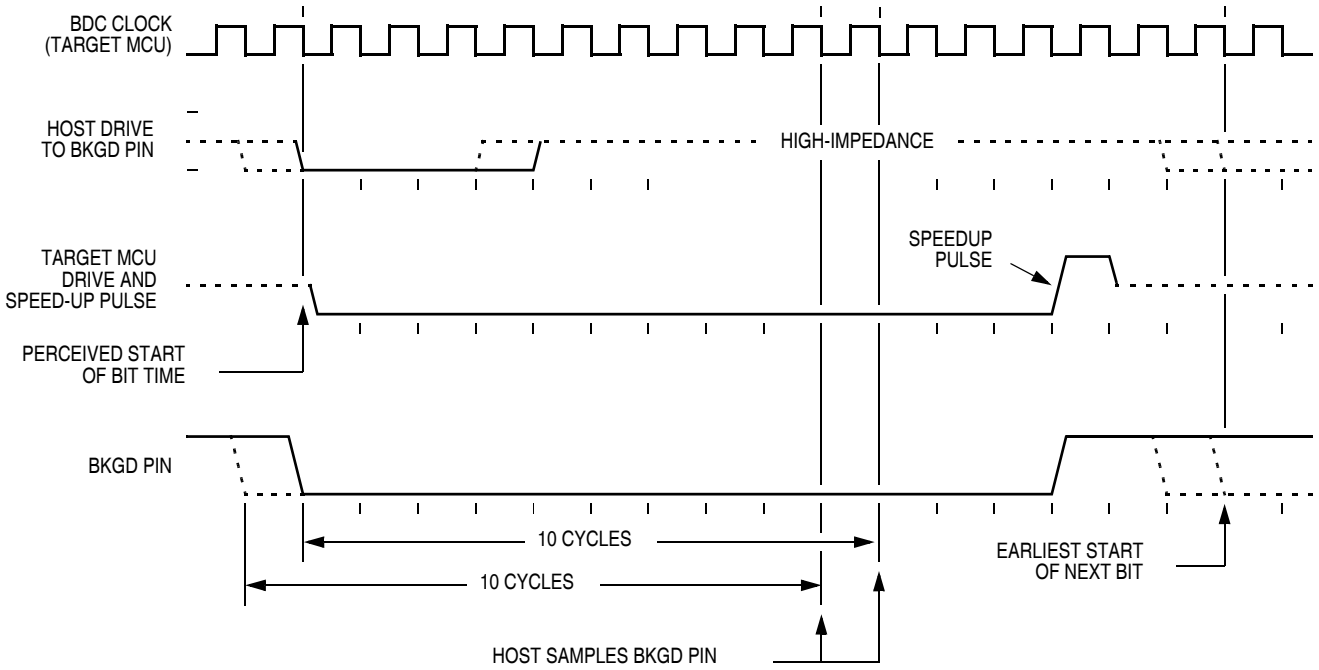



**Figure 17-4. BDM Target-to-Host Serial Bit Timing (Logic 0)**

## 17.2.3 BDC Commands

BDC commands are sent serially from a host computer to the BKGD pin of the target HCS08 MCU. All commands and data are sent MSB-first using a custom BDC communications protocol. Active background mode commands require that the target MCU is currently in the active background mode while non-intrusive commands may be issued at any time whether the target MCU is in active background mode or running a user application program.

Table 17-1 shows all HCS08 BDC commands, a shorthand description of their coding structure, and the meaning of each command.

### Coding Structure Nomenclature

This nomenclature is used in Table 17-1 to describe the coding structure of the BDC commands.

| | | |
|---|---|---|
| | | Commands begin with an 8-bit hexadecimal command code in the host-to-target direction (most significant bit first) |
| / | = | separates parts of the command |
| d | = | delay 16 target BDC clock cycles |
| AAAA | = | a 16-bit address in the host-to-target direction |
| RD | = | 8 bits of read data in the target-to-host direction |
| WD | = | 8 bits of write data in the host-to-target direction |
| RD16 | = | 16 bits of read data in the target-to-host direction |
| WD16 | = | 16 bits of write data in the host-to-target direction |
| SS | = | the contents of BDCSCR in the target-to-host direction (STATUS) |
| CC | = | 8 bits of write data for BDCSCR in the host-to-target direction (CONTROL) |
| RBKP | = | 16 bits of read data in the target-to-host direction (from BDCBKPT breakpoint register) |
| WBKP | = | 16 bits of write data in the host-to-target direction (for BDCBKPT breakpoint register) |

**Table 17-1. BDC Command Summary**

| Command Mnemonic | Active BDM/ Non-intrusive | Coding Structure | Description |
|---|---|---|---|
| SYNC | Non-intrusive | n/a[1] | Request a timed reference pulse to determine target BDC communication speed |
| ACK_ENABLE | Non-intrusive | D5/d | Enable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D. |
| ACK_DISABLE | Non-intrusive | D6/d | Disable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D. |
| BACKGROUND | Non-intrusive | 90/d | Enter active background mode if enabled (ignore if ENBDM bit equals 0) |
| READ_STATUS | Non-intrusive | E4/SS | Read BDC status from BDCSCR |
| WRITE_CONTROL | Non-intrusive | C4/CC | Write BDC controls in BDCSCR |
| READ_BYTE | Non-intrusive | E0/AAAA/d/RD | Read a byte from target memory |
| READ_BYTE_WS | Non-intrusive | E1/AAAA/d/SS/RD | Read a byte and report status |
| READ_LAST | Non-intrusive | E8/SS/RD | Re-read byte from address just read and report status |
| WRITE_BYTE | Non-intrusive | C0/AAAA/WD/d | Write a byte to target memory |
| WRITE_BYTE_WS | Non-intrusive | C1/AAAA/WD/d/SS | Write a byte and report status |
| READ_BKPT | Non-intrusive | E2/RBKP | Read BDCBKPT breakpoint register |
| WRITE_BKPT | Non-intrusive | C2/WBKP | Write BDCBKPT breakpoint register |
| GO | Active BDM | 08/d | Go to execute the user application program starting at the address currently in the PC |
| TRACE1 | Active BDM | 10/d | Trace 1 user instruction at the address in the PC, then return to active background mode |
| TAGGO | Active BDM | 18/d | Same as GO but enable external tagging (HCS08 devices have no external tagging pin) |
| READ_A | Active BDM | 68/d/RD | Read accumulator (A) |
| READ_CCR | Active BDM | 69/d/RD | Read condition code register (CCR) |
| READ_PC | Active BDM | 6B/d/RD16 | Read program counter (PC) |
| READ_HX | Active BDM | 6C/d/RD16 | Read H and X register pair (H:X) |
| READ_SP | Active BDM | 6F/d/RD16 | Read stack pointer (SP) |
| READ_NEXT | Active BDM | 70/d/RD | Increment H:X by one then read memory byte located at H:X |
| READ_NEXT_WS | Active BDM | 71/d/SS/RD | Increment H:X by one then read memory byte located at H:X. Report status and data. |
| WRITE_A | Active BDM | 48/WD/d | Write accumulator (A) |
| WRITE_CCR | Active BDM | 49/WD/d | Write condition code register (CCR) |
| WRITE_PC | Active BDM | 4B/WD16/d | Write program counter (PC) |
| WRITE_HX | Active BDM | 4C/WD16/d | Write H and X register pair (H:X) |
| WRITE_SP | Active BDM | 4F/WD16/d | Write stack pointer (SP) |
| WRITE_NEXT | Active BDM | 50/WD/d | Increment H:X by one, then write memory byte located at H:X |
| WRITE_NEXT_WS | Active BDM | 51/WD/d/SS | Increment H:X by one, then write memory byte located at H:X. Also report status. |

[1] The SYNC command is a special operation that does not have a command code.

The SYNC command is unlike other BDC commands because the host does not necessarily know the correct communications speed to use for BDC communications until after it has analyzed the response to the SYNC command.

To issue a SYNC command, the host:

- Drives the BKGD pin low for at least 128 cycles of the slowest possible BDC clock (The slowest clock is normally the reference oscillator/64 or the self-clocked rate/64.)
- Drives BKGD high for a brief speedup pulse to get a fast rise time (This speedup pulse is typically one cycle of the fastest clock in the system.)
- Removes all drive to the BKGD pin so it reverts to high impedance
- Monitors the BKGD pin for the sync response pulse

The target, upon detecting the SYNC request from the host (which is a much longer low time than would ever occur during normal BDC communications):

- Waits for BKGD to return to a logic high
- Delays 16 cycles to allow the host to stop driving the high speedup pulse
- Drives BKGD low for 128 BDC clock cycles
- Drives a 1-cycle high speedup pulse to force a fast rise time on BKGD
- Removes all drive to the BKGD pin so it reverts to high impedance

The host measures the low time of this 128-cycle sync response pulse and determines the correct speed for subsequent BDC communications. Typically, the host can determine the correct communication speed within a few percent of the actual target speed and the communication protocol can easily tolerate speed errors of several percent.

### 17.2.4 BDC Hardware Breakpoint

The BDC includes one relatively simple hardware breakpoint that compares the CPU address bus to a 16-bit match value in the BDCBKPT register. This breakpoint can generate a forced breakpoint or a tagged breakpoint. A forced breakpoint causes the CPU to enter active background mode at the first instruction boundary following any access to the breakpoint address. The tagged breakpoint causes the instruction opcode at the breakpoint address to be tagged so that the CPU will enter active background mode rather than executing that instruction if and when it reaches the end of the instruction queue. This implies that tagged breakpoints can only be placed at the address of an instruction opcode while forced breakpoints can be set at any address.

The breakpoint enable (BKPTEN) control bit in the BDC status and control register (BDCSCR) is used to enable the breakpoint logic (BKPTEN = 1). When BKPTEN = 0, its default value after reset, the breakpoint logic is disabled and no BDC breakpoints are requested regardless of the values in other BDC breakpoint registers and control bits. The force/tag select (FTS) control bit in BDCSCR is used to select forced (FTS = 1) or tagged (FTS = 0) type breakpoints.

The on-chip debug module (DBG) includes circuitry for two additional hardware breakpoints that are more flexible than the simple breakpoint in the BDC module.

## 17.3 On-Chip Debug System (DBG)

Because HCS08 devices do not have external address and data buses, the most important functions of an in-circuit emulator have been built onto the chip with the MCU. The debug system consists of an 8-stage FIFO that can store address or data bus information, and a flexible trigger system to decide when to capture bus information and what information to capture. The system relies on the single-wire background debug system to access debug control registers and to read results out of the eight stage FIFO.

The debug module includes control and status registers that are accessible in the user's memory map. These registers are located in the high register space to avoid using valuable direct page memory space.

Most of the debug module's functions are used during development, and user programs rarely access any of the control and status registers for the debug module. The one exception is that the debug system can provide the means to implement a form of ROM patching. This topic is discussed in greater detail in Section 17.3.6, "Hardware Breakpoints."

### 17.3.1 Comparators A and B

Two 16-bit comparators (A and B) can optionally be qualified with the R/W signal and an opcode tracking circuit. Separate control bits allow you to ignore R/W for each comparator. The opcode tracking circuitry optionally allows you to specify that a trigger will occur only if the opcode at the specified address is actually executed as opposed to only being read from memory into the instruction queue. The comparators are also capable of magnitude comparisons to support the inside range and outside range trigger modes. Comparators are disabled temporarily during all BDC accesses.

The A comparator is always associated with the 16-bit CPU address. The B comparator compares to the CPU address or the 8-bit CPU data bus, depending on the trigger mode selected. Because the CPU data bus is separated into a read data bus and a write data bus, the RWAEN and RWA control bits have an additional purpose, in full address plus data comparisons they are used to decide which of these buses to use in the comparator B data bus comparisons. If RWAEN = 1 (enabled) and RWA = 0 (write), the CPU's write data bus is used. Otherwise, the CPU's read data bus is used.

The currently selected trigger mode determines what the debugger logic does when a comparator detects a qualified match condition. A match can cause:

- Generation of a breakpoint to the CPU
- Storage of data bus values into the FIFO
- Starting to store change-of-flow addresses into the FIFO (begin type trace)
- Stopping the storage of change-of-flow addresses into the FIFO (end type trace)

### 17.3.2 Bus Capture Information and FIFO Operation

The usual way to use the FIFO is to setup the trigger mode and other control options, then arm the debugger. When the FIFO has filled or the debugger has stopped storing data into the FIFO, you would read the information out of it in the order it was stored into the FIFO. Status bits indicate the number of words of valid information that are in the FIFO as data is stored into it. If a trace run is manually halted by writing 0 to ARM before the FIFO is full (CNT = 1:0:0:0), the information is shifted by one position and

the host must perform ((8 – CNT) – 1) dummy reads of the FIFO to advance it to the first significant entry in the FIFO.

In most trigger modes, the information stored in the FIFO consists of 16-bit change-of-flow addresses. In these cases, read DBGFH then DBGFL to get one coherent word of information out of the FIFO. Reading DBGFL (the low-order byte of the FIFO data port) causes the FIFO to shift so the next word of information is available at the FIFO data port. In the event-only trigger modes (see Section 17.3.5, “Trigger Modes”), 8-bit data information is stored into the FIFO. In these cases, the high-order half of the FIFO (DBGFH) is not used and data is read out of the FIFO by simply reading DBGFL. Each time DBGFL is read, the FIFO is shifted so the next data value is available through the FIFO data port at DBGFL.

In trigger modes where the FIFO is storing change-of-flow addresses, there is a delay between CPU addresses and the input side of the FIFO. Because of this delay, if the trigger event itself is a change-of-flow address or a change-of-flow address appears during the next two bus cycles after a trigger event starts the FIFO, it will not be saved into the FIFO. In the case of an end-trace, if the trigger event is a change-of-flow, it will be saved as the last change-of-flow entry for that debug run.

The FIFO can also be used to generate a profile of executed instruction addresses when the debugger is not armed. When ARM = 0, reading DBGFL causes the address of the most-recently fetched opcode to be saved in the FIFO. To use the profiling feature, a host debugger would read addresses out of the FIFO by reading DBGFH then DBGFL at regular periodic intervals. The first eight values would be discarded because they correspond to the eight DBGFL reads needed to initially fill the FIFO. Additional periodic reads of DBGFH and DBGFL return delayed information about executed instructions so the host debugger can develop a profile of executed instruction addresses.

### 17.3.3 Change-of-Flow Information

To minimize the amount of information stored in the FIFO, only information related to instructions that cause a change to the normal sequential execution of instructions is stored. With knowledge of the source and object code program stored in the target system, an external debugger system can reconstruct the path of execution through many instructions from the change-of-flow information stored in the FIFO.

For conditional branch instructions where the branch is taken (branch condition was true), the source address is stored (the address of the conditional branch opcode). Because BRA and BRN instructions are not conditional, these events do not cause change-of-flow information to be stored in the FIFO.

Indirect JMP and JSR instructions use the current contents of the H:X index register pair to determine the destination address, so the debug system stores the run-time destination address for any indirect JMP or JSR. For interrupts, RTI, or RTS, the destination address is stored in the FIFO as change-of-flow information.

### 17.3.4 Tag vs. Force Breakpoints and Triggers

Tagging is a term that refers to identifying an instruction opcode as it is fetched into the instruction queue, but not taking any other action until and unless that instruction is actually executed by the CPU. This distinction is important because any change-of-flow from a jump, branch, subroutine call, or interrupt causes some instructions that have been fetched into the instruction queue to be thrown away without being executed.

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the 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.

### 17.3.5 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.

### 17.3.6 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 17.3.5, “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.

## 17.4 Register Definition

This section contains the descriptions of the BDC and DBG registers and control bits.

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

### 17.4.1 BDC Registers and Control Bits

The BDC has two registers:

- The BDC status and control register (BDCSCR) is an 8-bit register containing control and status bits for the background debug controller.
- The BDC breakpoint match register (BDCBKPT) holds a 16-bit breakpoint match address.

These registers are accessed with dedicated serial BDC commands and are not located in the memory space of the target MCU (so they do not have addresses and cannot be accessed by user programs).

Some of the bits in the BDCSCR have write limitations; otherwise, these registers may be read or written at any time. For example, the ENBDM control bit may not be written while the MCU is in active background mode. (This prevents the ambiguous condition of the control bit forbidding active background mode while the MCU is already in active background mode.) Also, the four status bits (BDMACT, WS, WSF, and DVF) are read-only status indicators and can never be written by the WRITE_CONTROL serial BDC command. The clock switch (CLKSW) control bit may be read or written at any time.

### 17.4.1.1 BDC Status and Control Register (BDCSCR)

This register can be read or written by serial BDC commands (READ_STATUS and WRITE_CONTROL) but is not accessible to user programs because it is not located in the normal memory map of the MCU.

| | 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 17-5. BDC Status and Control Register (BDCSCR)**

**Table 17-2. 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 17-2. 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 MC9S08QG8/4 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 |

#### 17.4.1.2 BDC Breakpoint Match Register (BDCBKPT)

This 16-bit register holds the address for the hardware breakpoint in the BDC. The BKPTEN and FTS control bits in BDCSCR are used to enable and configure the breakpoint logic. Dedicated serial BDC commands (READ_BKPT and WRITE_BKPT) are used to read and write the BDCBKPT register but is not accessible to user programs because it is not located in the normal memory map of the MCU. Breakpoints are normally set while the target MCU is in active background mode before running the user application program. For additional information about setup and use of the hardware breakpoint logic in the BDC, refer to Section 17.2.4, “BDC Hardware Breakpoint.”

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

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

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

= Unimplemented or Reserved

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

**Figure 17-6. System Background Debug Force Reset Register (SBDFR)**

**Table 17-3. 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. |

## 17.4.3 DBG Registers and Control Bits

The debug module includes nine bytes of register space for three 16-bit registers and three 8-bit control and status registers. These registers are located in the high register space of the normal memory map so they are accessible to normal application programs. These registers are rarely if ever accessed by normal user application programs with the possible exception of a ROM patching mechanism that uses the breakpoint logic.

### 17.4.3.1 Debug Comparator A High Register (DBGCAH)

This register contains compare value bits for the high-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

### 17.4.3.2 Debug Comparator A Low Register (DBGCAL)

This register contains compare value bits for the low-order eight bits of comparator A. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

### 17.4.3.3 Debug Comparator B High Register (DBGCBH)

This register contains compare value bits for the high-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

### 17.4.3.4 Debug Comparator B Low Register (DBGCBL)

This register contains compare value bits for the low-order eight bits of comparator B. This register is forced to 0x00 at reset and can be read at any time or written at any time unless ARM = 1.

#### 17.4.3.5 Debug FIFO High Register (DBGFH)

This register provides read-only access to the high-order eight bits of the FIFO. Writes to this register have no meaning or effect. In the event-only trigger modes, the FIFO only stores data into the low-order byte of each FIFO word, so this register is not used and will read 0x00.

Reading DBGFH does not cause the FIFO to shift to the next word. When reading 16-bit words out of the FIFO, read DBGFH before reading DBGFL because reading DBGFL causes the FIFO to advance to the next word of information.

#### 17.4.3.6 Debug FIFO Low Register (DBGFL)

This register provides read-only access to the low-order eight bits of the FIFO. Writes to this register have no meaning or effect.

Reading DBGFL causes the FIFO to shift to the next available word of information. When the debug module is operating in event-only modes, only 8-bit data is stored into the FIFO (high-order half of each FIFO word is unused). When reading 8-bit words out of the FIFO, simply read DBGFL repeatedly to get successive bytes of data from the FIFO. It isn't necessary to read DBGFH in this case.

Do not attempt to read data from the FIFO while it is still armed (after arming but before the FIFO is filled or ARMF is cleared) because the FIFO is prevented from advancing during reads of DBGFL. This can interfere with normal sequencing of reads from the FIFO.

Reading DBGFL while the debugger is not armed causes the address of the most-recently fetched opcode to be stored to the last location in the FIFO. By reading DBGFH then DBGFL periodically, external host software can develop a profile of program execution. After eight reads from the FIFO, the ninth read will return the information that was stored as a result of the first read. To use the profiling feature, read the FIFO eight times without using the data to prime the sequence and then begin using the data to get a delayed picture of what addresses were being executed. The information stored into the FIFO on reads of DBGFL (while the FIFO is not armed) is the address of the most-recently fetched opcode.

### 17.4.3.7 Debug Control Register (DBGC)

This register can be read or written at any time.

| | 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 17-7. Debug Control Register (DBGC)**

**Table 17-4. 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 |

### 17.4.3.8 Debug Trigger Register (DBGT)

This register can be read any time, but may be written only if ARM = 0, except bits 4 and 5 are hard-wired to 0s.

| | 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 17-8. Debug Trigger Register (DBGT)**

**Table 17-5. 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) |

### 17.4.3.9 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 17-9. Debug Status Register (DBGS)**

**Table 17-6. 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. 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)<br>C<br>M | $T_A$ | $T_L$ to $T_H$<br>–40 to 85<br>–40 to 125 | °C |
| Thermal resistance<br>Single-layer board | | | |
| 8-pin PDIP | $\theta_{JA}$ | 113 | °C/W |
| 8-pin NB SOIC | | 150 | |
| 8-pin DFN | | 179 | |
| 16-pin PDIP | | 78 | |
| 16-pin TSSOP | | 133 | |
| 16-pin QFN | | 132 | |
| 24-pin QFN | | 125 | |
| Thermal resistance<br>Four-layer board | | | |
| 8-pin PDIP | $\theta_{JA}$ | 72 | °C/W |
| 8-pin NB SOIC | | 87 | |
| 8-pin DFN | | 41 | |
| 16-pin PDIP | | 53 | |
| 16-pin TSSOP | | 86 | |
| 16-pin QFN | | 36 | |
| 24-pin QFN | | 44 | |

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 Equation A-1 and Equation A-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 Equation A-1 and Equation A-2 iteratively for any value of $T_A$.

## A.4 ESD Protection and Latch-Up Immunity

Although damage from electrostatic discharge (ESD) is much less common on these devices than on early CMOS circuits, normal handling precautions 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 AEC-Q100 Stress Test Qualification for Automotive Grade Integrated Circuits. During the device qualification ESD stresses were performed for the human body model (HBM), the machine model (MM) and the charge device model (CDM).

A device is defined as a failure if after exposure to ESD pulses the device no longer meets the device specification. Complete DC parametric and functional testing is performed per the applicable device specification at room temperature followed by hot temperature, unless specified otherwise in the device specification.

**Table A-3. ESD and Latch-up Test Conditions**

| Model | Description | Symbol | Value | Unit |
|---|---|---|---|---|
| Human Body | Series resistance | R1 | 1500 | Ω |
| | Storage capacitance | C | 100 | pF |
| | Number of pulses per pin | — | 3 | |
| Machine | Series resistance | R1 | 0 | Ω |
| | Storage capacitance | C | 200 | pF |
| | Number of pulses per pin | — | 3 | |
| Latch-up | Minimum input voltage limit | | – 2.5 | V |
| | Maximum input voltage limit | | 7.5 | V |

**Table A-4. ESD and Latch-Up Protection Characteristics**

| No. | Rating[1] | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | Human body model (HBM) | $V_{HBM}$ | ± 2000 | — | V |
| 2 | Machine model (MM) | $V_{MM}$ | ± 200 | — | V |
| 3 | Charge device model (CDM) | $V_{CDM}$ | ± 500 | — | V |
| 4 | Latch-up current at $T_A$ = 125°C | $I_{LAT}$ | ± 100 | — | mA |

1 Parameter is achieved by design characterization on a small sample size from typical devices under typical conditions unless otherwise noted.

## A.5 DC Characteristics

This section includes information about power supply requirements and I/O pin characteristics.

**Table A-5. Operating Range**

| Parameter | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Supply voltage (run, wait and stop modes.) | $V_{DD}$ | 1.8[1] | | 3.6 | V |
| Temperature<br>C<br>M | | <br>–40<br>–40 | <br>—<br>— | <br>85<br>125 | °C |

[1] As the supply voltage rises, the LVD circuit will hold the MCU in reset until the supply has risen above $V_{LVDL}$.

**Table A-6. DC Characteristics**

| Parameter | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Minimum RAM retention supply voltage applied to $V_{DD}$ | $V_{RAM}$ | $V_{POR}$[1, 2] | | — | V |
| Low-voltage detection threshold — high range<br>($V_{DD}$ falling)<br>($V_{DD}$ rising) | $V_{LVDH}$ | <br>2.08<br>2.16 | <br>2.1<br>2.19 | <br>2.2<br>2.27 | V |
| Low-voltage detection threshold — low range<br>($V_{DD}$ falling)<br>($V_{DD}$ rising) | $V_{LVDL}$ | <br>1.80<br>1.88 | <br>1.82<br>1.90 | <br>1.91<br>1.99 | V |
| Low-voltage warning threshold — high range<br>($V_{DD}$ falling)<br>($V_{DD}$ rising) | $V_{LVWH}$ | <br>2.35<br>2.35 | <br>2.40<br>2.40 | <br>2.5<br>2.5 | V |
| Low-voltage warning threshold — low range<br>($V_{DD}$ falling)<br>($V_{DD}$ rising) | $V_{LVWL}$ | <br>2.08<br>2.16 | <br>2.1<br>2.19 | <br>2.2<br>2.27 | V |
| Power on reset (POR) re-arm voltage | $V_{POR}$ | | 1.4 | | V |
| Bandgap Voltage Reference | $V_{BG}$ | 1.18 | 1.20 | 1.21 | V |
| Input high voltage ($V_{DD}$ > 2.3 V) (all digital inputs) | $V_{IH}$ | $0.70 \times V_{DD}$ | | — | V |
| Input high voltage (1.8 V ≤ $V_{DD}$ ≤ 2.3 V) (all digital inputs) | | $0.85 \times V_{DD}$ | | — | |
| Input low voltage ($V_{DD}$ > 2.3 V) (all digital inputs) | $V_{IL}$ | — | | $0.35 \times V_{DD}$ | V |
| Input low voltage (1.8 V ≤ $V_{DD}$ ≤ 2.3 V) (all digital inputs) | | — | | $0.30 \times V_{DD}$ | |
| Input hysteresis (all digital inputs) | $V_{hys}$ | $0.06 \times V_{DD}$ | | — | V |
| Input leakage current (Per pin)<br>$V_{In}$ = $V_{DD}$ or $V_{SS}$, all input only pins | $\lvert I_{In} \rvert$ | — | 0.025 | 1.0 | µA |
| High impedance (off-state) leakage current (per pin)<br>$V_{In}$ = $V_{DD}$ or $V_{SS}$, all input/output | $\lvert I_{OZ} \rvert$ | — | 0.025 | 1.0 | µA |
| Internal pullup resistors[3,4] | $R_{PU}$ | 17.5 | | 52.5 | kΩ |

**Table A-6. DC Characteristics (continued)**

| Parameter | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Internal pulldown resistor (KBI) | $R_{PD}$ | 17.5 | | 52.5 | kΩ |
| Output high voltage — low drive (PTxDSn = 0)<br>$I_{OH}$ = –2 mA ($V_{DD} \geq$ 1.8 V) | $V_{OH}$ | $V_{DD}$ – 0.5 | | — | V |
| Output high voltage — high drive (PTxDSn = 1)<br>$I_{OH}$ = –10 mA ($V_{DD} \geq$ 2.7 V)<br>$I_{OH}$ = –6 mA ($V_{DD} \geq$ 2.3 V)<br>$I_{OH}$ = –3 mA ($V_{DD} \geq$ 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 — low drive (PTxDSn = 0)<br>$I_{OL}$ = 2.0 mA ($V_{DD} \geq$ 1.8 V) | $V_{OL}$ | — | | 0.5 | V |
| Output low voltage — high drive (PTxDSn = 1)<br>$I_{OL}$ = 10.0 mA ($V_{DD} \geq$ 2.7 V)<br>$I_{OL}$ = 6 mA ($V_{DD} \geq$ 2.3 V)<br>$I_{OL}$ = 3 mA ($V_{DD} \geq$ 1.8 V) | | —<br>—<br>— | | 0.5<br>0.5<br>0.5 | |
| Maximum total $I_{OL}$ for all port pins | $I_{OLT}$ | — | | 60 | mA |
| DC injection current [2, 5, 6, 7]<br>$V_{IN} < V_{SS}$, $V_{IN} > V_{DD}$<br>Single pin limit<br>Total MCU limit, includes sum of all stressed pins | $I_{IC}$ | <br>–0.2<br>–5 | | <br>0.2<br>5 | <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 Measurement condition for pull resistors: $V_{In} = V_{SS}$ for pullup and $V_{In} = V_{DD}$ for pulldown.

4 PTA5/$\overline{\text{IRQ}}$/TCLK/$\overline{\text{RESET}}$ pullup resistor may not pullup 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.

5 All functional non-supply pins are internally clamped to $V_{SS}$ and $V_{DD}$.

6 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.

7 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).

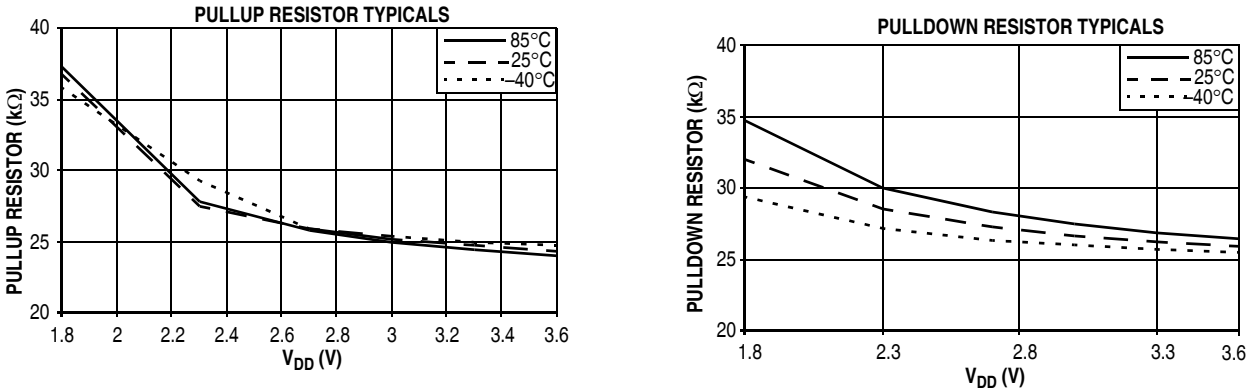



**Figure A-1. Pullup and Pulldown Typical Resistor Values ($V_{DD}$ = 3.0 V)**

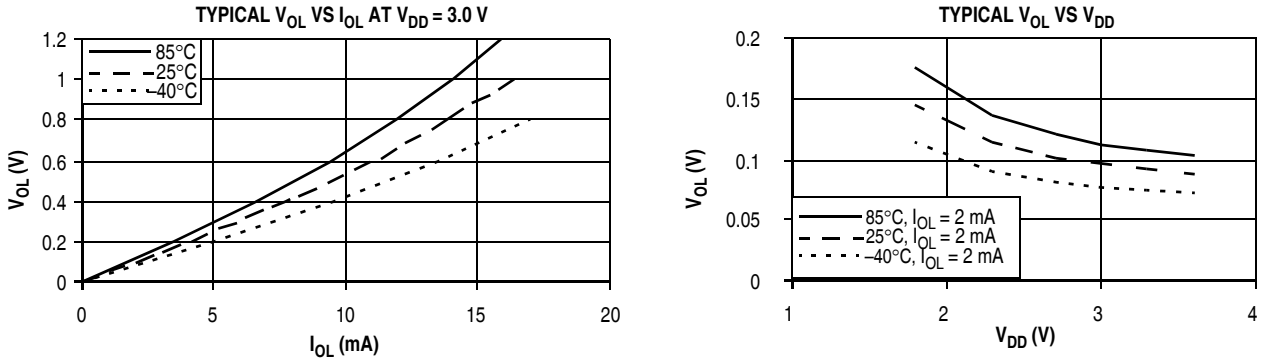



**Figure A-2. Typical Low-Side Driver (Sink) Characteristics — Low Drive (PTxDSn = 0)**

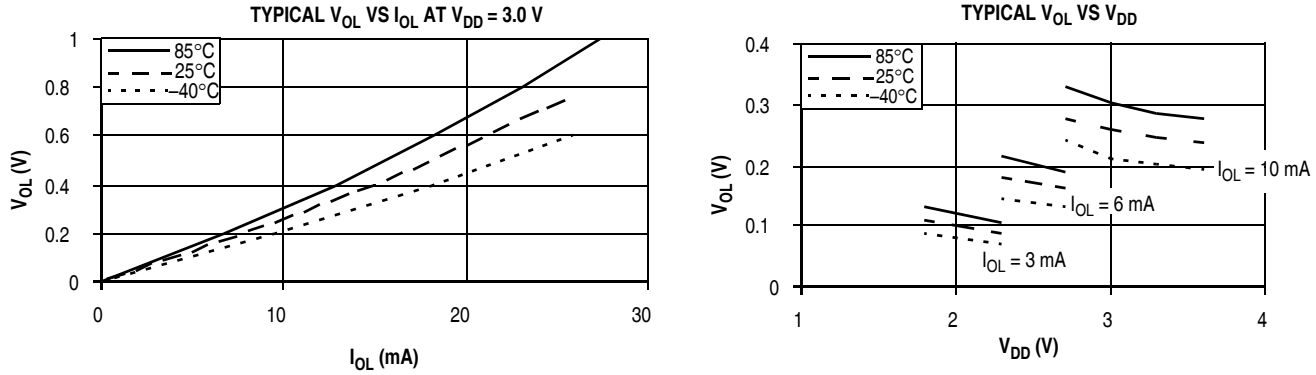



**Figure A-3. Typical Low-Side Driver (Sink) Characteristics — High Drive (PTxDSn = 1)**

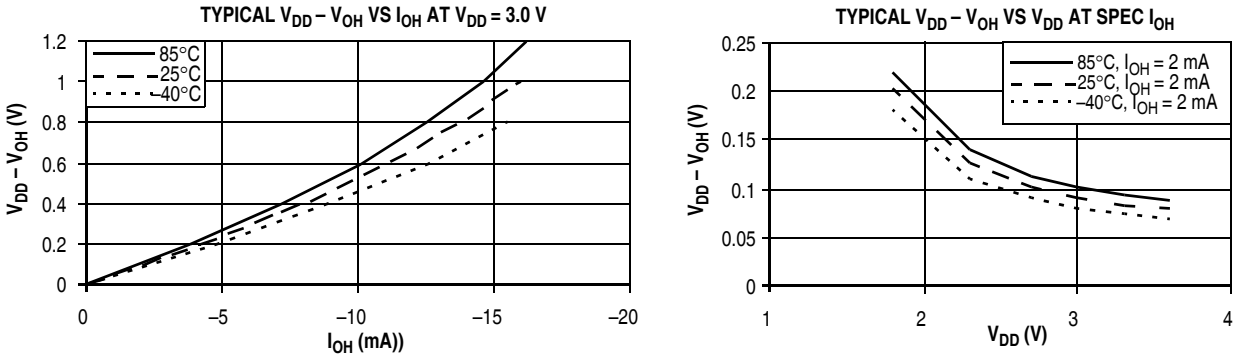



**Figure A-4. Typical High-Side (Source) Characteristics — Low Drive (PTxDSn = 0)**

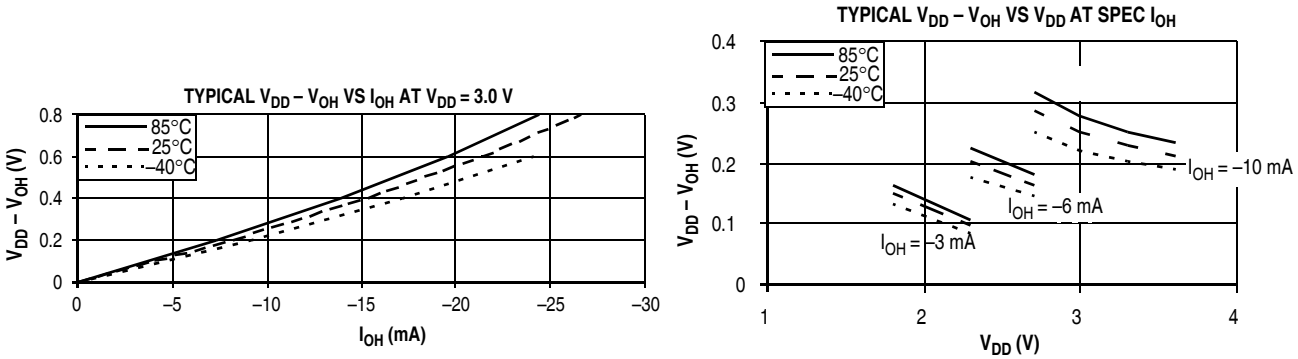



**Figure A-5. Typical High-Side (Source) Characteristics — High Drive (PTxDSn = 1)**

## A.6 Supply Current Characteristics

This section includes information about power supply current in various operating modes.

**Table A-7. Supply Current Characteristics**

| Parameter | Symbol | $V_{DD}$ (V)[1] | Typical[2] | Max | T (°C) |
|---|---|---|---|---|---|
| Run supply current [3] measured in FBE mode at $f_{Bus}$ = 8 MHz | $RI_{DD}$ | 3 | 3.5 mA | 5mA | 125 |
| | | 2 | 2.5 mA | — | 125 |
| Run supply current [3] measured in FBE mode at $f_{Bus}$ = 1 MHz | $RI_{DD}$ | 3 | 490 µA | 1mA | 125 |
| | | 2 | 370 µA | — | 125 |
| Wait mode supply current [4] measured in FBE at 8 MHz | $WI_{DD}$ | 3 | 1mA | 1.5mA | 125 |
| Stop1 mode supply current | $S1I_{DD}$ | 3 | 475 nA | 10µA<br>1.2µA | 125<br>85 |
| | | 2 | 470 nA | — | 85 |
| Stop2 mode supply current | $S2I_{DD}$ | 3 | 600 nA | 15 µA<br>2 µA | 125<br>85 |
| | | 2 | 550 nA | — | 85 |
| Stop3 mode supply current | $S3I_{DD}$ | 3 | 750 nA | 35 µA<br>6 µA | 125<br>85 |
| | | 2 | 680 nA | — | 85 |

**Table A-7. Supply Current Characteristics**

| Parameter | Symbol | $V_{DD}$ (V)[1] | Typical[2] | Max | T (°C) |
|---|---|---|---|---|---|
| RTI adder to stop1, stop2 or stop3 [4] | — | 3 | 300 nA | — | 85 |
| | | 2 | 300 nA | — | 85 |
| LVD adder to stop3 (LVDE = LVDSE = 1) | — | 3 | 70 µA | — | 85 |
| | | 2 | 60 µA | — | 85 |
| Adder to stop3 for oscillator enabled [5] (EREFSTEN =1) | — | 3 | 5 µA | — | 85 |
| | | 2 | 4 µA | — | 85 |

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.

4 Most customers are expected to find that auto-wakeup from a stop mode can be used instead of the higher current wait mode.

5 Values given under the following conditions: low range operation (RANGE = 0), Loss-of-clock disabled (LOCD = 1), low-power oscillator (HGO = 0).

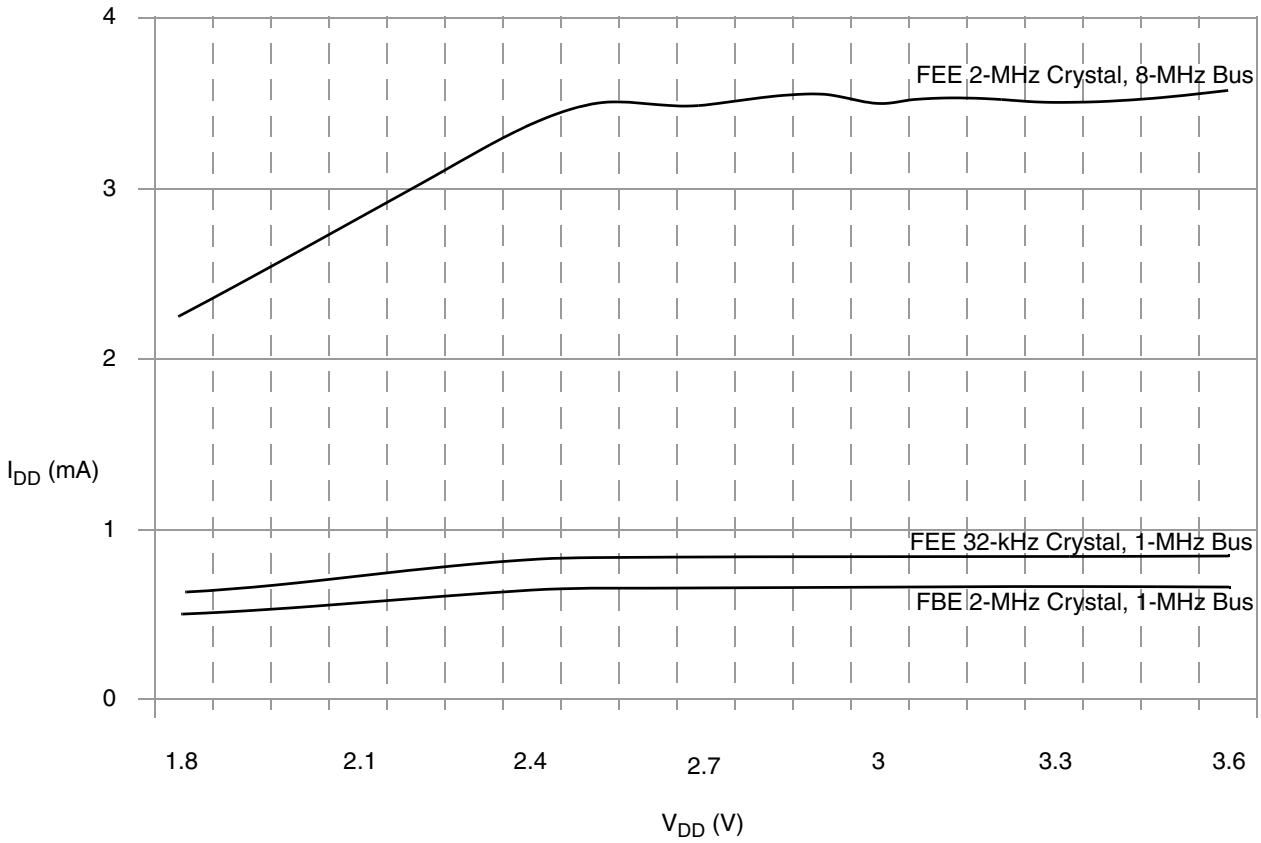



**Figure A-6. Typical Run $I_{DD}$ for FBE and FEE, $I_{DD}$ vs. $V_{DD}$ (ACMP and ADC off, All Other Modules Enabled)**

## A.7 External Oscillator (XOSC) and Internal Clock Source (ICS) Characteristics

Reference Figure A-7 for crystal or resonator circuit.

**Table A-8. XOSC and ICS Specifications (Temperature Range = –40 to 125°C Ambient)**

| Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Internal reference frequency — factory trimmed at $V_{DD}$ = 3.6V and temperature = 25°**C** | $f_{int_ft}$ | — | 31.25 | — | kHz |
| Oscillator crystal or resonator (EREFS = 1, ERCLKEN = 1)<br>Low range (RANGE = 0)<br>High range (RANGE = 1) FEE or FBE mode [1]<br>High range (RANGE = 1), high gain (HGO = 1), FBELP mode<br>High range (RANGE = 1), low power (HGO = 0), FBELP mode | <br>$f_{lo}$<br>$f_{hi}$<br>$f_{hi}$<br>$f_{hi}$ | <br>32<br>1<br>1<br>1 | <br>—<br>—<br>—<br>— | <br>38.4<br>5<br>16<br>8 | <br>kHz<br>MHz<br>MHz<br>MHz |
| Load capacitors | $C_1$<br>$C_2$ | See Note [2] | | | |
| Feedback resistor<br>Low range (32 kHz to 38.4 kHz)<br>High range (1 MHz to 16 MHz) | $R_F$ | | 10<br>1 | | MΩ<br>MΩ |
| Series resistor — Low range<br>Low Gain (HGO = 0)<br>High Gain (HGO = 1) | $R_S$ | —<br>— | 0<br>100 | —<br>— | kΩ |
| Series resistor — High range<br>Low Gain (HGO = 0)<br>High Gain (HGO = 1)<br>≥ 8 MHz<br>4 MHz<br>1 MHz | $R_S$ | <br><br>—<br>—<br>— | <br><br>0<br>0<br>0 | <br><br>0<br>10<br>20 | kΩ |
| Crystal start-up time [3, 4]<br>Low range, low power<br>Low range, high power<br>High range, low power<br>High range, high power | <br>$t_{CSTL}$<br><br>$t_{CSTH}$ | <br>—<br>—<br>—<br>— | <br>200<br>400<br>5<br>15 | <br>—<br>—<br>—<br>— | ms |
| Internal reference start-up time | $t_{IRST}$ | — | 60 | 100 | µs |
| Square wave input clock frequency (EREFS = 0, ERCLKEN = 1)<br>FEE or FBE mode [2]<br>FBELP mode | $f_{extal}$ | 0.03125<br>0 | —<br>— | 5<br>20 | MHz<br>MHz |
| Internal reference frequency - untrimmed[5] | $f_{int_ut}$ | 25 | 32.7 | 41.66 | kHz |
| Internal reference frequency - trimmed | $f_{int_t}$ | 31.25 | — | 39.06 | kHz |
| DCO output frequency range - untrimmed[5] $f_{dco}$= 512 * $f_{int_ut}$ | $f_{dco_ut}$ | 12.8 | 16.8 | 21.33 | MHz |
| DCO output frequency range - trimmed | $f_{dco_t}$ | 16 | — | 20 | MHz |

**Table A-8. XOSC and ICS Specifications (Temperature Range = –40 to 125°C Ambient)**

| Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Resolution of trimmed DCO output frequency at fixed voltage and temperature [4] | $\Delta f_{dco_res_t}$ | — | ±0.1 | ± 0.2 | $\%f_{dco}$ |
| Total deviation of DCO output from trimmed frequency:[3] At 8MHz over full voltage and temperature range (M Suffix) | $\Delta f_{dco_t}$ | — | –1.5 to ±0.5 | ± 3 | $\%f_{dco}$ |
| At 8MHz over full voltage and temperature rang (C Suffix) | | — | –1.0 to ±0.5 | ± 2 | |
| At 8MHz and 3.6V from 0 to 70°**C** (C Suffix) | | — | ±0.5 | ± 1 | |
| FLL acquisition time [4,6] | $t_{Acquire}$ | | | 1.5 | ms |
| Long term jitter of DCO output clock (averaged over 2-ms interval) [7] | $C_{Jitter}$ | — | 0.02 | 0.2 | $\%f_{dco}$ |

1 When ICS is configured for FEE or FBE mode, input clock source must be divisible using RDIV to within the range of 31.25 kHz to 39.0625 kHz.

2 See crystal or resonator manufacturer's recommendation.

3 This parameter is characterized and not tested on each device.

4 Proper PC board layout procedures must be followed to achieve specifications.

5 TRIM register at default value (0x80) and FTRIM control bit at default value (0x0).

6 This specification applies to any time the FLL reference source or reference divider is changed, trim value changed or changing from FLL disabled (FBELP, FBILP) to FLL enabled (FEI, FEE, FBE, FBI). If a crystal/resonator is being used as the reference, this specification assumes it is already running.

7 Jitter is the average deviation from the programmed frequency measured over the specified interval at maximum $f_{Bus}$. Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the FLL circuitry via $V_{DD}$ and $V_{SS}$ and variation in crystal oscillator frequency increase the $C_{Jitter}$ percentage for a given interval.

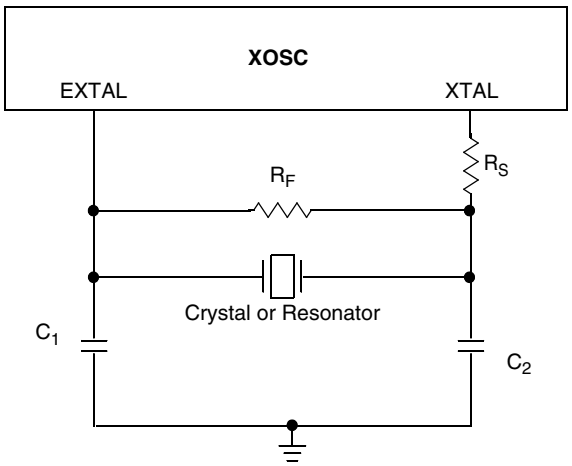



**Figure A-7. Typical Crystal or Resonator Circuit**

# A.8 AC Characteristics

This section describes timing characteristics for each peripheral system.

## A.8.1 Control Timing

**Table A-9. Control Timing**

| Parameter | Symbol | Min | Typ[1] | Max | Unit |
|---|---|---|---|---|---|
| Bus frequency ($t_{cyc}$ = 1/$f_{Bus}$) | $f_{Bus}$ | 0 | — | 10 | MHz |
| Real-time interrupt internal oscillator period | $t_{RTI}$ | 700 | 1000 | 1300 | µs |
| External reset pulse width[2] | $t_{extrst}$ | 100 | — | — | ns |
| $\overline{IRQ}$ pulse width<br>Asynchronous path[2]<br>Synchronous path[3] | $t_{ILIH}$ | <br>100<br>1.5 $t_{cyc}$ | — | — | ns |
| KBIPx pulse width<br>Asynchronous path[2]<br>Synchronous path[3] | $t_{ILIH}$, $t_{IHIL}$ | <br>100<br>1.5 $t_{cyc}$ | — | — | ns |
| Port rise and fall time (load = 50 pF)[4]<br>Slew rate control disabled (PTxSE = 0)<br>Slew rate control enabled (PTxSE = 1) | $t_{Rise}$, $t_{Fall}$ | <br>—<br>— | <br>3<br>30 | <br>—<br>— | ns |
| BKGD/MS setup time after issuing background debug force reset to enter user or BDM modes | $t_{MSSU}$ | 500 | — | — | ns |
| BKGD/MS hold time after issuing background debug force reset to enter user or BDM modes [5] | $t_{MSH}$ | 100 | — | — | µs |

1 Data in Typical column was characterized at 3.0 V, 25°C.

2 This is the shortest pulse that is guaranteed to be recognized.

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.

5 To enter BDM mode following a POR, BKGD/MS should be held low during the power-up and for a hold time of $t_{MSH}$ after $V_{DD}$ rises above $V_{LVD}$.

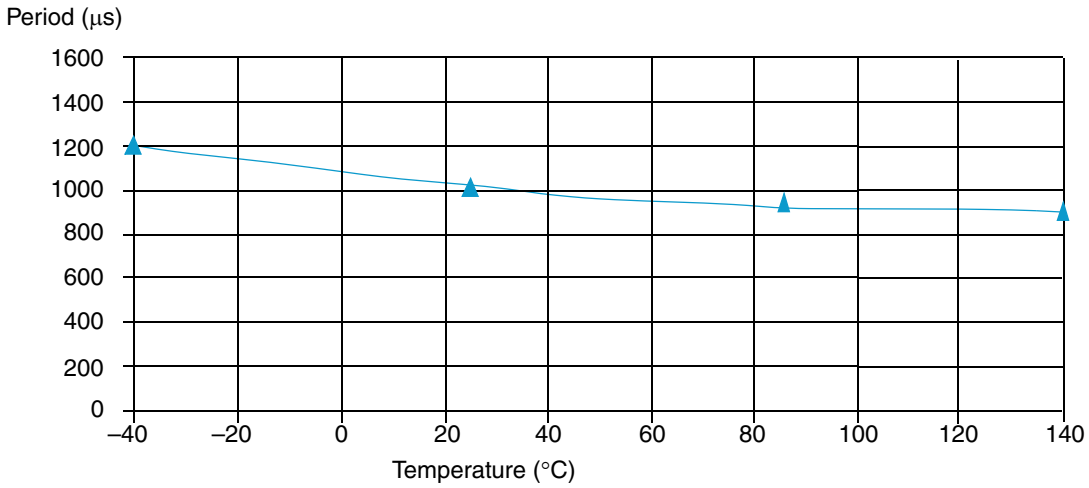



**Figure A-8. Typical RTI Clock Period vs. Temperature**

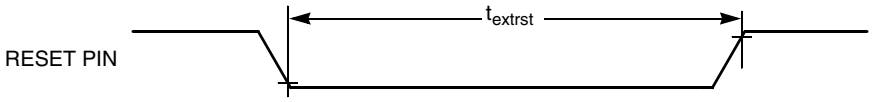



**Figure A-9. Reset Timing**

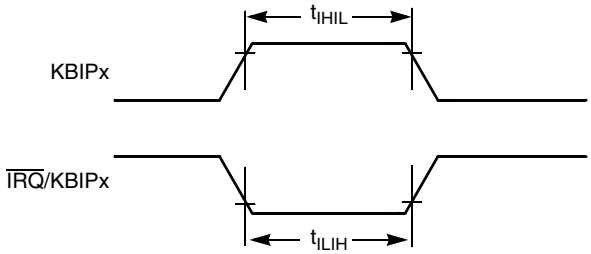



**Figure A-10. IRQ/KBIPx Timing**

## A.8.2 TPM/MTIM 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/MTIM Input Timing**

| Function | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| External clock frequency | $f_{TCLK}$ | 0 | $f_{Bus}/4$ | Hz |
| External clock period | $t_{TCLK}$ | 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 | — | $t_{cyc}$ |

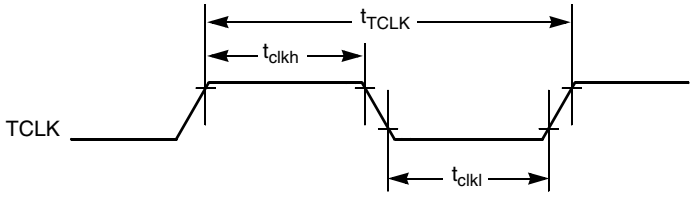



**Figure A-11. Timer External Clock**

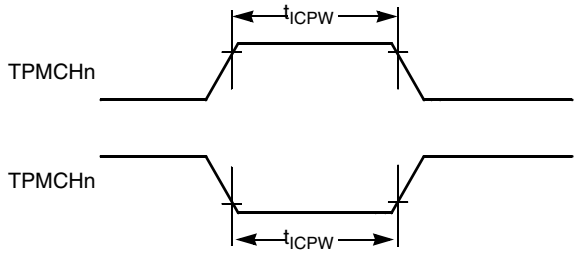



**Figure A-12. Timer Input Capture Pulse**

## A.8.3 SPI Timing

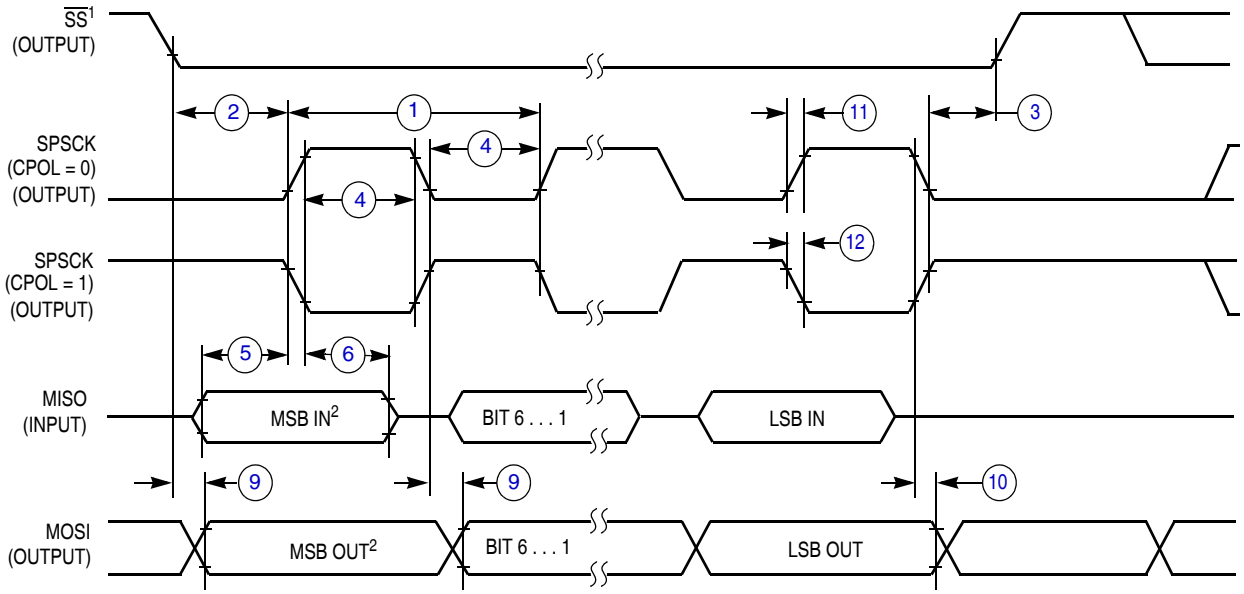
Table A-11 and Figure A-13 through Figure A-16 describe the timing requirements for the SPI system.

**Table A-11. SPI Timing**

| No. | Function | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| | Operating frequency<br>Master<br>Slave | $f_{op}$ | <br>$f_{Bus}/2048$<br>0 | <br>$f_{Bus}/2$<br>$f_{Bus}/4$ | Hz |
| 1 | SPSCK period<br>Master<br>Slave | $t_{SPSCK}$ | <br>2<br>4 | <br>2048<br>— | <br>$t_{cyc}$<br>$t_{cyc}$ |
| 2 | Enable lead time<br>Master<br>Slave | $t_{Lead}$ | <br>1/2<br>1 | <br>—<br>— | <br>$t_{SPSCK}$<br>$t_{cyc}$ |
| 3 | Enable lag time<br>Master<br>Slave | $t_{Lag}$ | <br>1/2<br>1 | <br>—<br>— | <br>$t_{SPSCK}$<br>$t_{cyc}$ |

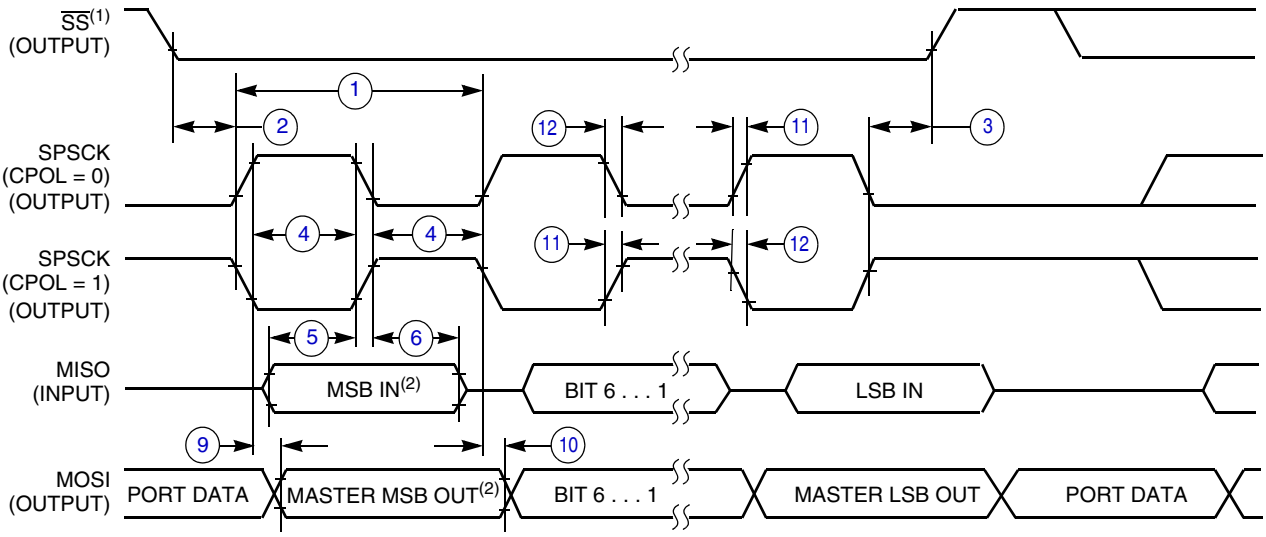
**Table A-11. SPI Timing (continued)**

| No. | Function | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 4 | Clock (SPSCK) high or low time<br>Master<br>Slave | $t_{WSPSCK}$ | <br>$t_{cyc} - 30$<br>$t_{cyc} - 30$ | <br>1024 $t_{cyc}$<br>— | <br>ns<br>ns |
| 5 | Data setup time (inputs)<br>Master<br>Slave | $t_{SU}$ | <br>15<br>15 | <br>—<br>— | <br>ns<br>ns |
| 6 | Data hold time (inputs)<br>Master<br>Slave | $t_{HI}$ | <br>0<br>25 | <br>—<br>— | <br>ns<br>ns |
| 7 | Slave access time | $t_a$ | — | 1 | $t_{cyc}$ |
| 8 | Slave MISO disable time | $t_{dis}$ | — | 1 | $t_{cyc}$ |
| 9 | Data valid (after SPSCK edge)<br>Master<br>Slave | $t_v$ | <br>—<br>— | <br>25<br>25 | <br>ns<br>ns |
| 10 | Data hold time (outputs)<br>Master<br>Slave | $t_{HO}$ | <br>0<br>0 | <br>—<br>— | <br>ns<br>ns |
| 11 | Rise time<br>Input<br>Output | <br>$t_{RI}$<br>$t_{RO}$ | <br>—<br>— | <br>$t_{cyc} - 25$<br>25 | <br>ns<br>ns |
| 12 | Fall time<br>Input<br>Output | <br>$t_{FI}$<br>$t_{FO}$ | <br>—<br>— | <br>$t_{cyc} - 25$<br>25 | <br>ns<br>ns |



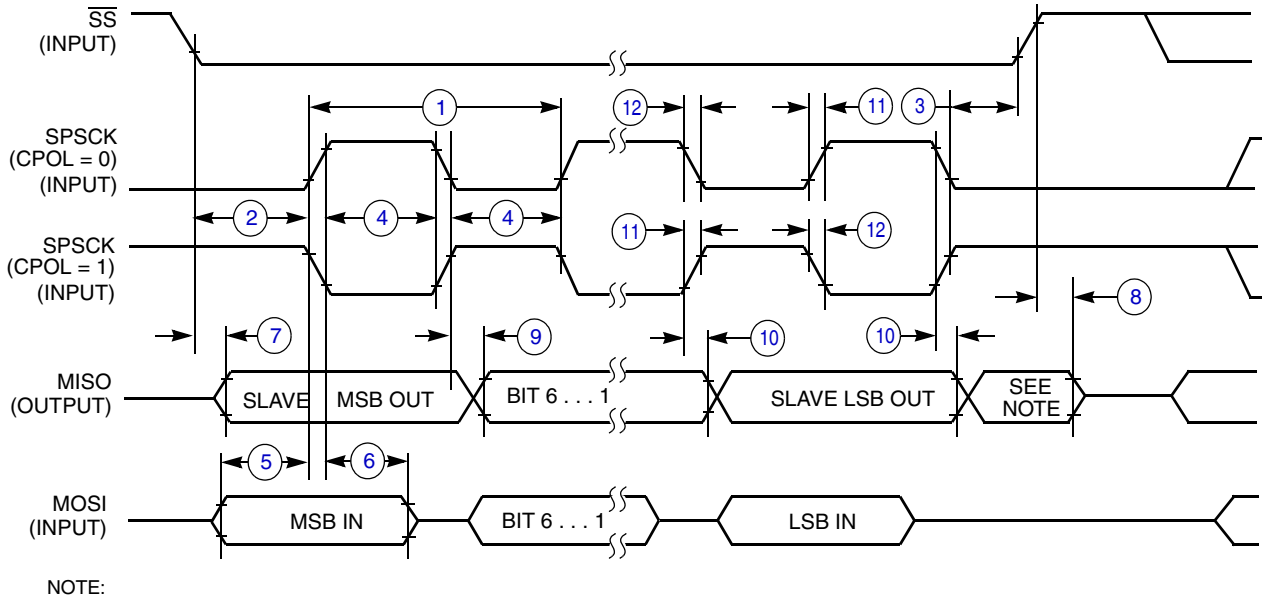

NOTES:
1. $\overline{SS}$ output mode (DDS7 = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

**Figure A-13. SPI Master Timing (CPHA = 0)**



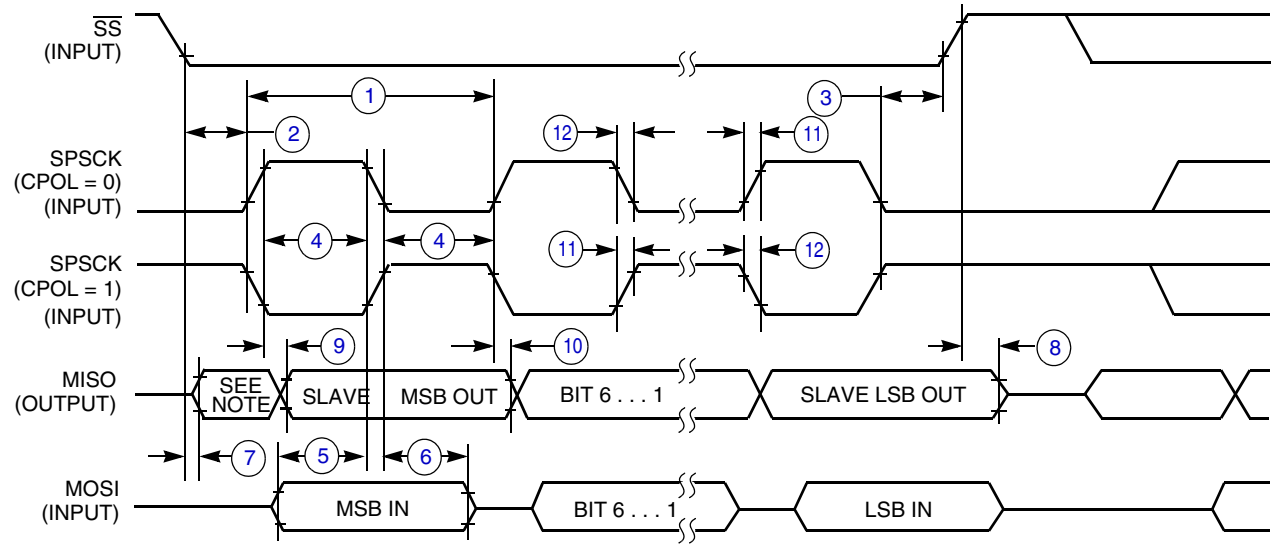

NOTES:
1. $\overline{SS}$ output mode (DDS7 = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

**Figure A-14. SPI Master Timing (CPHA =1)**




NOTE:
1. Not defined but normally MSB of character just received

**Figure A-15. SPI Slave Timing (CPHA = 0)**




NOTE:
1. Not defined but normally LSB of character just received

**Figure A-16. SPI Slave Timing (CPHA = 1)**

## A.9 Analog Comparator (ACMP) Electricals

**Table A-12. Analog Comparator Electrical Specifications**

| Characteristic | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | 1.80 | — | 3.6 | V |
| Supply current (active) | $I_{DDAC}$ | — | 20 | — | μA |
| Analog input voltage | $V_{AIN}$ | $V_{SS}$ – 0.3 | — | $V_{DD}$ | V |
| Analog input offset voltage | $V_{AIO}$ | | 20 | 40 | mV |
| Analog comparator hysteresis | $V_H$ | 3.0 | 9.0 | 15.0 | mV |
| Analog input leakage current | $I_{ALKG}$ | — | — | 1.0 | μA |
| Analog comparator initialization delay | $t_{AINIT}$ | — | — | 1.0 | μs |

## A.10 ADC Characteristics

**Table A-13. 3 Volt 10-bit ADC Operating Conditions**

| Characteristic | Conditions | Symbol | Min | Typical[1] | Max | Unit | Comment |
|---|---|---|---|---|---|---|---|
| Supply voltage | Absolute | $V_{DD}$ | 1.8 | — | 3.6 | V | |
| Input voltage | | $V_{ADIN}$ | $V_{SS}$ | — | $V_{DD}$ | V | |
| Input capacitance | | $C_{ADIN}$ | — | 4.5 | 5.5 | pF | |
| Input resistance | | $R_{ADIN}$ | — | 5 | 7 | kΩ | |
| Analog source resistance | 10 bit mode<br>$f_{ADCK}$ > 4MHz<br>$f_{ADCK}$ < 4MHz | $R_{AS}$ | <br>—<br>— | <br>—<br>— | <br>5<br>10 | kΩ | External to MCU |
| | 8 bit mode (all valid $f_{ADCK}$) | | — | — | 10 | | |
| ADC conversion clock frequency | High Speed (ADLPC=0) | $f_{ADCK}$ | 0.4 | — | 8.0 | MHz | |
| | Low Power (ADLPC=1) | | 0.4 | — | 4.0 | | |

[1] Typical values assume $V_{DD}$ = 3.0 V, Temp = 25°C, $f_{ADCK}$=1.0 MHz unless otherwise stated. Typical values are for reference only and are not tested in production.

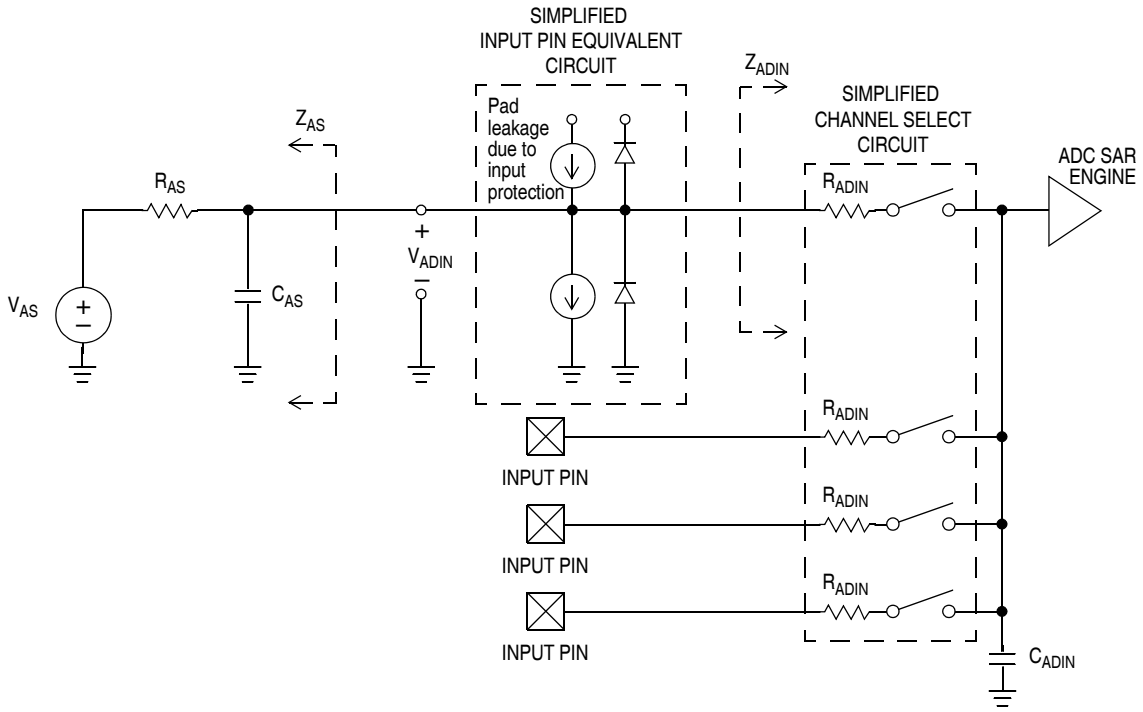



**Figure A-17. ADC Input Impedance Equivalency Diagram**

**Table A-14. 3 Volt 10-bit ADC Characteristics**

| Characteristic | Conditions | Symb | Min | Typ[1] | Max | Unit | Comment |
|---|---|---|---|---|---|---|---|
| Supply current<br>ADLPC=1<br>ADLSMP=1<br>ADCO=1 | | $I_{DDAD}$ | — | 120 | — | μA | |
| Supply current<br>ADLPC=1<br>ADLSMP=0<br>ADCO=1 | | $I_{DDAD}$ | — | 202 | — | μA | |
| Supply current<br>ADLPC=0<br>ADLSMP=1<br>ADCO=1 | | $I_{DDAD}$ | — | 288 | — | μA | |
| Supply current<br>ADLPC=0<br>ADLSMP=0<br>ADCO=1 | | $I_{DDAD}$ | — | 532 | 646 | μA | |
| ADC asynchronous clock source | High speed (ADLPC=0) | $f_{ADACK}$ | 2 | 3.3 | 5 | MHz | $t_{ADACK} = 1/f_{ADACK}$ |
| | Low power (ADLPC=1) | | 1.25 | 2 | 3.3 | | |

**Table A-14. 3 Volt 10-bit ADC Characteristics (continued)**

| Characteristic | Conditions | Symb | Min | Typ[1] | Max | Unit | Comment |
|---|---|---|---|---|---|---|---|
| Conversion time (including sample time) | Short sample (ADLSMP=0) | $t_{ADC}$ | — | 20 | — | ADCK cycles | See Table 9-12 for conversion time variances |
| | Long sample (ADLSMP=1) | | — | 40 | — | | |
| Sample time | Short sample (ADLSMP=0) | $t_{ADS}$ | — | 3.5 | — | ADCK cycles | |
| | Long sample (ADLSMP=1) | | — | 23.5 | — | | |
| Total unadjusted error | 10 bit mode | $E_{TUE}$ | — | ±1.5 | ±3.5 | LSB[2] | Includes quantization |
| | 8 bit mode | | — | ±0.7 | ±1.5 | | |
| Differential non-linearity | 10 bit mode | DNL | — | ±0.5 | ±1.0 | LSB[2] | Monotonicity and no missing codes guaranteed |
| | 8 bit mode | | — | ±0.3 | ±0.5 | | |
| Integral non-linearity | 10 bit mode | INL | — | ±0.5 | ±1.0 | LSB[2] | |
| | 8 bit mode | | — | ±0.3 | ±0.5 | | |
| Zero-scale error | 10 bit mode | $E_{ZS}$ | — | ±1.5 | ±2.1 | LSB[2] | $V_{ADIN} = V_{SS}$ |
| | 8 bit mode | | — | ±0.5 | ±0.7 | | |
| Full-scale error | 10 bit mode | $E_{FS}$ | 0 | ±1.0 | ±1.5 | LSB[2] | $V_{ADIN} = V_{DD}$ |
| | 8 bit mode | | 0 | ±0.5 | ±0.5 | | |
| Quantization error | 10 bit mode | $E_Q$ | — | — | ±0.5 | LSB[2] | |
| | 8 bit mode | | — | — | ±0.5 | | |
| Input leakage error | 10 bit mode | $E_{IL}$ | 0 | ±0.2 | ±4 | LSB[2] | Pad leakage[3] * $R_{AS}$ |
| | 8 bit mode | | 0 | ±0.1 | ±1.2 | | |
| Temp sensor slope | -40°C– 25°C | m | — | 1.646 | — | mV/°C | |
| | 25°C– 85°C | | — | 1.769 | — | | |
| Temp sensor voltage | 25°C | $V_{TEMP25}$ | — | 701.2 | — | mV | |

1 Typical values assume $V_{DD}$ = 3.0 V, Temp = 25°C, $f_{ADCK}$ = 1.0 MHz unless otherwise stated. Typical values are for reference only and are not tested in production.

2 1 LSB = $(V_{REFH} - V_{REFL})/2^N$

3 Based on input pad leakage current. Refer to pad electricals.

## A.11 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 section.

**Table A-15. FLASH Characteristics**

| Characteristic | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Supply voltage for program/erase:<br>T ≤ 85°C<br>T > 85 °C | $V_{prog/erase}$ | 1.8<br>2.1 | —<br>— | 3.6<br>3.6 | V |
| Supply voltage for read operation | $V_{Read}$ | 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 + 125°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 Motorola 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 Motorola defines typical data retention, please refer to Engineering Bulletin EB618/D, *Typical Data Retention for Nonvolatile Memory.*

## A.12 EMC Performance

Electromagnetic compatibility (EMC) performance is highly dependant on the environment in which the MCU resides. Board design and layout, circuit topology choices, location and characteristics of external components as well as MCU software operation all play a significant role in EMC performance. The system designer should consult Freescale applications notes such as AN2321, AN1050, AN1263, AN2764, and AN1259 for advice and guidance specifically targeted at optimizing EMC performance.

### A.12.1 Radiated Emissions

Microcontroller radiated RF emissions are measured from 150 kHz to 1 GHz using the TEM/GTEM Cell method in accordance with the IEC 61967-2 and SAE J1752/3 standards. The measurement is performed with the microcontroller installed on a custom EMC evaluation board while running specialized EMC test software. The radiated emissions from the microcontroller are measured in a TEM cell in two package orientations (North and East).

The maximum radiated RF emissions of the tested configuration in all orientations are less than or equal to the reported emissions levels.

**Table A-16. Radiated Emissions, Electric Field**

| Parameter | Symbol | Conditions | Frequency | $f_{OSC}/f_{BUS}$ | Level[1] (Max) | Unit |
|---|---|---|---|---|---|---|
| Radiated emissions, electric field | $V_{RE_TEM}$ | $V_{DD}$ = 3.3 V<br>$T_A$ = +25°C<br>package type 16 TSSOP | 0.15 – 50 MHz | 4-MHz crystal 10-MHz bus | TBD | dBμV |
| | | | 50 – 150 MHz | | TBD | |
| | | | 150 – 500 MHz | | TBD | |
| | | | 500 – 1000 MHz | | TBD | |
| | | | IEC Level | | TBD | — |
| | | | SAE Level | | TBD | — |

1 Data based on qualification test results.

### A.12.2 Conducted Transient Susceptibility

Microcontroller transient conducted susceptibility is measured in accordance with an internal Freescale test method. The measurement is performed with the microcontroller installed on a custom EMC evaluation board and running specialized EMC test software designed in compliance with the test method. The conducted susceptibility is determined by injecting the transient susceptibility signal on each pin of the microcontroller. The transient waveform and injection methodology is based on IEC 61000-4-4 (EFT/B). The transient voltage required to cause performance degradation on any pin in the tested configuration is greater than or equal to the reported levels unless otherwise indicated by footnotes below Table A-17.

**Table A-17. Conducted Susceptibility, EFT/B**

| Parameter | Symbol | Conditions | $f_{OSC}/f_{BUS}$ | Result | Amplitude[1] (Min) | Unit |
|---|---|---|---|---|---|---|
| Conducted susceptibility, electrical fast transient/burst (EFT/B) | $V_{CS_EFT}$ | $V_{DD}$ = 3.3V<br>$T_A$ = +25°C<br>package type TBD | TBD crystal<br>TBD bus | A | TBD | kV |
| | | | | B | TBD | |
| | | | | C | TBD | |
| | | | | D | TBD | |

1 Data based on qualification test results. Not tested in production.

The susceptibility performance classification is described in Table A-18.

**Table A-18. Susceptibility Performance Classification**

| Result | Performance Criteria | |
|---|---|---|
| A | No failure | The MCU performs as designed during and after exposure. |
| B | Self-recovering failure | The MCU does not perform as designed during exposure. The MCU returns automatically to normal operation after exposure is removed. |
| C | Soft failure | The MCU does not perform as designed during exposure. The MCU does not return to normal operation until exposure is removed and the RESET pin is asserted. |
| D | Hard failure | The MCU does not perform as designed during exposure. The MCU does not return to normal operation until exposure is removed and the power to the MCU is cycled. |
| E | Damage | The MCU does not perform as designed during and after exposure. The MCU cannot be returned to proper operation due to physical damage or other permanent performance degradation. |

# Appendix B
# Ordering Information and Mechanical Drawings

## B.1 Ordering Information

This section contains ordering information for MC9S08QG8 and MC9S08QG4 devices.

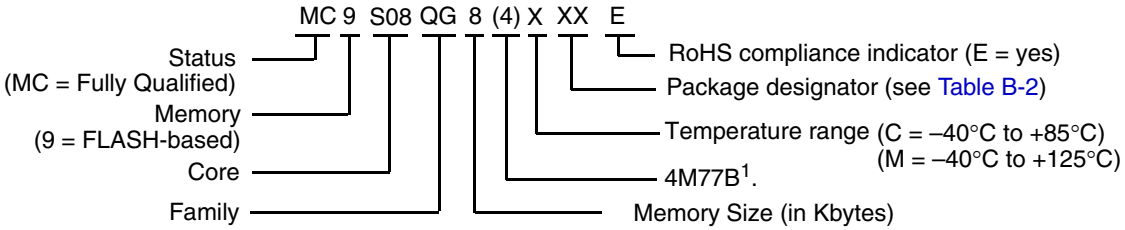
**Table B-1. Device Numbering System**

| Device Number[1] | Memory | | Available Packages[2] | | |
|---|---|---|---|---|---|
| | FLASH | RAM | 24-Pin | 16-Pin | 8-Pin |
| MC9S08QG8 | 8K | 512 | 24 QFN | 16 PDIP<br>16 QFN<br>16 TSSOP | 8 DFN<br>8 NB SOIC |
| MC9S08QG4 | 4K | 256 | 24 QFN | 16 QFN<br>16 TSSOP | 8 DFN<br>8 PDIP<br>8 NB SOIC |

1 See Table 1-1 for a complete description of modules included on each device.

2 See Table B-2 for package information.

### B.1.1 Device Numbering Scheme




1 Only maskset 4M77B has this additional number.

## B.2 Mechanical Drawings

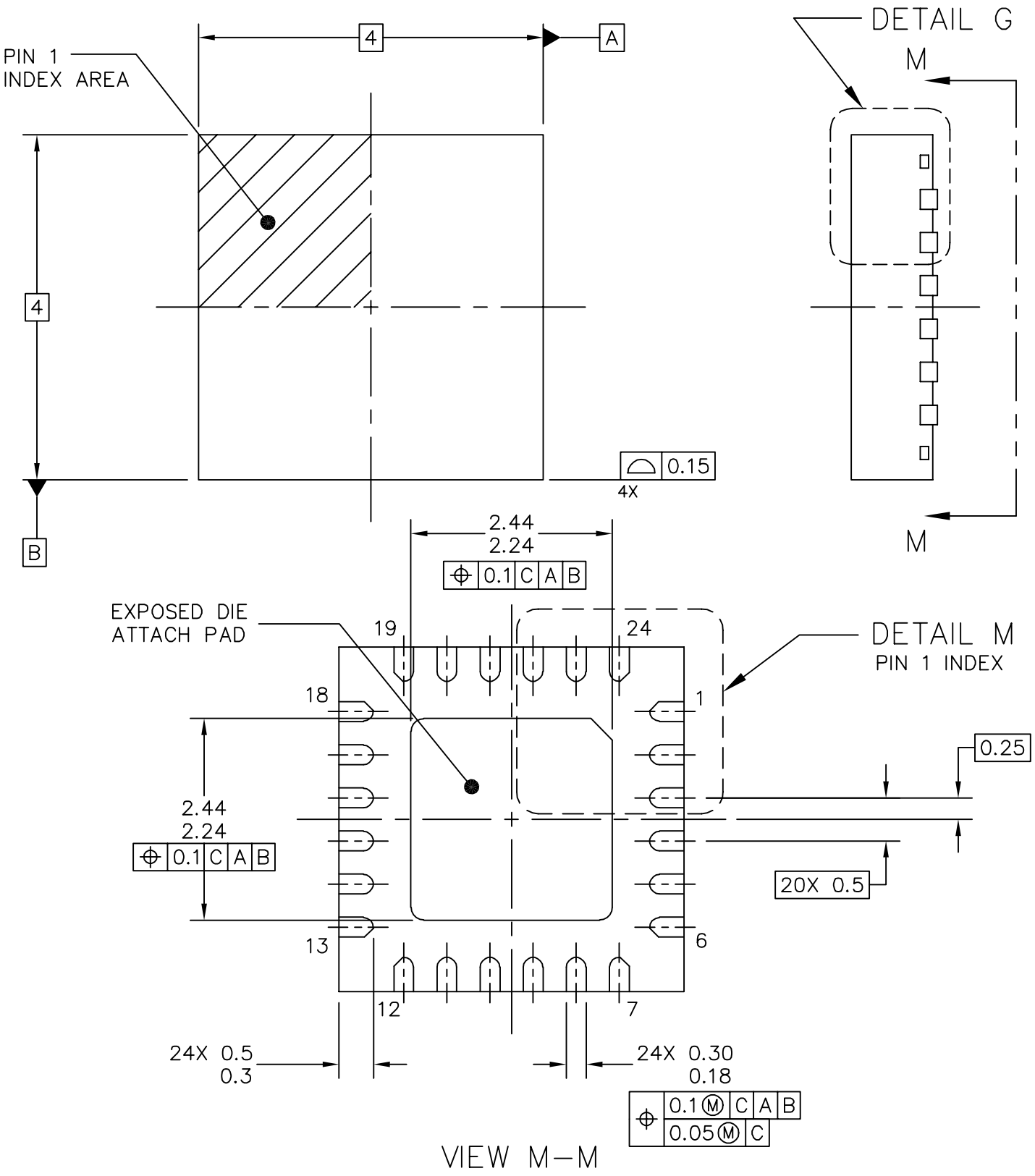
The following pages are mechanical specifications for MC9S08QG8/4 package options. See Table B-2 for the document number for each package type.

**Table B-2. Package Information**

| Pin Count | Type | Designator | Document No. |
|---|---|---|---|
| 24 | QFN | FK | 98ARL10605D |
| 16 | PDIP | PB | 98ASB42431B |
| 16 | QFN | FF | 98ARE10614D |
| 16 | TSSOP | DT | 98ASH70247A |

**Table B-2. Package Information (continued)**

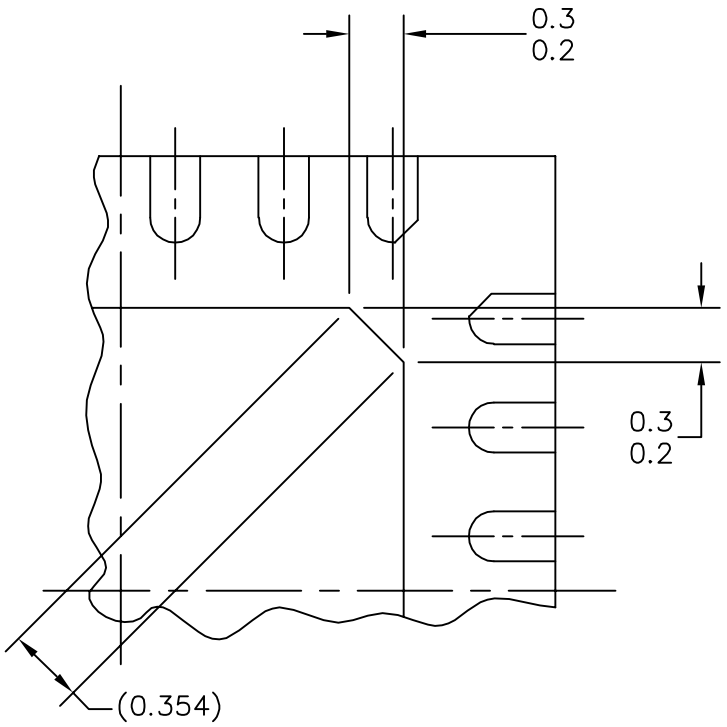
| Pin Count | Type | Designator | Document No. |
|---|---|---|---|
| 8 | DFN | FQ | 98ARL10557D |
| 8 | PDIP | PA | 98ASB42420B |
| 8 | NB SOIC | DN | 98ASB42564B |



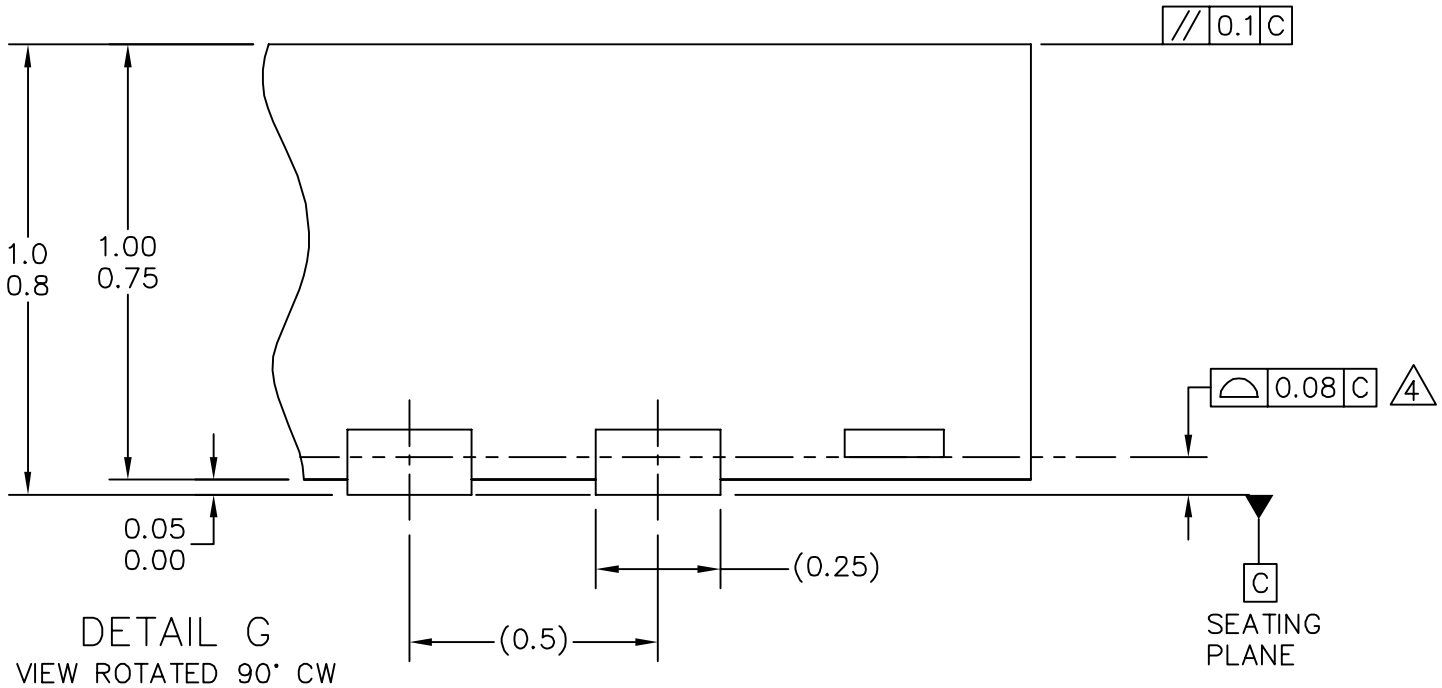

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: THERMALLY ENHANCED QUAD FLAT NON–LEADED PACKAGE (QFN) 24 TERMINAL, 0.5 PITCH (4 X 4 X 1) | DOCUMENT NO: 98ARL10605D | REV: O |
|---|---|---|
| | CASE NUMBER: 1897–01 | 08 SEP 2006 |
| | STANDARD: JEDEC MO–220 VGGD–8 | |




DETAIL M
PIN 1 BACKSIDE IDENTIFIER




DETAIL G
VIEW ROTATED 90° CW

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: THERMALLY ENHANCED QUAD FLAT NON-LEADED PACKAGE (QFN) 24 TERMINAL, 0.5 PITCH (4 X 4 X 1) | DOCUMENT NO: 98ARL10605D | REV: O |
|---|---|---|
| | CASE NUMBER: 1897-01 | 08 SEP 2006 |
| | STANDARD: JEDEC MO-220 VGGD-8 | |

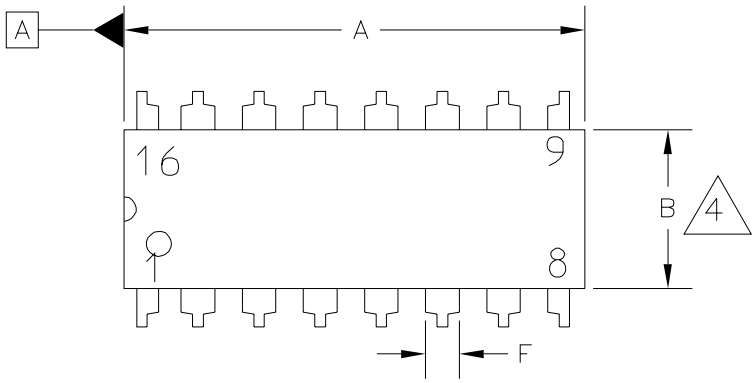
NOTES:

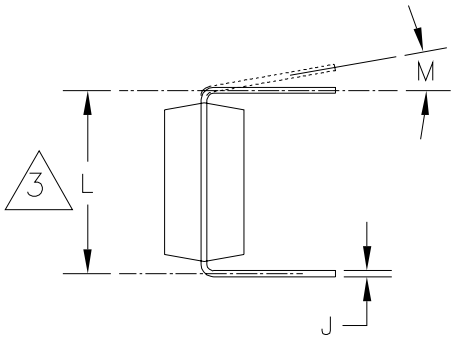
1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. DIMENSIONING AND TOLERANCING PER ASME Y14.5M–1994.
3. THE COMPLETE JEDEC DESIGNATOR FOR THIS PACKAGE IS: HF–PQFN.
4. COPLANARITY APPLIES TO LEADS AND DIE ATTACH PAD.
5. MIN. METAL GAP SHOULD BE 0.2 MM.

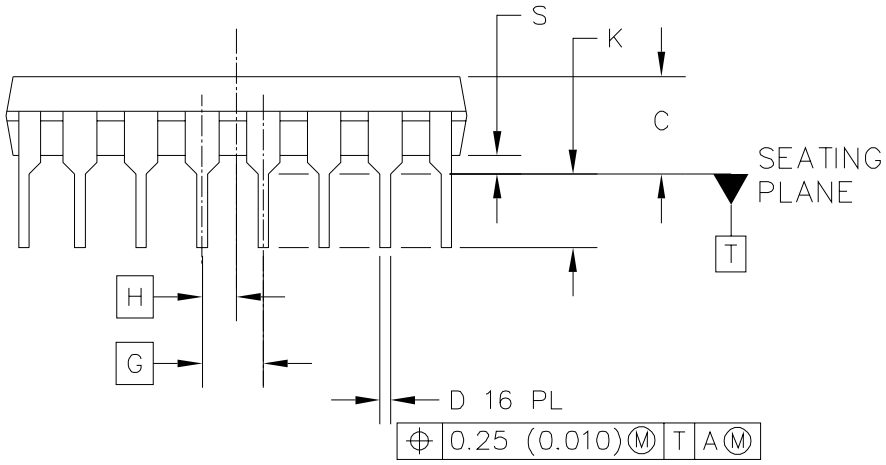
© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: THERMALLY ENHANCED QUAD FLAT NON–LEADED PACKAGE (QFN) 24 TERMINAL, 0.5 PITCH (4 X 4 X 1) | DOCUMENT NO: 98ARL10605D | REV: O |
|---|---|---|
| | CASE NUMBER: 1897–01 | 08 SEP 2006 |
| | STANDARD: JEDEC MO–220 VGGD–8 | |




© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASB42431B | REV: T |
|---|---|---|
| 16 LD PDIP | CASE NUMBER: 648-08 | 19 MAY 2005 |
| | STANDARD: NON-JEDEC | |

NOTES:

1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION TO CENTER OF LEADS WHEN FORMED PARALLEL.
4. DIMENSIONS DOES NOT INCLUDE MOLD FLASH.
5. ROUNDED CORNERS OPTIONAL.
6. 648-01 THRU -08 OBSOLETE, NEW STANDARD 648-09.

| DIM | MILLIMETERS MIN | MILLIMETERS MAX | INCHES MIN | INCHES MAX | DIM | MILLIMETERS MIN | MILLIMETERS MAX | INCHES MIN | INCHES MAX |
|---|---|---|---|---|---|---|---|---|---|
| A | 18.80 | 19.55 | 0.740 | 0.770 | | | | | |
| B | 6.35 | 6.85 | 0.250 | 0.270 | | | | | |
| C | 3.69 | 4.44 | 0.145 | 0.175 | | | | | |
| D | 0.39 | 0.53 | 0.015 | 0.021 | | | | | |
| F | 1.02 | 1.77 | 0.040 | 0.070 | | | | | |
| G | 2.54 BSC | | 0.100 BSC | | | | | | |
| H | 1.27 BSC | | 0.050 BSC | | | | | | |
| J | 0.21 | 0.38 | 0.008 | 0.015 | | | | | |
| K | 2.80 | 3.30 | 0.110 | 0.130 | | | | | |
| L | 7.50 | 7.74 | 0.295 | 0.305 | | | | | |
| M | 0° | 10° | 0° | 10° | | | | | |
| S | 0.51 | 1.01 | 0.020 | 0.040 | | | | | |

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE

TITLE:

16 LD PDIP

DOCUMENT NO: 98ASB42431B | REV: T

CASE NUMBER: 648-08 | 19 MAY 2005

STANDARD: NON-JEDEC

STYLE 1:

PIN 1. CATHODE
2. CATHODE
3. CATHODE
4. CATHODE
5. CATHODE
6. CATHODE
7. CATHODE
8. CATHODE
9. ANODE
10. ANODE
11. ANODE
12. ANODE
13. ANODE
14. ANODE
15. ANODE
16. ANODE

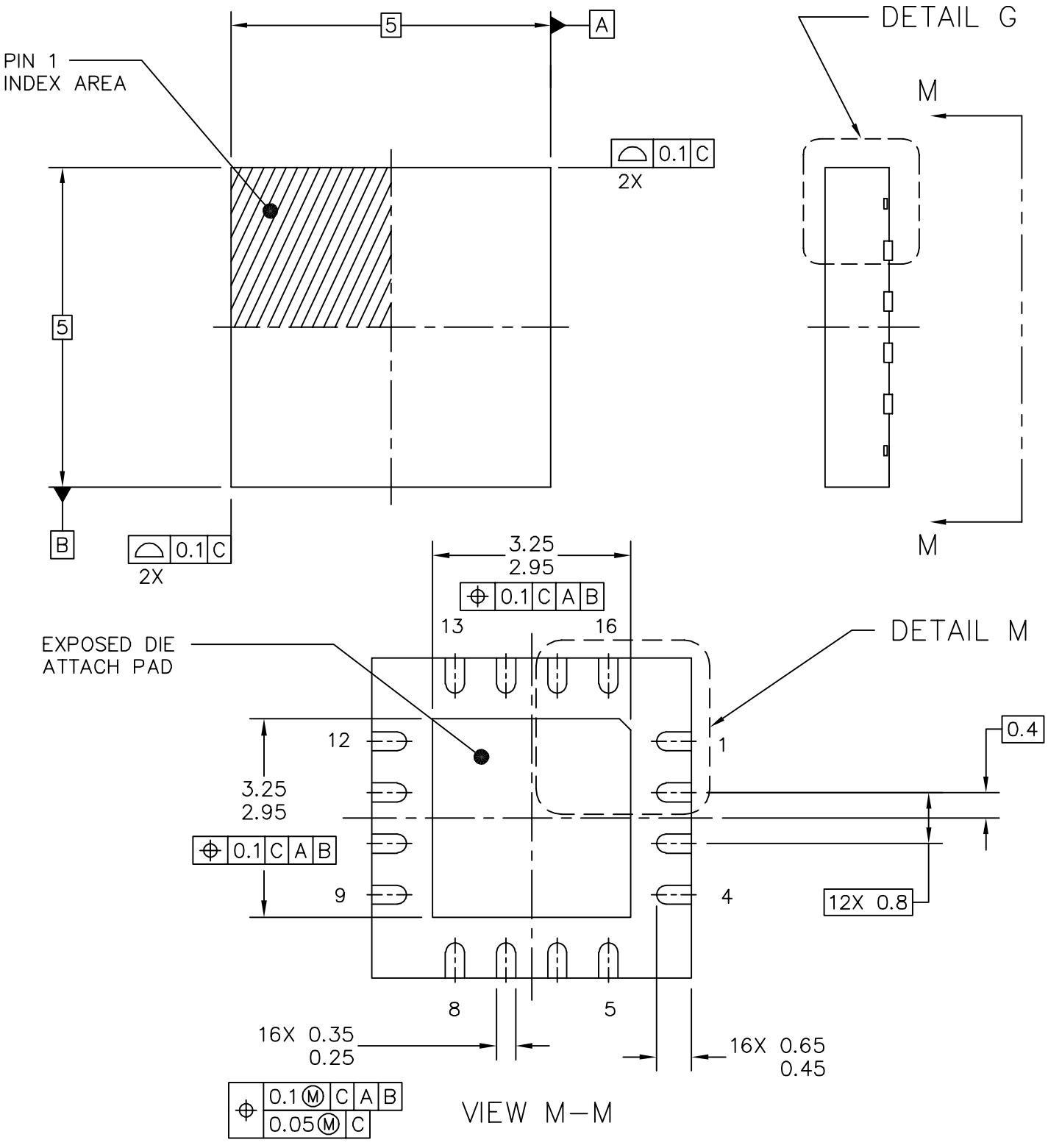
STYLE 2:

PIN 1. COMMON DRAIN
2. COMMON DRAIN
3. COMMON DRAIN
4. COMMON DRAIN
5. COMMON DRAIN
6. COMMON DRAIN
7. COMMON DRAIN
8. COMMON DRAIN
9. GATE
10. SOURCE
11. GATE
12. SOURCE
13. GATE
14. SOURCE
15. GATE
16. SOURCE

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

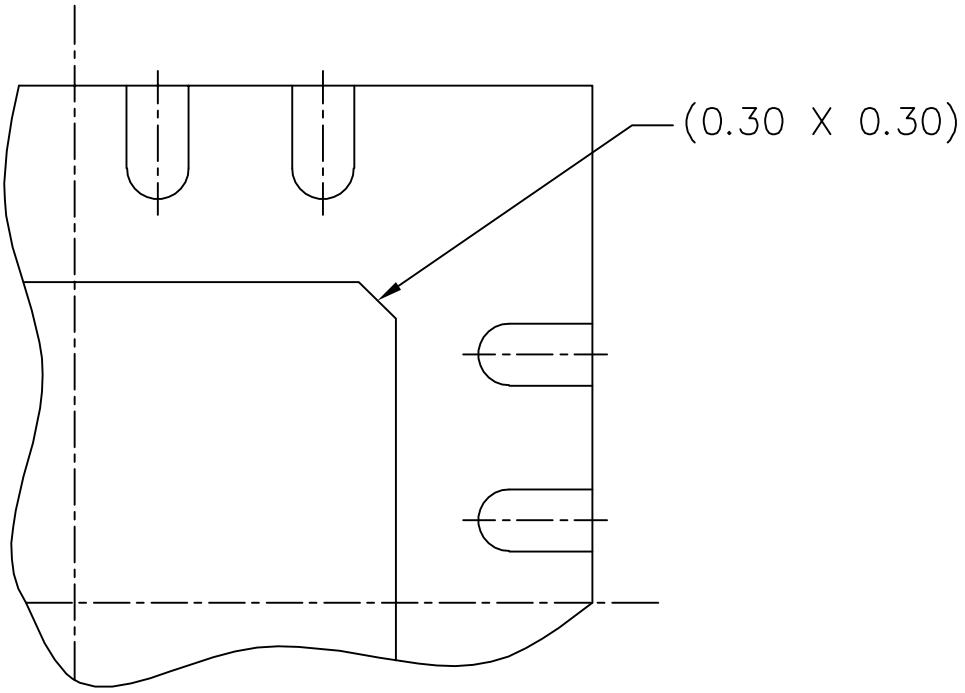
| TITLE: | DOCUMENT NO: 98ASB42431B | REV: T |
|---|---|---|
| 16 LD PDIP | CASE NUMBER: 648-08 | 19 MAY 2005 |
| | STANDARD: NON-JEDEC | |



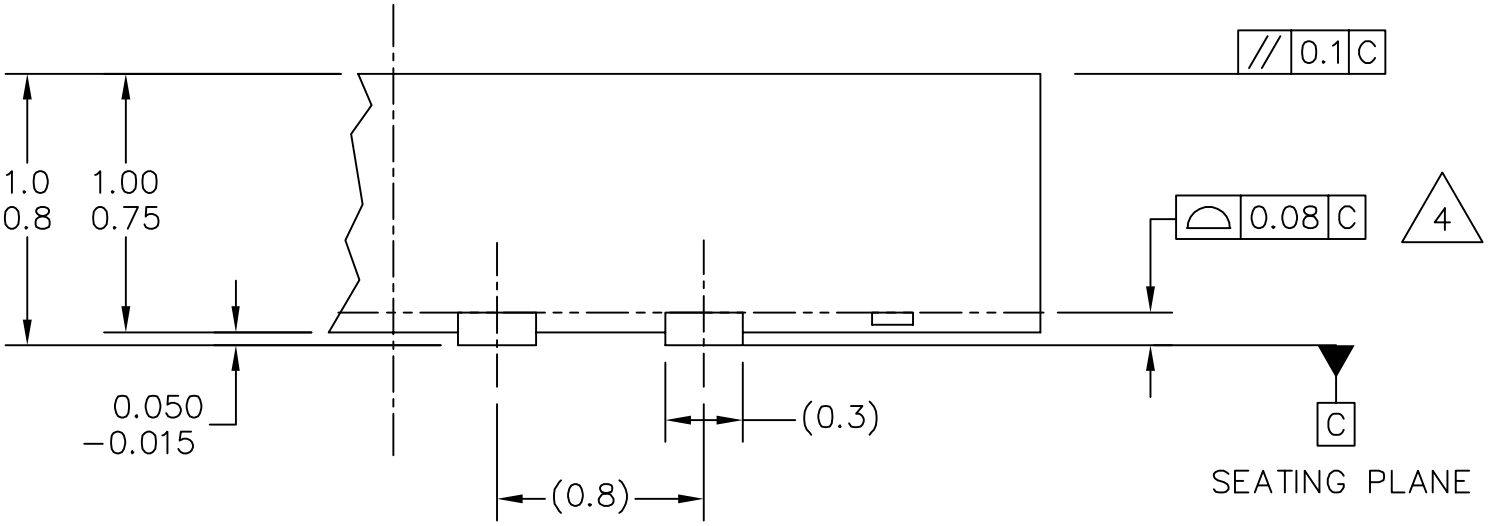

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: THERMALLY ENHANCED QUAD FLAT NON–LEADED PACKAGE (QFN) 16 TERMINAL, 0.8 PITCH (5 X 5 X 1) CASE OUTLINE | DOCUMENT NO: 98ARE10614D | REV: O |
|---|---|---|
| | CASE NUMBER: 1679–01 | 23 MAR 2005 |
| | STANDARD: FREESCALE | |




DETAIL M
PIN 1 BACKSIDE IDENTIFIER




DETAIL G
VIEW ROTATED 90° CW

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: THERMALLY ENHANCED QUAD FLAT NON−LEADED PACKAGE (QFN) 16 TERMINAL, 0.8 PITCH (5 X 5 X 1) CASE OUTLINE | DOCUMENT NO: 98ARE10614D | REV: O |
|---|---|---|
| | CASE NUMBER: 1679−01 | 23 MAR 2005 |
| | STANDARD: FREESCALE | |

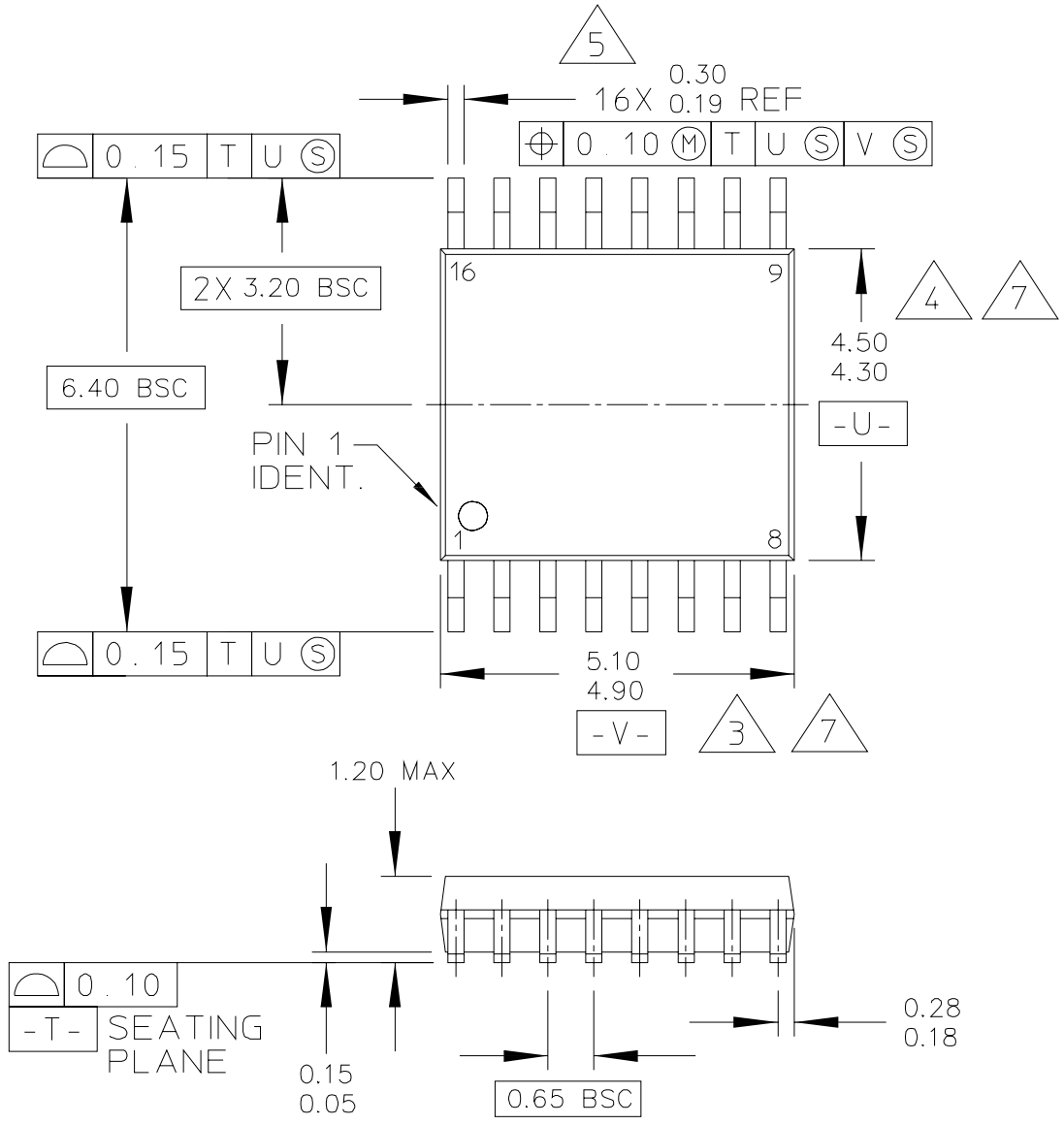
NOTES:

1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. DIMENSIONING AND TOLERANCING PER ASME Y14.5M–1994.
3. THE COMPLETE JEDEC DESIGNATOR FOR THIS PACKAGE IS: HF–PQFN.
4. COPLANARITY APPLIES TO LEADS, CORNER LEADS, AND DIE ATTACH PAD.
5. MIN METAL GAP SHOULD BE 0.2MM.

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

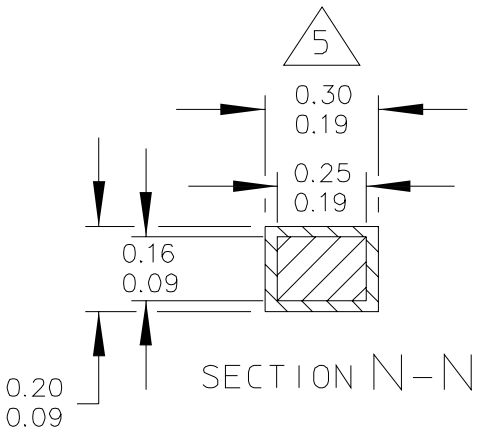
| TITLE: THERMALLY ENHANCED QUAD FLAT NON–LEADED PACKAGE (QFN) 16 TERMINAL, 0.8 PITCH (5 X 5 X 1) CASE OUTLINE | DOCUMENT NO: 98ARE10614D | REV: O |
|---|---|---|
| | CASE NUMBER: 1679–01 | 23 MAR 2005 |
| | STANDARD: FREESCALE | |

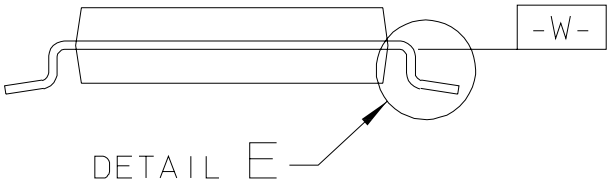


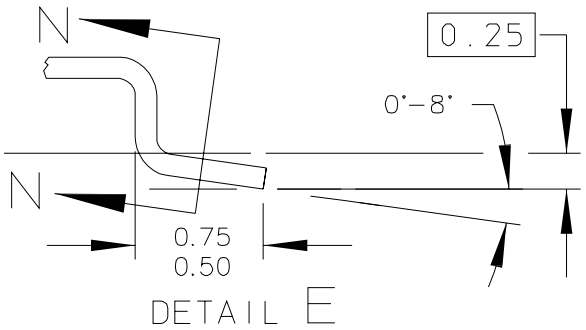

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASH70247A | REV: B |
|---|---|---|
| 16 LD TSSOP, PITCH 0.65MM | CASE NUMBER: 948F-01 | 19 MAY 2005 |
| | STANDARD: JEDEC | |




© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASH70247A | REV: B |
|---|---|---|
| 16 LD TSSOP, PITCH 0.65MM | CASE NUMBER: 948F-01 | 19 MAY 2005 |
| | STANDARD: JEDEC | |

NOTES:

1. CONTROLLING DIMENSION: MILLIMETER

2. DIMENSIONS AND TOLERANCES PER ANSI Y14.5M-1982.

3. DIMENSION DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS. MOLD FLASH OR GATE BURRS SHALL NOT EXCEED 0.15 PER SIDE.

4. DIMENSION DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION. INTERLEAD FLASH OR PROTRUSION SHALL NOT EXCEED 0.25 PER SIDE.

5. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 TOTAL IN EXCESS OF THE DIMENSION AT MAXIMUM MATERIAL CONDITION.

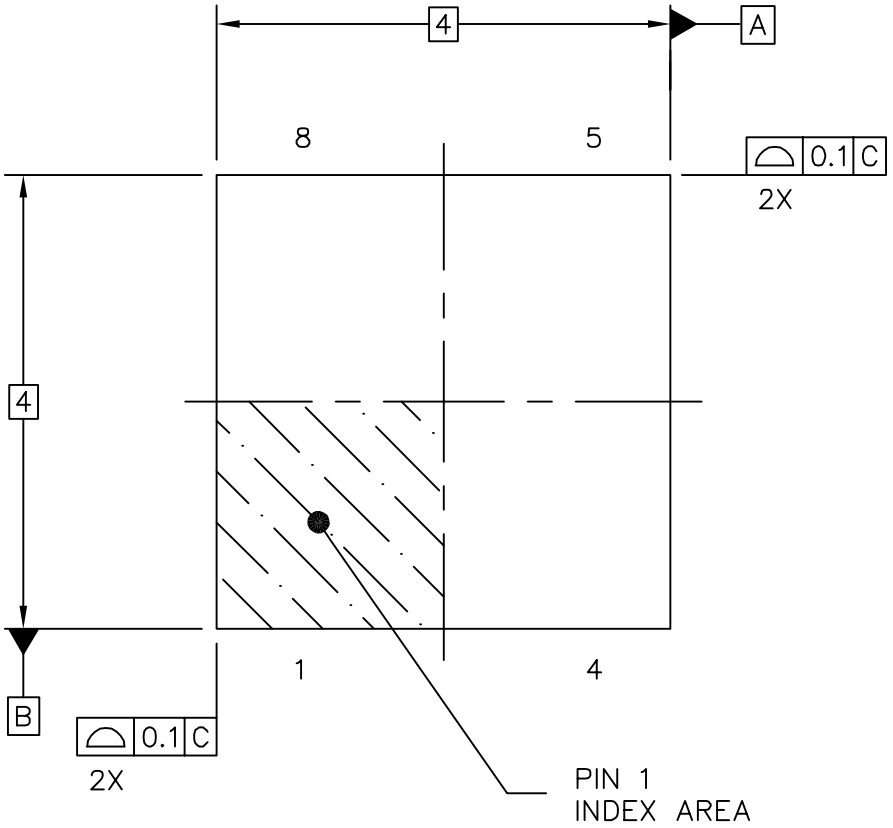
6. TERMINAL NUMBERS ARE SHOWN FOR REFERENCE ONLY.

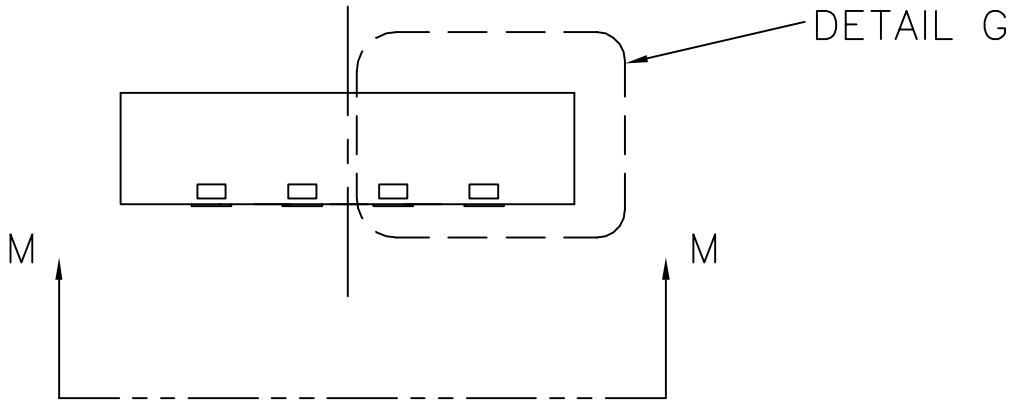
7. DIMENSIONS ARE TO BE DETERMINED AT DATUM PLANE -W-.

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASH70247A | REV: B |
|---|---|---|
| 16 LD TSSOP, PITCH 0.65MM | CASE NUMBER: 948F-01 | 19 MAY 2005 |
| | STANDARD: JEDEC | |

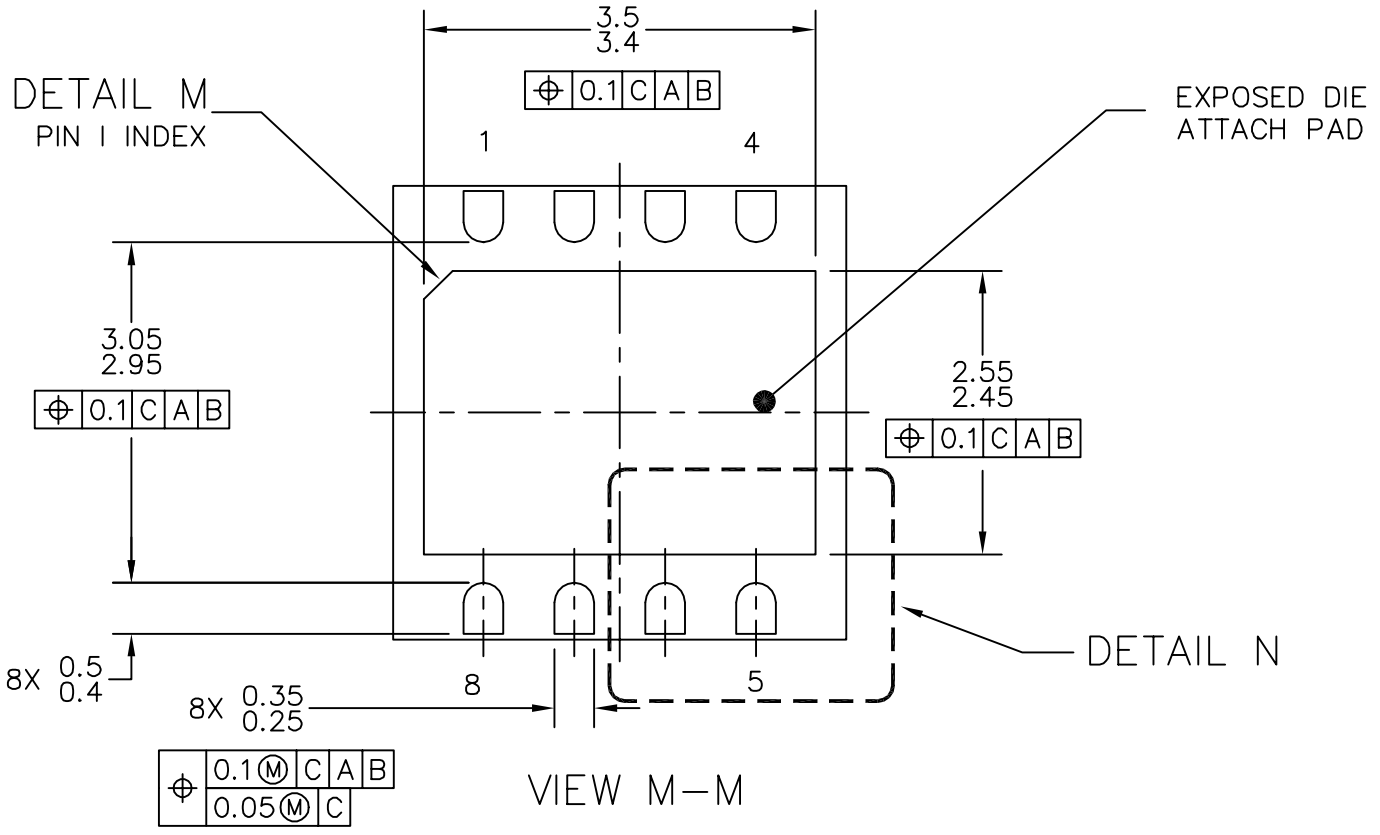


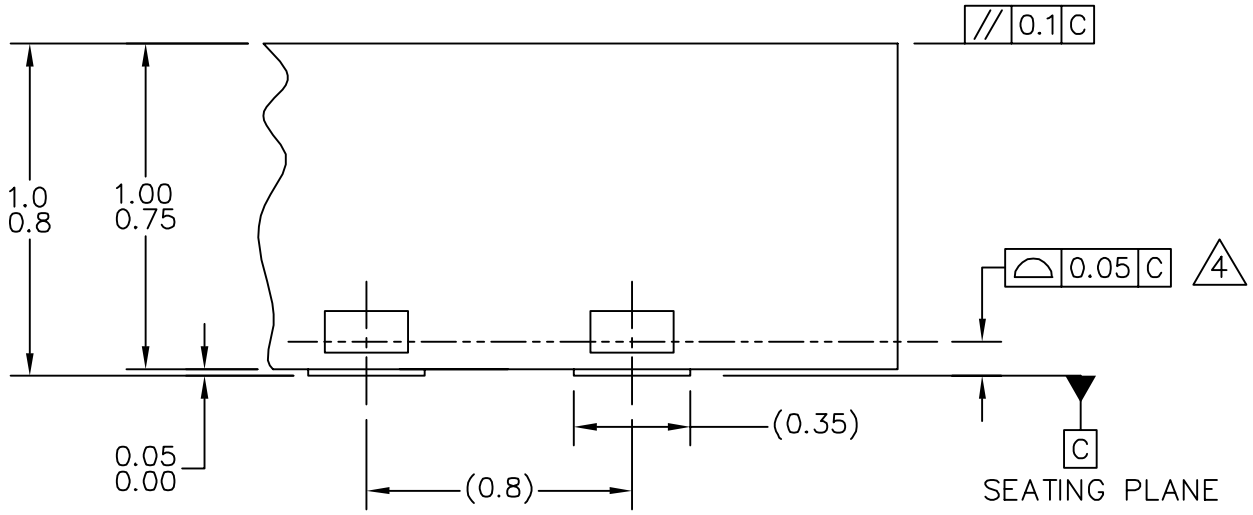




© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: THERMALLY ENHANCED DUAL FLAT NO LEAD PACKAGE (DFN) 8 TERMINAL, 0.8 PITCH (4 X 4 X 1) | DOCUMENT NO: 98ARL10557D | REV: B |
|---|---|---|
| | CASE NUMBER: 1452-02 | 28 DEC 2005 |
| | STANDARD: NON-JEDEC | |



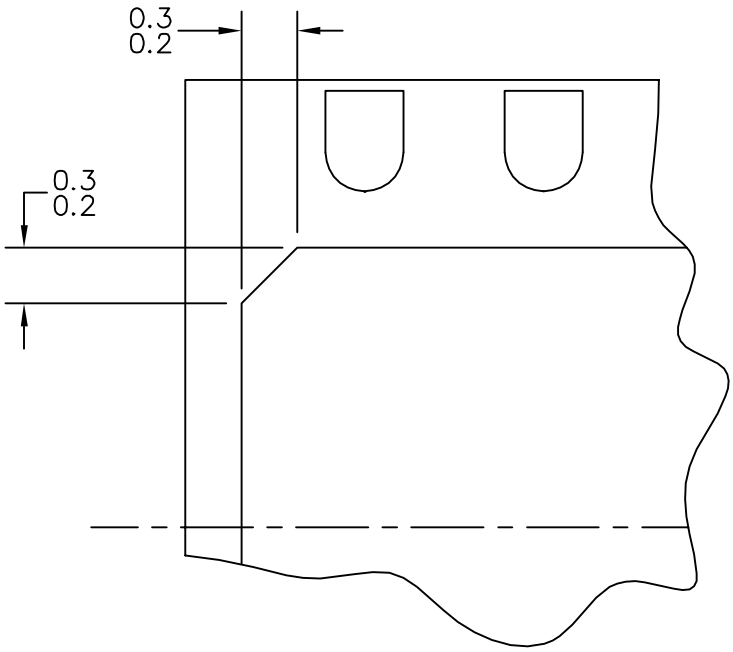

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

DETAIL G

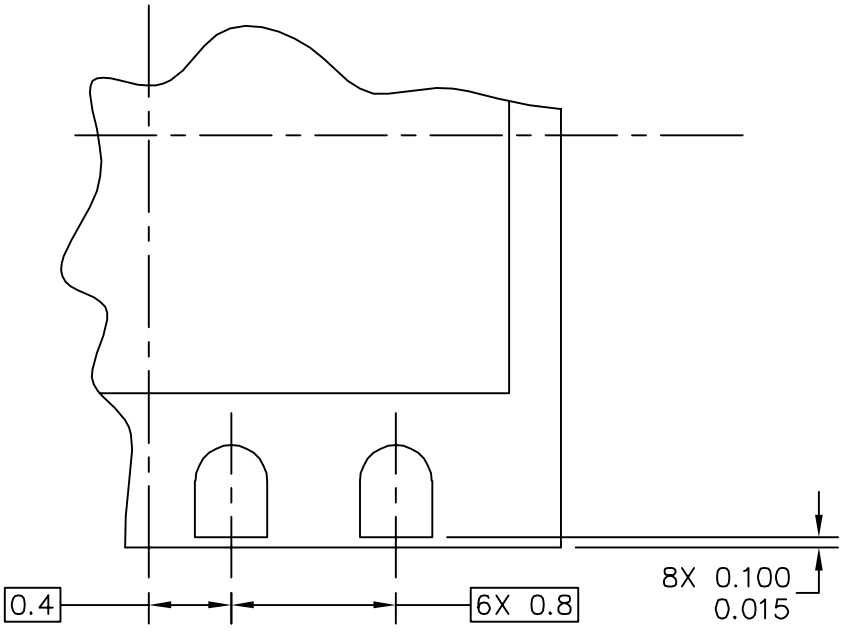
**MECHANICAL OUTLINE**

PRINT VERSION NOT TO SCALE

| | | |
|---|---|---|
| TITLE: THERMALLY ENHANCED DUAL FLAT NO LEAD PACKAGE (DFN) 8 TERMINAL, 0.8 PITCH (4 X 4 X 1) | DOCUMENT NO: 98ARL10557D | REV: B |
| | CASE NUMBER: 1452–02 | 28 DEC 2005 |
| | STANDARD: NON–JEDEC | |




DETAIL M
BACKSIDE PIN 1 INDEX




DETAIL N

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: THERMALLY ENHANCED DUAL FLAT NO LEAD PACKAGE (DFN) 8 TERMINAL, 0.8 PITCH (4 X 4 X 1) | DOCUMENT NO: 98ARL10557D | REV: B |
|---|---|---|
| | CASE NUMBER: 1452–02 | 28 DEC 2005 |
| | STANDARD: NON–JEDEC | |

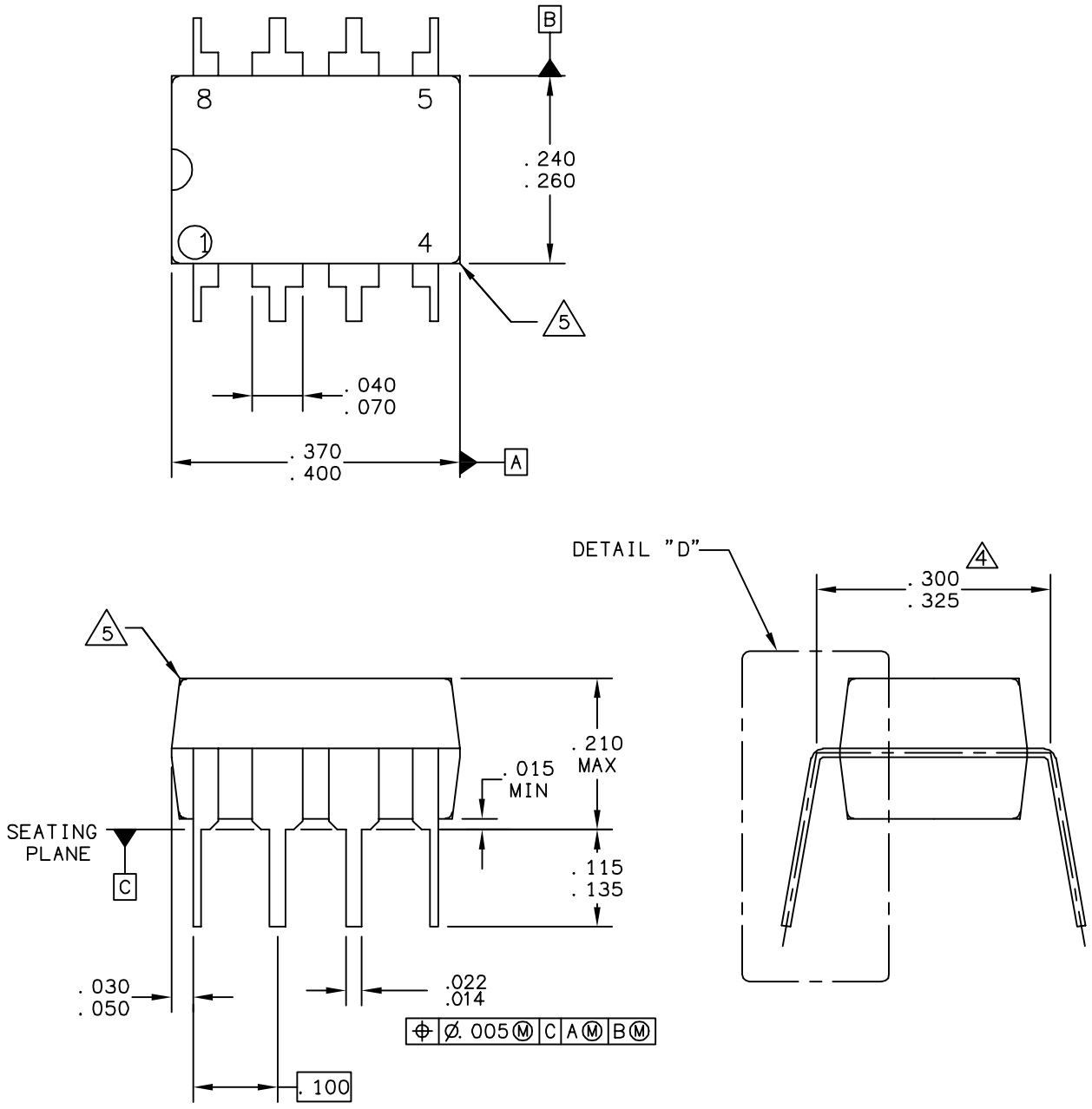
NOTES:

1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5M-1994.
3. THE COMPLETE JEDEC DESIGNATOR FOR THIS PACKAGE IS: HP-VFDFP-N.
4. COPLANARITY APPLIES TO LEADS AND DIE ATTACH PAD.
5. MIN. METAL GAP SHOULD BE 0.2MM.

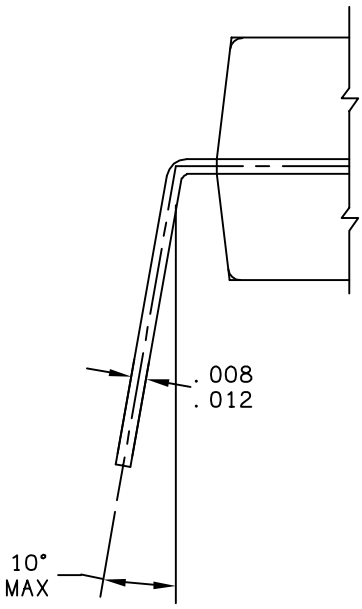
© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: THERMALLY ENHANCED DUAL FLAT NO LEAD PACKAGE (DFN) 8 TERMINAL, 0.8 PITCH (4 X 4 X 1) | DOCUMENT NO: 98ARL10557D | REV: B |
|---|---|---|
| | CASE NUMBER: 1452-02 | 28 DEC 2005 |
| | STANDARD: NON-JEDEC | |




| | | |
|---|---|---|
| © FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED. | **MECHANICAL OUTLINE** | PRINT VERSION NOT TO SCALE |
| TITLE: 8 LD PDIP | DOCUMENT NO: 98ASB42420B | REV: N |
| | CASE NUMBER: 626-06 | 19 MAY 2005 |
| | STANDARD: NON-JEDEC | |




DETAIL "D"

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASB42420B | REV: N |
|---|---|---|
| 8 LD PDIP | CASE NUMBER: 626-06 | 19 MAY 2005 |
| | STANDARD: NON-JEDEC | |

NOTES:

1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M – 1994.
2. ALL DIMENSIONS ARE IN INCHES.
3. 626-03 TO 626-06 OBSOLETE. NEW STANDARD 626-07.
4. DIMENSION TO CENTER OF LEAD WHEN FORMED PARALLEL.
5. PACKAGE CONTOUR OPTIONAL (ROUND OR SQUARE CONERS).

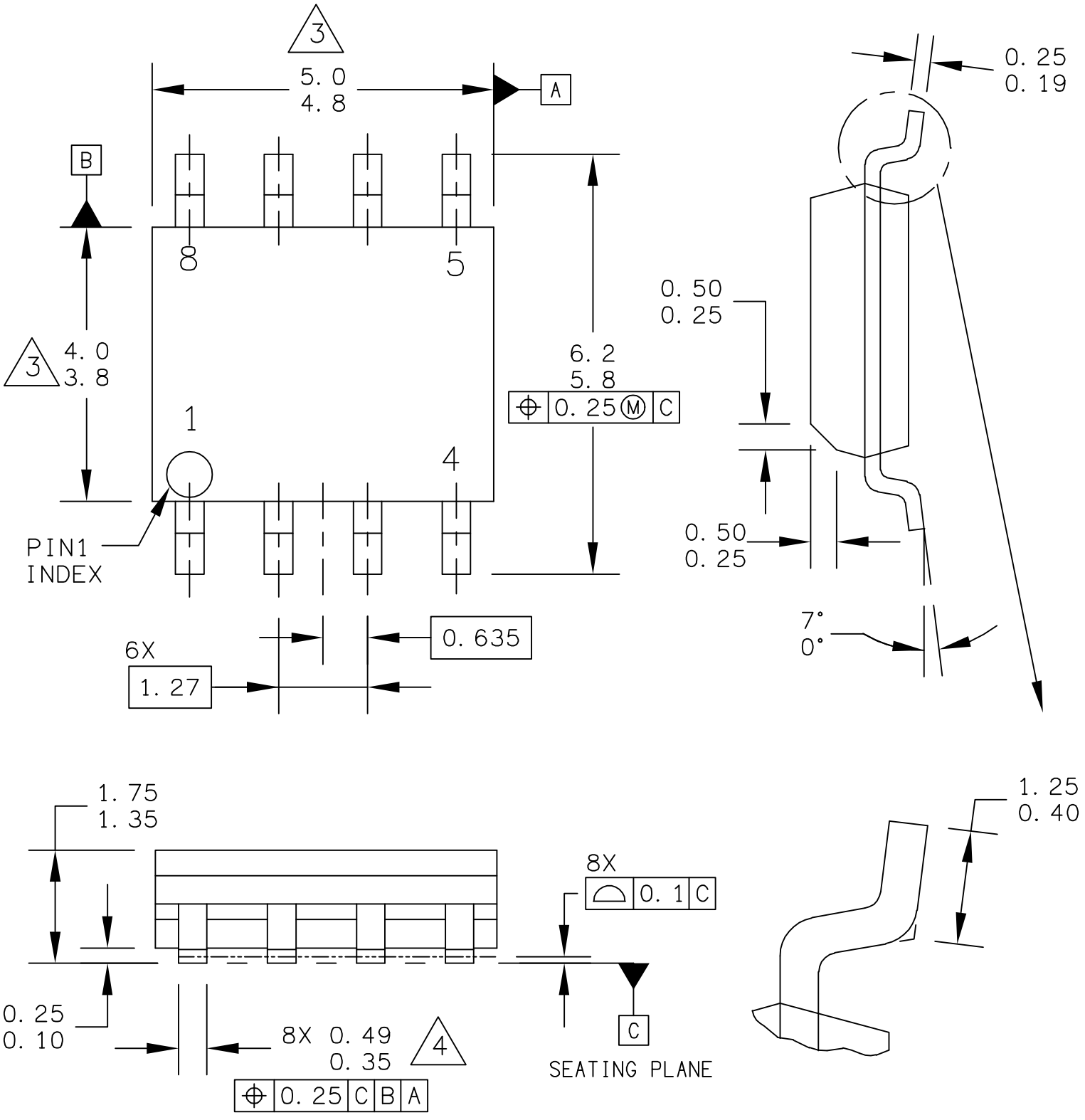
STYLE 1:

| | | | | |
|---|---|---|---|---|
| PIN | 1. | AC IN | 5. | GROUND |
| | 2. | DC + IN | 6. | OUTPUT |
| | 3. | DC – IN | 7. | AUXILIARY |
| | 4. | AC IN | 8. | VCC |

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASB42420B | REV: N |
|---|---|---|
| 8 LD PDIP | CASE NUMBER: 626-06 | 19 MAY 2005 |
| | STANDARD: NON-JEDEC | |




© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASB42564B | REV: V |
|---|---|---|
| 8LD SOIC NARROW BODY | CASE NUMBER: 751-07 | 20 NOV 2007 |
| | STANDARD: JEDEC MS-012AA | |

NOTES:

1. DIMENSIONS ARE IN MILLIMETERS.

2. DIMENSIONING AND TOLERANCING PER ASME Y14.5M–1994.

3. DIMENSION DOES NOT INCLUDE MOLD PROTRUSION. MAXIMUM MOLD PROTRUSION 0.15 PER SIDE.

4. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.127 TOTAL IN EXCESS OF THE DIMENSION AT MAXIMUM MATERIAL CONDITION.

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASB42564B | REV: V |
|---|---|---|
| 8LD SOIC NARROW BODY | CASE NUMBER: 751–07 | 20 NOV 2007 |
| | STANDARD: JEDEC MS–012AA | |

## *How to Reach Us:*


**Home Page:**
www.freescale.com

**Web Support:**
http://www.freescale.com/support

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

***For Literature Requests Only:***
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com



Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2007-2009. All rights reserved.



MC9S08QG8
Rev. 5, 11/2009


freescale™
semiconductor

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View MC9S08QG8CFFE 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