# THE DATASHEET OF

# MC68HC908GZ60CFA

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

# MC68HC908GZ60
# MC68HC908GZ48
# MC68HC908GZ32

Data Sheet

***M68HC08***
***Microcontrollers***

MC68HC908GZ60
Rev. 6.0
04/2007

*freescale.com*

freescale™
semiconductor

# MC68HC908GZ60
# MC68HC908GZ48
# MC68HC908GZ32

**Data Sheet**

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/

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.
This product incorporates SuperFlash® technology licensed from SST.

© Freescale Semiconductor, Inc., 2005, 2006, 2007. All rights reserved.

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

## Revision History

| Date | Revision Level | Description | Page Number(s) |
|---|---|---|---|
| April, 2004 | N/A | Initial release | N/A |
| May, 2004 | 1.0 | 9.7.3 Keyboard Interrupt Polarity Register — Corrected the bit description of the KBIP7–KBIP0 bits. | 119 |
| | | 14.8.8 ESCI Prescaler Register — Reworked note under PDS2–PDS0 description for clarity. | 212 |
| | | Table 22-1. MC Order Numbers — Corrected order numbers. | 329 |
| | | Figure 22-1. Device Numbering System — Reworked diagram to reflect correct order numbers. | 329 |
| | | Table A-1. MC Order Numbers — Corrected order numbers. | 342 |
| | | Figure A-3. Device Numbering System — Reworked diagram to reflect correct order numbers. | 342 |
| | | B.4 Ordering Information — Corrected order numbers. | 346 |
| | | Figure B-3. Device Numbering System — Reworked diagram to reflect correct order numbers. | 346 |
| June, 2005 | 2.0 | Reformatted to Freescale publication standards | Throughout |
| | | Table 14-6. ESCI LIN Control Bits — Corrected Functionality entries | 211 |
| | | 14.9.1 ESCI Arbiter Control Register — Corrected bit ACLK bit description | 215 |
| | | 14.9.3 Bit Time Measurement — Corrected definition for ACLK bit | 216 |
| March, 2006 | 3.0 | 10.5 Clock Generator Module (CGM) — Updated description to remove erroneous information. | 122 |
| July, 2006 | 4.0 | Added section 1.5.15 Unused Pin Termination | 31 |
| | | Chapter 13 Input/Output (I/O) Ports — Replaced note | 169 |
| | | Table 14-6. ESCI LIN Control Bits — Updated functionality column. | 213 |
| | | 18.6 TIM1 During Break Interrupts — Updated first paragraph for clarity. | 270 |
| | | 19.6 TIM2 During Break Interrupts — Updated first paragraph for clarity. | 290 |
| | | 20.2.1.2 TIM During Break Interrupts — Updated first paragraph for clarity. | 302 |
| | | Figure 20-10. Normal Monitor Mode Circuit and Figure 20-11. Forced Monitor Mode — Changed capacitor values | 307 |
| | | 21.5 5.0-Vdc Electrical Characteristics — Updated minimum value for low-voltage inhibit, trip rising voltage (VTRIPR). | 317 |
| | | 21.9.2 CGM Component Information — Updated values for feedback bias resistor | 322 |

## Revision History (Continued)

| Date | Revision Level | Description | Page Number(s) |
|---|---|---|---|
| October, 2006 | 5.0 | 12.2 Features — Corrected timer link connection from TIM2 channel 0 to TIM1 channel 0. | 135 |
| | | 12.9 Timer Link — Corrected timer link connection from TIM2 channel 0 to TIM1 channel 0. | 147 |
| | | 21.5 5.0-Vdc Electrical Characteristics and<br>21.6 3.3-Vdc Electrical Characteristics — Updated DC injection current specification. | 317<br>319 |
| April, 2007 | 6.0 | Figure 2-2. Control, Status, and Data Registers — Changed TBMCLKSEL to TMBCLKSEL to be compatible with development tool nomenclature | 37 |
| | | Chapter 5 Configuration Register (CONFIG) — Changed COPCLK to CGMXCLK and TBMCLKSEL to TMBCLKSEL to be compatible with development tool nomenclature | 91<br>92<br>93 |
| | | 10.6.2 Stop Mode — Changed COPCLK to CGMXCLK | 125 |
| | | Figure 14-3. ESCI Module Block Diagram — Changed BUS_CLK to BUS CLOCK and removed reference to 4xBUSCLK | 192 |
| | | 14.4.2 Transmitter — Changed ESCIBDSRC to SCIBDSRC | 194 |
| | | 14.9.1 ESCI Arbiter Control Register and 14.9.3 Bit Time Measurement — Replaced one quarter with one half in the definition for ACLK = 1 | 217<br>218 |
| | | Figure 17-1. Timebase Block Diagram, 17.5 TBM Interrupt Rate, and Table 17-1. Timebase Divider Selection — Changed TBMCLKSEL to TMBCLKSEL to be compatible with development tool nomenclature | 260<br>261 |
| | | 21.9 Clock Generation Module (CGM) Characteristics — Updated section to include the following:<br>21.9.1 CGM Operating Conditions<br>21.9.2 CGM Component Information<br>21.9.3 CGM Acquisition/Lock Time Information | <br>322<br>322<br>323 |

# List of Chapters


Chapter 1 General Description . . . . . 23
Chapter 2 Memory . . . . . 33
Chapter 3 Analog-to-Digital Converter (ADC) . . . . . 61
Chapter 4 Clock Generator Module (CGM) . . . . . 73
Chapter 5 Configuration Register (CONFIG) . . . . . 91
Chapter 6 Computer Operating Properly (COP) Module . . . . . 95
Chapter 7 Central Processor Unit (CPU) . . . . . 99
Chapter 8 External Interrupt (IRQ) . . . . . 111
Chapter 9 Keyboard Interrupt Module (KBI) . . . . . 115
Chapter 10 Low-Power Modes . . . . . 123
Chapter 11 Low-Voltage Inhibit (LVI) . . . . . 131
Chapter 12 MSCAN08 Controller (MSCAN08) . . . . . 135
Chapter 13 Input/Output (I/O) Ports . . . . . 169
Chapter 14 Enhanced Serial Communications Interface (ESCI) Module . . . . . 189
Chapter 15 System Integration Module (SIM) . . . . . 221
Chapter 16 Serial Peripheral Interface (SPI) Module . . . . . 239
Chapter 17 Timebase Module (TBM) . . . . . 259
Chapter 18 Timer Interface Module (TIM1) . . . . . 263
Chapter 19 Timer Interface Module (TIM2) . . . . . 279
Chapter 20 Development Support . . . . . 299
Chapter 21 Electrical Specifications . . . . . 315
Chapter 22 Ordering Information and Mechanical Specifications . . . . . 333
Appendix A MC68HC908GZ48 . . . . . 343
Appendix B MC68HC908GZ32 . . . . . 347

# Table of Contents

## Chapter 1
## General Description


1.1 Introduction . . . 23
1.2 Features . . . 23
1.2.1 Standard Features . . . 23
1.2.2 Features of the CPU08 . . . 25
1.3 MCU Block Diagram . . . 25
1.4 Pin Assignments . . . 25
1.5 Pin Functions . . . 28
1.5.1 Power Supply Pins ($V_{DD}$ and $V_{SS}$) . . . 28
1.5.2 Oscillator Pins (OSC1 and OSC2) . . . 29
1.5.3 External Reset Pin ($\overline{RST}$) . . . 29
1.5.4 External Interrupt Pin ($\overline{IRQ}$) . . . 29
1.5.5 CGM Power Supply Pins ($V_{DDA}$ and $V_{SSA}$) . . . 29
1.5.6 External Filter Capacitor Pin (CGMXFC) . . . 29
1.5.7 ADC Power Supply/Reference Pins ($V_{DDAD}$/$V_{REFH}$ and $V_{SSAD}$/$V_{REFL}$) . . . 29
1.5.8 Port A Input/Output (I/O) Pins (PTA7/KBD7/AD15–PTA0/KBD0/AD8) . . . 30
1.5.9 Port B I/O Pins (PTB7/AD7–PTB0/AD0) . . . 30
1.5.10 Port C I/O Pins (PTC6–PTC0/CANTX) . . . 30
1.5.11 Port D I/O Pins (PTD7/T2CH1–PTD0/$\overline{SS}$) . . . 30
1.5.12 Port E I/O Pins (PTE5–PTE2, PTE1/RxD, and PTE0/TxD) . . . 30
1.5.13 Port F I/O Pins (PTF7/T2CH5–PTF0) . . . 31
1.5.14 Port G I/O Pins (PTG7/AD23–PTBG0/AD16) . . . 31
1.5.15 Unused Pin Termination . . . 31


## Chapter 2
## Memory


2.1 Introduction . . . 33
2.2 Unimplemented Memory Locations . . . 33
2.3 Reserved Memory Locations . . . 33
2.4 Input/Output (I/O) Section . . . 33
2.5 Random-Access Memory (RAM) . . . 45
2.6 FLASH-1 Memory (FLASH-1) . . . 45
2.6.1 Functional Description . . . 45
2.6.2 FLASH-1 Control and Block Protect Registers . . . 46
2.6.2.1 FLASH-1 Control Register . . . 46
2.6.2.2 FLASH-1 Block Protect Register . . . 47
2.6.3 FLASH-1 Block Protection . . . 48
2.6.4 FLASH-1 Mass Erase Operation . . . 48
2.6.5 FLASH-1 Page Erase Operation . . . 49

2.6.6 FLASH-1 Program Operation . . . 49
2.6.7 Low-Power Modes . . . 52
2.6.7.1 Wait Mode . . . 52
2.6.7.2 Stop Mode . . . 52
2.7 FLASH-2 Memory (FLASH-2) . . . 52
2.7.1 Functional Description . . . 52
2.7.2 FLASH-2 Control and Block Protect Registers . . . 53
2.7.2.1 FLASH-2 Control Register . . . 53
2.7.2.2 FLASH-2 Block Protect Register . . . 54
2.7.3 FLASH-2 Block Protection . . . 55
2.7.4 FLASH-2 Mass Erase Operation . . . 55
2.7.5 FLASH-2 Page Erase Operation . . . 56
2.7.6 FLASH-2 Program Operation . . . 57
2.7.7 Low-Power Modes . . . 58
2.7.7.1 Wait Mode . . . 58
2.7.7.2 Stop Mode . . . 58

## Chapter 3
## Analog-to-Digital Converter (ADC)

3.1 Introduction . . . 61
3.2 Features . . . 61
3.3 Functional Description . . . 61
3.3.1 ADC Port I/O Pins . . . 61
3.3.2 Voltage Conversion . . . 63
3.3.3 Conversion Time . . . 64
3.3.4 Conversion . . . 64
3.3.5 Accuracy and Precision . . . 64
3.3.6 Result Justification . . . 64
3.4 Monotonicity . . . 65
3.5 Interrupts . . . 65
3.6 Low-Power Modes . . . 66
3.6.1 Wait Mode . . . 66
3.6.2 Stop Mode . . . 66
3.7 I/O Signals . . . 66
3.7.1 ADC Analog Power Pin ($V_{DDAD}$) . . . 66
3.7.2 ADC Analog Ground Pin ($V_{SSAD}$) . . . 66
3.7.3 ADC Voltage Reference High Pin ($V_{REFH}$) . . . 67
3.7.4 ADC Voltage Reference Low Pin ($V_{REFL}$) . . . 67
3.7.5 ADC Voltage In ($V_{ADIN}$) . . . 67
3.8 I/O Registers . . . 67
3.8.1 ADC Status and Control Register . . . 68
3.8.2 ADC Data Register High and Data Register Low . . . 70
3.8.2.1 Left Justified Mode . . . 70
3.8.2.2 Right Justified Mode . . . 70
3.8.2.3 Left Justified Signed Data Mode . . . 71
3.8.2.4 Eight Bit Truncation Mode . . . 71
3.8.3 ADC Clock Register . . . 72

## Chapter 4
## Clock Generator Module (CGM)


4.1 Introduction . . . 73
4.2 Features . . . 73
4.3 Functional Description . . . 73
4.3.1 Crystal Oscillator Circuit . . . 75
4.3.2 Phase-Locked Loop Circuit (PLL) . . . 75
4.3.3 PLL Circuits . . . 75
4.3.4 Acquisition and Tracking Modes . . . 76
4.3.5 Manual and Automatic PLL Bandwidth Modes . . . 76
4.3.6 Programming the PLL . . . 77
4.3.7 Special Programming Exceptions . . . 79
4.3.8 Base Clock Selector Circuit . . . 79
4.3.9 CGM External Connections . . . 80
4.4 I/O Signals . . . 81
4.4.1 Crystal Amplifier Input Pin (OSC1) . . . 81
4.4.2 Crystal Amplifier Output Pin (OSC2) . . . 81
4.4.3 External Filter Capacitor Pin (CGMXFC) . . . 81
4.4.4 PLL Analog Power Pin ($V_{DDA}$) . . . 81
4.4.5 PLL Analog Ground Pin ($V_{SSA}$) . . . 81
4.4.6 Oscillator Enable Signal (SIMOSCEN) . . . 81
4.4.7 Oscillator Enable in Stop Mode Bit (OSCENINSTOP) . . . 81
4.4.8 Crystal Output Frequency Signal (CGMXCLK) . . . 82
4.4.9 CGM Base Clock Output (CGMOUT) . . . 82
4.4.10 CGM CPU Interrupt (CGMINT) . . . 82
4.5 CGM Registers . . . 82
4.5.1 PLL Control Register . . . 83
4.5.2 PLL Bandwidth Control Register . . . 85
4.5.3 PLL Multiplier Select Register High . . . 86
4.5.4 PLL Multiplier Select Register Low . . . 86
4.5.5 PLL VCO Range Select Register . . . 87
4.6 Interrupts . . . 88
4.7 Special Modes . . . 88
4.7.1 Wait Mode . . . 88
4.7.2 Stop Mode . . . 88
4.7.3 CGM During Break Interrupts . . . 88
4.8 Acquisition/Lock Time Specifications . . . 89
4.8.1 Acquisition/Lock Time Definitions . . . 89
4.8.2 Parametric Influences on Reaction Time . . . 89
4.8.3 Choosing a Filter . . . 90


## Chapter 5
## Configuration Register (CONFIG)


5.1 Introduction . . . 91
5.2 Functional Description . . . 91

## Chapter 6
## Computer Operating Properly (COP) Module


6.1 Introduction . . . 95
6.2 Functional Description . . . 95
6.3 I/O Signals . . . 96
6.3.1 CGMXCLK . . . 96
6.3.2 STOP Instruction . . . 96
6.3.3 COPCTL Write . . . 96
6.3.4 Power-On Reset . . . 96
6.3.5 Internal Reset . . . 96
6.3.6 COPD (COP Disable) . . . 97
6.3.7 COPRS (COP Rate Select) . . . 97
6.4 COP Control Register . . . 97
6.5 Interrupts . . . 97
6.6 Monitor Mode . . . 97
6.7 Low-Power Modes . . . 97
6.7.1 Wait Mode . . . 97
6.7.2 Stop Mode . . . 97
6.8 COP Module During Break Mode . . . 98


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


7.1 Introduction . . . 99
7.2 Features . . . 99
7.3 CPU Registers . . . 99
7.3.1 Accumulator . . . 100
7.3.2 Index Register . . . 100
7.3.3 Stack Pointer . . . 101
7.3.4 Program Counter . . . 101
7.3.5 Condition Code Register . . . 102
7.4 Arithmetic/Logic Unit (ALU) . . . 103
7.5 Low-Power Modes . . . 103
7.5.1 Wait Mode . . . 103
7.5.2 Stop Mode . . . 103
7.6 CPU During Break Interrupts . . . 103
7.7 Instruction Set Summary . . . 104
7.8 Opcode Map . . . 109


## Chapter 8
## External Interrupt (IRQ)


8.1 Introduction . . . 111
8.2 Features . . . 111
8.3 Functional Description . . . 111
8.4 $\overline{\text{IRQ}}$ Pin . . . 113
8.5 IRQ Module During Break Interrupts . . . 113
8.6 IRQ Status and Control Register . . . 114

## Chapter 9
## Keyboard Interrupt Module (KBI)


9.1 Introduction . . . 115
9.2 Features . . . 115
9.3 Functional Description . . . 115
9.4 Keyboard Initialization . . . 118
9.5 Low-Power Modes . . . 119
9.5.1 Wait Mode . . . 119
9.5.2 Stop Mode . . . 119
9.6 Keyboard Module During Break Interrupts . . . 119
9.7 I/O Registers . . . 119
9.7.1 Keyboard Status and Control Register . . . 120
9.7.2 Keyboard Interrupt Enable Register . . . 121
9.7.3 Keyboard Interrupt Polarity Register . . . 121


## Chapter 10
## Low-Power Modes


10.1 Introduction . . . 123
10.1.1 Wait Mode . . . 123
10.1.2 Stop Mode . . . 123
10.2 Analog-to-Digital Converter (ADC) . . . 123
10.2.1 Wait Mode . . . 123
10.2.2 Stop Mode . . . 123
10.3 Break Module (BRK) . . . 124
10.3.1 Wait Mode . . . 124
10.3.2 Stop Mode . . . 124
10.4 Central Processor Unit (CPU) . . . 124
10.4.1 Wait Mode . . . 124
10.4.2 Stop Mode . . . 124
10.5 Clock Generator Module (CGM) . . . 124
10.5.1 Wait Mode . . . 124
10.5.2 Stop Mode . . . 124
10.6 Computer Operating Properly Module (COP) . . . 125
10.6.1 Wait Mode . . . 125
10.6.2 Stop Mode . . . 125
10.7 External Interrupt Module (IRQ) . . . 125
10.7.1 Wait Mode . . . 125
10.7.2 Stop Mode . . . 125
10.8 Keyboard Interrupt Module (KBI) . . . 125
10.8.1 Wait Mode . . . 125
10.8.2 Stop Mode . . . 125
10.9 Low-Voltage Inhibit Module (LVI) . . . 126
10.9.1 Wait Mode . . . 126
10.9.2 Stop Mode . . . 126

10.10 Enhanced Serial Communications Interface Module (ESCI) . . . . . 126
10.10.1 Wait Mode . . . . . 126
10.10.2 Stop Mode . . . . . 126
10.11 Serial Peripheral Interface Module (SPI) . . . . . 126
10.11.1 Wait Mode . . . . . 126
10.11.2 Stop Mode . . . . . 126
10.12 Timer Interface Module (TIM1 and TIM2) . . . . . 127
10.12.1 Wait Mode . . . . . 127
10.12.2 Stop Mode . . . . . 127
10.13 Timebase Module (TBM) . . . . . 127
10.13.1 Wait Mode . . . . . 127
10.13.2 Stop Mode . . . . . 127
10.14 Scalable Controller Area Network Module (MSCAN) . . . . . 127
10.14.1 Wait Mode . . . . . 127
10.14.2 Stop Mode . . . . . 127
10.15 Exiting Wait Mode . . . . . 128
10.16 Exiting Stop Mode . . . . . 129

**Chapter 11**
**Low-Voltage Inhibit (LVI)**

11.1 Introduction . . . . . 131
11.2 Features . . . . . 131
11.3 Functional Description . . . . . 131
11.3.1 Polled LVI Operation . . . . . 132
11.3.2 Forced Reset Operation . . . . . 132
11.3.3 Voltage Hysteresis Protection . . . . . 133
11.3.4 LVI Trip Selection . . . . . 133
11.4 LVI Status Register . . . . . 133
11.5 LVI Interrupts . . . . . 133
11.6 Low-Power Modes . . . . . 134
11.6.1 Wait Mode . . . . . 134
11.6.2 Stop Mode . . . . . 134

**Chapter 12**
**MSCAN08 Controller (MSCAN08)**

12.1 Introduction . . . . . 135
12.2 Features . . . . . 135
12.3 External Pins . . . . . 137
12.4 Message Storage . . . . . 137
12.4.1 Background . . . . . 137
12.4.2 Receive Structures . . . . . 138
12.4.3 Transmit Structures . . . . . 139
12.5 Identifier Acceptance Filter . . . . . 140
12.6 Interrupts . . . . . 143
12.6.1 Interrupt Acknowledge . . . . . 143
12.6.2 Interrupt Vectors . . . . . 143

12.7 Protocol Violation Protection . . . . . 144
12.8 Low-Power Modes . . . . . 144
12.8.1 MSCAN08 Sleep Mode . . . . . 145
12.8.2 MSCAN08 Soft Reset Mode . . . . . 146
12.8.3 MSCAN08 Power-Down Mode . . . . . 146
12.8.4 CPU Wait Mode . . . . . 147
12.8.5 Programmable Wakeup Function . . . . . 147
12.9 Timer Link . . . . . 147
12.10 Clock System . . . . . 147
12.11 Memory Map . . . . . 150
12.12 Programmer's Model of Message Storage . . . . . 151
12.12.1 Message Buffer Outline . . . . . 152
12.12.2 Identifier Registers . . . . . 153
12.12.3 Data Length Register (DLR) . . . . . 154
12.12.4 Data Segment Registers (DSRn) . . . . . 154
12.12.5 Transmit Buffer Priority Registers . . . . . 154
12.13 Programmer's Model of Control Registers . . . . . 155
12.13.1 MSCAN08 Module Control Register 0 . . . . . 156
12.13.2 MSCAN08 Module Control Register 1 . . . . . 157
12.13.3 MSCAN08 Bus Timing Register 0 . . . . . 158
12.13.4 MSCAN08 Bus Timing Register 1 . . . . . 159
12.13.5 MSCAN08 Receiver Flag Register (CRFLG) . . . . . 160
12.13.6 MSCAN08 Receiver Interrupt Enable Register . . . . . 162
12.13.7 MSCAN08 Transmitter Flag Register . . . . . 163
12.13.8 MSCAN08 Transmitter Control Register . . . . . 164
12.13.9 MSCAN08 Identifier Acceptance Control Register . . . . . 164
12.13.10 MSCAN08 Receive Error Counter . . . . . 165
12.13.11 MSCAN08 Transmit Error Counter . . . . . 166
12.13.12 MSCAN08 Identifier Acceptance Registers . . . . . 166
12.13.13 MSCAN08 Identifier Mask Registers (CIDMR0–CIDMR3) . . . . . 167

## Chapter 13
## Input/Output (I/O) Ports

13.1 Introduction . . . . . 169
13.2 Unused Pin Termination . . . . . 169
13.3 Port A . . . . . 173
13.3.1 Port A Data Register . . . . . 173
13.3.2 Data Direction Register A . . . . . 174
13.3.3 Port A Input Pullup Enable Register . . . . . 175
13.4 Port B . . . . . 176
13.4.1 Port B Data Register . . . . . 176
13.4.2 Data Direction Register B . . . . . 176
13.5 Port C . . . . . 178
13.5.1 Port C Data Register . . . . . 178
13.5.2 Data Direction Register C . . . . . 178
13.5.3 Port C Input Pullup Enable Register . . . . . 180

13.6 Port D . . . . . 180
13.6.1 Port D Data Register . . . . . 180
13.6.2 Data Direction Register D . . . . . 181
13.6.3 Port D Input Pullup Enable Register . . . . . 182
13.7 Port E . . . . . 183
13.7.1 Port E Data Register . . . . . 183
13.7.2 Data Direction Register E . . . . . 184
13.8 Port F . . . . . 185
13.8.1 Port F Data Register . . . . . 185
13.8.2 Data Direction Register F . . . . . 185
13.9 Port G . . . . . 186
13.9.1 Port G Data Register . . . . . 186
13.9.2 Data Direction Register G . . . . . 187

## Chapter 14
## Enhanced Serial Communications Interface (ESCI) Module

14.1 Introduction . . . . . 189
14.2 Features . . . . . 189
14.3 Pin Name Conventions . . . . . 191
14.4 Functional Description . . . . . 191
14.4.1 Data Format . . . . . 191
14.4.2 Transmitter . . . . . 194
14.4.2.1 Character Length . . . . . 194
14.4.2.2 Character Transmission . . . . . 194
14.4.2.3 Break Characters . . . . . 195
14.4.2.4 Idle Characters . . . . . 196
14.4.2.5 Inversion of Transmitted Output . . . . . 196
14.4.2.6 Transmitter Interrupts . . . . . 196
14.4.3 Receiver . . . . . 196
14.4.3.1 Character Length . . . . . 197
14.4.3.2 Character Reception . . . . . 198
14.4.3.3 Data Sampling . . . . . 198
14.4.3.4 Framing Errors . . . . . 199
14.4.3.5 Baud Rate Tolerance . . . . . 200
14.4.3.6 Receiver Wakeup . . . . . 201
14.4.3.7 Receiver Interrupts . . . . . 202
14.4.3.8 Error Interrupts . . . . . 202
14.5 Low-Power Modes . . . . . 202
14.5.1 Wait Mode . . . . . 202
14.5.2 Stop Mode . . . . . 203
14.6 ESCI During Break Module Interrupts . . . . . 203
14.7 I/O Signals . . . . . 203
14.7.1 PTE0/TxD (Transmit Data) . . . . . 203
14.7.2 PTE1/RxD (Receive Data) . . . . . 203
14.8 I/O Registers . . . . . 203
14.8.1 ESCI Control Register 1 . . . . . 204
14.8.2 ESCI Control Register 2 . . . . . 206

14.8.3 ESCI Control Register 3 . . . 207
14.8.4 ESCI Status Register 1 . . . 209
14.8.5 ESCI Status Register 2 . . . 211
14.8.6 ESCI Data Register . . . 212
14.8.7 ESCI Baud Rate Register . . . 212
14.8.8 ESCI Prescaler Register . . . 214
14.9 ESCI Arbiter . . . 217
14.9.1 ESCI Arbiter Control Register . . . 217
14.9.2 ESCI Arbiter Data Register . . . 218
14.9.3 Bit Time Measurement . . . 218
14.9.4 Arbitration Mode . . . 218

## Chapter 15
## System Integration Module (SIM)

15.1 Introduction . . . 221
15.2 SIM Bus Clock Control and Generation . . . 224
15.2.1 Bus Timing . . . 224
15.2.2 Clock Startup from POR or LVI Reset . . . 224
15.2.3 Clocks in Stop Mode and Wait Mode . . . 224
15.3 Reset and System Initialization . . . 225
15.3.1 External Pin Reset . . . 225
15.3.2 Active Resets from Internal Sources . . . 225
15.3.2.1 Power-On Reset . . . 226
15.3.2.2 Computer Operating Properly (COP) Reset . . . 226
15.3.2.3 Illegal Opcode Reset . . . 227
15.3.2.4 Illegal Address Reset . . . 227
15.3.2.5 Low-Voltage Inhibit (LVI) Reset . . . 227
15.3.2.6 Monitor Mode Entry Module Reset (MODRST) . . . 227
15.4 SIM Counter . . . 228
15.4.1 SIM Counter During Power-On Reset . . . 228
15.4.2 SIM Counter During Stop Mode Recovery . . . 228
15.4.3 SIM Counter and Reset States . . . 228
15.5 Exception Control . . . 228
15.5.1 Interrupts . . . 228
15.5.1.1 Hardware Interrupts . . . 229
15.5.1.2 SWI Instruction . . . 231
15.5.1.3 Interrupt Status Registers . . . 231
15.5.2 Reset . . . 234
15.5.3 Break Interrupts . . . 234
15.5.4 Status Flag Protection in Break Mode . . . 234
15.6 Low-Power Modes . . . 234
15.6.1 Wait Mode . . . 234
15.6.2 Stop Mode . . . 235
15.7 SIM Registers . . . 236
15.7.1 Break Status Register . . . 237
15.7.2 SIM Reset Status Register . . . 237
15.7.3 Break Flag Control Register . . . 238

## Chapter 16
## Serial Peripheral Interface (SPI) Module

16.1 Introduction . . . 239
16.2 Features . . . 239
16.3 Functional Description . . . 239
16.3.1 Master Mode . . . 242
16.3.2 Slave Mode . . . 242
16.4 Transmission Formats . . . 243
16.4.1 Clock Phase and Polarity Controls . . . 243
16.4.2 Transmission Format When CPHA = 0 . . . 243
16.4.3 Transmission Format When CPHA = 1 . . . 244
16.4.4 Transmission Initiation Latency . . . 245
16.5 Queuing Transmission Data . . . 247
16.6 Error Conditions . . . 248
16.6.1 Overflow Error . . . 248
16.6.2 Mode Fault Error . . . 249
16.7 Interrupts . . . 251
16.8 Resetting the SPI . . . 252
16.9 Low-Power Modes . . . 252
16.9.1 Wait Mode . . . 252
16.9.2 Stop Mode . . . 253
16.10 SPI During Break Interrupts . . . 253
16.11 I/O Signals . . . 253
16.11.1 MISO (Master In/Slave Out) . . . 253
16.11.2 MOSI (Master Out/Slave In) . . . 253
16.11.3 SPSCK (Serial Clock) . . . 254
16.11.4 $\overline{SS}$ (Slave Select) . . . 254
16.12 I/O Registers . . . 255
16.12.1 SPI Control Register . . . 255
16.12.2 SPI Status and Control Register . . . 256
16.12.3 SPI Data Register . . . 258

## Chapter 17
## Timebase Module (TBM)

17.1 Introduction . . . 259
17.2 Features . . . 259
17.3 Functional Description . . . 259
17.4 Interrupts . . . 259
17.5 TBM Interrupt Rate . . . 260
17.6 Low-Power Modes . . . 261
17.6.1 Wait Mode . . . 261
17.6.2 Stop Mode . . . 261
17.7 Timebase Control Register . . . 262

## Chapter 18
## Timer Interface Module (TIM1)


18.1 Introduction . . . 263
18.2 Features . . . 263
18.3 Functional Description . . . 263
18.3.1 TIM1 Counter Prescaler . . . 263
18.3.2 Input Capture . . . 263
18.3.3 Output Compare . . . 265
18.3.3.1 Unbuffered Output Compare . . . 265
18.3.3.2 Buffered Output Compare . . . 267
18.3.4 Pulse Width Modulation (PWM) . . . 267
18.3.4.1 Unbuffered PWM Signal Generation . . . 268
18.3.4.2 Buffered PWM Signal Generation . . . 269
18.3.4.3 PWM Initialization . . . 269
18.4 Interrupts . . . 270
18.5 Wait Mode . . . 270
18.6 TIM1 During Break Interrupts . . . 270
18.7 Input/Output Signals . . . 271
18.8 Input/Output Registers . . . 271
18.8.1 TIM1 Status and Control Register . . . 271
18.8.2 TIM1 Counter Registers . . . 273
18.8.3 TIM1 Counter Modulo Registers . . . 273
18.8.4 TIM1 Channel Status and Control Registers . . . 274
18.8.5 TIM1 Channel Registers . . . 276


## Chapter 19
## Timer Interface Module (TIM2)


19.1 Introduction . . . 279
19.2 Features . . . 279
19.3 Functional Description . . . 279
19.3.1 TIM2 Counter Prescaler . . . 284
19.3.2 Input Capture . . . 284
19.3.3 Output Compare . . . 284
19.3.3.1 Unbuffered Output Compare . . . 285
19.3.3.2 Buffered Output Compare . . . 285
19.3.4 Pulse Width Modulation (PWM) . . . 286
19.3.4.1 Unbuffered PWM Signal Generation . . . 287
19.3.4.2 Buffered PWM Signal Generation . . . 287
19.3.4.3 PWM Initialization . . . 288
19.4 Interrupts . . . 289
19.5 Low-Power Modes . . . 289
19.5.1 Wait Mode . . . 289
19.5.2 Stop Mode . . . 289
19.6 TIM2 During Break Interrupts . . . 290

19.7 I/O Signals . . . . . 290
19.7.1 TIM2 Clock Pin (T2CH0) . . . . . 290
19.7.2 TIM2 Channel I/O Pins (T2CH5:T2CH2 and T2CH1:T2CH0) . . . . . 290
19.8 I/O Registers . . . . . 290
19.8.1 TIM2 Status and Control Register . . . . . 291
19.8.2 TIM2 Counter Registers . . . . . 292
19.8.3 TIM2 Counter Modulo Registers . . . . . 293
19.8.4 TIM2 Channel Status and Control Registers . . . . . 293
19.8.5 TIM2 Channel Registers . . . . . 297

## Chapter 20
## Development Support

20.1 Introduction . . . . . 299
20.2 Break Module (BRK) . . . . . 299
20.2.1 Functional Description . . . . . 299
20.2.1.1 Flag Protection During Break Interrupts . . . . . 302
20.2.1.2 TIM During Break Interrupts . . . . . 302
20.2.1.3 COP During Break Interrupts . . . . . 302
20.2.2 Break Module Registers . . . . . 302
20.2.2.1 Break Status and Control Register . . . . . 303
20.2.2.2 Break Address Registers . . . . . 303
20.2.2.3 Break Status Register . . . . . 304
20.2.2.4 Break Flag Control Register . . . . . 304
20.2.3 Low-Power Modes . . . . . 304
20.3 Monitor Module (MON) . . . . . 305
20.3.1 Functional Description . . . . . 305
20.3.1.1 Normal Monitor Mode . . . . . 309
20.3.1.2 Forced Monitor Mode . . . . . 309
20.3.1.3 Monitor Vectors . . . . . 309
20.3.1.4 Data Format . . . . . 310
20.3.1.5 Break Signal . . . . . 310
20.3.1.6 Baud Rate . . . . . 310
20.3.1.7 Commands . . . . . 310
20.3.2 Security . . . . . 314

## Chapter 21
## Electrical Specifications

21.1 Introduction . . . . . 315
21.2 Absolute Maximum Ratings . . . . . 315
21.3 Functional Operating Range . . . . . 316
21.4 Thermal Characteristics . . . . . 316
21.5 5.0-Vdc Electrical Characteristics . . . . . 317
21.6 3.3-Vdc Electrical Characteristics . . . . . 319
21.7 5.0-Volt Control Timing . . . . . 321
21.8 3.3-Volt Control Timing . . . . . 321

21.9 Clock Generation Module (CGM) Characteristics . . . . . 322
21.9.1 CGM Operating Conditions . . . . . 322
21.9.2 CGM Component Information . . . . . 322
21.9.3 CGM Acquisition/Lock Time Information . . . . . 323
21.10 5.0-Volt ADC Characteristics . . . . . 324
21.11 3.3-Volt ADC Characteristics . . . . . 325
21.12 5.0-Volt SPI Characteristics . . . . . 326
21.13 3.3-Volt SPI Characteristics . . . . . 327
21.14 Timer Interface Module Characteristics . . . . . 330
21.15 Memory Characteristics . . . . . 331

**Chapter 22**
**Ordering Information and Mechanical Specifications**

22.1 Introduction . . . . . 333
22.2 MC Order Numbers . . . . . 333
22.3 Package Dimensions . . . . . 333

**Appendix A**
**MC68HC908GZ48**

A.1 Introduction . . . . . 343
A.2 Block Diagram . . . . . 343
A.3 Memory . . . . . 343
A.4 Ordering Information . . . . . 346

**Appendix B**
**MC68HC908GZ32**

B.1 Introduction . . . . . 347
B.2 Block Diagram . . . . . 347
B.3 Memory . . . . . 347
B.4 Ordering Information . . . . . 350

# Chapter 1
# General Description

## 1.1 Introduction

The MC68HC908GZ60, MC68HC908GZ48, and MC68HC908GZ32 are members of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

The information contained in this document pertains to all three devices with the exceptions noted in Appendix A MC68HC908GZ48 and Appendix B MC68HC908GZ32.

## 1.2 Features

For convenience, features have been organized to reflect:

- Standard features
- Features of the CPU08

### 1.2.1 Standard Features

Features of the MC68HC908GZ60 include:

- High-performance M68HC08 architecture optimized for C-compilers
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- 8-MHz internal bus frequency
- Clock generation module supporting 1-MHz to 8-MHz crystals
- MSCAN08 (scalable controller area network) controller (implementing 2.0b protocol as defined in BOSCH specification dated September 1991)
- FLASH program memory security[1]
- On-chip programming firmware for use with host personal computer which does not require high voltage for entry
- In-system programming (ISP)
- System protection features:
  - Optional computer operating properly (COP) reset
  - Low-voltage detection with optional reset and selectable trip points for 3.3-V and 5.0-V operation
  - Illegal opcode detection with reset
  - Illegal address detection with reset
- Low-power design; fully static with stop and wait modes

1. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH difficult for unauthorized users.

- Standard low-power modes of operation:
  - Wait mode
  - Stop mode
- Master reset pin and power-on reset (POR)
- On-chip FLASH memory:
  - MC68HC908GZ60 — 60 Kbytes
  - MC68HC908GZ48 — 48 Kbytes
  - MC68HC908GZ32 — 32 Kbytes
- Random-access memory (RAM):
  - MC68HC908GZ60 — 2048 bytes
  - MC68HC908GZ48 — 1536 bytes
  - MC68HC908GZ32 — 1536 bytes
- Serial peripheral interface (SPI) module
- Enhanced serial communications interface (ESCI) module
- One 16-bit, 2-channel timer interface module (TIM1) with selectable input capture, output compare, and pulse-width modulation (PWM) capability on each channel
- One 16-bit, 6-channel timer interface module (TIM2) with selectable input capture, output compare, and pulse-width modulation (PWM) capability on each channel
- Timebase module with clock prescaler circuitry for eight user selectable periodic real-time interrupts with optional active clock source during stop mode for periodic wakeup from stop using an external crystal
- 24-channel, 10-bit successive approximation analog-to-digital converter (ADC)
- 8-bit keyboard wakeup port with software selectable rising or falling edge detect, as well as high or low level detection
- Up to 53 general-purpose input/output (I/O) pins, including:
  - 40 shared-function I/O pins, depending on package choice
  - Up to 13 dedicated I/O pins, depending on package choice
- Selectable pullups on inputs only on ports A, C, and D. Selection is on an individual port bit basis. During output mode, pullups are disengaged.
- Internal pullups on $\overline{\text{IRQ}}$ and $\overline{\text{RST}}$ to reduce customer system cost
- High current 10-mA sink/source capability on all port pins
- Higher current 20-mA sink/source capability on PTC0–PTC4 and PTF0–PTF3
- User selectable clockout feature with divide by 1, 2, and 4 of the bus or crystal frequency
- User selection of having the oscillator enabled or disabled during stop mode
- BREAK module (BRK) to allow single breakpoint setting during in-circuit debugging
- Available packages:
  - 32-pin low-profile quad flat pack (LQFP)
  - 48-pin low-profile quad flat pack (LQFP)
  - 64-pin quad flat pack (QFP)
- Specific features in 32-pin LQFP are:
  - Port A is only 4 bits: PTA0–PTA3; shared with ADC and KBI modules
  - Port B is only 6 bits: PTB0–PTB5; shared with ADC module
  - Port C is only 2 bits: PTC0–PTC1; shared with MSCAN module
  - Port D is only 7 bits: PTD0–PTD6; shared with SPI, TIM1 and TIM2 modules
  - Port E is only 2 bits: PTE0–PTE1; shared with ESCI module

- Specific features in 48-pin LQFP are:
  - Port A is 8 bits: PTA0–PTA7; shared with ADC and KBI modules
  - Port B is 8 bits: PTB0–PTB7; shared with ADC module
  - Port C is only 7 bits: PTC0–PTC6; shared with MSCAN module
  - Port D is 8 bits: PTD0–PTD7; shared with SPI, TIM1, and TIM2 modules
  - Port E is only 6 bits: PTE0–PTE5; shared with ESCI module
- Specific features in 64-pin QFP are:
  - Port A is 8 bits: PTA0–PTA7; shared with ADC and KBI modules
  - Port B is 8 bits: PTB0–PTB7; shared with ADC module
  - Port C is only 7 bits: PTC0–PTC6; shared with MSCAN module
  - Port D is 8 bits: PTD0–PTD7; shared with SPI, TIM1, andTIM2 modules
  - Port E is only 6 bits: PTE0–PTE5; shared with ESCI module
  - Port F is 8 bits: PTF0–PTF7; shared with TIM2 module
  - Port G is 8 bits; PTG0–PTG7; shared with ADC module

### 1.2.2 Features of the CPU08

Features of the CPU08 include:

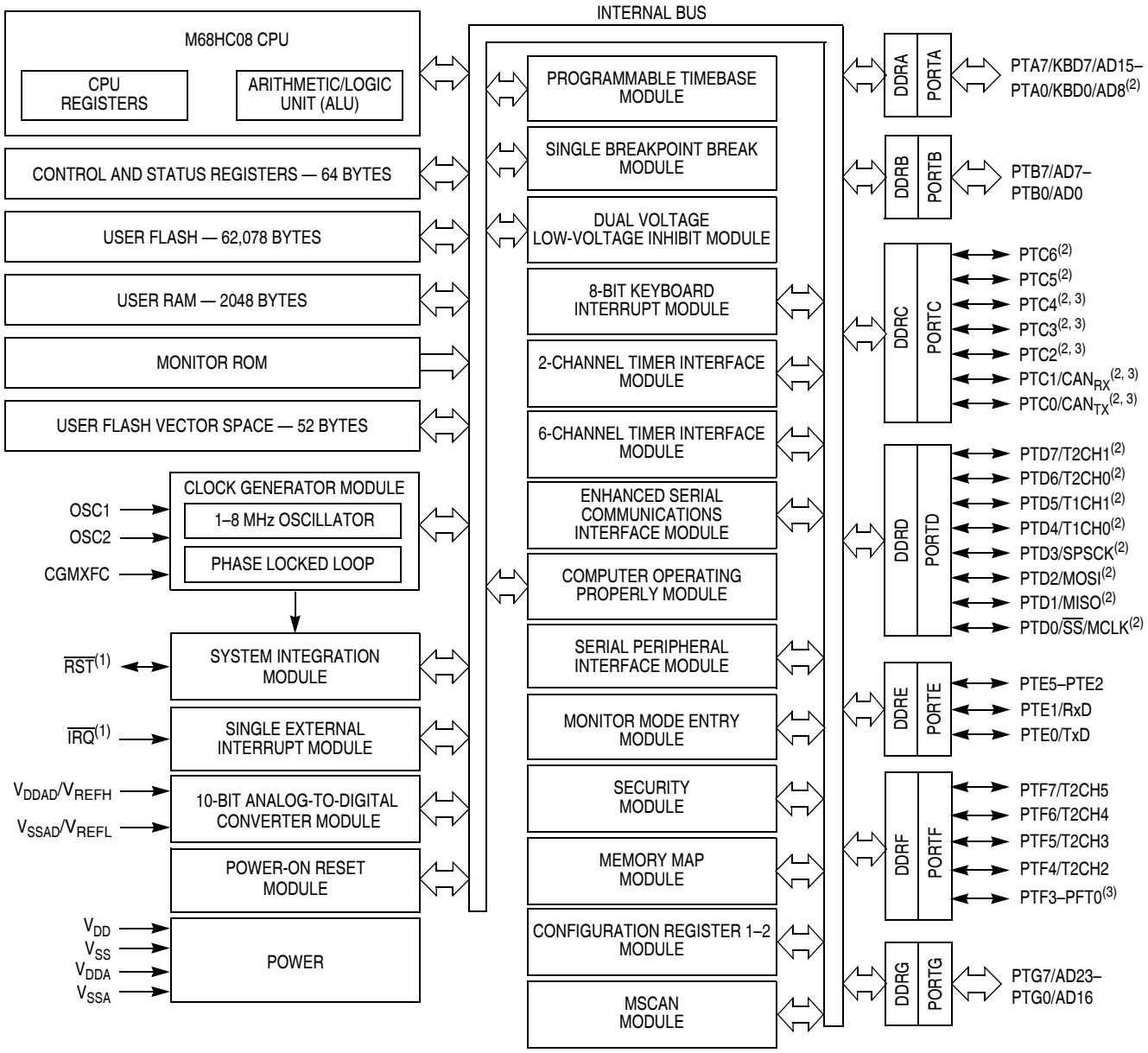
- Enhanced HC05 programming model
- Extensive loop control functions
- 16 addressing modes (eight more than the HC05)
- 16-bit index register and stack pointer
- Memory-to-memory data transfers
- Fast 8 × 8 multiply instruction
- Fast 16/8 divide instruction
- Binary-coded decimal (BCD) instructions
- Optimization for controller applications
- Efficient C language support

## 1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908GZ60. Refer to Appendix A MC68HC908GZ48 and Appendix B MC68HC908GZ32.

## 1.4 Pin Assignments

Figure 1-2, Figure 1-3, and Figure 1-4 illustrate the pin assignments for the 32-pin LQFP, 48-pin LQFP, and 64-pin QFP respectively.




1. Pin contains integrated pullup device.
2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.
3. Higher current drive port pins

**Figure 1-1. MC68HC908GZ60 Block Diagram**

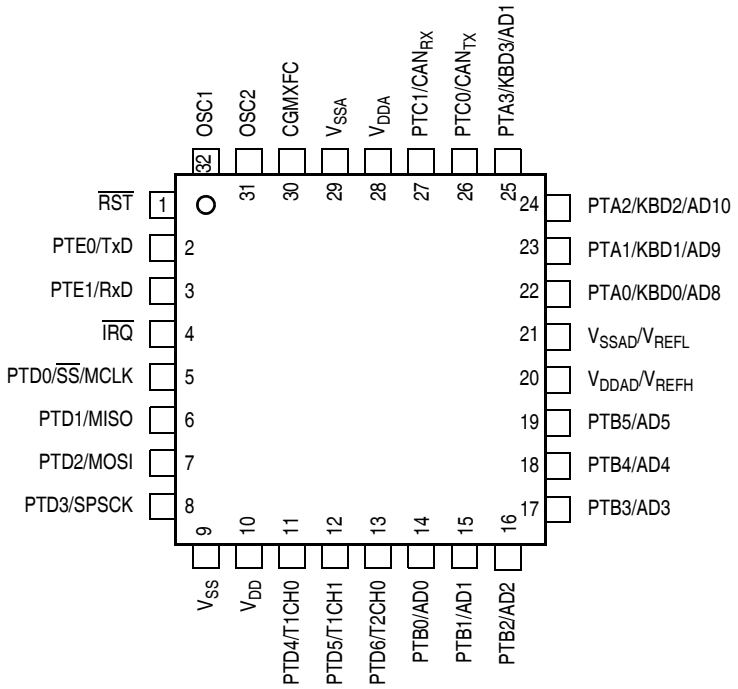



**Figure 1-2. 32-Pin LQFP Pin Assignments**

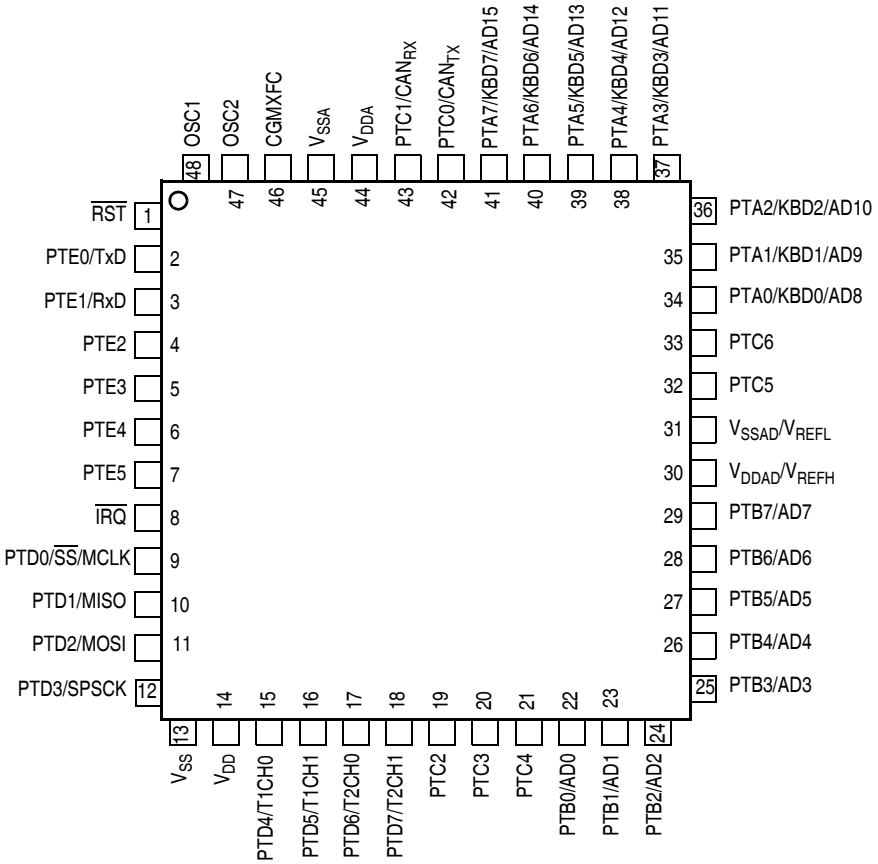



**Figure 1-3. 48-Pin LQFP Pin Assignments**

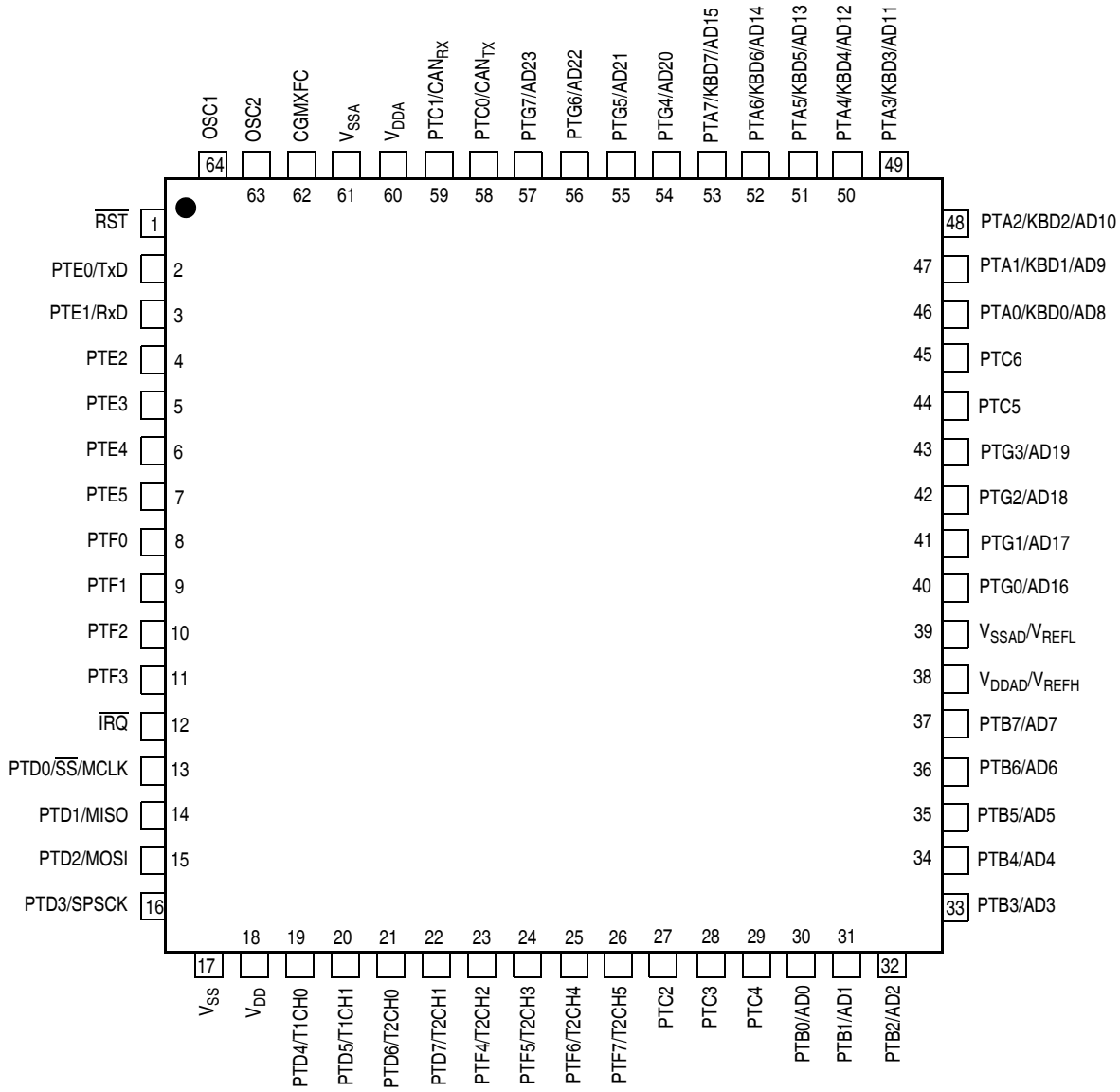



**Figure 1-4. 64-Pin QFP Pin Assignments**

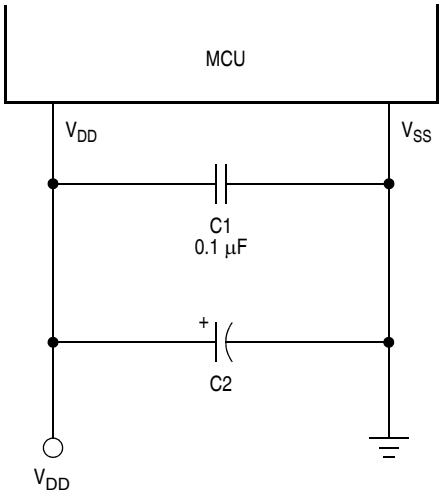
## 1.5 Pin Functions

Descriptions of the pin functions are provided here.

### 1.5.1 Power Supply Pins ($V_{DD}$ and $V_{SS}$)

$V_{DD}$ and $V_{SS}$ are the power supply and ground pins. The MCU operates from a single power supply.

Fast signal transitions on MCU pins place high, short-duration current demands on the power supply. To prevent noise problems, take special care to provide power supply bypassing at the MCU as Figure 1-5 shows. Place the C1 bypass capacitor as close to the MCU as possible. Use a high-frequency-response ceramic capacitor for C1. C2 is an optional bulk current bypass capacitor for use in applications that require the port pins to source high current levels.




Note: Component values shown represent typical applications.

**Figure 1-5. Power Supply Bypassing**

### 1.5.2 Oscillator Pins (OSC1 and OSC2)

OSC1 and OSC2 are the connections for an external crystal, resonator, or clock circuit. See Chapter 4 Clock Generator Module (CGM).

### 1.5.3 External Reset Pin ($\overline{RST}$)

A low on the $\overline{RST}$ pin forces the MCU to a known startup state. $\overline{RST}$ is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted. This pin contains an internal pullup resistor. See Chapter 15 System Integration Module (SIM).

### 1.5.4 External Interrupt Pin ($\overline{IRQ}$)

$\overline{IRQ}$ is an asynchronous external interrupt pin. This pin contains an internal pullup resistor. See Chapter 8 External Interrupt (IRQ).

### 1.5.5 CGM Power Supply Pins ($V_{DDA}$ and $V_{SSA}$)

$V_{DDA}$ and $V_{SSA}$ are the power supply pins for the analog portion of the clock generator module (CGM). Decoupling of these pins should be as per the digital supply. See Chapter 4 Clock Generator Module (CGM).

### 1.5.6 External Filter Capacitor Pin (CGMXFC)

CGMXFC is an external filter capacitor connection for the CGM. See Chapter 4 Clock Generator Module (CGM).

### 1.5.7 ADC Power Supply/Reference Pins ($V_{DDAD}/V_{REFH}$ and $V_{SSAD}/V_{REFL}$)

$V_{DDAD}$ and $V_{SSAD}$ are the power supply pins to the analog-to-digital converter (ADC). $V_{REFH}$ and $V_{REFL}$ are the reference voltage pins for the ADC. $V_{REFH}$ is the high reference supply for the ADC, and by default the $V_{DDAD}/V_{REFH}$ pin should be externally filtered and connected to the same voltage potential as $V_{DD}$.

$V_{REFL}$ is the low reference supply for the ADC, and by default the $V_{SSAD}/V_{REFL}$ pin should be connected to the same voltage potential as $V_{SS}$. See Chapter 3 Analog-to-Digital Converter (ADC).

### 1.5.8 Port A Input/Output (I/O) Pins (PTA7/KBD7/AD15–PTA0/KBD0/AD8)

PTA7–PTA0 are general-purpose, bidirectional I/O port pins. Any or all of the port A pins can be programmed to serve as keyboard interrupt pins or used as analog-to-digital inputs. PTA7–PTA4 are only available on the 48-pin LQFP and 64-pin QFP packages. See Chapter 13 Input/Output (I/O) Ports, Chapter 9 Keyboard Interrupt Module (KBI), and Chapter 3 Analog-to-Digital Converter (ADC).

These port pins also have selectable pullups when configured for input mode. The pullups are disengaged when configured for output mode. The pullups are selectable on an individual port bit basis.

### 1.5.9 Port B I/O Pins (PTB7/AD7–PTB0/AD0)

PTB7–PTB0 are general-purpose, bidirectional I/O port pins that can also be used for analog-to-digital converter (ADC) inputs. PTB7–PTB6 are only available on the 48-pin LQFP and 64-pin QFP packages. See Chapter 13 Input/Output (I/O) Ports and Chapter 3 Analog-to-Digital Converter (ADC).

### 1.5.10 Port C I/O Pins (PTC6–PTC0/$CAN_{TX}$)

PTC6 and PTC5 are general-purpose, bidirectional I/O port pins.

PTC4–PTC0 are general-purpose, bidirectional I/O port pins that contain higher current sink/source capability. PTC6–PTC2 are only available on the 48-pin LQFP and 64-pin QFP packages. See Chapter 13 Input/Output (I/O) Ports.

PTC1 and PTC0 can be programmed to be MSCAN08 pins.

These port pins also have selectable pullups when configured for input mode. The pullups are disengaged when configured for output mode. The pullups are selectable on an individual port bit basis.

### 1.5.11 Port D I/O Pins (PTD7/T2CH1–PTD0/$\overline{SS}$)

PTD7–PTD0 are special-function, bidirectional I/O port pins. PTD3–PTD0 can be programmed to be serial peripheral interface (SPI) pins, while PTD7–PTD4 can be individually programmed to be timer interface module (TIM1 and TIM2) pins. PTD0 can be used to output a clock, MCLK. PTD7 is only available on the 48-pin LQFP and 64-pin QFP packages. See Chapter 18 Timer Interface Module (TIM1), Chapter 19 Timer Interface Module (TIM2), Chapter 16 Serial Peripheral Interface (SPI) Module, Chapter 13 Input/Output (I/O) Ports. and Chapter 5 Configuration Register (CONFIG).

These port pins also have selectable pullups when configured for input mode. The pullups are disengaged when configured for output mode. The pullups are selectable on an individual port bit basis.

### 1.5.12 Port E I/O Pins (PTE5–PTE2, PTE1/RxD, and PTE0/TxD)

PTE5–PTE0 are general-purpose, bidirectional I/O port pins. PTE1 and PTE0 can also be programmed to be enhanced serial communications interface (ESCI) pins. PTE5–PTE2 are only available on the 48-pin LQFP and 64-pin QFP packages. See Chapter 14 Enhanced Serial Communications Interface (ESCI) Module and Chapter 13 Input/Output (I/O) Ports.

### 1.5.13 Port F I/O Pins (PTF7/T2CH5–PTF0)

PTF7–PTF4 are special-function, bidirectional I/O port pins that can be individually programmed to be timer interface module (TIM2) pins.

PTF3–PTF0 are general-purpose, bidirectional I/O port pins that contain higher current sink/source capability.

PTF7–PTF0 are only available on the 64-pin QFP package. See Chapter 18 Timer Interface Module (TIM1), Chapter 19 Timer Interface Module (TIM2), and Chapter 13 Input/Output (I/O) Ports.

### 1.5.14 Port G I/O Pins (PTG7/AD23–PTBG0/AD16)

PTG7–PTG0 are general-purpose, bidirectional I/O port pins that can also be used for analog-to-digital converter (ADC) inputs. PTG7–PTG0 are only available on the 64-pin QFP package. See Chapter 13 Input/Output (I/O) Ports and Chapter 3 Analog-to-Digital Converter (ADC).

### 1.5.15 Unused Pin Termination

Input pins and I/O port pins that are not used in the application must be terminated. This prevents excess current caused by floating inputs, and enhances immunity during noise or transient events. Termination methods include:

1. Configuring unused pins as outputs and driving high or low;
2. Configuring unused pins as inputs and enabling internal pull-ups;
3. Configuring unused pins as inputs and using external pull-up or pull-down resistors.

Never connect unused pins directly to $V_{DD}$ or $V_{SS}$.

Since some general-purpose I/O pins are not available on all packages, these pins must be terminated as well. Either method 1 or 2 above are appropriate.

# Chapter 2
# Memory

## 2.1 Introduction

The CPU08 can address 64 Kbytes of memory space. The memory map, shown in Figure 2-1, includes:

- 62,078 bytes of user FLASH memory
- 2048 bytes of random-access memory (RAM)
- 52 bytes of user-defined vectors

## 2.2 Unimplemented Memory Locations

Accessing an unimplemented location can cause an illegal address reset. In the memory map (Figure 2-1) and in register figures in this document, unimplemented locations are shaded.

## 2.3 Reserved Memory Locations

Accessing a reserved location can have unpredictable effects on microcontroller (MCU) operation. In the Figure 2-1 and in register figures in this document, reserved locations are marked with the word Reserved or with the letter R.

## 2.4 Input/Output (I/O) Section

Most of the control, status, and data registers are in the zero page area of $0000–$003F, or at $0440–$0461. Additional I/O registers have these addresses:

- $FE00; SIM break status register, BSR
- $FE01; SIM reset status register, SRSR
- $FE02; reserved
- $FE03; SIM break flag control register, BFCR
- $FE04; interrupt status register 1, INT1
- $FE05; interrupt status register 2, INT2
- $FE06; interrupt status register 3, INT3
- $FE07; interrupt status register 4, INT4
- $FE08; FLASH-2 control register, FL2CR
- $FE09; break address register high, BRKH
- $FE0A; break address register low, BRKL
- $FE0B; break status and control register, BRKSCR
- $FE0C; LVI status register, LVISR
- $FE0D; FLASH-2 test control register, FLTCR2
- $FE0E; FLASH-1 test control register, FLTCR1
- $FF80; FLASH-1 block protect register, FL1BPR

- $FF81; FLASH-2 block protect register, FL2BPR
- $FF88; FLASH-1 control register, FL1CR

Data registers are shown in Figure 2-2. Table 2-1 is a list of vector locations.

| | |
|---|---|
| $0000 ↓ $003F | I/O REGISTERS 64 BYTES |
| $0040 ↓ $043F | RAM-1 1024 BYTES |
| $0440 ↓ $0461 | I/O REGISTERS 34 BYTES |
| $0462 ↓ $04FF | FLASH-2 158 BYTES |
| $0500 ↓ $057F | MSCAN CONTROL AND MESSAGE BUFFER 128 BYTES |
| $0580 ↓ $097F | RAM-2 1024 BYTES |
| $0980 ↓ $1B7F | FLASH-2 4608 BYTES |
| $1B80 ↓ $1DFF | RESERVED 640 BYTES |
| $1E00 ↓ $1E0F | MONITOR ROM 16 BYTES |
| $1E10 ↓ $1E1F | RESERVED 16 BYTES |
| $1E20 ↓ $7FFF | FLASH-2 25,056 BYTES |
| $8000 ↓ $FDFF | FLASH-1 32,256 BYTES |

| | |
|---|---|
| $FE00 | SIM BREAK STATUS REGISTER (BSR) |
| $FE01 | SIM RESET STATUS REGISTER (SRSR) |
| $FE02 | RESERVED |
| $FE03 | SIM BREAK FLAG CONTROL REGISTER (BFCR) |
| $FE04 | INTERRUPT STATUS REGISTER 1 (INT1) |
| $FE05 | INTERRUPT STATUS REGISTER 2 (INT2) |
| $FE06 | INTERRUPT STATUS REGISTER 3 (INT3) |
| $FE07 | INTERRUPT STATUS REGISTER 4 (INT4) |
| $FE08 | FLASH-2 CONTROL REGISTER (FL2CR) |
| $FE09 | BREAK ADDRESS REGISTER HIGH (BRKH) |
| $FE0A | BREAK ADDRESS REGISTER LOW (BRKL) |
| $FE0B | BREAK STATUS AND CONTROL REGISTER (BRKSCR) |
| $FE0C | LVI STATUS REGISTER (LVISR) |
| $FE0D | FLASH-2 TEST CONTROL REGISTER (FLTCR2) |
| $FE0E | FLASH-1 TEST CONTROL REGISTER (FLTCR1) |
| $FE0F | UNIMPLEMENTED |
| $FE10 ↓ $FE1F | UNIMPLEMENTED 16 BYTES RESERVED FOR COMPATIBILITY WITH MONITOR CODE FOR A-FAMILY PART |
| $FE20 ↓ $FF7F | MONITOR ROM 352 BYTES |
| $FF80 | FLASH-1 BLOCK PROTECT REGISTER (FL1BPR) |
| $FF81 | FLASH-2 BLOCK PROTECT REGISTER (FL2BPR) |
| $FF82 ↓ $FF87 | RESERVED 6 BYTES |
| $FF88 | FLASH-1 CONTROL REGISTER (FL1CR) |
| $FF89 ↓ $FFCB | RESERVED 67 BYTES |
| $FFCC ↓ $FFFF(1) | FLASH-1 VECTORS 52 BYTES |

1. $FFF6–$FFFD used for eight security bytes

**Figure 2-1. MC68HC908GZ60 Memory Map**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0000 | Port A Data Register (PTA) See page 173. | Read: Write: | PTA7 | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0001 | Port B Data Register (PTB) See page 176. | Read: Write: | PTB7 | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1 | PTB0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0002 | Port C Data Register (PTC) See page 178. | Read: | 1 | PTC6 | PTC5 | PTC4 | PTC3 | PTC2 | PTC1 | PTC0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0003 | Port D Data Register (PTD) See page 180. | Read: Write: | PTD7 | PTD6 | PTD5 | PTD4 | PTD3 | PTD2 | PTD1 | PTD0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0004 | Data Direction Register A (DDRA) See page 174. | Read: Write: | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0005 | Data Direction Register B (DDRB) See page 176. | Read: Write: | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0006 | Data Direction Register C (DDRC) See page 178. | Read: | 0 | DDRC6 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0007 | Data Direction Register D (DDRD) See page 181. | Read: Write: | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0008 | Port E Data Register (PTE) See page 183. | Read: | 0 | 0 | PTE5 | PTE4 | PTE3 | PTE2 | PTE1 | PTE0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0009 | ESCI Prescaler Register (SCPSC) See page 214. | Read: Write: | PDS2 | PDS1 | PDS0 | PSSB4 | PSSB3 | PSSB2 | PSSB1 | PSSB0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000A | ESCI Arbiter Control Register (SCIACTL) See page 217. | Read: | AM1 | ALOST | AM0 | ACLK | AFIN | ARUN | AROVFL | ARD8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000B | ESCI Arbiter Data Register (SCIADAT) See page 218. | Read: | ARD7 | ARD6 | ARD5 | ARD4 | ARD3 | ARD2 | ARD1 | ARD0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | = Unimplemented | | R = Reserved | | U = Unaffected | | | |

**Figure 2-2. Control, Status, and Data Registers (Sheet 1 of 9)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $000C | Data Direction Register E (DDRE) See page 184. | Read: | 0 | 0 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000D | Port A Input Pullup Enable Register (PTAPUE) See page 175. | Read: | PTAPUE7 | PTAPUE6 | PTAPUE5 | PTAPUE4 | PTAPUE3 | PTAPUE2 | PTAPUE1 | PTAPUE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000E | Port C Input Pullup Enable Register (PTCPUE) See page 180. | Read: | 0 | PTCPUE6 | PTCPUE5 | PTCPUE4 | PTCPUE3 | PTCPUE2 | PTCPUE1 | PTCPUE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000F | Port D Input Pullup Enable Register (PTDPUE) See page 182. | Read: | PTDPUE7 | PTDPUE6 | PTDPUE5 | PTDPUE4 | PTDPUE3 | PTDPUE2 | PTDPUE1 | PTDPUE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0010 | SPI Control Register (SPCR) See page 255. | Read: | SPRIE | R | SPMSTR | CPOL | CPHA | SPWOM | SPE | SPTIE |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| $0011 | SPI Status and Control Register (SPSCR) See page 256. | Read: | SPRF | ERRIE | OVRF | MODF | SPTE | MODFEN | SPR1 | SPR0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $0012 | SPI Data Register (SPDR) See page 258. | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0013 | ESCI Control Register 1 (SCC1) See page 204. | Read: | LOOPS | ENSCI | TXINV | M | WAKE | ILTY | PEN | PTY |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0014 | ESCI Control Register 2 (SCC2) See page 206. | Read: | SCTIE | TCIE | SCRIE | ILIE | TE | RE | RWU | SBK |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0015 | ESCI Control Register 3 (SCC3) See page 208. | Read: | R8 | T8 | R | R | ORIE | NEIE | FEIE | PEIE |
| | | Write: | | | | | | | | |
| | | Reset: | U | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0016 | ESCI Status Register 1 (SCS1) See page 209. | Read: | SCTE | TC | SCRF | IDLE | OR | NF | FE | PE |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0017 | ESCI Status Register 2 (SCS2) See page 211. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | BKF | RPF |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 2 of 9)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0018 | ESCI Data Register (SCDR) See page 212. | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0019 | ESCI Baud Rate Register (SCBR) See page 212. | Read: | LINT | LINR | SCP1 | SCP0 | R | SCR2 | SCR1 | SCR0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $001A | Keyboard Status and Control Register (INTKBSCR) See page 120. | Read: | 0 | 0 | 0 | 0 | KEYF | 0 | IMASKK | MODEK |
| | | Write: | | | | | | ACKK | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $001B | Keyboard Interrupt Enable Register (INTKBIER) See page 121. | Read: | KBIE7 | KBIE6 | KBIE5 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $001C | Timebase Module Control Register (TBCR) See page 262. | Read: | TBIF | TBR2 | TBR1 | TBR0 | 0 | TBIE | TBON | R |
| | | Write: | | | | | TACK | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $001D | IRQ Status and Control Register (INTSCR) See page 114. | Read: | 0 | 0 | 0 | 0 | IRQF | 0 | IMASK | MODE |
| | | Write: | | | | | | ACK | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $001E | Configuration Register 2 (CONFIG2)(1) See page 92. | Read: | 0 | MCLKSEL | MCLK1 | MCLK0 | MSCAN-EN(1) | TMBCLK-SEL | OSCENIN-STOP | SCIBDSRC |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| $001F | Configuration Register 1 (CONFIG1)(1) See page 93. | Read: | COPRS | LVISTOP | LVIRSTD | LVIPWRD | LVI5OR3(1) | SSREC | STOP | COPD |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. One-time writable register after each reset, except MSCANEN and LVI5OR3 bits. MSCANEN andLVI5OR3 bits are only reset via POR (power-on reset).

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $0020 | TIM1 Status and Control Register (T1SC) See page 271. | Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $0021 | TIM1 Counter Register High (T1CNTH) See page 273. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0022 | TIM1 Counter Register Low (T1CNTL) See page 273. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0023 | TIM1 Counter Modulo Register High (T1MODH) See page 273. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 9)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0024 | TIM1 Counter Modulo Register Low (T1MODL) See page 273. | Read: Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $0025 | TIM1 Channel 0 Status and Control Register (T1SC0) See page 274. | Read: | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0026 | TIM1 Channel 0 Register High (T1CH0H) See page 277. | Read: Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0027 | TIM1 Channel 0 Register Low (T1CH0L) See page 277. | Read: Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0028 | TIM1 Channel 1 Status and Control Register (T1SC1) See page 274. | Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0029 | TIM1 Channel 1 Register High (T1CH1H) See page 277. | Read: Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $002A | TIM1 Channel 1 Register Low (T1CH1L) See page 277. | Read: Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $002B | TIM2 Status and Control Register (T2SC) See page 291. | Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $002C | TIM2 Counter Register High (T2CNTH) See page 292. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $002D | TIM2 Counter Register Low (T2CNTL) See page 292. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $002E | TIM2 Counter Modulo Register High (T2MODH) See page 293. | Read: Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $002F | TIM2 Counter Modulo Register Low (T2MODL) See page 293. | Read: Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 9)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0030 | TIM2 Channel 0 Status and Control Register (T2SC0) See page 293. | Read: | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0031 | TIM2 Channel 0 Register High (T2CH0H) See page 297. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0032 | TIM2 Channel 0 Register Low (T2CH0L) See page 297. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0033 | TIM2 Channel 1 Status and Control Register (T2SC1) See page 293. | Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0034 | TIM2 Channel 1 Register High (T2CH1H) See page 297. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0035 | TIM2 Channel 1 Register Low (T2CH1L) See page 297. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0036 | PLL Control Register (PCTL) See page 83. | Read: | PLLIE | PLLF | PLLON | BCS | R | R | VPR1 | VPR0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $0037 | PLL Bandwidth Control Register (PBWC) See page 85. | Read: | AUTO | LOCK | $\overline{ACQ}$ | 0 | 0 | 0 | 0 | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0038 | PLL Multiplier Select High Register (PMSH) See page 86. | Read: | 0 | 0 | 0 | 0 | MUL11 | MUL10 | MUL9 | MUL8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0039 | PLL Multiplier Select Low Register (PMSL) See page 86. | Read: | MUL7 | MUL6 | MUL5 | MUL4 | MUL3 | MUL2 | MUL1 | MUL0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $003A | PLL VCO Select Range Register (PMRS) See page 87. | Read: | VRS7 | VRS6 | VRS5 | VRS4 | VRS3 | VRS2 | VRS1 | VRS0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $003B | Reserved | Read: | 0 | 0 | 0 | 0 | R | R | R | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 9)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $003C | ADC Status and Control Register (ADSCR) See page 68. | Read: | COCO | AIEN | ADCO | ADCH4 | ADCH3 | ADCH2 | ADCH1 | ADCH0 |
| | | Write: | R | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| $003D | ADC Data High Register (ADRH) See page 70. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | AD9 | AD8 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $003E | ADC Data Low Register (ADRL) See page 70. | Read: | AD7 | AD6 | AD5 | AD4 | A3 | AD2 | AD1 | AD0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $003F | ADC Clock Register (ADCLK) See page 72. | Read: | ADIV2 | ADIV1 | ADIV0 | ADICLK | MODE1 | MODE0 | R | 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $0440 | Port F Data Register (PTF) See page 185. | Read: | PTF7 | PTF6 | PTF5 | PTF4 | PTAF3 | PTF2 | PTF1 | PTF0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0441 | Port G Data Register (PTG) See page 186. | Read: | PTG7 | PTG6 | PTG5 | PTG4 | PTG3 | PTG2 | PTG1 | PTG0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0444 | Data Direction Register F (DDRF) See page 185. | Read: | DDRF7 | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0445 | Data Direction Register G (DDRG) See page 187. | Read: | DDRG7 | DDRG6 | DDRG5 | DDRG4 | DDRG3 | DDRG2 | DDRG1 | DDRG0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0448 | Keyboard Interrupt Polarity Register (INTKBIPR) See page 121. | Read: | KBIP7 | KBIP6 | KBIP5 | KBIP4 | KBIP3 | KBIP2 | KBIP1 | KBIP0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0456 | TIM2 Channel 2 Status and Control Register (T2SC2) See page 297. | Read: | CH2F | CH2IE | MS2B | MS2A | ELS2B | ELS2A | TOV2 | CH2MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0457 | TIM2 Channel 2 Register High (T2CH2H) See page 297. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0458 | TIM2 Channel 2 Register Low (T2CH2L) See page 297. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 9)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0459 | TIM2 Channel 3 Status and Control Register (T2SC3) See page 293. | Read: | CH3F | CH3IE | 0 | MS3A | ELS3B | ELS3A | TOV3 | CH3MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $045A | TIM2 Channel 3 Register High (T2CH3H) See page 297. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $045B | TIM2 Channel 3 Register Low (T2CH3L) See page 297. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $045C | TIM2 Channel 4 Status and Control Register (T2SC4) See page 293. | Read: | CH4F | CH4IE | MS4B | MS4A | ELS4B | ELS4A | TOV4 | CH4MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $045D | TIM2 Channel 4 Register High (T2CH4H) See page 297. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $045E | TIM2 Channel 4 Register Low (T2CH4L) See page 297. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $045F | TIM2 Channel 5 Status and Control Register (T2SC5) See page 293. | Read: | CH5F | CH5IE | 0 | MS5A | ELS5B | ELS5A | TOV 5 | CH5MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0460 | TIM2 Channel 5 Register High (T2CH5H) See page 297. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0461 | TIM2 Channel 5 Register Low (T2CH5L) See page 297. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $FE00 | Break Status Register (BSR) See page 237. | Read: | R | R | R | R | R | R | SBSW | R |
| | | Write: | | | | | | | NOTE 1 | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. Writing a 0 clears SBSW.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $FE01 | SIM Reset Status Register (SRSR) See page 237. | Read: | POR | PIN | COP | ILOP | ILAD | MODRST | LVI | 0 |
| | | Write: | | | | | | | | |
| | | POR: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE02 | Reserved | Read: | R | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 7 of 9)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FE03 | Break Flag Control Register (BFCR) See page 238. | Read: | BCFE | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE04 | Interrupt Status Register 1 (INT1) See page 231. | Read: | IF6 | IF5 | IF4 | IF3 | IF2 | IF1 | 0 | 0 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE05 | Interrupt Status Register 2 (INT2) See page 233. | Read: | IF14 | IF13 | IF12 | IF11 | IF10 | IF9 | IF8 | IF7 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE06 | Interrupt Status Register 3 (INT3) See page 233. | Read: | IF22 | IF21 | IF20 | IF19 | IF18 | IF17 | IF16 | IF15 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE07 | Interrupt Status Register 4 (INT4) See page 233. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | IF24 | IF23 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE08 | FLASH-2 Control Register (FL2CR) See page 53. | Read: | 0 | 0 | 0 | 0 | HVEN | MASS | ERASE | PGM |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE09 | Break Address Register High (BRKH) See page 303. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE0A | Break Address Register Low (BRKL) See page 303. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE0B | Break Status and Control Register (BRKSCR) See page 303. | Read: | BRKE | BRKA | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE0C | LVI Status Register (LVISR) See page 133. | Read: | LVIOUT | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE0D | FLASH-2 Test Control Register (FLTCR2) | Read: | R | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE0E | FLASH-1 Test Control Register (FLTCR1) | Read: | R | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 8 of 9)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FF80 | FLASH-1 Block Protect Register (FL1BPR)(1) See page 47. | Read: Write: | BPR7 | BPR6 | BPR5 | BPR4 | BPR3 | BPR2 | BPR1 | BPR0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $FF81 | FLASH-2 Block Protect Register (FL2BPR)(1) See page 54. | Read: Write: | BPR7 | BPR6 | BPR5 | BPR4 | BPR3 | BPR2 | BPR1 | BPR0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $FF88 | FLASH-1 Control Register (FL1CR) See page 46. | Read: | 0 | 0 | 0 | 0 | HVEN | MASS | ERASE | PGM |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FFFF | COP Control Register (COPCTL) See page 97. | Read: | Low byte of reset vector | | | | | | | |
| | | Write: | Writing clears COP counter (any value) | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |

1. Non-volatile FLASH register

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 9 of 9)**

**Table 2-1. Vector Addresses**

| Vector Priority | Vector | Address | Vector |
|---|---|---|---|
| Lowest ↑ | IF24 | $FFCC | TIM2 Channel 5 Vector (High) |
| | | $FFCD | TIM2 Channel 5 Vector (Low) |
| | IF23 | $FFCE | TIM2 Channel 4 Vector (High) |
| | | $FFCF | TIM2 Channel 4 Vector (Low) |
| | IF22 | $FFD0 | TIM2 Channel 3 Vector (High) |
| | | $FFD1 | TIM2 Channel 3 Vector (Low) |
| | IF21 | $FFD2 | TIM2 Channel 2 Vector (High) |
| | | $FFD3 | TIM2 Channel 2 Vector (Low) |
| | IF20 | $FFD4 | MSCAN08 Transmit Vector (High) |
| | | $FFD5 | MSCAN08 Transmit Vector (Low) |
| | IF19 | $FFD6 | MSCAN08 Receive Vector (High) |
| | | $FFD7 | MSCAN08 Receive Vector (Low) |
| | IF18 | $FFD8 | MSCAN08 Error Vector (High) |
| | | $FFD9 | MSCAN08 Error Vector (Low) |
| | IF17 | $FFDA | MSCAN08 Wakeup Vector (High) |
| | | $FFDB | MSCAN08 Wakeup Vector (Low) |
| | IF16 | $FFDC | Timebase Vector (High) |
| | | $FFDD | Timebase Vector (Low) |

**Table 2-1. Vector Addresses (Continued)**

| Vector Priority | Vector | Address | Vector |
|---|---|---|---|
| | IF15 | $FFDE | ADC Conversion Complete Vector (High) |
| | | $FFDF | ADC Conversion Complete Vector (Low) |
| | IF14 | $FFE0 | Keyboard Vector (High) |
| | | $FFE1 | Keyboard Vector (Low) |
| | IF13 | $FFE2 | ESCI Transmit Vector (High) |
| | | $FFE3 | ESCI Transmit Vector (Low) |
| | IF12 | $FFE4 | ESCI Receive Vector (High) |
| | | $FFE5 | ESCI Receive Vector (Low) |
| | IF11 | $FFE6 | ESCI Error Vector (High) |
| | | $FFE7 | ESCI Error Vector (Low) |
| | IF10 | $FFE8 | SPI Transmit Vector (High) |
| | | $FFE9 | SPI Transmit Vector (Low) |
| | IF9 | $FFEA | SPI Receive Vector (High) |
| | | $FFEB | SPI Receive Vector (Low) |
| | IF8 | $FFEC | TIM2 Overflow Vector (High) |
| | | $FFED | TIM2 Overflow Vector (Low) |
| | IF7 | $FFEE | TIM2 Channel 1 Vector (High) |
| | | $FFEF | TIM2 Channel 1 Vector (Low) |
| | IF6 | $FFF0 | TIM2 Channel 0 Vector (High) |
| | | $FFF1 | TIM2 Channel 0 Vector (Low) |
| | IF5 | $FFF2 | TIM1 Overflow Vector (High) |
| | | $FFF3 | TIM1 Overflow Vector (Low) |
| | IF4 | $FFF4 | TIM1 Channel 1 Vector (High) |
| | | $FFF5 | TIM1 Channel 1 Vector (Low) |
| | IF3 | $FFF6 | TIM1 Channel 0 Vector (High) |
| | | $FFF7 | TIM1 Channel 0 Vector (Low) |
| | IF2 | $FFF8 | PLL Vector (High) |
| | | $FFF9 | PLL Vector (Low) |
| | IF1 | $FFFA | $\overline{\text{IRQ}}$ Vector (High) |
| | | $FFFB | $\overline{\text{IRQ}}$ Vector (Low) |
| | — | $FFFC | SWI Vector (High) |
| | | $FFFD | SWI Vector (Low) |
| Highest | — | $FFFE | Reset Vector (High) |
| | | $FFFF | Reset Vector (Low) |

## 2.5 Random-Access Memory (RAM)

The RAM locations are broken into two non-continuous memory blocks. The RAM addresses locations are $0040–$043F and $0580–$097F. The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in the 64-Kbyte memory space.

***NOTE***

*For correct operation, the stack pointer must point only to RAM locations.*

Within page zero are 192 bytes of RAM. Because the location of the stack RAM is programmable, all page zero RAM locations can be used for I/O control and user data or code. When the stack pointer is moved from its reset location at $00FF out of page zero, direct addressing mode instructions can efficiently access all page zero RAM locations. Page zero RAM, therefore, provides ideal locations for frequently accessed global variables.

Before processing an interrupt, the CPU uses five bytes of the stack to save the contents of the CPU registers.

***NOTE***

*For M6805 compatibility, the H register is not stacked.*

During a subroutine call, the CPU uses two bytes of the stack to store the return address. The stack pointer decrements during pushes and increments during pulls.

***NOTE***

*Be careful when using nested subroutines. The CPU may overwrite data in the RAM during a subroutine or during the interrupt stacking operation.*

## 2.6 FLASH-1 Memory (FLASH-1)

This subsection describes the operation of the embedded FLASH-1 memory. This memory can be read, programmed, and erased from a single external supply. The program and erase operations are enabled through the use of an internal charge pump.

### 2.6.1 Functional Description

The FLASH-1 memory is an array of 32,256 bytes with two bytes of block protection (one byte for protecting areas within FLASH-1 array and one byte for protecting areas within FLASH-2 array) and an additional 52 bytes of user vectors. An erased bit reads as a 1 and a programmed bit reads as a 0.

Memory in the FLASH-1 array is organized into rows within pages. There are two rows of memory per page with 64 bytes per row. The minimum erase block size is a single page,128 bytes. Programming is performed on a per-row basis, 64 bytes at a time. Program and erase operations are facilitated through control bits in the FLASH-1 control register (FL1CR). Details for these operations appear later in this subsection.

The FLASH-1 memory map consists of:

- $8000–$FDFF: user memory (32,256 bytes)
- $FF80: FLASH-1 block protect register (FL1BPR)
- $FF81: FLASH-2 block protect register (FL2BPR)
- $FF88: FLASH-1 control register (FL1CR)
- $FFCC–$FFFF: these locations are reserved for user-defined interrupt and reset vectors (see Table 2-1 for details)

Programming tools are available from Freescale. Contact your local Freescale representative for more information.

***NOTE***

*A security feature prevents viewing of the FLASH contents.*[1]

## 2.6.2 FLASH-1 Control and Block Protect Registers

The FLASH-1 array has two registers that control its operation, the FLASH-1 control register (FL1CR) and the FLASH-1 block protect register (FL1BPR).

### *2.6.2.1 FLASH-1 Control Register*

The FLASH-1 control register (FL1CR) controls FLASH program and erase operations.

Address: $FF88

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | HVEN | MASS | ERASE | PGM |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 2-3. FLASH-1 Control Register (FL1CR)**

**HVEN — High-Voltage Enable Bit**

This read/write bit enables the charge pump to drive high voltages for program and erase operations in the array. HVEN can only be set if either PGM = 1 or ERASE = 1 and the proper sequence for program or erase is followed.

1 = High voltage enabled to array and charge pump on
0 = High voltage disabled to array and charge pump off

**MASS — Mass Erase Control Bit**

Setting this read/write bit configures the FLASH-1 array for mass erase operation.

1 = MASS erase operation selected
0 = MASS erase operation unselected

**ERASE — Erase Control Bit**

This read/write bit configures the memory for erase operation. ERASE is interlocked with the PGM bit such that both bits cannot be equal to 1 or set to 1 at the same time.

1 = Erase operation selected
0 = Erase operation unselected

**PGM — Program Control Bit**

This read/write bit configures the memory for program operation. PGM is interlocked with the ERASE bit such that both bits cannot be equal to 1 or set to 1 at the same time.

1 = Program operation selected
0 = Program operation unselected

1. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH difficult for unauthorized users.

### 2.6.2.2 FLASH-1 Block Protect Register

The FLASH-1 block protect register (FL1BPR) is implemented as a byte within the FLASH-1 memory; therefore, it can only be written during a FLASH programming sequence. The value in this register determines the starting location of the protected range within the FLASH-1 memory.

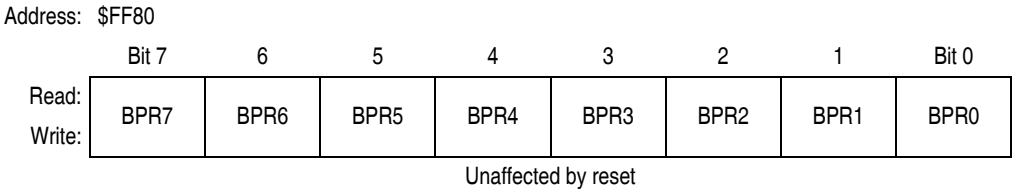



**Figure 2-4. FLASH-1 Block Protect Register (FL1BPR)**

**FL1BPR[7:0] — Block Protect Register Bits 7 to 0**

These eight bits represent bits [14:7] of a 16-bit memory address. Bit 15 is a 1 and bits [6:0] are 0s.

The resultant 16-bit address is used for specifying the start address of the FLASH-1 memory for block protection. FLASH-1 is protected from this start address to the end of FLASH-1 memory at $FFFF. With this mechanism, the protect start address can be $XX00 and $XX80 (128 byte page boundaries) within the FLASH-1 array.

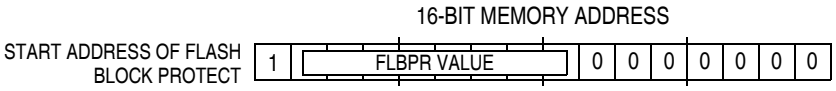



**Figure 2-5. FLASH-1 Block Protect Start Address**

**Table 2-2. FLASH-1 Protected Ranges**

| FL1BPR[7:0] | Protected Range |
|---|---|
| $FF | No protection |
| $FE | $FF00–$FFFF |
| $FD<br>↓<br>$0B | $FE80–$FFFF<br>↓<br>$8580–$FFFF |
| $0A | $8500–$FFFF |
| $09 | $8480–$FFFF |
| $08<br>↓<br>$04 | $8400–$FFFF<br>↓<br>$8200–$FFFF |
| $03 | $8180–$FFFF |
| $02 | $8100–$FFFF |
| $01 | $8080–$FFFF |
| $00 | $8000–$FFFF |

Decreasing the value in FL1BPR by one increases the protected range by one page (128 bytes). However, programming the block protect register with $FE protects a range twice that size, 256 bytes, in the corresponding array. $FE means that locations $FF00–$FFFF are protected in FLASH-1.

The FLASH memory does not exist at some locations. The block protection range configuration is unaffected if FLASH memory does not exist in that range. Refer to Figure 2-1 and make sure that the desired locations are protected.

### 2.6.3 FLASH-1 Block Protection

Due to the ability of the on-board charge pump to erase and program the FLASH memory in the target application, provision is made for protecting blocks of memory from unintentional erase or program operations due to system malfunction. This protection is done by using the FLASH-1 block protection register (FL1BPR). FL1BPR determines the range of the FLASH-1 memory which is to be protected. The range of the protected area starts from a location defined by FL1BPR and ends at the bottom of the FLASH-1 memory ($FFFF). When the memory is protected, the HVEN bit can not be set in either ERASE or PROGRAM operations.

***NOTE***

*In performing a program or erase operation, the FLASH-1 block protect register must be read after setting the PGM or ERASE bit and before asserting the HVEN bit.*

When the FLASH-1 block protect register is programmed with all 0's, the entire memory is protected from being programmed and erased. When all the bits are erased (all 1's), the entire memory is accessible for program and erase.

When bits within FL1BPR are programmed (0), they lock a block of memory address ranges as shown in Figure 2-4. If FL1BPR is programmed with any value other than $FF, the protected block of FLASH memory can not be erased or programmed.

***NOTE***

*The vector locations and the FLASH block protect registers are located in the same page. FL1BPR and FL2BPR are not protected with special hardware or software. Therefore, if this page is not protected by FL1BPR and the vector locations are erased by either a page or a mass erase operation, then both FL1BPR and FL2BPR will also get erased.*

### 2.6.4 FLASH-1 Mass Erase Operation

Use this step-by-step procedure to erase the entire FLASH-1 memory:

1. Set both the ERASE bit and the MASS bit in the FLASH-1 control register (FL1CR).
2. Read the FLASH-1 block protect register (FL1BPR).

***NOTE***

*Mass erase is disabled whenever any block is protected (FL1BPR does not equal $FF).*

3. Write to any FLASH-1 address within the FLASH-1 array with any data.
4. Wait for a time, $t_{NVS}$ (minimum 10 μs).
5. Set the HVEN bit.
6. Wait for a time, $t_{MERASE}$ (minimum 4 ms).
7. Clear the ERASE and MASS bits.
8. Wait for a time, $t_{NVHL}$ (minimum 100 μs).
9. Clear the HVEN bit.
10. Wait for a time, $t_{RCV}$, (typically 1 μs) after which the memory can be accessed in normal read mode.

***NOTES***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. However, care must be taken to ensure that these operations do not access any address within the FLASH array memory space such as the COP control register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

### 2.6.5 FLASH-1 Page Erase Operation

Use this step-by-step procedure to erase a page (128 bytes) of FLASH-1 memory:

1. Set the ERASE bit and clear the MASS bit in the FLASH-1 control register (FL1CR).
2. Read the FLASH-1 block protect register (FL1BPR).
3. Write any data to any FLASH-1 address within the address range of the page (128 byte block) to be erased.
4. Wait for time, $t_{NVS}$ (minimum 10 μs).
5. Set the HVEN bit.
6. Wait for time, $t_{ERASE}$ (minimum 1 ms or 4 ms).
7. Clear the ERASE bit.
8. Wait for time, $t_{NVH}$ (minimum 5 μs).
9. Clear the HVEN bit.
10. Wait for a time, $t_{RCV}$, (typically 1 μs) after which the memory can be accessed in normal read mode.

***NOTES***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. However, care must be taken to ensure that these operations do not access any address within the FLASH array memory space such as the COP control register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

In applications that require more than 1000 program/erase cycles, use the 4 ms page erase specification to get improved long-term reliability. Any application can use this 4 ms page erase specification. However, in applications where a FLASH location will be erased and reprogrammed less than 1000 times, and speed is important, use the 1 ms page erase specification to get a shorter cycle time.

### 2.6.6 FLASH-1 Program Operation

Programming of the FLASH-1 memory is done on a row basis. A row consists of 64 consecutive bytes with address ranges as follows:

- $XX00 to $XX3F
- $XX40 to $XX7F
- $XX80 to $XXBF
- $XXC0 to $XXFF

During the programming cycle, make sure that all addresses being written to fit within one of the ranges specified above. Attempts to program addresses in different row ranges in one programming cycle will fail.

Use this step-by-step procedure to program a row of FLASH-1 memory.

***NOTE***

*Only bytes which are currently $FF may be programmed.*

1. Set the PGM bit in the FLASH-1 control register (FL1CR). This configures the memory for program operation and enables the latching of address and data programming.
2. Read the FLASH-1 block protect register (FL1BPR).
3. Write to any FLASH-1 address within the row address range desired with any data.
4. Wait for time, $t_{NVS}$ (minimum 10 μs).
5. Set the HVEN bit.
6. Wait for time, $t_{PGS}$ (minimum 5 μs).
7. Write data byte to the FLASH-1 address to be programmed.
8. Wait for time, $t_{PROG}$ (minimum 30 μs).
9. Repeat steps 7 and 8 until all the bytes within the row are programmed.
10. Clear the PGM bit.
11. Wait for time, $t_{NVH}$ (minimum 5 μs)
12. Clear the HVEN bit.
13. Wait for a time, $t_{RCV}$, (typically 1 μs) after which the memory can be accessed in normal read mode.

The FLASH programming algorithm flowchart is shown in Figure 2-6.

***NOTES***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. However, care must be taken to ensure that these operations do not access any address within the FLASH array memory space such as the COP control register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

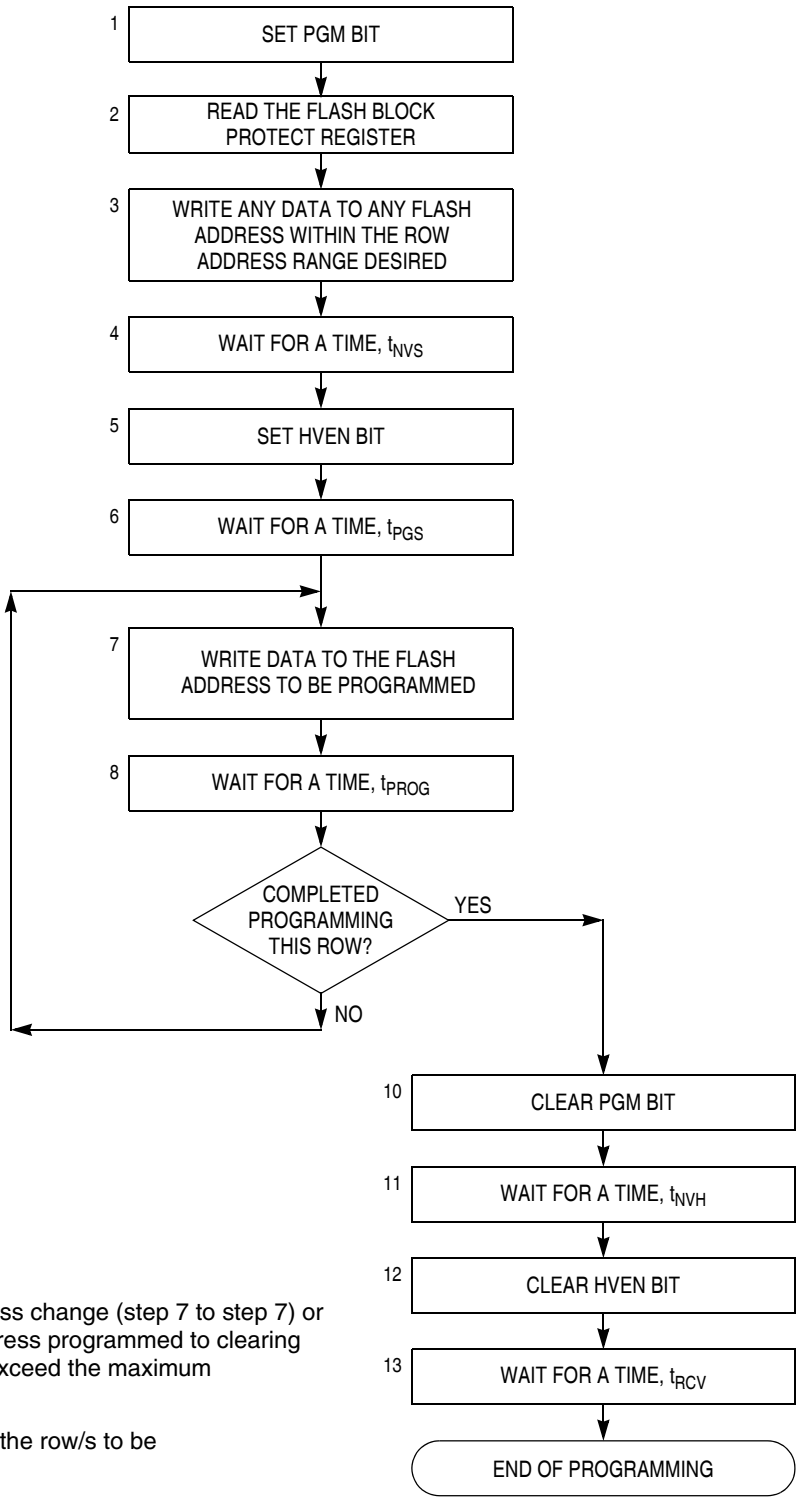
***D.*** *Do not exceed $t_{PROG}$ maximum or $t_{HV}$ maximum. $t_{HV}$ is defined as the cumulative high voltage programming time to the same row before next erase. $t_{HV}$ must satisfy this condition:*

$$t_{NVS} + t_{NVH} + t_{PGS} + (t_{PROG} \times 64) \leq t_{HV} \text{ maximum}$$

***E.*** *The time between each FLASH address change (step 7 to step 7), or the time between the last FLASH address programmed to clearing the PGM bit (step 7 to step 10) must not exceed the maximum programming time, $t_{PROG}$ maximum.*

***F.*** *Be cautious when programming the FLASH-1 array to ensure that non-FLASH locations are not used as the address that is written to when selecting either the desired row address range in step 3 of the algorithm or the byte to be programmed in step 7 of the algorithm.*

**Algorithm for programming a row (64 bytes) of FLASH memory**




NOTES:

The time between each FLASH address change (step 7 to step 7) or the time between the last FLASH address programmed to clearing PGM bit (step 7 to step10) must not exceed the maximum programming time, $t_{PROG}$, maximum.

This row program algorithm assumes the row/s to be programmed are initially erased.

**Figure 2-6. FLASH-1 Programming Algorithm Flowchart**

### 2.6.7 Low-Power Modes

The WAIT and STOP instructions will place the MCU in low power-consumption standby modes.

#### *2.6.7.1 Wait Mode*

Putting the MCU into wait mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH. Wait mode will suspend any FLASH program/erase operations and leave the memory in a standby mode.

#### *2.6.7.2 Stop Mode*

Putting the MCU into stop mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH. Stop mode will suspend any FLASH program/erase operations and leave the memory in a standby mode.

***NOTE***

*Standby mode is the power saving mode of the FLASH module, in which all internal control signals to the FLASH are inactive and the current consumption of the FLASH is minimum.*

## 2.7 FLASH-2 Memory (FLASH-2)

This subsection describes the operation of the embedded FLASH-2 memory. This memory can be read, programmed, and erased from a single external supply. The program and erase operations are enabled through the use of an internal charge pump.

### 2.7.1 Functional Description

The FLASH-2 memory is a non-continuous array consisting of a total of 29,822 bytes. An erased bit reads as a 1 and a programmed bit reads as a 0.

Memory in the FLASH-2 array is organized into rows within pages. There are two rows of memory per page with 64 bytes per row. The minimum erase block size is a single page,128 bytes. Programming is performed on a per-row basis, 64 bytes at a time. Program and erase operations are facilitated through control bits in the FLASH-2 control register (FL2CR). Details for these operations appear later in this subsection.

**The FLASH-2 memory map consists of:**

- $0462–$04FF: user memory (158 bytes)
- $0980–$1B7F: user memory (4608 bytes)
- $1E20–$7FFF: user memory (25056 bytes)
- $FF81: FLASH-2 block protect register (FL2BPR)

***NOTE***

*FL2BPR physically resides within FLASH-1 memory addressing space*

- $FE08: FLASH-2 control register (FL2CR)

Programming tools are available from Freescale. Contact your local Freescale representative for more information.

***NOTE***

*A security feature prevents viewing of the FLASH contents.*[1]

## 2.7.2 FLASH-2 Control and Block Protect Registers

The FLASH-2 array has two registers that control its operation, the FLASH-2 control register (FL2CR) and the FLASH-2 block protect register (FL2BPR).

### *2.7.2.1 FLASH-2 Control Register*

The FLASH-2 control register (FL2CR) controls FLASH-2 program and erase operations.

Address: $FE08

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | HVEN | MASS | ERASE | PGM |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 2-7. FLASH-2 Control Register (FL2CR)**

**HVEN — High-Voltage Enable Bit**

This read/write bit enables the charge pump to drive high voltages for program and erase operations in the array. HVEN can only be set if either PGM = 1 or ERASE = 1 and the proper sequence for program or erase is followed.

1 = High voltage enabled to array and charge pump on
0 = High voltage disabled to array and charge pump off

**MASS — Mass Erase Control Bit**

Setting this read/write bit configures the FLASH-2 array for mass or page erase operation.

1 = Mass erase operation selected
0 = Page erase operation selected

**ERASE — Erase Control Bit**

This read/write bit configures the memory for erase operation. ERASE is interlocked with the PGM bit such that both bits cannot be set at the same time.

1 = Erase operation selected
0 = Erase operation unselected

**PGM — Program Control Bit**

This read/write bit configures the memory for program operation. PGM is interlocked with the ERASE bit such that both bits cannot be equal to 1 or set to 1 at the same time.

1 = Program operation selected
0 = Program operation unselected

1. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH difficult for unauthorized users.

### 2.7.2.2 FLASH-2 Block Protect Register

The FLASH-2 block protect register (FL2BPR) is implemented as a byte within the FLASH-1 memory; therefore, can only be written during a FLASH-1 programming sequence. The value in this register determines the starting location of the protected range within the FLASH-2 memory.

Address: $FF81

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | BPR7 | BPR6 | BPR5 | BPR4 | BPR3 | BPR2 | BPR1 | BPR0 |
| Reset: | Unaffected by reset | | | | | | | |

**Figure 2-8. FLASH-2 Block Protect Register (FL2BPR)**

***NOTE***

*The FLASH-2 block protect register (FL2BPR) controls the block protection for the FLASH-2 array. However, FL2BPR is implemented within the FLASH-1 memory array and therefore, the FLASH-1 control register (FL1CR) must be used to program/erase FL2BPR.*

**FL2BPR[7:0] — Block Protect Register Bits 7 to 0**

These eight bits represent bits [14:7] of a 16-bit memory address. Bit 15 is a 0 and bits [6:0] are 0s.

The resultant 16-bit address is used for specifying the start address of the FLASH-2 memory for block protection. FLASH-2 is protected from this start address to the end of FLASH-2 memory at $7FFF. With this mechanism, the protect start address can be $XX00 and $XX80 (128 byte page boundaries) within the FLASH-2 array.

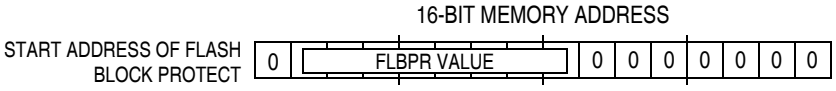



**Figure 2-9. FLASH-2 Block Protect Start Address**

**Table 2-3. FLASH-2 Protected Ranges**

| FL2BPR[7:0] | Protected Range |
|---|---|
| $FF | No Protection |
| $FE | $7F00–$7FFF |
| $FD<br>↓<br>$0B | $7E80–$7FFF<br>↓<br>$0580–$7FFF |
| $0A | $0500–$7FFF |
| $09 | $0480–$7FFF |
| $08<br>↓<br>$04 | $0462–$7FFF<br>↓<br>$0462–$7FFF |
| $03 | $0462–$7FFF |
| $02 | $0462–$7FFF |
| $01 | $0462–$7FFF |
| $00 | $0462–$7FFF |

Decreasing the value in FL2BPR by one increases the protected range by one page (128 bytes). However, programming the block protect register with $FE protects a range twice that size, 256 bytes, in the corresponding array. $FE means that locations $7F00–$7FFF are protected in FLASH-2.

The FLASH memory does not exist at some locations. The block protection range configuration is unaffected if FLASH memory does not exist in that range. Refer to Figure 2-1 and make sure that the desired locations are protected.

### 2.7.3 FLASH-2 Block Protection

Due to the ability of the on-board charge pump to erase and program the FLASH memory in the target application, provision is made for protecting blocks of memory from unintentional erase or program operations due to system malfunction. This protection is done by using the FLASH-2 block protection register (FL2BPR). FL2BPR determines the range of the FLASH-2 memory which is to be protected. The range of the protected area starts from a location defined by FL2BPR and ends at the bottom of the FLASH-2 memory ($7FFF). When the memory is protected, the HVEN bit can not be set in either ERASE or PROGRAM operations.

***NOTE***

*In performing a program or erase operation, the FLASH-2 block protect register must be read after setting the PGM or ERASE bit and before asserting the HVEN bit.*

When the FLASH-2 block protect register is programmed with all 0's, the entire memory is protected from being programmed and erased. When all the bits are erased (all 1's), the entire memory is accessible for program and erase.

When bits within FL2BPR are programmed (0), they lock a block of memory address ranges as shown in 2.7.2.2 FLASH-2 Block Protect Register. If FL2BPR is programmed with any value other than $FF, the protected block of FLASH memory can not be erased or programmed.

***NOTE***

*The vector locations and the FLASH block protect registers are located in the same page. FL1BPR and FL2BPR are not protected with special hardware or software. Therefore, if this page is not protected by FL1BPR and the vector locations are erased by either a page or a mass erase operation, both FL1BPR and FL2BPR will also get erased.*

### 2.7.4 FLASH-2 Mass Erase Operation

Use this step-by-step procedure to erase the entire FLASH-2 memory:

1. Set both the ERASE bit and the MASS bit in the FLASH-2 control register (FL2CR).
2. Read the FLASH-2 block protect register (FL2BPR).

***NOTE***

*Mass erase is disabled whenever any block is protected (FL2BPR does not equal $FF).*

3. Write to any FLASH-2 address within the FLASH-2 array with any data.
4. Wait for a time, $t_{NVS}$ (minimum 10 μs).
5. Set the HVEN bit.
6. Wait for a time, $t_{MERASE}$ (minimum 4 ms).
7. Clear the ERASE and MASS bits.

8. Wait for a time, $t_{NVHL}$ (minimum 100 μs).
9. Clear the HVEN bit.
10. Wait for a time, $t_{RCV}$, (typically 1 μs) after which the memory can be accessed in normal read mode.

***NOTES***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. However, care must be taken to ensure that these operations do not access any address within the FLASH array memory space such as the COP control register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

### 2.7.5 FLASH-2 Page Erase Operation

Use this step-by-step procedure to erase a page (128 bytes) of FLASH-2 memory:

1. Set the ERASE bit and clear the MASS bit in the FLASH-2 control register (FL2CR).
2. Read the FLASH-2 block protect register (FL2BPR).
3. Write any data to any FLASH-2 address within the address range of the page (128 byte block) to be erased.
4. Wait for time, $t_{NVS}$ (minimum 10 μs).
5. Set the HVEN bit.
6. Wait for time, $t_{ERASE}$ (minimum 1 ms or 4 ms).
7. Clear the ERASE bit.
8. Wait for time, $t_{NVH}$ (minimum 5 μs).
9. Clear the HVEN bit.
10. Wait for a time, $t_{RCV}$, (typically 1 μs) after which the memory can be accessed in normal read mode.

***NOTES***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. However, care must be taken to ensure that these operations do not access any address within the FLASH array memory space such as the COP control register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

In applications that require more than 1000 program/erase cycles, use the 4 ms page erase specification to get improved long-term reliability. Any application can use this 4 ms page erase specification. However, in applications where a FLASH location will be erased and reprogrammed less than 1000 times, and speed is important, use the 1 ms page erase specification to get a shorter cycle time.

### 2.7.6 FLASH-2 Program Operation

Programming of the FLASH memory is done on a row basis. A row consists of 64 consecutive bytes with address ranges as follows:

- $XX00 to $XX3F
- $XX40 to $XX7F
- $XX80 to $XXBF
- $XXC0 to $XXFF

During the programming cycle, make sure that all addresses being written to fit within one of the ranges specified above. Attempts to program addresses in different row ranges in one programming cycle will fail.

***NOTE***

*Only bytes which are currently $FF may be programmed.*

Use this step-by-step procedure to program a row of FLASH-2 memory:

1. Set the PGM bit in the FLASH-2 control register (FL2CR). This configures the memory for program operation and enables the latching of address and data programming.
2. Read the FLASH-2 block protect register (FL2BPR).
3. Write to any FLASH-2 address within the row address range desired with any data.
4. Wait for time, $t_{NVS}$ (minimum 10 μs).
5. Set the HVEN bit.
6. Wait for time, $t_{PGS}$ (minimum 5 μs).
7. Write data byte to the FLASH-2 address to be programmed.
8. Wait for time, $t_{PROG}$ (minimum 30 μs).
9. Repeat step 7 and 8 until all the bytes within the row are programmed.
10. Clear the PGM bit.
11. Wait for time, $t_{NVH}$ (minimum 5 μs).
12. Clear the HVEN bit.
13. Wait for a time, $t_{RCV}$, (typically 1 μs) after which the memory can be accessed in normal read mode.

The FLASH programming algorithm flowchart is shown in Figure 2-10.

***NOTES***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. However, care must be taken to ensure that these operations do not access any address within the FLASH array memory space such as the COP control register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

***D.*** *Do not exceed $t_{PROG}$ maximum or $t_{HV}$ maximum. $t_{HV}$ is defined as the cumulative high voltage programming time to the same row before next erase. $t_{HV}$ must satisfy this condition:*

$$t_{NVS} + t_{NVH} + t_{PGS} + (t_{PROG} \times 64) \leq t_{HV} \text{ maximum}$$

***E.*** *The time between each FLASH address change (step 7 to step 7), or the time between the last FLASH address programmed to clearing the PGM bit (step 7 to step 10) must not exceed the maximum programming time, $t_{PROG}$ maximum.*

***F.*** *Be cautious when programming the FLASH-2 array to ensure that non-FLASH locations are not used as the address that is written to when selecting either the desired row address range in step 3 of the algorithm or the byte to be programmed in step 7 of the algorithm.*

## 2.7.7 Low-Power Modes

The WAIT and STOP instructions will place the MCU in low power-consumption standby modes.

### *2.7.7.1 Wait Mode*

Putting the MCU into wait mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH. Wait mode will suspend any FLASH program/erase operations and leave the memory in a standby mode.

### *2.7.7.2 Stop Mode*

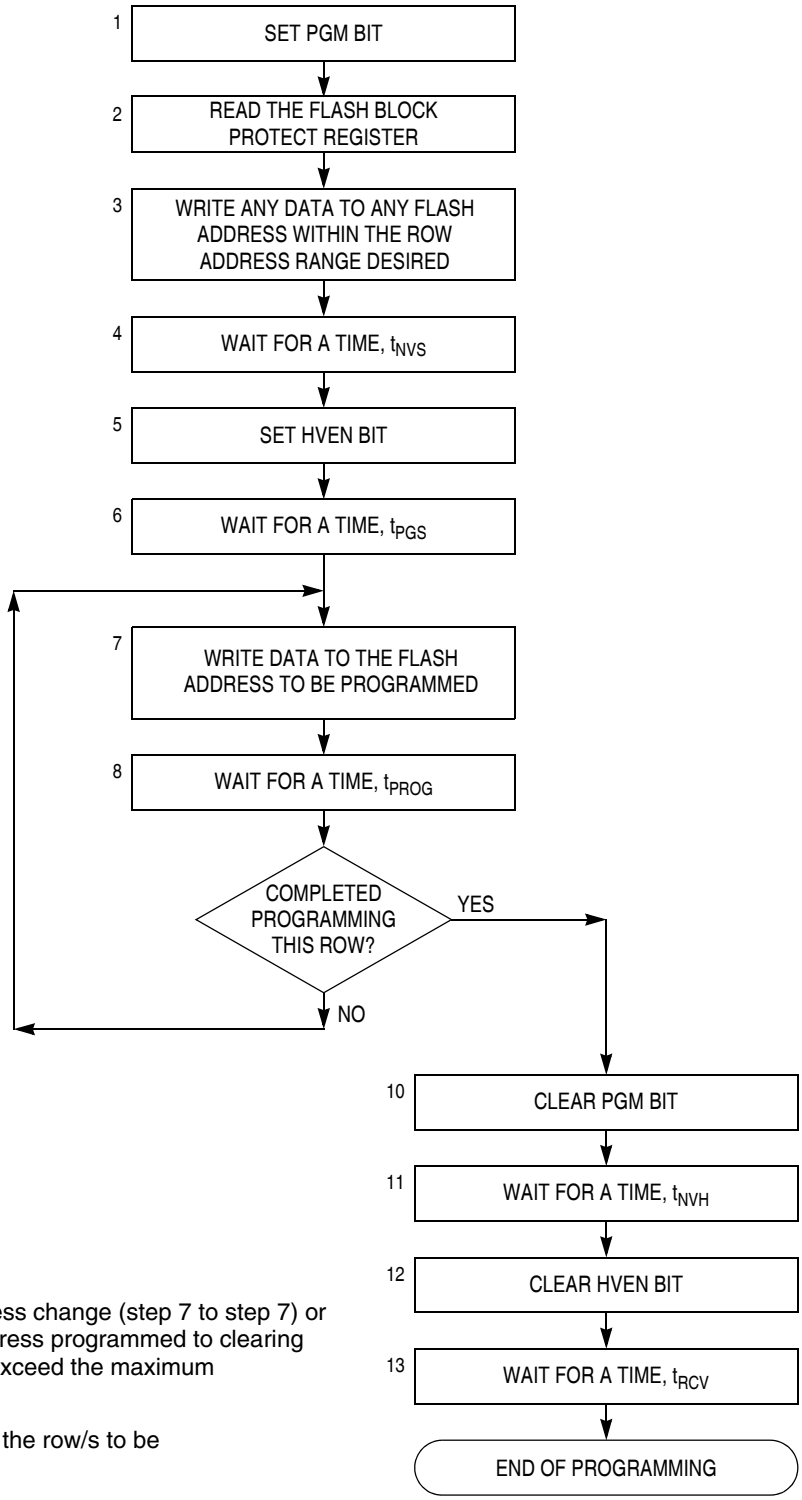
Putting the MCU into stop mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH. Stop mode will suspend any FLASH program/erase operations and leave the memory in a standby mode.

***NOTE***

*Standby mode is the power saving mode of the FLASH module, in which all internal control signals to the FLASH are inactive and the current consumption of the FLASH is minimum.*

**Algorithm for programming a row (64 bytes) of FLASH memory**




NOTES:

The time between each FLASH address change (step 7 to step 7) or the time between the last FLASH address programmed to clearing PGM bit (step 7 to step10) must not exceed the maximum programming time, $t_{PROG}$, maximum.

This row program algorithm assumes the row/s to be programmed are initially erased.

**Figure 2-10. FLASH-2 Programming Algorithm Flowchart**

# Chapter 3
# Analog-to-Digital Converter (ADC)

## 3.1 Introduction

This section describes the 10-bit analog-to-digital converter (ADC).

## 3.2 Features

Features of the ADC module include:

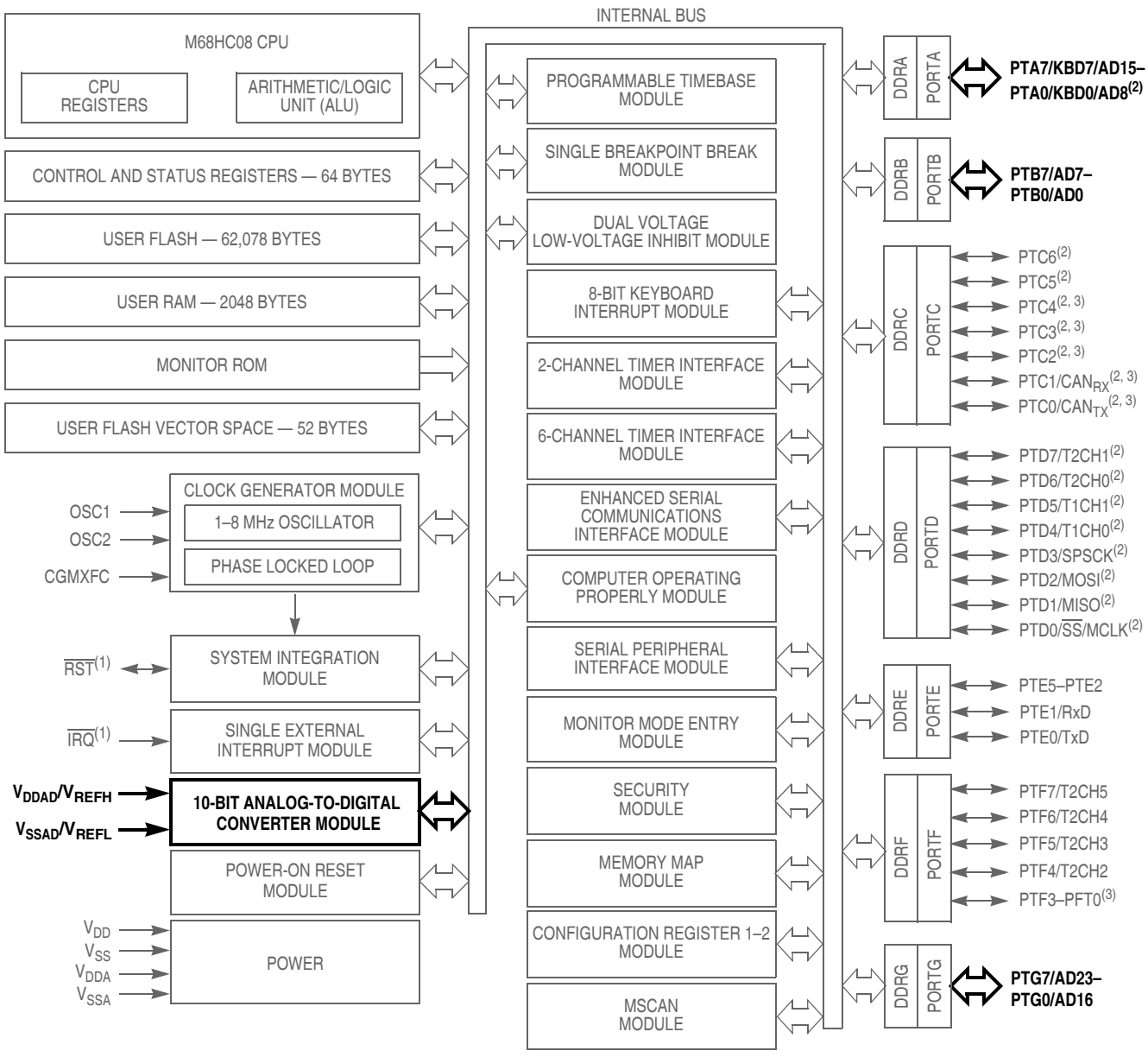
- 24 channels with multiplexed input
- Linear successive approximation with monotonicity
- 10-bit resolution
- Single or continuous conversion
- Conversion complete flag or conversion complete interrupt
- Selectable ADC clock
- Left or right justified result
- Left justified sign data mode

## 3.3 Functional Description

The ADC provides 24 pins for sampling external sources at pins PTG7/AD23–PTG0/AD16, PTA7/KBD7/AD15–PTA0/KBD0/AD8, and PTB7/AD7–PTB0/AD0. An analog multiplexer allows the single ADC converter to select one of 24 ADC channels as ADC voltage in ($V_{ADIN}$). $V_{ADIN}$ is converted by the successive approximation register-based analog-to-digital converter. When the conversion is completed, ADC places the result in the ADC data register and sets a flag or generates an interrupt. See Figure 3-2.

### 3.3.1 ADC Port I/O Pins

PTG7/AD23–PTG0/AD16, PTA7/KBD7/AD15–PTA0/KBD0/AD8, and PTB7/AD7–PTB0/AD0 are general-purpose I/O (input/output) pins that share with the ADC channels. The channel select bits define which ADC channel/port pin will be used as the input signal. The ADC overrides the port I/O logic by forcing that pin as input to the ADC. The remaining ADC channels/port pins are controlled by the port I/O logic and can be used as general-purpose I/O. Writes to the port register or data direction register (DDR) will not have any affect on the port pin that is selected by the ADC. A read of a port pin in use by the ADC will return a 0.




1. Pin contains integrated pullup device.
**2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.**
3. Higher current drive port pins

**Figure 3-1. Block Diagram Highlighting ADC Block and Pins**

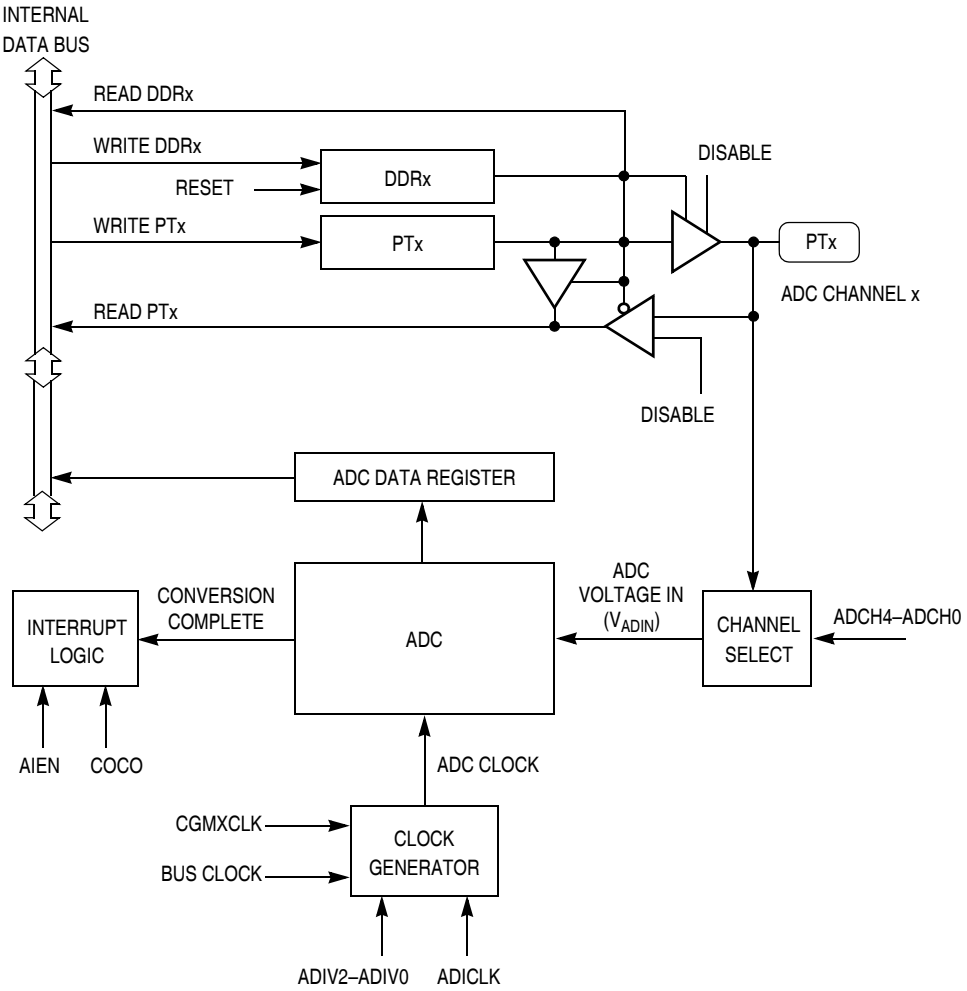



**Figure 3-2. ADC Block Diagram**

### 3.3.2 Voltage Conversion

When the input voltage to the ADC equals $V_{REFH}$, the ADC converts the signal to $3FF (full scale). If the input voltage equals $V_{REFL}$, the ADC converts it to $000. Input voltages between $V_{REFH}$ and $V_{REFL}$ are a straight-line linear conversion.

***NOTE***

*The ADC input voltage must always be greater than $V_{SSAD}$ and less than $V_{DDAD}$.*

*Connect the $V_{DDAD}$ pin to the same voltage potential as the $V_{DD}$ pin, and connect the $V_{SSAD}$ pin to the same voltage potential as the $V_{SS}$ pin.*

*The $V_{DDAD}$ pin should be routed carefully for maximum noise immunity.*

### 3.3.3 Conversion Time

Conversion starts after a write to the ADC status and control register (ADSCR). One conversion will take between 16 and 17 ADC clock cycles. The ADIVx and ADICLK bits should be set to provide a 1-MHz ADC clock frequency.

$$\text{Conversion time} = \frac{\text{16 to 17 ADC cycles}}{\text{ADC frequency}}$$

$$\text{Number of bus cycles} = \text{conversion time} \times \text{bus frequency}$$

### 3.3.4 Conversion

In continuous conversion mode, the ADC data register will be filled with new data after each conversion. Data from the previous conversion will be overwritten whether that data has been read or not. Conversions will continue until the ADCO bit is cleared. The COCO bit is set after each conversion and will stay set until the next read of the ADC data register.

In single conversion mode, conversion begins with a write to the ADSCR. Only one conversion occurs between writes to the ADSCR.

When a conversion is in process and the ADSCR is written, the current conversion data should be discarded to prevent an incorrect reading.

### 3.3.5 Accuracy and Precision

The conversion process is monotonic and has no missing codes.

### 3.3.6 Result Justification

The conversion result may be formatted in four different ways:

1. Left justified
2. Right justified
3. Left Justified sign data mode
4. 8-bit truncation mode

All four of these modes are controlled using MODE0 and MODE1 bits located in the ADC clock register (ADCLK).

Left justification will place the eight most significant bits (MSB) in the corresponding ADC data register high, ADRH. This may be useful if the result is to be treated as an 8-bit result where the two least significant bits (LSB), located in the ADC data register low, ADRL, can be ignored. However, ADRL must be read after ADRH or else the interlocking will prevent all new conversions from being stored.

Right justification will place only the two MSBs in the corresponding ADC data register high, ADRH, and the eight LSBs in ADC data register low, ADRL. This mode of operation typically is used when a 10-bit unsigned result is desired.

Left justified sign data mode is similar to left justified mode with one exception. The MSB of the 10-bit result, AD9 located in ADRH, is complemented. This mode of operation is useful when a result, represented as a signed magnitude from mid-scale, is needed. Finally, 8-bit truncation mode will place the eight MSBs in the ADC data register low, ADRL. The two LSBs are dropped. This mode of operation is used when compatibility with 8-bit ADC designs are required. No interlocking between ADRH and ADRL is present.

***NOTE***

*Quantization error is affected when only the most significant eight bits are used as a result. See Figure 3-3.*

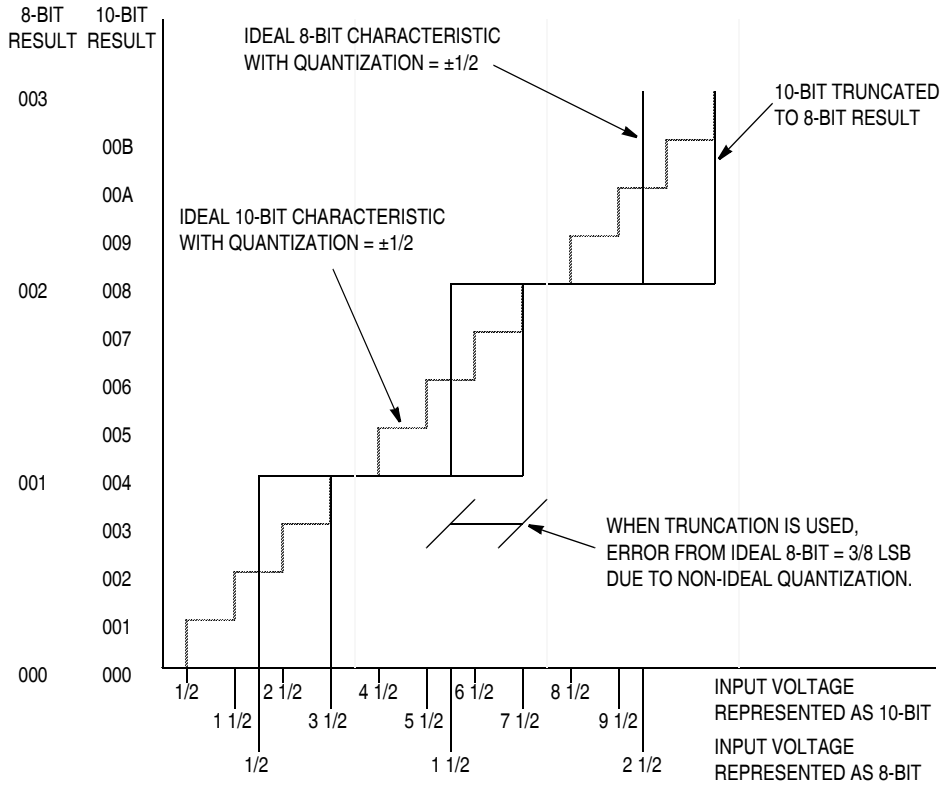



**Figure 3-3. Bit Truncation Mode Error**

## 3.4 Monotonicity

The conversion process is monotonic and has no missing codes.

## 3.5 Interrupts

When the AIEN bit is set, the ADC module is capable of generating CPU interrupts after each ADC conversion. A CPU interrupt is generated if the COCO bit is a 0. The COCO bit is not used as a conversion complete flag when interrupts are enabled.

## 3.6 Low-Power Modes

The WAIT and STOP instruction can put the MCU in low power- consumption standby modes.

### 3.6.1 Wait Mode

The ADC continues normal operation during wait mode. Any enabled CPU interrupt request from the ADC can bring the MCU out of wait mode. If the ADC is not required to bring the MCU out of wait mode, power down the ADC by setting ADCH4–ADCH0 bits in the ADC status and control register before executing the WAIT instruction.

### 3.6.2 Stop Mode

The ADC module is inactive after the execution of a STOP instruction. Any pending conversion is aborted. ADC conversions resume when the MCU exits stop mode after an external interrupt. Allow one conversion cycle to stabilize the analog circuitry.

## 3.7 I/O Signals

The ADC module has eight pins shared with port A and the KBI module:
PTA7/KBD7/AD15–PTA0/KBD0/AD8

The ADC module has eight pins shared with port B:
PTB7/AD7–PTB0/AD0

The ADC module has eight pins shared with port G:
PTG7/AD23–PTG0/AD16

### 3.7.1 ADC Analog Power Pin ($V_{DDAD}$)

The ADC analog portion uses $V_{DDAD}$ as its power pin. 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.

***NOTE***

*For maximum noise immunity, route $V_{DDAD}$ carefully and place bypass capacitors as close as possible to the package.*

$V_{DDAD}$ and $V_{REFH}$ are bonded internally.

### 3.7.2 ADC Analog Ground Pin ($V_{SSAD}$)

The ADC analog portion uses $V_{SSAD}$ as its ground pin. Connect the $V_{SSAD}$ pin to the same voltage potential as $V_{SS}$.

***NOTE***

*Route $V_{SSAD}$ cleanly to avoid any offset errors.*

$V_{SSAD}$ and $V_{REFL}$ are bonded internally.

### 3.7.3 ADC Voltage Reference High Pin ($V_{REFH}$)

The ADC analog portion uses $V_{REFH}$ as its upper voltage reference pin. By default, connect the $V_{REFH}$ pin to the same voltage potential as $V_{DD}$. External filtering is often necessary to ensure a clean $V_{REFH}$ for good results. Any noise present on this pin will be reflected and possibly magnified in A/D conversion values.

**NOTE**

*For maximum noise immunity, route $V_{REFH}$ carefully and place bypass capacitors as close as possible to the package. Routing $V_{REFH}$ close and parallel to $V_{REFL}$ may improve common mode noise rejection.*

$V_{DDAD}$ and $V_{REFH}$ are bonded internally.

### 3.7.4 ADC Voltage Reference Low Pin ($V_{REFL}$)

The ADC analog portion uses $V_{REFL}$ as its lower voltage reference pin. By default, connect the $V_{REFL}$ pin to the same voltage potential as $V_{SS}$. External filtering is often necessary to ensure a clean $V_{REFL}$ for good results. Any noise present on this pin will be reflected and possibly magnified in A/D conversion values.

**NOTE**

*For maximum noise immunity, route $V_{REFL}$ carefully and, if not connected to $V_{SS}$, place bypass capacitors as close as possible to the package. Routing $V_{REFH}$ close and parallel to $V_{REFL}$ may improve common mode noise rejection.*

$V_{SSAD}$ and $V_{REFL}$ are bonded internally.

### 3.7.5 ADC Voltage In ($V_{ADIN}$)

$V_{ADIN}$ is the input voltage signal from one of the 24 ADC channels to the ADC module.

## 3.8 I/O Registers

These I/O registers control and monitor ADC operation:

- ADC status and control register (ADSCR)
- ADC data register (ADRH and ADRL)
- ADC clock register (ADCLK)

### 3.8.1 ADC Status and Control Register

Function of the ADC status and control register (ADSCR) is described here.

Address: $003C

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | COCO | AIEN | ADCO | ADCH4 | ADCH3 | ADCH2 | ADCH1 | ADCH0 |
| Write: | R | | | | | | | |
| Reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

R = Reserved

**Figure 3-4. ADC Status and Control Register (ADSCR)**

**COCO — Conversions Complete Bit**

In non-interrupt mode (AIEN = 0), COCO is a read-only bit that is set at the end of each conversion. COCO will stay set until cleared by a read of the ADC data register. Reset clears this bit.

In interrupt mode (AIEN = 1), COCO is a read-only bit that is not set at the end of a conversion. It always reads as a 0.

1 = Conversion completed (AIEN = 0)
0 = Conversion not completed (AIEN = 0) or CPU interrupt enabled (AIEN = 1)

***NOTE***

*The write function of the COCO bit is reserved. When writing to the ADSCR register, always have a 0 in the COCO bit position.*

**AIEN — ADC Interrupt Enable Bit**

When this bit is set, an interrupt is generated at the end of an ADC conversion. The interrupt signal is cleared when the data register is read or the status/control register is written. Reset clears the AIEN bit.

1 = ADC interrupt enabled
0 = ADC interrupt disabled

**ADCO — ADC Continuous Conversion Bit**

When set, the ADC will convert samples continuously and update the ADR register at the end of each conversion. Only one conversion is completed between writes to the ADSCR when this bit is cleared. Reset clears the ADCO bit.

1 = Continuous ADC conversion
0 = One ADC conversion

**ADCH4–ADCH0 — ADC Channel Select Bits**

ADCH4–ADCH0 form a 5-bit field which is used to select one of 32 ADC channels. Only 24 channels, AD23–AD0, are available on this MCU. The channels are detailed in Table 3-1. Care should be taken when using a port pin as both an analog and digital input simultaneously to prevent switching noise from corrupting the analog signal. See Table 3-1.

The ADC subsystem is turned off when the channel select bits are all set to 1. This feature allows for reduced power consumption for the MCU when the ADC is not being used.

***NOTE***

*Recovery from the disabled state requires one conversion cycle to stabilize.*

The voltage levels supplied from internal reference nodes, as specified in Table 3-1, are used to verify the operation of the ADC converter both in production test and for user applications.

**Table 3-1. Mux Channel Select(1)**

| ADCH4 | ADCH3 | ADCH2 | ADCH1 | ADCH0 | Input Select |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | PTB0/AD0 |
| 0 | 0 | 0 | 0 | 1 | PTB1/AD1 |
| 0 | 0 | 0 | 1 | 0 | PTB2/AD2 |
| 0 | 0 | 0 | 1 | 1 | PTB3/AD3 |
| 0 | 0 | 1 | 0 | 0 | PTB4/AD4 |
| 0 | 0 | 1 | 0 | 1 | PTB5/AD5 |
| 0 | 0 | 1 | 1 | 0 | PTB6/AD6 |
| 0 | 0 | 1 | 1 | 1 | PTB7/AD7 |
| 0 | 1 | 0 | 0 | 0 | PTA0/KBD0/AD8 |
| 0 | 1 | 0 | 0 | 1 | PTA1/KBD1/AD9 |
| 0 | 1 | 0 | 1 | 0 | PTA2/KBD2/AD10 |
| 0 | 1 | 0 | 1 | 1 | PTA3/KBD3/AD11 |
| 0 | 1 | 1 | 0 | 0 | PTA4/KBD4/AD12 |
| 0 | 1 | 1 | 0 | 1 | PTA5/KBD5/AD13 |
| 0 | 1 | 1 | 1 | 0 | PTA6/KBD6/AD14 |
| 0 | 1 | 1 | 1 | 1 | PTA7/KBD7/AD15 |
| 1 | 0 | 0 | 0 | 0 | PTG0/AD16 |
| 1 | 0 | 0 | 0 | 1 | PTG1/AD17 |
| 1 | 0 | 0 | 1 | 0 | PTG2/AD18 |
| 1 | 0 | 0 | 1 | 1 | PTG3/AD19 |
| 1 | 0 | 1 | 0 | 0 | PTG4/AD20 |
| 1 | 0 | 1 | 0 | 1 | PTG5/AD21 |
| 1 | 0 | 1 | 1 | 0 | PTG6/AD22 |
| 1 | 0 | 1 | 1 | 1 | PTG7/AD23 |
| 1 ↓ 1 | 1 ↓ 1 | 0 ↓ 1 | 0 ↓ 0 | 0 ↓ 0 | Unused |
| 1 | 1 | 1 | 0 | 1 | $V_{REFH}$ |
| 1 | 1 | 1 | 1 | 0 | $V_{REFL}$ |
| 1 | 1 | 1 | 1 | 1 | ADC power off |

1. If any unused channels are selected, the resulting ADC conversion will be unknown or reserved.

## 3.8.2 ADC Data Register High and Data Register Low

### *3.8.2.1 Left Justified Mode*

In left justified mode, the ADRH register holds the eight MSBs of the 10-bit result. The ADRL register holds the two LSBs of the 10-bit result. All other bits read as 0. ADRH and ADRL are updated each time an ADC single channel conversion completes. Reading ADRH latches the contents of ADRL until ADRL is read. All subsequent results will be lost until the ADRH and ADRL reads are completed.

| Address: | $003D | | | | | | | ADRH |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |
| Address: | $003E | | | | | | | ADRL |
| Read: | AD1 | AD0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |

= Unimplemented

**Figure 3-5. ADC Data Register High (ADRH) and Low (ADRL)**

### *3.8.2.2 Right Justified Mode*

In right justified mode, the ADRH register holds the two MSBs of the 10-bit result. All other bits read as 0. The ADRL register holds the eight LSBs of the 10-bit result. ADRH and ADRL are updated each time an ADC single channel conversion completes. Reading ADRH latches the contents of ADRL until ADRL is read. All subsequent results will be lost until the ADRH and ADRL reads are completed.

| Address: | $003D | | | | | | | ADRH |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | 0 | 0 | 0 | 0 | 0 | 0 | AD9 | AD8 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |
| Address: | $003E | | | | | | | ADRL |
| Read: | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |

= Unimplemented

**Figure 3-6. ADC Data Register High (ADRH) and Low (ADRL)**

### 3.8.2.3 Left Justified Signed Data Mode

In left justified signed data mode, the ADRH register holds the eight MSBs of the 10-bit result. The only difference from left justified mode is that the AD9 is complemented. The ADRL register holds the two LSBs of the 10-bit result. All other bits read as 0. ADRH and ADRL are updated each time an ADC single channel conversion completes. Reading ADRH latches the contents of ADRL until ADRL is read. All subsequent results will be lost until the ADRH and ADRL reads are completed.

| Address: | $003D | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | $\overline{AD9}$ | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |
| Address: | $003E | | | | | | | |
| Read: | AD1 | AD0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |

= Unimplemented

**Figure 3-7. ADC Data Register High (ADRH) and Low (ADRL)**

### 3.8.2.4 Eight Bit Truncation Mode

In 8-bit truncation mode, the ADRL register holds the eight MSBs of the 10-bit result. The ADRH register is unused and reads as 0. The ADRL register is updated each time an ADC single channel conversion completes. In 8-bit mode, the ADRL register contains no interlocking with ADRH.

| Address: | $003D | | | | | | | ADRH |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |
| Address: | $003E | | | | | | | ADRL |
| Read: | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |

= Unimplemented

**Figure 3-8. ADC Data Register High (ADRH) and Low (ADRL)**

### 3.8.3 ADC Clock Register

The ADC clock register (ADCLK) selects the clock frequency for the ADC.

Address: $003F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | ADIV2 | ADIV1 | ADIV0 | ADICLK | MODE1 | MODE0 | R | 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

= Unimplemented  R = Reserved

**Figure 3-9. ADC Clock Register (ADCLK)**

**ADIV2–ADIV0 — ADC Clock Prescaler Bits**

ADIV2–ADIV0 form a 3-bit field which selects the divide ratio used by the ADC to generate the internal ADC clock. Table 3-2 shows the available clock configurations. The ADC clock should be set to approximately 1 MHz.

**Table 3-2. ADC Clock Divide Ratio**

| ADIV2 | ADIV1 | ADIV0 | ADC Clock Rate |
|---|---|---|---|
| 0 | 0 | 0 | ADC input clock ÷ 1 |
| 0 | 0 | 1 | ADC input clock ÷ 2 |
| 0 | 1 | 0 | ADC input clock ÷ 4 |
| 0 | 1 | 1 | ADC input clock ÷ 8 |
| 1 | X(1) | X(1) | ADC input clock ÷ 16 |

1. X = Don't care

**ADICLK — ADC Input Clock Select Bit**

ADICLK selects either the bus clock or the oscillator output clock (CGMXCLK) as the input clock source to generate the internal ADC clock. Reset selects CGMXCLK as the ADC clock source.

1 = Internal bus clock
0 = Oscillator output clock (CGMXCLK)

The ADC requires a clock rate of approximately 1 MHz for correct operation. If the selected clock source is not fast enough, the ADC will generate incorrect conversions. See 21.10 5.0-Volt ADC Characteristics.

$$f_{ADIC} = \frac{f_{CGMXCLK} \text{ or bus frequency}}{ADIV[2:0]} \cong 1 \text{ MHz}$$

**MODE1 and MODE0 — Modes** of Result Justification Bits

MODE1 and MODE0 select among four modes of operation. The manner in which the ADC conversion results will be placed in the ADC data registers is controlled by these modes of operation. Reset returns right-justified mode.

00 = 8-bit truncation mode
01 = Right justified mode
10 = Left justified mode
11 = Left justified signed data mode

# Chapter 4
# Clock Generator Module (CGM)

## 4.1 Introduction

This section describes the clock generator module. The CGM generates the crystal clock signal, CGMXCLK, which operates at the frequency of the crystal. The CGM also generates the base clock signal, CGMOUT, which is based on either the crystal clock divided by two or the phase-locked loop (PLL) clock, CGMVCLK, divided by two. In user mode, CGMOUT is the clock from which the SIM derives the system clocks, including the bus clock, which is at a frequency of CGMOUT/2. The PLL is a fully functional frequency generator designed for use with crystals or ceramic resonators. The PLL can generate a maximum bus frequency of 8 MHz using a 1-8MHz crystal or external clock source.

## 4.2 Features

Features of the CGM include:

- Phase-locked loop with output frequency in integer multiples of an integer dividend of the crystal reference
- High-frequency crystal operation with low-power operation and high-output frequency resolution
- Programmable hardware voltage-controlled oscillator (VCO) for low-jitter operation
- Automatic bandwidth control mode for low-jitter operation
- Automatic frequency lock detector
- CPU interrupt on entry or exit from locked condition
- Configuration register bit to allow oscillator operation during stop mode

## 4.3 Functional Description

The CGM consists of three major submodules:

- Crystal oscillator circuit — The crystal oscillator circuit generates the constant crystal frequency clock, CGMXCLK.
- Phase-locked loop (PLL) — The PLL generates the programmable VCO frequency clock, CGMVCLK.
- Base clock selector circuit — This software-controlled circuit selects either CGMXCLK divided by two or the VCO clock, CGMVCLK, divided by two as the base clock, CGMOUT. The SIM derives the system clocks from either CGMOUT or CGMXCLK.

Figure 4-1 shows the structure of the CGM.

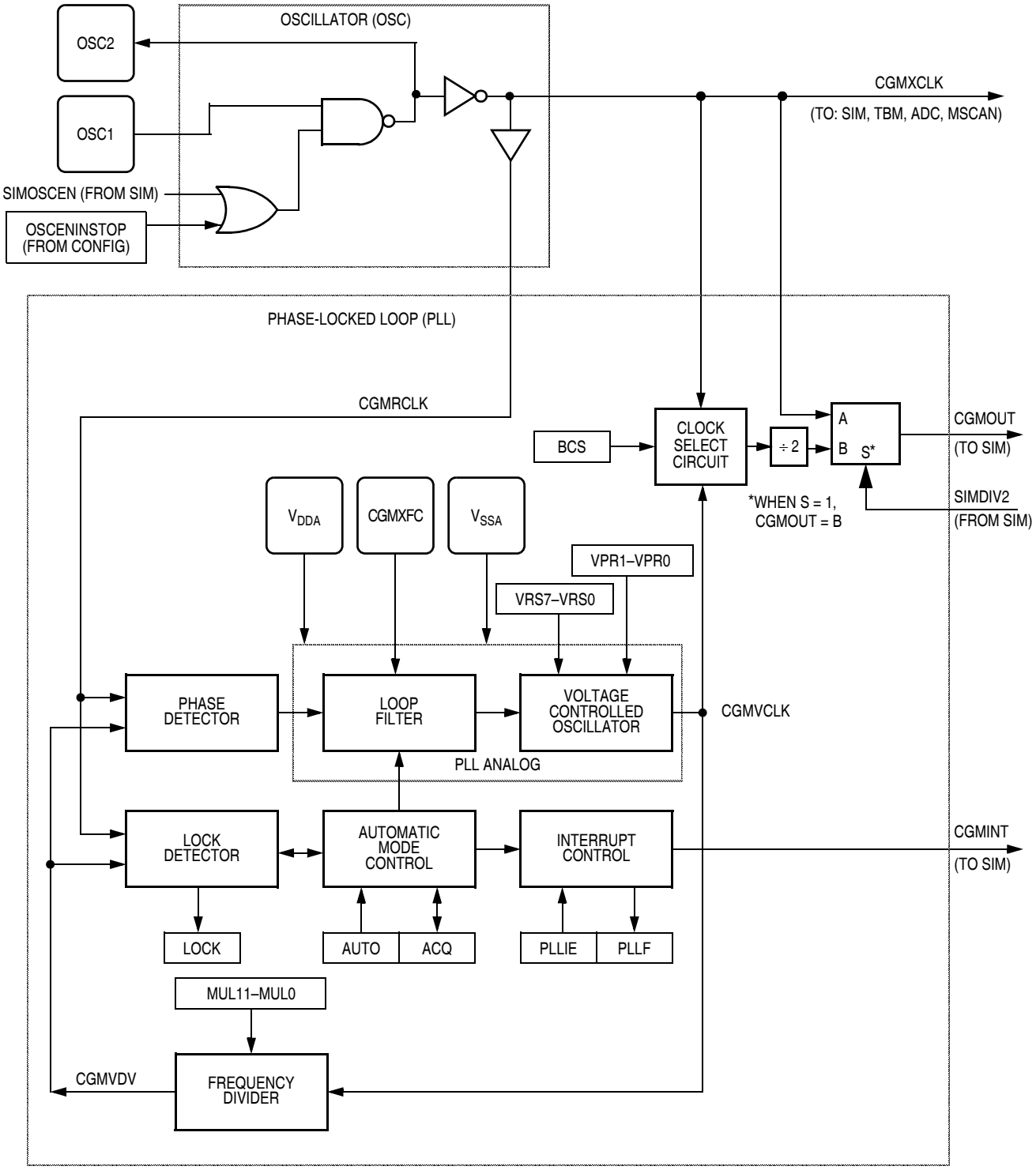



**Figure 4-1. CGM Block Diagram**

### 4.3.1 Crystal Oscillator Circuit

The crystal oscillator circuit consists of an inverting amplifier and an external crystal. The OSC1 pin is the input to the amplifier and the OSC2 pin is the output. The SIMOSCEN signal from the system integration module (SIM) or the OSCENINSTOP bit in the CONFIG register enable the crystal oscillator circuit.

The CGMXCLK signal is the output of the crystal oscillator circuit and runs at a rate equal to the crystal frequency. CGMXCLK is then buffered to produce CGMRCLK, the PLL reference clock.

CGMXCLK can be used by other modules which require precise timing for operation. The duty cycle of CGMXCLK is not guaranteed to be 50% and depends on external factors, including the crystal and related external components. An externally generated clock also can feed the OSC1 pin of the crystal oscillator circuit. Connect the external clock to the OSC1 pin and let the OSC2 pin float.

### 4.3.2 Phase-Locked Loop Circuit (PLL)

The PLL is a frequency generator that can operate in either acquisition mode or tracking mode, depending on the accuracy of the output frequency. The PLL can change between acquisition and tracking modes either automatically or manually.

### 4.3.3 PLL Circuits

The PLL consists of these circuits:

- Voltage-controlled oscillator (VCO)
- Modulo VCO frequency divider
- Phase detector
- Loop filter
- Lock detector

The operating range of the VCO is programmable for a wide range of frequencies and for maximum immunity to external noise, including supply and CGMXFC noise. The VCO frequency is bound to a range from roughly one-half to twice the center-of-range frequency, $f_{VRS}$. Modulating the voltage on the CGMXFC pin changes the frequency within this range. By design, $f_{VRS}$ is equal to the nominal center-of-range frequency, $f_{NOM}$, (71.4 kHz) times a linear factor, L, and a power-of-two factor, E, or $(L \times 2^E)f_{NOM}$.

CGMRCLK is the PLL reference clock, a buffered version of CGMXCLK. CGMRCLK runs at a frequency, $f_{RCLK}$. The VCO's output clock, CGMVCLK, running at a frequency, $f_{VCLK}$, is fed back through a programmable modulo divider. The modulo divider reduces the VCO clock by a factor, N. The dividers output is the VCO feedback clock, CGMVDV, running at a frequency, $f_{VDV} = f_{VCLK}/(N)$. (For more information, see 4.3.6 Programming the PLL.)

The phase detector then compares the VCO feedback clock, CGMVDV, with the final reference clock, CGMRDV. A correction pulse is generated based on the phase difference between the two signals. The loop filter then slightly alters the DC voltage on the external capacitor connected to CGMXFC based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, described in 4.3.4 Acquisition and Tracking Modes. The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

The lock detector compares the frequencies of the VCO feedback clock, CGMVDV, and the reference clock, CGMRCLK. Therefore, the speed of the lock detector is directly proportional to the reference

frequency, $f_{RCLK}$. The circuit determines the mode of the PLL and the lock condition based on this comparison.

### 4.3.4 Acquisition and Tracking Modes

The PLL filter is manually or automatically configurable into one of two operating modes:

- Acquisition mode — In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL start up or when the PLL has suffered a severe noise hit and the VCO frequency is far off the desired frequency. When in acquisition mode, the $\overline{ACQ}$ bit is clear in the PLL bandwidth control register. (See 4.5.2 PLL Bandwidth Control Register.)
- Tracking mode — In tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source. (See 4.3.8 Base Clock Selector Circuit.) The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{ACQ}$ bit is set.

### 4.3.5 Manual and Automatic PLL Bandwidth Modes

The PLL can change the bandwidth or operational mode of the loop filter manually or automatically. Automatic mode is recommended for most users.

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT. (See 4.5.2 PLL Bandwidth Control Register.) If PLL interrupts are enabled, the software can wait for a PLL interrupt request and then check the LOCK bit. If interrupts are disabled, software can poll the LOCK bit continuously (for example, during PLL start up) or at periodic intervals. In either case, when the LOCK bit is set, the VCO clock is safe to use as the source for the base clock. (See 4.3.8 Base Clock Selector Circuit.) If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application. (See 4.6 Interrupts for information and precautions on using interrupts.)

The following conditions apply when the PLL is in automatic bandwidth control mode:

- The $\overline{ACQ}$ bit (See 4.5.2 PLL Bandwidth Control Register.) is a read-only indicator of the mode of the filter. (See 4.3.4 Acquisition and Tracking Modes.)
- The $\overline{ACQ}$ bit is set when the VCO frequency is within a certain tolerance and is cleared when the VCO frequency is out of a certain tolerance. (See 4.8 Acquisition/Lock Time Specifications for more information.)
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance and is cleared when the VCO frequency is out of a certain tolerance. (See 4.8 Acquisition/Lock Time Specifications for more information.)
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. (See 4.5.1 PLL Control Register.)

The PLL also may operate in manual mode (AUTO = 0). Manual mode is used by systems that do not require an indicator of the lock condition for proper operation. Such systems typically operate well below $f_{BUSMAX}$.

The following conditions apply when in manual mode:

- $\overline{ACQ}$ is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the $\overline{ACQ}$ bit must be clear.
- Before entering tracking mode ($\overline{ACQ}$ = 1), software must wait a given time, $t_{ACQ}$ (See 4.8 Acquisition/Lock Time Specifications.), after turning on the PLL by setting PLLON in the PLL control register (PCTL).
- Software must wait a given time, $t_{AL}$, after entering tracking mode before selecting the PLL as the clock source to CGMOUT (BCS = 1).
- The LOCK bit is disabled.
- CPU interrupts from the CGM are disabled.

### 4.3.6 Programming the PLL

Use the following procedure to program the PLL. For reference, the variables used and their meaning are shown in Table 4-1.

**Table 4-1. Variable Definitions**

| Variable | Definition |
|---|---|
| $f_{BUSDES}$ | Desired bus clock frequency |
| $f_{VCLKDES}$ | Desired VCO clock frequency |
| $f_{RCLK}$ | Chosen reference crystal frequency |
| $f_{VCLK}$ | Calculated VCO clock frequency |
| $f_{BUS}$ | Calculated bus clock frequency |
| $f_{NOM}$ | Nominal VCO center frequency |
| $f_{VRS}$ | Programmed VCO center frequency |

***NOTE***

*The round function in the following equations means that the real number should be rounded to the nearest integer number.*

1. Choose the desired bus frequency, $f_{BUSDES}$.
2. Calculate the desired VCO frequency (four times the desired bus frequency).

$$f_{VCLKDES} = 4 \times f_{BUSDES}$$

3. Choose a practical PLL (crystal) reference frequency, $f_{RCLK}$. Typically, the reference crystal is 1–8 MHz.

   Frequency errors to the PLL are corrected at a rate of $f_{RCLK}$.
   For stability and lock time reduction, this rate must be as fast as possible. The VCO frequency must be an integer multiple of this rate. The relationship between the VCO frequency, $f_{VCLK}$, and the reference frequency, $f_{RCLK}$, is:

$$f_{VCLK} = (N)\ (f_{RCLK})$$

   N, the range multiplier, must be an integer.

In cases where desired bus frequency has some tolerance, choose $f_{RCLK}$ to a value determined either by other module requirements (such as modules which are clocked by CGMXCLK), cost requirements, or ideally, as high as the specified range allows. See Chapter 21 Electrical Specifications. After choosing N, the actual bus frequency can be determined using equation in 2 above.

4. Select a VCO frequency multiplier, N.

$$N = \text{round}\left(\frac{f_{VCLKDES}}{f_{RCLK}}\right)$$

5. Calculate and verify the adequacy of the VCO and bus frequencies $f_{VCLK}$ and $f_{BUS}$.

$$f_{VCLK} = (N) \times f_{RCLK}$$
$$f_{BUS} = (f_{VCLK})/4$$

6. Select the VCO's power-of-two range multiplier E, according to Table 4-2.

**Table 4-2. Power-of-Two Range Selectors**

| Frequency Range | E |
|---|---|
| $0 < f_{VCLK} \leq 8$ MHz | 0 |
| 8 MHz$< f_{VCLK} \leq 16$ MHz | 1 |
| 16 MHz$< f_{VCLK} \leq 32$ MHz | 2(1) |

1. Do not program E to a value of 3.

7. Select a VCO linear range multiplier, L, where $f_{NOM}$ = 71.4 kHz

$$L = \text{Round}\left(\frac{f_{VCLK}}{2^E \times f_{NOM}}\right)$$

8. Calculate and verify the adequacy of the VCO programmed center-of-range frequency, $f_{VRS}$. The center-of-range frequency is the midpoint between the minimum and maximum frequencies attainable by the PLL.

$$f_{VRS} = (L \times 2^E)\, f_{NOM}$$

9. For proper operation,

$$|f_{VRS} - f_{VCLK}| \leq \frac{f_{NOM} \times 2^E}{2}$$

10. Verify the choice of N, E, and L by comparing $f_{VCLK}$ to $f_{VRS}$ and $f_{VCLKDES}$. For proper operation, $f_{VCLK}$ must be within the application's tolerance of $f_{VCLKDES}$, and $f_{VRS}$ must be as close as possible to $f_{VCLK}$.

***NOTE***

*Exceeding the recommended maximum bus frequency or VCO frequency can crash the MCU.*

11. Program the PLL registers accordingly:
    a. In the VPR bits of the PLL control register (PCTL), program the binary equivalent of E.
    b. In the PLL multiplier select register low (PMSL) and the PLL multiplier select register high (PMSH), program the binary equivalent of N. If using a 1–8 MHz reference, the PMSL register must be reprogrammed from the reset value before enabling the PLL.
    c. In the PLL VCO range select register (PMRS), program the binary coded equivalent of L.

Table 4-3 provides numeric examples (register values are in hexadecimal notation):

**Table 4-3. Numeric Example**

| $f_{BUS}$ | $f_{RCLK}$ | N | E | L |
|---|---|---|---|---|
| 500 kHz | 1 MHz | 002 | 0 | 1B |
| 1.25 MHz | 1 MHz | 005 | 0 | 45 |
| 2.0 MHz | 1 MHz | 008 | 0 | 70 |
| 2.5 MHz | 1 MHz | 00A | 1 | 45 |
| 3.0 MHz | 1 MHz | 00C | 1 | 53 |
| 4.0 MHz | 1 MHz | 010 | 1 | 70 |
| 5.0 MHz | 1 MHz | 014 | 2 | 46 |
| 7.0 MHz | 1 MHz | 01C | 2 | 62 |
| 8.0 MHz | 1 MHz | 020 | 2 | 70 |

### 4.3.7 Special Programming Exceptions

The programming method described in 4.3.6 Programming the PLL does not account for two possible exceptions. A value of 0 for N or L is meaningless when used in the equations given. To account for these exceptions:

- A 0 value for N is interpreted exactly the same as a value of 1.
- A 0 value for L disables the PLL and prevents its selection as the source for the base clock.

See 4.3.8 Base Clock Selector Circuit.

### 4.3.8 Base Clock Selector Circuit

This circuit is used to select either the crystal clock, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the base clock, CGMOUT. The two input clocks go through a transition control circuit that waits up to three CGMXCLK cycles and three CGMVCLK cycles to change from one clock source to the other. During this time, CGMOUT is held in stasis. The output of the transition control circuit is then divided by two to correct the duty cycle. Therefore, the bus clock frequency, which is one-half of the base clock frequency, is one-fourth the frequency of the selected clock (CGMXCLK or CGMVCLK).

The BCS bit in the PLL control register (PCTL) selects which clock drives CGMOUT. The VCO clock cannot be selected as the base clock source if the PLL is not turned on. The PLL cannot be turned off if the VCO clock is selected. The PLL cannot be turned on or off simultaneously with the selection or deselection of the VCO clock. The VCO clock also cannot be selected as the base clock source if the factor L is programmed to a 0. This value would set up a condition inconsistent with the operation of the PLL, so that the PLL would be disabled and the crystal clock would be forced as the source of the base clock.

### 4.3.9 CGM External Connections

In its typical configuration, the CGM requires external components. Five of these are for the crystal oscillator and two or four are for the PLL.

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in Figure 4-2. Figure 4-2 shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

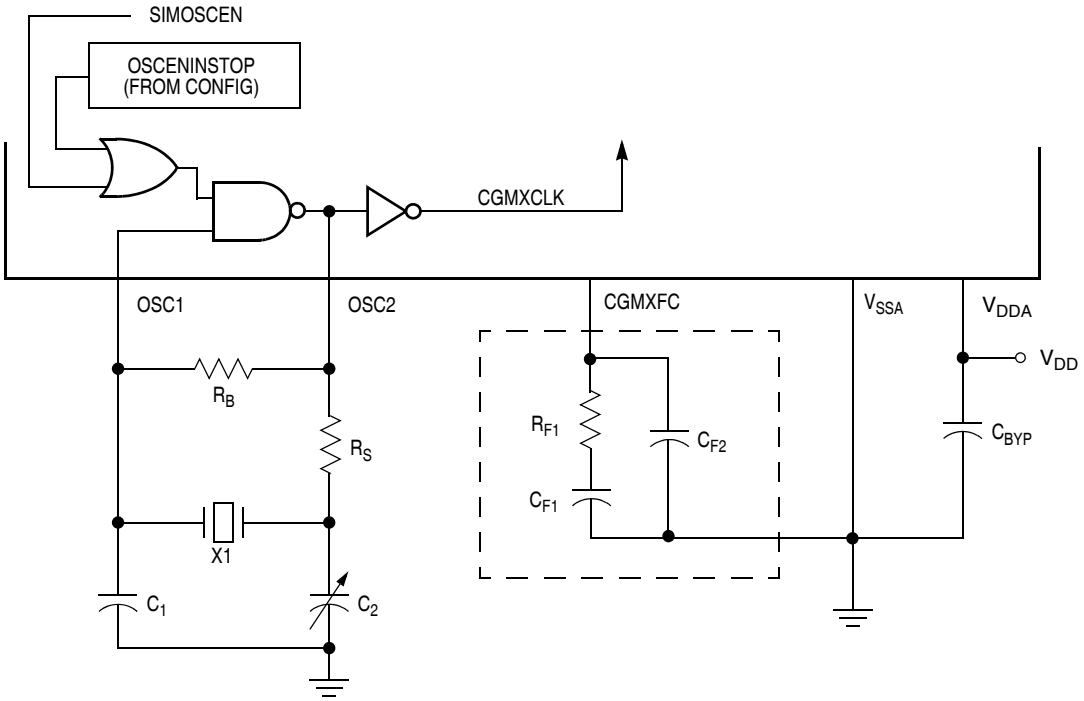
- Crystal, $X_1$
- Fixed capacitor, $C_1$
- Tuning capacitor, $C_2$ (can also be a fixed capacitor)
- Feedback resistor, $R_B$
- Series resistor, $R_S$

The series resistor ($R_S$) is included in the diagram to follow strict Pierce oscillator guidelines. Refer to the crystal manufacturer's data for more information regarding values for C1 and C2.

Figure 4-2 also shows the external components for the PLL:

- Bypass capacitor, $C_{BYP}$
- Filter network

Routing should be done with great care to minimize signal cross talk and noise.




Note: Filter network in box can be replaced with a single capacitor, but will degrade stability.

**Figure 4-2. CGM External Connections**

## 4.4 I/O Signals

The following paragraphs describe the CGM I/O signals.

### 4.4.1 Crystal Amplifier Input Pin (OSC1)

The OSC1 pin is an input to the crystal oscillator amplifier.

### 4.4.2 Crystal Amplifier Output Pin (OSC2)

The OSC2 pin is the output of the crystal oscillator inverting amplifier.

### 4.4.3 External Filter Capacitor Pin (CGMXFC)

The CGMXFC pin is required by the loop filter to filter out phase corrections. An external filter network is connected to this pin. (See Figure 4-2.)

***NOTE***

*To prevent noise problems, the filter network should be placed as close to the CGMXFC pin as possible, with minimum routing distances and no routing of other signals across the network.*

### 4.4.4 PLL Analog Power Pin ($V_{DDA}$)

$V_{DDA}$ is a power pin used by the analog portions of the PLL. Connect the $V_{DDA}$ pin to the same voltage potential as the $V_{DD}$ pin.

***NOTE***

*Route $V_{DDA}$ carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.*

### 4.4.5 PLL Analog Ground Pin ($V_{SSA}$)

$V_{SSA}$ is a ground pin used by the analog portions of the PLL. Connect the $V_{SSA}$ pin to the same voltage potential as the $V_{SS}$ pin.

***NOTE***

*Route $V_{SSA}$ carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.*

### 4.4.6 Oscillator Enable Signal (SIMOSCEN)

The SIMOSCEN signal comes from the system integration module (SIM) and enables the oscillator and PLL.

### 4.4.7 Oscillator Enable in Stop Mode Bit (OSCENINSTOP)

OSCENINSTOP is a bit in the CONFIG2 register that enables the oscillator to continue operating during stop mode. If this bit is set, the oscillator continues running during stop mode. If this bit is not set (default), the oscillator is controlled by the SIMOSCEN signal which will disable the oscillator during stop mode.

### 4.4.8 Crystal Output Frequency Signal (CGMXCLK)

CGMXCLK is the crystal oscillator output signal. It runs at the full speed of the crystal ($f_{XCLK}$) and comes directly from the crystal oscillator circuit. Figure 4-2 shows only the logical relation of CGMXCLK to OSC1 and OSC2 and may not represent the actual circuitry. The duty cycle of CGMXCLK is unknown and may depend on the crystal and other external factors. Also, the frequency and amplitude of CGMXCLK can be unstable at start up.

### 4.4.9 CGM Base Clock Output (CGMOUT)

CGMOUT is the clock output of the CGM. This signal goes to the SIM, which generates the MCU clocks. CGMOUT is a 50 percent duty cycle clock running at twice the bus frequency. CGMOUT is software programmable to be either the oscillator output, CGMXCLK, divided by two or the VCO clock, CGMVCLK, divided by two.

### 4.4.10 CGM CPU Interrupt (CGMINT)

CGMINT is the interrupt signal generated by the PLL lock detector.

## 4.5 CGM Registers

These registers control and monitor operation of the CGM:

- PLL control register (PCTL)
  (See 4.5.1 PLL Control Register.)
- PLL bandwidth control register (PBWC)
  (See 4.5.2 PLL Bandwidth Control Register.)
- PLL multiplier select register high (PMSH)
  (See 4.5.3 PLL Multiplier Select Register High.)
- PLL multiplier select register low (PMSL)
  (See 4.5.4 PLL Multiplier Select Register Low.)
- PLL VCO range select register (PMRS)
  (See 4.5.5 PLL VCO Range Select Register.)

Figure 4-3 is a summary of the CGM registers.

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0036 | PLL Control Register (PCTL) See page 83. | Read: | PLLIE | PLLF | PLLON | BCS | R | R | VPR1 | VPR0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $0037 | PLL Bandwidth Control Register (PBWC) See page 85. | Read: | AUTO | LOCK | $\overline{\text{ACQ}}$ | 0 | 0 | 0 | 0 | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0038 | PLL Multiplier Select High Register (PMSH) See page 86. | Read: | 0 | 0 | 0 | 0 | MUL11 | MUL10 | MUL9 | MUL8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | = Unimplemented | | | R | = Reserved | | | |

**Figure 4-3. CGM I/O Register Summary**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0039 | PLL Multiplier Select Low Register (PMSL) See page 86. | Read: Write: | MUL7 | MUL6 | MUL5 | MUL4 | MUL3 | MUL2 | MUL1 | MUL0 |
| | | Reset: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $003A | PLL VCO Select Range Register (PMRS) See page 87. | Read: Write: | VRS7 | VRS6 | VRS5 | VRS4 | VRS3 | VRS2 | VRS1 | VRS0 |
| | | Reset: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $003B | Reserved Register | Read: | 0 | 0 | 0 | 0 | R | R | R | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

= Unimplemented  R = Reserved

NOTES:
1. When AUTO = 0, PLLIE is forced clear and is read-only.
2. When AUTO = 0, PLLF and LOCK read as clear.
3. When AUTO = 1, $\overline{ACQ}$ is read-only.
4. When PLLON = 0 or VRS7:VRS0 = $0, BCS is forced clear and is read-only.
5. When PLLON = 1, the PLL programming register is read-only.
6. When BCS = 1, PLLON is forced set and is read-only.

**Figure 4-3. CGM I/O Register Summary (Continued)**

## 4.5.1 PLL Control Register

The PLL control register (PCTL) contains the interrupt enable and flag bits, the on/off switch, the base clock selector bit, and the VCO power-of-two range selector bits.

Address: $0036

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | PLLIE | PLLF | PLLON | BCS | R | R | VPR1 | VPR0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented  R = Reserved

**Figure 4-4. PLL Control Register (PCTL)**

**PLLIE — PLL Interrupt Enable Bit**

This read/write bit enables the PLL to generate an interrupt request when the LOCK bit toggles, setting the PLL flag, PLLF. When the AUTO bit in the PLL bandwidth control register (PBWC) is clear, PLLIE cannot be written and reads as 0. Reset clears the PLLIE bit.

1 = PLL interrupts enabled
0 = PLL interrupts disabled

**PLLF — PLL Interrupt Flag Bit**

This read-only bit is set whenever the LOCK bit toggles. PLLF generates an interrupt request if the PLLIE bit also is set. PLLF always reads as 0 when the AUTO bit in the PLL bandwidth control register (PBWC) is clear. Clear the PLLF bit by reading the PLL control register. Reset clears the PLLF bit.

1 = Change in lock condition
0 = No change in lock condition

***NOTE***

*Do not inadvertently clear the PLLF bit. Any read or read-modify-write operation on the PLL control register clears the PLLF bit.*

**PLLON — PLL On Bit**

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). (See 4.3.8 Base Clock Selector Circuit.) Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

**BCS — Base Clock Select Bit**

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. (See 4.3.8 Base Clock Selector Circuit.) Reset clears the BCS bit.

1 = CGMVCLK divided by two drives CGMOUT
0 = CGMXCLK divided by two drives CGMOUT

***NOTE***

*PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMVCLK requires two writes to the PLL control register. (See 4.3.8 Base Clock Selector Circuit.).*

**VPR1 and VPR0 — VCO Power-of-Two Range Select Bits**

These read/write bits control the VCO's hardware power-of-two range multiplier E that, in conjunction with L controls the hardware center-of-range frequency, $f_{VRS}$. VPR1:VPR0 cannot be written when the PLLON bit is set. Reset clears these bits. (See 4.3.3 PLL Circuits, 4.3.6 Programming the PLL, and 4.5.5 PLL VCO Range Select Register.)

**Table 4-4. VPR1 and VPR0 Programming**

| VPR1 and VPR0 | E | VCO Power-of-Two Range Multiplier |
|---|---|---|
| 00 | 0 | 1 |
| 01 | 1 | 2 |
| 10 | 2(1) | 4 |

1. Do not program E to a value of 3.

***NOTE***

*Verify that the value of the VPR1 and VPR0 bits in the PCTL register are appropriate for the given reference and VCO clock frequencies before enabling the PLL. See 4.3.6 Programming the PLL for detailed instructions on selecting the proper value for these control bits.*

### 4.5.2 PLL Bandwidth Control Register

The PLL bandwidth control register (PBWC):

- Selects automatic or manual (software-controlled) bandwidth control mode
- Indicates when the PLL is locked
- In automatic bandwidth control mode, indicates when the PLL is in acquisition or tracking mode
- In manual operation, forces the PLL into acquisition or tracking mode

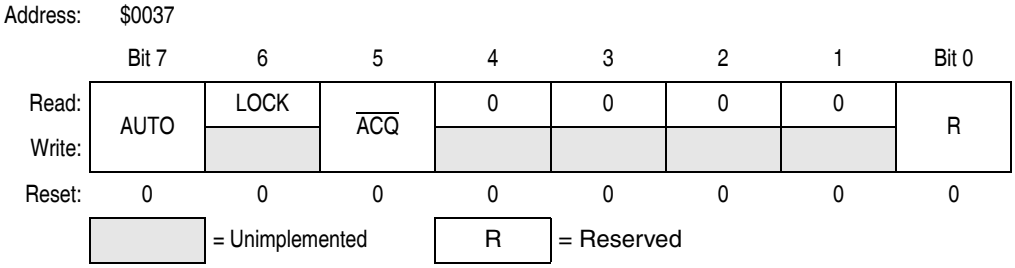



**Figure 4-5. PLL Bandwidth Control Register (PBWC)**

**AUTO — Automatic Bandwidth Control Bit**

This read/write bit selects automatic or manual bandwidth control. When initializing the PLL for manual operation (AUTO = 0), clear the $\overline{ACQ}$ bit before turning on the PLL. Reset clears the AUTO bit.

1 = Automatic bandwidth control
0 = Manual bandwidth control

**LOCK — Lock Indicator Bit**

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock, CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as 0 and has no meaning. The write one function of this bit is reserved for test, so this bit must ***always*** be written a 0. Reset clears the LOCK bit.

1 = VCO frequency correct or locked
0 = VCO frequency incorrect or unlocked

**$\overline{ACQ}$ — Acquisition Mode Bit**

When the AUTO bit is set, $\overline{ACQ}$ is a read-only bit that indicates whether the PLL is in acquisition mode or tracking mode. When the AUTO bit is clear, $\overline{ACQ}$ is a read/write bit that controls whether the PLL is in acquisition or tracking mode.

In automatic bandwidth control mode (AUTO = 1), the last-written value from manual operation is stored in a temporary location and is recovered when manual operation resumes. Reset clears this bit, enabling acquisition mode.

1 = Tracking mode
0 = Acquisition mode

### 4.5.3 PLL Multiplier Select Register High

The PLL multiplier select register high (PMSH) contains the programming information for the high byte of the modulo feedback divider.

Address: $0038

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | MUL11 | MUL10 | MUL9 | MUL8 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 4-6. PLL Multiplier Select Register High (PMSH)**

**MUL11–MUL8 — Multiplier Select Bits**

These read/write bits control the high byte of the modulo feedback divider that selects the VCO frequency multiplier N. (See 4.3.3 PLL Circuits and 4.3.6 Programming the PLL.) A value of $0000 in the multiplier select registers configures the modulo feedback divider the same as a value of $0001. Reset initializes the registers to $0040 for a default multiply value of 64.

***NOTE***

*The multiplier select bits have built-in protection such that they cannot be written when the PLL is on (PLLON = 1).*

**PMSH[7:4] — Unimplemented Bits**

These bits have no function and always read as 0s.

### 4.5.4 PLL Multiplier Select Register Low

The PLL multiplier select register low (PMSL) contains the programming information for the low byte of the modulo feedback divider.

Address: $0038

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | MUL7 | MUL6 | MUL5 | MUL4 | MUL3 | MUL2 | MUL1 | MUL0 |
| Write: | | | | | | | | |
| Reset: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4-7. PLL Multiplier Select Register Low (PMSL)**

***NOTE***

*For applications using 1–8 MHz reference frequencies this register must be reprogrammed before enabling the PLL. The reset value of this register will cause applications using 1–8 MHz reference frequencies to become unstable if the PLL is enabled without programming an appropriate value. The programmed value must not allow the VCO clock to exceed 32 MHz. See 4.3.6 Programming the PLL for detailed instructions on choosing the proper value for PMSL.*

**MUL7–MUL0 — Multiplier Select Bits**

These read/write bits control the low byte of the modulo feedback divider that selects the VCO frequency multiplier, N. (See 4.3.3 PLL Circuits and 4.3.6 Programming the PLL.) MUL7–MUL0 cannot be written when the PLLON bit in the PCTL is set. A value of $0000 in the multiplier select registers configures the modulo feedback divider the same as a value of $0001. Reset initializes the register to $40 for a default multiply value of 64.

***NOTE***

*The multiplier select bits have built-in protection such that they cannot be written when the PLL is on (PLLON = 1).*

## 4.5.5 PLL VCO Range Select Register

The PLL VCO range select register (PMRS) contains the programming information required for the hardware configuration of the VCO.

Address: $003A

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | VRS7 | VRS6 | VRS5 | VRS4 | VRS3 | VRS2 | VRS1 | VRS0 |
| Reset: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4-8. PLL VCO Range Select Register (PMRS)**

***NOTE***

*Verify that the value of the PMRS register is appropriate for the given reference and VCO clock frequencies before enabling the PLL. See 4.3.6 Programming the PLL for detailed instructions on selecting the proper value for these control bits.*

**VRS7–VRS0 — VCO Range Select Bits**

These read/write bits control the hardware center-of-range linear multiplier L which, in conjunction with E (See 4.3.3 PLL Circuits, 4.3.6 Programming the PLL, and 4.5.1 PLL Control Register.), controls the hardware center-of-range frequency, $f_{VRS}$. VRS7–VRS0 cannot be written when the PLLON bit in the PCTL is set. (See 4.3.7 Special Programming Exceptions.) A value of $00 in the VCO range select register disables the PLL and clears the BCS bit in the PLL control register (PCTL). (See 4.3.8 Base Clock Selector Circuit and 4.3.7 Special Programming Exceptions.). Reset initializes the register to $40 for a default range multiply value of 64.

***NOTE***

*The VCO range select bits have built-in protection such that they cannot be written when the PLL is on (PLLON = 1) and such that the VCO clock cannot be selected as the source of the base clock (BCS = 1) if the VCO range select bits are all clear.*

*The PLL VCO range select register must be programmed correctly. Incorrect programming can result in failure of the PLL to achieve lock.*

## 4.6 Interrupts

When the AUTO bit is set in the PLL bandwidth control register (PBWC), the PLL can generate a CPU interrupt request every time the LOCK bit changes state. The PLLIE bit in the PLL control register (PCTL) enables CPU interrupts from the PLL. PLLF, the interrupt flag in the PCTL, becomes set whether interrupts are enabled or not. When the AUTO bit is clear, CPU interrupts from the PLL are disabled and PLLF reads as 0.

Software should read the LOCK bit after a PLL interrupt request to see if the request was due to an entry into lock or an exit from lock. When the PLL enters lock, the VCO clock, CGMVCLK, divided by two can be selected as the CGMOUT source by setting BCS in the PCTL. When the PLL exits lock, the VCO clock frequency is corrupt, and appropriate precautions should be taken. If the application is not frequency sensitive, interrupts should be disabled to prevent PLL interrupt service routines from impeding software performance or from exceeding stack limitations.

***NOTE***

*Software can select the CGMVCLK divided by two as the CGMOUT source even if the PLL is not locked (LOCK = 0). Therefore, software should make sure the PLL is locked before setting the BCS bit.*

## 4.7 Special Modes

The WAIT instruction puts the MCU in low power-consumption standby modes.

### 4.7.1 Wait Mode

The WAIT instruction does not affect the CGM. Before entering wait mode, software can disengage and turn off the PLL by clearing the BCS and PLLON bits in the PLL control register (PCTL) to save power. Less power-sensitive applications can disengage the PLL without turning it off, so that the PLL clock is immediately available at WAIT exit. This would be the case also when the PLL is to wake the MCU from wait mode, such as when the PLL is first enabled and waiting for LOCK or LOCK is lost.

### 4.7.2 Stop Mode

If the OSCENINSTOP bit in the CONFIG2 register is cleared (default), then the STOP instruction disables the CGM (oscillator and phase locked loop) and holds low all CGM outputs (CGMXCLK, CGMOUT, and CGMINT).

If the OSCENINSTOP bit in the CONFIG2 register is set, then the phase locked loop is shut off but the oscillator will continue to operate in stop mode.

### 4.7.3 CGM During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See 15.7.3 Break Flag Control Register.)

To allow software to clear status bits during a break interrupt, write a 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect the PLLF bit during the break state, write a 0 to the BCFE bit. With BCFE at 0 (its default state), software can read and write the PLL control register during the break state without affecting the PLLF bit.

## 4.8 Acquisition/Lock Time Specifications

The acquisition and lock times of the PLL are, in many applications, the most critical PLL design parameters. Proper design and use of the PLL ensures the highest stability and lowest acquisition/lock times.

### 4.8.1 Acquisition/Lock Time Definitions

Typical control systems refer to the acquisition time or lock time as the reaction time, within specified tolerances, of the system to a step input. In a PLL, the step input occurs when the PLL is turned on or when it suffers a noise hit. The tolerance is usually specified as a percent of the step input or when the output settles to the desired value plus or minus a percent of the frequency change. Therefore, the reaction time is constant in this definition, regardless of the size of the step input. For example, consider a system with a 5 percent acquisition time tolerance. If a command instructs the system to change from 0 Hz to 1 MHz, the acquisition time is the time taken for the frequency to reach 1 MHz ±50 kHz. Fifty kHz = 5% of the 1-MHz step input. If the system is operating at 1 MHz and suffers a –100-kHz noise hit, the acquisition time is the time taken to return from 900 kHz to 1 MHz ±5 kHz. Five kHz = 5% of the 100-kHz step input.

Other systems refer to acquisition and lock times as the time the system takes to reduce the error between the actual output and the desired output to within specified tolerances. Therefore, the acquisition or lock time varies according to the original error in the output. Minor errors may not even be registered. Typical PLL applications prefer to use this definition because the system requires the output frequency to be within a certain tolerance of the desired frequency regardless of the size of the initial error.

### 4.8.2 Parametric Influences on Reaction Time

Acquisition and lock times are designed to be as short as possible while still providing the highest possible stability. These reaction times are not constant, however. Many factors directly and indirectly affect the acquisition time.

The most critical parameter which affects the reaction times of the PLL is the reference frequency, $f_{RCLK}$. This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, the corrections must be small compared to the desired frequency, so several corrections are required to reduce the frequency error. Therefore, the slower the reference the longer it takes to make these corrections. This parameter is under user control via the choice of crystal frequency $f_{XCLK}$. (See 4.3.3 PLL Circuits and 4.3.6 Programming the PLL.)

Another critical parameter is the external filter network. The PLL modifies the voltage on the VCO by adding or subtracting charge from capacitors in this network. Therefore, the rate at which the voltage changes for a given frequency error (thus change in charge) is proportional to the capacitance. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor is too large, the PLL may not be able to adjust the voltage in a reasonable time. (See 4.8.3 Choosing a Filter.)

Also important is the operating voltage potential applied to $V_{DDA}$. The power supply potential alters the characteristics of the PLL. A fixed value is best. Variable supplies, such as batteries, are acceptable if they vary within a known range at very slow speeds. Noise on the power supply is not acceptable, because it causes small frequency errors which continually change the acquisition time of the PLL.

Temperature and processing also can affect acquisition time because the electrical characteristics of the PLL change. The part operates as specified as long as these influences stay within the specified limits.

External factors, however, can cause drastic changes in the operation of the PLL. These factors include noise injected into the PLL through the filter capacitor, filter capacitor leakage, stray impedances on the circuit board, and even humidity or circuit board contamination.

### 4.8.3 Choosing a Filter

As described in 4.8.2 Parametric Influences on Reaction Time, the external filter network is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage.

Figure 4-9 shows two types of filter circuits. In low-cost applications, where stability and reaction time of the PLL are not critical, the three component filter network shown in Figure 4-9 (B) can be replaced by a single capacitor, $C_F$, as shown in shown in Figure 4-9 (A). Refer to Table 4-5 for recommended filter components at various reference frequencies. For reference frequencies between the values listed in the table, extrapolate to the nearest common capacitor value. In general, a slightly larger capacitor provides more stability at the expense of increased lock time.

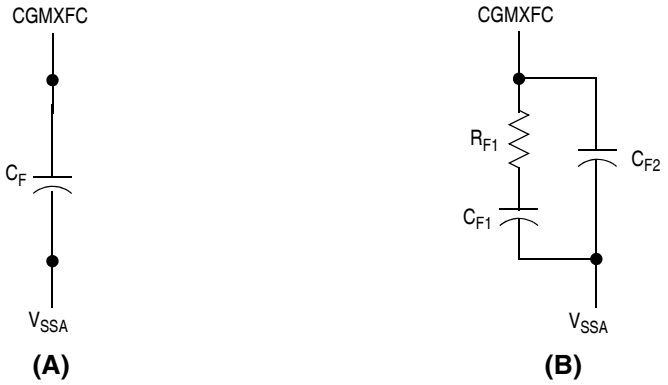



**Figure 4-9. PLL Filter**

**Table 4-5. Example Filter Component Values**

| $f_{RCLK}$ | $C_{F1}$ | $C_{F2}$ | $R_{F1}$ | $C_F$ |
|---|---|---|---|---|
| 1 MHz | 8.2 nF | 820 pF | 2k | 18 nF |
| 2 MHz | 4.7 nF | 470 pF | 2k | 6.8 nF |
| 3 MHz | 3.3 nF | 330 pF | 2k | 5.6 nF |
| 4 MHz | 2.2 nF | 220 pF | 2k | 4.7 nF |
| 5 MHz | 1.8 nF | 180 pF | 2k | 3.9 nF |
| 6 MHz | 1.5 nF | 150 pF | 2k | 3.3 nF |
| 7 MHz | 1.2 nF | 120 pF | 2k | 2.7 nF |
| 8 MHz | 1 nF | 100 pF | 2k | 2.2 nF |

# Chapter 5
# Configuration Register (CONFIG)

## 5.1 Introduction

This section describes the configuration registers, CONFIG1 and CONFIG2. The configuration registers enable or disable these options:

- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- COP timeout period (262,128 or 8176 CGMXCLK cycles)
- STOP instruction
- Computer operating properly module (COP)
- Low-voltage inhibit (LVI) module control and voltage trip point selection
- Enable/disable the oscillator (OSC) during stop mode
- Enable/disable an extra divide by 128 prescaler in timebase module
- Enable for scalable controller area network (MSCAN)
- Selectable clockout (MCLK) feature with divide by 1, 2, and 4 of the bus or crystal frequency
- Timebase clock select

## 5.2 Functional Description

The configuration registers are used in the initialization of various options. The configuration registers can be written once after each reset. All of the configuration register bits are cleared during reset. Since the various options affect the operation of the microcontroller unit (MCU), it is recommended that these registers be written immediately after reset. The configuration registers are located at $001E and $001F and may be read at anytime.

***NOTE***

*On a FLASH device, the options except MSCANEN and LVI5OR3 are one-time writable by the user after each reset. These bits are one-time writable by the user only after each POR (power-on reset). The CONFIG registers are not in the FLASH memory but are special registers containing one-time writable latches after each reset. Upon a reset, the CONFIG registers default to predetermined settings as shown in Figure 5-1 and Figure 5-2.*

Address: $001E

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | MCLKSEL | MCLK1 | MCLK0 | MSCANEN | TMBCLKSEL | OSCENINSTOP | SCIBDSRC |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | See note | 0 | 0 | 1 |

Note: MSCANEN is only reset via POR (power-on reset).

= Unimplemented

**Figure 5-1. Configuration Register 2 (CONFIG2)**

### MCLKSEL — MCLK Source Select Bit

1 = Crystal frequency
0 = Bus frequency

### MCLK1 and MCLK0 — MCLK Output Select Bits

Setting the MCLK1 and MCLK0 bits enables the PTD0/$\overline{SS}$ pin to be used as a MCLK output clock. Once configured for MCLK, the PTD data direction register for PTD0 is used to enable and disable the MCLK output. See Table 5-1 for MCLK options.

**Table 5-1. MCLK Output Select**

| MCLK1 | MCLK0 | MCLK Frequency |
|---|---|---|
| 0 | 0 | MCLK not enabled |
| 0 | 1 | Clock |
| 1 | 0 | Clock divided by 2 |
| 1 | 1 | Clock divided by 4 |

### MSCANEN— MSCAN08 Enable Bit

Setting the MSCANEN enables the MSCAN08 module and allows the MSCAN08 to use the PTC0/PTC1 pins. See Chapter 12 MSCAN08 Controller (MSCAN08) for a more detailed description of the MSCAN08 operation.

1 = Enables MSCAN08 module
0 = Disables the MSCAN08 module

***NOTE***

*The MSCANEN bit is cleared by a power-on reset (POR) only. Other resets will leave this bit unaffected.*

### TMBCLKSEL— Timebase Clock Select Bit

TMBCLKSEL enables an extra divide-by-128 prescaler in the timebase module. Setting this bit enables the extra prescaler and clearing this bit disables it. See Chapter 17 Timebase Module (TBM) for a more detailed description of the external clock operation.

1 = Enables extra divide-by-128 prescaler in timebase module
0 = Disables extra divide-by-128 prescaler in timebase module

**OSCENINSTOP — Oscillator Enable In Stop Mode Bit**

OSCENINSTOP, when set, will enable the oscillator to continue to generate clocks in stop mode. See Chapter 4 Clock Generator Module (CGM). This function is used to keep the timebase running while the rest of the MCU stops. See Chapter 17 Timebase Module (TBM). When clear, the oscillator will cease to generate clocks while in stop mode. The default state for this option is clear, disabling the oscillator in stop mode.

1 = Oscillator enabled during stop mode
0 = Oscillator disabled during stop mode (default)

**SCIBDSRC — SCI Baud Rate Clock Source Bit**

SCIBDSRC controls the clock source used for the serial communications interface (SCI). The setting of this bit affects the frequency at which the SCI operates.See Chapter 14 Enhanced Serial Communications Interface (ESCI) Module.

1 = Internal data bus clock used as clock source for SCI (default)
0 = External oscillator used as clock source for SCI

Address: $001F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | COPRS | LVISTOP | LVIRSTD | LVIPWRD | LVI5OR3 | SSREC | STOP | COPD |
| Reset: | 0 | 0 | 0 | 0 | See note | 0 | 0 | 0 |

Note: LVI5OR3 is only reset via POR (power-on reset).

**Figure 5-2. Configuration Register 1 (CONFIG1)**

**COPRS — COP Rate Select Bit**

COPRS selects the COP timeout period. Reset clears COPRS. See Chapter 6 Computer Operating Properly (COP) Module

1 = COP timeout period = 8176 CGMXCLK cycles
0 = COP timeout period = 262,128 CGMXCLK cycles

**LVISTOP — LVI Enable in Stop Mode Bit**

When the LVIPWRD bit is clear, setting the LVISTOP bit enables the LVI to operate during stop mode. Reset clears LVISTOP.

1 = LVI enabled during stop mode
0 = LVI disabled during stop mode

**LVIRSTD — LVI Reset Disable Bit**

LVIRSTD disables the reset signal from the LVI module.
See Chapter 11 Low-Voltage Inhibit (LVI).

1 = LVI module resets disabled
0 = LVI module resets enabled

**LVIPWRD — LVI Power Disable Bit**

LVIPWRD disables the LVI module. See Chapter 11 Low-Voltage Inhibit (LVI).

1 = LVI module power disabled
0 = LVI module power enabled

### LVI5OR3 — LVI 5-V or 3-V Operating Mode Bit

LVI5OR3 selects the voltage operating mode of the LVI module (see Chapter 11 Low-Voltage Inhibit (LVI)). The voltage mode selected for the LVI should match the operating $V_{DD}$ (see Chapter 21 Electrical Specifications) for the LVI's voltage trip points for each of the modes.

1 = LVI operates in 5-V mode
0 = LVI operates in 3-V mode

***NOTE***

*The LVI5OR3 bit is cleared by a power-on reset (POR) only. Other resets will leave this bit unaffected.*

### SSREC — Short Stop Recovery Bit

SSREC enables the CPU to exit stop mode with a delay of 32 CGMXCLK cycles instead of a 4096-CGMXCLK cycle delay.

1 = Stop mode recovery after 32 CGMXCLK cycles
0 = Stop mode recovery after 4096 CGMXCLK cycles

***NOTE***

*Exiting stop mode by any reset will result in the long stop recovery.*

The short stop recovery delay can be enabled when using a crystal or resonator and the OSCENINSTOP bit is set. The short stop recovery delay can be enabled when an external oscillator is used, regardless of the OSCENINSTOP setting.

The short stop recovery delay must be disabled when the OSCENINSTOP bit is clear and a crystal or resonator is used.

### STOP — STOP Instruction Enable Bit

STOP enables the STOP instruction.

1 = STOP instruction enabled
0 = STOP instruction treated as illegal opcode

### COPD — COP Disable Bit

COPD disables the COP module. See Chapter 6 Computer Operating Properly (COP) Module.

1 = COP module disabled
0 = COP module enabled

# Chapter 6
# Computer Operating Properly (COP) Module

## 6.1 Introduction

The computer operating properly (COP) module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by clearing the COP counter periodically. The COP module can be disabled through the COPD bit in the CONFIG register.

## 6.2 Functional Description

Figure 6-1 shows the structure of the COP module.

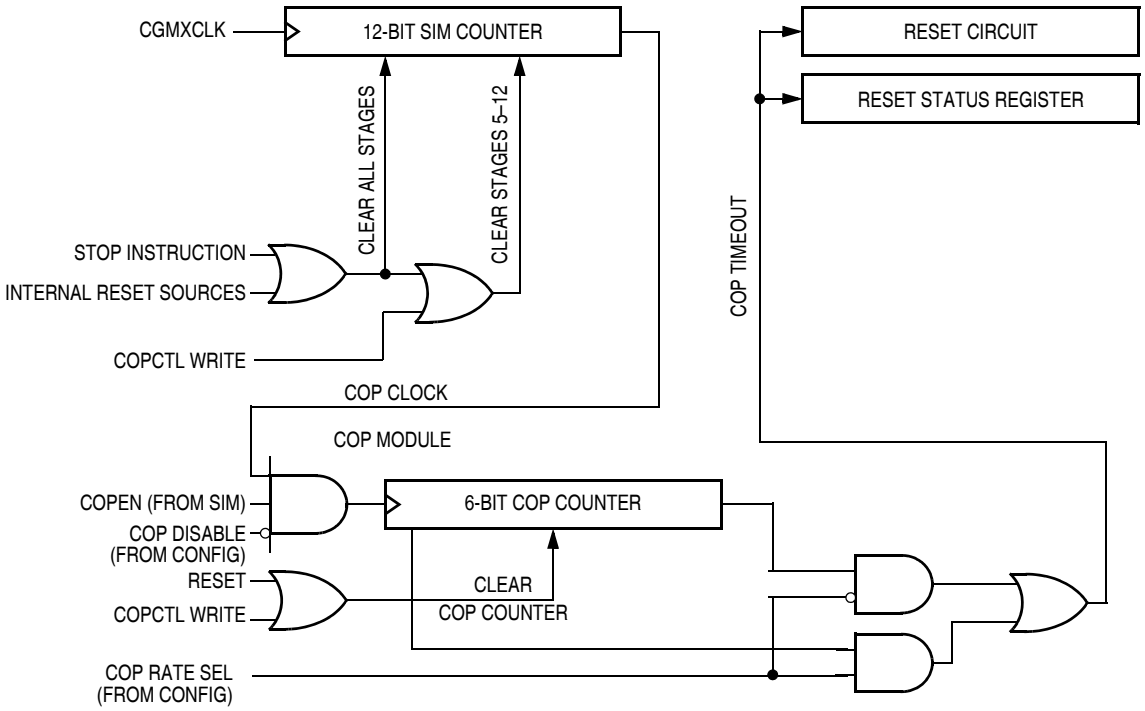



**Figure 6-1. COP Block Diagram**

The COP counter is a free-running 6-bit counter preceded by the 12-bit SIM counter. If not cleared by software, the COP counter overflows and generates an asynchronous reset after 262,128 or 8176 CGMXCLK cycles, depending on the state of the COP rate select bit, COPRS, in the configuration register. With a 262,128 CGMXCLK cycle overflow option, a 4.9152-MHz crystal gives a COP timeout period of 53.3 ms. Writing any value to location $FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 12–5 of the SIM counter.

***NOTE***

*Service the COP immediately after reset and before entering or after exiting stop mode to guarantee the maximum time before the first COP counter overflow.*

A COP reset pulls the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles and sets the COP bit in the reset status register (RSR).

In monitor mode, the COP is disabled if the $\overline{\text{RST}}$ pin or the $\overline{\text{IRQ}}$ is held at $V_{TST}$. During the break state, $V_{TST}$ on the $\overline{\text{RST}}$ pin disables the COP.

***NOTE***

*Place COP clearing instructions in the main program and not in an interrupt subroutine. Such an interrupt subroutine could keep the COP from generating a reset even while the main program is not working properly.*

## 6.3 I/O Signals

The following paragraphs describe the signals shown in Figure 6-1.

### 6.3.1 CGMXCLK

CGMXCLK is the crystal oscillator output signal. CGMXCLK frequency is equal to the crystal frequency.

### 6.3.2 STOP Instruction

The STOP instruction clears the SIM counter.

### 6.3.3 COPCTL Write

Writing any value to the COP control register (COPCTL) clears the COP counter and clears stages 12–5 of the SIM counter. Reading the COP control register returns the low byte of the reset vector. See 6.4 COP Control Register.

### 6.3.4 Power-On Reset

The power-on reset (POR) circuit clears the SIM counter 4096 CGMXCLK cycles after power-up.

### 6.3.5 Internal Reset

An internal reset clears the SIM counter and the COP counter.

### 6.3.6 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register. See Chapter 5 Configuration Register (CONFIG).

### 6.3.7 COPRS (COP Rate Select)

The COPRS signal reflects the state of the COP rate select bit (COPRS) in the configuration register. See Chapter 5 Configuration Register (CONFIG).

## 6.4 COP Control Register

The COP control register (COPCTL) is located at address $FFFF and overlaps the reset vector. Writing any value to $FFFF clears the COP counter and starts a new timeout period. Reading location $FFFF returns the low byte of the reset vector.

Address: $FFFF

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | Low byte of reset vector | | | | | | | |
| Write: | Clear COP counter | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |

**Figure 6-2. COP Control Register (COPCTL)**

## 6.5 Interrupts

The COP does not generate central processor unit (CPU) interrupt requests.

## 6.6 Monitor Mode

When monitor mode is entered with $V_{TST}$ on the $\overline{IRQ}$ pin, the COP is disabled as long as $V_{TST}$ remains on the $\overline{IRQ}$ pin or the $\overline{RST}$ pin. When monitor mode is entered by having blank reset vectors and not having $V_{TST}$ on the $\overline{IRQ}$ pin, the COP is automatically disabled until a POR occurs.

## 6.7 Low-Power Modes

The WAIT and STOP instructions put the microcontroller unit (MCU) in low power-consumption standby modes.

### 6.7.1 Wait Mode

The COP remains active during wait mode. If COP is enabled, a reset will occur at COP timeout.

### 6.7.2 Stop Mode

Stop mode turns off the CGMXCLK input to the COP and clears the SIM counter. Service the COP immediately before entering or after exiting stop mode to ensure a full COP timeout period after entering or exiting stop mode.

To prevent inadvertently turning off the COP with a STOP instruction, a configuration option is available that disables the STOP instruction. When the STOP bit in the configuration register has the STOP instruction disabled, execution of a STOP instruction results in an illegal opcode reset.

## 6.8 COP Module During Break Mode

The COP is disabled during a break interrupt when $V_{TST}$ is present on the $\overline{RST}$ pin.

# Chapter 7
# Central Processor Unit (CPU)

## 7.1 Introduction

The M68HC08 CPU (central processor unit) is an enhanced and fully object-code-compatible version of the M68HC05 CPU. The *CPU08 Reference Manual* (document order number CPU08RM/AD) contains a description of the CPU instruction set, addressing modes, and architecture.

## 7.2 Features

Features of the CPU include:

- Object code fully upward-compatible with M68HC05 Family
- 16-bit stack pointer with stack manipulation instructions
- 16-bit index register with x-register manipulation instructions
- 8-MHz CPU internal bus frequency
- 64-Kbyte program/data memory space
- 16 addressing modes
- Memory-to-memory data moves without using accumulator
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- Enhanced binary-coded decimal (BCD) data handling
- Modular architecture with expandable internal bus definition for extension of addressing range beyond 64 Kbytes
- Low-power stop and wait modes

## 7.3 CPU Registers

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

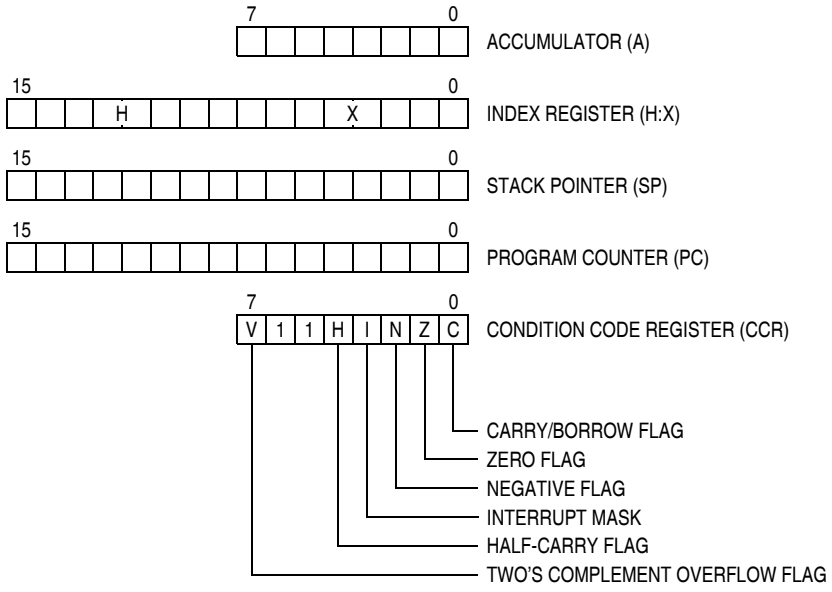



**Figure 7-1. CPU Registers**

### 7.3.1 Accumulator

The accumulator is a general-purpose 8-bit register. The CPU uses the accumulator to hold operands and the results of arithmetic/logic operations.

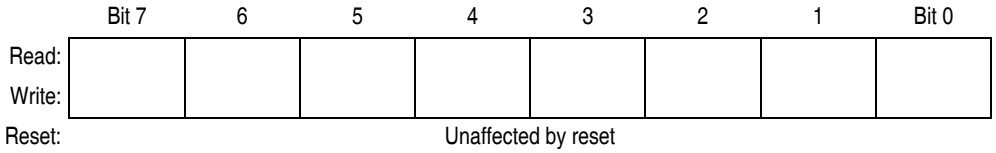



**Figure 7-2. Accumulator (A)**

### 7.3.2 Index Register

The 16-bit index register allows indexed addressing of a 64-Kbyte memory space. H is the upper byte of the index register, and X is the lower byte. H:X is the concatenated 16-bit index register.

In the indexed addressing modes, the CPU uses the contents of the index register to determine the conditional address of the operand.

The index register can serve also as a temporary data storage location.

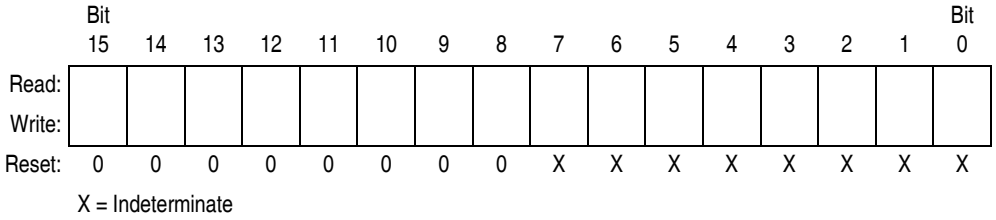



**Figure 7-3. Index Register (H:X)**

### 7.3.3 Stack Pointer

The stack pointer is a 16-bit register that contains the address of the next location on the stack. During a reset, the stack pointer is preset to $00FF. The reset stack pointer (RSP) instruction sets the least significant byte to $FF and does not affect the most significant byte. The stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

In the stack pointer 8-bit offset and 16-bit offset addressing modes, the stack pointer can function as an index register to access data on the stack. The CPU uses the contents of the stack pointer to determine the conditional address of the operand.

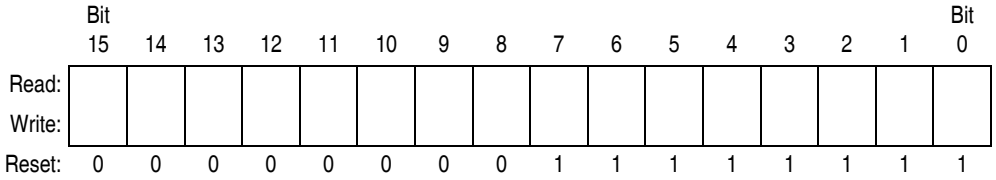



**Figure 7-4. Stack Pointer (SP)**

***NOTE***

*The location of the stack is arbitrary and may be relocated anywhere in random-access memory (RAM). Moving the SP out of page 0 ($0000 to $00FF) frees direct address (page 0) space. For correct operation, the stack pointer must point only to RAM locations.*

### 7.3.4 Program Counter

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

Normally, the program counter automatically increments to the next sequential memory location every time an instruction or operand is fetched. Jump, branch, and interrupt operations load the program counter with an address other than that of the next sequential location.

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

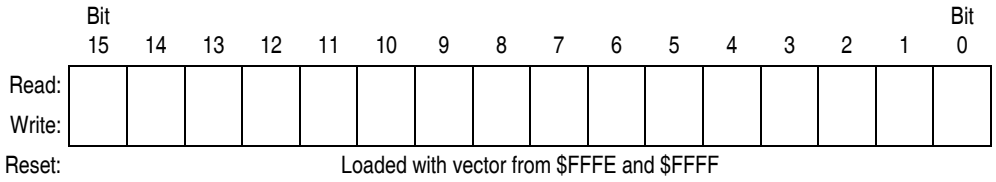



**Figure 7-5. Program Counter (PC)**

### 7.3.5 Condition Code Register

The 8-bit condition code register contains the interrupt mask 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 register.

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | V | 1 | 1 | H | I | N | Z | C |
| Reset: | X | 1 | 1 | X | 1 | X | X | X |

X = Indeterminate

**Figure 7-6. Condition Code Register (CCR)**

**V — 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.

1 = Overflow
0 = No overflow

**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 flags to determine the appropriate correction factor.

1 = Carry between bits 3 and 4
0 = No carry between bits 3 and 4

**I — Interrupt Mask**

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 interrupt vector is fetched.

1 = Interrupts disabled
0 = Interrupts enabled

***NOTE***

*To maintain M6805 Family compatibility, the upper byte of the index register (H) is not stacked automatically. If the interrupt service routine modifies H, then the user must stack and unstack H using the PSHH and PULH instructions.*

After the I bit is cleared, the highest-priority interrupt request is serviced first.

A return-from-interrupt (RTI) instruction pulls the CPU registers from the stack and restores the interrupt mask from the stack. After any reset, the interrupt mask is set and can be cleared only by the clear interrupt mask software instruction (CLI).

**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.

1 = Negative result
0 = Non-negative result

**Z — Zero Flag**

The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of $00.

1 = Zero result
0 = Non-zero result

**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.

1 = Carry out of bit 7
0 = No carry out of bit 7

## 7.4 Arithmetic/Logic Unit (ALU)

The ALU performs the arithmetic and logic operations defined by the instruction set.

Refer to the *CPU08 Reference Manual* (document order number CPU08RM/AD) for a description of the instructions and addressing modes and more detail about the architecture of the CPU.

## 7.5 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

### 7.5.1 Wait Mode

The WAIT instruction:

- Clears the interrupt mask (I bit) in the condition code register, enabling interrupts. After exit from wait mode by interrupt, the I bit remains clear. After exit by reset, the I bit is set.
- Disables the CPU clock

### 7.5.2 Stop Mode

The STOP instruction:

- Clears the interrupt mask (I bit) in the condition code register, enabling external interrupts. After exit from stop mode by external interrupt, the I bit remains clear. After exit by reset, the I bit is set.
- Disables the CPU clock

After exiting stop mode, the CPU clock begins running after the oscillator stabilization delay.

## 7.6 CPU During Break Interrupts

If a break module is present on the MCU, the CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with $FFFC:$FFFD or with $FEFC:$FEFD in monitor mode

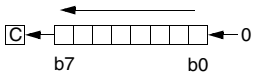
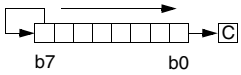
The break interrupt begins after completion of the CPU instruction in progress. If the break address register match occurs on the last cycle of a CPU instruction, the break interrupt begins immediately.

A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation if the break interrupt has been deasserted.

## 7.7 Instruction Set Summary

Table 7-1 provides a summary of the M68HC08 instruction set.

**Table 7-1. Instruction Set Summary (Sheet 1 of 6)**

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| ADC #*opr*<br>ADC *opr*<br>ADC *opr*<br>ADC *opr*,X<br>ADC *opr*,X<br>ADC ,X<br>ADC *opr*,SP<br>ADC *opr*,SP | Add with Carry | A ← (A) + (M) + (C) | ↕ | ↕ | – | ↕ | ↕ | ↕ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A9<br>B9<br>C9<br>D9<br>E9<br>F9<br>9EE9<br>9ED9 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| ADD #*opr*<br>ADD *opr*<br>ADD *opr*<br>ADD *opr*,X<br>ADD *opr*,X<br>ADD ,X<br>ADD *opr*,SP<br>ADD *opr*,SP | Add without Carry | A ← (A) + (M) | ↕ | ↕ | – | ↕ | ↕ | ↕ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | AB<br>BB<br>CB<br>DB<br>EB<br>FB<br>9EEB<br>9EDB | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| AIS #*opr* | Add Immediate Value (Signed) to SP | SP ← (SP) + (16 « M) | – | – | – | – | – | – | IMM | A7 | ii | 2 |
| AIX #*opr* | Add Immediate Value (Signed) to H:X | H:X ← (H:X) + (16 « M) | – | – | – | – | – | – | IMM | AF | ii | 2 |
| AND #*opr*<br>AND *opr*<br>AND *opr*<br>AND *opr*,X<br>AND *opr*,X<br>AND ,X<br>AND *opr*,SP<br>AND *opr*,SP | Logical AND | A ← (A) & (M) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A4<br>B4<br>C4<br>D4<br>E4<br>F4<br>9EE4<br>9ED4 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| ASL *opr*<br>ASLA<br>ASLX<br>ASL *opr*,X<br>ASL ,X<br>ASL *opr*,SP | Arithmetic Shift Left<br>(Same as LSL) |  | ↕ | – | – | ↕ | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38<br>48<br>58<br>68<br>78<br>9E68 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| ASR *opr*<br>ASRA<br>ASRX<br>ASR *opr*,X<br>ASR *opr*,X<br>ASR *opr*,SP | Arithmetic Shift Right |  | ↕ | – | – | ↕ | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 37<br>47<br>57<br>67<br>77<br>9E67 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| BCC *rel* | Branch if Carry Bit Clear | PC ← (PC) + 2 + rel ? (C) = 0 | – | – | – | – | – | – | REL | 24 | rr | 3 |
| BCLR *n*, *opr* | Clear Bit n in M | Mn ← 0 | – | – | – | – | – | – | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 11<br>13<br>15<br>17<br>19<br>1B<br>1D<br>1F | dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd | 4<br>4<br>4<br>4<br>4<br>4<br>4<br>4 |
| BCS *rel* | Branch if Carry Bit Set (Same as BLO) | PC ← (PC) + 2 + *rel* ? (C) = 1 | – | – | – | – | – | – | REL | 25 | rr | 3 |
| BEQ *rel* | Branch if Equal | PC ← (PC) + 2 + *rel* ? (Z) = 1 | – | – | – | – | – | – | REL | 27 | rr | 3 |
| BGE *opr* | Branch if Greater Than or Equal To (Signed Operands) | PC ← (PC) + 2 + *rel* ? (N ⊕ V) = 0 | – | – | – | – | – | – | REL | 90 | rr | 3 |
| BGT *opr* | Branch if Greater Than (Signed Operands) | PC ← (PC) + 2 + *rel* ? (Z) ǀ (N ⊕ V) = 0 | – | – | – | – | – | – | REL | 92 | rr | 3 |
| BHCC *rel* | Branch if Half Carry Bit Clear | PC ← (PC) + 2 + *rel* ? (H) = 0 | – | – | – | – | – | – | REL | 28 | rr | 3 |
| BHCS *rel* | Branch if Half Carry Bit Set | PC ← (PC) + 2 + *rel* ? (H) = 1 | – | – | – | – | – | – | REL | 29 | rr | 3 |
| BHI *rel* | Branch if Higher | PC ← (PC) + 2 + *rel* ? (C) ǀ (Z) = 0 | – | – | – | – | – | – | REL | 22 | rr | 3 |

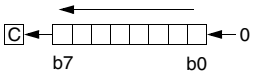
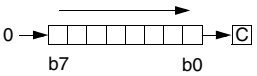
## Table 7-1. Instruction Set Summary (Sheet 2 of 6)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| BHS *rel* | Branch if Higher or Same (Same as BCC) | PC ← (PC) + 2 + *rel* ? (C) = 0 | – | – | – | – | – | – | REL | 24 | rr | 3 |
| BIH *rel* | Branch if $\overline{\text{IRQ}}$ Pin High | PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 1 | – | – | – | – | – | – | REL | 2F | rr | 3 |
| BIL *rel* | Branch if $\overline{\text{IRQ}}$ Pin Low | PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 0 | – | – | – | – | – | – | REL | 2E | rr | 3 |
| BIT #*opr*<br>BIT *opr*<br>BIT *opr*<br>BIT *opr*,X<br>BIT *opr*,X<br>BIT ,X<br>BIT *opr*,SP<br>BIT *opr*,SP | Bit Test | (A) & (M) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A5<br>B5<br>C5<br>D5<br>E5<br>F5<br>9EE5<br>9ED5 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| BLE *opr* | Branch if Less Than or Equal To (Signed Operands) | PC ← (PC) + 2 + *rel* ? (Z) ∣ (N ⊕ V) = 1 | – | – | – | – | – | – | REL | 93 | rr | 3 |
| BLO *rel* | Branch if Lower (Same as BCS) | PC ← (PC) + 2 + *rel* ? (C) = 1 | – | – | – | – | – | – | REL | 25 | rr | 3 |
| BLS *rel* | Branch if Lower or Same | PC ← (PC) + 2 + *rel* ? (C) ∣ (Z) = 1 | – | – | – | – | – | – | REL | 23 | rr | 3 |
| BLT *opr* | Branch if Less Than (Signed Operands) | PC ← (PC) + 2 + *rel* ? (N ⊕ V) =1 | – | – | – | – | – | – | REL | 91 | rr | 3 |
| BMC *rel* | Branch if Interrupt Mask Clear | PC ← (PC) + 2 + *rel* ? (I) = 0 | – | – | – | – | – | – | REL | 2C | rr | 3 |
| BMI *rel* | Branch if Minus | PC ← (PC) + 2 + *rel* ? (N) = 1 | – | – | – | – | – | – | REL | 2B | rr | 3 |
| BMS *rel* | Branch if Interrupt Mask Set | PC ← (PC) + 2 + *rel* ? (I) = 1 | – | – | – | – | – | – | REL | 2D | rr | 3 |
| BNE *rel* | Branch if Not Equal | PC ← (PC) + 2 + *rel* ? (Z) = 0 | – | – | – | – | – | – | REL | 26 | rr | 3 |
| BPL *rel* | Branch if Plus | PC ← (PC) + 2 + *rel* ? (N) = 0 | – | – | – | – | – | – | REL | 2A | rr | 3 |
| BRA *rel* | Branch Always | PC ← (PC) + 2 + *rel* | – | – | – | – | – | – | REL | 20 | rr | 3 |
| BRCLR *n*,*opr*,*rel* | Branch if Bit *n* in M Clear | PC ← (PC) + 3 + *rel* ? (Mn) = 0 | – | – | – | – | – | ↕ | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 01<br>03<br>05<br>07<br>09<br>0B<br>0D<br>0F | dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BRN *rel* | Branch Never | PC ← (PC) + 2 | – | – | – | – | – | – | REL | 21 | rr | 3 |
| BRSET *n*,*opr*,*rel* | Branch if Bit *n* in M Set | PC ← (PC) + 3 + *rel* ? (Mn) = 1 | – | – | – | – | – | ↕ | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 00<br>02<br>04<br>06<br>08<br>0A<br>0C<br>0E | dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BSET *n*,*opr* | Set Bit *n* in M | Mn ← 1 | – | – | – | – | – | – | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 10<br>12<br>14<br>16<br>18<br>1A<br>1C<br>1E | dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd | 4<br>4<br>4<br>4<br>4<br>4<br>4<br>4 |
| BSR *rel* | Branch to Subroutine | PC ← (PC) + 2; push (PCL)<br>SP ← (SP) – 1; push (PCH)<br>SP ← (SP) – 1<br>PC ← (PC) + *rel* | – | – | – | – | – | – | REL | AD | rr | 4 |
| CBEQ *opr*,*rel*<br>CBEQA #*opr*,*rel*<br>CBEQX #*opr*,*rel*<br>CBEQ *opr*,X+,*rel*<br>CBEQ X+,*rel*<br>CBEQ *opr*,SP,*rel* | Compare and Branch if Equal | PC ← (PC) + 3 + rel ? (A) – (M) = $00<br>PC ← (PC) + 3 + rel ? (A) – (M) = $00<br>PC ← (PC) + 3 + rel ? (X) – (M) = $00<br>PC ← (PC) + 3 + rel ? (A) – (M) = $00<br>PC ← (PC) + 2 + rel ? (A) – (M) = $00<br>PC ← (PC) + 4 + rel ? (A) – (M) = $00 | – | – | – | – | – | – | DIR<br>IMM<br>IMM<br>IX1+<br>IX+<br>SP1 | 31<br>41<br>51<br>61<br>71<br>9E61 | dd rr<br>ii rr<br>ii rr<br>ff rr<br>rr<br>ff rr | 5<br>4<br>4<br>5<br>4<br>6 |
| CLC | Clear Carry Bit | C ← 0 | – | – | – | – | – | 0 | INH | 98 | | 1 |
| CLI | Clear Interrupt Mask | I ← 0 | – | – | 0 | – | – | – | INH | 9A | | 2 |

## Table 7-1. Instruction Set Summary (Sheet 3 of 6)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| CLR *opr*<br>CLRA<br>CLRX<br>CLRH<br>CLR *opr*,X<br>CLR ,X<br>CLR *opr*,SP | Clear | M ← $00<br>A ← $00<br>X ← $00<br>H ← $00<br>M ← $00<br>M ← $00<br>M ← $00 | 0 | – | – | 0 | 1 | – | DIR<br>INH<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3F<br>4F<br>5F<br>8C<br>6F<br>7F<br>9E6F | dd<br><br><br><br>ff<br><br>ff | 3<br>1<br>1<br>1<br>3<br>2<br>4 |
| CMP #*opr*<br>CMP *opr*<br>CMP *opr*<br>CMP *opr*,X<br>CMP *opr*,X<br>CMP ,X<br>CMP *opr*,SP<br>CMP *opr*,SP | Compare A with M | (A) – (M) | ↕ | – | – | ↕ | ↕ | ↕ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A1<br>B1<br>C1<br>D1<br>E1<br>F1<br>9EE1<br>9ED1 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| COM *opr*<br>COMA<br>COMX<br>COM *opr*,X<br>COM ,X<br>COM *opr*,SP | Complement (One's Complement) | M ← ($\overline{M}$) = $FF – (M)<br>A ← ($\overline{A}$) = $FF – (M)<br>X ← ($\overline{X}$) = $FF – (M)<br>M ← ($\overline{M}$) = $FF – (M)<br>M ← ($\overline{M}$) = $FF – (M)<br>M ← ($\overline{M}$) = $FF – (M) | 0 | – | – | ↕ | ↕ | 1 | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 33<br>43<br>53<br>63<br>73<br>9E63 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| CPHX #*opr*<br>CPHX *opr* | Compare H:X with M | (H:X) – (M:M + 1) | ↕ | – | – | ↕ | ↕ | ↕ | IMM<br>DIR | 65<br>75 | ii ii+1<br>dd | 3<br>4 |
| CPX #*opr*<br>CPX *opr*<br>CPX *opr*<br>CPX ,X<br>CPX *opr*,X<br>CPX *opr*,X<br>CPX *opr*,SP<br>CPX *opr*,SP | Compare X with M | (X) – (M) | ↕ | – | – | ↕ | ↕ | ↕ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A3<br>B3<br>C3<br>D3<br>E3<br>F3<br>9EE3<br>9ED3 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| DAA | Decimal Adjust A | $(A)_{10}$ | U | – | – | ↕ | ↕ | ↕ | INH | 72 | | 2 |
| DBNZ *opr,rel*<br>DBNZA *rel*<br>DBNZX *rel*<br>DBNZ *opr*,X,*rel*<br>DBNZ X,*rel*<br>DBNZ *opr*,SP,*rel* | Decrement and Branch if Not Zero | A ← (A) – 1 or M ← (M) – 1 or X ← (X) – 1<br>PC ← (PC) + 3 + *rel* ? (result) ≠ 0<br>PC ← (PC) + 2 + *rel* ? (result) ≠ 0<br>PC ← (PC) + 2 + *rel* ? (result) ≠ 0<br>PC ← (PC) + 3 + *rel* ? (result) ≠ 0<br>PC ← (PC) + 2 + *rel* ? (result) ≠ 0<br>PC ← (PC) + 4 + *rel* ? (result) ≠ 0 | – | – | – | – | – | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3B<br>4B<br>5B<br>6B<br>7B<br>9E6B | dd rr<br>rr<br>rr<br>ff rr<br>rr<br>ff rr | 5<br>3<br>3<br>5<br>4<br>6 |
| DEC *opr*<br>DECA<br>DECX<br>DEC *opr*,X<br>DEC ,X<br>DEC *opr*,SP | Decrement | M ← (M) – 1<br>A ← (A) – 1<br>X ← (X) – 1<br>M ← (M) – 1<br>M ← (M) – 1<br>M ← (M) – 1 | ↕ | – | – | ↕ | ↕ | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3A<br>4A<br>5A<br>6A<br>7A<br>9E6A | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| DIV | Divide | A ← (H:A)/(X)<br>H ← Remainder | – | – | – | – | ↕ | ↕ | INH | 52 | | 7 |
| EOR #*opr*<br>EOR *opr*<br>EOR *opr*<br>EOR *opr*,X<br>EOR *opr*,X<br>EOR ,X<br>EOR *opr*,SP<br>EOR *opr*,SP | Exclusive OR M with A | A ← (A ⊕ M) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A8<br>B8<br>C8<br>D8<br>E8<br>F8<br>9EE8<br>9ED8 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| INC *opr*<br>INCA<br>INCX<br>INC *opr*,X<br>INC ,X<br>INC *opr*,SP | Increment | M ← (M) + 1<br>A ← (A) + 1<br>X ← (X) + 1<br>M ← (M) + 1<br>M ← (M) + 1<br>M ← (M) + 1 | ↕ | – | – | ↕ | ↕ | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3C<br>4C<br>5C<br>6C<br>7C<br>9E6C | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |

## Table 7-1. Instruction Set Summary (Sheet 4 of 6)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| JMP *opr*<br>JMP *opr*<br>JMP *opr*,X<br>JMP *opr*,X<br>JMP ,X | Jump | PC ← Jump Address | – | – | – | – | – | – | DIR<br>EXT<br>IX2<br>IX1<br>IX | BC<br>CC<br>DC<br>EC<br>FC | dd<br>hh ll<br>ee ff<br>ff | 2<br>3<br>4<br>3<br>2 |
| JSR *opr*<br>JSR *opr*<br>JSR *opr*,X<br>JSR *opr*,X<br>JSR ,X | Jump to Subroutine | PC ← (PC) + *n* (*n* = 1, 2, or 3)<br>Push (PCL); SP ← (SP) – 1<br>Push (PCH); SP ← (SP) – 1<br>PC ← Unconditional Address | – | – | – | – | – | – | DIR<br>EXT<br>IX2<br>IX1<br>IX | BD<br>CD<br>DD<br>ED<br>FD | dd<br>hh ll<br>ee ff<br>ff | 4<br>5<br>6<br>5<br>4 |
| LDA #*opr*<br>LDA *opr*<br>LDA *opr*<br>LDA *opr*,X<br>LDA *opr*,X<br>LDA ,X<br>LDA *opr*,SP<br>LDA *opr*,SP | Load A from M | A ← (M) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A6<br>B6<br>C6<br>D6<br>E6<br>F6<br>9EE6<br>9ED6 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| LDHX #*opr*<br>LDHX *opr* | Load H:X from M | H:X ← (M:M + 1) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR | 45<br>55 | ii jj<br>dd | 3<br>4 |
| LDX #*opr*<br>LDX *opr*<br>LDX *opr*<br>LDX *opr*,X<br>LDX *opr*,X<br>LDX ,X<br>LDX *opr*,SP<br>LDX *opr*,SP | Load X from M | X ← (M) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | AE<br>BE<br>CE<br>DE<br>EE<br>FE<br>9EEE<br>9EDE | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| LSL *opr*<br>LSLA<br>LSLX<br>LSL *opr*,X<br>LSL ,X<br>LSL *opr*,SP | Logical Shift Left<br>(Same as ASL) |  | ↕ | – | – | ↕ | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38<br>48<br>58<br>68<br>78<br>9E68 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| LSR *opr*<br>LSRA<br>LSR*X*<br>LSR *opr*,X<br>LSR ,X<br>LSR *opr*,SP | Logical Shift Right |  | ↕ | – | – | 0 | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 34<br>44<br>54<br>64<br>74<br>9E64 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| MOV *opr*,*opr*<br>MOV *opr*,X+<br>MOV #*opr*,*opr*<br>MOV X+,*opr* | Move | $(M)_{Destination} \leftarrow (M)_{Source}$<br><br>H:X ← (H:X) + 1 (IX+D, DIX+) | 0 | – | – | ↕ | ↕ | – | DD<br>DIX+<br>IMD<br>IX+D | 4E<br>5E<br>6E<br>7E | dd dd<br>dd<br>ii dd<br>dd | 5<br>4<br>4<br>4 |
| MUL | Unsigned multiply | X:A ← (X) × (A) | – | 0 | – | – | – | 0 | INH | 42 | | 5 |
| NEG *opr*<br>NEGA<br>NEGX<br>NEG *opr*,X<br>NEG ,X<br>NEG *opr*,SP | Negate (Two's Complement) | M ← –(M) = $00 – (M)<br>A ← –(A) = $00 – (A)<br>X ← –(X) = $00 – (X)<br>M ← –(M) = $00 – (M)<br>M ← –(M) = $00 – (M) | ↕ | – | – | ↕ | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 30<br>40<br>50<br>60<br>70<br>9E60 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| NOP | No Operation | None | – | – | – | – | – | – | INH | 9D | | 1 |
| NSA | Nibble Swap A | A ← (A[3:0]:A[7:4]) | – | – | – | – | – | – | INH | 62 | | 3 |
| ORA #*opr*<br>ORA *opr*<br>ORA *opr*<br>ORA *opr*,X<br>ORA *opr*,X<br>ORA ,X<br>ORA *opr*,SP<br>ORA *opr*,SP | Inclusive OR A and M | A ← (A) \| (M) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | AA<br>BA<br>CA<br>DA<br>EA<br>FA<br>9EEA<br>9EDA | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| PSHA | Push A onto Stack | Push (A); SP ← (SP) – 1 | – | – | – | – | – | – | INH | 87 | | 2 |
| PSHH | Push H onto Stack | Push (H); SP ← (SP) – 1 | – | – | – | – | – | – | INH | 8B | | 2 |
| PSHX | Push X onto Stack | Push (X); SP ← (SP) – 1 | – | – | – | – | – | – | INH | 89 | | 2 |

## Table 7-1. Instruction Set Summary (Sheet 5 of 6)

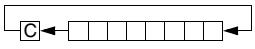
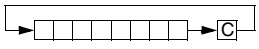
| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| PULA | Pull A from Stack | SP ← (SP + 1); Pull (A) | – | – | – | – | – | – | INH | 86 | | 2 |
| PULH | Pull H from Stack | SP ← (SP + 1); Pull (H) | – | – | – | – | – | – | INH | 8A | | 2 |
| PULX | Pull X from Stack | SP ← (SP + 1); Pull (X) | – | – | – | – | – | – | INH | 88 | | 2 |
| ROL *opr*<br>ROLA<br>ROLX<br>ROL *opr*,X<br>ROL ,X<br>ROL *opr*,SP | Rotate Left through Carry |  | ↕ | – | – | ↕ | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 39<br>49<br>59<br>69<br>79<br>9E69 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| ROR *opr*<br>RORA<br>RORX<br>ROR *opr*,X<br>ROR ,X<br>ROR *opr*,SP | Rotate Right through Carry |  | ↕ | – | – | ↕ | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 36<br>46<br>56<br>66<br>76<br>9E66 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| RSP | Reset Stack Pointer | SP ← $FF | – | – | – | – | – | – | INH | 9C | | 1 |
| RTI | Return from Interrupt | SP ← (SP) + 1; Pull (CCR)<br>SP ← (SP) + 1; Pull (A)<br>SP ← (SP) + 1; Pull (X)<br>SP ← (SP) + 1; Pull (PCH)<br>SP ← (SP) + 1; Pull (PCL) | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | INH | 80 | | 7 |
| RTS | Return from Subroutine | SP ← SP + 1; Pull (PCH)<br>SP ← SP + 1; Pull (PCL) | – | – | – | – | – | – | INH | 81 | | 4 |
| SBC #*opr*<br>SBC *opr*<br>SBC *opr*<br>SBC *opr*,X<br>SBC *opr*,X<br>SBC ,X<br>SBC *opr*,SP<br>SBC *opr*,SP | Subtract with Carry | A ← (A) – (M) – (C) | ↕ | – | – | ↕ | ↕ | ↕ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A2<br>B2<br>C2<br>D2<br>E2<br>F2<br>9EE2<br>9ED2 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| SEC | Set Carry Bit | C ← 1 | – | – | – | – | – | 1 | INH | 99 | | 1 |
| SEI | Set Interrupt Mask | I ← 1 | – | – | 1 | – | – | – | INH | 9B | | 2 |
| STA *opr*<br>STA *opr*<br>STA *opr*,X<br>STA *opr*,X<br>STA ,X<br>STA *opr*,SP<br>STA *opr*,SP | Store A in M | M ← (A) | 0 | – | – | ↕ | ↕ | – | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | B7<br>C7<br>D7<br>E7<br>F7<br>9EE7<br>9ED7 | dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 3<br>4<br>4<br>3<br>2<br>4<br>5 |
| STHX *opr* | Store H:X in M | (M:M + 1) ← (H:X) | 0 | – | – | ↕ | ↕ | – | DIR | 35 | dd | 4 |
| STOP | Enable Interrupts, Stop Processing, Refer to MCU Documentation | I ← 0; Stop Processing | – | – | 0 | – | – | – | INH | 8E | | 1 |
| STX *opr*<br>STX *opr*<br>STX *opr*,X<br>STX *opr*,X<br>STX ,X<br>STX *opr*,SP<br>STX *opr*,SP | Store X in M | M ← (X) | 0 | – | – | ↕ | ↕ | – | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | BF<br>CF<br>DF<br>EF<br>FF<br>9EEF<br>9EDF | dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 3<br>4<br>4<br>3<br>2<br>4<br>5 |
| SUB #*opr*<br>SUB *opr*<br>SUB *opr*<br>SUB *opr*,X<br>SUB *opr*,X<br>SUB ,X<br>SUB *opr*,SP<br>SUB *opr*,SP | Subtract | A ← (A) – (M) | ↕ | – | – | ↕ | ↕ | ↕ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A0<br>B0<br>C0<br>D0<br>E0<br>F0<br>9EE0<br>9ED0 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |

**Table 7-1. Instruction Set Summary (Sheet 6 of 6)**

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| SWI | Software Interrupt | PC ← (PC) + 1; Push (PCL)<br>SP ← (SP) – 1; Push (PCH)<br>SP ← (SP) – 1; Push (X)<br>SP ← (SP) – 1; Push (A)<br>SP ← (SP) – 1; Push (CCR)<br>SP ← (SP) – 1; I ← 1<br>PCH ← Interrupt Vector High Byte<br>PCL ← Interrupt Vector Low Byte | – | – | 1 | – | – | – | INH | 83 | | 9 |
| TAP | Transfer A to CCR | CCR ← (A) | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | INH | 84 | | 2 |
| TAX | Transfer A to X | X ← (A) | – | – | – | – | – | – | INH | 97 | | 1 |
| TPA | Transfer CCR to A | A ← (CCR) | – | – | – | – | – | – | INH | 85 | | 1 |
| TST *opr*<br>TSTA<br>TSTX<br>TST *opr*,X<br>TST ,X<br>TST *opr*,SP | Test for Negative or Zero | (A) – $00 or (X) – $00 or (M) – $00 | 0 | – | – | ↕ | ↕ | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3D<br>4D<br>5D<br>6D<br>7D<br>9E6D | dd<br><br><br>ff<br><br>ff | 3<br>1<br>1<br>3<br>2<br>4 |
| TSX | Transfer SP to H:X | H:X ← (SP) + 1 | – | – | – | – | – | – | INH | 95 | | 2 |
| TXA | Transfer X to A | A ← (X) | – | – | – | – | – | – | INH | 9F | | 1 |
| TXS | Transfer H:X to SP | (SP) ← (H:X) – 1 | – | – | – | – | – | – | INH | 94 | | 2 |
| WAIT | Enable Interrupts; Wait for Interrupt | I bit ← 0; Inhibit CPU clocking until interrupted | – | – | 0 | – | – | – | INH | 8F | | 1 |

| | |
|---|---|
| A | Accumulator |
| C | Carry/borrow bit |
| CCR | Condition code register |
| dd | Direct address of operand |
| dd rr | Direct address of operand and relative offset of branch instruction |
| DD | Direct to direct addressing mode |
| DIR | Direct addressing mode |
| DIX+ | Direct to indexed with post increment addressing mode |
| ee ff | High and low bytes of offset in indexed, 16-bit offset addressing |
| EXT | Extended addressing mode |
| ff | Offset byte in indexed, 8-bit offset addressing |
| H | Half-carry bit |
| H | Index register high byte |
| hh ll | High and low bytes of operand address in extended addressing |
| I | Interrupt mask |
| ii | Immediate operand byte |
| IMD | Immediate source to direct destination addressing mode |
| IMM | Immediate addressing mode |
| INH | Inherent addressing mode |
| IX | Indexed, no offset addressing mode |
| IX+ | Indexed, no offset, post increment addressing mode |
| IX+D | Indexed with post increment to direct addressing mode |
| IX1 | Indexed, 8-bit offset addressing mode |
| IX1+ | Indexed, 8-bit offset, post increment addressing mode |
| IX2 | Indexed, 16-bit offset addressing mode |
| M | Memory location |
| N | Negative bit |
| *n* | Any bit |
| *opr* | Operand (one or two bytes) |
| PC | Program counter |
| PCH | Program counter high byte |
| PCL | Program counter low byte |
| REL | Relative addressing mode |
| *rel* | Relative program counter offset byte |
| rr | Relative program counter offset byte |
| SP1 | Stack pointer, 8-bit offset addressing mode |
| SP2 | Stack pointer 16-bit offset addressing mode |
| SP | Stack pointer |
| U | Undefined |
| V | Overflow bit |
| X | Index register low byte |
| Z | Zero bit |
| & | Logical AND |
| \| | Logical OR |
| ⊕ | Logical EXCLUSIVE OR |
| ( ) | Contents of |
| –( ) | Negation (two's complement) |
| # | Immediate value |
| « | Sign extend |
| ← | Loaded with |
| ? | If |
| : | Concatenated with |
| ↕ | Set or cleared |
| — | Not affected |

## 7.8 Opcode Map

See Table 7-2.

# Table 7-2. Opcode Map

| | Bit Manipulation | | Branch | Read-Modify-Write | | | | | | Control | | Register/Mem | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DIR | DIR | REL | DIR | INH | INH | IX1 | SP1 | IX | INH | INH | IMM | DIR | EXT | IX2 | S |
| MSB / LSB | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 9E6 | 7 | 8 | 9 | A | B | C | D | 9I |
| 0 | 5 BRSET0 3 DIR | 4 BSET0 2 DIR | 3 BRA 2 REL | 4 NEG 2 DIR | 1 NEGA 1 INH | 1 NEGX 1 INH | 4 NEG 2 IX1 | 5 NEG 3 SP1 | 3 NEG 1 IX | 7 RTI 1 INH | 3 BGE 2 REL | 2 SUB 2 IMM | 3 SUB 2 DIR | 4 SUB 3 EXT | 4 SUB 3 IX2 | SU 4 |
| 1 | 5 BRCLR0 3 DIR | 4 BCLR0 2 DIR | 3 BRN 2 REL | 5 CBEQ 3 DIR | 4 CBEQA 3 IMM | 4 CBEQX 3 IMM | 5 CBEQ 3 IX1+ | 6 CBEQ 4 SP1 | 4 CBEQ 2 IX+ | 4 RTS 1 INH | 3 BLT 2 REL | 2 CMP 2 IMM | 3 CMP 2 DIR | 4 CMP 3 EXT | 4 CMP 3 IX2 | CI 4 |
| 2 | 5 BRSET1 3 DIR | 4 BSET1 2 DIR | 3 BHI 2 REL | | 5 MUL 1 INH | 7 DIV 1 INH | 3 NSA 1 INH | | 2 DAA 1 INH | | 3 BGT 2 REL | 2 SBC 2 IMM | 3 SBC 2 DIR | 4 SBC 3 EXT | 4 SBC 3 IX2 | SI 4 |
| 3 | 5 BRCLR1 3 DIR | 4 BCLR1 2 DIR | 3 BLS 2 REL | 4 COM 2 DIR | 1 COMA 1 INH | 1 COMX 1 INH | 4 COM 2 IX1 | 5 COM 3 SP1 | 3 COM 1 IX | 9 SWI 1 INH | 3 BLE 2 REL | 2 CPX 2 IMM | 3 CPX 2 DIR | 4 CPX 3 EXT | 4 CPX 3 IX2 | C 4 |
| 4 | 5 BRSET2 3 DIR | 4 BSET2 2 DIR | 3 BCC 2 REL | 4 LSR 2 DIR | 1 LSRA 1 INH | 1 LSRX 1 INH | 4 LSR 2 IX1 | 5 LSR 3 SP1 | 3 LSR 1 IX | 2 TAP 1 INH | 2 TXS 1 INH | 2 AND 2 IMM | 3 AND 2 DIR | 4 AND 3 EXT | 4 AND 3 IX2 | AI 4 |
| 5 | 5 BRCLR2 3 DIR | 4 BCLR2 2 DIR | 3 BCS 2 REL | 4 STHX 2 DIR | 3 LDHX 3 IMM | 4 LDHX 2 DIR | 3 CPHX 3 IMM | | 4 CPHX 2 DIR | 1 TPA 1 INH | 2 TSX 1 INH | 2 BIT 2 IMM | 3 BIT 2 DIR | 4 BIT 3 EXT | 4 BIT 3 IX2 | B 4 |
| 6 | 5 BRSET3 3 DIR | 4 BSET3 2 DIR | 3 BNE 2 REL | 4 ROR 2 DIR | 1 RORA 1 INH | 1 RORX 1 INH | 4 ROR 2 IX1 | 5 ROR 3 SP1 | 3 ROR 1 IX | 2 PULA 1 INH | | 2 LDA 2 IMM | 3 LDA 2 DIR | 4 LDA 3 EXT | 4 LDA 3 IX2 | LI 4 |
| 7 | 5 BRCLR3 3 DIR | 4 BCLR3 2 DIR | 3 BEQ 2 REL | 4 ASR 2 DIR | 1 ASRA 1 INH | 1 ASRX 1 INH | 4 ASR 2 IX1 | 5 ASR 3 SP1 | 3 ASR 1 IX | 2 PSHA 1 INH | 1 TAX 1 INH | 2 AIS 2 IMM | 3 STA 2 DIR | 4 STA 3 EXT | 4 STA 3 IX2 | S 4 |
| 8 | 5 BRSET4 3 DIR | 4 BSET4 2 DIR | 3 BHCC 2 REL | 4 LSL 2 DIR | 1 LSLA 1 INH | 1 LSLX 1 INH | 4 LSL 2 IX1 | 5 LSL 3 SP1 | 3 LSL 1 IX | 2 PULX 1 INH | 1 CLC 1 INH | 2 EOR 2 IMM | 3 EOR 2 DIR | 4 EOR 3 EXT | 4 EOR 3 IX2 | EO 4 |
| 9 | 5 BRCLR4 3 DIR | 4 BCLR4 2 DIR | 3 BHCS 2 REL | 4 ROL 2 DIR | 1 ROLA 1 INH | 1 ROLX 1 INH | 4 ROL 2 IX1 | 5 ROL 3 SP1 | 3 ROL 1 IX | 2 PSHX 1 INH | 1 SEC 1 INH | 2 ADC 2 IMM | 3 ADC 2 DIR | 4 ADC 3 EXT | 4 ADC 3 IX2 | AI 4 |
| A | 5 BRSET5 3 DIR | 4 BSET5 2 DIR | 3 BPL 2 REL | 4 DEC 2 DIR | 1 DECA 1 INH | 1 DECX 1 INH | 4 DEC 2 IX1 | 5 DEC 3 SP1 | 3 DEC 1 IX | 2 PULH 1 INH | 2 CLI 1 INH | 2 ORA 2 IMM | 3 ORA 2 DIR | 4 ORA 3 EXT | 4 ORA 3 IX2 | O 4 |
| B | 5 BRCLR5 3 DIR | 4 BCLR5 2 DIR | 3 BMI 2 REL | 5 DBNZ 3 DIR | 3 DBNZA 2 INH | 3 DBNZX 2 INH | 5 DBNZ 3 IX1 | 6 DBNZ 4 SP1 | 4 DBNZ 2 IX | 2 PSHH 1 INH | 2 SEI 1 INH | 2 ADD 2 IMM | 3 ADD 2 DIR | 4 ADD 3 EXT | 4 ADD 3 IX2 | AI 4 |
| C | 5 BRSET6 3 DIR | 4 BSET6 2 DIR | 3 BMC 2 REL | 4 INC 2 DIR | 1 INCA 1 INH | 1 INCX 1 INH | 4 INC 2 IX1 | 5 INC 3 SP1 | 3 INC 1 IX | 1 CLRH 1 INH | 1 RSP 1 INH | | 2 JMP 2 DIR | 3 JMP 3 EXT | 4 JMP 3 IX2 | |
| D | 5 BRCLR6 3 DIR | 4 BCLR6 2 DIR | 3 BMS 2 REL | 3 TST 2 DIR | 1 TSTA 1 INH | 1 TSTX 1 INH | 3 TST 2 IX1 | 4 TST 3 SP1 | 2 TST 1 IX | | 1 NOP 1 INH | 4 BSR 2 REL | 4 JSR 2 DIR | 5 JSR 3 EXT | 6 JSR 3 IX2 | |
| E | 5 BRSET7 3 DIR | 4 BSET7 2 DIR | 3 BIL 2 REL | | 5 MOV 3 DD | 4 MOV 2 DIX+ | 4 MOV 3 IMD | | 4 MOV 2 IX+D | 1 STOP 1 INH | * | 2 LDX 2 IMM | 3 LDX 2 DIR | 4 LDX 3 EXT | 4 LDX 3 IX2 | LI 4 |
| F | 5 BRCLR7 3 DIR | 4 BCLR7 2 DIR | 3 BIH 2 REL | 3 CLR 2 DIR | 1 CLRA 1 INH | 1 CLRX 1 INH | 3 CLR 2 IX1 | 4 CLR 3 SP1 | 2 CLR 1 IX | 1 WAIT 1 INH | 1 TXA 1 INH | 2 AIX 2 IMM | 3 STX 2 DIR | 4 STX 3 EXT | 4 STX 3 IX2 | ST 4 |

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

*Pre-byte for stack pointer indexed instructions

Low Byte of Opcode in Hexadecimal

| MSB / LSB | 0 |
|---|---|
| 0 | 5 BRSET0 3 DIR |

High Byte of C
Cycles
Opcode Mnem
Number of Byt

# Chapter 8
# External Interrupt (IRQ)

## 8.1 Introduction

The IRQ (external interrupt) module provides a maskable interrupt input.

## 8.2 Features

Features of the IRQ module include:

- A dedicated external interrupt pin ($\overline{\text{IRQ}}$)
- IRQ interrupt control bits
- Hysteresis buffer
- Programmable edge-only or edge and level interrupt sensitivity
- Automatic interrupt acknowledge
- Internal pullup resistor

## 8.3 Functional Description

A low applied to the external interrupt pin can latch a central processor unit (CPU) interrupt request. Figure 8-1 shows the structure of the IRQ module.

Interrupt signals on the $\overline{\text{IRQ}}$ pin are latched into the IRQ latch. An interrupt latch remains set until one of the following actions occurs:

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the latch that caused the vector fetch.
- Software clear — Software can clear an interrupt latch by writing to the appropriate acknowledge bit in the interrupt status and control register (INTSCR). Writing a 1 to the ACK bit clears the IRQ latch.
- Reset — A reset automatically clears the interrupt latch.

The external interrupt pin is falling-edge triggered out of reset and is software-configurable to be either falling-edge or falling-edge and low-level triggered. The MODE bit in the INTSCR controls the triggering sensitivity of the $\overline{\text{IRQ}}$ pin.

When an interrupt pin is edge-triggered only (MODE = 0), the interrupt remains set until a vector fetch, software clear, or reset occurs.

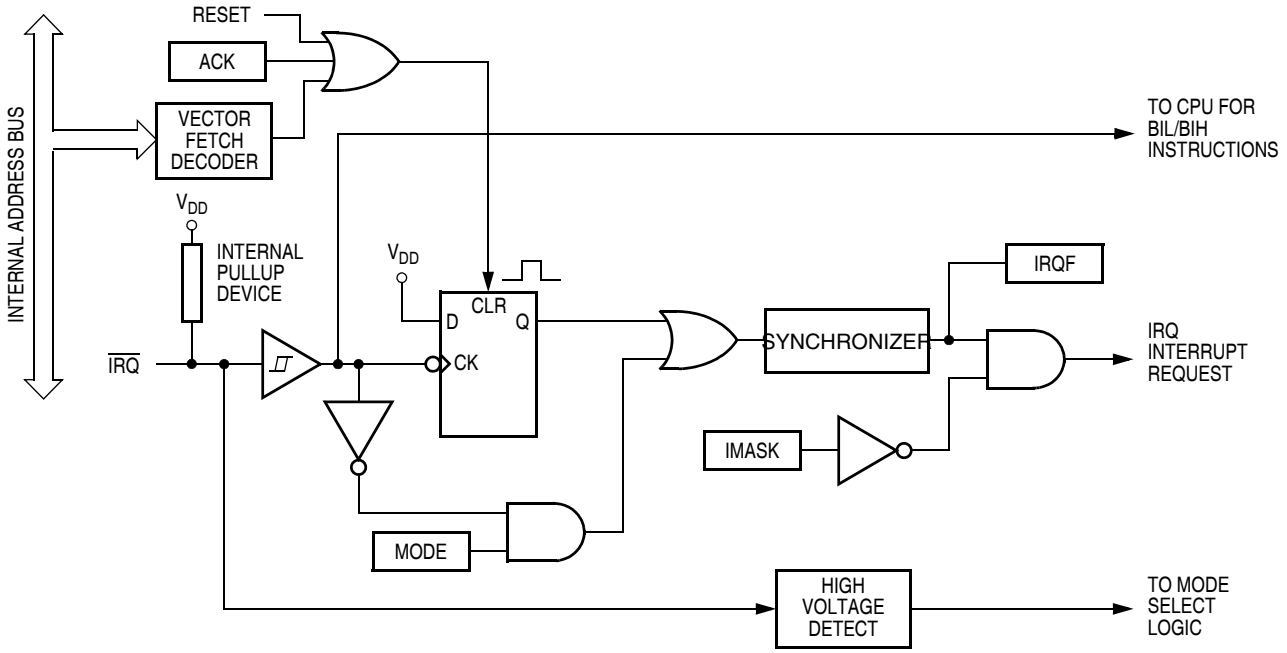



**Figure 8-1. IRQ Module Block Diagram**

When an interrupt pin is both falling-edge and low-level triggered (MODE = 1), the interrupt remains set until both of these events occur:

- Vector fetch or software clear
- Return of the interrupt pin to a high level

The vector fetch or software clear may occur before or after the interrupt pin returns to a high level. As long as the pin is low, the interrupt request remains pending. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

When set, the IMASK bit in the INTSCR masks all external interrupt requests. A latched interrupt request is not presented to the interrupt priority logic unless the IMASK bit is clear.

***NOTE***

*The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests.*

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $001D | IRQ Status and Control Register (INTSCR) See page 114. | Read: | 0 | 0 | 0 | 0 | IRQF | 0 | IMASK | MODE |
| | | Write: | | | | | | ACK | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 8-2. IRQ I/O Register Summary**

## 8.4 $\overline{\text{IRQ}}$ Pin

A falling edge on the $\overline{\text{IRQ}}$ pin can latch an interrupt request into the IRQ latch. A vector fetch, software clear, or reset clears the IRQ latch.

If the MODE bit is set, the $\overline{\text{IRQ}}$ pin is both falling-edge-sensitive and low-level-sensitive. With MODE set, both of the following actions must occur to clear IRQ:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the latch. Software may generate the interrupt acknowledge signal by writing a 1 to the ACK bit in the interrupt status and control register (INTSCR). The ACK bit is useful in applications that poll the $\overline{\text{IRQ}}$ pin and require software to clear the IRQ latch. Writing to the ACK bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACK does not affect subsequent transitions on the $\overline{\text{IRQ}}$ pin. A falling edge that occurs after writing to the ACK bit latches another interrupt request. If the IRQ mask bit, IMASK, is clear, the CPU loads the program counter with the vector address at locations $FFFA and $FFFB.
- Return of the $\overline{\text{IRQ}}$ pin to a high level — As long as the $\overline{\text{IRQ}}$ pin is low, IRQ remains active.

The vector fetch or software clear and the return of the $\overline{\text{IRQ}}$ pin to a high level may occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ}}$ pin is low. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

If the MODE bit is clear, the $\overline{\text{IRQ}}$ pin is falling-edge-sensitive only. With MODE clear, a vector fetch or software clear immediately clears the IRQ latch.

The IRQF bit in the INTSCR register can be used to check for pending interrupts. The IRQF bit is not affected by the IMASK bit, which makes it useful in applications where polling is preferred.

Use the BIH or BIL instruction to read the logic level on the $\overline{\text{IRQ}}$ pin.

***NOTE***

*When using the level-sensitive interrupt trigger, avoid false interrupts by masking interrupt requests in the interrupt routine.*

## 8.5 IRQ Module During Break Interrupts

The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear the latch during the break state. See Chapter 20 Development Support.

To allow software to clear the IRQ latch during a break interrupt, write a 1 to the BCFE bit. If a latch is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect CPU interrupt flags during the break state, write a 0 to the BCFE bit. With BCFE at 0 (its default state), writing to the ACK bit in the IRQ status and control register during the break state has no effect on the IRQ interrupt flags.

## 8.6 IRQ Status and Control Register

The IRQ status and control register (INTSCR) controls and monitors operation of the IRQ module. The INTSCR:

- Shows the state of the IRQ flag
- Clears the IRQ latch
- Masks IRQ interrupt request
- Controls triggering sensitivity of the $\overline{\text{IRQ}}$ interrupt pin

Address: $001D

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | IRQF | 0 | IMASK | MODE |
| Write: | | | | | | ACK | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 8-3. IRQ Status and Control Register (INTSCR)**

**IRQF — IRQ Flag Bit**

This read-only status bit is high when the IRQ interrupt is pending.

1 = $\overline{\text{IRQ}}$ interrupt pending
0 = $\overline{\text{IRQ}}$ interrupt not pending

**ACK — IRQ Interrupt Request Acknowledge Bit**

Writing a 1 to this write-only bit clears the IRQ latch. ACK always reads as 0. Reset clears ACK.

**IMASK — IRQ Interrupt Mask Bit**

Writing a 1 to this read/write bit disables IRQ interrupt requests. Reset clears IMASK.

1 = IRQ interrupt requests disabled
0 = IRQ interrupt requests enabled

**MODE — IRQ Edge/Level Select Bit**

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ}}$ pin. Reset clears MODE.

1 = $\overline{\text{IRQ}}$ interrupt requests on falling edges and low levels
0 = $\overline{\text{IRQ}}$ interrupt requests on falling edges only

# Chapter 9
# Keyboard Interrupt Module (KBI)

## 9.1 Introduction

The keyboard interrupt module (KBI) provides eight independently maskable external interrupts which are accessible via PTA0–PTA7. When a port pin is enabled for keyboard interrupt function, an internal pullup/pulldown device is also enabled on the pin.

## 9.2 Features

Features include:

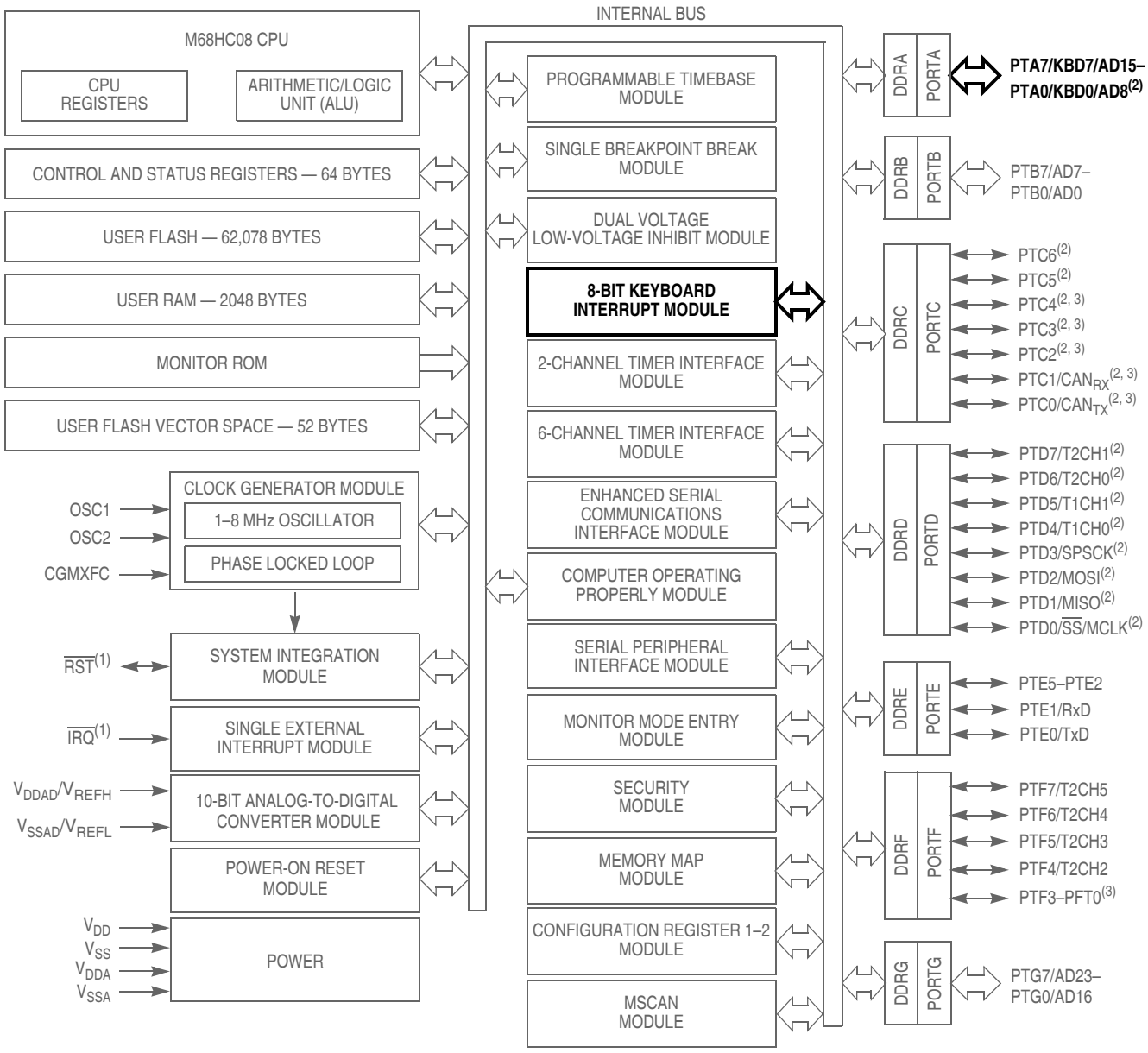
- Eight keyboard interrupt pins with separate keyboard interrupt enable bits and one keyboard interrupt mask
- Hysteresis buffers
- Programmable edge-only or edge- and level- interrupt sensitivity
- Edge detect programmable for rising or falling edges
- Level detect programmable for high or low levels
- Exit from low-power modes
- Pullup/pulldown device automatically configured based on polarity of edge/level selection

## 9.3 Functional Description

Writing to the KBIE7–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port A pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its internal pullup/pulldown device. On falling edge or low level selection a pullup device is configured. On rising edge or high level selection a pulldown device is configured.

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

A keyboard interrupt is latched when one or more keyboard pins are asserted. The MODEK bit in the keyboard status and control register controls the triggering mode of the keyboard interrupt.




1. Pin contains integrated pullup device.
**2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.**
3. Higher current drive port pins

**Figure 9-1. Block Diagram Highlighting KBI Block and Pins**

The KBIP7–KBIP0 bits determine the polarity of the keyboard pin detection. These bits along with the MODEK bit determine whether a logic level (0 or 1) and/or a falling (or rising) edge is being detected.

- If the keyboard interrupt is edge-sensitive only, a falling (or rising) edge on a keyboard pin does not latch an interrupt request if another keyboard pin is already asserted. To prevent losing an interrupt request on one pin because another pin is still asserted, software can disable the latter pin while it is asserted.
- If the keyboard interrupt is edge and level sensitive, an interrupt request is present as long as any keyboard interrupt pin is asserted and the pin is keyboard interrupt enabled.

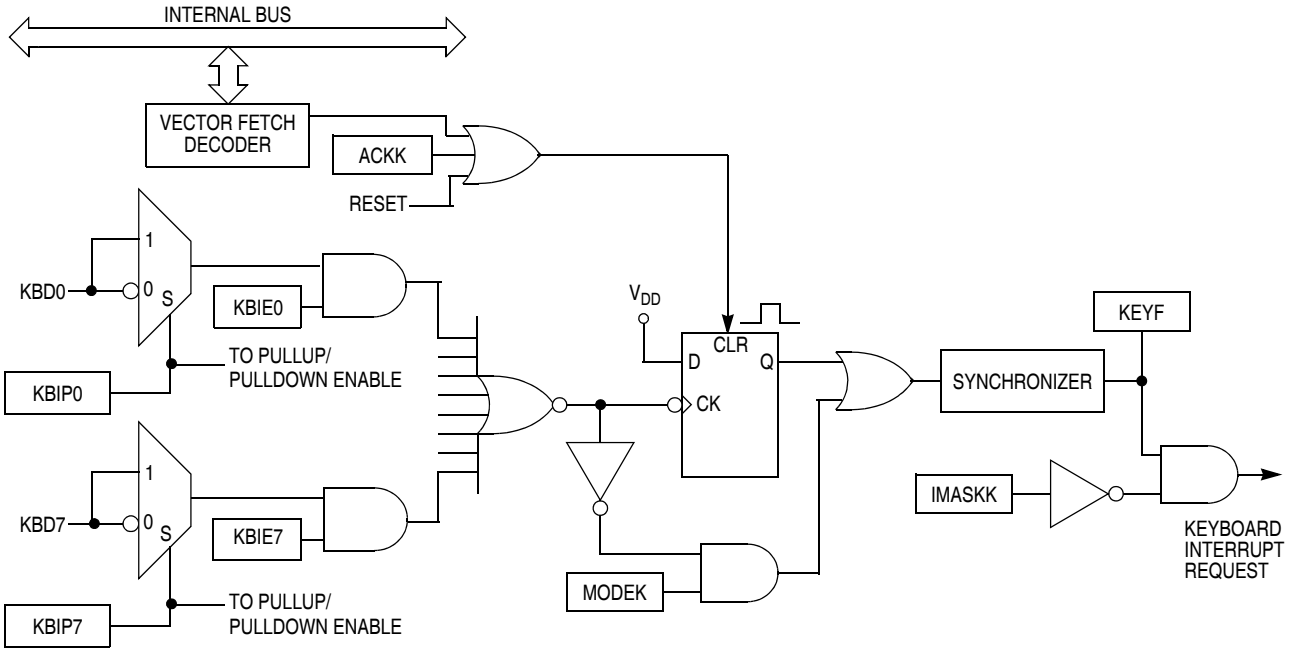



**Figure 9-2. Keyboard Module Block Diagram**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $001A | Keyboard Status and Control Register (INTKBSCR) See page 120. | Read: | 0 | 0 | 0 | 0 | KEYF | 0 | IMASKK | MODEK |
| | | Write: | | | | | | ACKK | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $001B | Keyboard Interrupt Enable Register (INTKBIER) See page 121. | Read: | KBIE7 | KBIE6 | KBIE5 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0448 | Keyboard Interrupt Polarity Register (INTKBIPR) See page 121. | Read: | KBIP7 | KBIP6 | KBIP5 | KBIP4 | KBIP3 | KBIP2 | KBIP1 | KBIP0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 9-3. I/O Register Summary**

If the MODEK bit is set and depending on the KBIPx bit, the keyboard interrupt pins are both falling (or rising) edge and low (or high) level sensitive, and both of the following actions must occur to clear a keyboard interrupt request:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software may generate the interrupt acknowledge signal by writing a 1 to the ACKK bit in the keyboard status and control register (INTKBSCR). The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling (or rising) edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the vector address at locations $FFE0 and $FFE1.
- Return of all enabled keyboard interrupt pins to 1 (or 0) — As long as any enabled keyboard interrupt pin is 0 (or 1), the keyboard interrupt remains set.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to 1 (or 0) may occur in any order.

If the MODEK bit is clear and depending on the KBIPx bit, the keyboard interrupt pin is falling (or rising) edge sensitive only. With MODEK clear, a vector fetch or software clear immediately clears the keyboard interrupt request.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt pin stays at 0 (or 1).

The keyboard flag bit (KEYF) in the keyboard status and control register can be used to see if a pending interrupt exists. The KEYF bit is not affected by the keyboard interrupt mask bit (IMASKK) which makes it useful in applications where polling is preferred.

To determine the logic level on a keyboard interrupt pin, use the data direction register to configure the pin as an input and read the data register.

***NOTE***

*Setting a keyboard interrupt enable bit (KBIEx) forces the corresponding keyboard interrupt pin to be an input, overriding the data direction register. However, the data direction register bit must be a 0 for software to read the pin.*

## 9.4 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup/pulldown device to reach a 1 (or 0). Therefore, a false interrupt can occur as soon as the pin is enabled.

To prevent a false interrupt on keyboard initialization:

1. Mask keyboard interrupts by setting the IMASKK bit in the keyboard status and control register.
2. Enable the KBI pins and polarity by setting the appropriate KBIEx bits in the keyboard interrupt enable register and the KBIPx bits in the keyboard interrupt polarity register.
3. Write to the ACKK bit in the keyboard status and control register to clear any false interrupts.
4. Clear the IMASKK bit.

An interrupt signal on an edge-triggered pin can be acknowledged immediately after enabling the pin. An interrupt signal on an edge- and level-triggered interrupt pin must be acknowledged after a delay that depends on the external load.

Another way to avoid a false interrupt:

1. Configure the keyboard pins as outputs by setting the appropriate DDRA bits in data direction register A.
2. Write 1s (or 0s) to the appropriate port A data register bits.
3. Enable the KBI pins and polarity by setting the appropriate KBIEx bits in the keyboard interrupt enable register and the KBIPx bits in the keyboard interrupt polarity register.

## 9.5 Low-Power Modes

The WAIT and STOP instructions put the microcontroller unit (MCU) in low power-consumption standby modes.

### 9.5.1 Wait Mode

The keyboard module remains active in wait mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of wait mode.

### 9.5.2 Stop Mode

The keyboard module remains active in stop mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of stop mode.

## 9.6 Keyboard Module During Break Interrupts

The system integration module (SIM) controls whether the keyboard interrupt latch can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state.

To allow software to clear the keyboard interrupt latch during a break interrupt, write a 1 to the BCFE bit. If a latch is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect the latch during the break state, write a 0 to the BCFE bit. With BCFE at 0 (its default state), writing to the keyboard acknowledge bit (ACKK) in the keyboard status and control register during the break state has no effect. See 9.7.1 Keyboard Status and Control Register.

## 9.7 I/O Registers

These registers control and monitor operation of the keyboard module:

- Keyboard status and control register (INTKBSCR)
- Keyboard interrupt enable register (INTKBIER)
- Keyboard interrupt polarity register (INTKBIPR)

### 9.7.1 Keyboard Status and Control Register

The keyboard status and control register:

- Flags keyboard interrupt requests
- Acknowledges keyboard interrupt requests
- Masks keyboard interrupt requests
- Controls keyboard interrupt triggering sensitivity

Address: $001A

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | KEYF | 0 | IMASKK | MODEK |
| Write: | | | | | | ACKK | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 9-4. Keyboard Status and Control Register (INTKBSCR)**

**Bits 7–4 — Not used**

These read-only bits always read as 0s.

**KEYF — Keyboard Flag Bit**

This read-only bit is set when a keyboard interrupt is pending. Reset clears the KEYF bit.

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

**ACKK — Keyboard Acknowledge Bit**

Writing a 1 to this write-only bit clears the keyboard interrupt request. ACKK always reads as 0. Reset clears ACKK.

**IMASKK — Keyboard Interrupt Mask Bit**

Writing a 1 to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests. Reset clears the IMASKK bit.

1 = Keyboard interrupt requests masked
0 = Keyboard interrupt requests not masked

**MODEK — Keyboard Triggering Sensitivity Bit**

This read/write bit controls the triggering sensitivity of the keyboard interrupt pins. Reset clears MODEK.

1 = Keyboard interrupt requests on edge and level detect
0 = Keyboard interrupt requests on edges only

### 9.7.2 Keyboard Interrupt Enable Register

The keyboard interrupt enable register enables or disables each port A pin to operate as a keyboard interrupt pin.

Address: $001B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | KBIE7 | KBIE6 | KBIE5 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-5. Keyboard Interrupt Enable Register (INTKBIER)**

**KBIE7–KBIE0 — Keyboard Interrupt Enable Bits**

Each of these read/write bits enables the corresponding keyboard interrupt pin to latch interrupt requests. Reset clears the keyboard interrupt enable register.

1 = PTAx pin enabled as keyboard interrupt pin
0 = PTAx pin not enabled as keyboard interrupt pin

### 9.7.3 Keyboard Interrupt Polarity Register

The KBIP7–KBIP0 bits determine the polarity of the keyboard pin detection. These bits along with the MODEK bit determine whether a logic level (0 or 1) and/or a falling (or rising) edge is being detected. The KBIPx bits also select the pullup resistor (KBIPx = 0) or pulldown resistor (KBIPx = 1) for each enabled keyboard interrupt pin.

Address: $0448

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | KBIP7 | KBIP6 | KBIP5 | KBIP4 | KBIP3 | KBIP2 | KBIP1 | KBIP0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 9-6. Keyboard Interrupt Polarity Register (INTKBIPR)**

**KBIP7–KBIP0 — Keyboard Interrupt Polarity Bits**

Each of these read/write bits enables the polarity of the keyboard interrupt pin. Reset clears the keyboard interrupt polarity register.

1 = Keyboard polarity is rising edge and/or high level
0 = Keyboard polarity is falling edge and/or low level

# Chapter 10
# Low-Power Modes

## 10.1 Introduction

The microcontroller (MCU) may enter two low-power modes: wait mode and stop mode. They are common to all HC08 MCUs and are entered through instruction execution. This section describes how each module acts in the low-power modes.

### 10.1.1 Wait Mode

The WAIT instruction puts the MCU in a low-power standby mode in which the central processor unit (CPU) clock is disabled but the bus clock continues to run. Power consumption can be further reduced by disabling the low-voltage inhibit (LVI) module through bits in the CONFIG1 register. See Chapter 5 Configuration Register (CONFIG).

### 10.1.2 Stop Mode

Stop mode is entered when a STOP instruction is executed. The CPU clock is disabled and the bus clock is disabled if the OSCENINSTOP bit in the CONFIG2 register is a 0. See Chapter 5 Configuration Register (CONFIG).

## 10.2 Analog-to-Digital Converter (ADC)

### 10.2.1 Wait Mode

The analog-to-digital converter (ADC) continues normal operation during wait mode. Any enabled CPU interrupt request from the ADC can bring the MCU out of wait mode. If the ADC is not required to bring the MCU out of wait mode, power down the ADC by setting ADCH4–ADCH0 bits in the ADC status and control register before executing the WAIT instruction.

### 10.2.2 Stop Mode

The ADC module is inactive after the execution of a STOP instruction. Any pending conversion is aborted. ADC conversions resume when the MCU exits stop mode after an external interrupt. Allow one conversion cycle to stabilize the analog circuitry.

## 10.3 Break Module (BRK)

### 10.3.1 Wait Mode

The break (BRK) module is active in wait mode. In the break routine, the user can subtract one from the return address on the stack if the SBSW bit in the break status register is set.

### 10.3.2 Stop Mode

The break module is inactive in stop mode. The STOP instruction does not affect break module register states.

## 10.4 Central Processor Unit (CPU)

### 10.4.1 Wait Mode

The WAIT instruction:

- Clears the interrupt mask (I bit) in the condition code register, enabling interrupts. After exit from wait mode by interrupt, the I bit remains clear. After exit by reset, the I bit is set.
- Disables the CPU clock

### 10.4.2 Stop Mode

The STOP instruction:

- Clears the interrupt mask (I bit) in the condition code register, enabling external interrupts. After exit from stop mode by external interrupt, the I bit remains clear. After exit by reset, the I bit is set.
- Disables the CPU clock

After exiting stop mode, the CPU clock begins running after the oscillator stabilization delay.

## 10.5 Clock Generator Module (CGM)

### 10.5.1 Wait Mode

The clock generator module (CGM) remains active in wait mode. Before entering wait mode, software can disengage and turn off the PLL by clearing the BCS and PLLON bits in the PLL control register (PCTL). Less power-sensitive applications can disengage the PLL without turning it off. Applications that require the PLL to wake the MCU from wait mode also can deselect the PLL output without turning off the PLL.

### 10.5.2 Stop Mode

If the OSCENINSTOP bit in the CONFIG2 register is cleared (default), then the STOP instruction disables the CGM (oscillator and phase-locked loop) and holds low all CGM outputs (CGMXCLK, CGMOUT, and CGMINT).

If the OSCENINSTOP bit in the CONFIG2 register is set, then the phase locked loop is shut off, but the oscillator will continue to operate in stop mode.

## 10.6 Computer Operating Properly Module (COP)

### 10.6.1 Wait Mode

The COP remains active during wait mode. If COP is enabled, a reset will occur at COP timeout.

### 10.6.2 Stop Mode

Stop mode turns off the CGMXCLK input to the COP and clears the SIM counter. Service the COP immediately before entering or after exiting stop mode to ensure a full COP timeout period after entering or exiting stop mode.

The STOP bit in the CONFIG1 register enables the STOP instruction. To prevent inadvertently turning off the COP with a STOP instruction, disable the STOP instruction by clearing the STOP bit.

## 10.7 External Interrupt Module (IRQ)

### 10.7.1 Wait Mode

The external interrupt (IRQ) module remains active in wait mode. Clearing the IMASK bit in the IRQ status and control register enables $\overline{\text{IRQ}}$ CPU interrupt requests to bring the MCU out of wait mode.

### 10.7.2 Stop Mode

The IRQ module remains active in stop mode. Clearing the IMASK bit in the IRQ status and control register enables $\overline{\text{IRQ}}$ CPU interrupt requests to bring the MCU out of stop mode.

## 10.8 Keyboard Interrupt Module (KBI)

### 10.8.1 Wait Mode

The keyboard interrupt (KBI) module remains active in wait mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of wait mode.

### 10.8.2 Stop Mode

The keyboard module remains active in stop mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of stop mode.

## 10.9 Low-Voltage Inhibit Module (LVI)

### 10.9.1 Wait Mode

If enabled, the low-voltage inhibit (LVI) module remains active in wait mode. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of wait mode.

### 10.9.2 Stop Mode

If enabled, the LVI module remains active in stop mode. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of stop mode.

## 10.10 Enhanced Serial Communications Interface Module (ESCI)

### 10.10.1 Wait Mode

The enhanced serial communications interface (ESCI), or SCI module for short, module remains active in wait mode. Any enabled CPU interrupt request from the SCI module can bring the MCU out of wait mode.

If SCI module functions are not required during wait mode, reduce power consumption by disabling the module before executing the WAIT instruction.

### 10.10.2 Stop Mode

The SCI module is inactive in stop mode. The STOP instruction does not affect SCI register states. SCI module operation resumes after the MCU exits stop mode.

Because the internal clock is inactive during stop mode, entering stop mode during an SCI transmission or reception results in invalid data.

## 10.11 Serial Peripheral Interface Module (SPI)

### 10.11.1 Wait Mode

The serial peripheral interface (SPI) module remains active in wait mode. Any enabled CPU interrupt request from the SPI module can bring the MCU out of wait mode.

If SPI module functions are not required during wait mode, reduce power consumption by disabling the SPI module before executing the WAIT instruction.

### 10.11.2 Stop Mode

The SPI module is inactive in stop mode. The STOP instruction does not affect SPI register states. SPI operation resumes after an external interrupt. If stop mode is exited by reset, any transfer in progress is aborted, and the SPI is reset.

## 10.12 Timer Interface Module (TIM1 and TIM2)

### 10.12.1 Wait Mode

The timer interface modules (TIM) remain active in wait mode. Any enabled CPU interrupt request from the TIM can bring the MCU out of wait mode.

If TIM functions are not required during wait mode, reduce power consumption by stopping the TIM before executing the WAIT instruction.

### 10.12.2 Stop Mode

The TIM is inactive in stop mode. The STOP instruction does not affect register states or the state of the TIM counter. TIM operation resumes when the MCU exits stop mode after an external interrupt.

## 10.13 Timebase Module (TBM)

### 10.13.1 Wait Mode

The timebase module (TBM) remains active after execution of the WAIT instruction. In wait mode, the timebase register is not accessible by the CPU.

If the timebase functions are not required during wait mode, reduce the power consumption by stopping the timebase before enabling the WAIT instruction.

### 10.13.2 Stop Mode

The timebase module may remain active after execution of the STOP instruction if the oscillator has been enabled to operate during stop mode through the OSCENINSTOP bit in the CONFIG2 register. The timebase module can be used in this mode to generate a periodic wakeup from stop mode.

If the oscillator has not been enabled to operate in stop mode, the timebase module will not be active during stop mode. In stop mode, the timebase register is not accessible by the CPU.

If the timebase functions are not required during stop mode, reduce the power consumption by stopping the timebase before enabling the STOP instruction.

## 10.14 Scalable Controller Area Network Module (MSCAN)

### 10.14.1 Wait Mode

The scalable controller area network (MSCAN) module remains active after execution of the WAIT instruction. In wait mode, the MSCAN08 registers are not accessible by the CPU.

If the MSCAN08 functions are not required during wait mode, reduce the power consumption by disabling the MSCAN08 module before enabling the WAIT instruction.

### 10.14.2 Stop Mode

The MSCAN08 module is inactive in stop mode. The STOP instruction does not affect MSCAN08 register states.

Because the internal clock is inactive during stop mode, entering stop mode during an MSCAN08 transmission or reception results in invalid data.

## 10.15 Exiting Wait Mode

These events restart the CPU clock and load the program counter with the reset vector or with an interrupt vector:

- External reset — A low on the $\overline{\text{RST}}$ pin resets the MCU and loads the program counter with the contents of locations $FFFE and $FFFF.
- External interrupt — A high-to-low transition on an external interrupt pin ($\overline{\text{IRQ}}$ pin) loads the program counter with the contents of locations: $FFFA and $FFFB; $\overline{\text{IRQ}}$ pin.
- Break interrupt — In emulation mode, a break interrupt loads the program counter with the contents of $FFFC and $FFFD.
- Computer operating properly (COP) module reset — A timeout of the COP counter resets the MCU and loads the program counter with the contents of $FFFE and $FFFF.
- Low-voltage inhibit (LVI) module reset — A power supply voltage below the $V_{TRIPF}$ voltage resets the MCU and loads the program counter with the contents of locations $FFFE and $FFFF.
- Clock generator module (CGM) interrupt — A CPU interrupt request from the CGM loads the program counter with the contents of $FFF8 and $FFF9.
- Keyboard interrupt (KBI) module — A CPU interrupt request from the KBI module loads the program counter with the contents of $FFE0 and $FFE1.
- Timer 1 interface (TIM1) module interrupt — A CPU interrupt request from the TIM1 loads the program counter with the contents of:
  - $FFF2 and $FFF3; TIM1 overflow
  - $FFF4 and $FFF5; TIM1 channel 1
  - $FFF6 and $FFF7; TIM1 channel 0
- Timer 2 interface module (TIM2) interrupt — A CPU interrupt request from the TIM2 loads the program counter with the contents of:
  - $FFEC and $FFED; TIM2 overflow
  - $FFEE and $FFEF; TIM2 channel 1
  - $FFF0 and $FFF1; TIM2 channel 0
  - $FFCC and $FFCD; TIM2 channel 5
  - $FFCE and $FFCF; TIM2 channel 4
  - $FFD0 and $FFD1; TIM2 channel 3
  - $FFD2 and $FFD3; TIM2 channel 2
- Serial peripheral interface (SPI) module interrupt — A CPU interrupt request from the SPI loads the program counter with the contents of:
  - $FFE8 and $FFE9; SPI transmitter
  - $FFEA and $FFEB; SPI receiver
- Serial communications interface (SCI) module interrupt — A CPU interrupt request from the SCI loads the program counter with the contents of:
  - $FFE2 and $FFE3; SCI transmitter
  - $FFE4 and $FFE5; SCI receiver
  - $FFE6 and $FFE7; SCI receiver error
- Analog-to-digital converter (ADC) module interrupt — A CPU interrupt request from the ADC loads the program counter with the contents of: $FFDE and $FFDF; ADC conversion complete.
- Timebase module (TBM) interrupt — A CPU interrupt request from the TBM loads the program counter with the contents of: $FFDC and $FFDD; TBM interrupt.

- MSCAN module interrupt — A CPU interrupt request from the MSCAN08 loads the program counter with the contents of:
  - $FFD4 and $FFD5; MSCAN08 transmitter
  - $FFD6 and $FFD7; MSCAN08 receiver
  - $FFD8 and $FFD9; MSCAN08 error
  - $FFDA and $FFDB; MSCAN08 wakeup

## 10.16 Exiting Stop Mode

These events restart the system clocks and load the program counter with the reset vector or with an interrupt vector:

- External reset — A low on the $\overline{\text{RST}}$ pin resets the MCU and loads the program counter with the contents of locations $FFFE and $FFFF.
- External interrupt — A high-to-low transition on an external interrupt pin loads the program counter with the contents of locations:
  - $FFFA and $FFFB; $\overline{\text{IRQ}}$ pin
  - $FFE0 and $FFE1; keyboard interrupt pins (low-to-high transition when KBIPx bits are set)
- Low-voltage inhibit (LVI) reset — A power supply voltage below the $V_{TRIPF}$ voltage resets the MCU and loads the program counter with the contents of locations $FFFE and $FFFF.
- Break interrupt — In emulation mode, a break interrupt loads the program counter with the contents of locations $FFFC and $FFFD.
- Timebase module (TBM) interrupt — A TBM interrupt loads the program counter with the contents of locations $FFDC and $FFDD when the timebase counter has rolled over. This allows the TBM to generate a periodic wakeup from stop mode.
- MSCAN08 interrupt — MSCAN08 bus activity can wake the MCU from CPU stop. However, until the oscillator starts up and synchronization is achieved the MSCAN08 will not respond to incoming data.

Upon exit from stop mode, the system clocks begin running after an oscillator stabilization delay. A 12-bit stop recovery counter inhibits the system clocks for 4096 CGMXCLK cycles after the reset or external interrupt.

The short stop recovery bit, SSREC, in the CONFIG1 register controls the oscillator stabilization delay during stop recovery. Setting SSREC reduces stop recovery time from 4096 CGMXCLK cycles to 32 CGMXCLK cycles.

***NOTE***

*Use the full stop recovery time (SSREC = 0) in applications that use an external crystal unless the OSCENINSTOP* bit is set.

# Chapter 11
# Low-Voltage Inhibit (LVI)

## 11.1 Introduction

This section describes the low-voltage inhibit (LVI) module, which monitors the voltage on the $V_{DD}$ pin and can force a reset when the $V_{DD}$ voltage falls below the LVI trip falling voltage, $V_{TRIPF}$.

## 11.2 Features

Features of the LVI module include:

- Programmable LVI reset
- Selectable LVI trip voltage
- Programmable stop mode operation

## 11.3 Functional Description

Figure 11-1 shows the structure of the LVI module. The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. Clearing the LVI power disable bit, LVIPWRD, enables the LVI to monitor $V_{DD}$ voltage. Clearing the LVI reset disable bit, LVIRSTD, enables the LVI module to generate a reset when $V_{DD}$ falls below a voltage, $V_{TRIPF}$. Setting the LVI enable in stop mode bit, LVISTOP, enables the LVI to operate in stop mode. Setting the LVI 5-V or 3-V trip point bit, LVI5OR3, enables the trip point voltage, $V_{TRIPF}$, to be configured for 5-V operation. Clearing the LVI5OR3 bit enables the trip point voltage, $V_{TRIPF}$, to be configured for 3-V operation. The actual trip points are shown in Chapter 21 Electrical Specifications.

***NOTE***

*After a power-on reset (POR) the LVI's default mode of operation is 3 V. If a 5-V system is used, the user must set the LVI5OR3 bit to raise the trip point to 5-V operation. Note that this must be done after every power-on reset since the default will revert back to 3-V mode after each power-on reset. If the $V_{DD}$ supply is below the 5-V mode trip voltage but above the 3-V mode trip voltage when POR is released, the part will operate because $V_{TRIPF}$ defaults to 3-V mode after a POR. So, in a 5-V system care must be taken to ensure that $V_{DD}$ is above the 5-V mode trip voltage after POR is released.*

*If the user requires 5-V mode and sets the LVI5OR3 bit after a power-on reset while the $V_{DD}$ supply is not above the $V_{TRIPR}$ for 5-V mode, the microcontroller unit (MCU) will immediately go into reset. The LVI in this case will hold the part in reset until either $V_{DD}$ goes above the rising 5-V trip point, $V_{TRIPR}$, which will release reset or $V_{DD}$ decreases to approximately 0 V which will re-trigger the power-on reset and reset the trip point to 3-V operation.*

LVISTOP, LVIPWRD, LVI5OR3, and LVIRSTD are in the configuration register (CONFIG1). See Figure 5-2. Configuration Register 1 (CONFIG1) for details of the LVI's configuration bits. Once an LVI reset occurs, the MCU remains in reset until $V_{DD}$ rises above a voltage, $V_{TRIPR}$, which causes the MCU to exit reset. See 15.3.2.5 Low-Voltage Inhibit (LVI) Reset for details of the interaction between the SIM and the LVI. The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

An LVI reset also drives the $\overline{RST}$ pin low to provide low-voltage protection to external peripheral devices.

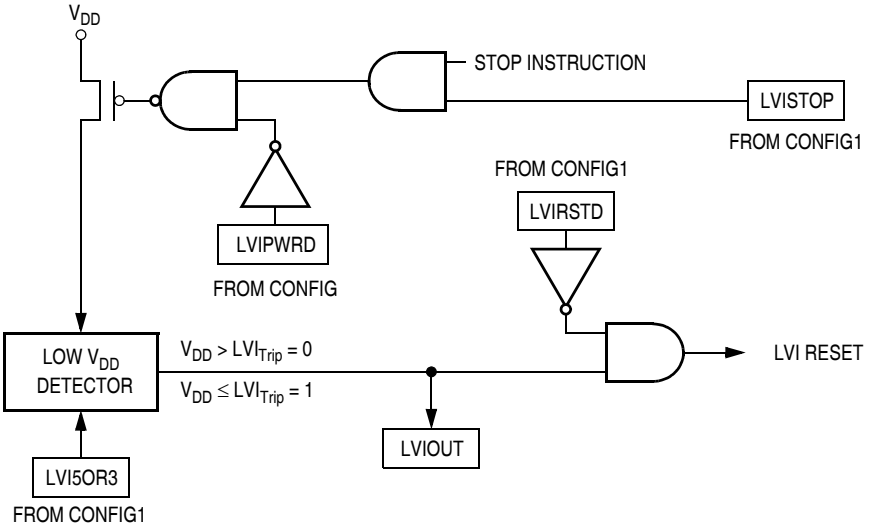



**Figure 11-1. LVI Module Block Diagram**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FE0C | LVI Status Register (LVISR) See page 133. | Read: | LVIOUT | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-2. LVI I/O Register Summary**

### 11.3.1 Polled LVI Operation

In applications that can operate at $V_{DD}$ levels below the $V_{TRIPF}$ level, software can monitor $V_{DD}$ by polling the LVIOUT bit. In the configuration register, the LVIPWRD bit must be 0 to enable the LVI module, and the LVIRSTD bit must be 1 to disable LVI resets.

### 11.3.2 Forced Reset Operation

In applications that require $V_{DD}$ to remain above the $V_{TRIPF}$ level, enabling LVI resets allows the LVI module to reset the MCU when $V_{DD}$ falls below the $V_{TRIPF}$ level. In the configuration register, the LVIPWRD and LVIRSTD bits must be cleared to enable the LVI module and to enable LVI resets.

### 11.3.3 Voltage Hysteresis Protection

Once the LVI has triggered (by having $V_{DD}$ fall below $V_{TRIPF}$), the LVI will maintain a reset condition until $V_{DD}$ rises above the rising trip point voltage, $V_{TRIPR}$. This prevents a condition in which the MCU is continually entering and exiting reset if $V_{DD}$ is approximately equal to $V_{TRIPF}$. $V_{TRIPR}$ is greater than $V_{TRIPF}$ by the hysteresis voltage, $V_{HYS}$.

### 11.3.4 LVI Trip Selection

The LVI5OR3 bit in the configuration register selects whether the LVI is configured for 5-V or 3-V protection.

***NOTE***

*The microcontroller is guaranteed to operate at a minimum supply voltage. The trip point ($V_{TRIPF}$ [5 V] or $V_{TRIPF}$ [3 V]) may be lower than this. See Chapter 21 Electrical Specifications for the actual trip point voltages.*

## 11.4 LVI Status Register

The LVI status register (LVISR) indicates if the $V_{DD}$ voltage was detected below the $V_{TRIPF}$ level.

Address: $FE0C

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | LVIOUT | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 11-3. LVI Status Register (LVISR)**

**LVIOUT — LVI Output Bit**

This read-only flag becomes set when the $V_{DD}$ voltage falls below the $V_{TRIPF}$ trip voltage (see Table 11-1). Reset clears the LVIOUT bit.

**Table 11-1. LVIOUT Bit Indication**

| $V_{DD}$ | LVIOUT |
|---|---|
| $V_{DD} > V_{TRIPR}$ | 0 |
| $V_{DD} < V_{TRIPF}$ | 1 |
| $V_{TRIPF} < V_{DD} < V_{TRIPR}$ | Previous value |

## 11.5 LVI Interrupts

The LVI module does not generate interrupt requests.

## 11.6 Low-Power Modes

The STOP and WAIT instructions put the MCU in low power-consumption standby modes.

### 11.6.1 Wait Mode

If enabled, the LVI module remains active in wait mode. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of wait mode.

### 11.6.2 Stop Mode

If enabled in stop mode (LVISTOP bit in the configuration register is set), the LVI module remains active in stop mode. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of stop mode.

# Chapter 12
# MSCAN08 Controller (MSCAN08)

## 12.1 Introduction

The MSCAN08 is the specific implementation of the scalable controller area network (MSCAN) concept targeted for the M68HC08 Microcontroller Family.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September, 1991.

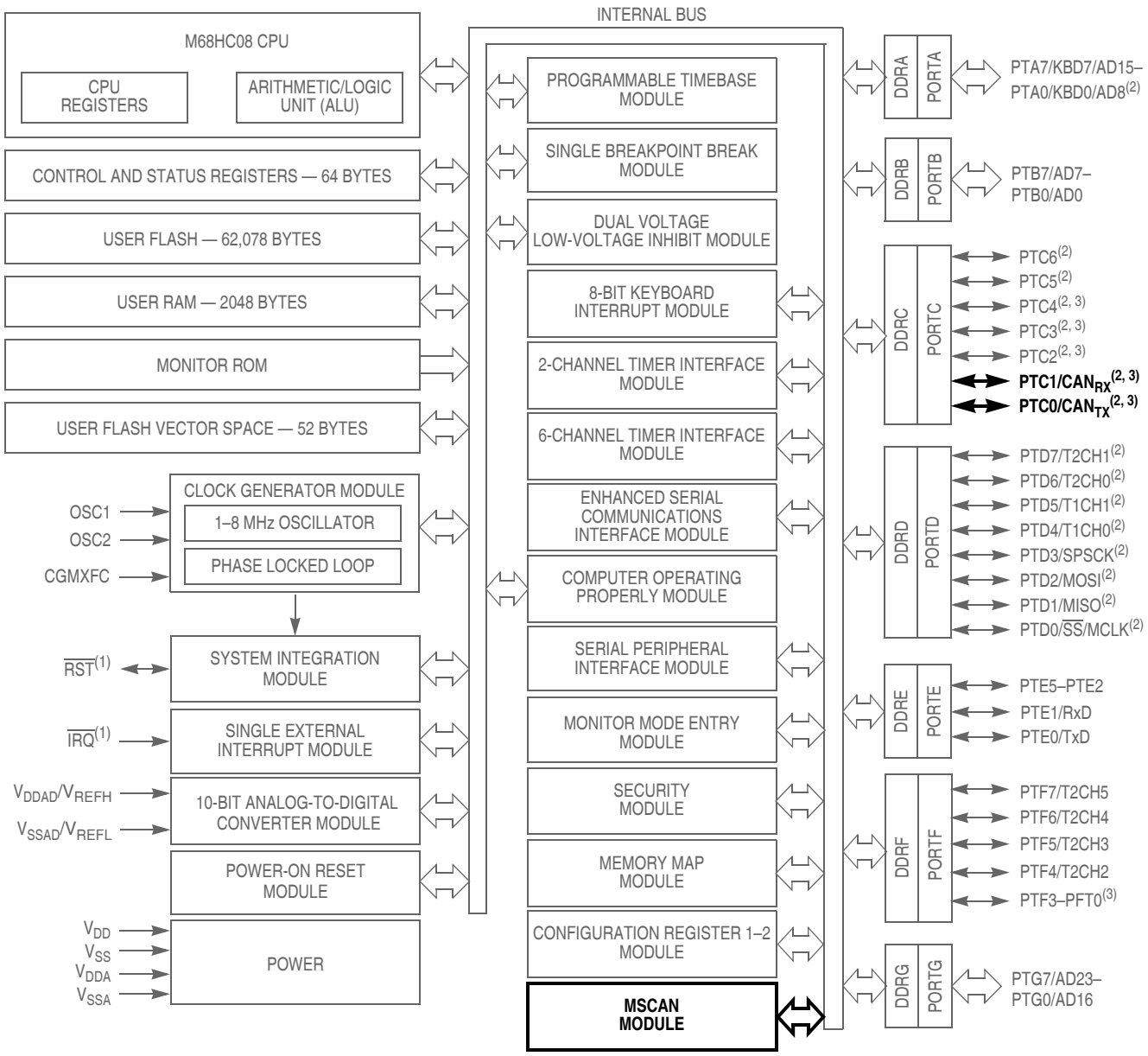
The CAN protocol was primarily, but not exclusively, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the electromagnetic interference (EMI) environment of a vehicle, cost-effectiveness, and required bandwidth.

MSCAN08 utilizes an advanced buffer arrangement, resulting in a predictable real-time behavior, and simplifies the application software.

## 12.2 Features

Basic features of the MSCAN08 are:

- MSCAN08 enable is software controlled by bit (MSCANEN) in configuration register (CONFIG2)
- Modular architecture
- Implementation of the CAN Protocol — Version 2.0A/B
  - Standard and extended data frames
  - 0–8 bytes data length.
  - Programmable bit rate up to 1 Mbps depending on the actual bit timing and the clock jitter of the phase-locked loop (PLL)
- Support for remote frames
- Double-buffered receive storage scheme
- Triple-buffered transmit storage scheme with internal prioritization using a “local priority” concept
- Flexible maskable identifier filter supports alternatively one full size extended identifier filter or two 16-bit filters or four 8-bit filters
- Programmable wakeup functionality with integrated low-pass filter
- Programmable loop-back mode supports self-test operation
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus off)
- Programmable MSCAN08 clock source either CPU bus clock or crystal oscillator output
- Programmable link to timer interface module 1 channel 0 for time-stamping and network synchronization
- Low-power sleep mode




1. Pin contains integrated pullup device.
**2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.**
**3. Higher current drive port pins**

**Figure 12-1. Block Diagram Highlighting MSCAN08 Block and Pins**

## 12.3 External Pins

The MSCAN08 uses two external pins, one input ($CAN_{RX}$) and one output ($CAN_{TX}$). The $CAN_{TX}$ output pin represents the logic level on the CAN: 0 is for a dominant state, and 1 is for a recessive state.

A typical CAN system with MSCAN08 is shown in Figure 12-2.

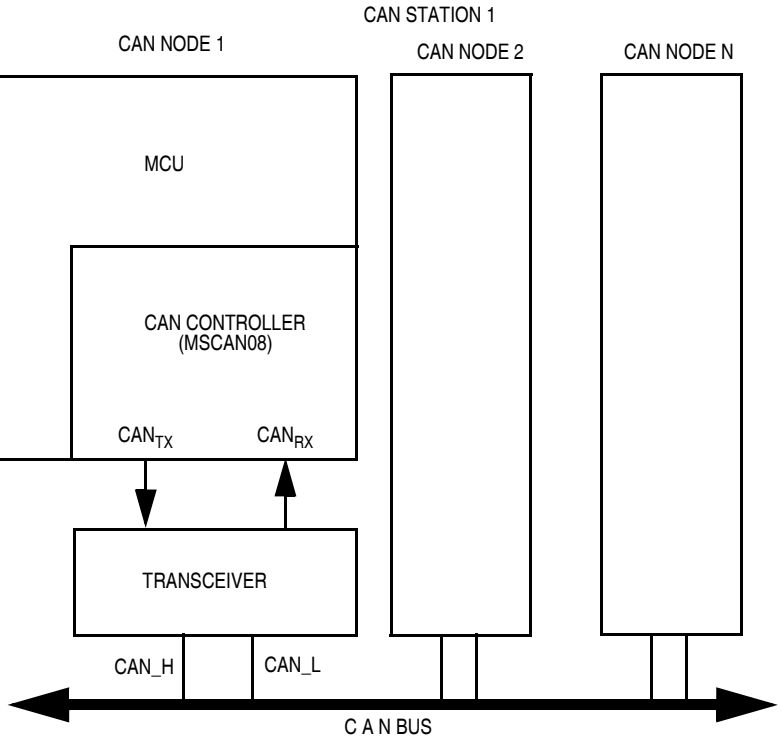



**Figure 12-2. The CAN System**

Each CAN station is connected physically to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN and has current protection against defected CAN or defected stations.

## 12.4 Message Storage

MSCAN08 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

### 12.4.1 Background

Modern application layer software is built under two fundamental assumptions:

1. Any CAN node is able to send out a stream of scheduled messages without releasing the bus between two messages. Such nodes will arbitrate for the bus right after sending the previous message and will only release the bus in case of lost arbitration.
2. The internal message queue within any CAN node is organized as such that the highest priority message will be sent out first if more than one message is ready to be sent.

Above behavior cannot be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message has been sent. This loading process lasts a definite amount of time and has to be

completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme would de-couple the re-loading of the transmit buffers from the actual message being sent and as such reduces the reactiveness requirements on the CPU. Problems may arise if the sending of a message would be finished just while the CPU re-loads the second buffer. In that case, no buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN08 has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN08 implements with the “local priority” concept described in 12.4.2 Receive Structures.

### 12.4.2 Receive Structures

The received messages are stored in a 2-stage input first in first out (FIFO). The two message buffers are mapped using a "ping pong" arrangement into a single memory area (see Figure 12-3). While the background receive buffer (RxBG) is exclusively associated to the MSCAN08, the foreground receive buffer (RxFG) is addressable by the central processor unit (CPU08). This scheme simplifies the handler software, because only one address area is applicable for the receive process.

Both buffers have a size of 13 bytes to store the CAN control bits, the identifier (standard or extended), and the data content. For details, see 12.12 Programmer’s Model of Message Storage.

The receiver full flag (RXF) in the MSCAN08 receiver flag register (CRFLG), signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier, this flag is set. See 12.13.5 MSCAN08 Receiver Flag Register (CRFLG)

On reception, each message is checked to see if it passes the filter (for details see 12.5 Identifier Acceptance Filter) and in parallel is written into RxBG. The MSCAN08 copies the content of RxBG into RxFG[1], sets the RXF flag, and generates a receive interrupt to the CPU[2]. The user’s receive handler has to read the received message from RxFG and to reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message which can follow immediately after the IFS field of the CAN frame, is received into RxBG. The overwriting of the background buffer is independent of the identifier filter function.

When the MSCAN08 module is transmitting, the MSCAN08 receives its own messages into the background receive buffer, RxBG. It does NOT overwrite RxFG, generate a receive interrupt or acknowledge its own messages on the CAN bus. The exception to this rule is in loop-back mode (see 12.13.2 MSCAN08 Module Control Register 1), where the MSCAN08 treats its own messages exactly like all other incoming messages. The MSCAN08 receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN08 must be prepared to become the receiver.

An overrun condition occurs when both the foreground and the background receive message buffers are filled with correctly received messages with accepted identifiers and another message is correctly received from the bus with an accepted identifier. The latter message will be discarded and an error interrupt with overrun indication will be generated if enabled. The MSCAN08 is still able to transmit messages with both receive message buffers filled, but all incoming messages are discarded.

1. Only if the RXF flag is not set.
2. The receive interrupt will occur only if not masked. A polling scheme can be applied on RXF also.

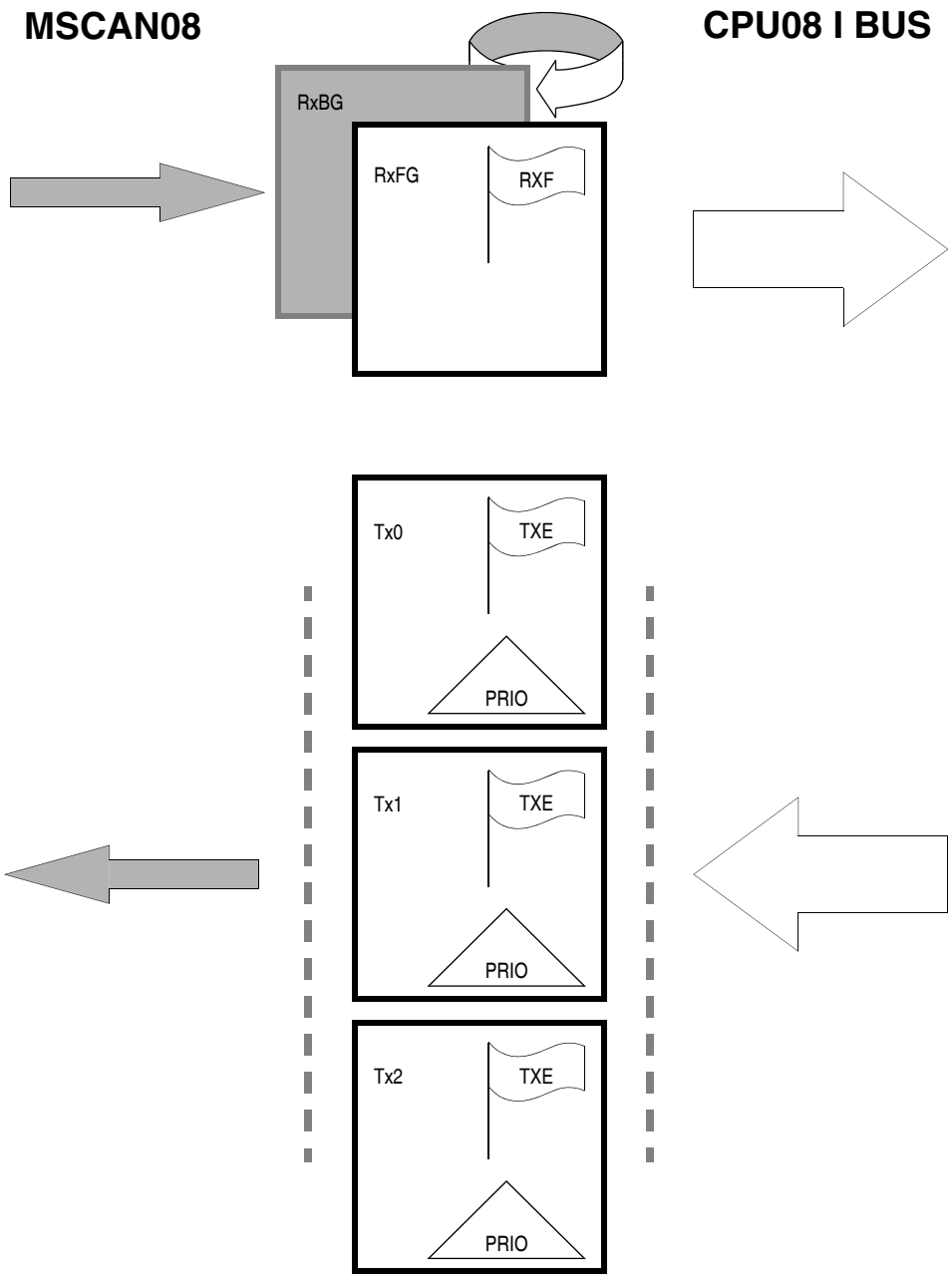



**Figure 12-3. User Model for Message Buffer Organization**

### 12.4.3 Transmit Structures

The MSCAN08 has a triple transmit buffer scheme to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in Figure 12-3.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see 12.12 Programmer's Model of Message Storage). An additional transmit buffer priority register (TBPR) contains an 8-bit "local priority" field (PRIO) (see 12.12.5 Transmit Buffer Priority Registers).

To transmit a message, the CPU08 has to identify an available transmit buffer which is indicated by a set transmit buffer empty (TXE) flag in the MSCAN08 transmitter flag register (CTFLG) (see 12.13.7 MSCAN08 Transmitter Flag Register).

The CPU08 then stores the identifier, the control bits and the data content into one of the transmit buffers. Finally, the buffer has to be flagged ready for transmission by clearing the TXE flag.

The MSCAN08 then will schedule the message for transmission and will signal the successful transmission of the buffer by setting the TXE flag. A transmit interrupt is generated[1] when TXE is set and can be used to drive the application software to re-load the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN08 uses the local priority setting of the three buffers for prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software sets this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being emitted from this node. The lowest binary value of the PRIO field is defined as the highest priority.

The internal scheduling process takes place whenever the MSCAN08 arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message being set up in one of the three transmit buffers. As messages that are already under transmission cannot be aborted, the user has to request the abort by setting the corresponding abort request flag (ABTRQ) in the transmission control register (CTCR). The MSCAN08 will then grant the request, if possible, by setting the corresponding abort request acknowledge (ABTAK) and the TXE flag in order to release the buffer and by generating a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was actually aborted (ABTAK = 1) or sent (ABTAK = 0).

## 12.5 Identifier Acceptance Filter

The identifier acceptance registers (CIDAR0–CIDAR3) define the acceptance patterns of the standard or extended identifier (ID10–ID0 or ID28–ID0). Any of these bits can be marked ‘don’t care’ in the identifier mask registers (CIDMR0–CIDMR3).

A filter hit is indicated to the application on software by a set RXF (receive buffer full flag, see 12.13.5 MSCAN08 Receiver Flag Register (CRFLG)) and two bits in the identifier acceptance control register (see 12.13.9 MSCAN08 Identifier Acceptance Control Register). These identifier hit flags (IDHIT1 and IDHIT0) clearly identify the filter section that caused the acceptance. They simplify the application software’s task to identify the cause of the receiver interrupt. In case that more than one hit occurs (two or more filters match) the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

1. Single identifier acceptance filter, each to be applied to a) the full 29 bits of the extended identifier and to the following bits of the CAN frame: RTR, IDE, SRR or b) the 11 bits of the standard identifier plus the RTR and IDE bits of CAN 2.0A/B messages. This mode implements a single filter for a full length CAN 2.0B compliant extended identifier. Figure 12-4 shows how the 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces a filter 0 hit.

1. The transmit interrupt will occur only if not masked. A polling scheme can be applied on TXE also.

2. Two identifier acceptance filters, each to be applied to:
   a. The 14 most significant bits of the extended identifier plus the SRR and the IDE bits of CAN2.0B messages, or
   b. The 11 bits of the identifier plus the RTR and IDE bits of CAN 2.0A/B messages.

   Figure 12-5 shows how the 32-bit filter bank (CIDAR0–CIDAR3 and CIDMR0–CIDMR3) produces filter 0 and 1 hits.
3. Four identifier acceptance filters, each to be applied to the first eight bits of the identifier. This mode implements four independent filters for the first eight bits of a CAN 2.0A/B compliant standard identifier. Figure 12-6 shows how the 32-bit filter bank (CIDAR0–CIDAR3 and CIDMR0–CIDMR3) produces filter 0 to 3 hits.
4. Closed filter. No CAN message will be copied into the foreground buffer RxFG, and the RXF flag will never be set.

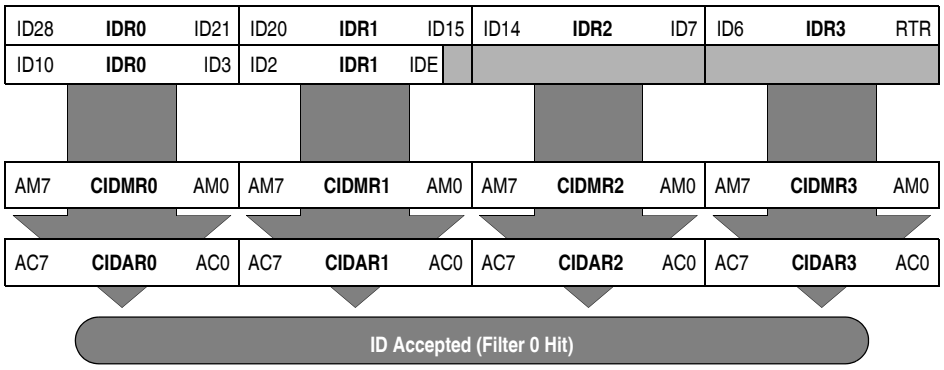



**Figure 12-4. Single 32-Bit Maskable Identifier Acceptance Filter**

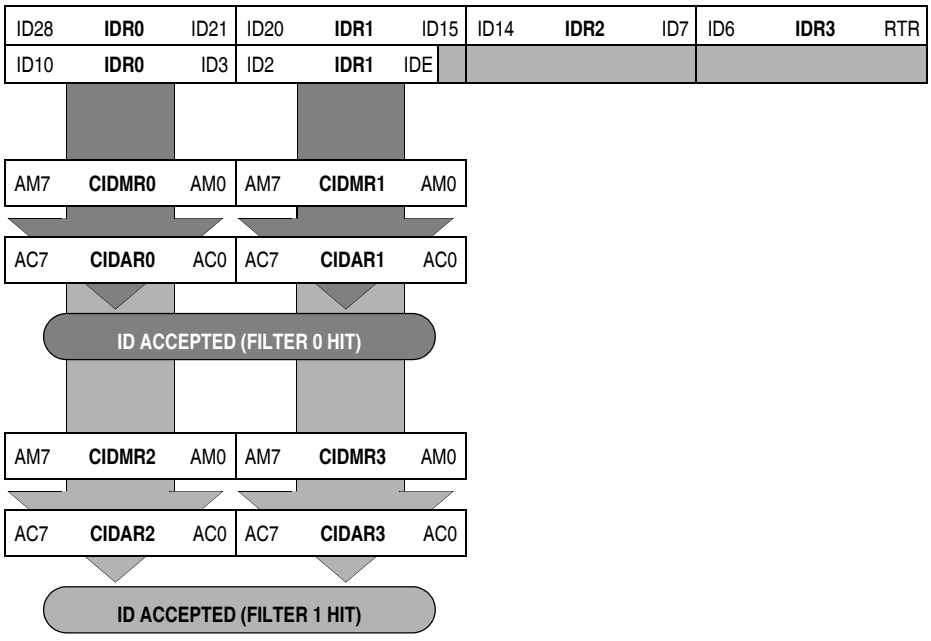



**Figure 12-5. Dual 16-Bit Maskable Acceptance Filters**

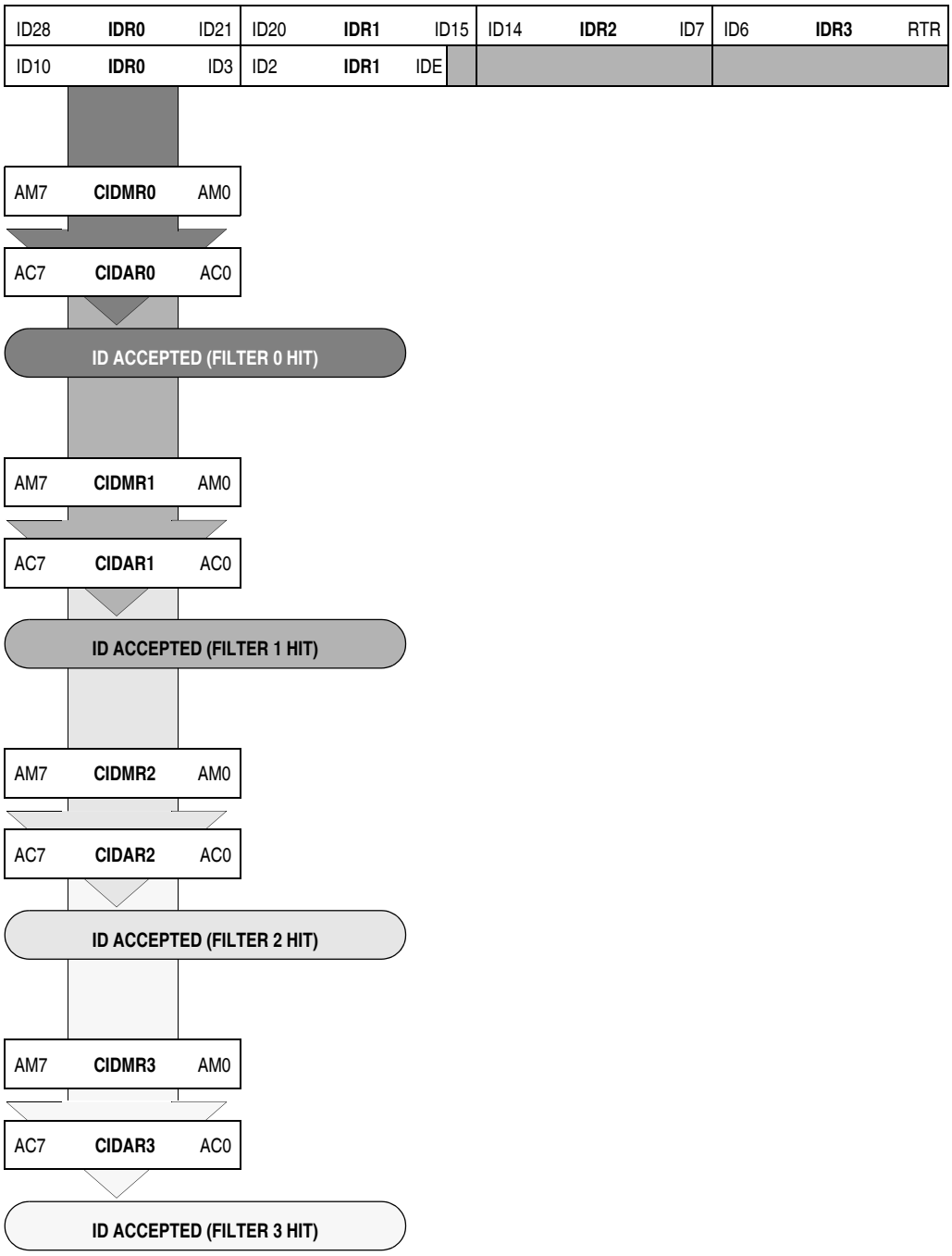



**Figure 12-6. Quadruple 8-Bit Maskable Acceptance Filters**

## 12.6 Interrupts

The MSCAN08 supports four interrupt vectors mapped onto eleven different interrupt sources, any of which can be individually masked. For details, see 12.13.5 MSCAN08 Receiver Flag Register (CRFLG) through 12.13.8 MSCAN08 Transmitter Control Register.

1. *Transmit Interrupt*: At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXE flags of the empty message buffers are set.
2. *Receive Interrupt*: A message has been received successfully and loaded into the foreground receive buffer. This interrupt will be emitted immediately after receiving the EOF symbol. The RXF flag is set.
3. *Wakeup Interrupt*: An activity on the CAN bus occurred during MSCAN08 internal sleep mode or power-down mode (provided SLPAK = WUPIE = 1).
4. *Error Interrupt*: An overrun, error, or warning condition occurred. The receiver flag register (CRFLG) will indicate one of the following conditions:
   - *Overrun:* An overrun condition as described in 12.4.2 Receive Structures, has occurred.
   - *Receiver Warning*: The receive error counter has reached the CPU warning limit of 96.
   - *Transmitter Warning*: The transmit error counter has reached the CPU warning limit of 96.
   - *Receiver Error Passive*: The receive error counter has exceeded the error passive limit of 127 and MSCAN08 has gone to error passive state.
   - *Transmitter Error Passive*: The transmit error counter has exceeded the error passive limit of 127 and MSCAN08 has gone to error passive state.
   - *Bus Off*: The transmit error counter has exceeded 255 and MSCAN08 has gone to bus off state.

### 12.6.1 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN08 receiver flag register (CRFLG) or the MSCAN08 transmitter flag register (CTFLG). Interrupts are pending as long as one of the corresponding flags is set. The flags in the above registers must be reset within the interrupt handler in order to handshake the interrupt. The flags are reset through writing a ‘1’ to the corresponding bit position. A flag cannot be cleared if the respective condition still prevails.

***NOTE***

*Bit manipulation instructions (BSET) shall not be used to clear interrupt flags.*

### 12.6.2 Interrupt Vectors

The MSCAN08 supports four interrupt vectors as shown in Table 12-1. The vector addresses and the relative interrupt priority are dependent on the chip integration and to be defined.

**Table 12-1. MSCAN08 Interrupt Vector Addresses**

| Function | Source | Local Mask | Global Mask |
|---|---|---|---|
| Wakeup | WUPIF | WUPIE | I bit |
| Error interrupts | RWRNIF | RWRNIE | |
| | TWRNIF | TWRNIE | |
| | RERRIF | RERRIE | |
| | TERRIF | TERRIE | |
| | BOFFIF | BOFFIE | |
| | OVRIF | OVRIE | |
| Receive | RXF | RXFIE | |
| Transmit | TXE0 | TXEIE0 | |
| | TXE1 | TXEIE1 | |
| | TXE2 | TXEIE2 | |

## 12.7 Protocol Violation Protection

The MSCAN08 will protect the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN08 can not be modified while the MSCAN08 is on-line. The SFTRES bit in the MSCAN08 module control register (see 12.13.1 MSCAN08 Module Control Register 0) serves as a lock to protect the following registers:
  - MSCAN08 module control register 1 (CMCR1)
  - MSCAN08 bus timing register 0 and 1 (CBTR0 and CBTR1)
  - MSCAN08 identifier acceptance control register (CIDAC)
  - MSCAN08 identifier acceptance registers (CIDAR0–3)
  - MSCAN08 identifier mask registers (CIDMR0–3)
- The $CAN_{TX}$ pin is forced to recessive when the MSCAN08 is in any of the low-power modes.

## 12.8 Low-Power Modes

In addition to normal mode, the MSCAN08 has three modes with reduced power consumption: sleep, soft reset, and power down. In sleep and soft reset mode, power consumption is reduced by stopping all clocks except those to access the registers. In power-down mode, all clocks are stopped and no power is consumed.

The WAIT and STOP instructions put the MCU in low-power consumption stand-by modes. Table 12-2 summarizes the combinations of MSCAN08 and CPU modes. A particular combination of modes is entered for the given settings of the bits SLPAK and SFTRES. For all modes, an MSCAN08 wakeup interrupt can occur only if SLPAK = WUPIE = 1.

**Table 12-2. MSCAN08 versus CPU Operating Modes**

| MSCAN08 Mode | CPU Mode | |
|---|---|---|
| | STOP | WAIT or RUN |
| Power Down | SLPAK = X(1)<br>SFTRES = X | |
| Sleep | | SLPAK = 1<br>SFTRES = 0 |
| Soft Reset | | SLPAK = 0<br>SFTRES = 1 |
| Normal | | SLPAK = 0<br>SFTRES = 0 |

1. 'X' means don't care.

### 12.8.1 MSCAN08 Sleep Mode

The CPU can request the MSCAN08 to enter the low-power mode by asserting the SLPRQ bit in the module configuration register (see Figure 12-7). The time when the MSCAN08 enters sleep mode depends on its activity:

- If it is transmitting, it continues to transmit until there is no more message to be transmitted, and then goes into sleep mode
- If it is receiving, it waits for the end of this message and then goes into sleep mode
- If it is neither transmitting or receiving, it will immediately go into sleep mode

***NOTE***

*The application software must avoid setting up a transmission (by clearing or more TXE flags) and immediately request sleep mode (by setting SLPRQ). It then depends on the exact sequence of operations whether MSCAN08 starts transmitting or goes into sleep mode directly.*

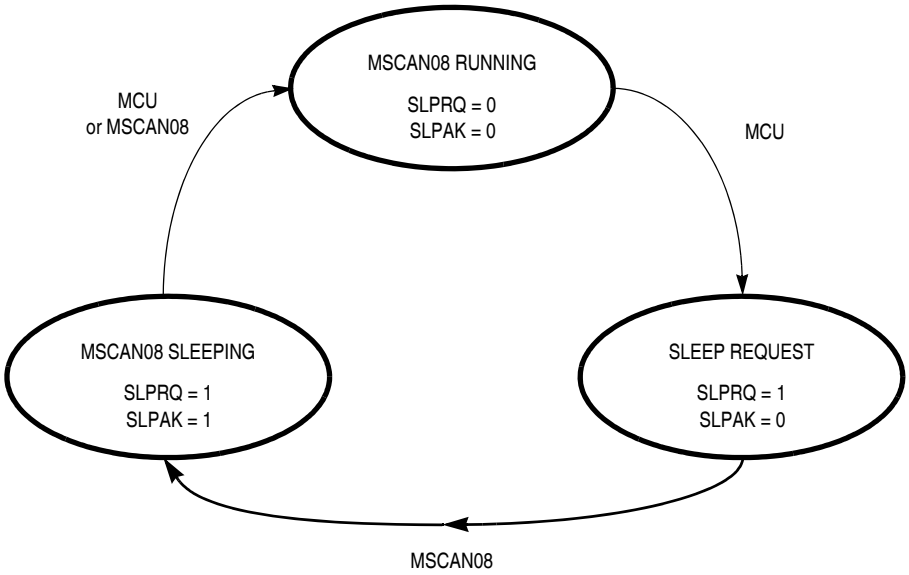



**Figure 12-7. Sleep Request/Acknowledge Cycle**

During sleep mode, the SLPAK flag is set. The application software should use SLPAK as a handshake indication for the request (SLPRQ) to go into sleep mode. When in sleep mode, the MSCAN08 stops its internal clocks. However, clocks to allow register accesses still run. If the MSCAN08 is in bus-off state, it stops counting the 128*11 consecutive recessive bits due to the stopped clocks. The $CAN_{TX}$ pin stays in recessive state. If RXF = 1, the message can be read and RXF can be cleared. Copying of RxGB into RxFG doesn't take place while in sleep mode. It is possible to access the transmit buffers and to clear the TXE flags. No message abort takes place while in sleep mode.

The MSCAN08 leaves sleep mode (wakes-up) when:

- Bus activity occurs, or
- The MCU clears the SLPRQ bit, or
- The MCU sets the SFTRES bit

***NOTE***

*The MCU cannot clear the SLPRQ bit before the MSCAN08 is in sleep mode (SLPAK=1).*

After wakeup, the MSCAN08 waits for 11 consecutive recessive bits to synchronize to the bus. As a consequence, if the MSCAN08 is woken-up by a CAN frame, this frame is not received. The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions are executed upon wakeup: copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN08 is still in bus-off state after sleep mode was left, it continues counting the 128*11 consecutive recessive bits.

### 12.8.2 MSCAN08 Soft Reset Mode

In soft reset mode, the MSCAN08 is stopped. Registers can still be accessed. This mode is used to initialize the module configuration, bit timing and the CAN message filter. See 12.13.1 MSCAN08 Module Control Register 0 for a complete description of the soft reset mode.

When setting the SFTRES bit, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations.

***NOTE***

*The user is responsible to take care that the MSCAN08 is not active when soft reset mode is entered. The recommended procedure is to bring the MSCAN08 into sleep mode before the SFTRES bit is set.*

### 12.8.3 MSCAN08 Power-Down Mode

The MSCAN08 is in power-down mode when the CPU is in stop mode.

When entering the power-down mode, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations.

***NOTE***

*The user is responsible to take care that the MSCAN08 is not active when power-down mode is entered. The recommended procedure is to bring the MSCAN08 into sleep mode before the STOP instruction is executed.*

To protect the CAN bus system from fatal consequences resulting from violations of the above rule, the MSCAN08 drives the $CAN_{TX}$ pin into recessive state.

In power-down mode, no registers can be accessed.

MSCAN08 bus activity can wake the MCU from CPU stop/MSCAN08 power-down mode. However, until the oscillator starts up and synchronization is achieved the MSCAN08 will not respond to incoming data.

### 12.8.4 CPU Wait Mode

The MSCAN08 module remains active during CPU wait mode. The MSCAN08 will stay synchronized to the CAN bus and generates transmit, receive, and error interrupts to the CPU, if enabled. Any such interrupt will bring the MCU out of wait mode.

### 12.8.5 Programmable Wakeup Function

The MSCAN08 can be programmed to apply a low-pass filter function to the $CAN_{RX}$ input line while in internal sleep mode (see information on control bit WUPM in 12.13.2 MSCAN08 Module Control Register 1). This feature can be used to protect the MSCAN08 from wakeup due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic inference within noisy environments.

## 12.9 Timer Link

The MSCAN08 will generate a timer signal whenever a valid frame has been received. Because the CAN specification defines a frame to be valid if no errors occurred before the EOF field has been transmitted successfully, the timer signal will be generated right after the EOF. A pulse of one bit time is generated. As the MSCAN08 receiver engine also receives the frames being sent by itself, a timer signal also will be generated after a successful transmission.

The previously described timer signal can be routed into the on-chip timer interface module (TIM). This signal is connected to channel 0 of timer interface module 1 (TIM1) under the control of the timer link enable (TLNKEN) bit in CMCR0.

After timer n has been programmed to capture rising edge events, it can be used under software control to generate 16-bit time stamps which can be stored with the received message.

## 12.10 Clock System

Figure 12-8 shows the structure of the MSCAN08 clock generation circuitry and its interaction with the clock generation module (CGM). With this flexible clocking scheme the MSCAN08 is able to handle CAN bus rates ranging from 10 kbps up to 1 Mbps.

The clock source bit (CLKSRC) in the MSCAN08 module control register (CMCR1) (see 12.13.1 MSCAN08 Module Control Register 0) defines whether the MSCAN08 is connected to the output of the crystal oscillator or to the PLL output.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met.

***NOTE***

*If the system clock is generated from a PLL, it is recommended to select the crystal clock source rather than the system clock source due to jitter considerations, especially at faster CAN bus rates.*

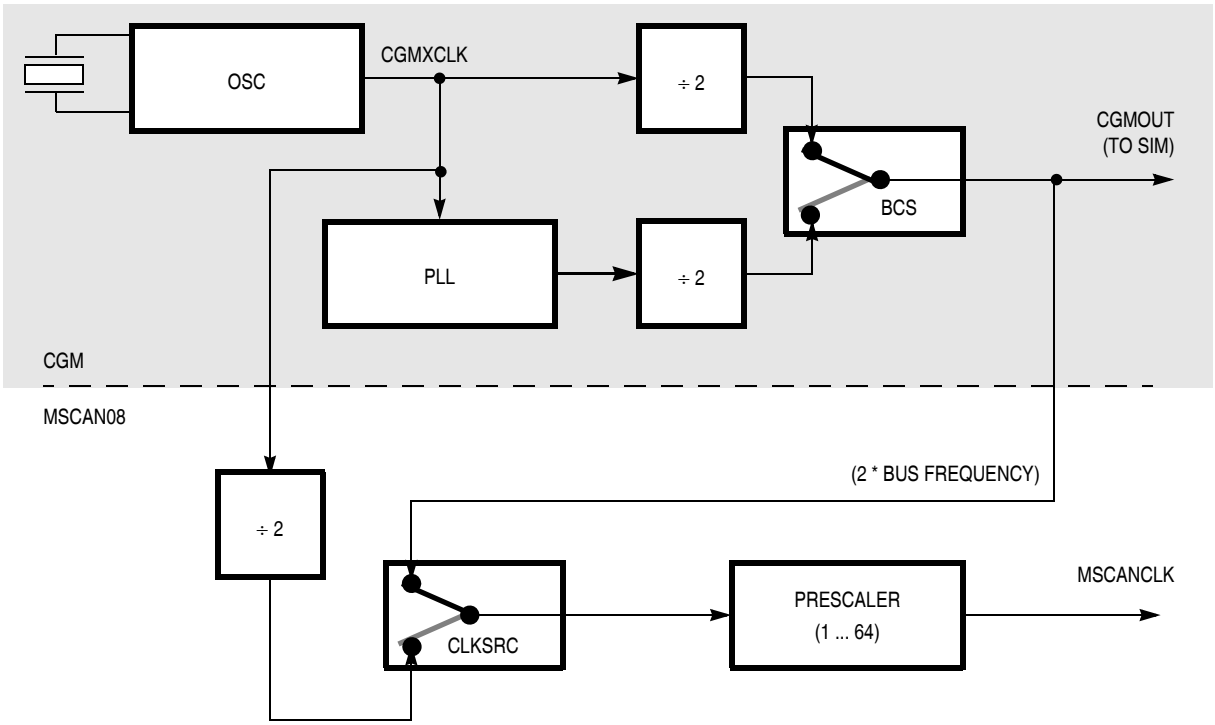



**Figure 12-8. Clocking Scheme**

A programmable prescaler is used to generate out of the MSCAN08 clock the time quanta (Tq) clock. A time quantum is the atomic unit of time handled by the MSCAN08.

$$f_{Tq} = \frac{f_{MSCANCLK}}{\text{Presc value}}$$

A bit time is subdivided into three segments[1] (see Figure 12-9):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time segment 2: This segment represents PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

$$\text{Bit rate} = \frac{f_{Tq}}{\text{No. of time quanta}}$$

The synchronization jump width (SJW) can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

1. For further explanation of the underlying concepts please refer to ISO/DIS 11 519-1, Section 10.3.

The above parameters can be set by programming the bus timing registers, CBTR0 and CBTR1. See 12.13.3 MSCAN08 Bus Timing Register 0 and 12.13.4 MSCAN08 Bus Timing Register 1.

***NOTE***

*It is the user's responsibility to make sure that the bit timing settings are in compliance with the CAN standard,*

Table 12-8 gives an overview on the CAN conforming segment settings and the related parameter values.

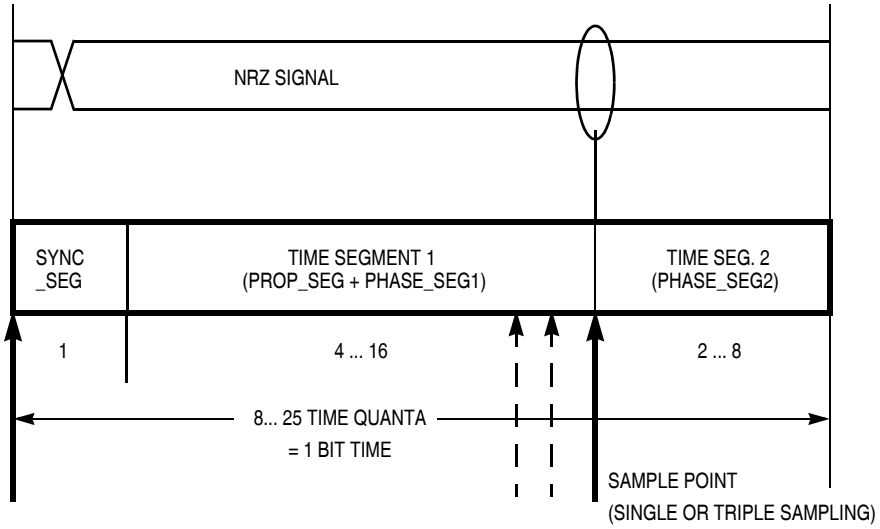



**Figure 12-9. Segments Within the Bit Time**

**Table 12-3. Time Segment Syntax**

| | |
|---|---|
| SYNC_SEG | System expects transitions to occur on the bus during this period. |
| Transmit point | A node in transmit mode will transfer a new value to the CAN bus at this point. |
| Sample point | A node in receive mode will sample the bus at this point. If the three samples per bit option is selected then this point marks the position of the third sample. |

**Table 12-4. CAN Standard Compliant Bit Time Segment Settings**

| Time Segment 1 | TSEG1 | Time Segment 2 | TSEG2 | Synchronized Jump Width | SJW |
|---|---|---|---|---|---|
| 5 .. 10 | 4 .. 9 | 2 | 1 | 1 .. 2 | 0 .. 1 |
| 4 .. 11 | 3 .. 10 | 3 | 2 | 1 .. 3 | 0 .. 2 |
| 5 .. 12 | 4 .. 11 | 4 | 3 | 1 .. 4 | 0 .. 3 |
| 6 .. 13 | 5 .. 12 | 5 | 4 | 1 .. 4 | 0 .. 3 |
| 7 .. 14 | 6 .. 13 | 6 | 5 | 1 .. 4 | 0 .. 3 |
| 8 .. 15 | 7 .. 14 | 7 | 6 | 1 .. 4 | 0 .. 3 |
| 9 .. 16 | 8 .. 15 | 8 | 7 | 1 .. 4 | 0 .. 3 |

## 12.11 Memory Map

The MSCAN08 occupies 128 bytes in the CPU08 memory space. The absolute mapping is implementation dependent with the base address being a multiple of 128.

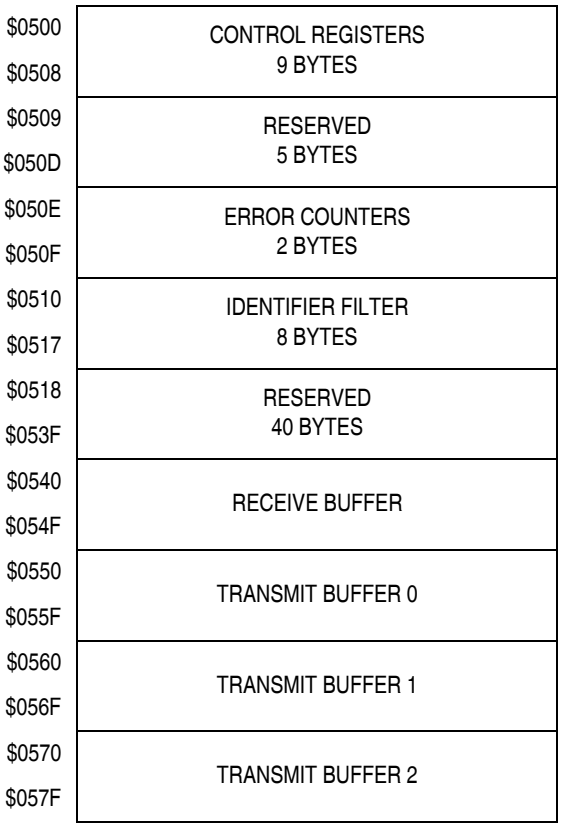



**Figure 12-10. MSCAN08 Memory Map**

## 12.12 Programmer's Model of Message Storage

This section details the organization of the receive and transmit message buffers and the associated control registers. For reasons of programmer interface simplification, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13-byte data structure. An additional transmit buffer priority register (TBPR) is defined for the transmit buffers.

| Addr[1] | Register Name |
|---|---|
| $05b0 | IDENTIFIER REGISTER 0 |
| $05b1 | IDENTIFIER REGISTER 1 |
| $05b2 | IDENTIFIER REGISTER 2 |
| $05b3 | IDENTIFIER REGISTER 3 |
| $05b4 | DATA SEGMENT REGISTER 0 |
| $05b5 | DATA SEGMENT REGISTER 1 |
| $05b6 | DATA SEGMENT REGISTER 2 |
| $05b7 | DATA SEGMENT REGISTER 3 |
| $05b8 | DATA SEGMENT REGISTER 4 |
| $05b9 | DATA SEGMENT REGISTER 5 |
| $05bA | DATA SEGMENT REGISTER 6 |
| $05bB | DATA SEGMENT REGISTER 7 |
| $05bC | DATA LENGTH REGISTER |
| $05bD | TRANSMIT BUFFER PRIORITY REGISTER[2] |
| $05bE | UNUSED |
| $05bF | UNUSED |

1. Where b equals the following:
   b = 4 for receive buffer
   b = 5 for transmit buffer 0
   b = 6 for transmit buffer 1
   b = 7 for transmit buffer 2
2. Not applicable for receive buffers

**Figure 12-11. Message Buffer Organization**

### 12.12.1 Message Buffer Outline

Figure 12-12 shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure 12-13. All bits of the 13-byte data structure are undefined out of reset.

***NOTE***

*The foreground receive buffer can be read anytime but cannot be written.*
*The transmit buffers can be read or written anytime.*

| Addr. | Register | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $05b0 | IDR0 | Read:<br>Write: | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| $05b1 | IDR1 | Read:<br>Write: | ID20 | ID19 | ID18 | SRR (=1) | IDE (=1) | ID17 | ID16 | ID15 |
| $05b2 | IDR2 | Read:<br>Write: | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| $05b3 | IDR3 | Read:<br>Write: | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |
| $05b4 | DSR0 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05b5 | DSR1 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05b6 | DSR2 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05b7 | DSR3 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05b8 | DSR4 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05b9 | DSR5 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05bA | DSR6 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05bB | DSR7 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05bC | DLR | Read:<br>Write: | | | | | DLC3 | DLC2 | DLC1 | DLC0 |

= Unimplemented

**Figure 12-12. Receive/Transmit Message Buffer Extended Identifier (IDRn)**

| Addr. | Register | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $05b0 | IDR0 | Read:<br>Write: | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 |
| $05b1 | IDR1 | Read:<br>Write: | ID2 | ID1 | ID0 | RTR | IDE (=0) | | | |
| $05b2 | IDR2 | Read:<br>Write: | | | | | | | | |
| $05b3 | IDR3 | Read:<br>Write: | | | | | | | | |

= Unimplemented

**Figure 12-13. Standard Identifier Mapping**

## 12.12.2 Identifier Registers

The identifiers consist of either 11 bits (ID10–ID0) for the standard, or 29 bits (ID28–ID0) for the extended format. ID10/28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

**SRR — Substitute Remote Request**

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and will be stored as received on the CAN bus for receive buffers.

**IDE — ID Extended**

This flag indicates whether the extended or standard identifier format is applied in this buffer. In case of a receive buffer, the flag is set as being received and indicates to the CPU how to process the buffer identifier registers. In case of a transmit buffer, the flag indicates to the MSCAN08 what type of identifier to send.

1 = Extended format, 29 bits
0 = Standard format, 11 bits

**RTR — Remote Transmission Request**

This flag reflects the status of the remote transmission request bit in the CAN frame. In case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.

1 = Remote frame
0 = Data frame

### 12.12.3 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

**DLC3–DLC0 — Data Length Code Bits**

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table 12-5 shows the effect of setting the DLC bits.

**Table 12-5. Data Length Codes**

| Data Length Code | | | | Data Byte Count |
|---|---|---|---|---|
| DLC3 | DLC2 | DLC1 | DLC0 | |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |

### 12.12.4 Data Segment Registers (DSRn)

The eight data segment registers contain the data to be transmitted or received. The number of bytes to be transmitted or being received is determined by the data length code in the corresponding DLR.

### 12.12.5 Transmit Buffer Priority Registers

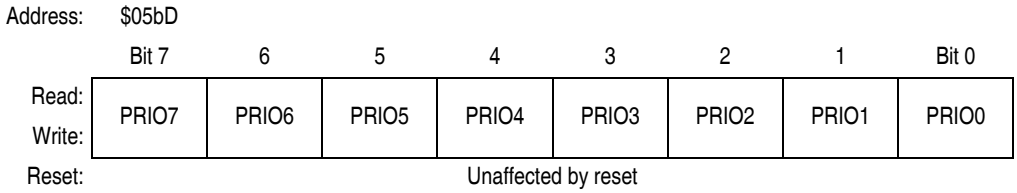



**Figure 12-14. Transmit Buffer Priority Register (TBPR)**

**PRIO7–PRIO0 — Local Priority**

This field defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the MSCAN08 and is defined to be highest for the smallest binary number. The MSCAN08 implements the following internal prioritization mechanism:

- All transmission buffers with a cleared TXE flag participate in the prioritization right before the SOF is sent.
- The transmission buffer with the lowest local priority field wins the prioritization.
- In case more than one buffer has the same lowest priority, the message buffer with the lower index number wins.

## 12.13 Programmer's Model of Control Registers

The programmer's model has been laid out for maximum simplicity and efficiency. Figure 12-15 gives an overview on the control register block of the MSCAN08.

| Addr. | Register | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0500 | CMCR0 | Read: | 0 | 0 | 0 | SYNCH | TLNKEN | SLPAK | SLPRQ | SFTRES |
| | | Write: | | | | | | | | |
| $0501 | CMCR1 | Read: | 0 | 0 | 0 | 0 | 0 | LOOPB | WUPM | CLKSRC |
| | | Write: | | | | | | | | |
| $0502 | CBTR0 | Read: | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| | | Write: | | | | | | | | |
| $0503 | CBTR1 | Read: | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| | | Write: | | | | | | | | |
| $0504 | CRFLG | Read: | WUPIF | RWRNIF | TWRNIF | RERRIF | TERRIF | BOFFIF | OVRIF | RXF |
| | | Write: | | | | | | | | |
| $0505 | CRIER | Read: | WUPIE | RWRNIE | TWRNIE | RERRIE | TERRIE | BOFFIE | OVRIE | RXFIE |
| | | Write: | | | | | | | | |
| $0506 | CTFLG | Read: | 0 | ABTAK2 | ABTAK1 | ABTAK0 | 0 | TXE2 | TXE1 | TXE0 |
| | | Write: | | | | | | | | |
| $0507 | CTCR | Read: | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| | | Write: | | | | | | | | |
| $0508 | CIDAC | Read: | 0 | IDAM2 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| | | Write: | | | | | | | | |
| $0509 | Reserved | Read: | R | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| $050E | CRXERR | Read: | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| | | Write: | | | | | | | | |
| $050F | CTXERR | Read: | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| | | Write: | | | | | | | | |
| $0510 | CIDAR0 | Read: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | Write: | | | | | | | | |
| $0511 | CIDAR1 | Read: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | Write: | | | | | | | | |
| $0512 | CIDAR2 | Read: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | Write: | | | | | | | | |
| $0513 | CIDAR3 | Read: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | Write: | | | | | | | | |

= Unimplemented  R = Reserved

**Figure 12-15. MSCAN08 Control Register Structure**

| Addr. | Register | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0514 | CIDMR0 | Read:<br>Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| $0515 | CIDMR1 | Read:<br>Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| $0516 | CIDMR2 | Read:<br>Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| $0517 | CIDMR3 | Read:<br>Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |

= Unimplemented R = Reserved

**Figure 12-15. MSCAN08 Control Register Structure (Continued)**

## 12.13.1 MSCAN08 Module Control Register 0

Address: $0500

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | SYNCH | TLNKEN | SLPAK | SLPRQ | SFTRES |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

= Unimplemented

**Figure 12-16. Module Control Register 0 (CMCR0)**

**SYNCH — Synchronized Status**

This bit indicates whether the MSCAN08 is synchronized to the CAN bus and as such can participate in the communication process.

1 = MSCAN08 synchronized to the CAN bus
0 = MSCAN08 not synchronized to the CAN bus

**TLNKEN — Timer Enable**

This flag is used to establish a link between the MSCAN08 and the on-chip timer (see 12.9 Timer Link).

1 = The MSCAN08 timer signal output is connected to the timer input.
0 = The port is connected to the timer input.

**SLPAK — Sleep Mode Acknowledge**

This flag indicates whether the MSCAN08 is in module internal sleep mode. It shall be used as a handshake for the sleep mode request (see 12.8.1 MSCAN08 Sleep Mode). If the MSCAN08 detects bus activity while in sleep mode, it clears the flag.

1 = Sleep – MSCAN08 in internal sleep mode
0 = Wakeup – MSCAN08 is not in sleep mode

**SLPRQ — Sleep Request, Go to Internal Sleep Mode**

This flag requests the MSCAN08 to go into an internal power-saving mode (see 12.8.1 MSCAN08 Sleep Mode).

1 = Sleep — The MSCAN08 will go into internal sleep mode.
0 = Wakeup — The MSCAN08 will function normally.

**SFTRES — Soft Reset**

When this bit is set by the CPU, the MSCAN08 immediately enters the soft reset state. Any ongoing transmission or reception is aborted and synchronization to the bus is lost.

The following registers enter and stay in their hard reset state:

CMCR0, CRFLG, CRIER, CTFLG, and CTCR.

The registers CMCR1, CBTR0, CBTR1, CIDAC, CIDAR0–CIDAR3, and CIDMR0–CIDMR3 can only be written by the CPU when the MSCAN08 is in soft reset state. The values of the error counters are not affected by soft reset.

When this bit is cleared by the CPU, the MSCAN08 tries to synchronize to the CAN bus. If the MSCAN08 is not in bus-off state, it will be synchronized after 11 recessive bits on the bus; if the MSCAN08 is in bus-off state, it continues to wait for 128 occurrences of 11 recessive bits.

Clearing SFTRES and writing to other bits in CMCR0 must be in separate instructions.

1 = MSCAN08 in soft reset state
0 = Normal operation

## 12.13.2 MSCAN08 Module Control Register 1

Address: $0501

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | 0 | LOOPB | WUPM | CLKSRC |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 12-17. Module Control Register (CMCR1)**

**LOOPB — Loop Back Self-Test Mode**

When this bit is set, the MSCAN08 performs an internal loop back which can be used for self-test operation: the bit stream output of the transmitter is fed back to the receiver internally. The $CAN_{RX}$ input pin is ignored and the $CAN_{TX}$ output goes to the recessive state (1). The MSCAN08 behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state the MSCAN08 ignores the bit sent during the ACK slot of the CAN frame Acknowledge field to insure proper reception of its own message. Both transmit and receive interrupts are generated.

1 = Activate loop back self-test mode
0 = Normal operation

**WUPM — Wakeup Mode**

This flag defines whether the integrated low-pass filter is applied to protect the MSCAN08 from spurious wakeups (see 12.8.5 Programmable Wakeup Function).

1 = MSCAN08 will wakeup the CPU only in cases of a dominant pulse on the bus which has a length of at least $t_{wup}$.
0 = MSCAN08 will wakeup the CPU after any recessive-to-dominant edge on the CAN bus.

### CLKSRC — Clock Source

This flag defines which clock source the MSCAN08 module is driven from (see 12.10 Clock System).

1 = The MSCAN08 clock source is CGMOUT (see Figure 12-8).

0 = The MSCAN08 clock source is CGMXCLK/2 (see Figure 12-8).

***NOTE***

*The CMCR1 register can be written only if the SFTRES bit in the MSCAN08 module control register is set*

## 12.13.3 MSCAN08 Bus Timing Register 0

Address: $0502

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-18. Bus Timing Register 0 (CBTR0)**

### SJW1 and SJW0 — Synchronization Jump Width

The synchronization jump width (SJW) defines the maximum number of time quanta ($T_q$) clock cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see Table 12-6).

**Table 12-6. Synchronization Jump Width**

| SJW1 | SJW0 | Synchronization Jump Width |
|---|---|---|
| 0 | 0 | 1 $T_q$ cycle |
| 0 | 1 | 2 $T_q$ cycle |
| 1 | 0 | 3 $T_q$ cycle |
| 1 | 1 | 4 $T_q$ cycle |

### BRP5–BRP0 — Baud Rate Prescaler

These bits determine the time quanta ($T_q$) clock, which is used to build up the individual bit timing, according to Table 12-7.

**Table 12-7. Baud Rate Prescaler**

| BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 | Prescaler Value (P) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| : | : | : | : | : | : | : |
| : | : | : | : | : | : | : |
| 1 | 1 | 1 | 1 | 1 | 1 | 64 |

***NOTE***

*The CBTR0 register can be written only if the SFTRES bit in the MSCAN08 module control register is set.*

### 12.13.4 MSCAN08 Bus Timing Register 1

Address: $0503

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-19. Bus Timing Register 1 (CBTR1)**

**SAMP — Sampling**

This bit determines the number of serial bus samples to be taken per bit time. If set, three samples per bit are taken, the regular one (sample point) and two preceding samples, using a majority rule. For higher bit rates, SAMP should be cleared, which means that only one sample will be taken per bit.

1 = Three samples per bit(1)
0 = One sample per bit

**TSEG22–TSEG10 — Time Segment**

Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point. Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in Table 12-8. The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta ($T_q$) clock cycles per bit as shown in Table 12-4).

$$\text{Bit time} = \frac{\text{Pres value}}{f_{MSCANCLK}} \bullet \text{number of time quanta}$$

***NOTE***

*The CBTR1 register can only be written if the SFTRES bit in the MSCAN08 module control register is set.*

**Table 12-8. Time Segment Values**

| TSEG13 | TSEG12 | TSEG11 | TSEG10 | Time Segment 1 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 $T_q$ Cycle(1) |
| 0 | 0 | 0 | 1 | 2 $T_q$ Cycles(1) |
| 0 | 0 | 1 | 0 | 3$T_q$ Cycles(1) |
| 0 | 0 | 1 | 1 | 4 $T_q$ Cycles |
| . | . | . | . | . |
| . | . | . | . | . |
| 1 | 1 | 1 | 1 | 16 $T_q$ Cycles |

| TSEG22 | TSEG21 | TSEG20 | Time Segment 2 |
|---|---|---|---|
| 0 | 0 | 0 | 1 $T_q$ Cycle(1) |
| 0 | 0 | 1 | 2 $T_q$ Cycles |
| . | . | . | . |
| . | . | . | . |
| 1 | 1 | 1 | 8$T_q$ Cycles |

1. This setting is not valid. Please refer to Table 12-4 for valid settings.

1. In this case PHASE_SEG1 must be at least 2 time quanta.

### 12.13.5 MSCAN08 Receiver Flag Register (CRFLG)

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. A flag can be cleared only when the condition which caused the setting is valid no more. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CRIER register. A hard or soft reset will clear the register.

Address: $0504

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | WUPIF | RWRNIF | TWRNIF | RERRIF | TERRIF | BOFFIF | OVRIF | RXF |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-20. Receiver Flag Register (CRFLG)**

**WUPIF — Wakeup Interrupt Flag**

If the MSCAN08 detects bus activity while in sleep mode, it sets the WUPIF flag. If not masked, a wakeup interrupt is pending while this flag is set.

1 = MSCAN08 has detected activity on the bus and requested wakeup.
0 = No wakeup interrupt has occurred.

**RWRNIF — Receiver Warning Interrupt Flag**

This flag is set when the MSCAN08 goes into warning status due to the receive error counter (REC) exceeding 96 and neither one of the error interrupt flags or the bus-off interrupt flag is set(1). If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 has gone into receiver warning status.
0 = No receiver warning status has been reached.

**TWRNIF — Transmitter Warning Interrupt Flag**

This flag is set when the MSCAN08 goes into warning status due to the transmit error counter (TEC) exceeding 96 and neither one of the error interrupt flags or the bus-off interrupt flag is set(2). If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 has gone into transmitter warning status.
0 = No transmitter warning status has been reached.

**RERRIF — Receiver Error Passive Interrupt Flag**

This flag is set when the MSCAN08 goes into error passive status due to the receive error counter exceeding 127 and the bus-off interrupt flag is not set(3). If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 has gone into receiver error passive status.
0 = No receiver error passive status has been reached.

1. Condition to set the flag: RWRNIF = (96 → REC) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
2. Condition to set the flag: TWRNIF = (96 → TEC) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
3. Condition to set the flag: RERRIF = (127 → REC → 255) & $\overline{\text{BOFFIF}}$

**TERRIF — Transmitter Error Passive Interrupt Flag**

This flag is set when the MSCAN08 goes into error passive status due to the transmit error counter exceeding 127 and the bus-off interrupt flag is not set(1). If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 went into transmit error passive status.
0 = No transmit error passive status has been reached.

**BOFFIF — Bus-Off Interrupt Flag**

This flag is set when the MSCAN08 goes into bus-off status, due to the transmit error counter exceeding 255. It cannot be cleared before the MSCAN08 has monitored 128 times 11 consecutive 'recessive' bits on the bus. If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08has gone into bus-off status.
0 = No bus-off status has been reached.

**OVRIF — Overrun Interrupt Flag**

This flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set.

1 = A data overrun has been detected since last clearing the flag.
0 = No data overrun has occurred.

**RXF — Receive Buffer Full**

The RXF flag is set by the MSCAN08 when a new message is available in the foreground receive buffer. This flag indicates whether the buffer is loaded with a correctly received message. After the CPU has read that message from the receive buffer the RXF flag must be cleared to release the buffer. A set RXF flag prohibits the exchange of the background receive buffer into the foreground buffer. If not masked, a receive interrupt is pending while this flag is set.

1 = The receive buffer is full. A new message is available.
0 = The receive buffer is released (not full).

***NOTE***

*To ensure data integrity, no registers of the receive buffer shall be read while the RXF flag is cleared.*
*The CRFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.*

1. Condition to set the flag: TERRIF = (128 → TEC → 255) & $\overline{\text{BOFFIF}}$

### 12.13.6 MSCAN08 Receiver Interrupt Enable Register

Address: $0505

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | WUPIE | RWRNIE | TWRNIE | RERRIE | TERRIE | BOFFIE | OVRIE | RXFIE |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 12-21. Receiver Interrupt Enable Register (CRIER)**

**WUPIE — Wakeup Interrupt Enable**

1 = A wakeup event will result in a wakeup interrupt.
0 = No interrupt will be generated from this event.

**RWRNIE — Receiver Warning Interrupt Enable**

1 = A receiver warning status event will result in an error interrupt.
0 = No interrupt is generated from this event.

**TWRNIE — Transmitter Warning Interrupt Enable**

1 = A transmitter warning status event will result in an error interrupt.
0 = No interrupt is generated from this event.

**RERRIE — Receiver Error Passive Interrupt Enable**

1 = A receiver error passive status event will result in an error interrupt.
0 = No interrupt is generated from this event.

**TERRIE — Transmitter Error Passive Interrupt Enable**

1 = A transmitter error passive status event will result in an error interrupt.
0 = No interrupt is generated from this event.

**BOFFIE — Bus-Off Interrupt Enable**

1 = A bus-off event will result in an error interrupt.
0 = No interrupt is generated from this event.

**OVRIE — Overrun Interrupt Enable**

1 = An overrun event will result in an error interrupt.
0 = No interrupt is generated from this event.

**RXFIE — Receiver Full Interrupt Enable**

1 = A receive buffer full (successful message reception) event will result in a receive interrupt.
0 = No interrupt will be generated from this event.

***NOTE***

*The CRIER register is held in the reset state when the SFTRES bit in CMCR0 is set.*

### 12.13.7 MSCAN08 Transmitter Flag Register

The abort acknowledge flags are read only. The transmitter buffer empty flags are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag setting. The transmitter buffer empty flags each have an associated interrupt enable bit in the CTCR register. A hard or soft reset will resets the register.

Address: $0506 5

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | ABTAK2 | ABTAK1 | ABTAK0 | 0 | TXE2 | TXE1 | TXE0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

= Unimplemented

**Figure 12-22. Transmitter Flag Register (CTFLG)**

**ABTAK2–ABTAK0 — Abort Acknowledge**

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent. The ABTAKx flag is cleared implicitly whenever the corresponding TXE flag is cleared.

1 = The message has been aborted.

0 = The message has not been aborted, thus has been sent out.

**TXE2–TXE0 — Transmitter Empty**

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The MSCAN08 sets the flag after the message has been sent successfully. The flag is also set by the MSCAN08 when the transmission request was successfully aborted due to a pending abort request (see 12.12.5 Transmit Buffer Priority Registers). If not masked, a receive interrupt is pending while this flag is set.

Clearing a TXEx flag also clears the corresponding ABTAKx flag (ABTAK, see above). When a TXEx flag is set, the corresponding ABTRQx bit (ABTRQ) is cleared. See 12.13.8 MSCAN08 Transmitter Control Register

1 = The associated message buffer is empty (not scheduled).

0 = The associated message buffer is full (loaded with a message due for transmission).

***NOTE***

*To ensure data integrity, no registers of the transmit buffers should be written to while the associated TXE flag is cleared.*
*The CTFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.*

## 12.13.8 MSCAN08 Transmitter Control Register

Address: $0507

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 12-23. Transmitter Control Register (CTCR)**

### ABTRQ2–ABTRQ0 — Abort Request

The CPU sets an ABTRQx bit to request that an already scheduled message buffer (TXE = 0) be aborted. The MSCAN08 will grant the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted the associated TXE and the abort acknowledge flag (ABTAK) (see 12.13.7 MSCAN08 Transmitter Flag Register) will be set and an TXE interrupt is generated if enabled. The CPU cannot reset ABTRQx. ABTRQx is cleared implicitly whenever the associated TXE flag is set.

1 = Abort request pending
0 = No abort request

***NOTE***

*The software must not clear one or more of the TXE flags in CTFLG and simultaneously set the respective ABTRQ bit(s).*

### TXEIE2–TXEIE0 — Transmitter Empty Interrupt Enable

1 = A transmitter empty (transmit buffer available for transmission) event results in a transmitter empty interrupt.
0 = No interrupt is generated from this event.

***NOTE***

*The CTCR register is held in the reset state when the SFTRES bit in CMCR0 is set.*

## 12.13.9 MSCAN08 Identifier Acceptance Control Register

Address: $0508

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | IDAM2 | IDAM1 | IDAM0 | 0 | IDHIT2 | IDHIT1 | IDHIT0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 12-24. Identifier Acceptance Control Register (CIDAC)**

### IDAM2–IDAM0— Identifier Acceptance Mode

The CPU sets these flags to define the identifier acceptance filter organization (see 12.5 Identifier Acceptance Filter). Table 12-9 summarizes the different settings. In "filter closed" mode no messages will be accepted so that the foreground buffer will never be reloaded.

**Table 12-9. Identifier Acceptance Mode Settings**

| IDAM2 | IDAM1 | IDAM0 | Identifier Acceptance Mode |
|---|---|---|---|
| 0 | 0 | 0 | Single 32-bit acceptance filter |
| 0 | 0 | 1 | Two 16-bit acceptance filter |
| 0 | 1 | 0 | Four 8-bit acceptance filters |
| 0 | 1 | 1 | Filter closed |
| 1 | X | X | Reserved |

### IDHIT2–IDHIT0— Identifier Acceptance Hit Indicator

The MSCAN08 sets these flags to indicate an identifier acceptance hit (see 12.5 Identifier Acceptance Filter). Table 12-9 summarizes the different settings.

**Table 12-10. Identifier Acceptance Hit Indication**

| IDHIT2 | IDHIT1 | IDHIT0 | Identifier Acceptance Hit |
|---|---|---|---|
| 0 | 0 | 0 | Filter 0 hit |
| 0 | 0 | 1 | Filter 1 hit |
| 0 | 1 | 0 | Filter 2 hit |
| 0 | 1 | 1 | Filter 3 hit |
| 1 | X | X | Reserved |

The IDHIT indicators are always related to the message in the foreground buffer. When a message gets copied from the background to the foreground buffer, the indicators are updated as well.

***NOTE***

*The CIDAC register can be written only if the SFTRES bit in the CMCR0 is set.*

## 12.13.10 MSCAN08 Receive Error Counter

Address: $050E

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 12-25. Receiver Error Counter (CRXERR)**

This read-only register reflects the status of the MSCAN08 receive error counter.

### 12.13.11 MSCAN08 Transmit Error Counter

Address: $050F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 12-26. Transmit Error Counter (CTXERR)**

This read-only register reflects the status of the MSCAN08 transmit error counter.

***NOTE***

*Both error counters may only be read when in sleep or soft reset mode.*

### 12.13.12 MSCAN08 Identifier Acceptance Registers

On reception each message is written into the background receive buffer. The CPU is only signalled to read the message, however, if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message will be overwritten by the next message (dropped).

The acceptance registers of the MSCAN08 are applied on the IDR0 to IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers only the first two (CIDMR0/CIDMR1 and CIDAR0/CIDAR1) are applied.

CIDAR0 Address: $0510

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| Reset: | Unaffected by reset | | | | | | | |

CIDAR1 Address: $050511

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| Reset: | Unaffected by reset | | | | | | | |

CIDAR2 Address: $0512

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| Reset: | Unaffected by reset | | | | | | | |

CIDAR3 Address: $0513

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| Reset: | Unaffected by reset | | | | | | | |

**Figure 12-27. Identifier Acceptance Registers (CIDAR0–CIDAR3)**

### AC7–AC0 — Acceptance Code Bits

AC7–AC0 comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

***NOTE***

*The CIDAR0–CIDAR3 registers can be written only if the SFTRES bit in CMCR0 is set*

## 12.13.13 MSCAN08 Identifier Mask Registers (CIDMR0–CIDMR3)

The identifier mask registers specify which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. For standard identifiers it is required to program the last three bits (AM2–AM0) in the mask register CIDMR1 to 'don't care'.

CIDMRO Address: $0514

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset: | Unaffected by reset | | | | | | | |

CIDMR1 Address: $0515

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset: | Unaffected by reset | | | | | | | |

CIDMR2 Address: $0516

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset: | Unaffected by reset | | | | | | | |

CIDMR3 Address: $0517

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset: | Unaffected by reset | | | | | | | |

**Figure 12-28. Identifier Mask Registers (CIDMR0–CIDMR3)**

### AM7–AM0 — Acceptance Mask Bits

If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match will be detected. The message will be accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted.

1 = Ignore corresponding acceptance code register bit.

0 = Match corresponding acceptance code register and identifier bits.

***NOTE***

*The CIDMR0–CIDMR3 registers can be written only if the SFTRES bit in the CMCR0 is set*

# Chapter 13
# Input/Output (I/O) Ports

## 13.1 Introduction

Bidirectional input-output (I/O) pins form seven parallel ports. All I/O pins are programmable as inputs or outputs. All individual bits within port A, port C, port D and port F are software configurable with pullup devices if configured as input port bits. The pullup devices are automatically and dynamically disabled when a port bit is switched to output mode.

## 13.2 Unused Pin Termination

Input pins and I/O port pins that are not used in the application must be terminated. This prevents excess current caused by floating inputs, and enhances immunity during noise or transient events. Termination methods include:

1. Configuring unused pins as outputs and driving high or low;
2. Configuring unused pins as inputs and enabling internal pull-ups;
3. Configuring unused pins as inputs and using external pull-up or pull-down resistors.

Never connect unused pins directly to $V_{DD}$ or $V_{SS}$.

Since some general-purpose I/O pins are not available on all packages, these pins must be terminated as well. Either method 1 or 2 above are appropriate.

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0000 | Port A Data Register (PTA) See page 173. | Read: | PTA7 | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0001 | Port B Data Register (PTB) See page 176. | Read: | PTB7 | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1 | PTB0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0002 | Port C Data Register (PTC) See page 178. | Read: | 1 | PTC6 | PTC5 | PTC4 | PTC3 | PTC2 | PTC1 | PTC0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0003 | Port D Data Register (PTD) See page 180. | Read: | PTD7 | PTD6 | PTD5 | PTD4 | PTD3 | PTD2 | PTD1 | PTD0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |

= Unimplemented

**Figure 13-1. I/O Port Register Summary (Sheet 1 of 3)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0004 | Data Direction Register A (DDRA) See page 174. | Read: Write: | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0005 | Data Direction Register B (DDRB) See page 176. | Read: Write: | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0006 | Data Direction Register C (DDRC) See page 178. | Read: | 0 | DDRC6 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0007 | Data Direction Register D (DDRD) See page 181. | Read: Write: | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0008 | Port E Data Register (PTE) See page 183. | Read: | 0 | 0 | PTE5 | PTE4 | PTE3 | PTE2 | PTE1 | PTE0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $000C | Data Direction Register E (DDRE) See page 184. | Read: | 0 | 0 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000D | Port A Input Pullup Enable Register (PTAPUE) See page 175. | Read: Write: | PTAPUE7 | PTAPUE6 | PTAPUE5 | PTAPUE4 | PTAPUE3 | PTAPUE2 | PTAPUE1 | PTAPUE0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000E | Port C Input Pullup Enable Register (PTCPUE) See page 180. | Read: | 0 | PTCPUE6 | PTCPUE5 | PTCPUE4 | PTCPUE3 | PTCPUE2 | PTCPUE1 | PTCPUE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000F | Port D Input Pullup Enable Register (PTDPUE) See page 182. | Read: Write: | PTDPUE7 | PTDPUE6 | PTDPUE5 | PTDPUE4 | PTDPUE3 | PTDPUE2 | PTDPUE1 | PTDPUE0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0440 | Port F Data Register (PTF) See page 185. | Read: Write: | PTF7 | PTF6 | PTF5 | PTF4 | PTAF3 | PTF2 | PTF1 | PTF0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0441 | Port G Data Register (PTG) See page 186. | Read: Write: | PTG7 | PTG6 | PTG5 | PTG4 | PTG3 | PTG2 | PTG1 | PTG0 |
| | | Reset: | Unaffected by reset | | | | | | | |

= Unimplemented

**Figure 13-1. I/O Port Register Summary (Sheet 2 of 3)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0444 | Data Direction Register F (DDRF) See page 185. | Read: Write: | DDRF7 | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0445 | Data Direction Register G (DDRG) See page 187. | Read: Write: | DDRG7 | DDRG6 | DDRG5 | DDRG4 | DDRG3 | DDRG2 | DDRG1 | DDRG0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 13-1. I/O Port Register Summary (Sheet 3 of 3)**

**Table 13-1. Port Control Register Bits Summary**

| Port | Bit | DDR | Module Control | | Module Control | | Pin |
|---|---|---|---|---|---|---|---|
| A | 0 | DDRA0 | KBD | KBIE0 | ADC[15:8] | ADCH4–ADCH0 | PTA0/KBD0/AD8 |
| | 1 | DDRA1 | | KBIE1 | | | PTA1/KBD1/AD9 |
| | 2 | DDRA2 | | KBIE2 | | | PTA2/KBD2/AD10 |
| | 3 | DDRA3 | | KBIE3 | | | PTA3/KBD3/AD11 |
| | 4 | DDRA4 | | KBIE4 | | | PTA4/KBD4/AD12 |
| | 5 | DDRA5 | | KBIE5 | | | PTA5/KBD5/AD13 |
| | 6 | DDRA6 | | KBIE6 | | | PTA6/KBD6/AD14 |
| | 7 | DDRA7 | | KBIE7 | | | PTA7/KBD7/AD15 |
| B | 0 | DDRB0 | ADC | ADCH4–ADCH0 | — | — | PTB0/AD0 |
| | 1 | DDRB1 | | | | | PTB1/AD1 |
| | 2 | DDRB2 | | | | | PTB2/AD2 |
| | 3 | DDRB3 | | | | | PTB3/AD3 |
| | 4 | DDRB4 | | | | | PTB4/AD4 |
| | 5 | DDRB5 | | | | | PTB5/AD5 |
| | 6 | DDRB6 | | | | | PTB6/AD6 |
| | 7 | DDRB7 | | | | | PTB7/AD7 |
| C | 0 | DDRC0 | MSCAN | CANEN | — | — | PTC0 |
| | 1 | DDRC1 | | | | | PTC1 |
| | 2 | DDRC2 | | | | | PTC2 |
| | 3 | DDRC3 | | | | | PTC3 |
| | 4 | DDRC4 | | | | | PTC4 |
| | 5 | DDRC5 | | | | | PTC5 |
| | 6 | DDRC6 | | | | | PTC6 |

— Continued on next page

**Table 13-1. Port Control Register Bits Summary (Continued)**

| Port | Bit | DDR | Module Control | | Module Control | | Pin |
|---|---|---|---|---|---|---|---|
| D | 0 | DDRD0 | SPI | SPE | — | — | PTD0/$\overline{SS}$/MCLK |
| | 1 | DDRD1 | | | | | PTD1/MISO |
| | 2 | DDRD2 | | | | | PTD2/MOSI |
| | 3 | DDRD3 | | | | | PTD3/SPSCK |
| | 4 | DDRD4 | TIM1 | ELS0B:ELS0A | | | PTD4/T1CH0 |
| | 5 | DDRD5 | | ELS1B:ELS1A | | | PTD5/T1CH1 |
| | 6 | DDRD6 | TIM2 | ELS0B:ELS0A | | | PTD6/T2CH0 |
| | 7 | DDRD7 | | ELS1B:ELS1A | | | PTD7/T2CH1 |
| E | 0 | DDRE0 | SCI | ENSCI | — | — | PTE0/TxD |
| | 1 | DDRE1 | | | | | PTE1/RxD |
| | 2 | DDRE2 | | | | | PTE2 |
| | 3 | DDRE3 | | | | | PTE3 |
| | 4 | DDRE4 | | | | | PTE4 |
| | 5 | DDRE5 | | | | | PTE5 |
| F | 0 | DDRF0 | | | — | — | PTF0 |
| | 1 | DDRF1 | | | | | PTF1 |
| | 2 | DDRF2 | | | | | PTF2 |
| | 3 | DDRF3 | | | | | PTF3 |
| | 4 | DDRF4 | TIM2 | ELS2B:ELS2A | | | PTF4/T2CH2 |
| | 5 | DDRF5 | | ELS3B:ELS3A | | | PTF5/T2CH3 |
| | 6 | DDRF6 | | ELS4B:ELS4A | | | PTF6/T2CH4 |
| | 7 | DDRF7 | | ELS5B:ELS5A | | | PTF7/T2CH5 |
| G | 0 | DDRG0 | ADC | ADCH[23:16] | — | — | PTG0/AD16 |
| | 1 | DDRG1 | | | | | PTG1/AD17 |
| | 2 | DDRG2 | | | | | PTG2/AD18 |
| | 3 | DDRG3 | | | | | PTG3/AD19 |
| | 4 | DDRG4 | | | | | PTG4/AD20 |
| | 5 | DDRG5 | | | | | PTG5/AD21 |
| | 6 | DDRG6 | | | | | PTG6/AD22 |
| | 7 | DDRG7 | | | | | PTG7/AD23 |

## 13.3 Port A

Port A is an 8-bit special-function port that shares all eight of its pins with the keyboard interrupt (KBI) module and the ADC module. Port A also has software configurable pullup devices if configured as an input port.

### 13.3.1 Port A Data Register

The port A data register (PTA) contains a data latch for each of the eight port A pins.

Address: $0000

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | PTA7 | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0 |
| Reset: | Unaffected by reset | | | | | | | |
| Alternate Function: | KBD7 | KBD6 | KBD5 | KBD4 | KBD3 | KBD2 | KBD1 | KBD0 |
| Alternate Function: | AD15 | AD14 | AD13 | AD12 | AD11 | AD10 | AD9 | AD8 |

**Figure 13-2. Port A Data Register (PTA)**

**PTA7–PTA0 — Port A Data Bits**

These read/write bits are software programmable. Data direction of each port A pin is under the control of the corresponding bit in data direction register A. Reset has no effect on port A data.

**KBD7–KBD0 — Keyboard Inputs**

The keyboard interrupt enable bits, KBIE7–KBIE0, in the keyboard interrupt control register (KBICR) enable the port A pins as external interrupt pins. See Chapter 9 Keyboard Interrupt Module (KBI)

**AD15–AD8 — Analog-to-Digital Input Bits**

AD15–AD8 are pins used for the input channels to the analog-to-digital converter module. The channel select bits in the ADC status and control register define which port A pin will be used as an ADC input and overrides any control from the port I/O logic by forcing that pin as the input to the analog circuitry.

***NOTE***

*Care must be taken when reading port A while applying analog voltages to AD15–AD8 pins. If the appropriate ADC channel is not enabled, excessive current drain may occur if analog voltages are applied to the PTAx/KBDx/ADx pin, while PTA is read as a digital input during the CPU read cycle. Those ports not selected as analog input channels are considered digital I/O ports.*

### 13.3.2 Data Direction Register A

Data direction register A (DDRA) determines whether each port A pin is an input or an output. Writing a 1 to a DDRA bit enables the output buffer for the corresponding port A pin; a 0 disables the output buffer.

Address: $0004

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-3. Data Direction Register A (DDRA)**

**DDRA7–DDRA0 — Data Direction Register A Bits**

These read/write bits control port A data direction. Reset clears DDRA7–DDRA0, configuring all port A pins as inputs.

1 = Corresponding port A pin configured as output
0 = Corresponding port A pin configured as input

***NOTE***

*Avoid glitches on port A pins by writing to the port A data register before changing data direction register A bits from 0 to 1.*

Figure 13-4 shows the port A I/O logic.

When bit DDRAx is a 1, reading address $0000 reads the PTAx data latch. When bit DDRAx is a 0, reading address $0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-2 summarizes the operation of the port A pins.

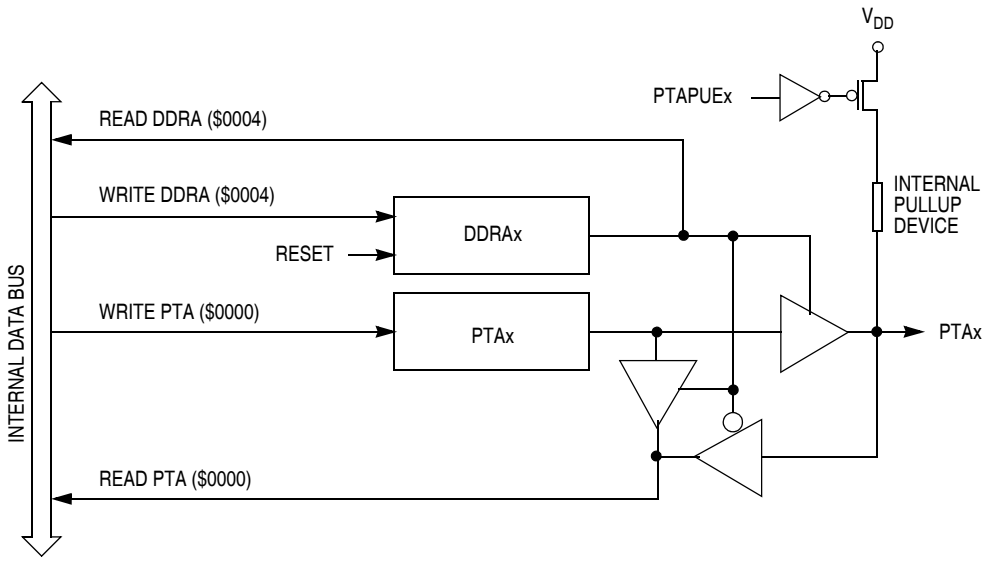



**Figure 13-4. Port A I/O Circuit**

**Table 13-2. Port A Pin Functions**

| PTAPUE Bit | DDRA Bit | PTA Bit | I/O Pin Mode | Accesses to DDRA | Accesses to PTA | |
|---|---|---|---|---|---|---|
| | | | | Read/Write | Read | Write |
| 1 | 0 | X(1) | Input, $V_{DD}$(2) | DDRA7–DDRA0 | Pin | PTA7–PTA0(3) |
| 0 | 0 | X | Input, Hi-Z(4) | DDRA7–DDRA0 | Pin | PTA7–PTA0(3) |
| X | 1 | X | Output | DDRA7–DDRA0 | PTA7–PTA0 | PTA7–PTA0 |

1. X = Don't care
2. I/O pin pulled up to $V_{DD}$ by internal pullup device
3. Writing affects data register, but does not affect input.
4. Hi-Z = High impedance

### 13.3.3 Port A Input Pullup Enable Register

The port A input pullup enable register (PTAPUE) contains a software configurable pullup device for each of the eight port A pins. Each bit is individually configurable and requires that the data direction register, DDRA, bit be configured as an input. Each pullup is automatically and dynamically disabled when a port bit's DDRA is configured for output mode.

***NOTE***

*Pullup or pulldown resistors are automatically selected for keyboard interrupt pins depending on the bit settings in the keyboard interrupt polarity register (INTKBIPR) see 9.7.3 Keyboard Interrupt Polarity Register.*

Address: $000D

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | PTAPUE7 | PTAPUE6 | PTAPUE5 | PTAPUE4 | PTAPUE3 | PTAPUE2 | PTAPUE1 | PTAPUE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-5. Port A Input Pullup Enable Register (PTAPUE)**

**PTAPUE7–PTAPUE0 — Port A Input Pullup Enable Bits**

These writable bits are software programmable to enable pullup devices on an input port bit.

1 = Corresponding port A pin configured to have internal pullup
0 = Corresponding port A pin has internal pullup disconnected

## 13.4 Port B

Port B is an 8-bit special-function port that shares all eight of its pins with the analog-to-digital converter (ADC) module.

### 13.4.1 Port B Data Register

The port B data register (PTB) contains a data latch for each of the eight port pins.

| Address: | $0001 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read:<br>Write: | PTB7 | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1 | PTB0 |
| Reset: | Unaffected by reset | | | | | | | |
| Alternate Function: | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 |

**Figure 13-6. Port B Data Register (PTB)**

**PTB7–PTB0 — Port B Data Bits**

These read/write bits are software-programmable. Data direction of each port B pin is under the control of the corresponding bit in data direction register B. Reset has no effect on port B data.

**AD7–AD0 — Analog-to-Digital Input Bits**

AD7–AD0 are pins used for the input channels to the analog-to-digital converter module. The channel select bits in the ADC status and control register define which port B pin will be used as an ADC input and overrides any control from the port I/O logic by forcing that pin as the input to the analog circuitry.

***NOTE***

*Care must be taken when reading port B while applying analog voltages to AD7–AD0 pins. If the appropriate ADC channel is not enabled, excessive current drain may occur if analog voltages are applied to the PTBx/ADx pin, while PTB is read as a digital input during the CPU read cycle. Those ports not selected as analog input channels are considered digital I/O ports.*

### 13.4.2 Data Direction Register B

Data direction register B (DDRB) determines whether each port B pin is an input or an output. Writing a 1 to a DDRB bit enables the output buffer for the corresponding port B pin; a 0 disables the output buffer.

| Address: | $0005 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read:<br>Write: | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-7. Data Direction Register B (DDRB)**

### DDRB7–DDRB0 — Data Direction Register B Bits

These read/write bits control port B data direction. Reset clears DDRB7–DDRB0, configuring all port B pins as inputs.

1 = Corresponding port B pin configured as output
0 = Corresponding port B pin configured as input

***NOTE***

*Avoid glitches on port B pins by writing to the port B data register before changing data direction register B bits from 0 to 1.*

Figure 13-8 shows the port B I/O logic.

When bit DDRBx is a 1, reading address $0001 reads the PTBx data latch. When bit DDRBx is a 0, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-3 summarizes the operation of the port B pins.

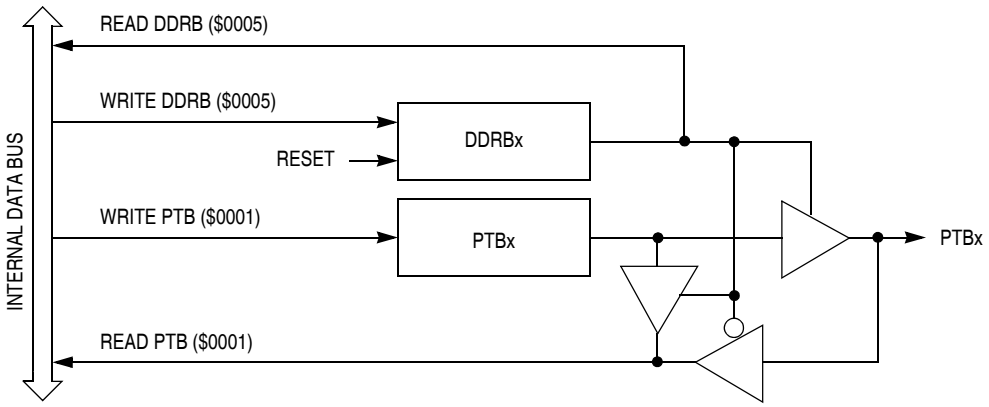



**Figure 13-8. Port B I/O Circuit**

**Table 13-3. Port B Pin Functions**

| DDRB Bit | PTB Bit | I/O Pin Mode | Accesses to DDRB | Accesses to PTB | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X(1) | Input, Hi-Z(2) | DDRB7–DDRB0 | Pin | PTB7–PTB0(3) |
| 1 | X | Output | DDRB7–DDRB0 | PTB7–PTB0 | PTB7–PTB0 |

1. X = Don't care
2. Hi-Z = High impedance
3. Writing affects data register, but does not affect input.

## 13.5 Port C

Port C is a 7-bit, general-purpose bidirectional I/O port. Port C also has software configurable pullup devices if configured as an input port. PTC[1:0] are shared with the MSCAN module.

### 13.5.1 Port C Data Register

The port C data register (PTC) contains a data latch for each of the seven port C pins.

***NOTE***

*Bit 6 through bit 2 of PTC are not available in the 32-pin LQFP package.*

Address: $0002

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 1 | PTC6 | PTC5 | PTC4 | PTC3 | PTC2 | PTC1 | PTC0 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |
| Alternate Function: | | | | | | | $CAN_{RX}$ | $CAN_{TX}$ |

= Unimplemented

**Figure 13-9. Port C Data Register (PTC)**

**PTC6–PTC0 — Port C Data Bits**

These read/write bits are software-programmable. Data direction of each port C pin is under the control of the corresponding bit in data direction register C. Reset has no effect on port C data.

**$CAN_{RX}$ and $CAN_{TX}$ — MSCAN08 Bits**

The $CAN_{RX}$–$CAN_{TX}$ pins are the MSCAN08 modules receive and transmit pins. The CANEN bit in the MSCAN08 control register determines, whether the PTC1/$CAN_{RX}$–PTC0/$CAN_{TX}$ pins are MSCAN08 pins or general-purpose I/O pins. See Chapter 12 MSCAN08 Controller (MSCAN08).

### 13.5.2 Data Direction Register C

Data direction register C (DDRC) determines whether each port C pin is an input or an output. Writing a 1 to a DDRC bit enables the output buffer for the corresponding port C pin; a 0 disables the output buffer.

Address: $0006

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | DDRC6 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 13-10. Data Direction Register C (DDRC)**

### DDRC6–DDRC0 — Data Direction Register C Bits

These read/write bits control port C data direction. Reset clears DDRC6–DDRC0, configuring all port C pins as inputs.

1 = Corresponding port C pin configured as output
0 = Corresponding port C pin configured as input

***NOTE***

*Avoid glitches on port C pins by writing to the port C data register before changing data direction register C bits from 0 to 1.*

Figure 13-11 shows the port C I/O logic.

When bit DDRCx is a 1, reading address $0002 reads the PTCx data latch. When bit DDRCx is a 0, reading address $0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-4 summarizes the operation of the port C pins.

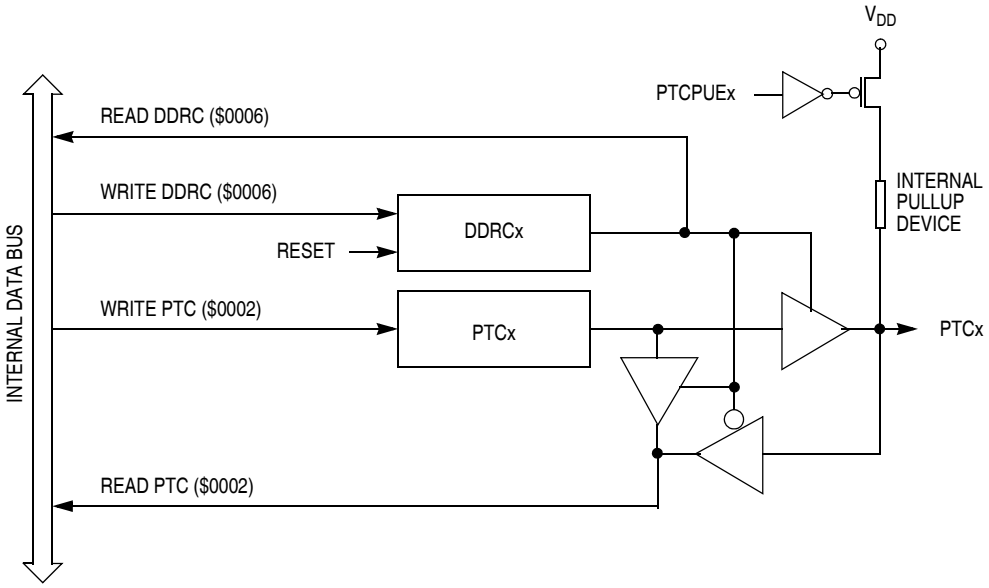



**Figure 13-11. Port C I/O Circuit**

**Table 13-4. Port C Pin Functions**

| PTCPUE Bit | DDRC Bit | PTC Bit | I/O Pin Mode | Accesses to DDRC | Accesses to PTC | |
|---|---|---|---|---|---|---|
| | | | | Read/Write | Read | Write |
| 1 | 0 | X(1) | Input, $V_{DD}$(2) | DDRC6–DDRC0 | Pin | PTC6–PTC0(3) |
| 0 | 0 | X | Input, Hi-Z(4) | DDRC6–DDRC0 | Pin | PTC6–PTC0(3) |
| X | 1 | X | Output | DDRC6–DDRC0 | PTC6–PTC0 | PTC6–PTC0 |

1. X = Don't care
2. I/O pin pulled up to $V_{DD}$ by internal pullup device.
3. Writing affects data register, but does not affect input.
4. Hi-Z = High impedance

### 13.5.3 Port C Input Pullup Enable Register

The port C input pullup enable register (PTCPUE) contains a software configurable pullup device for each of the seven port C pins. Each bit is individually configurable and requires that the data direction register, DDRC, bit be configured as an input. Each pullup is automatically and dynamically disabled when a port bit's DDRC is configured for output mode.

Address: $000E

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | PTCPUE6 | PTCPUE5 | PTCPUE4 | PTCPUE3 | PTCPUE2 | PTCPUE1 | PTCPUE0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 13-12. Port C Input Pullup Enable Register (PTCPUE)**

**PTCPUE6–PTCPUE0 — Port C Input Pullup Enable Bits**

These writable bits are software programmable to enable pullup devices on an input port bit.

1 = Corresponding port C pin configured to have internal pullup
0 = Corresponding port C pin internal pullup disconnected

## 13.6 Port D

Port D is an 8-bit special-function port that shares four of its pins with the serial peripheral interface (SPI) module and four of its pins with two timer interface (TIM1 and TIM2) modules. Port D also has software configurable pullup devices if configured as an input port. PTD0 is shared with the MCLK output.

### 13.6.1 Port D Data Register

The port D data register (PTD) contains a data latch for each of the eight port D pins.

Address: $0003

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | PTD7 | PTD6 | PTD5 | PTD4 | PTD3 | PTD2 | PTD1 | PTD0 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |
| Alternate Function: | T2CH1 | T2CH0 | $\overline{\text{T1CH1}}$ | T1CH0 | SPSCK | $\overline{\text{MOSI}}$ | MISO | $\overline{\text{SS}}$ |
| | | | | | | | | MCLK |

**Figure 13-13. Port D Data Register (PTD)**

**PTD7–PTD0 — Port D Data Bits**

These read/write bits are software-programmable. Data direction of each port D pin is under the control of the corresponding bit in data direction register D. Reset has no effect on port D data.

**T2CH1 and T2CH0 — Timer 2 Channel I/O Bits**

The PTD5/T2CH1–PTD4/T2CH0 pins are the TIM2 input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTD7/T2CH1–PTD6/T2CH0 pins are timer channel I/O pins or general-purpose I/O pins. See Chapter 18 Timer Interface Module (TIM1) and Chapter 19 Timer Interface Module (TIM2).

**T1CH1 and T1CH0 — Timer 1 Channel I/O Bits**

The PTD7/T1CH1–PTD6/T1CH0 pins are the TIM1 input capture/output compare pins. The edge/level select bits, ELSxB and ELSxA, determine whether the PTD7/T1CH1–PTD6/T1CH0 pins are timer channel I/O pins or general-purpose I/O pins. See Chapter 18 Timer Interface Module (TIM1) and Chapter 19 Timer Interface Module (TIM2).

**SPSCK — SPI Serial Clock**

The PTD3/SPSCK pin is the serial clock input of the SPI module. When the SPE bit is clear, the PTD3/SPSCK pin is available for general-purpose I/O.

**MOSI — Master Out/Slave In**

The PTD2/MOSI pin is the master out/slave in terminal of the SPI module. When the SPE bit is clear, the PTD2/MOSI pin is available for general-purpose I/O.

**MISO — Master In/Slave Out**

The PTD1/MISO pin is the master in/slave out terminal of the SPI module. When the SPI enable bit, SPE, is clear, the SPI module is disabled, and the PTD1/MISO pin is available for general-purpose I/O.

**$\overline{SS}$ — Slave Select**

The PTD0/$\overline{SS}$ pin is the slave select input of the SPI module. When the SPE bit is clear, or when the SPI master bit, SPMSTR, is set, the PTD0/$\overline{SS}$ pin is available for general-purpose I/O. When the SPI is enabled, the DDRD0 bit in data direction register D (DDRD) has no effect on the PTD0/$\overline{SS}$ pin.

Data direction register D (DDRD) does not affect the data direction of port D pins that are being used by the SPI module. However, the DDRD bits always determine whether reading port D returns the states of the latches or the states of the pins. See Table 13-5.

## 13.6.2 Data Direction Register D

Data direction register D (DDRD) determines whether each port D pin is an input or an output. Writing a 1 to a DDRD bit enables the output buffer for the corresponding port D pin; a 0 disables the output buffer.

Address: $0007

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-14. Data Direction Register D (DDRD)**

**DDRD7–DDRD0 — Data Direction Register D Bits**

These read/write bits control port D data direction. Reset clears DDRD7–DDRD0, configuring all port D pins as inputs.

1 = Corresponding port D pin configured as output
0 = Corresponding port D pin configured as input

***NOTE***

*Avoid glitches on port D pins by writing to the port D data register before changing data direction register D bits from 0 to 1.*

Figure 13-15 shows the port D I/O logic.

When bit DDRDx is a 1, reading address $0003 reads the PTDx data latch. When bit DDRDx is a 0, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-5 summarizes the operation of the port D pins.

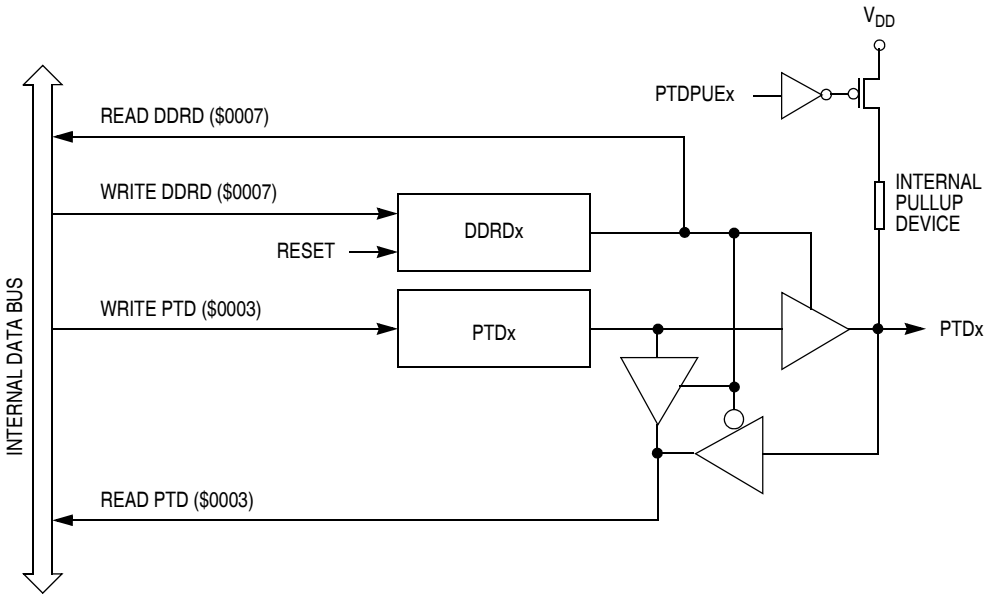



**Figure 13-15. Port D I/O Circuit**

**Table 13-5. Port D Pin Functions**

| PTDPUE Bit | DDRD Bit | PTD Bit | I/O Pin Mode | Accesses to DDRD | Accesses to PTD | |
|---|---|---|---|---|---|---|
| | | | | Read/Write | Read | Write |
| 1 | 0 | X(1) | Input, $V_{DD}$(2) | DDRD7–DDRD0 | Pin | PTD7–PTD0(3) |
| 0 | 0 | X | Input, Hi-Z(4) | DDRD7–DDRD0 | Pin | PTD7–PTD0(3) |
| X | 1 | X | Output | DDRD7–DDRD0 | PTD7–PTD0 | PTD7–PTD0 |

1. X = Don't care
2. I/O pin pulled up to $V_{DD}$ by internal pullup device.
3. Writing affects data register, but does not affect input.
4. Hi-Z = High imp[edance

## 13.6.3 Port D Input Pullup Enable Register

The port D input pullup enable register (PTDPUE) contains a software configurable pullup device for each of the eight port D pins. Each bit is individually configurable and requires that the data direction register, DDRD, bit be configured as an input. Each pullup is automatically and dynamically disabled when a port bit's DDRD is configured for output mode.

Address: $000F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | PTDPUE7 | PTDPUE6 | PTDPUE5 | PTDPUE4 | PTDPUE3 | PTDPUE2 | PTDPUE1 | PTDPUE0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-16. Port D Input Pullup Enable Register (PTDPUE)**

**PTDPUE7–PTDPUE0 — Port D Input Pullup Enable Bits**

These writable bits are software programmable to enable pullup devices on an input port bit.

1 = Corresponding port D pin configured to have internal pullup
0 = Corresponding port D pin has internal pullup disconnected

## 13.7 Port E

Port E is a 6-bit special-function port that shares two of its pins with the enhanced serial communications interface (ESCI) module.

### 13.7.1 Port E Data Register

The port E data register contains a data latch for each of the six port E pins.

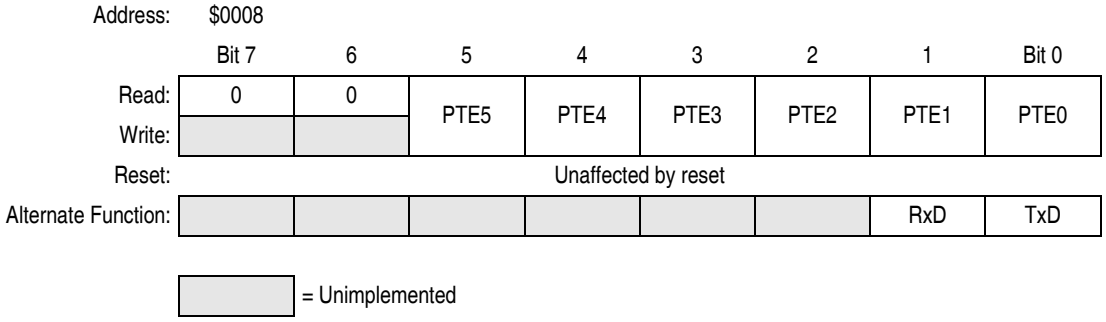



**Figure 13-17. Port E Data Register (PTE)**

**PTE5–PTE0 — Port E Data Bits**

These read/write bits are software-programmable. Data direction of each port E pin is under the control of the corresponding bit in data direction register E. Reset has no effect on port E data.

***NOTE***

*Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the ESCI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. See Table 13-6.*

**RxD — SCI Receive Data Input**

The PTE1/RxD pin is the receive data input for the ESCI module.

When the enable SCI bit, ENSCI, is clear, the ESCI module is disabled, and the PTE1/RxD pin is available for general-purpose I/O. See Chapter 14 Enhanced Serial Communications Interface (ESCI) Module.

**TxD — SCI Transmit Data Output**

The PTE0/TxD pin is the transmit data output for the ESCI module. When the enable SCI bit, ENSCI, is clear, the ESCI module is disabled, and the PTE0/TxD pin is available for general-purpose I/O. See Chapter 14 Enhanced Serial Communications Interface (ESCI) Module.

### 13.7.2 Data Direction Register E

Data direction register E (DDRE) determines whether each port E pin is an input or an output. Writing a 1 to a DDRE bit enables the output buffer for the corresponding port E pin; a 0 disables the output buffer.

Address: $000C

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 13-18. Data Direction Register E (DDRE)**

**DDRE5–DDRE0 — Data Direction Register E Bits**

These read/write bits control port E data direction. Reset clears DDRE5–DDRE0, configuring all port E pins as inputs.

1 = Corresponding port E pin configured as output
0 = Corresponding port E pin configured as input

***NOTE***

*Avoid glitches on port E pins by writing to the port E data register before changing data direction register E bits from 0 to 1.*

Figure 13-19 shows the port E I/O logic.

When bit DDREx is a 1, reading address $0008 reads the PTEx data latch. When bit DDREx is a 0, reading address $0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-6 summarizes the operation of the port E pins.

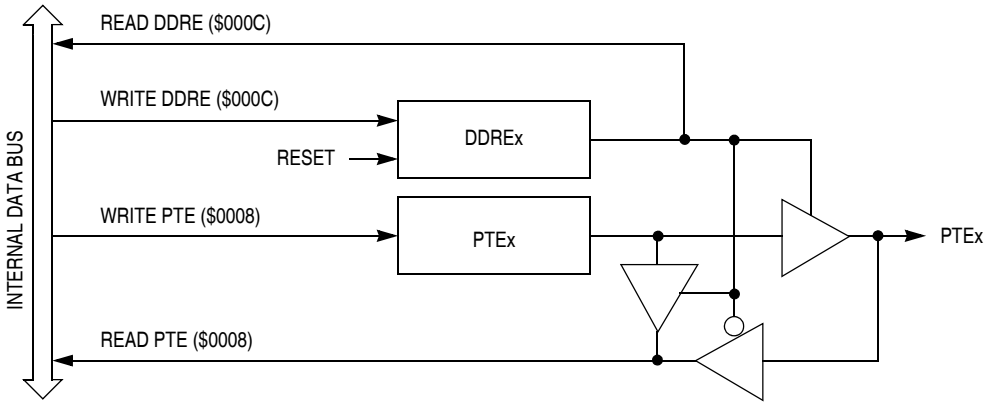



**Figure 13-19. Port E I/O Circuit**

**Table 13-6. Port E Pin Functions**

| DDRE Bit | PTE Bit | I/O Pin Mode | Accesses to DDRE | Accesses to PTE | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X(1) | Input, Hi-Z(2) | DDRE5–DDRE0 | Pin | PTE5–PTE0(3) |
| 1 | X | Output | DDRE5–DDRE0 | PTE5–PTE0 | PTE5–PTE0 |

1. X = Don't care
2. Hi-Z = High impedance
3. Writing affects data register, but does not affect input.

## 13.8 Port F

Port F is an 8-bit special-function port that shares four of its pins with the timer interface (TIM2) module.

### 13.8.1 Port F Data Register

The port F data register (PTF) contains a data latch for each of the eight port F pins.

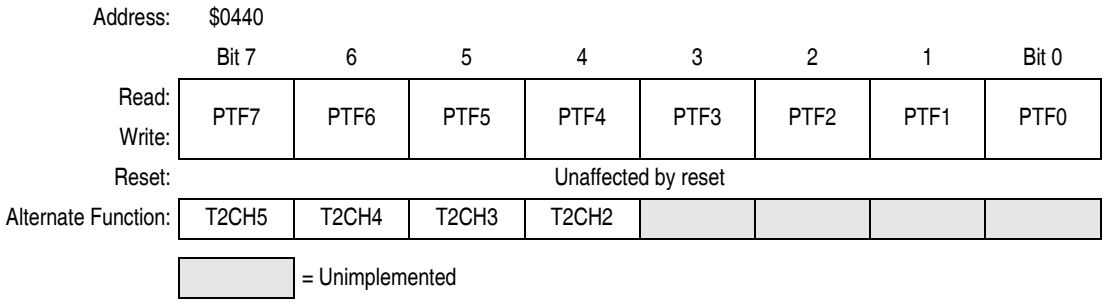



**Figure 13-20. Port F Data Register (PTF)**

**PTF7–PTF0 — Port F Data Bits**

These read/write bits are software-programmable. Data direction of each port F pin is under the control of the corresponding bit in data direction register F. Reset has no effect on port F data.

**T2CH5–T2CH2 — Timer 2 Channel I/O Bits**

The PTF7/T2CH5–PTF4/T2CH2 pins are the TIM2 input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF7/T2CH5–PTF4/T2CH2 pins are timer channel I/O pins or general-purpose I/O pins. See Chapter 18 Timer Interface Module (TIM1) and Chapter 19 Timer Interface Module (TIM2).

### 13.8.2 Data Direction Register F

Data direction register F (DDRF) determines whether each port F pin is an input or an output. Writing a 1 to a DDRF bit enables the output buffer for the corresponding port F pin; a 0 disables the output buffer.

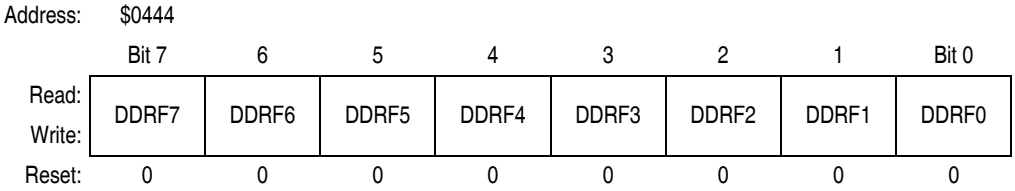



**Figure 13-21. Data Direction Register F (DDRF)**

**DDRF7–DDRF0 — Data Direction Register F Bits**

These read/write bits control port F data direction. Reset clears DDRF7–DDRF0, configuring all port F pins as inputs.

1 = Corresponding port F pin configured as output
0 = Corresponding port F pin configured as input

***NOTE***

*Avoid glitches on port F pins by writing to the port F data register before changing data direction register F bits from 0 to 1.*

Figure 13-22 shows the port F I/O logic.

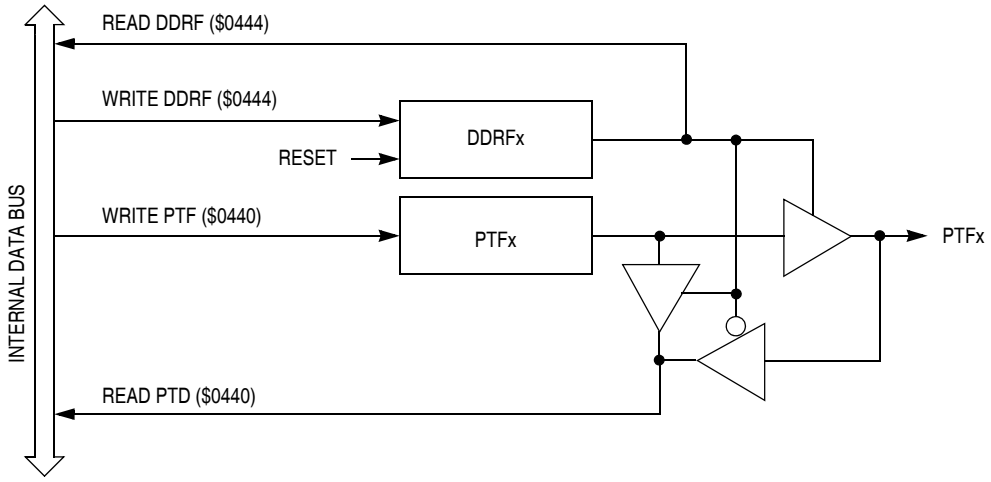



**Figure 13-22. Port F I/O Circuit**

When bit DDRFx is a 1, reading address $0440 reads the PTFx data latch. When bit DDRFx is a 0, reading address $0440 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-7 summarizes the operation of the port F pins.

**Table 13-7. Port F Pin Functions**

| DDRF Bit | PTF Bit | I/O Pin Mode | Accesses to DDRF | Accesses to PTF | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | WritE |
| 0 | X(1) | Input, Hi-Z(2) | DDRF7–DDRF0 | Pin | PTF7–PTF0(3) |
| 1 | X | Output | DDRF7–DDRF0 | PTF7–PTF0 | PTF7–PTF0 |

1. X = Don't care
2. Hi-Z = High impedance
3. Writing affects data register, but does not affect input.

## 13.9 Port G

Port G is an 8-bit special-function port that shares all eight of its pins with the analog-to-digital converter (ADC) module.

### 13.9.1 Port G Data Register

The port G data register (PTG) contains a data latch for each of the eight port pins.

| Address: | $0441 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read:<br>Write: | PTG7 | PTG6 | PTG5 | PTG4 | PTG3 | PTG2 | PTG1 | PTG0 |
| Reset: | Unaffected by reset | | | | | | | |
| Alternate Function: | AD23 | AD22 | AD21 | AD20 | AD19 | AD18 | AD17 | AD16 |

**Figure 13-23. Port G Data Register (PTG)**

**PTG7–PTG0 — Port G Data Bits**

These read/write bits are software-programmable. Data direction of each port G pin is under the control of the corresponding bit in data direction register G. Reset has no effect on port G data.

**AD23–AD16 — Analog-to-Digital Input Bits**

AD23–AD16 are pins used for the input channels to the analog-to-digital converter module. The channel select bits in the ADC status and control register define which port G pin will be used as an ADC input and overrides any control from the port I/O logic by forcing that pin as the input to the analog circuitry.

***NOTE***

*Care must be taken when reading port G while applying analog voltages to AD23–AD16 pins. If the appropriate ADC channel is not enabled, excessive current drain may occur if analog voltages are applied to the PTGx/ADx pin, while PTG is read as a digital input during the CPU read cycle. Those ports not selected as analog input channels are considered digital I/O ports.*

## 13.9.2 Data Direction Register G

Data direction register G (DDRG) determines whether each port G pin is an input or an output. Writing a 1 to a DDRG bit enables the output buffer for the corresponding port G pin; a 0 disables the output buffer.

Address: $0445

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | DDRG7 | DDRG6 | DDRG5 | DDRG4 | DDRG3 | DDRG2 | DDRG1 | DDRG0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-24. Data Direction Register G (DDRG)**

**DDRG7–DDRG0 — Data Direction Register G Bits**

These read/write bits control port G data direction. Reset clears DDRG7–DDRG0], configuring all port G pins as inputs.

1 = Corresponding port G pin configured as output
0 = Corresponding port G pin configured as input

***NOTE***

*Avoid glitches on port G pins by writing to the port G data register before changing data direction register G bits from 0 to 1.*

Figure 13-25 shows the port G I/O logic.

When bit DDRGx is a 1, reading address $0441 reads the PTGx data latch. When bit DDRGx is a 0, reading address $0441 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-8 summarizes the operation of the port G pins.

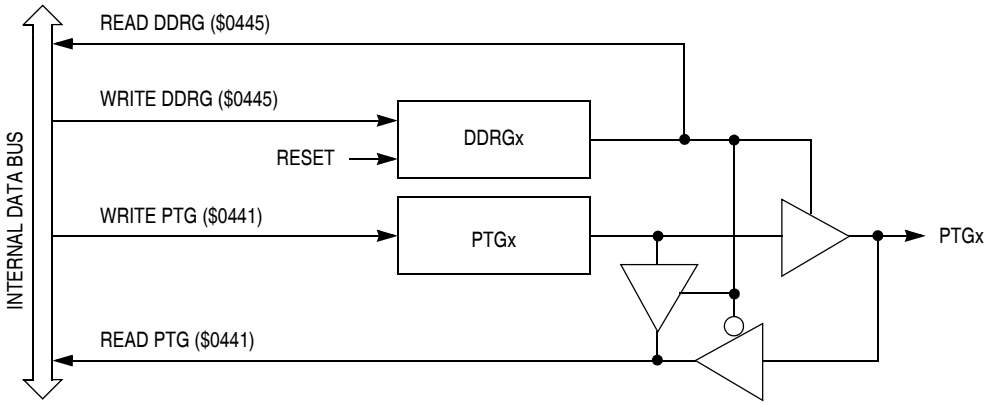



**Figure 13-25. Port G I/O Circuit**

**Table 13-8. Port G Pin Functions**

| DDRG Bit | PTG Bit | I/O Pin Mode | Accesses to DDRG | Accesses to PTG | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X(1) | Input, Hi-Z(2) | DDRG7–DDRG0 | Pin | PTG7–PTG0(3) |
| 1 | X | Output | DDRG7–DDRG0 | PTG7–PTG0 | PTG7–PTG0 |

1. X = Don't care
2. Hi-Z = High impedance
3. Writing affects data register, but does not affect input.

# Chapter 14
# Enhanced Serial Communications Interface (ESCI) Module

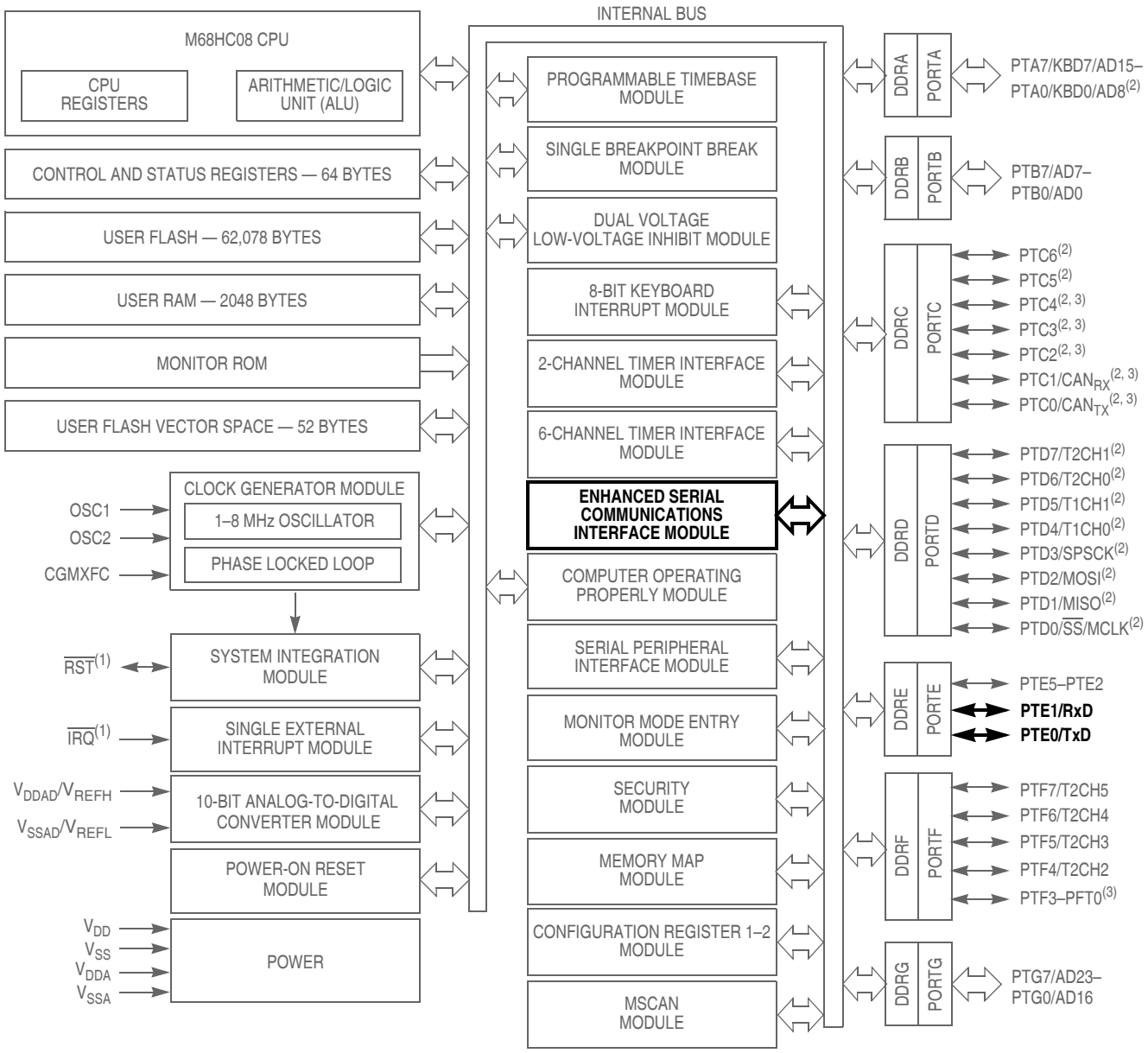
## 14.1 Introduction

The enhanced serial communications interface (ESCI) module allows asynchronous communications with peripheral devices and other microcontroller units (MCU).

## 14.2 Features

Features include:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- Programmable baud rates
- Programmable 8-bit or 9-bit character length
- Separately enabled transmitter and receiver
- Separate receiver and transmitter central processor unit (CPU) interrupt requests
- Programmable transmitter output polarity
- Two receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
- Interrupt-driven operation with eight interrupt flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection




1. Pin contains integrated pullup device.
2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.
3. Higher current drive port pins

**Figure 14-1. Block Diagram Highlighting ESCI Block and Pins**

## 14.3 Pin Name Conventions

The generic names of the ESCI input/output (I/O) pins are:

- RxD (receive data)
- TxD (transmit data)

ESCI I/O lines are implemented by sharing parallel I/O port pins. The full name of an ESCI input or output reflects the name of the shared port pin. Table 14-1 shows the full names and the generic names of the ESCI I/O pins. The generic pin names appear in the text of this section.

**Table 14-1. Pin Name Conventions**

| Generic Pin Names | RxD | TxD |
|---|---|---|
| Full Pin Names | PTE1/RxD | PTE0/TxD |

## 14.4 Functional Description

Figure 14-3 shows the structure of the ESCI module. The ESCI allows full-duplex, asynchronous, NRZ serial communication between the MCU and remote devices, including other MCUs. The transmitter and receiver of the ESCI operate independently, although they use the same baud rate generator. During normal operation, the CPU monitors the status of the ESCI, writes the data to be transmitted, and processes received data.

The baud rate clock source for the ESCI can be selected via the configuration bit, SCIBDSRC, of the CONFIG2 register ($001E)

For reference, a summary of the ESCI module input/output registers is provided in Figure 14-4.

### 14.4.1 Data Format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in Figure 14-2.

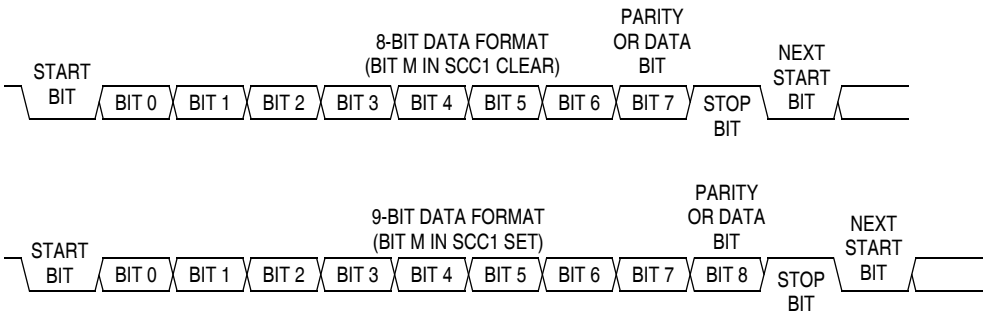



**Figure 14-2. SCI Data Formats**

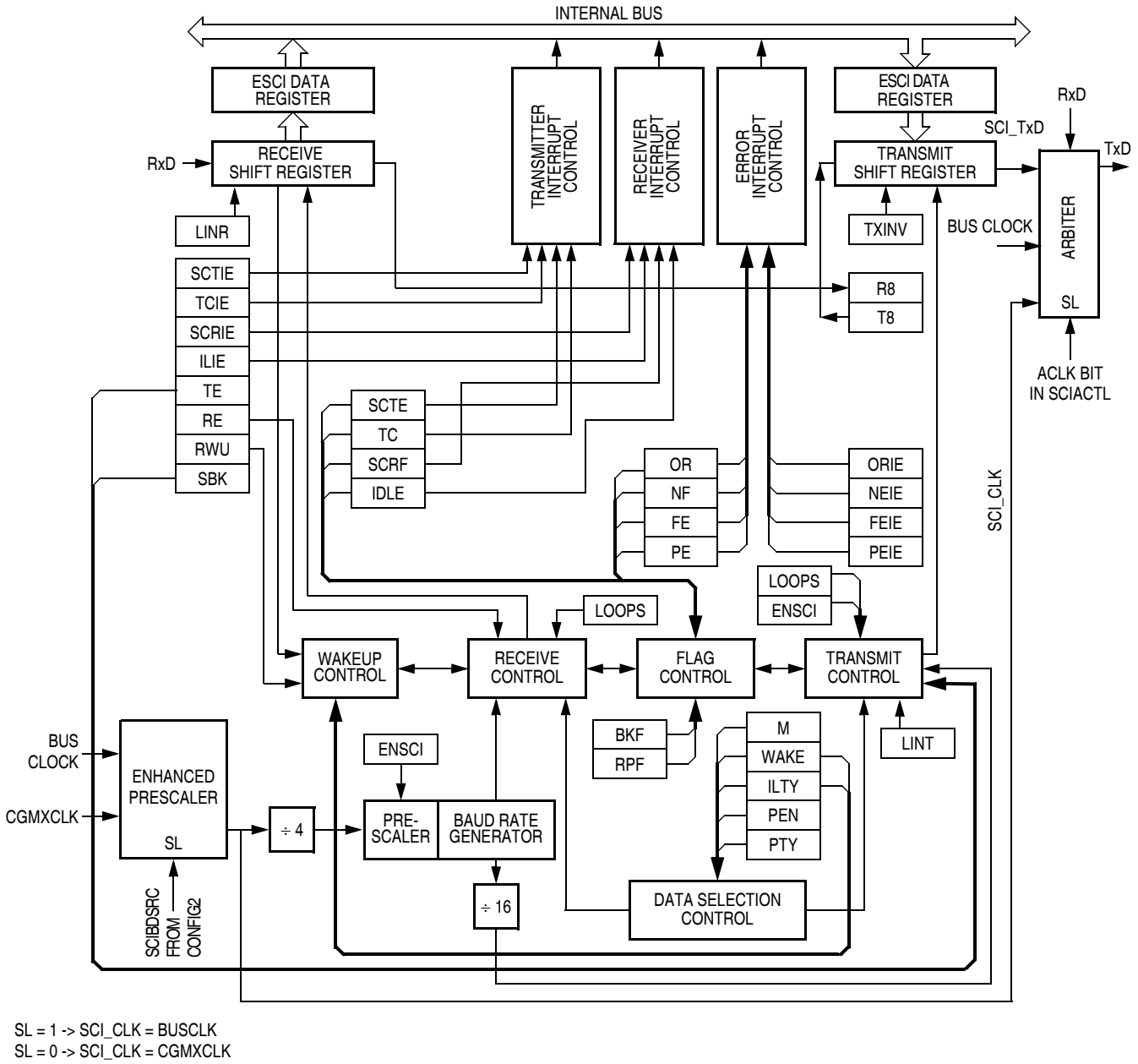



**Figure 14-3. ESCI Module Block Diagram**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0009 | ESCI Prescaler Register (SCPSC) See page 214. | Read: | PDS2 | PDS1 | PDS0 | PSSB4 | PSSB3 | PSSB2 | PSSB1 | PSSB0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000A | ESCI Arbiter Control Register (SCIACTL) See page 217. | Read: | AM1 | ALOST | AM0 | ACLK | AFIN | ARUN | AROVFL | ARD8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000B | ESCI Arbiter Data Register (SCIADAT) See page 218. | Read: | ARD7 | ARD6 | ARD5 | ARD4 | ARD3 | ARD2 | ARD1 | ARD0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0013 | ESCI Control Register 1 (SCC1) See page 204. | Read: | LOOPS | ENSCI | TXINV | M | WAKE | ILTY | PEN | PTY |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0014 | ESCI Control Register 2 (SCC2) See page 206. | Read: | SCTIE | TCIE | SCRIE | ILIE | TE | RE | RWU | SBK |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0015 | ESCI Control Register 3 (SCC3) See page 208. | Read: | R8 | T8 | R | R | ORIE | NEIE | FEIE | PEIE |
| | | Write: | | | | | | | | |
| | | Reset: | U | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0016 | ESCI Status Register 1 (SCS1) See page 209. | Read: | SCTE | TC | SCRF | IDLE | OR | NF | FE | PE |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0017 | ESCI Status Register 2 (SCS2) See page 211. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | BKF | RPF |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0018 | ESCI Data Register (SCDR) See page 212. | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0019 | ESCI Baud Rate Register (SCBR) See page 212. | Read: | LINT | LINR | SCP1 | SCP0 | R | SCR2 | SCR1 | SCR0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented  R = Reserved

**Figure 14-4. ESCI I/O Register Summary**

## 14.4.2 Transmitter

Figure 14-5 shows the structure of the SCI transmitter and the registers are summarized in Figure 14-4. The baud rate clock source for the ESCI can be selected via the configuration bit, SCIBDSRC.

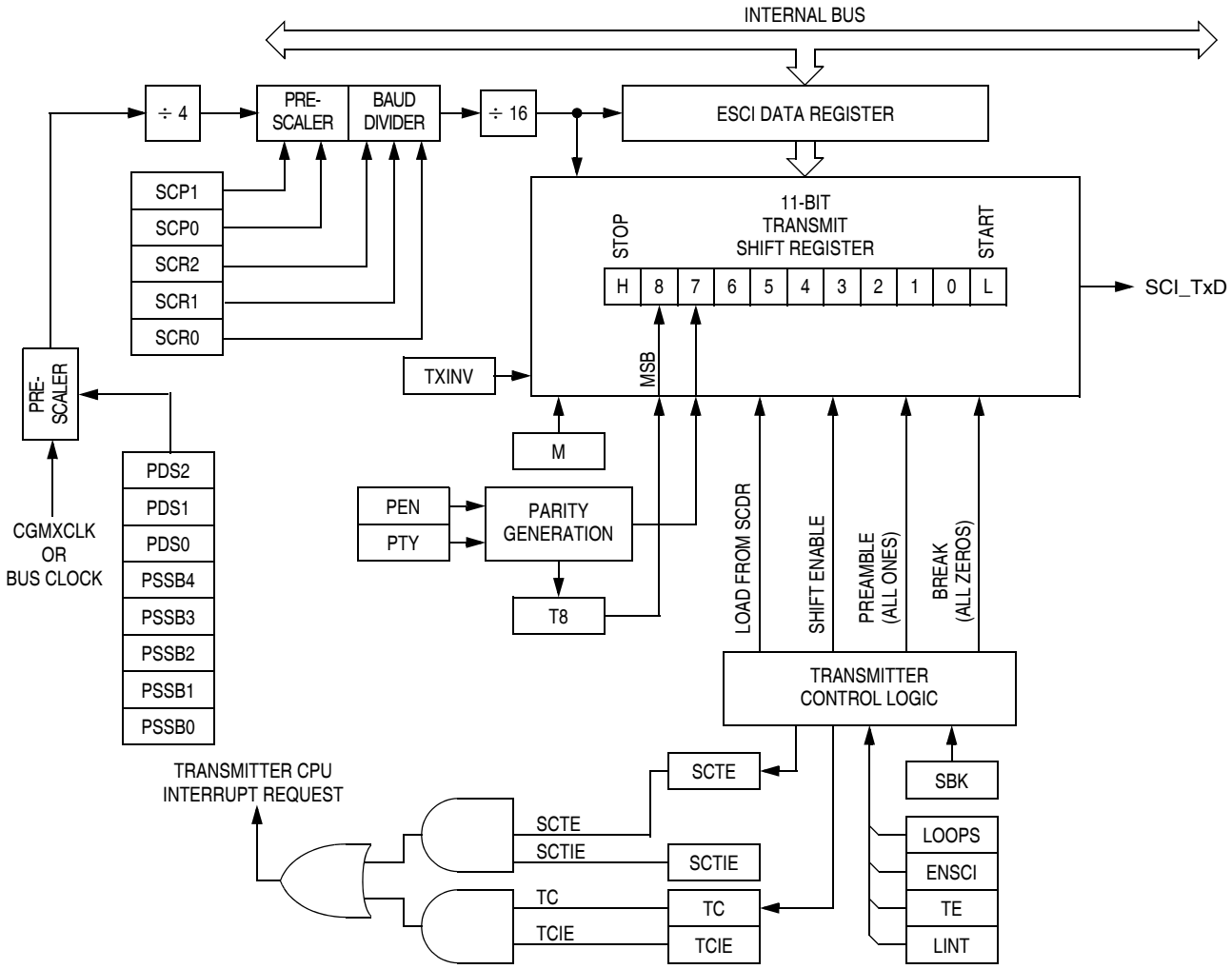



**Figure 14-5. ESCI Transmitter**

### *14.4.2.1 Character Length*

The transmitter can accommodate either 8-bit or 9-bit data. The state of the M bit in ESCI control register 1 (SCC1) determines character length. When transmitting 9-bit data, bit T8 in ESCI control register 3 (SCC3) is the ninth bit (bit 8).

### *14.4.2.2 Character Transmission*

During an ESCI transmission, the transmit shift register shifts a character out to the TxD pin. The ESCI data register (SCDR) is the write-only buffer between the internal data bus and the transmit shift register.

To initiate an ESCI transmission:

1. Enable the ESCI by writing a 1 to the enable ESCI bit (ENSCI) in ESCI control register 1 (SCC1).
2. Enable the transmitter by writing a 1 to the transmitter enable bit (TE) in ESCI control register 2 (SCC2).
3. Clear the ESCI transmitter empty bit (SCTE) by first reading ESCI status register 1 (SCS1) and then writing to the SCDR. For 9-bit data, also write the T8 bit in SCC3.
4. Repeat step 3 for each subsequent transmission.

At the start of a transmission, transmitter control logic automatically loads the transmit shift register with a preamble of 1s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A 0 start bit automatically goes into the least significant bit (LSB) position of the transmit shift register. A 1 stop bit goes into the most significant bit (MSB) position.

The ESCI transmitter empty bit, SCTE, in SCS1 becomes set when the SCDR transfers a byte to the transmit shift register. The SCTE bit indicates that the SCDR can accept new data from the internal data bus. If the ESCI transmit interrupt enable bit, SCTIE, in SCC2 is also set, the SCTE bit generates a transmitter CPU interrupt request.

When the transmit shift register is not transmitting a character, the TxD pin goes to the idle condition, high. If at any time software clears the ENSCI bit in ESCI control register 1 (SCC1), the transmitter and receiver relinquish control of the port E pins.

#### *14.4.2.3 Break Characters*

Writing a 1 to the send break bit, SBK, in SCC2 loads the transmit shift register with a break character. For TXINV = 0 (output not inverted), a transmitted break character contains all 0s and has no start, stop, or parity bit. Break character length depends on the M bit in SCC1 and the LINR bits in SCBR. As long as SBK is at 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one 1. The automatic 1 at the end of a break character guarantees the recognition of the start bit of the next character.

When LINR is cleared in SCBR, the ESCI recognizes a break character when a start bit is followed by eight or nine 0 data bits and a 0 where the stop bit should be, resulting in a total of 10 or 11 consecutive 0 data bits. When LINR is set in SCBR, the ESCI recognizes a break character when a start bit is followed by 9 or 10 0 data bits and a 0 where the stop bit should be, resulting in a total of 11 or 12 consecutive 0 data bits.

Receiving a break character has these effects on ESCI registers:

- Sets the framing error bit (FE) in SCS1
- Sets the ESCI receiver full bit (SCRF) in SCS1
- Clears the ESCI data register (SCDR)
- Clears the R8 bit in SCC3
- Sets the break flag bit (BKF) in SCS2
- May set the overrun (OR), noise flag (NF), parity error (PE), or reception in progress flag (RPF) bits

#### 14.4.2.4 Idle Characters

For TXINV = 0 (output not inverted), a transmitted idle character contains all 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in SCC1. The preamble is a synchronizing idle character that begins every transmission.

If the TE bit is cleared during a transmission, the TxD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the character currently being transmitted.

**NOTE**

*When a break sequence is followed immediately by an idle character, this SCI design exhibits a condition in which the break character length is reduced by one half bit time. In this instance, the break sequence will consist of a valid start bit, eight or nine data bits (as defined by the M bit in SCC1) of 0 and one half data bit length of 0 in the stop bit position followed immediately by the idle character. To ensure a break character of the proper length is transmitted, always queue up a byte of data to be transmitted while the final break sequence is in progress.*

*When queueing an idle character, return the TE bit to 1 before the stop bit of the current character shifts out to the TxD pin. Setting TE after the stop bit appears on TxD causes data previously written to the SCDR to be lost. A good time to toggle the TE bit for a queued idle character is when the SCTE bit becomes set and just before writing the next byte to the SCDR.*

#### 14.4.2.5 Inversion of Transmitted Output

The transmit inversion bit (TXINV) in ESCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values including idle, break, start, and stop bits, are inverted when TXINV is at 1. See 14.8.1 ESCI Control Register 1.

#### 14.4.2.6 Transmitter Interrupts

These conditions can generate CPU interrupt requests from the ESCI transmitter:

- ESCI transmitter empty (SCTE) — The SCTE bit in SCS1 indicates that the SCDR has transferred a character to the transmit shift register. SCTE can generate a transmitter CPU interrupt request. Setting the ESCI transmit interrupt enable bit, SCTIE, in SCC2 enables the SCTE bit to generate transmitter CPU interrupt requests.
- Transmission complete (TC) — The TC bit in SCS1 indicates that the transmit shift register and the SCDR are empty and that no break or idle character has been generated. The transmission complete interrupt enable bit, TCIE, in SCC2 enables the TC bit to generate transmitter CPU interrupt requests.

### 14.4.3 Receiver

Figure 14-6 shows the structure of the ESCI receiver. The receiver I/O registers are summarized in Figure 14-4.

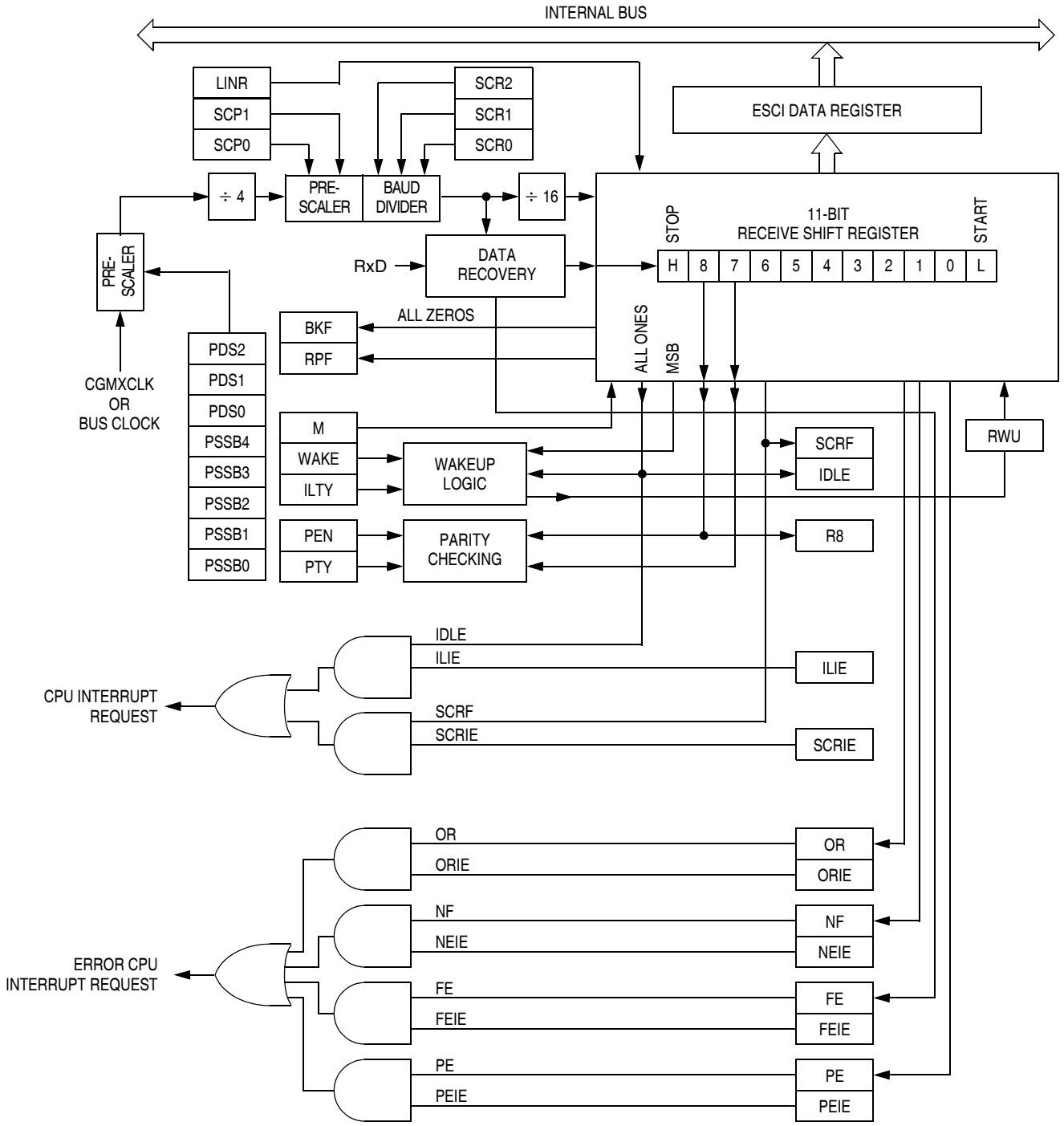



**Figure 14-6. ESCI Receiver Block Diagram**

### *14.4.3.1 Character Length*

The receiver can accommodate either 8-bit or 9-bit data. The state of the M bit in ESCI control register 1 (SCC1) determines character length. When receiving 9-bit data, bit R8 in ESCI control register 3 (SCC3) is the ninth bit (bit 8). When receiving 8-bit data, bit R8 is a copy of the eighth bit (bit 7).

### 14.4.3.2 Character Reception

During an ESCI reception, the receive shift register shifts characters in from the RxD pin. The ESCI data register (SCDR) is the read-only buffer between the internal data bus and the receive shift register.

After a complete character shifts into the receive shift register, the data portion of the character transfers to the SCDR. The ESCI receiver full bit, SCRF, in ESCI status register 1 (SCS1) becomes set, indicating that the received byte can be read. If the ESCI receive interrupt enable bit, SCRIE, in SCC2 is also set, the SCRF bit generates a receiver CPU interrupt request.

### 14.4.3.3 Data Sampling

The receiver samples the RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at these times (see Figure 14-7):

- After every start bit
- After the receiver detects a data bit change from 1 to 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid 0)

To locate the start bit, data recovery logic does an asynchronous search for a 0 preceded by three 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

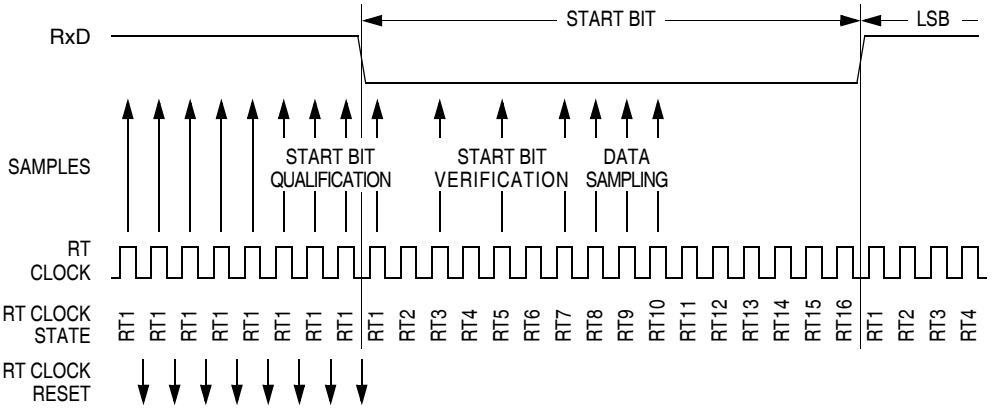



**Figure 14-7. Receiver Data Sampling**

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 14-2 summarizes the results of the start bit verification samples.

**Table 14-2. Start Bit Verification**

| RT3, RT5, and RT7 Samples | Start Bit Verification | Noise Flag |
|---|---|---|
| 000 | Yes | 0 |
| 001 | Yes | 1 |
| 010 | Yes | 1 |
| 011 | No | 0 |
| 100 | Yes | 1 |
| 101 | No | 0 |
| 110 | No | 0 |
| 111 | No | 0 |

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 14-3 summarizes the results of the data bit samples.

**Table 14-3. Data Bit Recovery**

| RT8, RT9, and RT10 Samples | Data Bit Determination | Noise Flag |
|---|---|---|
| 000 | 0 | 0 |
| 001 | 0 | 1 |
| 010 | 0 | 1 |
| 011 | 1 | 1 |
| 100 | 0 | 1 |
| 101 | 1 | 1 |
| 110 | 1 | 1 |
| 111 | 1 | 0 |

***NOTE***

*The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit.*

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 14-4 summarizes the results of the stop bit samples.

**Table 14-4. Stop Bit Recovery**

| RT8, RT9, and RT10 Samples | Framing Error Flag | Noise Flag |
|---|---|---|
| 000 | 1 | 0 |
| 001 | 1 | 1 |
| 010 | 1 | 1 |
| 011 | 0 | 1 |
| 100 | 1 | 1 |
| 101 | 0 | 1 |
| 110 | 0 | 1 |
| 111 | 0 | 0 |

#### *14.4.3.4 Framing Errors*

If the data recovery logic does not detect a 1 where the stop bit should be in an incoming character, it sets the framing error bit, FE, in SCS1. A break character also sets the FE bit because a break character has no stop bit. The FE bit is set at the same time that the SCRF bit is set.

### 14.4.3.5 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples to fall outside the actual stop bit. Then a noise error occurs. If more than one of the samples is outside the stop bit, a framing error occurs. In most applications, the baud rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming character, it resynchronizes the RT clock on any valid falling edge within the character. Resynchronization within characters corrects misalignments between transmitter bit times and receiver bit times.

**Slow Data Tolerance**

Figure 14-8 shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

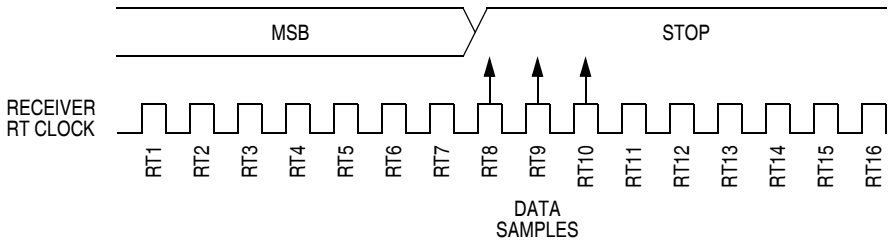



**Figure 14-8. Slow Data**

For an 8-bit character, data sampling of the stop bit takes the receiver
9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.

With the misaligned character shown in Figure 14-8, the receiver counts 154 RT cycles at the point when the count of the transmitting device is
9 bit times × 16 RT cycles + 3 RT cycles = 147 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit character with no errors is:

$$\left|\frac{154 - 147}{154}\right| \times 100 = 4.54\%$$

For a 9-bit character, data sampling of the stop bit takes the receiver
10 bit times × 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in Figure 14-8, the receiver counts 170 RT cycles at the point when the count of the transmitting device is
10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$\left|\frac{170 - 163}{170}\right| \times 100 = 4.12\%$$

### Fast Data Tolerance

Figure 14-9 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

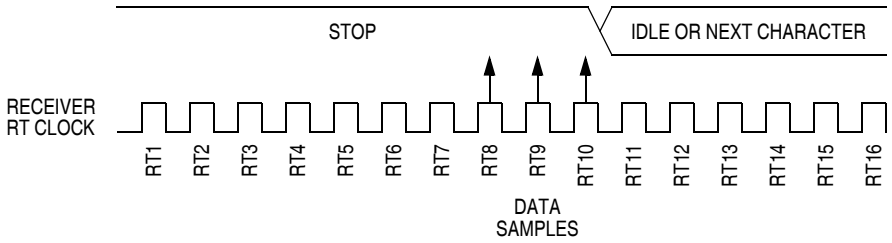



**Figure 14-9. Fast Data**

For an 8-bit character, data sampling of the stop bit takes the receiver
9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.

With the misaligned character shown in Figure 14-9, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles = 160 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is

$$\left|\frac{154 - 160}{154}\right| \times 100 = 3.90\%.$$

For a 9-bit character, data sampling of the stop bit takes the receiver
10 bit times × 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in Figure 14-9, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times × 16 RT cycles = 176 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

$$\left|\frac{170 - 176}{170}\right| \times 100 = 3.53\%.$$

#### *14.4.3.6 Receiver Wakeup*

So that the MCU can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCC2 puts the receiver into a standby state during which receiver interrupts are disabled.

Depending on the state of the WAKE bit in SCC1, either of two conditions on the RxD pin can bring the receiver out of the standby state:

1. Address mark — An address mark is a 1 in the MSB position of a received character. When the WAKE bit is set, an address mark wakes the receiver from the standby state by clearing the RWU bit. The address mark also sets the ESCI receiver full bit, SCRF. Software can then compare the character containing the address mark to the user-defined address of the receiver. If they are the same, the receiver remains awake and processes the characters that follow. If they are not the same, software can set the RWU bit and put the receiver back into the standby state.
2. Idle input line condition — When the WAKE bit is clear, an idle character on the RxD pin wakes the receiver from the standby state by clearing the RWU bit. The idle character that wakes the receiver

does not set the receiver idle bit, IDLE, or the ESCI receiver full bit, SCRF. The idle line type bit, ILTY, determines whether the receiver begins counting 1s as idle character bits after the start bit or after the stop bit.

**NOTE**

*With the WAKE bit clear, setting the RWU bit after the RxD pin has been idle will cause the receiver to wake up.*

#### 14.4.3.7 Receiver Interrupts

These sources can generate CPU interrupt requests from the ESCI receiver:

- ESCI receiver full (SCRF) — The SCRF bit in SCS1 indicates that the receive shift register has transferred a character to the SCDR. SCRF can generate a receiver CPU interrupt request. Setting the ESCI receive interrupt enable bit, SCRIE, in SCC2 enables the SCRF bit to generate receiver CPU interrupts.
- Idle input (IDLE) — The IDLE bit in SCS1 indicates that 10 or 11 consecutive 1s shifted in from the RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables the IDLE bit to generate CPU interrupt requests.

#### 14.4.3.8 Error Interrupts

These receiver error flags in SCS1 can generate CPU interrupt requests:

- Receiver overrun (OR) — The OR bit indicates that the receive shift register shifted in a new character before the previous character was read from the SCDR. The previous character remains in the SCDR, and the new character is lost. The overrun interrupt enable bit, ORIE, in SCC3 enables OR to generate ESCI error CPU interrupt requests.
- Noise flag (NF) — The NF bit is set when the ESCI detects noise on incoming data or break characters, including start, data, and stop bits. The noise error interrupt enable bit, NEIE, in SCC3 enables NF to generate ESCI error CPU interrupt requests.
- Framing error (FE) — The FE bit in SCS1 is set when a 0 occurs where the receiver expects a stop bit. The framing error interrupt enable bit, FEIE, in SCC3 enables FE to generate ESCI error CPU interrupt requests.
- Parity error (PE) — The PE bit in SCS1 is set when the ESCI detects a parity error in incoming data. The parity error interrupt enable bit, PEIE, in SCC3 enables PE to generate ESCI error CPU interrupt requests.

## 14.5 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

### 14.5.1 Wait Mode

The ESCI module remains active in wait mode. Any enabled CPU interrupt request from the ESCI module can bring the MCU out of wait mode.

If ESCI module functions are not required during wait mode, reduce power consumption by disabling the module before executing the WAIT instruction.

### 14.5.2 Stop Mode

The ESCI module is inactive in stop mode. The STOP instruction does not affect ESCI register states. ESCI module operation resumes after the MCU exits stop mode.

Because the internal clock is inactive during stop mode, entering stop mode during an ESCI transmission or reception results in invalid data.

## 14.6 ESCI During Break Module Interrupts

The BCFE bit in the break flag control register (SBFCR) enables software to clear status bits during the break state. See 20.2 Break Module (BRK).

To allow software to clear status bits during a break interrupt, write a 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a 0 to the BCFE bit. With BCFE at 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a two-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at 0. After the break, doing the second step clears the status bit.

## 14.7 I/O Signals

Port E shares two of its pins with the ESCI module. The two ESCI I/O pins are:

- PTE0/TxD — transmit data
- PTE1/RxD — receive data

### 14.7.1 PTE0/TxD (Transmit Data)

The PTE0/TxD pin is the serial data output from the ESCI transmitter. The ESCI shares the PTE0/TxD pin with port E. When the ESCI is enabled, the PTE0/TxD pin is an output regardless of the state of the DDRE0 bit in data direction register E (DDRE).

### 14.7.2 PTE1/RxD (Receive Data)

The PTE1/RxD pin is the serial data input to the ESCI receiver. The ESCI shares the PTE1/RxD pin with port E. When the ESCI is enabled, the PTE1/RxD pin is an input regardless of the state of the DDRE1 bit in data direction register E (DDRE).

## 14.8 I/O Registers

These I/O registers control and monitor ESCI operation:

- ESCI control register 1, SCC1
- ESCI control register 2, SCC2
- ESCI control register 3, SCC3
- ESCI status register 1, SCS1
- ESCI status register 2, SCS2
- ESCI data register, SCDR

- ESCI baud rate register, SCBR
- ESCI prescaler register, SCPSC
- ESCI arbiter control register, SCIACTL
- ESCI arbiter data register, SCIADAT

### 14.8.1 ESCI Control Register 1

ESCI control register 1 (SCC1):

- Enables loop mode operation
- Enables the ESCI
- Controls output polarity
- Controls character length
- Controls ESCI wakeup method
- Controls idle character detection
- Enables parity function
- Controls parity type

Address: $0013

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | LOOPS | ENSCI | TXINV | M | WAKE | ILTY | PEN | PTY |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-10. ESCI Control Register 1 (SCC1)**

**LOOPS — Loop Mode Select Bit**

This read/write bit enables loop mode operation. In loop mode the RxD pin is disconnected from the ESCI, and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use loop mode. Reset clears the LOOPS bit.

1 = Loop mode enabled
0 = Normal operation enabled

**ENSCI — Enable ESCI Bit**

This read/write bit enables the ESCI and the ESCI baud rate generator. Clearing ENSCI sets the SCTE and TC bits in ESCI status register 1 and disables transmitter interrupts. Reset clears the ENSCI bit.

1 = ESCI enabled
0 = ESCI disabled

**TXINV — Transmit Inversion Bit**

This read/write bit reverses the polarity of transmitted data. Reset clears the TXINV bit.

1 = Transmitter output inverted
0 = Transmitter output not inverted

***NOTE***

*Setting the TXINV bit inverts all transmitted values including idle, break, start, and stop bits.*

### M — Mode (Character Length) Bit

This read/write bit determines whether ESCI characters are eight or nine bits long (See Table 14-5).The ninth bit can serve as a receiver wakeup signal or as a parity bit. Reset clears the M bit.

1 = 9-bit ESCI characters
0 = 8-bit ESCI characters

**Table 14-5. Character Format Selection**

| Control Bits | | Character Format | | | | |
|---|---|---|---|---|---|---|
| M | PEN:PTY | Start Bits | Data Bits | Parity | Stop Bits | Character Length |
| 0 | 0 X | 1 | 8 | None | 1 | 10 bits |
| 1 | 0 X | 1 | 9 | None | 1 | 11 bits |
| 0 | 1 0 | 1 | 7 | Even | 1 | 10 bits |
| 0 | 1 1 | 1 | 7 | Odd | 1 | 10 bits |
| 1 | 1 0 | 1 | 8 | Even | 1 | 11 bits |
| 1 | 1 1 | 1 | 8 | Odd | 1 | 11 bits |

### WAKE — Wakeup Condition Bit

This read/write bit determines which condition wakes up the ESCI: a 1 (address mark) in the MSB position of a received character or an idle condition on the RxD pin. Reset clears the WAKE bit.

1 = Address mark wakeup
0 = Idle line wakeup

### ILTY — Idle Line Type Bit

This read/write bit determines when the ESCI starts counting 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. Reset clears the ILTY bit.

1 = Idle character bit count begins after stop bit
0 = Idle character bit count begins after start bit

### PEN — Parity Enable Bit

This read/write bit enables the ESCI parity function (see Table 14-5). When enabled, the parity function inserts a parity bit in the MSB position (see Table 14-3). Reset clears the PEN bit.

1 = Parity function enabled
0 = Parity function disabled

### PTY — Parity Bit

This read/write bit determines whether the ESCI generates and checks for odd parity or even parity (see Table 14-5). Reset clears the PTY bit.

1 = Odd parity
0 = Even parity

***NOTE***

*Changing the PTY bit in the middle of a transmission or reception can generate a parity error.*

### 14.8.2 ESCI Control Register 2

ESCI control register 2 (SCC2):

- Enables these CPU interrupt requests:
  - SCTE bit to generate transmitter CPU interrupt requests
  - TC bit to generate transmitter CPU interrupt requests
  - SCRF bit to generate receiver CPU interrupt requests
  - IDLE bit to generate receiver CPU interrupt requests
- Enables the transmitter
- Enables the receiver
- Enables ESCI wakeup
- Transmits ESCI break characters

Address: $0014

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | SCTIE | TCIE | SCRIE | ILIE | TE | RE | RWU | SBK |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-11. ESCI Control Register 2 (SCC2)**

**SCTIE — ESCI Transmit Interrupt Enable Bit**

This read/write bit enables the SCTE bit to generate ESCI transmitter CPU interrupt requests. Setting the SCTIE bit in SCC2 enables the SCTE bit to generate CPU interrupt requests. Reset clears the SCTIE bit.

1 = SCTE enabled to generate CPU interrupt
0 = SCTE not enabled to generate CPU interrupt

**TCIE — Transmission Complete Interrupt Enable Bit**

This read/write bit enables the TC bit to generate ESCI transmitter CPU interrupt requests. Reset clears the TCIE bit.

1 = TC enabled to generate CPU interrupt requests
0 = TC not enabled to generate CPU interrupt requests

**SCRIE — ESCI Receive Interrupt Enable Bit**

This read/write bit enables the SCRF bit to generate ESCI receiver CPU interrupt requests. Setting the SCRIE bit in SCC2 enables the SCRF bit to generate CPU interrupt requests. Reset clears the SCRIE bit.

1 = SCRF enabled to generate CPU interrupt
0 = SCRF not enabled to generate CPU interrupt

**ILIE — Idle Line Interrupt Enable Bit**

This read/write bit enables the IDLE bit to generate ESCI receiver CPU interrupt requests. Reset clears the ILIE bit.

1 = IDLE enabled to generate CPU interrupt requests
0 = IDLE not enabled to generate CPU interrupt requests

**TE — Transmitter Enable Bit**

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 1s from the transmit shift register to the TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the TxD returns to the idle condition (high). Clearing and then setting TE during a transmission queues an idle character to be sent after the character currently being transmitted. Reset clears the TE bit.

1 = Transmitter enabled
0 = Transmitter disabled

***NOTE***

*Writing to the TE bit is not allowed when the enable ESCI bit (ENSCI) is clear. ENSCI is in ESCI control register 1.*

**RE — Receiver Enable Bit**

Setting this read/write bit enables the receiver. Clearing the RE bit disables the receiver but does not affect receiver interrupt flag bits. Reset clears the RE bit.

1 = Receiver enabled
0 = Receiver disabled

***NOTE***

*Writing to the RE bit is not allowed when the enable ESCI bit (ENSCI) is clear. ENSCI is in ESCI control register 1.*

**RWU — Receiver Wakeup Bit**

This read/write bit puts the receiver in a standby state during which receiver interrupts are disabled. The WAKE bit in SCC1 determines whether an idle input or an address mark brings the receiver out of the standby state and clears the RWU bit. Reset clears the RWU bit.

1 = Standby state
0 = Normal operation

**SBK — Send Break Bit**

Setting and then clearing this read/write bit transmits a break character followed by a 1. The 1 after the break character guarantees recognition of a valid start bit. If SBK remains set, the transmitter continuously transmits break characters with no 1s between them. Reset clears the SBK bit.

1 = Transmit break characters
0 = No break characters being transmitted

***NOTE***

*Do not toggle the SBK bit immediately after setting the SCTE bit. Toggling SBK before the preamble begins causes the ESCI to send a break character instead of a preamble.*

### 14.8.3 ESCI Control Register 3

ESCI control register 3 (SCC3):

- Stores the ninth ESCI data bit received and the ninth ESCI data bit to be transmitted.
- Enables these interrupts:
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error

Address: $0015

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | R8 | T8 | R | R | ORIE | NEIE | FEIE | PEIE |
| Write: | | | | | | | | |
| Reset: | U | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented R = Reserved U = Unaffected

**Figure 14-12. ESCI Control Register 3 (SCC3)**

**R8 — Received Bit 8**

When the ESCI is receiving 9-bit characters, R8 is the read-only ninth bit (bit 8) of the received character. R8 is received at the same time that the SCDR receives the other 8 bits.

When the ESCI is receiving 8-bit characters, R8 is a copy of the eighth bit (bit 7). Reset has no effect on the R8 bit.

**T8 — Transmitted Bit 8**

When the ESCI is transmitting 9-bit characters, T8 is the read/write ninth bit (bit 8) of the transmitted character. T8 is loaded into the transmit shift register at the same time that the SCDR is loaded into the transmit shift register. Reset clears the T8 bit.

**ORIE — Receiver Overrun Interrupt Enable Bit**

This read/write bit enables ESCI error CPU interrupt requests generated by the receiver overrun bit, OR. Reset clears ORIE.

1 = ESCI error CPU interrupt requests from OR bit enabled
0 = ESCI error CPU interrupt requests from OR bit disabled

**NEIE — Receiver Noise Error Interrupt Enable Bit**

This read/write bit enables ESCI error CPU interrupt requests generated by the noise error bit, NE. Reset clears NEIE.

1 = ESCI error CPU interrupt requests from NE bit enabled
0 = ESCI error CPU interrupt requests from NE bit disabled

**FEIE — Receiver Framing Error Interrupt Enable Bit**

This read/write bit enables ESCI error CPU interrupt requests generated by the framing error bit, FE. Reset clears FEIE.

1 = ESCI error CPU interrupt requests from FE bit enabled
0 = ESCI error CPU interrupt requests from FE bit disabled

**PEIE — Receiver Parity Error Interrupt Enable Bit**

This read/write bit enables ESCI receiver CPU interrupt requests generated by the parity error bit, PE. Reset clears PEIE.

1 = ESCI error CPU interrupt requests from PE bit enabled
0 = ESCI error CPU interrupt requests from PE bit disabled

### 14.8.4 ESCI Status Register 1

ESCI status register 1 (SCS1) contains flags to signal these conditions:

- Transfer of SCDR data to transmit shift register complete
- Transmission complete
- Transfer of receive shift register data to SCDR complete
- Receiver input idle
- Receiver overrun
- Noisy data
- Framing error
- Parity error

Address: $0016

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | SCTE | TC | SCRF | IDLE | OR | NF | FE | PE |
| Write: | | | | | | | | |
| Reset: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 14-13. ESCI Status Register 1 (SCS1)**

**SCTE — ESCI Transmitter Empty Bit**

This clearable, read-only bit is set when the SCDR transfers a character to the transmit shift register. SCTE can generate an ESCI transmitter CPU interrupt request. When the SCTIE bit in SCC2 is set, SCTE generates an ESCI transmitter CPU interrupt request. In normal operation, clear the SCTE bit by reading SCS1 with SCTE set and then writing to SCDR. Reset sets the SCTE bit.

1 = SCDR data transferred to transmit shift register
0 = SCDR data not transferred to transmit shift register

**TC — Transmission Complete Bit**

This read-only bit is set when the SCTE bit is set, and no data, preamble, or break character is being transmitted. TC generates an ESCI transmitter CPU interrupt request if the TCIE bit in SCC2 is also set. TC is cleared automatically when data, preamble, or break is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queueing data, preamble, and break and the transmission actually starting. Reset sets the TC bit.

1 = No transmission in progress
0 = Transmission in progress

**SCRF — ESCI Receiver Full Bit**

This clearable, read-only bit is set when the data in the receive shift register transfers to the ESCI data register. SCRF can generate an ESCI receiver CPU interrupt request. When the SCRIE bit in SCC2 is set the SCRF generates a CPU interrupt request. In normal operation, clear the SCRF bit by reading SCS1 with SCRF set and then reading the SCDR. Reset clears SCRF.

1 = Received data available in SCDR
0 = Data not available in SCDR

### IDLE — Receiver Idle Bit

This clearable, read-only bit is set when 10 or 11 consecutive 1s appear on the receiver input. IDLE generates an ESCI receiver CPU interrupt request if the ILIE bit in SCC2 is also set. Clear the IDLE bit by reading SCS1 with IDLE set and then reading the SCDR. After the receiver is enabled, it must receive a valid character that sets the SCRF bit before an idle condition can set the IDLE bit. Also, after the IDLE bit has been cleared, a valid character must again set the SCRF bit before an idle condition can set the IDLE bit. Reset clears the IDLE bit.

1 = Receiver input idle
0 = Receiver input active (or idle since the IDLE bit was cleared)

### OR — Receiver Overrun Bit

This clearable, read-only bit is set when software fails to read the SCDR before the receive shift register receives the next character. The OR bit generates an ESCI error CPU interrupt request if the ORIE bit in SCC3 is also set. The data in the shift register is lost, but the data already in the SCDR is not affected. Clear the OR bit by reading SCS1 with OR set and then reading the SCDR. Reset clears the OR bit.

1 = Receive shift register full and SCRF = 1
0 = No receiver overrun

Software latency may allow an overrun to occur between reads of SCS1 and SCDR in the flag-clearing sequence. Figure 14-14 shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

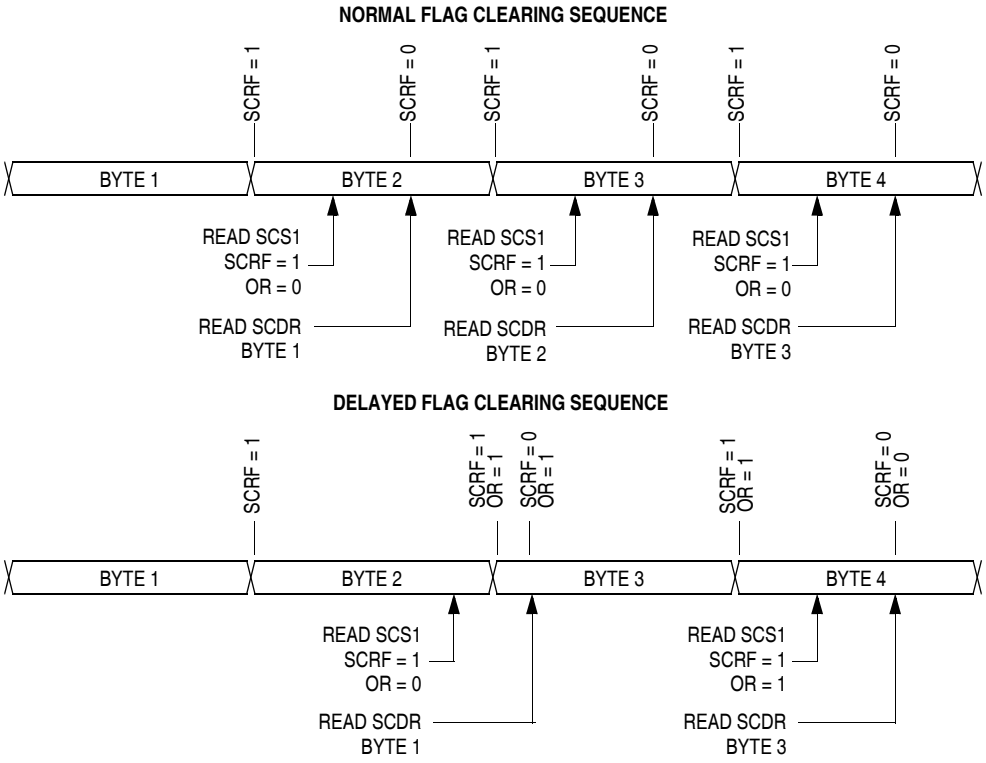



**Figure 14-14. Flag Clearing Sequence**

In applications that are subject to software latency or in which it is important to know which byte is lost due to an overrun, the flag-clearing routine can check the OR bit in a second read of SCS1 after reading the data register.

**NF — Receiver Noise Flag Bit**

This clearable, read-only bit is set when the ESCI detects noise on the RxD pin. NF generates an NF CPU interrupt request if the NEIE bit in SCC3 is also set. Clear the NF bit by reading SCS1 and then reading the SCDR. Reset clears the NF bit.

1 = Noise detected
0 = No noise detected

**FE — Receiver Framing Error Bit**

This clearable, read-only bit is set when a 0 is accepted as the stop bit. FE generates an ESCI error CPU interrupt request if the FEIE bit in SCC3 also is set. Clear the FE bit by reading SCS1 with FE set and then reading the SCDR. Reset clears the FE bit.

1 = Framing error detected
0 = No framing error detected

**PE — Receiver Parity Error Bit**

This clearable, read-only bit is set when the ESCI detects a parity error in incoming data. PE generates a PE CPU interrupt request if the PEIE bit in SCC3 is also set. Clear the PE bit by reading SCS1 with PE set and then reading the SCDR. Reset clears the PE bit.

1 = Parity error detected
0 = No parity error detected

### 14.8.5 ESCI Status Register 2

ESCI status register 2 (SCS2) contains flags to signal these conditions:

- Break character detected
- Incoming data

Address: $0017

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | 0 | 0 | BKF | RPF |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 14-15. ESCI Status Register 2 (SCS2)**

**BKF — Break Flag Bit**

This clearable, read-only bit is set when the ESCI detects a break character on the RxD pin. In SCS1, the FE and SCRF bits are also set. In 9-bit character transmissions, the R8 bit in SCC3 is cleared. BKF does not generate a CPU interrupt request. Clear BKF by reading SCS2 with BKF set and then reading the SCDR. Once cleared, BKF can become set again only after 1s again appear on the RxD pin followed by another break character. Reset clears the BKF bit.

1 = Break character detected
0 = No break character detected

### RPF — Reception in Progress Flag Bit

This read-only bit is set when the receiver detects a 0 during the RT1 time period of the start bit search. RPF does not generate an interrupt request. RPF is reset after the receiver detects false start bits (usually from noise or a baud rate mismatch), or when the receiver detects an idle character. Polling RPF before disabling the ESCI module or entering stop mode can show whether a reception is in progress.

1 = Reception in progress
0 = No reception in progress

## 14.8.6 ESCI Data Register

The ESCI data register (SCDR) is the buffer between the internal data bus and the receive and transmit shift registers. Reset has no effect on data in the ESCI data register.

Address: $0018

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| Reset: | Unaffected by reset | | | | | | | |

**Figure 14-16. ESCI Data Register (SCDR)**

### R7/T7:R0/T0 — Receive/Transmit Data Bits

Reading address $0018 accesses the read-only received data bits, R7:R0. Writing to address $0018 writes the data to be transmitted, T7:T0. Reset has no effect on the ESCI data register.

***NOTE***

*Do not use read-modify-write instructions on the ESCI data register.*

## 14.8.7 ESCI Baud Rate Register

The ESCI baud rate register (SCBR) together with the ESCI prescaler register selects the baud rate for both the receiver and the transmitter.

***NOTE***

*There are two prescalers available to adjust the baud rate. One in the ESCI baud rate register and one in the ESCI prescaler register.*

Address: $0019

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | LINT | LINR | SCP1 | SCP0 | R | SCR2 | SCR1 | SCR0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 14-17. ESCI Baud Rate Register (SCBR)**

### LINT — LIN Transmit Enable

This read/write bit selects the enhanced ESCI features for the local interconnect network (LIN) protocol as shown in Table 14-6. Reset clears LINT.

### LINR — LIN Receiver Bits

This read/write bit selects the enhanced ESCI features for the local interconnect network (LIN) protocol as shown in Table 14-6. Reset clears LINR.

**Table 14-6. ESCI LIN Control Bits**

| LINT | LINR | M | Functionality |
|---|---|---|---|
| 0 | 0 | X | Normal ESCI functionality |
| 0 | 1 | 0 | 11-bit break detect enabled for LIN receiver |
| 0 | 1 | 1 | 12-bit break detect enabled for LIN receiver |
| 1 | 0 | 0 | 13-bit generation enabled for LIN transmitter |
| 1 | 0 | 1 | 14-bit generation enabled for LIN transmitter |
| 1 | 1 | 0 | 11-bit break detect/13-bit generation enabled for LIN |
| 1 | 1 | 1 | 12-bit break detect/14-bit generation enabled for LIN |

In LIN (version 1.2) systems, the master node transmits a break character which will appear as 11.05–14.95 dominant bits to the slave node. A data character of 0x00 sent from the master might appear as 7.65–10.35 dominant bit times. This is due to the oscillator tolerance requirement that the slave node must be within ±15% of the master node's oscillator. Since a slave node cannot know if it is running faster or slower than the master node (prior to synchronization), the LINR bit allows the slave node to differentiate between a 0x00 character of 10.35 bits and a break character of 11.05 bits. The break symbol length must be verified in software in any case, but the LINR bit serves as a filter, preventing false detections of break characters that are really 0x00 data characters.

### SCP1 and SCP0 — ESCI Baud Rate Register Prescaler Bits

These read/write bits select the baud rate register prescaler divisor as shown in Table 14-7. Reset clears SCP1 and SCP0.

**Table 14-7. ESCI Baud Rate Prescaling**

| SCP[1:0] | Baud Rate Register Prescaler Divisor (BPD) |
|---|---|
| 0 0 | 1 |
| 0 1 | 3 |
| 1 0 | 4 |
| 1 1 | 13 |

### SCR2–SCR0 — ESCI Baud Rate Select Bits

These read/write bits select the ESCI baud rate divisor as shown in Table 14-8. Reset clears SCR2–SCR0.

**Table 14-8. ESCI Baud Rate Selection**

| SCR[2:1:0] | Baud Rate Divisor (BD) |
|---|---|
| 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 |

### 14.8.8 ESCI Prescaler Register

The ESCI prescaler register (SCPSC) together with the ESCI baud rate register selects the baud rate for both the receiver and the transmitter.

***NOTE***

*There are two prescalers available to adjust the baud rate. One in the ESCI baud rate register and one in the ESCI prescaler register.*

Address: $0009

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | PDS2 | PDS1 | PDS0 | PSSB4 | PSSB3 | PSSB2 | PSSB1 | PSSB0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14-18. ESCI Prescaler Register (SCPSC)**

**PDS2–PDS0 — Prescaler Divisor Select Bits**

These read/write bits select the prescaler divisor as shown in Table 14-9. Reset clears PDS2–PDS0.

***NOTE***

*The setting of '000' will bypass not only this prescaler but also the prescaler divisor fine adjust (PDFA). It is not recommended to bypass the prescaler while ENSCI is set, because the switching is not glitch free.*

**Table 14-9. ESCI Prescaler Division Ratio**

| PDS[2:1:0] | Prescaler Divisor (PD) |
|---|---|
| 0 0 0 | Bypass this prescaler |
| 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 |

**PSSB4–PSSB0 — Clock Insertion Select Bits**

These read/write bits select the number of clocks inserted in each 32 output cycle frame to achieve more timing resolution on the **average** prescaler frequency as shown in Table 14-10. Reset clears PSSB4–PSSB0.

**Table 14-10. ESCI Prescaler Divisor Fine Adjust**

| PSSB[4:3:2:1:0] | Prescaler Divisor Fine Adjust (PDFA) |
|---|---|
| 0 0 0 0 0 | 0/32 = 0 |
| 0 0 0 0 1 | 1/32 = 0.03125 |
| 0 0 0 1 0 | 2/32 = 0.0625 |
| 0 0 0 1 1 | 3/32 = 0.09375 |
| 0 0 1 0 0 | 4/32 = 0.125 |

**Table 14-10. ESCI Prescaler Divisor Fine Adjust (Continued)**

| PSSB[4:3:2:1:0] | Prescaler Divisor Fine Adjust (PDFA) |
|---|---|
| 0 0 1 0 1 | 5/32 = 0.15625 |
| 0 0 1 1 0 | 6/32 = 0.1875 |
| 0 0 1 1 1 | 7/32 = 0.21875 |
| 0 1 0 0 0 | 8/32 = 0.25 |
| 0 1 0 0 1 | 9/32 = 0.28125 |
| 0 1 0 1 0 | 10/32 = 0.3125 |
| 0 1 0 1 1 | 11/32 = 0.34375 |
| 0 1 1 0 0 | 12/32 = 0.375 |
| 0 1 1 0 1 | 13/32 = 0.40625 |
| 0 1 1 1 0 | 14/32 = 0.4375 |
| 0 1 1 1 1 | 15/32 = 0.46875 |
| 1 0 0 0 0 | 16/32 = 0.5 |
| 1 0 0 0 1 | 17/32 = 0.53125 |
| 1 0 0 1 0 | 18/32 = 0.5625 |
| 1 0 0 1 1 | 19/32 = 0.59375 |
| 1 0 1 0 0 | 20/32 = 0.625 |
| 1 0 1 0 1 | 21/32 = 0.65625 |
| 1 0 1 1 0 | 22/32 = 0.6875 |
| 1 0 1 1 1 | 23/32 = 0.71875 |
| 1 1 0 0 0 | 24/32 = 0.75 |
| 1 1 0 0 1 | 25/32 = 0.78125 |
| 1 1 0 1 0 | 26/32 = 0.8125 |
| 1 1 0 1 1 | 27/32 = 0.84375 |
| 1 1 1 0 0 | 28/32 = 0.875 |
| 1 1 1 0 1 | 29/32 = 0.90625 |
| 1 1 1 1 0 | 30/32 = 0.9375 |
| 1 1 1 1 1 | 31/32 = 0.96875 |

Use the following formula to calculate the ESCI baud rate:

$$\text{Baud rate} = \frac{\text{Frequency of the SCI clock source}}{64 \times \text{BPD} \times \text{BD} \times (\text{PD} + \text{PDFA})}$$

where:

Frequency of the SCI clock source = $f_{Bus}$ or CGMXCLK (selected by SCIBDSRC in the CONFIG2 register)
BPD = Baud rate register prescaler divisor
BD = Baud rate divisor
PD = Prescaler divisor
PDFA = Prescaler divisor fine adjust

Table 14-11 shows the ESCI baud rates that can be generated with a 4.9152-MHz bus frequency.

## Table 14-11. ESCI Baud Rate Selection Examples

| PDS[2:1:0] | PSSB[4:3:2:1:0] | SCP[1:0] | Prescaler Divisor (BPD) | SCR[2:1:0] | Baud Rate Divisor (BD) | Baud Rate ($f_{Bus}$= 4.9152 MHz) |
|---|---|---|---|---|---|---|
| 0 0 0 | X X X X X | 0 0 | 1 | 0 0 0 | 1 | 76,800 |
| 1 1 1 | 0 0 0 0 0 | 0 0 | 1 | 0 0 0 | 1 | 9600 |
| 1 1 1 | 0 0 0 0 1 | 0 0 | 1 | 0 0 0 | 1 | 9562.65 |
| 1 1 1 | 0 0 0 1 0 | 0 0 | 1 | 0 0 0 | 1 | 9525.58 |
| 1 1 1 | 1 1 1 1 1 | 0 0 | 1 | 0 0 0 | 1 | 8563.07 |
| 0 0 0 | X X X X X | 0 0 | 1 | 0 0 1 | 2 | 38,400 |
| 0 0 0 | X X X X X | 0 0 | 1 | 0 1 0 | 4 | 19,200 |
| 0 0 0 | X X X X X | 0 0 | 1 | 0 1 1 | 8 | 9600 |
| 0 0 0 | X X X X X | 0 0 | 1 | 1 0 0 | 16 | 4800 |
| 0 0 0 | X X X X X | 0 0 | 1 | 1 0 1 | 32 | 2400 |
| 0 0 0 | X X X X X | 0 0 | 1 | 1 1 0 | 64 | 1200 |
| 0 0 0 | X X X X X | 0 0 | 1 | 1 1 1 | 128 | 600 |
| 0 0 0 | X X X X X | 0 1 | 3 | 0 0 0 | 1 | 25,600 |
| 0 0 0 | X X X X X | 0 1 | 3 | 0 0 1 | 2 | 12,800 |
| 0 0 0 | X X X X X | 0 1 | 3 | 0 1 0 | 4 | 6400 |
| 0 0 0 | X X X X X | 0 1 | 3 | 0 1 1 | 8 | 3200 |
| 0 0 0 | X X X X X | 0 1 | 3 | 1 0 0 | 16 | 1600 |
| 0 0 0 | X X X X X | 0 1 | 3 | 1 0 1 | 32 | 800 |
| 0 0 0 | X X X X X | 0 1 | 3 | 1 1 0 | 64 | 400 |
| 0 0 0 | X X X X X | 0 1 | 3 | 1 1 1 | 128 | 200 |
| 0 0 0 | X X X X X | 1 0 | 4 | 0 0 0 | 1 | 19,200 |
| 0 0 0 | X X X X X | 1 0 | 4 | 0 0 1 | 2 | 9600 |
| 0 0 0 | X X X X X | 1 0 | 4 | 0 1 0 | 4 | 4800 |
| 0 0 0 | X X X X X | 1 0 | 4 | 0 1 1 | 8 | 2400 |
| 0 0 0 | X X X X X | 1 0 | 4 | 1 0 0 | 16 | 1200 |
| 0 0 0 | X X X X X | 1 0 | 4 | 1 0 1 | 32 | 600 |
| 0 0 0 | X X X X X | 1 0 | 4 | 1 1 0 | 64 | 300 |
| 0 0 0 | X X X X X | 1 0 | 4 | 1 1 1 | 128 | 150 |
| 0 0 0 | X X X X X | 1 1 | 13 | 0 0 0 | 1 | 5908 |
| 0 0 0 | X X X X X | 1 1 | 13 | 0 0 1 | 2 | 2954 |
| 0 0 0 | X X X X X | 1 1 | 13 | 0 1 0 | 4 | 1477 |
| 0 0 0 | X X X X X | 1 1 | 13 | 0 1 1 | 8 | 739 |
| 0 0 0 | X X X X X | 1 1 | 13 | 1 0 0 | 16 | 369 |
| 0 0 0 | X X X X X | 1 1 | 13 | 1 0 1 | 32 | 185 |
| 0 0 0 | X X X X X | 1 1 | 13 | 1 1 0 | 64 | 92 |
| 0 0 0 | X X X X X | 1 1 | 13 | 1 1 1 | 128 | 46 |

## 14.9 ESCI Arbiter

The ESCI module comprises an arbiter module designed to support software for communication tasks as bus arbitration, baud rate recovery and break time detection. The arbiter module consists of an 9-bit counter with 1-bit overflow and control logic. The CPU can control operation mode via the ESCI arbiter control register (SCIACTL).

### 14.9.1 ESCI Arbiter Control Register

Address: $000A

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | AM1 | ALOST | AM0 | ACLK | AFIN | ARUN | AROVFL | ARD8 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 14-19. ESCI Arbiter Control Register (SCIACTL)**

**AM1 and AM0 — Arbiter Mode Select Bits**

These read/write bits select the mode of the arbiter module as shown in Table 14-12. Reset clears AM1 and AM0.

**Table 14-12. ESCI Arbiter Selectable Modes**

| AM[1:0] | ESCI Arbiter Mode |
|---|---|
| 0 0 | Idle / counter reset |
| 0 1 | Bit time measurement |
| 1 0 | Bus arbitration |
| 1 1 | Reserved / do not use |

**ALOST — Arbitration Lost Flag**

This read-only bit indicates loss of arbitration. Clear ALOST by writing a 0 to AM1. Reset clears ALOST.

**ACLK — Arbiter Counter Clock Select Bit**

This read/write bit selects the arbiter counter clock source. Reset clears ACLK.

1 = Arbiter counter is clocked with one half of the ESCI input clock generated by the ESCI prescaler
0 = Arbiter counter is clocked with the bus clock divided by four

***NOTE***

*For ACLK = 1, the arbiter input clock is driven from the ESCI prescaler. The prescaler can be clocked by either the bus clock or CGMXCLK depending on the state of the SCIBDSRC bit in CONFIG2.*

**AFIN— Arbiter Bit Time Measurement Finish Flag**

This read-only bit indicates bit time measurement has finished. Clear AFIN by writing any value to SCIACTL. Reset clears AFIN.

1 = Bit time measurement has finished
0 = Bit time measurement not yet finished

**ARUN— Arbiter Counter Running Flag**

This read-only bit indicates the arbiter counter is running. Reset clears ARUN.

1 = Arbiter counter running
0 = Arbiter counter stopped

**AROVFL— Arbiter Counter Overflow Bit**

This read-only bit indicates an arbiter counter overflow. Clear AROVFL by writing any value to SCIACTL. Writing 0s to AM1 and AM0 resets the counter keeps it in this idle state. Reset clears AROVFL.

1 = Arbiter counter overflow has occurred
0 = No arbiter counter overflow has occurred

**ARD8— Arbiter Counter MSB**

This read-only bit is the MSB of the 9-bit arbiter counter. Clear ARD8 by writing any value to SCIACTL. Reset clears ARD8.

### 14.9.2 ESCI Arbiter Data Register

Address: $000B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | ARD7 | ARD6 | ARD5 | ARD4 | ARD3 | ARD2 | ARD1 | ARD0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 14-20. ESCI Arbiter Data Register (SCIADAT)**

**ARD7–ARD0 — Arbiter Least Significant Counter Bits**

These read-only bits are the eight LSBs of the 9-bit arbiter counter. Clear ARD7–ARD0 by writing any value to SCIACTL. Writing 0s to AM1 and AM0 permanently resets the counter and keeps it in this idle state. Reset clears ARD7–ARD0.

### 14.9.3 Bit Time Measurement

Two bit time measurement modes, described here, are available according to the state of ACLK.

1. ACLK = 0 — The counter is clocked with the bus clock divided by four. The counter is started when a falling edge on the RxD pin is detected. The counter will be stopped on the next falling edge. ARUN is set while the counter is running, AFIN is set on the second falling edge on RxD (for instance, the counter is stopped). This mode is used to recover the received baud rate. See Figure 14-21.
2. ACLK = 1 — The counter is clocked with one half of the ESCI input clock generated by the ESCI prescaler. The counter is started when a 0 is detected on RxD (see Figure 14-22). A 0 on RxD on enabling the bit time measurement with ACLK = 1 leads to immediate start of the counter (see Figure 14-23). The counter will be stopped on the next rising edge of RxD. This mode is used to measure the length of a received break.

### 14.9.4 Arbitration Mode

If AM[1:0] is set to 10, the arbiter module operates in arbitration mode. On every rising edge of SCI_TxD (output of the ESCI module, internal chip signal), the counter is started. When the counter reaches $38 (ACLK = 0) or $08 (ACLK = 1), RxD is statically sensed. If in this case, RxD is sensed low (for example,

another bus is driving the bus dominant) ALOST is set. As long as ALOST is set, the TxD pin is forced to 1, resulting in a seized transmission.

If SCI_TxD senses 0 without having sensed a 0 before on RxD, the counter will be reset, arbitration operation will be restarted after the next rising edge of SCI_TxD.

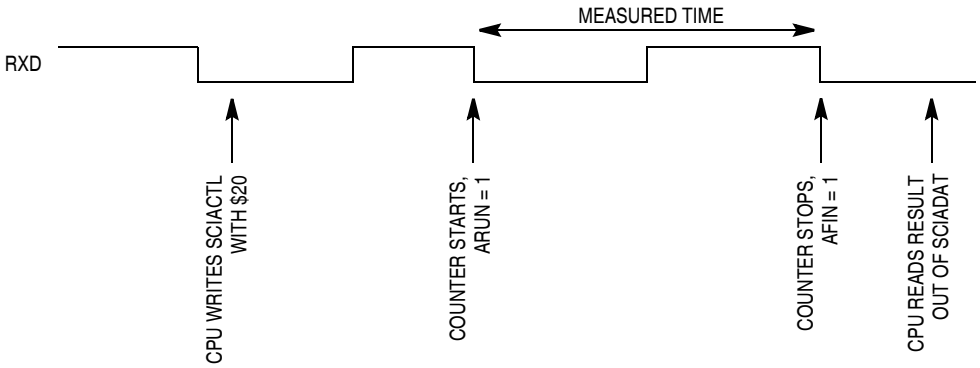



**Figure 14-21. Bit Time Measurement with ACLK = 0**

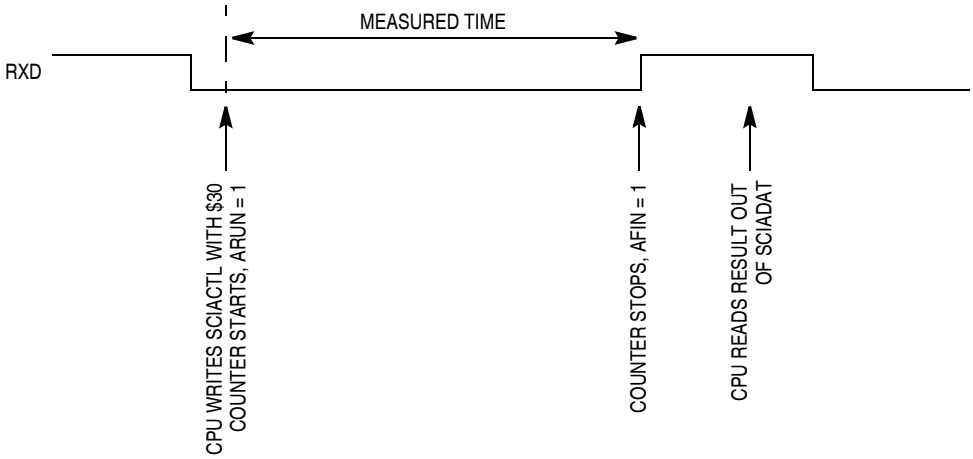



**Figure 14-22. Bit Time Measurement with ACLK = 1, Scenario A**

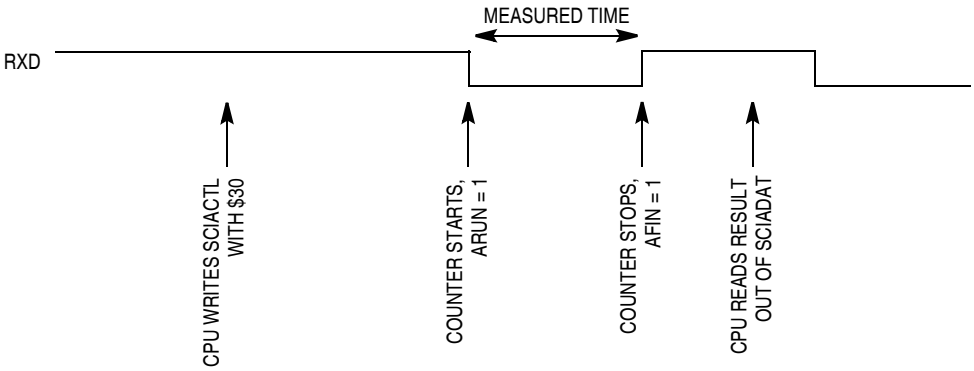



**Figure 14-23. Bit Time Measurement with ACLK = 1, Scenario B**

# Chapter 15
# System Integration Module (SIM)

## 15.1 Introduction

This section describes the system integration module (SIM). Together with the central processor unit (CPU), the SIM controls all microcontroller unit (MCU) activities. A block diagram of the SIM is shown in Figure 15-1. Table 15-1 is a summary of the SIM input/output (I/O) registers. The SIM is a system state controller that coordinates CPU and exception timing.

The SIM is responsible for:

- Bus clock generation and control for CPU and peripherals:
  - Stop/wait/reset/break entry and recovery
  - Internal clock control
- Master reset control, including power-on reset (POR) and computer operating properly (COP) timeout
- Interrupt arbitration

Table 15-1 shows the internal signal names used in this section.

**Table 15-1. Signal Name Conventions**

| Signal Name | Description |
|---|---|
| CGMXCLK | Buffered version of OSC1 from clock generator module (CGM) |
| CGMVCLK | PLL output |
| CGMOUT | PLL-based or OSC1-based clock output from CGM module (Bus clock = CGMOUT divided by two) |
| IAB | Internal address bus |
| IDB | Internal data bus |
| PORRST | Signal from the power-on reset module to the SIM |
| IRST | Internal reset signal |
| R/$\overline{W}$ | Read/write signal |

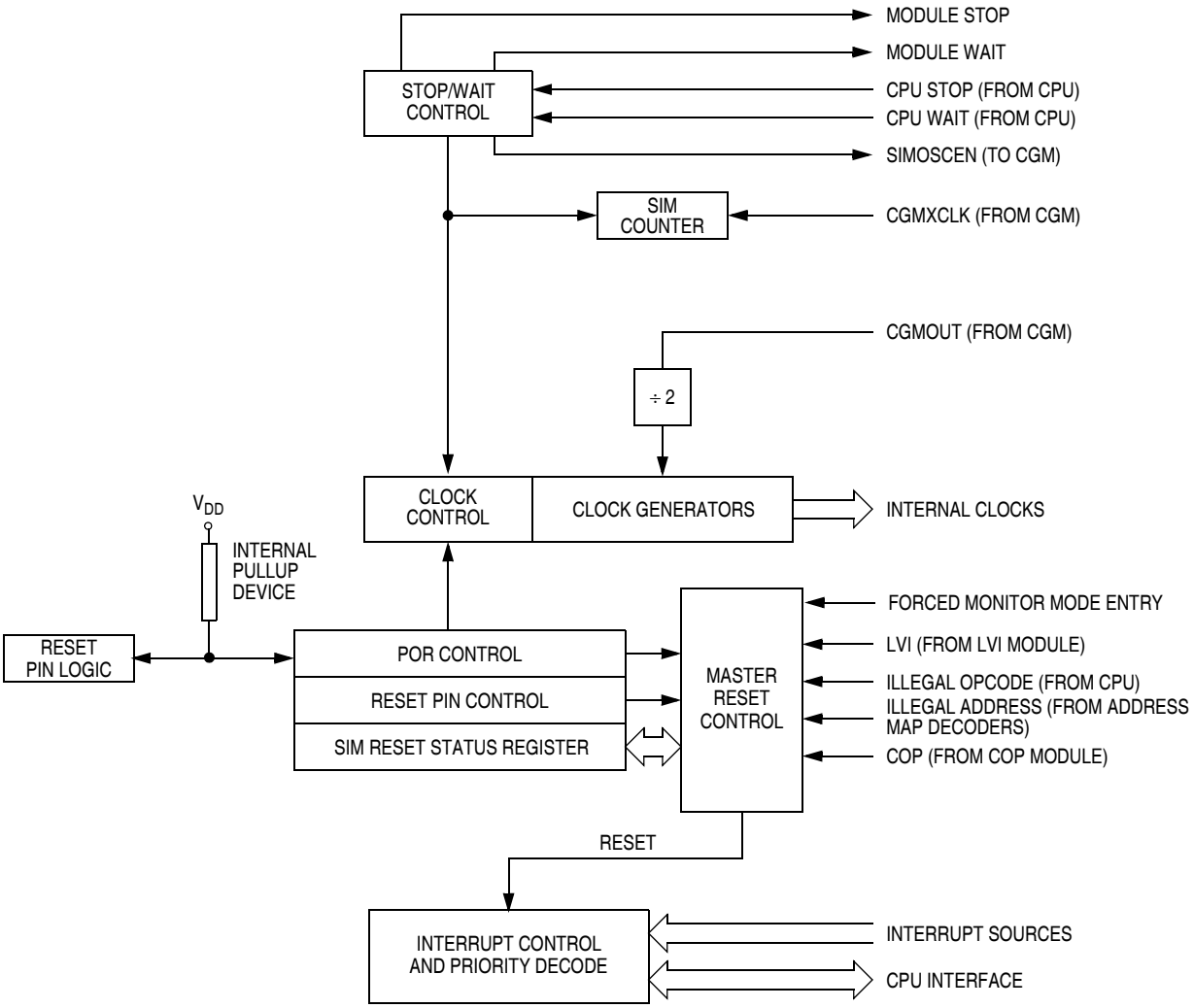



**Figure 15-1. SIM Block Diagram**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FE00 | Break Status Register (BSR) See page 237. | Read: | R | R | R | R | R | R | SBSW | R |
| | | Write: | | | | | | | Note(1) | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE01 | SIM Reset Status Register (SRSR) See page 237. | Read: | POR | PIN | COP | ILOP | ILAD | MODRST | LVI | 0 |
| | | Write: | | | | | | | | |
| | | POR: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE03 | Break Flag Control Register (BFCR) See page 238. | Read: | BCFE | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | | | | | | | |
| $FE04 | Interrupt Status Register 1 (INT1) See page 231. | Read: | IF6 | IF5 | IF4 | IF3 | IF2 | IF1 | 0 | 0 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE05 | Interrupt Status Register 2 (INT2) See page 233. | Read: | IF14 | IF13 | IF12 | IF11 | IF10 | IF9 | IF8 | IF7 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE06 | Interrupt Status Register 3 (INT3) See page 233. | Read: | IF22 | IF32 | IF20 | IF19 | IF18 | IF17 | IF16 | IF15 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE07 | Interrupt Status Register 4 (INT4) See page 233. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | IF24 | IF23 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. Writing a 0 clears SBSW.

= Unimplemented  R = Reserved

**Figure 15-2. SIM I/O Register Summary**

## 15.2 SIM Bus Clock Control and Generation

The bus clock generator provides system clock signals for the CPU and peripherals on the MCU. The system clocks are generated from an incoming clock, CGMOUT, as shown in Figure 15-3. This clock originates from either an external oscillator or from the on-chip PLL.

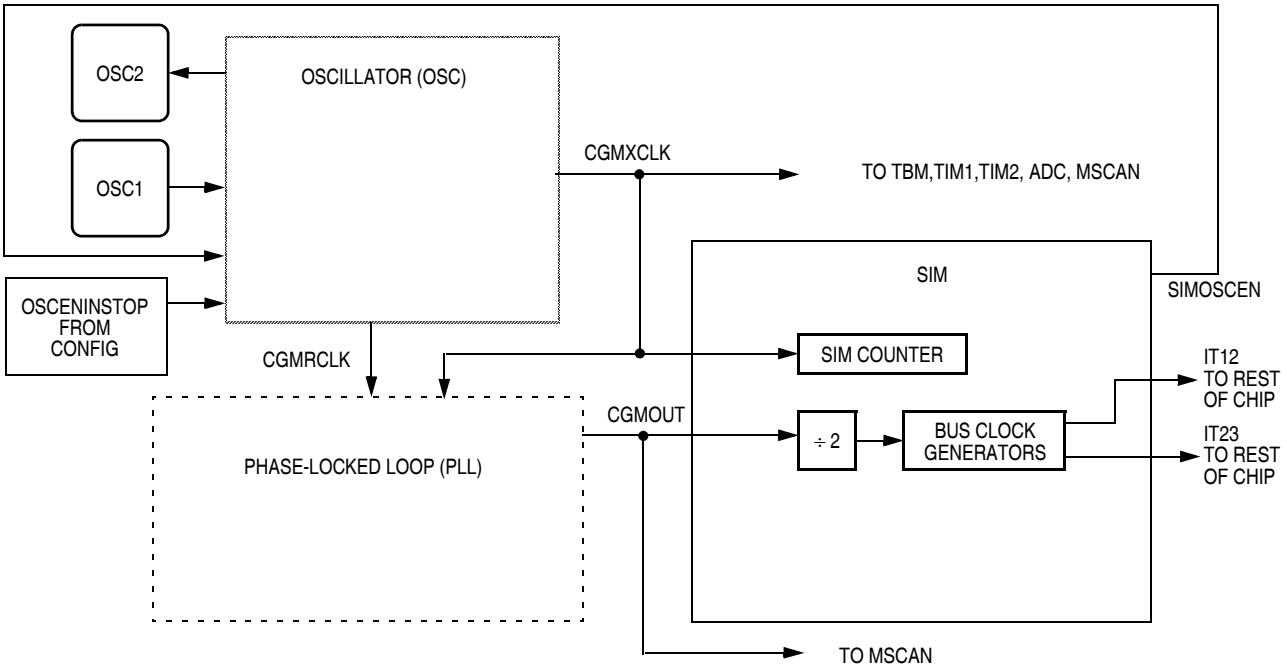



**Figure 15-3. System Clock Signals**

### 15.2.1 Bus Timing

In user mode, the internal bus frequency is either the crystal oscillator output (CGMXCLK) divided by four or the PLL output (CGMVCLK) divided by four.

### 15.2.2 Clock Startup from POR or LVI Reset

When the power-on reset module or the low-voltage inhibit module generates a reset, the clocks to the CPU and peripherals are inactive and held in an inactive phase until after the 4096 CGMXCLK cycle POR timeout has completed. The $\overline{\text{RST}}$ pin is driven low by the SIM during this entire period. The bus clocks start upon completion of the timeout.

### 15.2.3 Clocks in Stop Mode and Wait Mode

Upon exit from stop mode by an interrupt or reset, the SIM allows CGMXCLK to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay timeout. This timeout is selectable as 4096 or 32 CGMXCLK cycles. See 15.6.2 Stop Mode.

In wait mode, the CPU clocks are inactive. The SIM also produces two sets of clocks for other modules. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

## 15.3 Reset and System Initialization

The MCU has these reset sources:

- Power-on reset module (POR)
- External reset pin ($\overline{RST}$)
- Computer operating properly module (COP)
- Low-voltage inhibit module (LVI)
- Illegal opcode
- Illegal address
- Forced monitor mode entry reset (MODRST)

All of these resets produce the vector $FFFE:$FFFF ($FEFE:$FEFF in monitor mode) and assert the internal reset signal (IRST). IRST causes all registers to be returned to their default values and all modules to be returned to their reset states.

An internal reset clears the SIM counter (see 15.4 SIM Counter), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). See 15.7 SIM Registers.

A reset immediately stops the operation of the instruction being executed. Reset initializes certain control and status bits. Reset selects CGMXCLK divided by four as the bus clock.

### 15.3.1 External Pin Reset

The $\overline{RST}$ pin circuit includes an internal pullup device. Pulling the asynchronous $\overline{RST}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{RST}$ is held low for at least the minimum $t_{RL}$ time and no other reset sources are present. Figure 15-4 shows the relative timing.

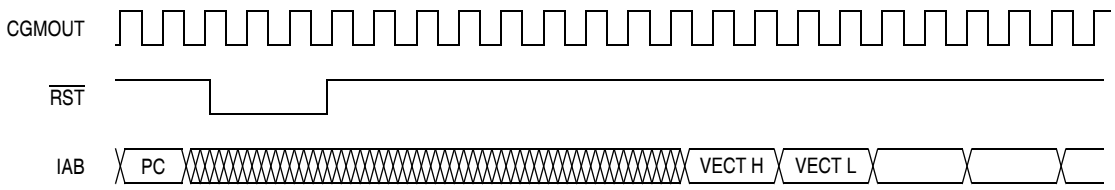



**Figure 15-4. External Reset Timing**

### 15.3.2 Active Resets from Internal Sources

All internal reset sources actively pull the $\overline{RST}$ pin low for 32 CGMXCLK cycles to allow resetting of external peripherals. The internal reset continues to be asserted for an additional 32 cycles at which point the reset vector will be fetched. See Figure 15-5. An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR. See Figure 15-6.

***NOTE***

*For LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM forces the $\overline{RST}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{RST}$ shown in Figure 15-5.*

The COP reset is asynchronous to the bus clock.

The active reset feature allows the part to issue a reset to peripherals and other chips within a system built around the MCU.

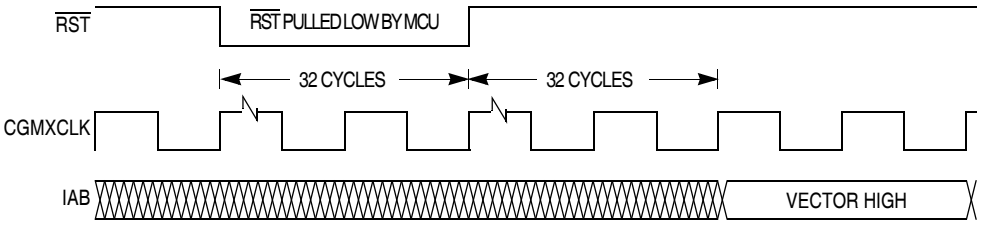



**Figure 15-5. Internal Reset Timing**

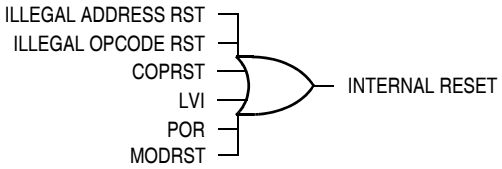



**Figure 15-6. Sources of Internal Reset**

**Table 15-2. Reset Recovery**

| Reset Recovery Type | Actual Number of Cycles |
|---|---|
| POR/LVI | 4163 (4096 + 64 + 3) |
| All others | 67 (64 + 3) |

#### *15.3.2.1 Power-On Reset*

When power is first applied to the MCU, the power-on reset module (POR) generates a pulse to indicate that power-on has occurred. The external reset pin ($\overline{RST}$) is held low while the SIM counter counts out 4096 + 32 CGMXCLK cycles. Thirty-two CGMXCLK cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur.

At power-on, these events occur:

- A POR pulse is generated.
- The internal reset signal is asserted.
- The SIM enables CGMOUT.
- Internal clocks to the CPU and modules are held inactive for 4096 CGMXCLK cycles to allow stabilization of the oscillator.
- The $\overline{RST}$ pin is driven low during the oscillator stabilization time.
- The POR bit of the SIM reset status register (SRSR) is set.

#### *15.3.2.2 Computer Operating Properly (COP) Reset*

An input to the SIM is reserved for the COP reset signal. The overflow of the COP counter causes an internal reset and sets the COP bit in the SIM reset status register (SRSR) if the COPD bit in the CONFIG1 register is cleared. The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

The COP module is disabled if the $\overline{RST}$ pin or the $\overline{IRQ}$ pin is held at $V_{TST}$ while the MCU is in monitor mode. During a break state, $V_{TST}$ on the $\overline{RST}$ pin disables the COP module.

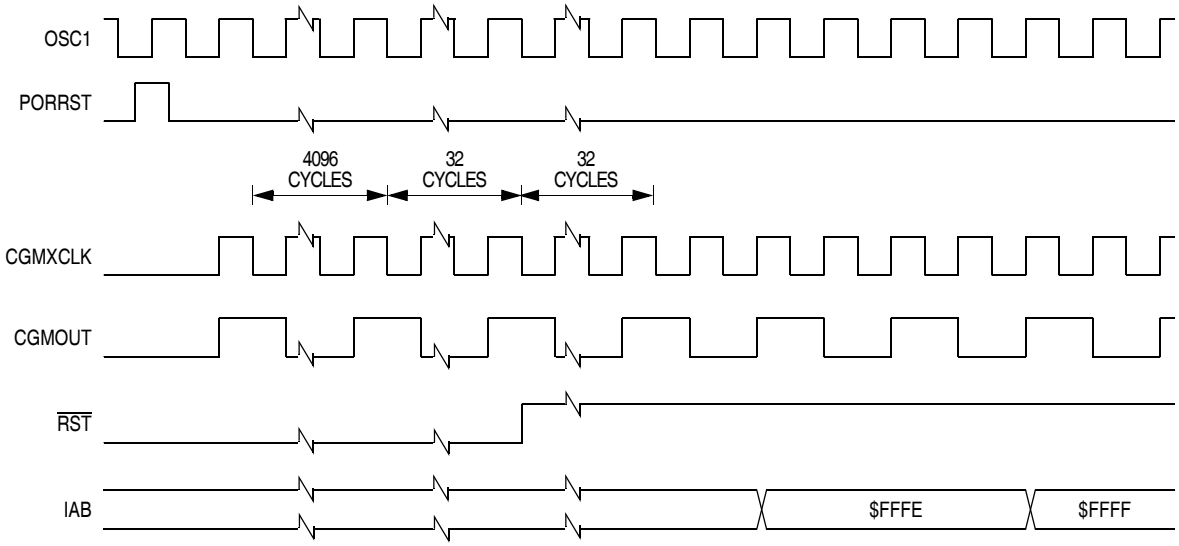



**Figure 15-7. POR Recovery**

#### *15.3.2.3 Illegal Opcode Reset*

The SIM decodes signals from the CPU to detect illegal instructions. An illegal instruction sets the ILOP bit in the SIM reset status register (SRSR) and causes a reset.

If the stop enable bit, STOP, in the CONFIG1 register is 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

#### *15.3.2.4 Illegal Address Reset*

An opcode fetch from an unmapped address generates an illegal address reset. The SIM verifies that the CPU is fetching an opcode prior to asserting the ILAD bit in the SIM reset status register (SRSR) and resetting the MCU. A data fetch from an unmapped address does not generate a reset. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

#### *15.3.2.5 Low-Voltage Inhibit (LVI) Reset*

The low-voltage inhibit module (LVI) asserts its output to the SIM when the $V_{DD}$ voltage falls to the $V_{TRIPF}$ voltage. The LVI bit in the SIM reset status register (SRSR) is set, and the external reset pin ($\overline{\text{RST}}$) is asserted if the LVIPWRD and LVIRSTD bits in the CONFIG1 register are 0. The $\overline{\text{RST}}$ pin will be held low while the SIM counter counts out 4096 + 32 CGMXCLK cycles after $V_{DD}$ rises above $V_{TRIPR}$. Thirty-two CGMXCLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

#### *15.3.2.6 Monitor Mode Entry Module Reset (MODRST)*

The monitor mode entry module reset (MODRST) asserts its output to the SIM when monitor mode is entered in the condition where the reset vectors are erased ($FF) (see 20.3.1.1 Normal Monitor Mode). When MODRST gets asserted, an internal reset occurs. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

## 15.4 SIM Counter

The SIM counter is used by the power-on reset module (POR) and in stop mode recovery to allow the oscillator time to stabilize before enabling the internal bus clocks. The SIM counter also serves as a prescaler for the computer operating properly (COP) module. The SIM counter overflow supplies the clock for the COP module. The SIM counter is 12 bits long.

### 15.4.1 SIM Counter During Power-On Reset

The power-on reset module (POR) detects power applied to the MCU. At power-on, the POR circuit asserts the signal PORRST. Once the SIM is initialized, it enables the clock generation module (CGM) to drive the bus clock state machine.

### 15.4.2 SIM Counter During Stop Mode Recovery

The SIM counter also is used for stop mode recovery. The STOP instruction clears the SIM counter. After an interrupt, break, or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the CONFIG1 register. If the SSREC bit is a 1, then the stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32 CGMXCLK cycles. This is ideal for applications using crystals with the OSCENINSTOP bit set. External crystal applications should use the full stop recovery time, SSREC cleared, with the OSCENINSTOP bit cleared. See 5.2 Functional Description.

### 15.4.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. See 15.6.2 Stop Mode for details. The SIM counter is free-running after all reset states. See 15.3.2 Active Resets from Internal Sources for counter control and internal reset recovery sequences.

## 15.5 Exception Control

Normal, sequential program execution can be changed in three different ways:

- Interrupts:
  - Maskable hardware CPU interrupts
  - Non-maskable software interrupt instruction (SWI)
- Reset
- Break interrupts

### 15.5.1 Interrupts

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. Figure 15-8 shows interrupt entry timing. Figure 15-9 shows interrupt recovery timing.

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared). See Figure 15-10.

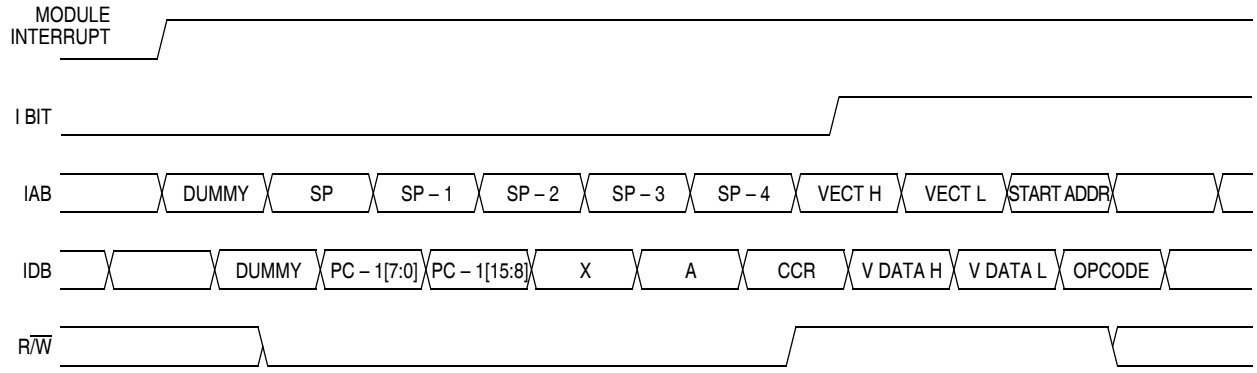



**Figure 15-8. Interrupt Entry Timing**

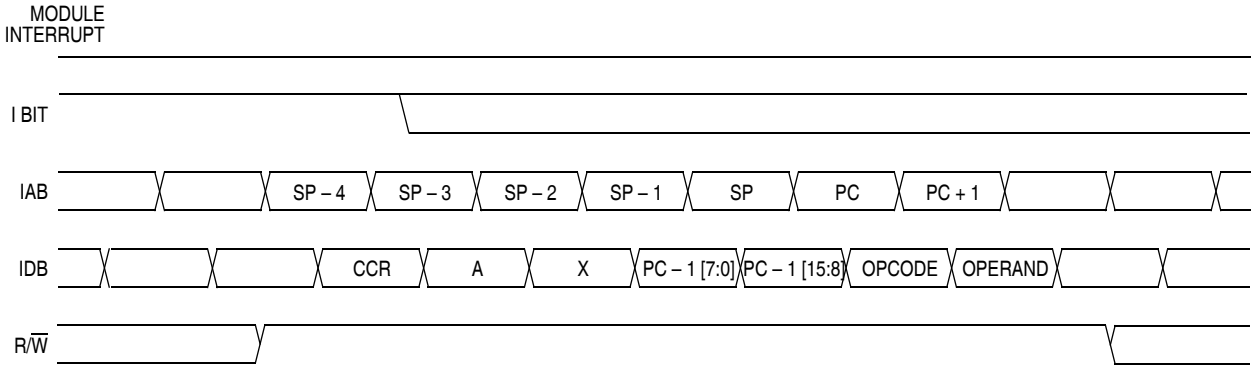



**Figure 15-9. Interrupt Recovery Timing**

### *15.5.1.1 Hardware Interrupts*

A hardware interrupt does not stop the current instruction. Processing of a hardware interrupt begins after completion of the current instruction. When the current instruction is complete, the SIM checks all pending hardware interrupts. If interrupts are not masked (I bit clear in the condition code register) and if the corresponding interrupt enable bit is set, the SIM proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. Figure 15-11 demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

The LDA opcode is prefetched by both the INT1 and INT2 RTI instructions. However, in the case of the INT1 RTI prefetch, this is a redundant operation.

***NOTE***

*To maintain compatibility with the M6805 Family, the H register is not pushed on the stack during interrupt entry. If the interrupt service routine modifies the H register or uses the indexed addressing mode, software should save the H register and then restore it prior to exiting the routine.*

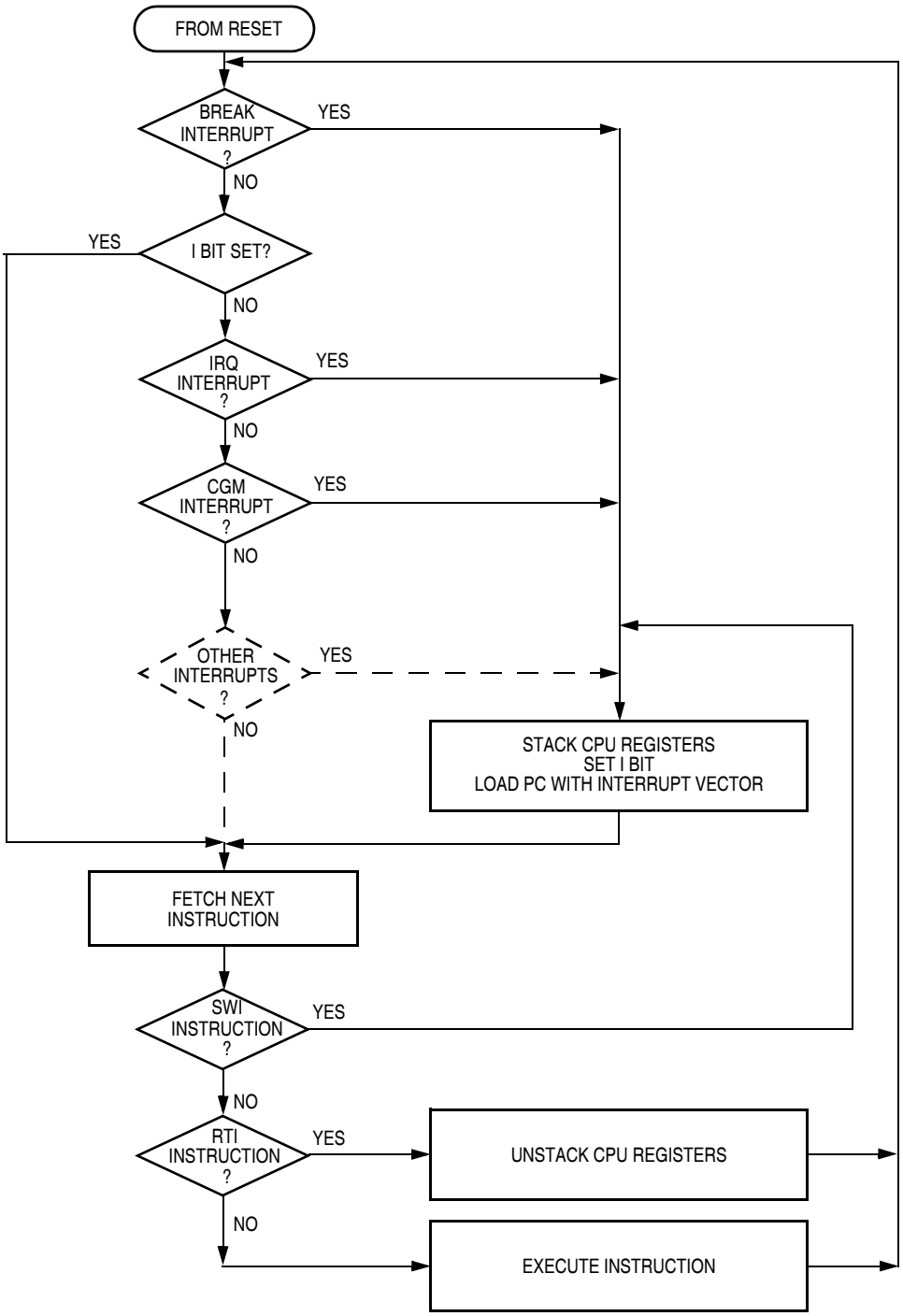



**Figure 15-10. Interrupt Processing**

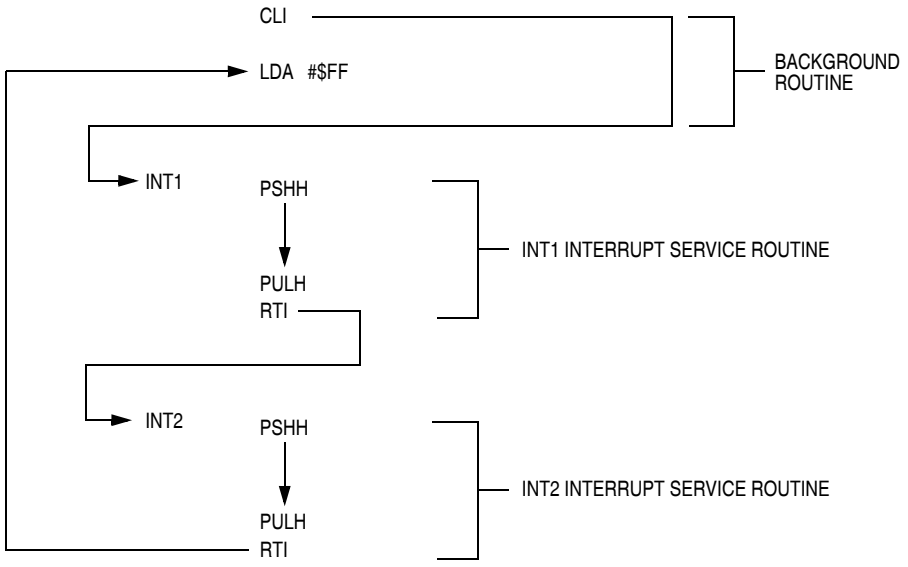



**Figure 15-11. Interrupt Recognition Example**

### *15.5.1.2 SWI Instruction*

The SWI instruction is a non-maskable instruction that causes an interrupt regardless of the state of the interrupt mask (I bit) in the condition code register.

***NOTE***

*A software interrupt pushes PC onto the stack. A software interrupt does not push PC – 1, as a hardware interrupt does.*

### *15.5.1.3 Interrupt Status Registers*

The flags in the interrupt status registers identify maskable interrupt sources. Table 15-3 summarizes the interrupt sources, hardware flag bits, hardware interrupt mask bits, interrupt status register flags, interrupt priority, and exception vectors. The interrupt status registers can be useful for debugging.

*Interrupt Status Register 1*

Address: $FE04

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | IF6 | IF5 | IF4 | IF3 | IF2 | IF1 | 0 | 0 |
| Write: | R | R | R | R | R | R | R | R |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 15-12. Interrupt Status Register 1 (INT1)**

**IF6–IF1 — Interrupt Flags 1–6**

These flags indicate the presence of interrupt requests from the sources shown in Table 15-3.

1 = Interrupt request present
0 = No interrupt request present

**Bit 0 and Bit 1 — Always read 0**

## Table 15-3. Interrupt Sources

| Source | Flag | Mask[1] | INT Register Flag | Priority[2] | Vector Address |
|---|---|---|---|---|---|
| Reset | None | None | None | 0 | $FFFE–$FFFF |
| SWI instruction | None | None | None | 0 | $FFFC–$FFFD |
| $\overline{\text{IRQ}}$ pin | IRQF | IMASK1 | IF1 | 1 | $FFFA–$FFFB |
| CGM change in lock | PLLF | PLLIE | IF2 | 2 | $FFF8–$FFF9 |
| TIM1 channel 0 | CH0F | CH0IE | IF3 | 3 | $FFF6–$FFF7 |
| TIM1 channel 1 | CH1F | CH1IE | IF4 | 4 | $FFF4–$FFF5 |
| TIM1 overflow | TOF | TOIE | IF5 | 5 | $FFF2–$FFF3 |
| TIM2 channel 0 | CH0F | CH0IE | IF6 | 6 | $FFF0–$FFF1 |
| TIM2 channel 1 | CH1F | CH1IE | IF7 | 7 | $FFEE–$FFEF |
| TIM2 overflow | TOF | TOIE | IF8 | 8 | $FFEC–$FFED |
| SPI receiver full | SPRF | SPRIE | IF9 | 9 | $FFEA–$FFEB |
| SPI overflow | OVRF | ERRIE | | | |
| SPI mode fault | MODF | ERRIE | | | |
| SPI transmitter empty | SPTE | SPTIE | IF10 | 10 | $FFE8–$FFE9 |
| SCI receiver overrun | OR | ORIE | IF11 | 11 | $FFE6–$FFE7 |
| SCI noise flag | NF | NEIE | | | |
| SCI framing error | FE | FEIE | | | |
| SCI parity error | PE | PEIE | | | |
| SCI receiver full | SCRF | SCRIE | IF12 | 12 | $FFE4–$FFE5 |
| SCI input idle | IDLE | ILIE | | | |
| SCI transmitter empty | SCTE | SCTIE | IF13 | 13 | $FFE2–$FFE3 |
| SCI transmission complete | TC | TCIE | | | |
| Keyboard pin | KEYF | IMASKK | IF14 | 14 | $FFE0–$FFE1 |
| ADC conversion complete | COCO | AIEN | IF15 | 15 | $FFDE–$FFDF |
| Timebase | TBIF | TBIE | IF16 | 16 | $FFDC–$FFDD |
| MSCAN08 receiver wakeup | WUPIF | WUPIE | IF17 | 17 | $FFDA–$FFDB |
| MSCAN08 error | RWRNIF<br>TWRNIF<br>RERIF<br>TERRIF<br>BOFFIF<br>OVRIF | RWRNIE<br>TWRNIE<br>RERRIE<br>TERRIE<br>BOFFIE<br>OVRIE | IF18 | 18 | $FFD8–$FFD9 |
| MSCAN08 receiver | RXF | RXFIE | IF19 | 19 | $FFD6–$FFD7 |
| MSCAN08 transmitter | TXE2<br>TXE1<br>TXE0 | TXEIE2<br>TXEIE1<br>TXEIE0 | IF20 | 20 | $FFD4–$FFD5 |
| TIM2 channel 2 | CH2F | CH2IE | IF21 | 21 | $FFD2–FFD3 |
| TIM2 channel 3 | CH3F | CH3IE | IF22 | 22 | $FFD0–FFD1 |
| TIM2 channel 4 | CH4F | CH4IE | IF23 | 23 | $FFCE–FFCF |
| TIM2 channel 5 | CH5F | CH5IE | IF24 | 24 | $FFCC–FFCD |

1. The I bit in the condition code register is a global mask for all interrupt sources except the SWI instruction.
2. 0 = highest priority

*Interrupt Status Register 2*

Address: $FE05

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | IF14 | IF13 | IF12 | IF11 | IF10 | IF9 | IF8 | IF7 |
| Write: | R | R | R | R | R | R | R | R |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 15-13. Interrupt Status Register 2 (INT2)**

**IF14–IF7 — Interrupt Flags 14–7**

These flags indicate the presence of interrupt requests from the sources shown in Table 15-3.

1 = Interrupt request present
0 = No interrupt request present

*Interrupt Status Register 3*

Address: $FE06

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | IF22 | IF21 | IF20 | IF19 | IF18 | IF17 | IF16 | IF15 |
| Write: | R | R | R | R | R | R | R | R |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 15-14. Interrupt Status Register 3 (INT3)**

**IF22–IF15 — Interrupt Flags 22–15**

These flags indicate the presence of an interrupt request from the source shown in Table 15-3.

1 = Interrupt request present
0 = No interrupt request present

*Interrupt Status Register 4*

Address: $FE07

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | 0 | 0 | IF24 | IF23 |
| Write: | R | R | R | R | R | R | R | R |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 15-15. Interrupt Status Register 4 (INT4)**

**Bits 7–2 — Always read 0**

**IF24–IF23 — Interrupt Flags 24–23**

These flags indicate the presence of an interrupt request from the source shown in Table 15-3.

1 = Interrupt request present
0 = No interrupt request present

### 15.5.2 Reset

All reset sources always have equal and highest priority and cannot be arbitrated.

### 15.5.3 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output (see Chapter 18 Timer Interface Module (TIM1) and Chapter 19 Timer Interface Module (TIM2)). The SIM puts the CPU into the break state by forcing it to the SWI vector location. Refer to the break interrupt subsection of each module to see how each module is affected by the break state.

### 15.5.4 Status Flag Protection in Break Mode

The SIM controls whether status flags contained in other modules can be cleared during break mode. The user can select whether flags are protected from being cleared by properly initializing the break clear flag enable bit (BCFE) in the SIM break flag control register (BFCR).

Protecting flags in break mode ensures that set flags will not be cleared while in break mode. This protection allows registers to be freely read and written during break mode without losing status flag information.

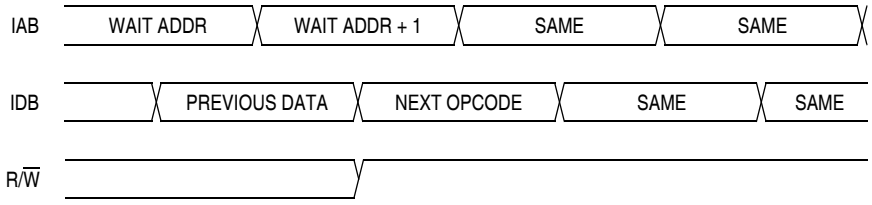
Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a 2-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step is accomplished prior to entering break mode. Upon leaving break mode, execution of the second step will clear the flag as normal.

## 15.6 Low-Power Modes

Executing the WAIT or STOP instruction puts the MCU in a low power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described in the following subsections. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

### 15.6.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. Figure 15-16 shows the timing for wait mode entry.



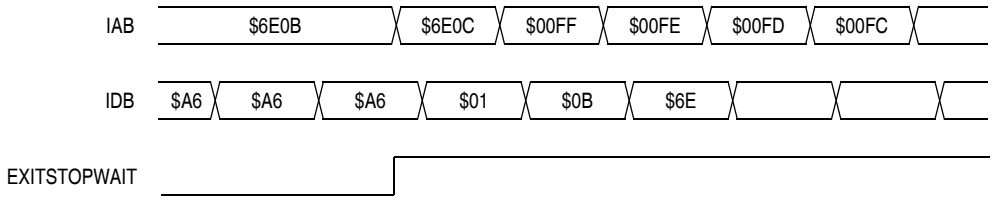

Note: Previous data can be operand data or the WAIT opcode, depending on the last instruction.

**Figure 15-16. Wait Mode Entry Timing**

A module that is active during wait mode can wakeup the CPU with an interrupt if the interrupt is enabled. Stacking for the interrupt begins one cycle after the WAIT instruction during which the interrupt occurred. In wait mode, the CPU clocks are inactive. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

Wait mode also can be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the SIM break status register (BSR). If the COP disable bit, COPD, in the CONFIG1 register is 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.

Figure 15-17 and Figure 15-18 show the timing for WAIT recovery.




Note: EXITSTOPWAIT = $\overline{\text{RST}}$ pin, CPU interrupt, or break interrupt

**Figure 15-17. Wait Recovery from Interrupt or Break**

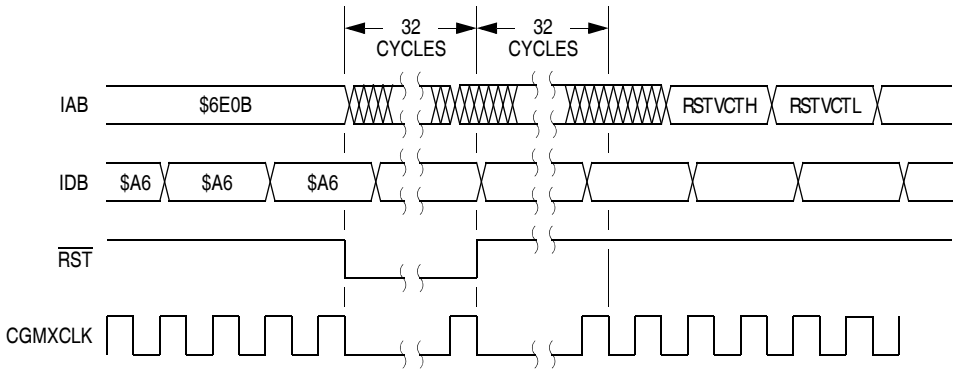



**Figure 15-18. Wait Recovery from Internal Reset**

## 15.6.2 Stop Mode

In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset also causes an exit from stop mode.

The SIM disables the clock generator module outputs (CGMOUT and CGMXCLK) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in CONFIG1. If SSREC is set, stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32. This is ideal for applications using canned oscillators that do not require long startup times from stop mode.

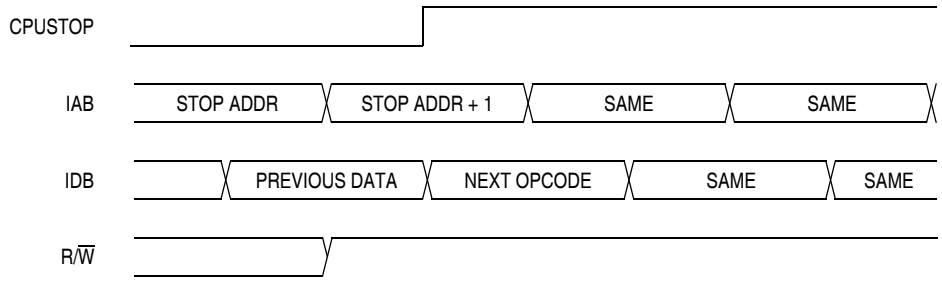
***NOTE***

*External crystal applications should use the full stop recovery time by clearing the SSREC bit unless OSCENINSTOP bit is set in CONFIG2.*

The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. Figure 15-19 shows stop mode entry timing. Figure 15-20 shows stop mode recovery time from interrupt.

***NOTE***

*To minimize stop current, all pins configured as inputs should be driven to a 1 or 0.*




Note: Previous data can be operand data or the STOP opcode, depending on the last instruction.

**Figure 15-19. Stop Mode Entry Timing**

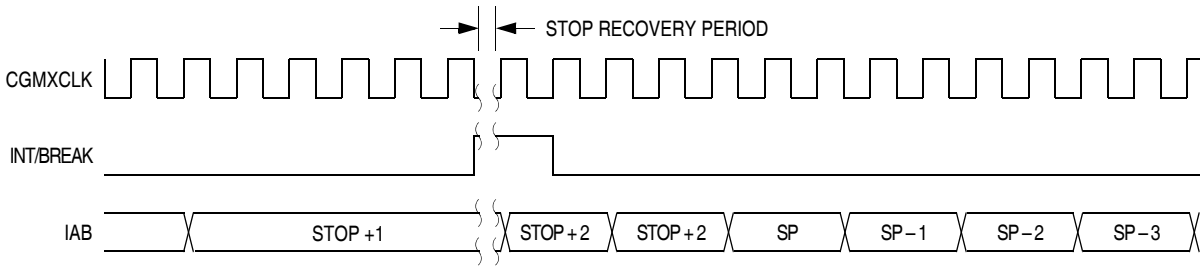



**Figure 15-20. Stop Mode Recovery from Interrupt**

## 15.7 SIM Registers

The SIM has three memory-mapped registers. Table 15-4 shows the mapping of these registers.

**Table 15-4. SIM Registers**

| Address | Register | Access Mode |
|---|---|---|
| $FE00 | BSR | User |
| $FE01 | SRSR | User |
| $FE03 | BFCR | User |

### 15.7.1 Break Status Register

The break status register (BSR) contains a flag to indicate that a break caused an exit from wait mode. This register is only used in emulation mode.

Address: $FE00

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | R | R | R | R | R | R | SBSW | R |
| Write: | | | | | | | Note(1) | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

1. Writing a 0 clears SBSW.

**Figure 15-21. Break Status Register (BSR)**

**SBSW — SIM Break Stop/Wait**

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it.

1 = Wait mode was exited by break interrupt.
0 = Wait mode was not exited by break interrupt.

### 15.7.2 SIM Reset Status Register

This register contains six flags that show the source of the last reset provided all previous reset status bits have been cleared. Clear the SIM reset status register by reading it. A power-on reset sets the POR bit and clears all other bits in the register.

The register is initialized on power up with the POR bit set and all other bits cleared. During a POR or any other internal reset, the $\overline{RST}$ pin is pulled low. After the pin is released, it will be sampled 32 CGMXCLK cycles later. If the pin is not above $V_{IH}$ at this time, then the PIN bit may be set, in addition to whatever other bits are set.

Address: $FE01

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | POR | PIN | COP | ILOP | ILAD | MODRST | LVI | 0 |
| Write: | | | | | | | | |
| Reset: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 15-22. SIM Reset Status Register (SRSR)**

**POR — Power-On Reset Bit**

1 = Last reset caused by POR circuit
0 = Read of SRSR

**PIN — External Reset Bit**

1 = Last reset caused by external reset pin ($\overline{RST}$)
0 = POR or read of SRSR

**COP — Computer Operating Properly Reset Bit**

1 = Last reset caused by COP counter
0 = POR or read of SRSR

**ILOP — Illegal Opcode Reset Bit**

1 = Last reset caused by an illegal opcode
0 = POR or read of SRSR

**ILAD — Illegal Address Reset Bit (opcode fetches only)**

1 = Last reset caused by an opcode fetch from an illegal address
0 = POR or read of SRSR

**MODRST — Monitor Mode Entry Module Reset Bit**

1 = Last reset caused by monitor mode entry when vector locations $FFFE and $FFFF are $FF after POR while $\overline{IRQ}$ = $V_{DD}$
0 = POR or read of SRSR

**LVI — Low-Voltage Inhibit Reset Bit**

1 = Last reset caused by the LVI circuit
0 = POR or read of SRSR

### 15.7.3 Break Flag Control Register

The break flag control register contains a bit that enables software to clear status bits while the MCU is in a break state.

Address: $FE03

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: / Write: | BCFE | R | R | R | R | R | R | R |
| Reset: | 0 | | | | | | | |

R = Reserved

**Figure 15-23. Break Flag Control Register (BFCR)**

**BCFE — Break Clear Flag Enable Bit**

This read/write bit enables software to clear status bits by accessing status registers while the MCU is in a break state. To clear status bits during the break state, the BCFE bit must be set.

1 = Status bits clearable during break
0 = Status bits not clearable during break

# Chapter 16
# Serial Peripheral Interface (SPI) Module

## 16.1 Introduction

This section describes the serial peripheral interface (SPI) module, which allows full-duplex, synchronous, serial communications with peripheral devices.

The text that follows describes the SPI. The SPI I/O pin names are $\overline{SS}$ (slave select), SPSCK (SPI serial clock), MOSI (master out slave in), and MISO (master in/slave out). The SPI shares four I/O pins with four parallel I/O ports.

## 16.2 Features

Features of the SPI module include:

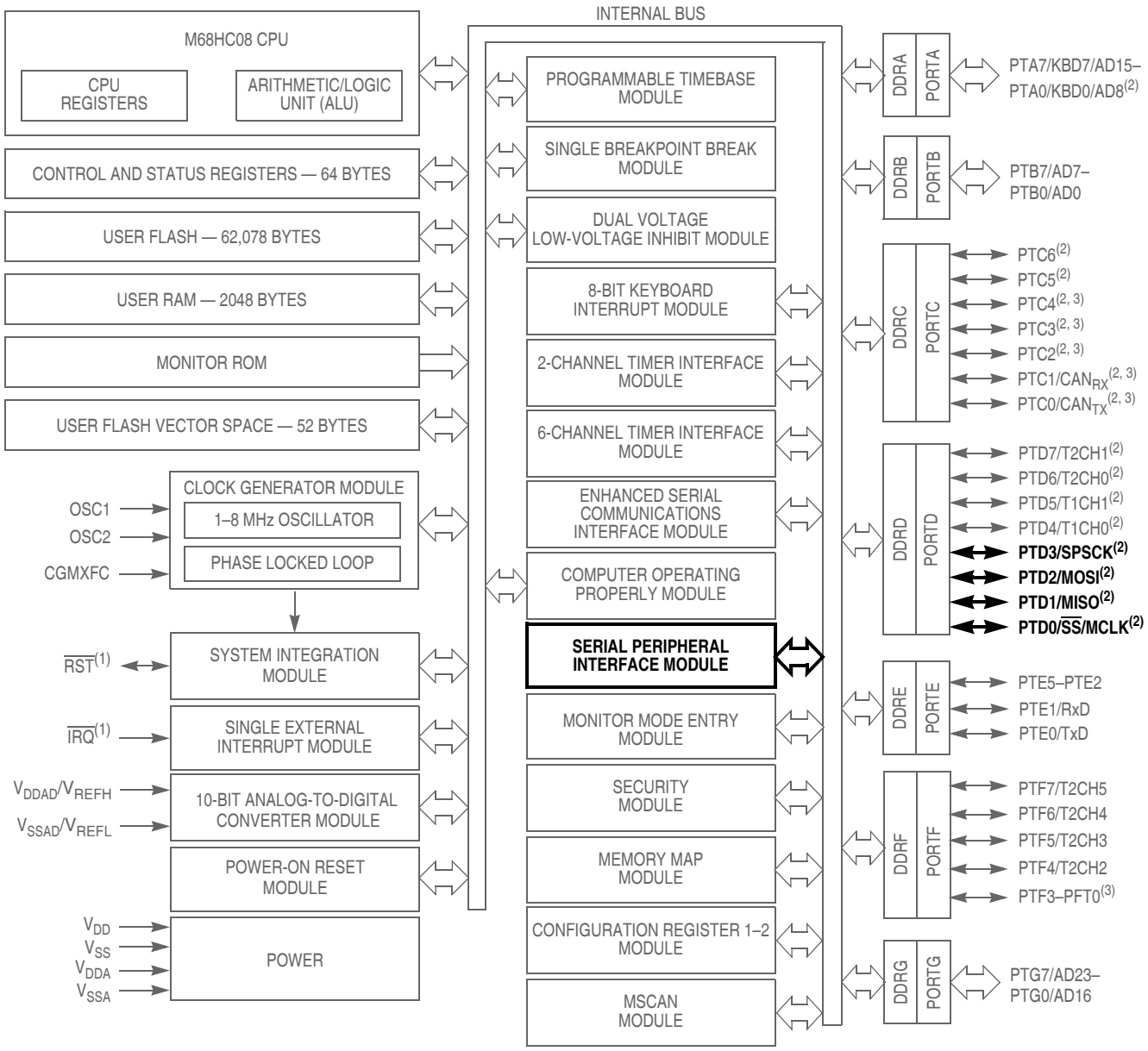
- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four master mode frequencies (maximum = bus frequency ÷ 2)
- Maximum slave mode frequency = bus frequency
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts:
  - SPRF (SPI receiver full)
  - SPTE (SPI transmitter empty)
- Mode fault error flag with CPU interrupt capability
- Overflow error flag with CPU interrupt capability
- Programmable wired-OR mode
- I/O (input/output) port bit(s) software configurable with pullup device(s) if configured as input port bit(s)

## 16.3 Functional Description

The SPI module allows full-duplex, synchronous, serial communication between the MCU and peripheral devices, including other MCUs. Software can poll the SPI status flags or SPI operation can be interrupt driven.

If a port bit is configured for input, then an internal pullup device may be enabled for that port bit.

The following paragraphs describe the operation of the SPI module. Refer to Figure 16-3 for a summary of the SPI I/O registers.




1. Pin contains integrated pullup device.
**2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.**
3. Higher current drive port pins

**Figure 16-1. Block Diagram Highlighting SPI Block and Pins**

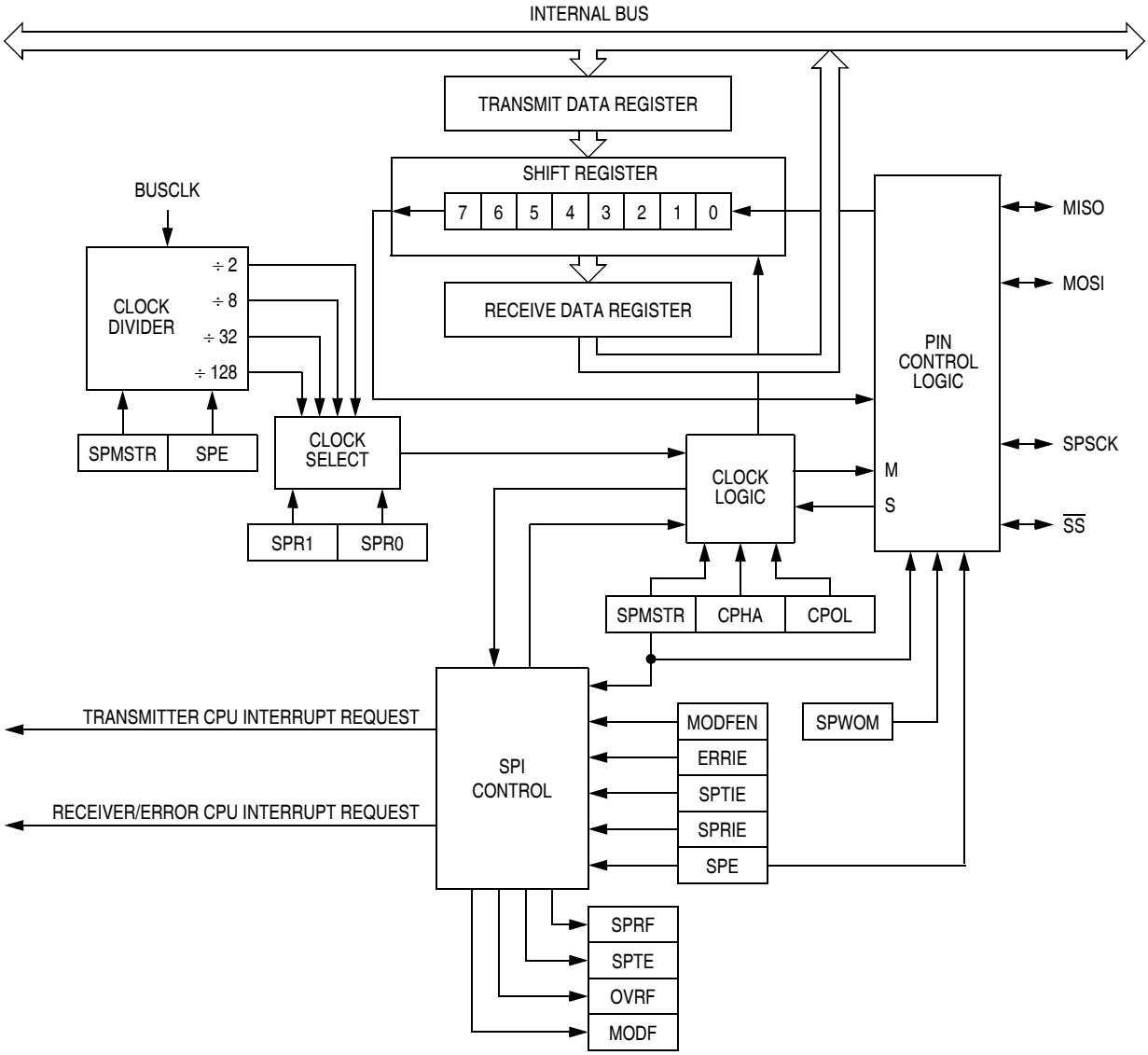



**Figure 16-2. SPI Module Block Diagram**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0010 | SPI Control Register (SPCR) See page 255. | Read: Write: | SPRIE | R | SPMSTR | CPOL | CPHA | SPWOM | SPE | SPTIE |
| | | Reset: | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| $0011 | SPI Status and Control Register (SPSCR) See page 256. | Read: | SPRF | ERRIE | OVRF | MODF | SPTE | MODFEN | SPR1 | SPR0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $0012 | SPI Data Register (SPDR) See page 258. | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| | | Reset: | Unaffected by reset | | | | | | | |

R = Reserved  = Unimplemented

**Figure 16-3. SPI I/O Register Summary**

### 16.3.1 Master Mode

The SPI operates in master mode when the SPI master bit, SPMSTR, is set.

***NOTE***

*In a multi-SPI system, configure the SPI modules as master or slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. See 16.12.1 SPI Control Register.*

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the transmit data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The byte begins shifting out on the MOSI pin under the control of the serial clock. See Figure 16-4.

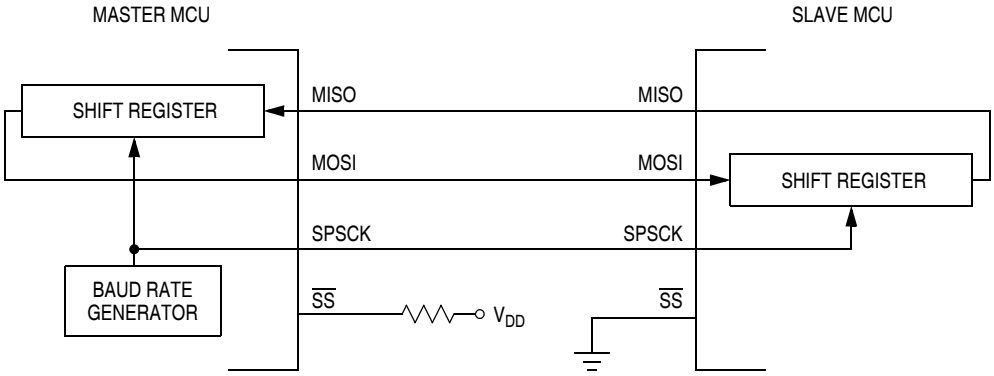



**Figure 16-4. Full-Duplex Master-Slave Connections**

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See 16.12.2 SPI Status and Control Register.) Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

As the byte shifts out on the MOSI pin of the master, another byte shifts in from the slave on the master's MISO pin. The transmission ends when the receiver full bit, SPRF, becomes set. At the same time that SPRF becomes set, the byte from the slave transfers to the receive data register. In normal operation, SPRF signals the end of a transmission. Software clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register. Writing to the SPI data register (SPDR) clears SPTE.

### 16.3.2 Slave Mode

The SPI operates in slave mode when SPMSTR is clear. In slave mode, the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be low. $\overline{SS}$ must remain low until the transmission is complete. See 16.6.2 Mode Fault Error.

In a slave SPI module, data enters the shift register under the control of the serial clock from the master SPI module. After a byte enters the shift register of a slave SPI, it transfers to the receive data register, and the SPRF bit is set. To prevent an overflow condition, slave software then must read the receive data register before another full byte enters the shift register.

The maximum frequency of the SPSCK for an SPI configured as a slave is the bus clock speed (which is twice as fast as the fastest master SPSCK clock that can be generated). The frequency of the SPSCK for an SPI configured as a slave does not have to correspond to any SPI baud rate. The baud rate only controls the speed of the SPSCK generated by an SPI configured as a master. Therefore, the frequency of the SPSCK for an SPI configured as a slave can be any frequency less than or equal to the bus speed.

When the master SPI starts a transmission, the data in the slave shift register begins shifting out on the MISO pin. The slave can load its shift register with a new byte for the next transmission by writing to its transmit data register. The slave must write to its transmit data register at least one bus cycle before the master starts the next transmission. Otherwise, the byte already in the slave shift register shifts out on the MISO pin. Data written to the slave shift register during a transmission remains in a buffer until the end of the transmission.

When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transmission. See 16.4 Transmission Formats.

***NOTE***

*SPSCK must be in the proper idle state before the slave is enabled to prevent SPSCK from appearing as a clock edge.*

## 16.4 Transmission Formats

During an SPI transmission, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock synchronizes shifting and sampling on the two serial data lines. A slave select line allows selection of an individual slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. On a master SPI device, the slave select line can optionally be used to indicate multiple-master bus contention.

### 16.4.1 Clock Phase and Polarity Controls

Software can select any of four combinations of serial clock (SPSCK) phase and polarity using two bits in the SPI control register (SPCR). The clock polarity is specified by the CPOL control bit, which selects an active high or low clock and has no significant effect on the transmission format.

The clock phase (CPHA) control bit selects one of two fundamentally different transmission formats. The clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

***NOTE***

*Before writing to the CPOL bit or the CPHA bit, disable the SPI by clearing the SPI enable bit (SPE).*

### 16.4.2 Transmission Format When CPHA = 0

Figure 16-5 shows an SPI transmission in which CPHA = 0. The figure should not be used as a replacement for data sheet parametric information.

Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select

input ($\overline{SS}$) is low, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See 16.6.2 Mode Fault Error.) When CPHA = 0, the first SPSCK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SPSCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the slave data transmission. The slave's $\overline{SS}$ pin must be toggled back to high and then low again between each byte transmitted as shown in Figure 16-6.

When CPHA = 0 for a slave, the falling edge of $\overline{SS}$ indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. Once the transmission begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the transmit data register and transferred to the shift register after the current transmission.

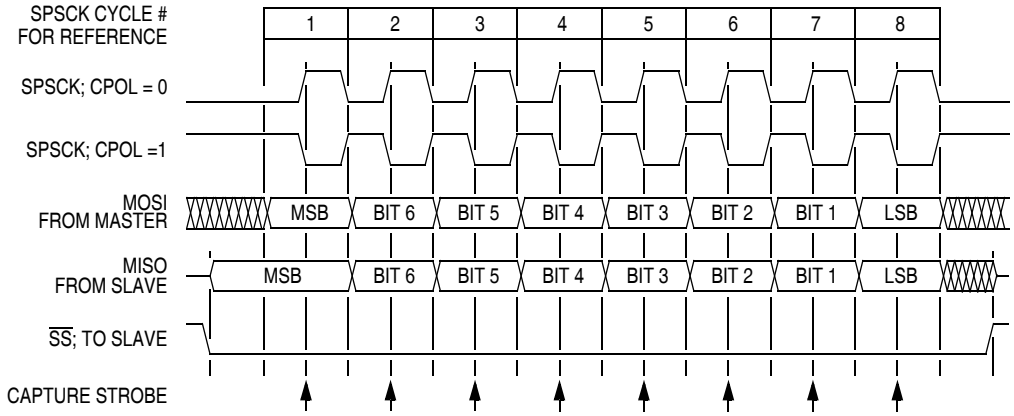



**Figure 16-5. Transmission Format (CPHA = 0)**

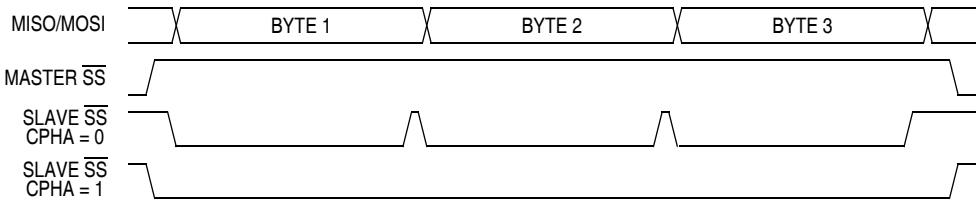



**Figure 16-6. CPHA/$\overline{SS}$ Timing**

### 16.4.3 Transmission Format When CPHA = 1

Figure 16-7 shows an SPI transmission in which CPHA = 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is low, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$

pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See 16.6.2 Mode Fault Error.) When CPHA = 1, the master begins driving its MOSI pin on the first SPSCK edge. Therefore, the slave uses the first SPSCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

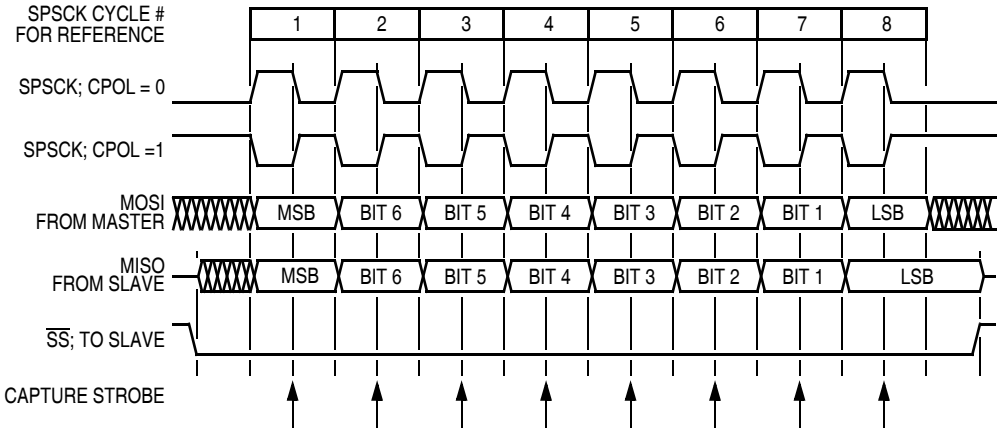



**Figure 16-7. Transmission Format (CPHA = 1)**

When CPHA = 1 for a slave, the first edge of the SPSCK indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. Once the transmission begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the first edge of SPSCK. Any data written after the first edge is stored in the transmit data register and transferred to the shift register after the current transmission.

### 16.4.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), writing to the SPDR starts a transmission. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SPSCK signal. When CPHA = 0, the SPSCK signal remains inactive for the first half of the first SPSCK cycle. When CPHA = 1, the first SPSCK cycle begins with an edge on the SPSCK line from its inactive to its active level. The SPI clock rate (selected by SPR1:SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See Figure 16-8.) The internal SPI clock in the master is a free-running derivative of the internal MCU clock. To conserve power, it is enabled only when both the SPE and SPMSTR bits are set. Since the SPI clock is free-running, it is uncertain where the write to the SPDR occurs relative to the slower SPSCK. This uncertainty causes the variation in the initiation delay shown in Figure 16-8. This delay is no longer than a single SPI bit time. That is, the maximum delay is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

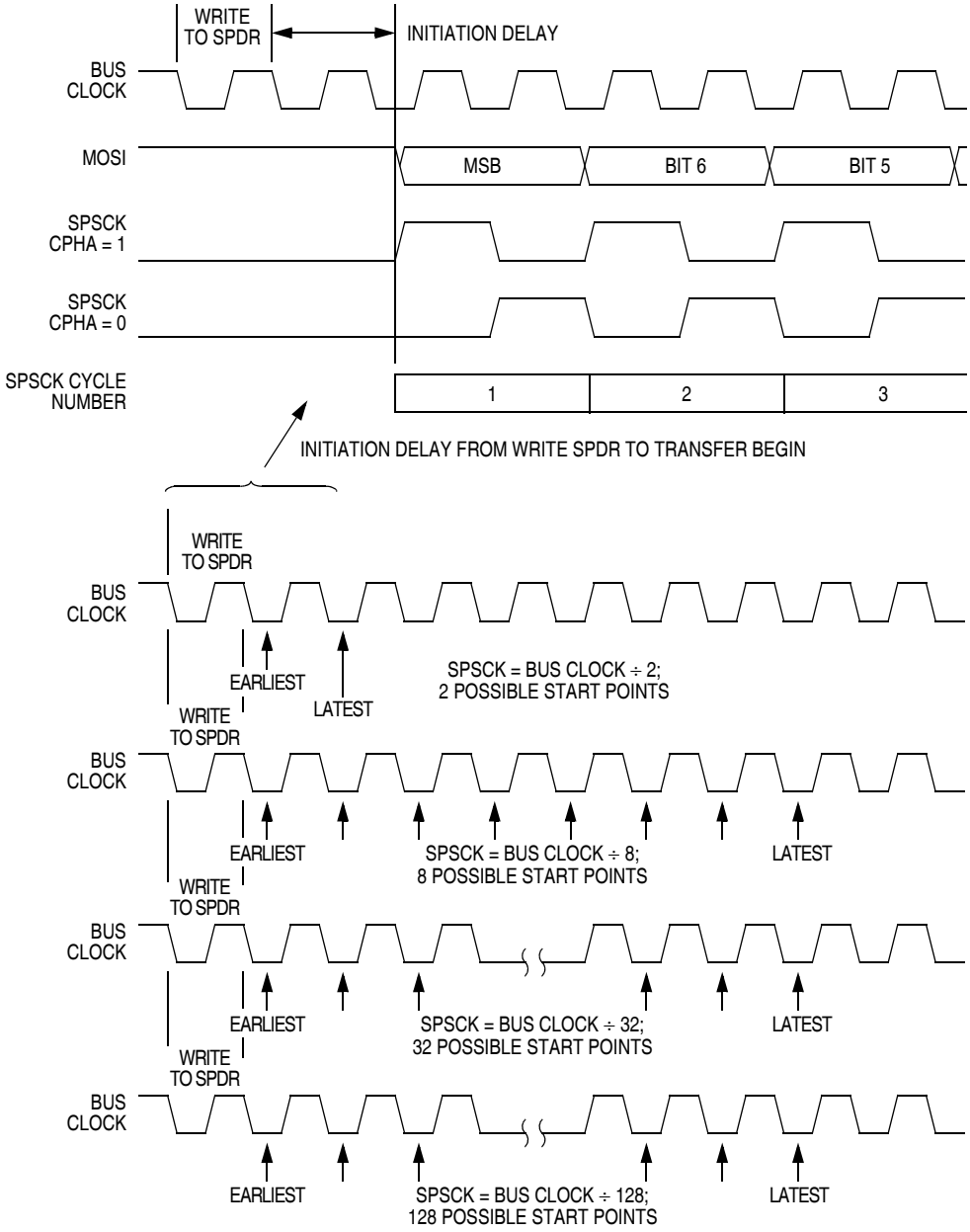



**Figure 16-8. Transmission Start Delay (Master)**

## 16.5 Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the transmit data register only when SPTE is high. Figure 16-9 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA: CPOL = 1:0).

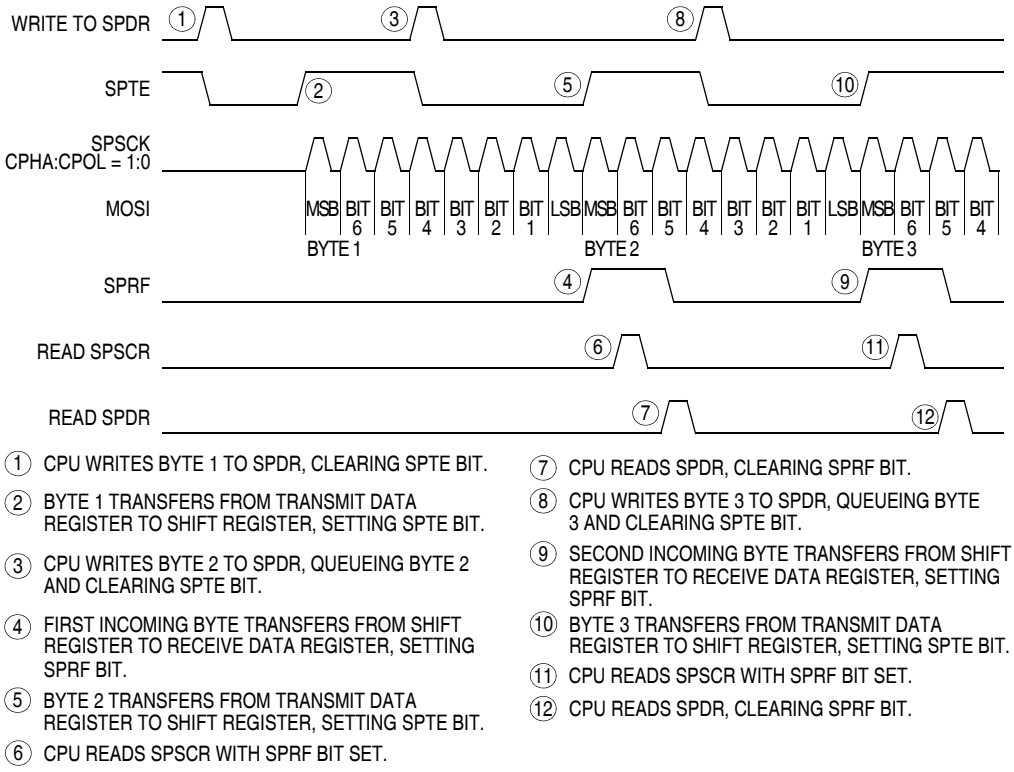



**Figure 16-9. SPRF/SPTE CPU Interrupt Timing**

The transmit data buffer allows back-to-back transmissions without the slave precisely timing its writes between transmissions as in a system with a single data buffer. Also, if no new data is written to the data buffer, the last value contained in the shift register is the next data word to be transmitted.

For an idle master or idle slave that has no data loaded into its transmit buffer, the SPTE is set again no more than two bus cycles after the transmit buffer empties into the shift register. This allows the user to queue up a 16-bit value to send. For an already active slave, the load of the shift register cannot occur until the transmission is completed. This implies that a back-to-back write to the transmit data register is not possible. SPTE indicates when the next write can occur.

## 16.6 Error Conditions

The following flags signal SPI error conditions:

- Overflow (OVRF) — Failing to read the SPI data register before the next full byte enters the shift register sets the OVRF bit. The new byte does not transfer to the receive data register, and the unread byte still can be read. OVRF is in the SPI status and control register.
- Mode fault error (MODF) — The MODF bit indicates that the voltage on the slave select pin ($\overline{SS}$) is inconsistent with the mode of the SPI. MODF is in the SPI status and control register.

### 16.6.1 Overflow Error

The overflow flag (OVRF) becomes set if the receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. The bit 1 capture strobe occurs in the middle of SPSCK cycle 7 (see Figure 16-5 and Figure 16-7.) If an overflow occurs, all data received after the overflow and before the OVRF bit is cleared does not transfer to the receive data register and does not set the SPI receiver full bit (SPRF). The unread data that transferred to the receive data register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPI status and control register and then reading the SPI data register.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector (see Figure 16-12.) It is not possible to enable MODF or OVRF individually to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. Figure 16-10 shows how it is possible to miss an overflow. The first part of Figure 16-10 shows how it is possible to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF bit can be set in between the time that SPSCR and SPDR are read.

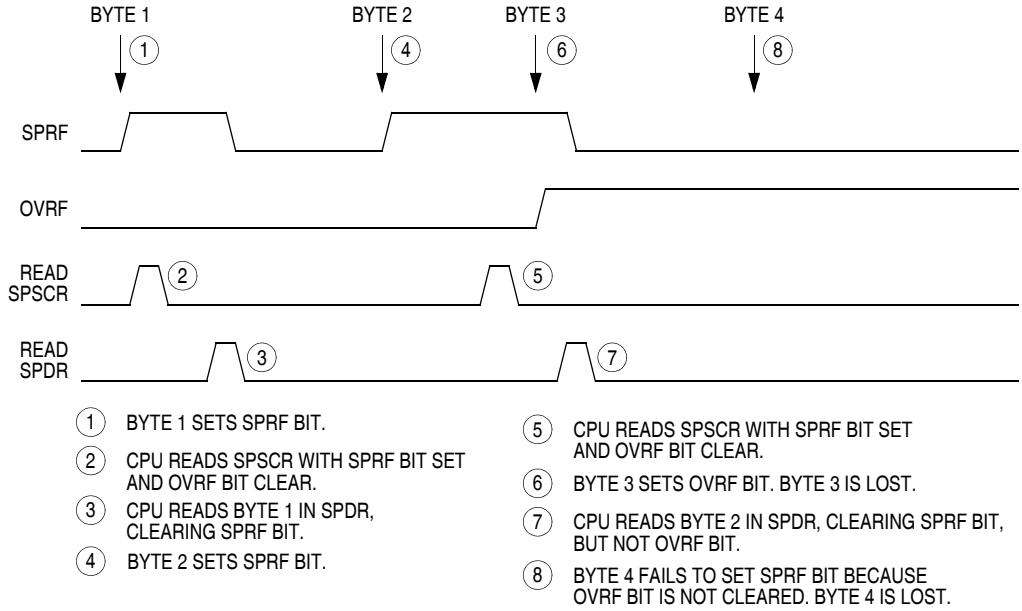



**Figure 16-10. Missed Read of Overflow Condition**

In this case, an overflow can be missed easily. Since no more SPRF interrupts can be generated until this OVRF is serviced, it is not obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR following the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions can set the SPRF bit. Figure 16-11 illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit.

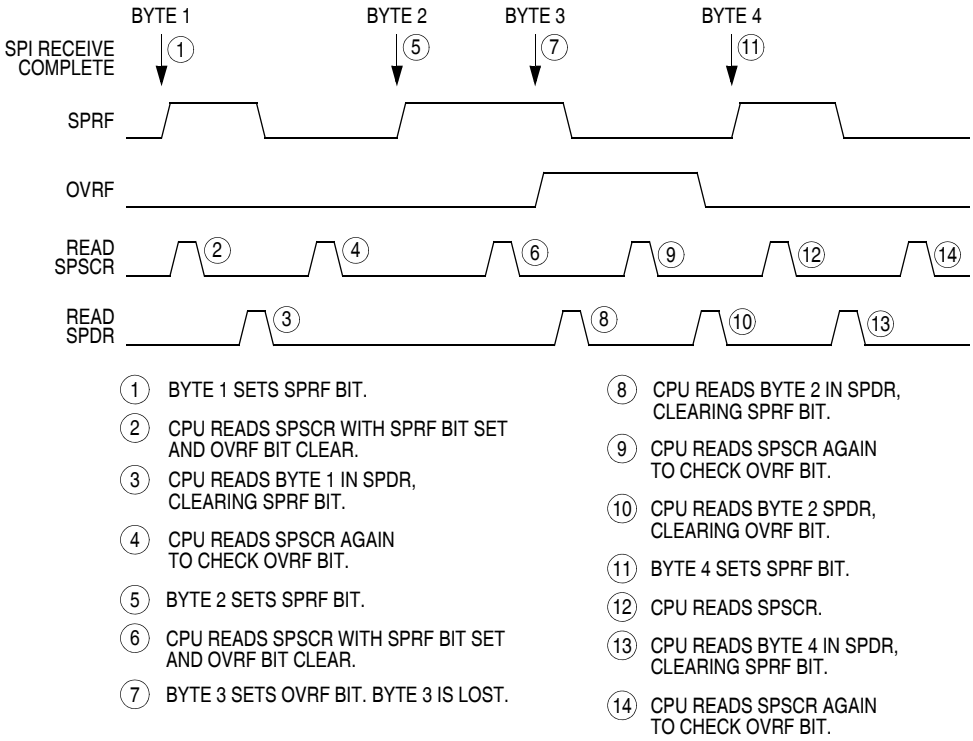



**Figure 16-11. Clearing SPRF When OVRF Interrupt Is Not Enabled**

### 16.6.2 Mode Fault Error

Setting SPMSTR selects master mode and configures the SPSCK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SPSCK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, becomes set any time the state of the slave select pin, $\overline{SS}$, is inconsistent with the mode selected by SPMSTR.

To prevent SPI pin contention and damage to the MCU, a mode fault error occurs if:

- The $\overline{SS}$ pin of a slave SPI goes high during a transmission
- The $\overline{SS}$ pin of a master SPI goes low at any time

For the MODF flag to be set, the mode fault error enable bit (MODFEN) must be set. Clearing the MODFEN bit does not clear the MODF flag but does prevent MODF from being set again after MODF is cleared.

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. (See Figure 16-12.) It is not possible to enable MODF or OVRF individually to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes low. A mode fault in a master SPI causes the following events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error CPU interrupt request.
- The SPE bit is cleared.
- The SPTE bit is set.
- The SPI state counter is cleared.
- The data direction register of the shared I/O port regains control of port drivers.

***NOTE***

*To prevent bus contention with another master SPI after a mode fault error, clear all SPI bits of the data direction register of the shared I/O port before enabling the SPI.*

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{SS}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{SS}$ goes low and ends once the incoming SPSCK goes back to its idle level following the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SPSCK returns to its idle level following the shift of the last data bit. See 16.4 Transmission Formats.

***NOTE***

*Setting the MODF flag does not clear the SPMSTR bit. SPMSTR has no function when SPE = 0. Reading SPMSTR when MODF = 1 shows the difference between a MODF occurring when the SPI is a master and when it is a slave.*

*When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is low) and later unselected ($\overline{SS}$ is high) even if no SPSCK is sent to that slave. This happens because $\overline{SS}$ low indicates the start of the transmission (MISO driven out with the value of MSB) for CPHA = 0. When CPHA = 1, a slave can be selected and then later unselected with no transmission occurring. Therefore, MODF does not occur since a transmission was never begun.*

In a slave SPI (MSTR = 0), MODF generates an SPI receiver/error CPU interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transmission by clearing the SPE bit of the slave.

***NOTE***

*A high on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.*

To clear the MODF flag, read the SPSCR with the MODF bit set and then write to the SPCR register. This entire clearing mechanism must occur with no MODF condition existing or else the flag is not cleared.

## 16.7 Interrupts

Four SPI status flags can be enabled to generate CPU interrupt requests. See Table 16-1.

**Table 16-1. SPI Interrupts**

| Flag | Request |
|---|---|
| SPTE<br>Transmitter empty | SPI transmitter CPU interrupt request<br>(SPTIE = 1, SPE = 1) |
| SPRF<br>Receiver full | SPI receiver CPU interrupt request<br>(SPRIE = 1) |
| OVRF<br>Overflow | SPI receiver/error interrupt request<br>(ERRIE = 1) |
| MODF<br>Mode fault | SPI receiver/error interrupt request<br>(ERRIE = 1) |

Reading the SPI status and control register with SPRF set and then reading the receive data register clears SPRF. The clearing mechanism for the SPTE flag is always just a write to the transmit data register.

The SPI transmitter interrupt enable bit (SPTIE) enables the SPTE flag to generate transmitter CPU interrupt requests, provided that the SPI is enabled (SPE = 1).

The SPI receiver interrupt enable bit (SPRIE) enables SPRF to generate receiver CPU interrupt requests, regardless of the state of SPE. See Figure 16-12.

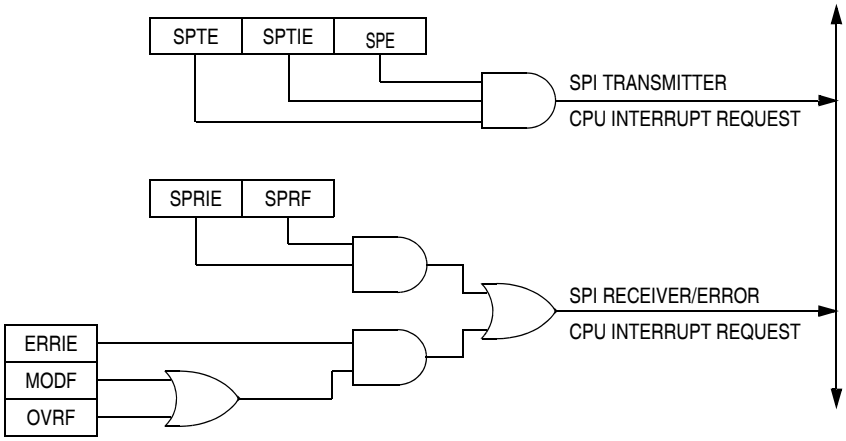



**Figure 16-12. SPI Interrupt Request Generation**

The error interrupt enable bit (ERRIE) enables both the MODF and OVRF bits to generate a receiver/error CPU interrupt request.

The mode fault enable bit (MODFEN) can prevent the MODF flag from being set so that only the OVRF bit is enabled by the ERRIE bit to generate receiver/error CPU interrupt requests.

The following sources in the SPI status and control register can generate CPU interrupt requests:

- SPI receiver full bit (SPRF) — SPRF becomes set every time a byte transfers from the shift register to the receive data register. If the SPI receiver interrupt enable bit, SPRIE, is also set, SPRF generates an SPI receiver/error CPU interrupt request.
- SPI transmitter empty (SPTE) — SPTE becomes set every time a byte transfers from the transmit data register to the shift register. If the SPI transmit interrupt enable bit, SPTIE, is also set, SPTE generates an SPTE CPU interrupt request.

## 16.8 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is 0. Whenever SPE is 0, the following occurs:

- The SPTE flag is set.
- Any transmission currently in progress is aborted.
- The shift register is cleared.
- The SPI state counter is cleared, making it ready for a new complete transmission.
- All the SPI port logic is defaulted back to being general-purpose I/O.

These items are reset only by a system reset:

- All control bits in the SPCR register
- All control bits in the SPSCR register (MODFEN, ERRIE, SPR1, and SPR0)
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, the user can clear SPE between transmissions without having to set all control bits again when SPE is set back high for the next transmission.

By not resetting the SPRF, OVRF, and MODF flags, the user can still service these interrupts after the SPI has been disabled. The user can disable the SPI by writing 0 to the SPE bit. The SPI can also be disabled by a mode fault occurring in an SPI that was configured as a master with the MODFEN bit set.

## 16.9 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

### 16.9.1 Wait Mode

The SPI module remains active after the execution of a WAIT instruction. In wait mode the SPI module registers are not accessible by the CPU. Any enabled CPU interrupt request from the SPI module can bring the MCU out of wait mode.

If SPI module functions are not required during wait mode, reduce power consumption by disabling the SPI module before executing the WAIT instruction.

To exit wait mode when an overflow condition occurs, enable the OVRF bit to generate CPU interrupt requests by setting the error interrupt enable bit (ERRIE). See 16.7 Interrupts.

### 16.9.2 Stop Mode

The SPI module is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions. SPI operation resumes after an external interrupt. If stop mode is exited by reset, any transfer in progress is aborted, and the SPI is reset.

## 16.10 SPI During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. BCFE in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. See Chapter 15 System Integration Module (SIM).

To allow software to clear status bits during a break interrupt, write a 1 to BCFE. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a 0 to BCFE. With BCFE at 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is 0. After the break, doing the second step clears the status bit.

Since the SPTE bit cannot be cleared during a break with BCFE cleared, a write to the transmit data register in break mode does not initiate a transmission nor is this data transferred into the shift register. Therefore, a write to the SPDR in break mode with BCFE cleared has no effect.

## 16.11 I/O Signals

The SPI module has four I/O pins:

- MISO — Master input/slave output
- MOSI — Master output/slave input
- SPSCK — Serial clock
- $\overline{SS}$ — Slave select

### 16.11.1 MISO (Master In/Slave Out)

MISO is one of the two SPI module pins that transmits serial data. In full duplex operation, the MISO pin of the master SPI module is connected to the MISO pin of the slave SPI module. The master SPI simultaneously receives data on its MISO pin and transmits data from its MOSI pin.

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is 0 and its $\overline{SS}$ pin is low. To support a multiple-slave system, a high on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

When enabled, the SPI controls data direction of the MISO pin regardless of the state of the data direction register of the shared I/O port.

### 16.11.2 MOSI (Master Out/Slave In)

MOSI is one of the two SPI module pins that transmits serial data. In full-duplex operation, the MOSI pin of the master SPI module is connected to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin.

When enabled, the SPI controls data direction of the MOSI pin regardless of the state of the data direction register of the shared I/O port.

### 16.11.3 SPSCK (Serial Clock)

The serial clock synchronizes data transmission between master and slave devices. In a master MCU, the SPSCK pin is the clock output. In a slave MCU, the SPSCK pin is the clock input. In full-duplex operation, the master and slave MCUs exchange a byte of data in eight serial clock cycles.

When enabled, the SPI controls data direction of the SPSCK pin regardless of the state of the data direction register of the shared I/O port.

### 16.11.4 $\overline{SS}$ (Slave Select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{SS}$ is used to select a slave. For CPHA = 0, the $\overline{SS}$ is used to define the start of a transmission. (See 16.4 Transmission Formats.) Since it is used to indicate the start of a transmission, $\overline{SS}$ must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low between transmissions for the CPHA = 1 format. See Figure 16-13.

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. It cannot be used as a general-purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of $\overline{SS}$ from creating a MODF error. See 16.12.2 SPI Status and Control Register.

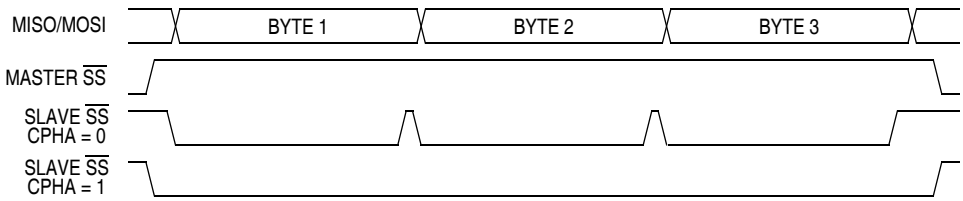



**Figure 16-13. CPHA/$\overline{SS}$ Timing**

***NOTE***

*A high on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.*

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. (See 16.6.2 Mode Fault Error.) For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If MODFEN is 0 for an SPI master, the $\overline{SS}$ pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. When MODFEN is 1, $\overline{SS}$ is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{SS}$ pin by configuring the appropriate pin as an input and reading the port data register. See Table 16-2.

**Table 16-2. SPI Configuration**

| SPE | SPMSTR | MODFEN | SPI Configuration | Function of $\overline{SS}$ Pin |
|---|---|---|---|---|
| 0 | X(1)) | X | Not enabled | General-purpose I/O; $\overline{SS}$ ignored by SPI |
| 1 | 0 | X | Slave | Input-only to SPI |
| 1 | 1 | 0 | Master without MODF | General-purpose I/O; $\overline{SS}$ ignored by SPI |
| 1 | 1 | 1 | Master with MODF | Input-only to SPI |

1. X = Don't care

## 16.12 I/O Registers

Three registers control and monitor SPI operation:

- SPI control register (SPCR)
- SPI status and control register (SPSCR)
- SPI data register (SPDR)

### 16.12.1 SPI Control Register

The SPI control register:

- Enables SPI module interrupt requests
- Configures the SPI module as master or slave
- Selects serial clock polarity and phase
- Configures the SPSCK, MOSI, and MISO pins as open-drain outputs
- Enables the SPI module

Address: $0010

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | SPRIE | R | SPMSTR | CPOL | CPHA | SPWOM | SPE | SPTIE |
| Reset: | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

R = Reserved

**Figure 16-14. SPI Control Register (SPCR)**

**SPRIE — SPI Receiver Interrupt Enable Bit**

This read/write bit enables CPU interrupt requests generated by the SPRF bit. The SPRF bit is set when a byte transfers from the shift register to the receive data register. Reset clears the SPRIE bit.

1 = SPRF CPU interrupt requests enabled
0 = SPRF CPU interrupt requests disabled

**SPMSTR — SPI Master Bit**

This read/write bit selects master mode operation or slave mode operation. Reset sets the SPMSTR bit.

1 = Master mode
0 = Slave mode

**CPOL — Clock Polarity Bit**

This read/write bit determines the logic state of the SPSCK pin between transmissions. (See Figure 16-5 and Figure 16-7.) To transmit data between SPI modules, the SPI modules must have identical CPOL values. Reset clears the CPOL bit.

**CPHA — Clock Phase Bit**

This read/write bit controls the timing relationship between the serial clock and SPI data. (See Figure 16-5 and Figure 16-7.) To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the $\overline{SS}$ pin of the slave SPI module must be high between bytes. (See Figure 16-13.) Reset sets the CPHA bit.

**SPWOM — SPI Wired-OR Mode Bit**

This read/write bit disables the pullup devices on pins SPSCK, MOSI, and MISO so that those pins become open-drain outputs.

1 = Wired-OR SPSCK, MOSI, and MISO pins
0 = Normal push-pull SPSCK, MOSI, and MISO pins

**SPE — SPI Enable**

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. (See 16.8 Resetting the SPI.) Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

**SPTIE— SPI Transmit Interrupt Enable**

This read/write bit enables CPU interrupt requests generated by the SPTE bit. SPTE is set when a byte transfers from the transmit data register to the shift register. Reset clears the SPTIE bit.

1 = SPTE CPU interrupt requests enabled
0 = SPTE CPU interrupt requests disabled

### 16.12.2 SPI Status and Control Register

The SPI status and control register contains flags to signal these conditions:

- Receive data register full
- Failure to clear SPRF bit before next byte is received (overflow error)
- Inconsistent logic level on $\overline{SS}$ pin (mode fault error)
- Transmit data register empty

The SPI status and control register also contains bits that perform these functions:

- Enable error interrupts
- Enable mode fault error detection
- Select master SPI baud rate

Address: $0011

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | SPRF | ERRIE | OVRF | MODF | SPTE | MODFEN | SPR1 | SPR0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

= Unimplemented

**Figure 16-15. SPI Status and Control Register (SPSCR)**

**SPRF — SPI Receiver Full Bit**

This clearable, read-only flag is set each time a byte transfers from the shift register to the receive data register. SPRF generates a CPU interrupt request if the SPRIE bit in the SPI control register is set also.

During an SPRF CPU interrupt, the CPU clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register.

Reset clears the SPRF bit.

1 = Receive data register full
0 = Receive data register not full

**ERRIE — Error Interrupt Enable Bit**

This read/write bit enables the MODF and OVRF bits to generate CPU interrupt requests. Reset clears the ERRIE bit.

1 = MODF and OVRF can generate CPU interrupt requests
0 = MODF and OVRF cannot generate CPU interrupt requests

**OVRF — Overflow Bit**

This clearable, read-only flag is set if software does not read the byte in the receive data register before the next full byte enters the shift register. In an overflow condition, the byte already in the receive data register is unaffected, and the byte that shifted in last is lost. Clear the OVRF bit by reading the SPI status and control register with OVRF set and then reading the receive data register. Reset clears the OVRF bit.

1 = Overflow
0 = No overflow

**MODF — Mode Fault Bit**

This clearable, read-only flag is set in a slave SPI if the $\overline{SS}$ pin goes high during a transmission with MODFEN set. In a master SPI, the MODF flag is set if the $\overline{SS}$ pin goes low at any time with the MODFEN bit set. Clear MODF by reading the SPI status and control register (SPSCR) with MODF set and then writing to the SPI control register (SPCR). Reset clears the MODF bit.

1 = $\overline{SS}$ pin at inappropriate logic level
0 = $\overline{SS}$ pin at appropriate logic level

**SPTE — SPI Transmitter Empty Bit**

This clearable, read-only flag is set each time the transmit data register transfers a byte into the shift register. SPTE generates an SPTE CPU interrupt request if SPTIE in the SPI control register is set also.

***NOTE***

*Do not write to the SPI data register unless SPTE is high.*

During an SPTE CPU interrupt, the CPU clears SPTE by writing to the transmit data register.

Reset sets the SPTE bit.

1 = Transmit data register empty
0 = Transmit data register not empty

**MODFEN — Mode Fault Enable Bit**

This read/write bit, when set, allows the MODF flag to be set. If the MODF flag is set, clearing MODFEN does not clear the MODF flag. If the SPI is enabled as a master and the MODFEN bit is 0, then the $\overline{SS}$ pin is available as a general-purpose I/O.

If the MODFEN bit is 1, then the $\overline{SS}$ pin is not available as a general-purpose I/O. When the SPI is enabled as a slave, the $\overline{SS}$ pin is not available as a general-purpose I/O regardless of the value of MODFEN. See 16.11.4 SS (Slave Select).

If the MODFEN bit is 0, the level of the $\overline{SS}$ pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation. See 16.6.2 Mode Fault Error.

**SPR1 and SPR0 — SPI Baud Rate Select Bits**

In master mode, these read/write bits select one of four baud rates as shown in Table 16-3. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

**Table 16-3. SPI Master Baud Rate Selection**

| SPR1 and SPR0 | Baud Rate Divisor (BD) |
|---|---|
| 00 | 2 |
| 01 | 8 |
| 10 | 32 |
| 11 | 128 |

Use this formula to calculate the SPI baud rate:

$$\text{Baud rate} = \frac{\text{BUSCLK}}{\text{BD}}$$

## 16.12.3 SPI Data Register

The SPI data register consists of the read-only receive data register and the write-only transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate registers that can contain different values. See Figure 16-2.

Address: $0012

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| Reset: | Unaffected by reset | | | | | | | |

**Figure 16-16. SPI Data Register (SPDR)**

**R7–R0/T7–T0 — Receive/Transmit Data Bits**

***NOTE***

*Do not use read-modify-write instructions on the SPI data register since the register read is not the same as the register written.*

# Chapter 17
# Timebase Module (TBM)

## 17.1 Introduction

This section describes the timebase module (TBM). The TBM will generate periodic interrupts at user selectable rates using a counter clocked by the external clock source. This TBM version uses 15 divider stages, eight of which are user selectable. A configuration option bit to select an additional 128 divide of the external clock source can be selected. See Chapter 5 Configuration Register (CONFIG)

## 17.2 Features

Features of the TBM module include:

- External clock or an additional divide-by-128 selected by configuration option bit as clock source
- Software configurable periodic interrupts with divide-by: 8, 16, 32, 64, 128, 2048, 8192, and 32768 taps of the selected clock source
- Configurable for operation during stop mode to allow periodic wakeup from stop

## 17.3 Functional Description

This module can generate a periodic interrupt by dividing the clock source supplied from the clock generator module, CGMXCLK.

The counter is initialized to all 0s when TBON bit is cleared. The counter, shown in Figure 17-1, starts counting when the TBON bit is set. When the counter overflows at the tap selected by TBR2–TBR0, the TBIF bit gets set. If the TBIE bit is set, an interrupt request is sent to the CPU. The TBIF flag is cleared by writing a 1 to the TACK bit. The first time the TBIF flag is set after enabling the timebase module, the interrupt is generated at approximately half of the overflow period. Subsequent events occur at the exact period.

The timebase module may remain active after execution of the STOP instruction if the crystal oscillator has been enabled to operate during stop mode through the OSCENINSTOP bit in the configuration register. The timebase module can be used in this mode to generate a periodic wakeup from stop mode.

## 17.4 Interrupts

The timebase module can periodically interrupt the CPU with a rate defined by the selected TBMCLK and the select bits TBR2–TBR0. When the timebase counter chain rolls over, the TBIF flag is set. If the TBIE bit is set, enabling the timebase interrupt, the counter chain overflow will generate a CPU interrupt request.

***NOTE***

*Interrupts must be acknowledged by writing a 1 to the TACK bit.*

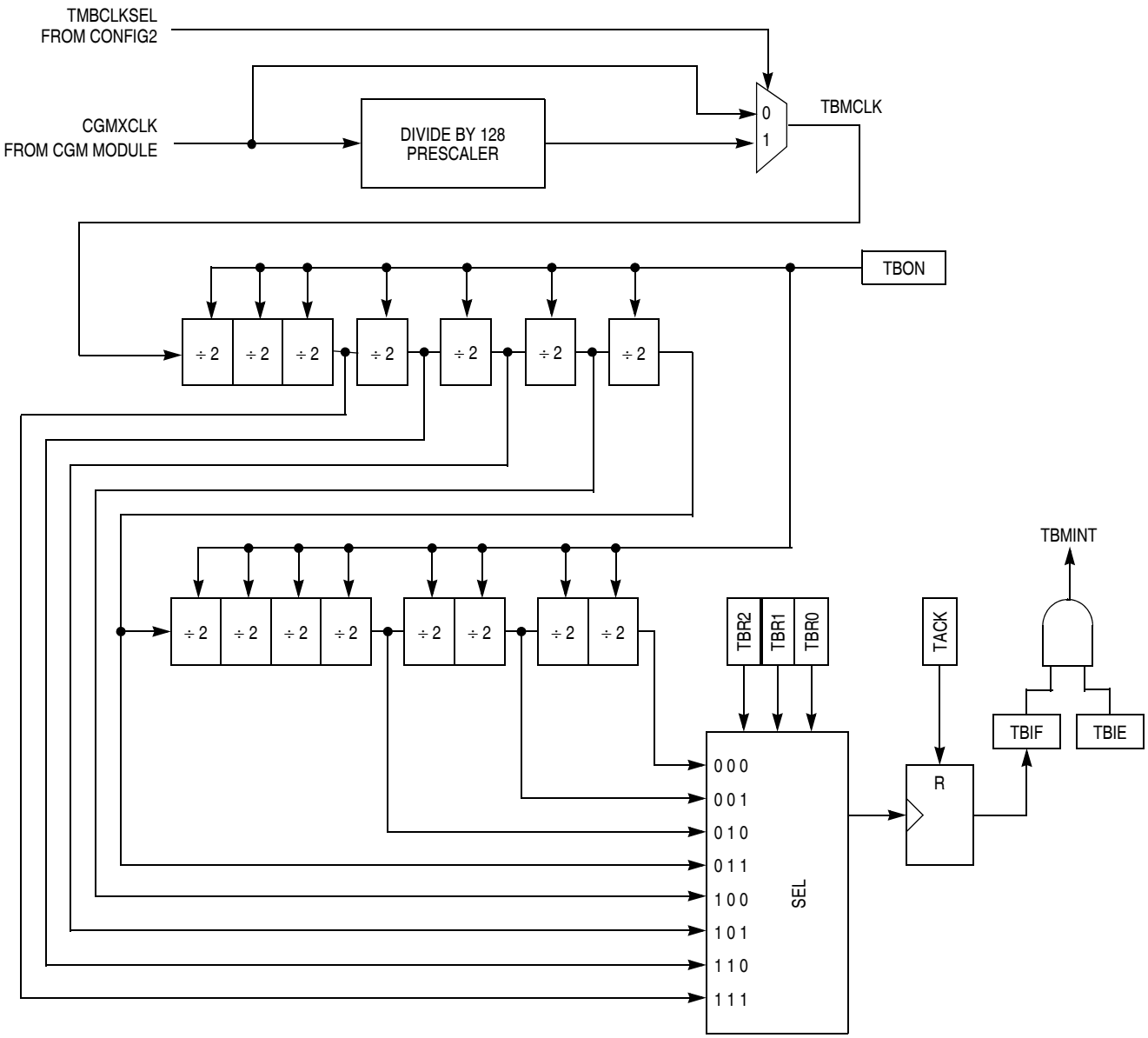



**Figure 17-1. Timebase Block Diagram**

## 17.5 TBM Interrupt Rate

The interrupt rate is determined by the equation:

$$t_{TBMRATE} = \frac{Divider}{f_{CGMXCLK}}$$

where:

$f_{CGMXCLK}$ =Frequency supplied from the clock generator (CGM) module

Divider = Divider value as determined by TBR2–TBR0 settings and TMBCLKSEL, see Table 17-1

**Table 17-1. Timebase Divider Selection**

| TBR2 | TBR1 | TBR0 | Divider | |
|---|---|---|---|---|
| | | | TMBCLKSEL | |
| | | | 0 | 1 |
| 0 | 0 | 0 | 32,768 | 4,194,304 |
| 0 | 0 | 1 | 8192 | 1,048,576 |
| 0 | 1 | 0 | 2048 | 262144 |
| 0 | 1 | 1 | 128 | 16,384 |
| 1 | 0 | 0 | 64 | 8192 |
| 1 | 0 | 1 | 32 | 4096 |
| 1 | 1 | 0 | 16 | 2048 |
| 1 | 1 | 1 | 8 | 1024 |

As an example, a 4.9152 MHz crystal, with the TMBCLKSEL set for divide-by-128 and the TBR2–TBR0 set to {011}, the divider is 16,384 and the interrupt rate calculates to:

$$\frac{16{,}384}{4.9152 \times 10^{6}} = 3.33 \text{ ms}$$

***NOTE***

*Do not change TBR2–TBR0 bits while the timebase is enabled (TBON = 1).*

## 17.6 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power- consumption standby modes.

### 17.6.1 Wait Mode

The timebase module remains active after execution of the WAIT instruction. In wait mode the timebase register is not accessible by the CPU.

If the timebase functions are not required during wait mode, reduce the power consumption by stopping the timebase before executing the WAIT instruction.

### 17.6.2 Stop Mode

The timebase module may remain active after execution of the STOP instruction if the oscillator has been enabled to operate during stop mode through the OSCENINSTOP bit in the configuration register. The timebase module can be used in this mode to generate a periodic wakeup from stop mode.

If the oscillator has not been enabled to operate in stop mode, the timebase module will not be active during stop mode. In stop mode, the timebase register is not accessible by the CPU.

If the timebase functions are not required during stop mode, reduce power consumption by disabling the timebase module before executing the STOP instruction.

## 17.7 Timebase Control Register

The timebase has one register, the timebase control register (TBCR), which is used to enable the timebase interrupts and set the rate.

Address: $001C

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | TBIF | TBR2 | TBR1 | TBR0 | 0 | TBIE | TBON | R |
| Write: | | | | | TACK | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented  R = Reserved

**Figure 17-2. Timebase Control Register (TBCR)**

**TBIF — Timebase Interrupt Flag**

This read-only flag bit is set when the timebase counter has rolled over.

1 = Timebase interrupt pending
0 = Timebase interrupt not pending

**TBR2–TBR0 — Timebase Divider Selection Bits**

These read/write bits select the tap in the counter to be used for timebase interrupts as shown in Table 17-1.

***NOTE***

*Do not change TBR2–TBR0 bits while the timebase is enabled (TBON = 1).*

**TACK— Timebase Acknowledge Bit**

The TACK bit is a write-only bit and always reads as 0. Writing a 1 to this bit clears TBIF, the timebase interrupt flag bit. Writing a 0 to this bit has no effect.

1 = Clear timebase interrupt flag
0 = No effect

**TBIE — Timebase Interrupt Enabled Bit**

This read/write bit enables the timebase interrupt when the TBIF bit becomes set. Reset clears the TBIE bit.

1 = Timebase interrupt is enabled.
0 = Timebase interrupt is disabled.

**TBON — Timebase Enabled Bit**

This read/write bit enables the timebase. Timebase may be turned off to reduce power consumption when its function is not necessary. The counter can be initialized by clearing and then setting this bit. Reset clears the TBON bit.

1 = Timebase is enabled.
0 = Timebase is disabled and the counter initialized to 0s.

# Chapter 18
# Timer Interface Module (TIM1)

## 18.1 Introduction

This section describes the timer interface module (TIM1). TIM1 is a two-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. Figure 18-2 is a block diagram of the TIM1.

## 18.2 Features

Features of the TIM1 include the following:

- Two input capture/output compare channels
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
- Buffered and unbuffered pulse width modulation (PWM) signal generation
- Programmable TIM1 clock input with 7-frequency internal bus clock prescaler selection
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIM1 counter stop and reset bits

## 18.3 Functional Description

Figure 18-2 shows the structure of the TIM1. The central component of the TIM1 is the 16-bit TIM1 counter that can operate as a free-running counter or a modulo up-counter. The TIM1 counter provides the timing reference for the input capture and output compare functions. The TIM1 counter modulo registers, T1MODH:T1MODL, control the modulo value of the TIM1 counter. Software can read the TIM1 counter value at any time without affecting the counting sequence.

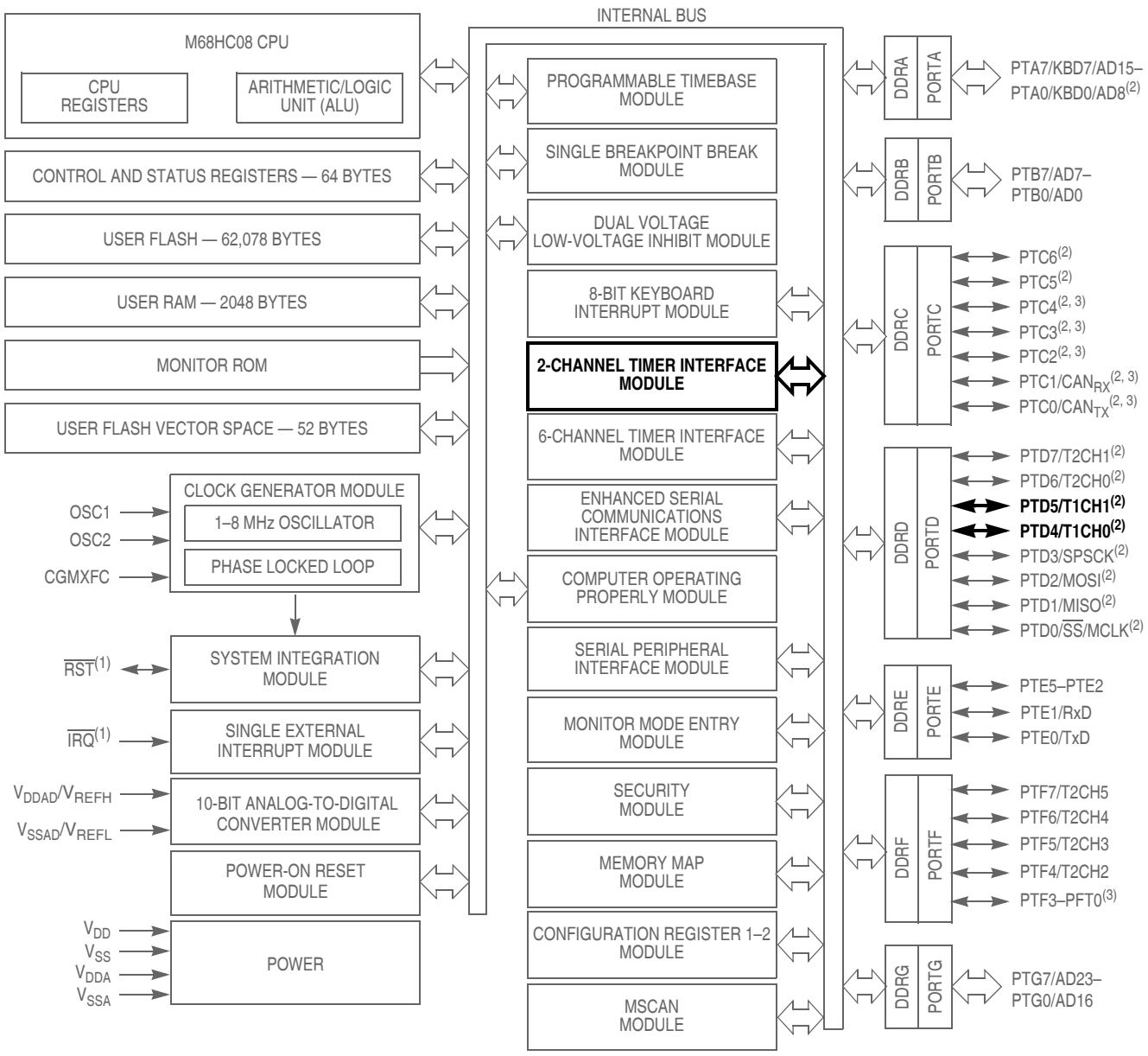
The two TIM1 channels are programmable independently as input capture or output compare channels.

### 18.3.1 TIM1 Counter Prescaler

The TIM1 clock source is one of the seven prescaler outputs. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIM1 status and control register (T1SC) select the TIM1 clock source.

### 18.3.2 Input Capture

With the input capture function, the TIM1 can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TIM1 latches the contents of the TIM1 counter into the TIM1 channel registers, T1CHxH:T1CHxL. The polarity of the active edge is programmable. Input captures can generate TIM1 central processor unit (CPU) interrupt requests.




1. Pin contains integrated pullup device.
**2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.**
3. Higher current drive port pins

**Figure 18-1. Block Diagram Highlighting TIM1 Block and Pins**

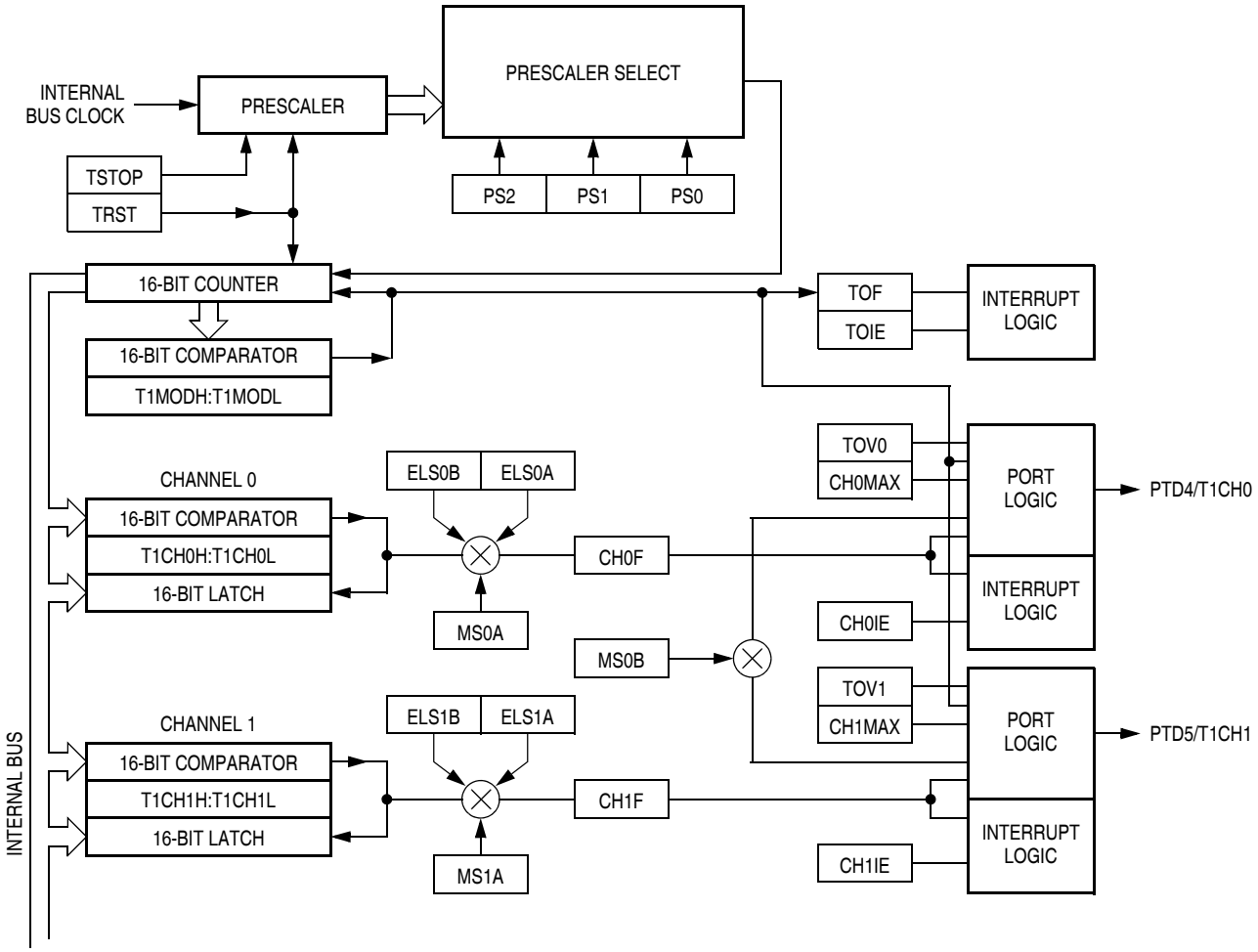



**Figure 18-2. TIM1 Block Diagram**

## 18.3.3 Output Compare

With the output compare function, the TIM1 can generate a periodic pulse with a programmable polarity, duration, and frequency. When the counter reaches the value in the registers of an output compare channel, the TIM1 can set, clear, or toggle the channel pin. Output compares can generate TIM1 CPU interrupt requests.

### *18.3.3.1 Unbuffered Output Compare*

Any output compare channel can generate unbuffered output compare pulses as described in 18.3.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM1 channel registers.

An unsynchronized write to the TIM1 channel registers to change an output compare value could cause incorrect operation for up to two counter overflow periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that counter overflow period. Also, using a TIM1 overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIM1 may pass the new value before it is written.

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0020 | TIM1 Status and Control Register (T1SC) See page 271. | Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $0021 | TIM1 Counter Register High (T1CNTH) See page 273. | Read: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0022 | TIM1 Counter Register Low (T1CNTL) See page 273. | Read: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0023 | TIM1 Counter Modulo Register High (T1MODH) See page 273. | Read: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $0024 | TIM1 Counter Modulo Register Low (T1MODL) See page 273. | Read: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $0025 | TIM1 Channel 0 Status and Control Register (T1SC0) See page 274. | Read: | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0026 | TIM1 Channel 0 Register High (T1CH0H) See page 277. | Read: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0027 | TIM1 Channel 0 Register Low (T1CH0L) See page 277. | Read: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0028 | TIM1 Channel 1 Status and Control Register (T1SC1) See page 274. | Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0029 | TIM1 Channel 1 Register High (T1CH1H) See page 277. | Read: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $002A | TIM1 Channel 1 Register Low (T1CH1L) See page 277. | Read: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |

= Unimplemented

**Figure 18-3. TIM1 I/O Register Summary**

Use the following methods to synchronize unbuffered changes in the output compare value on channel x:

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable TIM1 overflow interrupts and write the new value in the TIM1 overflow interrupt routine. The TIM1 overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

#### *18.3.3.2 Buffered Output Compare*

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the T1CH0 pin. The TIM1 channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIM1 channel 0 status and control register (TSC0) links channel 0 and channel 1. The output compare value in the TIM1 channel 0 registers initially controls the output on the T1CH0 pin. Writing to the TIM1 channel 1 registers enables the TIM1 channel 1 registers to synchronously control the output after the TIM1 overflows. At each subsequent overflow, the TIM1 channel registers (0 or 1) that control the output are the ones written to last. T1SC0 controls and monitors the buffered output compare function, and TIM1 channel 1 status and control register (T1SC1) is unused. While the MS0B bit is set, the channel 1 pin, T1CH1, is available as a general-purpose I/O pin.

***NOTE***

*In buffered output compare operation, do not write new output compare values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered output compares.*

### 18.3.4 Pulse Width Modulation (PWM)

By using the toggle-on-overflow feature with an output compare channel, the TIM1 can generate a PWM signal. The value in the TIM1 counter modulo registers determines the period of the PWM signal. The channel pin toggles when the counter reaches the value in the TIM1 counter modulo registers. The time between overflows is the period of the PWM signal.

As Figure 18-4 shows, the output compare value in the TIM1 channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIM1 to clear the channel pin on output compare if the polarity of the PWM pulse is 1 (ELSxA = 0). Program the TIM1 to set the pin if the polarity of the PWM pulse is 0 (ELSxA = 1).

The value in the TIM1 counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIM1 counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is 000. See 18.8.1 TIM1 Status and Control Register.

The value in the TIM1 channel registers determines the pulse width of the PWM output. The pulse width of an 8-bit PWM signal is variable in 256 increments. Writing $0080 (128) to the TIM1 channel registers produces a duty cycle of 128/256 or 50%.

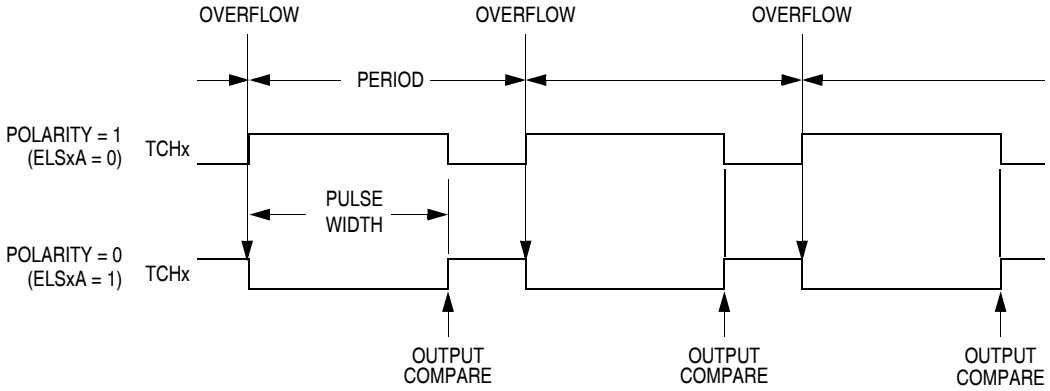



**Figure 18-4. PWM Period and Pulse Width**

### *18.3.4.1 Unbuffered PWM Signal Generation*

Any output compare channel can generate unbuffered PWM pulses as described in 18.3.4 Pulse Width Modulation (PWM). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM1 channel registers.

An unsynchronized write to the TIM1 channel registers to change a pulse width value could cause incorrect operation for up to two PWM periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that PWM period. Also, using a TIM1 overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIM1 may pass the new value before it is written to the timer channel (T1CHxH:T1CHxL) registers.

Use the following methods to synchronize unbuffered changes in the PWM pulse width on channel x:

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable TIM1 overflow interrupts and write the new value in the TIM1 overflow interrupt routine. The TIM1 overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

***NOTE***

*In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare also can cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

#### 18.3.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the T1CH0 pin. The TIM1 channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIM1 channel 0 status and control register (T1SC0) links channel 0 and channel 1. The TIM1 channel 0 registers initially control the pulse width on the T1CH0 pin. Writing to the TIM1 channel 1 registers enables the TIM1 channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM1 channel registers (0 or 1) that control the pulse width are the ones written to last. T1SC0 controls and monitors the buffered PWM function, and TIM1 channel 1 status and control register (T1SC1) is unused. While the MS0B bit is set, the channel 1 pin, T1CH1, is available as a general-purpose I/O pin.

**NOTE**

*In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.*

#### 18.3.4.3 PWM Initialization

To ensure correct operation when generating unbuffered or buffered PWM signals, use the following initialization procedure:

1. In the TIM1 status and control register (T1SC):
   a. Stop the TIM1 counter by setting the TIM1 stop bit, TSTOP.
   b. Reset the TIM1 counter and prescaler by setting the TIM1 reset bit, TRST.
2. In the TIM1 counter modulo registers (T1MODH:T1MODL), write the value for the required PWM period.
3. In the TIM1 channel x registers (T1CHxH:T1CHxL), write the value for the required pulse width.
4. In TIM1 channel x status and control register (T1SCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. See Table 18-2.
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (polarity 1 — to clear output on compare) or 1:1 (polarity 0 — to set output on compare) to the edge/level select bits, ELSxB:ELSxA. The output action on compare must force the output to the complement of the pulse width level. See Table 18-2.

   **NOTE**

   *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*
5. In the TIM1 status control register (T1SC), clear the TIM1 stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIM1 channel 0 registers (TCH0H:TCH0L) initially control the buffered PWM output. TIM1 status control

register 0 (TSCR0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIM1 overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and setting the TOVx bit generates a 100% duty cycle output. See 18.8.4 TIM1 Channel Status and Control Registers.

## 18.4 Interrupts

The following TIM1 sources can generate interrupt requests:

- TIM1 overflow flag (TOF) — The TOF bit is set when the TIM1 counter reaches the modulo value programmed in the TIM1 counter modulo registers. The TIM1 overflow interrupt enable bit, TOIE, enables TIM1 overflow CPU interrupt requests. TOF and TOIE are in the TIM1 status and control register.
- TIM1 channel flags (CH1F:CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE. Channel x TIM CPU interrupt requests are enabled when CHxIE =1. CHxF and CHxIE are in the TIM1 channel x status and control register.

## 18.5 Wait Mode

The WAIT instruction puts the MCU in low power-consumption standby mode.

The TIM1 remains active after the execution of a WAIT instruction. In wait mode the TIM1 registers are not accessible by the CPU. Any enabled CPU interrupt request from the TIM1 can bring the MCU out of wait mode.

If TIM1 functions are not required during wait mode, reduce power consumption by stopping the TIM1 before executing the WAIT instruction.

## 18.6 TIM1 During Break Interrupts

A break interrupt stops the TIM1 counter and inhibits input captures.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See Figure 15-21. Break Status Register (BSR).

To allow software to clear status bits during a break interrupt, write a 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a 0 to the BCFE bit. With BCFE at 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a two-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at 0. After the break, doing the second step clears the status bit.

## 18.7 Input/Output Signals

Port D shares two of its pins with the TIM1. The two TIM1 channel I/O pins are PTD4/T1CH0 and PTD5/T1CH1.

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTD4/T1CH0 can be configured as a buffered output compare or buffered PWM pin.

## 18.8 Input/Output Registers

The following I/O registers control and monitor operation of the TIM:

- TIM1 status and control register (T1SC)
- TIM1 counter registers (T1CNTH:T1CNTL)
- TIM1 counter modulo registers (T1MODH:T1MODL)
- TIM1 channel status and control registers (T1SC0 and T1SC1)
- TIM1 channel registers (T1CH0H:T1CH0L and T1CH1H:T1CH1L)

### 18.8.1 TIM1 Status and Control Register

The TIM1 status and control register (T1SC) does the following:

- Enables TIM1 overflow interrupts
- Flags TIM1 overflows
- Stops the TIM1 counter
- Resets the TIM1 counter
- Prescales the TIM1 counter clock

Address: $0020

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| Write: | 0 | | | TRST | | | | |
| Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 18-5. TIM1 Status and Control Register (T1SC)**

**TOF — TIM1 Overflow Flag Bit**

This read/write flag is set when the TIM1 counter reaches the modulo value programmed in the TIM1 counter modulo registers. Clear TOF by reading the TIM1 status and control register when TOF is set and then writing a 0 to TOF. If another TIM1 overflow occurs before the clearing sequence is complete, then writing 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a 1 to TOF has no effect.

1 = TIM1 counter has reached modulo value
0 = TIM1 counter has not reached modulo value

**TOIE — TIM1 Overflow Interrupt Enable Bit**

This read/write bit enables TIM1 overflow interrupts when the TOF bit becomes set. Reset clears the TOIE bit.

1 = TIM1 overflow interrupts enabled
0 = TIM1 overflow interrupts disabled

### TSTOP — TIM1 Stop Bit

This read/write bit stops the TIM1 counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the TIM1 counter until software clears the TSTOP bit.

1 = TIM1 counter stopped
0 = TIM1 counter active

***NOTE***

*Do not set the TSTOP bit before entering wait mode if the TIM1 is required to exit wait mode. Also, when the TSTOP bit is set and the timer is configured for input capture operation, input captures are inhibited until the TSTOP bit is cleared.*

### TRST — TIM1 Reset Bit

Setting this write-only bit resets the TIM1 counter and the TIM1 prescaler. Setting TRST has no effect on any other registers. Counting resumes from $0000. TRST is cleared automatically after the TIM1 counter is reset and always reads as 0. Reset clears the TRST bit.

1 = Prescaler and TIM1 counter cleared
0 = No effect

***NOTE***

*Setting the TSTOP and TRST bits simultaneously stops the TIM1 counter at a value of $0000.*

### PS[2:0] — Prescaler Select Bits

These read/write bits select one of the seven prescaler outputs as the input to the TIM1 counter as Table 18-1 shows. Reset clears the PS[2:0] bits.

**Table 18-1. Prescaler Selection**

| PS2 | PS1 | PS0 | TIM1 Clock Source |
|---|---|---|---|
| 0 | 0 | 0 | Internal bus clock ÷ 1 |
| 0 | 0 | 1 | Internal bus clock ÷ 2 |
| 0 | 1 | 0 | Internal bus clock ÷ 4 |
| 0 | 1 | 1 | Internal bus clock ÷ 8 |
| 1 | 0 | 0 | Internal bus clock ÷ 16 |
| 1 | 0 | 1 | Internal bus clock ÷ 32 |
| 1 | 1 | 0 | Internal bus clock ÷ 64 |
| 1 | 1 | 1 | Not available |

### 18.8.2 TIM1 Counter Registers

The two read-only TIM1 counter registers contain the high and low bytes of the value in the TIM1 counter. Reading the high byte (T1CNTH) latches the contents of the low byte (T1CNTL) into a buffer. Subsequent reads of T1CNTH do not affect the latched T1CNTL value until T1CNTL is read. Reset clears the TIM1 counter registers. Setting the TIM1 reset bit (TRST) also clears the TIM1 counter registers.

***NOTE***

*If you read T1CNTH during a break interrupt, be sure to unlatch T1CNTL by reading T1CNTL before exiting the break interrupt. Otherwise, T1CNTL retains the value latched during the break.*

Address: $0021 T1CNTH

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: $0022 T1CNTL

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 18-6. TIM1 Counter Registers (T1CNTH:T1CNTL)**

### 18.8.3 TIM1 Counter Modulo Registers

The read/write TIM1 modulo registers contain the modulo value for the TIM1 counter. When the TIM1 counter reaches the modulo value, the overflow flag (TOF) becomes set, and the TIM1 counter resumes counting from $0000 at the next timer clock. Writing to the high byte (T1MODH) inhibits the TOF bit and overflow interrupts until the low byte (T1MODL) is written. Reset sets the TIM1 counter modulo registers.

Address: $0023 T1MODH

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Address: $0024 T1MODL

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 18-7. TIM1 Counter Modulo Registers (T1MODH:T1MODL)**

***NOTE***

*Reset the TIM1 counter before writing to the TIM1 counter modulo registers.*

### 18.8.4 TIM1 Channel Status and Control Registers

Each of the TIM1 channel status and control registers does the following:

- Flags input captures and output compares
- Enables input capture and output compare interrupts
- Selects input capture, output compare, or PWM operation
- Selects high, low, or toggling output on output compare
- Selects rising edge, falling edge, or any edge as the active input capture trigger
- Selects output toggling on TIM1 overflow
- Selects 0% and 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Address: $0025 T1SC0

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| Write: | 0 | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: $0028 T1SC1

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| Write: | 0 | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 18-8. TIM1 Channel Status and Control Registers (T1SC0:T1SC1)**

**CHxF — Channel x Flag Bit**

When channel x is an input capture channel, this read/write bit is set when an active edge occurs on the channel x pin. When channel x is an output compare channel, CHxF is set when the value in the TIM1 counter registers matches the value in the TIM1 channel x registers.

Clear CHxF by reading the TIM1 channel x status and control register with CHxF set and then writing a 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a 1 to CHxF has no effect.

1 = Input capture or output compare on channel x
0 = No input capture or output compare on channel x

**CHxIE — Channel x Interrupt Enable Bit**

This read/write bit enables TIM1 CPU interrupt service requests on channel x. Reset clears the CHxIE bit.

1 = Channel x CPU interrupt requests enabled
0 = Channel x CPU interrupt requests disabled

### MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIM1 channel 0 status and control register.

Setting MS0B disables the channel 1 status and control register and reverts T1CH1 to general-purpose I/O.

Reset clears the MSxB bit.

1 = Buffered output compare/PWM operation enabled
0 = Buffered output compare/PWM operation disabled

### MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. See Table 18-2.

1 = Unbuffered output compare/PWM operation
0 = Input capture operation

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin (see Table 18-2). Reset clears the MSxA bit.

1 = Initial output level low
0 = Initial output level high

***NOTE***

*Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIM1 status and control register (T1SC).*

**Table 18-2. Mode, Edge, and Level Selection**

| MSxB | MSxA | ELSxB | ELSxA | Mode | Configuration |
|---|---|---|---|---|---|
| X | 0 | 0 | 0 | Output preset | Pin under port control; initial output level high |
| X | 1 | 0 | 0 | | Pin under port control; initial output level low |
| 0 | 0 | 0 | 1 | Input capture | Capture on rising edge only |
| 0 | 0 | 1 | 0 | | Capture on falling edge only |
| 0 | 0 | 1 | 1 | | Capture on rising or falling edge |
| 0 | 1 | 0 | 0 | Output compare or PWM | Software compare only |
| 0 | 1 | 0 | 1 | | Toggle output on compare |
| 0 | 1 | 1 | 0 | | Clear output on compare |
| 0 | 1 | 1 | 1 | | Set output on compare |
| 1 | X | 0 | 1 | Buffered output compare or buffered PWM | Toggle output on compare |
| 1 | X | 1 | 0 | | Clear output on compare |
| 1 | X | 1 | 1 | | Set output on compare |

### ELSxB and ELSxA — Edge/Level Select Bits

When channel x is an input capture channel, these read/write bits control the active edge-sensing logic on channel x.

When channel x is an output compare channel, ELSxB and ELSxA control the channel x output behavior when an output compare occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to an I/O port, and pin TCHx is available as a general-purpose I/O pin. Table 18-2 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

**NOTE**

*After initially enabling a TIM1 channel register for input capture operation and selecting the edge sensitivity, clear CHxF to ignore any erroneous edge detection flags.*

**TOVx — Toggle-On-Overflow Bit**

When channel x is an output compare channel, this read/write bit controls the behavior of the channel x output when the TIM1 counter overflows. When channel x is an input capture channel, TOVx has no effect. Reset clears the TOVx bit.

1 = Channel x pin toggles on TIM1 counter overflow.
0 = Channel x pin does not toggle on TIM1 counter overflow.

**NOTE**

*When TOVx is set, a TIM1 counter overflow takes precedence over a channel x output compare if both occur at the same time.*

**CHxMAX — Channel x Maximum Duty Cycle Bit**

When the TOVx bit is at 1, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 18-9 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.

**NOTE**

*The 100% PWM duty cycle is defined as a continuous high level if the PWM polarity is 1 and a continuous low level if the PWM polarity is 0. Conversely, a 0% PWM duty cycle is defined as a continuous low level if the PWM polarity is 1 and a continuous high level if the PWM polarity is 0.*

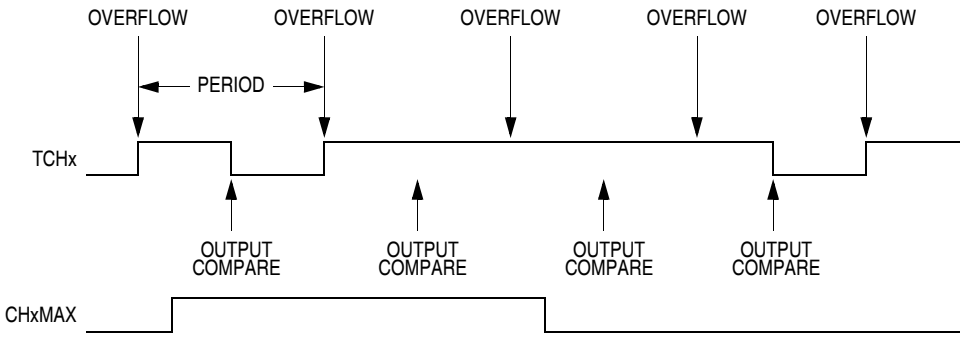



**Figure 18-9. CHxMAX Latency**

## 18.8.5 TIM1 Channel Registers

These read/write registers contain the captured TIM1 counter value of the input capture function or the output compare value of the output compare function. The state of the TIM1 channel registers after reset is unknown.

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the TIM1 channel x registers (T1CHxH) inhibits input captures until the low byte (T1CHxL) is read.

In output compare mode (MSxB:MSxA ≠ 0:0), writing to the high byte of the TIM1 channel x registers (T1CHxH) inhibits output compares until the low byte (T1CHxL) is written.

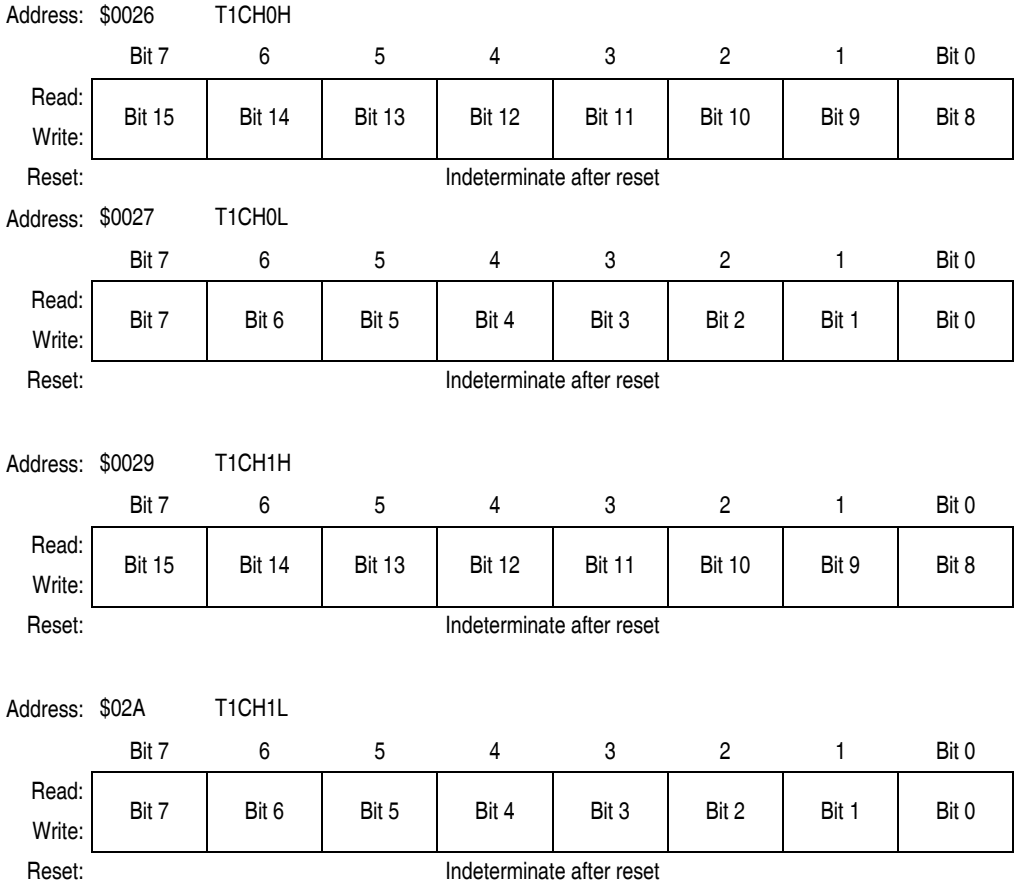



**Figure 18-10. TIM1 Channel Registers (T1CH0H/L:T1CH1H/L)**

# Chapter 19
# Timer Interface Module (TIM2)

## 19.1 Introduction

This section describes the timer interface module (TIM2). The TIM2 is a 6-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. Figure 19-2 is a block diagram of the TIM2.

## 19.2 Features

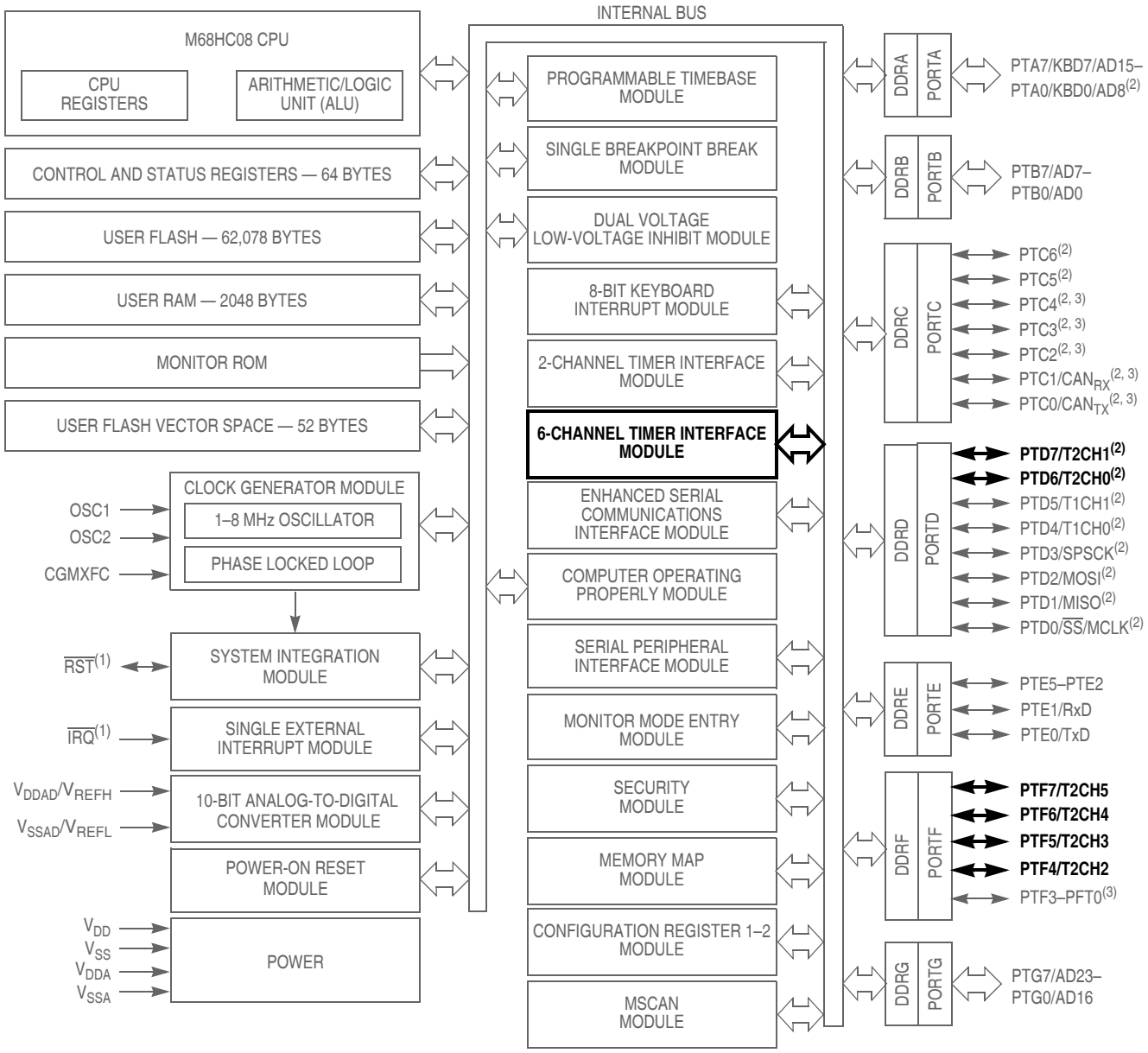
Features of the TIM2 include:

- Six input capture/output compare channels:
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
- Buffered and unbuffered pulse width modulation (PWM) signal generation
- Programmable TIM2 clock input
  - 7-frequency internal bus clock prescaler selection
  - External TIM2 clock input (4-MHz maximum frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIM2 counter stop and reset bits

## 19.3 Functional Description

Figure 19-2 shows the TIM2 structure. The central component of the TIM2 is the 16-bit TIM2 counter that can operate as a free-running counter or a modulo up-counter. The TIM2 counter provides the timing reference for the input capture and output compare functions. The TIM2 counter modulo registers, T2MODH:T2MODL, control the modulo value of the TIM2 counter. Software can read the TIM2 counter value at any time without affecting the counting sequence.

The six TIM2 channels are programmable independently as input capture or output compare channels.




1. Pin contains integrated pullup device.
**2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.**
3. Higher current drive port pins

**Figure 19-1. Block Diagram Highlighting TIM2 Block and Pins**

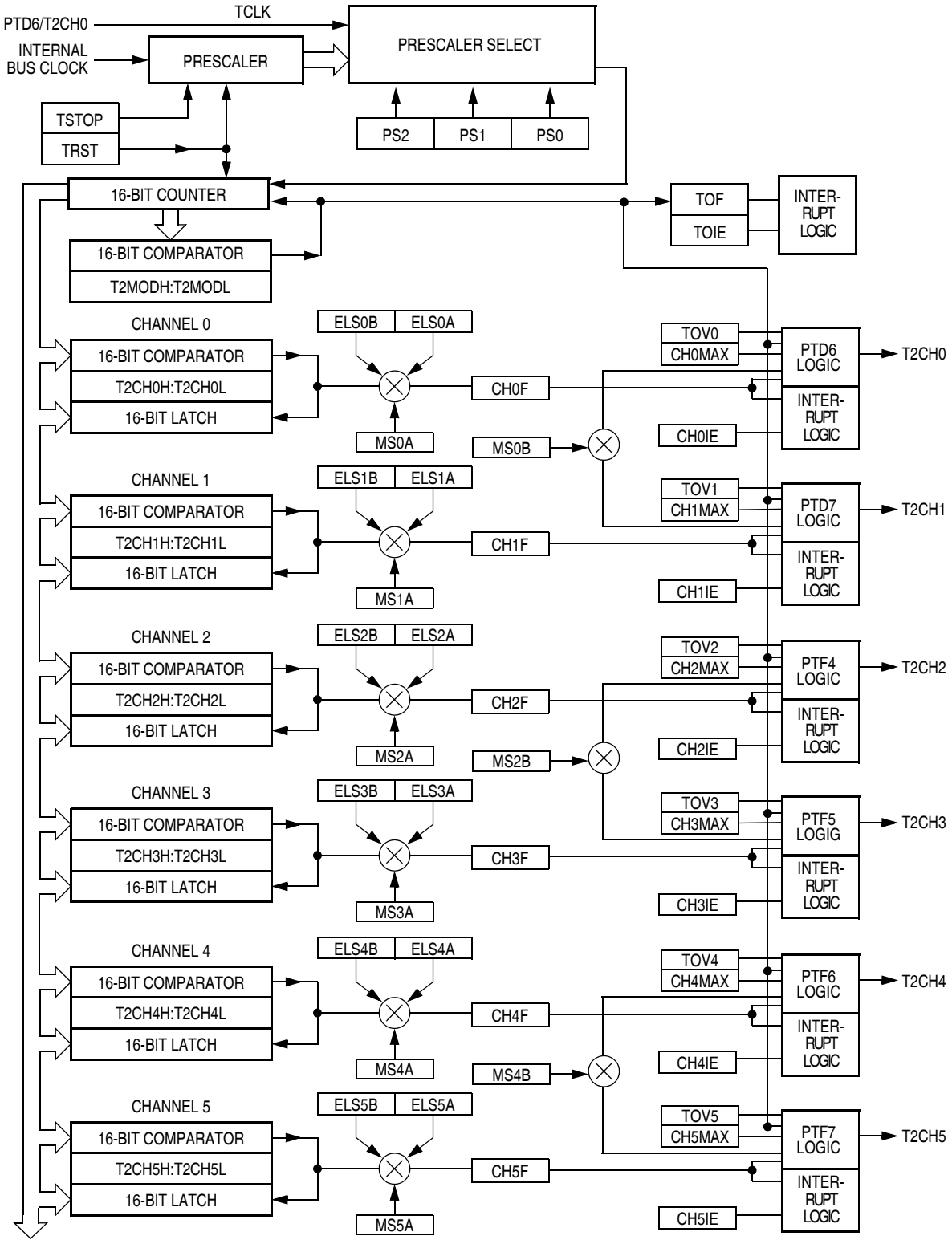



**Figure 19-2. TIM2 Block Diagram**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $002B | TIM2 Status and Control Register (T2SC) See page 291. | Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $002C | TIM2 Counter Register High (T2CNTH) See page 292. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $002D | TIM2 Counter Register Low (T2CNTL) See page 292. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $002E | TIM2 Modulo Register High (T2MODH) See page 293. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $002F | TIM2 Modulo Register Low (T2MODL) See page 293. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $0030 | TIM2 Channel 0 Status and Control Register (T2SC0) See page 293. | Read: | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0031 | TIM2 Channel 0 Register High (T2CH0H) See page 297. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0032 | TIM2 Channel 0 Register Low (T2CH0L) See page 297. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0033 | TIM2 Channel 1 Status and Control Register (T2SC1) See page 293. | Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0034 | TIM2 Channel 1 Register High (T2CH1H) See page 297. | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0035 | TIM2 Channel 1 Register Low (T2CH1L) See page 297. | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0456 | TIM2 Channel 2 Status and Control Register (T2SC2) See page 293. | Read: | CH2F | CH2IE | MS2B | MS2A | ELS2B | ELS2A | TOV2 | CH2MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 19-3. TIM2 I/O Register Summary (Sheet 1 of 2)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0457 | TIM2 Channel 2 Register High (T2CH2H) See page 297. | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0458 | TIM2 Channel 2 Register Low (T2CH2L) See page 297. | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0459 | TIM2 Channel 3 Status and Control Register (T2SC3) See page 293. | Read:<br>Write: | CH3F<br>0 | CH3IE | 0 | MS3A | ELS3B | ELS3A | TOV3 | CH3MAX |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $045A | TIM2 Channel 3 Register High (T2CH3H) See page 297. | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $045B | TIM2 Channel 3 Register Low (T2CH3L) See page 297. | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $045C | TIM2 Channel 4 Status and Control Register (T2SC4) See page 293. | Read:<br>Write: | CH4F<br>0 | CH4IE | MS4B | MS4A | ELS4B | ELS4A | TOV4 | CH4MAX |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $045D | TIM2 Channel 4 Register High (T2CH4H) See page 297. | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $045E | TIM2 Channel 4 Register Low (T2CH4L) See page 297. | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $045F | TIM2 Channel 5 Status and Control Register (T2SC5) See page 293. | Read:<br>Write: | CH5F<br>0 | CH5IE | 0 | MS5A | ELS5B | ELS5A | TOV5 | CH5MAX |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0460 | TIM2 Channel 5 Register High (T2CH5H) See page 297. | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0461 | TIM2 Channel 5 Register Low (T2CH5L) See page 297. | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Reset: | Indeterminate after reset | | | | | | | |

= Unimplemented

**Figure 19-3. TIM2 I/O Register Summary (Sheet 2 of 2)**

### 19.3.1 TIM2 Counter Prescaler

The TIM2 clock source can be one of the seven prescaler outputs or the TIM2 clock pin, T2CH0. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIM2 status and control register select the TIM2 clock source.

### 19.3.2 Input Capture

An input capture function has three basic parts: edge select logic, an input capture latch, and a 16-bit counter. Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in T2SC0 through T2SC5 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIM2 latches the contents of the TIM2 counter into the TIM2 channel registers, T2CHxH:T2CHxL. Input captures can generate TIM2 CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The free-running counter contents are transferred to the TIM2 channel registers (T2CHxH:T2CHxL) (see 19.8.5 TIM2 Channel Registers) on each proper signal transition regardless of whether the TIM2 channel flag (CH0F–CH5F in T2SC0–T2SC5 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register when the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see 19.8.5 TIM2 Channel Registers). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the input capture channel (T2CHxH:T2CHxL) registers.

### 19.3.3 Output Compare

With the output compare function, the TIM2 can generate a periodic pulse with a programmable polarity, duration, and frequency. When the counter reaches the value in the registers of an output compare channel, the TIM2 can set, clear, or toggle the channel pin. Output compares can generate TIM2 CPU interrupt requests.

#### 19.3.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in 19.3.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM2 channel registers.

An unsynchronized write to the TIM2 channel registers to change an output compare value could cause incorrect operation for up to two counter overflow periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that counter overflow period. Also, using a TIM2 overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIM2 may pass the new value before it is written.

Use the following methods to synchronize unbuffered changes in the output compare value on channel x:

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable TIM2 overflow interrupts and write the new value in the TIM2 overflow interrupt routine. The TIM2 overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

#### 19.3.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the T2CH0 pin. The TIM2 channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIM2 channel 0 status and control register (T2SC0) links channel 0 and channel 1. The output compare value in the TIM2 channel 0 registers initially controls the output on the T2CH0 pin. Writing to the TIM2 channel 1 registers enables the TIM2 channel 1 registers to synchronously control the output after the TIM2 overflows. At each subsequent overflow, the TIM2 channel registers (0 or 1) that control the output are the ones written to last. T2SC0 controls and monitors the buffered output compare function, and TIM2 channel 1 status and control register (T2SC1) is unused. While the MS0B bit is set, the channel 1 pin, T2CH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered output compare channel whose output appears on the T2CH2 pin. The TIM2 channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIM2 channel 2 status and control register (T2SC2) links channel 2 and channel 3. The output compare value in the TIM2 channel 2 registers initially controls the output on the T2CH2 pin. Writing to the TIM2 channel 3 registers enables the TIM2 channel 3 registers to synchronously control the output after the TIM2 overflows. At each subsequent overflow, the TIM2 channel registers (2 or 3) that control the output are the ones written to last. T2SC2 controls and monitors the buffered output compare function, and TIM2 channel 3 status and control register (T2SC3) is unused. While the MS2B bit is set, the channel 3 pin, T2CH3, is available as a general-purpose I/O pin.

Channels 4 and 5 can be linked to form a buffered output compare channel whose output appears on the T2CH4 pin. The TIM2 channel registers of the linked pair alternately control the output.

Setting the MS4B bit in TIM2 channel 4 status and control register (T2SC4) links channel 4 and channel 5. The output compare value in the TIM2 channel 4 registers initially controls the output on the T2CH4 pin.

Writing to the TIM2 channel 5 registers enables the TIM2 channel 5 registers to synchronously control the output after the TIM2 overflows. At each subsequent overflow, the TIM2 channel registers (4 or 5) that control the output are the ones written to last. T2SC4 controls and monitors the buffered output compare function, and TIM2 channel 5 status and control register (T2SC5) is unused. While the MS4B bit is set, the channel 5 pin, T2CH5, is available as a general-purpose I/O pin.

***NOTE***

*In buffered output compare operation, do not write new output compare values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered output compares.*

### 19.3.4 Pulse Width Modulation (PWM)

By using the toggle-on-overflow feature with an output compare channel, the TIM2 can generate a PWM signal. The value in the TIM2 counter modulo registers determines the period of the PWM signal. The channel pin toggles when the counter reaches the value in the TIM2 counter modulo registers. The time between overflows is the period of the PWM signal.

As Figure 19-4 shows, the output compare value in the TIM2 channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIM2 to clear the channel pin on output compare if the polarity of the PWM pulse is 1 (ELSxA = 0). Program the TIM2 to set the pin if the polarity of the PWM pulse is 0 (ELSxA = 1).

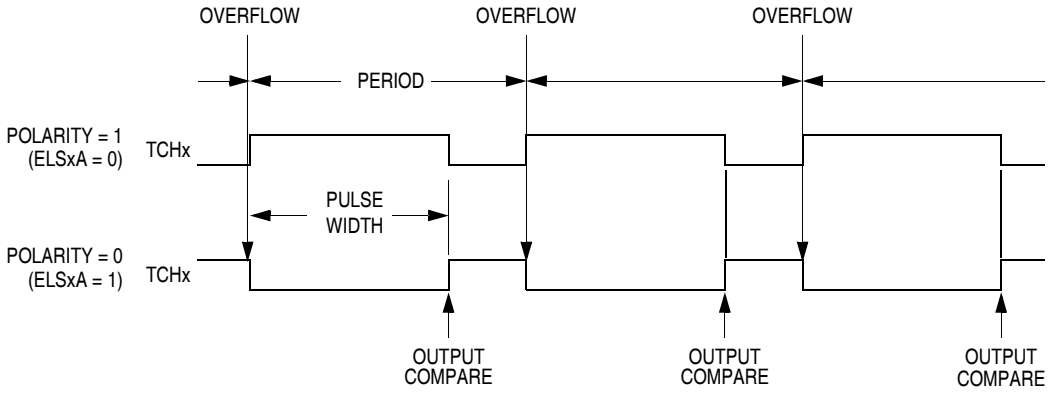



**Figure 19-4. PWM Period and Pulse Width**

The value in the TIM2 counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIM2 counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is 000 (see 19.8.1 TIM2 Status and Control Register).

The value in the TIM2 channel registers determines the pulse width of the PWM output. The pulse width of an 8-bit PWM signal is variable in 256 increments. Writing $0080 (128) to the TIM2 channel registers produces a duty cycle of 128/256 or 50%.

#### 19.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in 19.3.4 Pulse Width Modulation (PWM). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIM2 channel registers.

An unsynchronized write to the TIM2 channel registers to change a pulse width value could cause incorrect operation for up to two PWM periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that PWM period. Also, using a TIM2 overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIM2 may pass the new value before it is written to the timer channel (T2CHxH:T2CHxL) registers.

Use the following methods to synchronize unbuffered changes in the PWM pulse width on channel x:

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable TIM2 overflow interrupts and write the new value in the TIM2 overflow interrupt routine. The TIM2 overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

**NOTE**

*In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare also can cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

#### 19.3.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the T2CH0 pin. The TIM2 channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIM2 channel 0 status and control register (T2SC0) links channel 0 and channel 1. The TIM2 channel 0 registers initially control the pulse width on the T2CH0 pin. Writing to the TIM2 channel 1 registers enables the TIM2 channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM2 channel registers (0 or 1) that control the pulse width are the ones written to last. T2SC0 controls and monitors the buffered PWM function, and TIM2 channel 1 status and control register (T2SC1) is unused. While the MS0B bit is set, the channel 1 pin, T2CH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered PWM channel whose output appears on the T2CH2 pin. The TIM2 channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS2B bit in TIM2 channel 2 status and control register (T2SC2) links channel 2 and channel 3. The TIM2 channel 2 registers initially control the pulse width on the T2CH2 pin. Writing to the TIM2 channel 3 registers enables the TIM2 channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM2 channel registers (2 or 3) that control the pulse width are the ones written to last. T2SC2 controls and monitors the buffered PWM

function, and TIM2 channel 3 status and control register (T2SC3) is unused. While the MS2B bit is set, the channel 3 pin, T2CH3, is available as a general-purpose I/O pin.

Channels 4 and 5 can be linked to form a buffered PWM channel whose output appears on the T2CH4 pin. The TIM2 channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS4B bit in TIM2 channel 4 status and control register (T2SC4) links channel 4 and channel 5. The TIM2 channel 4 registers initially control the pulse width on the T2CH4 pin. Writing to the TIM2 channel 5 registers enables the TIM2 channel 5 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM2 channel registers (4 or 5) that control the pulse width are the ones written to last. T2SC4 controls and monitors the buffered PWM function, and TIM2 channel 5 status and control register (T2SC5) is unused. While the MS4B bit is set, the channel 5 pin, T2CH5, is available as a general-purpose I/O pin.

**NOTE**

*In buffered PWM signal generation, do not write pulse width values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.*

#### 19.3.4.3 PWM Initialization

To ensure correct operation when generating unbuffered or buffered PWM signals, use the following initialization procedure:

1. In the TIM2 status and control register (T2SC):
   a. Stop the TIM2 counter by setting the TIM2 stop bit, TSTOP.
   b. Reset the TIM2 counter and prescaler by setting the TIM2 reset bit, TRST.
2. In the TIM2 counter modulo registers (T2MODH:T2MODL), write the value for the required PWM period.
3. In the TIM2 channel x registers (T2CHxH:T2CHxL), write the value for the required pulse width.
4. In TIM2 channel x status and control register (T2SCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. (See Table 19-2.)
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (polarity 1 — to clear output on compare) or 1:1 (polarity 0 — to set output on compare) to the edge/level select bits, ELSxB:ELSxA. The output action on compare must force the output to the complement of the pulse width level. (See Table 19-2.)

   **NOTE**

   *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*
5. In the TIM2 status control register (T2SC), clear the TIM2 stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIM2 channel 0 registers (T2CH0H:T2CH0L) initially control the buffered PWM output. TIM2 status control register 0 (T2SC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Setting MS2B links channels 2 and 3 and configures them for buffered PWM operation. The TIM2 channel 2 registers (T2CH2H:T2CH2L) initially control the buffered PWM output. TIM2 status control register 2 (T2SC2) controls and monitors the PWM signal from the linked channels. MS2B takes priority over MS2A.

Setting MS4B links channels 4 and 5 and configures them for buffered PWM operation. The TIM2 channel 4 registers (T2CH4H:T2CH4L) initially control the buffered PWM output. TIM2 status control register 4 (T2SC4) controls and monitors the PWM signal from the linked channels. MS4B takes priority over MS4A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIM2 overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and setting the TOVx bit generates a 100% duty cycle output. (See 19.8.4 TIM2 Channel Status and Control Registers.)

## 19.4 Interrupts

The following TIM2 sources can generate interrupt requests:

- TIM2 overflow flag (TOF) — The TOF bit is set when the TIM2 counter reaches the modulo value programmed in the TIM2 counter modulo registers. The TIM2 overflow interrupt enable bit, TOIE, enables TIM2 overflow interrupt requests. TOF and TOIE are in the TIM2 status and control register.
- TIM2 channel flags (CH5F:CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM2 CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

## 19.5 Low-Power Modes

The WAIT and STOP instructions put the MCU in low-power standby modes.

### 19.5.1 Wait Mode

The TIM2 remains active after the execution of a WAIT instruction. In wait mode, the TIM2 registers are not accessible by the CPU. Any enabled CPU interrupt request from the TIM2 can bring the MCU out of wait mode.

If TIM2 functions are not required during wait mode, reduce power consumption by stopping the TIM2 before executing the WAIT instruction.

### 19.5.2 Stop Mode

The TIM2 is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions or the state of the TIM2 counter. TIM2 operation resumes when the MCU exits stop mode.

## 19.6 TIM2 During Break Interrupts

A break interrupt stops the TIM2 counter and inhibits input captures.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See 15.7.3 Break Flag Control Register.)

To allow software to clear status bits during a break interrupt, write a 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a 0 to the BCFE bit. With BCFE at 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at 0. After the break, doing the second step clears the status bit.

## 19.7 I/O Signals

Port D shares two of its pins with the TIM2. Port F shares four of its pins with the TIM2. PTD6/T2CH0 is an external clock input to the TIM2 prescaler. The six TIM2 channel I/O pins are PTD6/T2CH0, PTD7/T2CH1, PTF4/T2CH2, PTF5/T2CH3, PTF6/T2CH4, and PTF7/T2CH5.

### 19.7.1 TIM2 Clock Pin (T2CH0)

T2CH0 is an external clock input that can be the clock source for the TIM2 counter instead of the prescaled internal bus clock. Select the T2CH0 input by writing 1s to the three prescaler select bits, PS[2:0]. (See 19.8.1 TIM2 Status and Control Register.) The minimum TCLK pulse width is specified in 21.14 Timer Interface Module Characteristics. The maximum TCLK frequency is the least: 4 MHz or bus frequency ÷ 2.

When the PTD6/T2CH0 pin is the TIM2 clock input, it is an input regardless of the state of the DDRD6 bit in data direction register D.

### 19.7.2 TIM2 Channel I/O Pins (T2CH5:T2CH2 and T2CH1:T2CH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. T2CH0, T2CH2, and T2CH4 can be configured as buffered output compare or buffered PWM pins.

## 19.8 I/O Registers

These I/O registers control and monitor TIM2 operation:

- TIM2 status and control register (T2SC)
- TIM2 counter registers (T2CNTH:T2CNTL)
- TIM2 counter modulo registers (T2MODH:T2MODL)
- TIM2 channel status and control registers (T2SC0, T2SC1, T2SC2, T2SC3, T2SC4, and T2SC5)
- TIM2 channel registers (T2CH0H:T2CH0L, T2CH1H:T2CH1L, T2CH2H:T2CH2L, T2CH3H:T2CH3L, T2CH4H:T2CH4L, and T2CH5H:T2CH5L)

### 19.8.1 TIM2 Status and Control Register

The TIM2 status and control register:

- Enables TIM2 overflow interrupts
- Flags TIM2 overflows
- Stops the TIM2 counter
- Resets the TIM2 counter
- Prescales the TIM2 counter clock

Address: $002B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| Write: | 0 | | | TRST | | | | |
| Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 19-5. TIM2 Status and Control Register (T2SC)**

**TOF — TIM2 Overflow Flag Bit**

This read/write flag is set when the TIM2 counter resets reaches the modulo value programmed in the TIM2 counter modulo registers. Clear TOF by reading the TIM2 status and control register when TOF is set and then writing a 0 to TOF. If another TIM2 overflow occurs before the clearing sequence is complete, then writing 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a 1 to TOF has no effect.

1 = TIM2 counter has reached modulo value
0 = TIM2 counter has not reached modulo value

**TOIE — TIM2 Overflow Interrupt Enable Bit**

This read/write bit enables TIM2 overflow interrupts when the TOF bit becomes set. Reset clears the TOIE bit.

1 = TIM2 overflow interrupts enabled
0 = TIM2 overflow interrupts disabled

**TSTOP — TIM2 Stop Bit**

This read/write bit stops the TIM2 counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the TIM2 counter until software clears the TSTOP bit.

1 = TIM2 counter stopped
0 = TIM2 counter active

***NOTE***

*Do not set the TSTOP bit before entering wait mode if the TIM2 is required to exit wait mode. Also when the TSTOP bit is set and the timer is configured for input capture operation, input captures are inhibited until the TSTOP bit is cleared.*

**TRST — TIM2 Reset Bit**

Setting this write-only bit resets the TIM2 counter and the TIM2 prescaler. Setting TRST has no effect on any other registers. Counting resumes from $0000. TRST is cleared automatically after the TIM2 counter is reset and always reads as 0. Reset clears the TRST bit.

1 = Prescaler and TIM2 counter cleared
0 = No effect

***NOTE***

*Setting the TSTOP and TRST bits simultaneously stops the TIM2 counter at a value of $0000.*

**PS[2:0] — Prescaler Select Bits**

These read/write bits select either the T2CH0 pin or one of the seven prescaler outputs as the input to the TIM2 counter as Table 19-1 shows. Reset clears the PS[2:0] bits.

**Table 19-1. Prescaler Selection**

| PS[2:0] | TIM2 Clock Source |
|---|---|
| 000 | Internal bus clock ÷1 |
| 001 | Internal bus clock ÷ 2 |
| 010 | Internal bus clock ÷ 4 |
| 011 | Internal bus clock ÷ 8 |
| 100 | Internal bus clock ÷ 16 |
| 101 | Internal bus clock ÷ 32 |
| 110 | Internal bus clock ÷ 64 |
| 111 | T2CH0 |

## 19.8.2 TIM2 Counter Registers

The two read-only TIM2 counter registers contain the high and low bytes of the value in the TIM2 counter. Reading the high byte (T2CNTH) latches the contents of the low byte (T2CNTL) into a buffer. Subsequent reads of T2CNTH do not affect the latched T2CNTL value until T2CNTL is read. Reset clears the TIM2 counter registers. Setting the TIM2 reset bit (TRST) also clears the TIM2 counter registers.

***NOTE***

*If T2CNTH is read during a break interrupt, be sure to unlatch T2CNTL by reading T2CNTL before exiting the break interrupt. Otherwise, T2CNTL retains the value latched during the break.*

Address: $002C T2CNTH

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Address: $002D T2CNTL

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 19-6. TIM2 Counter Registers (T2CNTH and T2CNTL)**

### 19.8.3 TIM2 Counter Modulo Registers

The read/write TIM2 modulo registers contain the modulo value for the TIM2 counter. When the TIM2 counter reaches the modulo value, the overflow flag (TOF) becomes set, and the TIM2 counter resumes counting from $0000 at the next timer clock. Writing to the high byte (T2MODH) inhibits the TOF bit and overflow interrupts until the low byte (T2MODL) is written. Reset sets the TIM2 counter modulo registers.

Address: $002E T2MODH

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Address: $002F T2MODL

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 19-7. TIM2 Counter Modulo Registers (T2MODH and T2MODL)**

***NOTE***

*Reset the TIM2 counter before writing to the TIM2 counter modulo registers.*

### 19.8.4 TIM2 Channel Status and Control Registers

Each of the TIM2 channel status and control registers:

- Flags input captures and output compares
- Enables input capture and output compare interrupts
- Selects input capture, output compare, or PWM operation
- Selects high, low, or toggling output on output compare
- Selects rising edge, falling edge, or any edge as the active input capture trigger
- Selects output toggling on TIM2 overflow
- Selects 0% and 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Address: $0030 T2SC0

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| Write: | 0 | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 19-8. TIM2 Channel Status and Control Registers (T2SC0:T2SC5)**

| Address: | $0033 | T2SC1 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| Write: | 0 | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Address: | $0456 | T2SC2 | | | | | | |
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | CH2F | CH2IE | MS2B | MS2A | ELS2B | ELS2A | TOV2 | CH2MAX |
| Write: | 0 | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Address: | $0459 | T2SC3 | | | | | | |
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | CH3F | CH3IE | 0 | MS3A | ELS3B | ELS3A | TOV3 | CH3MAX |
| Write: | 0 | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Address: | $045C | T2SC4 | | | | | | |
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | CH4F | CH4IE | MS4B | MS4A | ELS4B | ELS4A | TOV4 | CH4MAX |
| Write: | 0 | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Address: | $045F | T2SC5 | | | | | | |
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | CH5F | CH5IE | 0 | MS5A | ELS5B | ELS5A | TOV5 | CH5MAX |
| Write: | 0 | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 19-8. TIM2 Channel Status and Control Registers (T2SC0:T2SC5) (Continued)**

### CHxF — Channel x Flag Bit

When channel x is an input capture channel, this read/write bit is set when an active edge occurs on the channel x pin. When channel x is an output compare channel, CHxF is set when the value in the TIM2 counter registers matches the value in the TIM2 channel x registers.

When CHxIE = 1, clear CHxF by reading TIM2 channel x status and control register with CHxF set, and then writing a 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a 1 to CHxF has no effect.

1 = Input capture or output compare on channel x
0 = No input capture or output compare on channel x

### CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIM2 CPU interrupts on channel x.
Reset clears the CHxIE bit.

1 = Channel x CPU interrupt requests enabled
0 = Channel x CPU interrupt requests disabled

**MSxB — Mode Select Bit B**

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIM2 channel 0, TIM2 channel 2, and TIM2 channel 4 status and control registers.

Setting MS0B disables the channel 1 status and control register and reverts T2CH1 pin to general-purpose I/O.

Setting MS2B disables the channel 3 status and control register and reverts T2CH3 pin to general-purpose I/O.

Setting MS4B disables the channel 5 status and control register and reverts T2CH5 pin to general-purpose I/O.

Reset clears the MSxB bit.

1 = Buffered output compare/PWM operation enabled
0 = Buffered output compare/PWM operation disabled

**MSxA — Mode Select Bit A**

When ELSxB:ELSxA ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. (See Table 19-2.)

1 = Unbuffered output compare/PWM operation
0 = Input capture operation

When ELSxB:ELSxA = 00, this read/write bit selects the initial output level of the T2CHx pin once PWM, input capture, or output compare operation is enabled. (See Table 19-2.) Reset clears the MSxA bit.

1 = Initial output level low
0 = Initial output level high

***NOTE***

*Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIM2 status and control register (T2SC).*

**Table 19-2. Mode, Edge, and Level Selection**

| MSxB | MSxA | ELSxB | ELSxA | Mode | Configuration |
|---|---|---|---|---|---|
| X | 0 | 0 | 0 | Output preset | Pin under port control; initial output level high |
| X | 1 | 0 | 0 | | Pin under port control; initial output level low |
| 0 | 0 | 0 | 1 | Input capture | Capture on rising edge only |
| 0 | 0 | 1 | 0 | | Capture on falling edge only |
| 0 | 0 | 1 | 1 | | Capture on rising or falling edge |
| 0 | 1 | 0 | 0 | Output compare or PWM | Software compare only |
| 0 | 1 | 0 | 1 | | Toggle output on compare |
| 0 | 1 | 1 | 0 | | Clear output on compare |
| 0 | 1 | 1 | 1 | | Set output on compare |
| 1 | X | 0 | 1 | Buffered output compare or buffered PWM | Toggle output on compare |
| 1 | X | 1 | 0 | | Clear output on compare |
| 1 | X | 1 | 1 | | Set output on compare |

### ELSxB and ELSxA — Edge/Level Select Bits

When channel x is an input capture channel, these read/write bits control the active edge-sensing logic on channel x.

When channel x is an output compare channel, ELSxB and ELSxA control the channel x output behavior when an output compare occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to port D or port F, and pin PTDx/T2CHx or pin PTFx/T2CHx is available as a general- purpose I/O pin. Table 19-2 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

**NOTE**

*After initially enabling a TIM2 channel register for input capture operation and selecting the edge sensitivity, clear CHxF to ignore any erroneous edge detection flags.*

### TOVx — Toggle-On-Overflow Bit

When channel x is an output compare channel, this read/write bit controls the behavior of the channel x output when the TIM2 counter overflows. When channel x is an input capture channel, TOVx has no effect. Reset clears the TOVx bit.

1 = Channel x pin toggles on TIM2 counter overflow.
0 = Channel x pin does not toggle on TIM2 counter overflow.

**NOTE**

*When TOVx is set, a TIM2 counter overflow takes precedence over a channel x output compare if both occur at the same time.*

### CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at a 1 and clear output on compare is selected, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 19-9 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at 100% duty cycle level until the cycle after CHxMAX is cleared.

**NOTE**

*The 100% PWM duty cycle is defined as a continuous high level if the PWM polarity is 1 and a continuous low level if the PWM polarity is 0. Conversely, a 0% PWM duty cycle is defined as a continuous low level if the PWM polarity is 1 and a continuous high level if the PWM polarity is 0.*

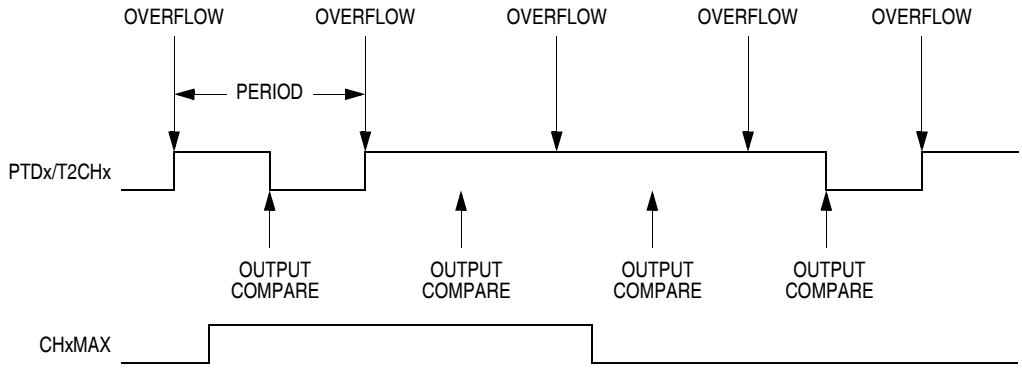



**Figure 19-9. CHxMAX Latency**

### 19.8.5 TIM2 Channel Registers

These read/write registers contain the captured TIM2 counter value of the input capture function or the output compare value of the output compare function. The state of the TIM2 channel registers after reset is unknown.

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the TIM2 channel x registers (T2CHxH) inhibits input captures until the low byte (T2CHxL) is read.

In output compare mode (MSxB:MSxA ≠ 0:0), writing to the high byte of the TIM2 channel x registers (T2CHxH) inhibits output compares until the low byte (T2CHxL) is written.

Address: $0031 T2CH0H

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset: | Indeterminate after reset | | | | | | | |

Address: $0032 T2CH0L

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset: | Indeterminate after reset | | | | | | | |

Address: $0034 T2CH1H

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset: | Indeterminate after reset | | | | | | | |

Address: $0035 T2CH1L

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset: | Indeterminate after reset | | | | | | | |

Address: $0457 T2CH2H

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Reset: | Indeterminate after reset | | | | | | | |

Address: $0458 T2CH2L

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset: | Indeterminate after reset | | | | | | | |

**Figure 19-10. TIM2 Channel Registers (T2CH0H/L:T2CH5H/L)**

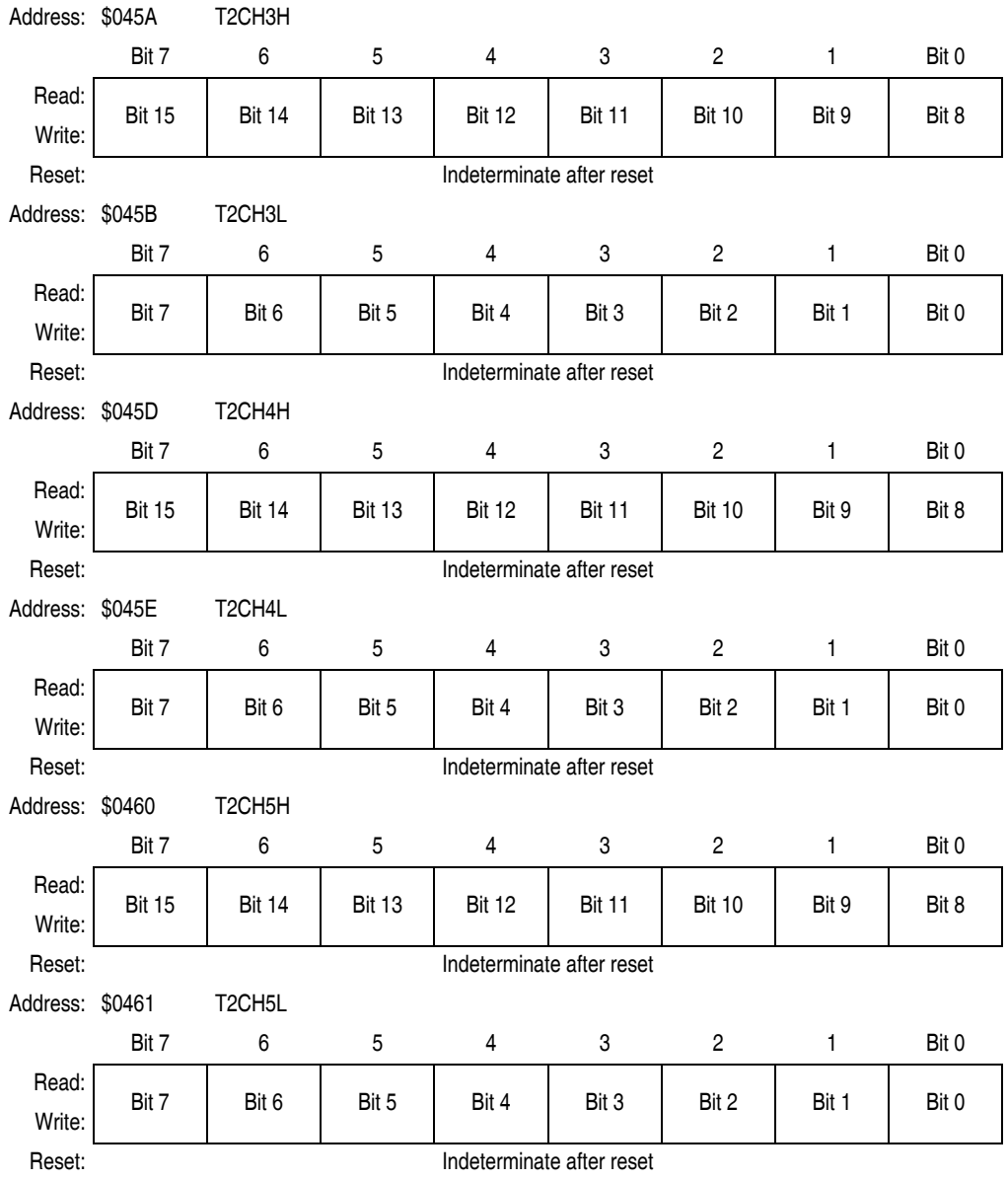



**Figure 19-10. TIM2 Channel Registers (T2CH0H/L:T2CH5H/L) (Continued)**

# Chapter 20
# Development Support

## 20.1 Introduction

This section describes the break module, the monitor module (MON), and the monitor mode entry methods.

## 20.2 Break Module (BRK)

The break module can generate a break interrupt that stops normal program flow at a defined address to enter a background program.

Features of the break module include:

- Accessible input/output (I/O) registers during the break Interrupt
- Central processor unit (CPU) generated break interrupts
- Software-generated break interrupts
- Computer operating properly (COP) disabling during break interrupts

### 20.2.1 Functional Description

When the internal address bus matches the value written in the break address registers, the break module issues a breakpoint signal ($\overline{\text{BKPT}}$) to the system integration module (SIM). The SIM then causes the CPU to load the instruction register with a software interrupt instruction (SWI). The program counter vectors to $FFFC and $FFFD ($FEFC and $FEFD in monitor mode).

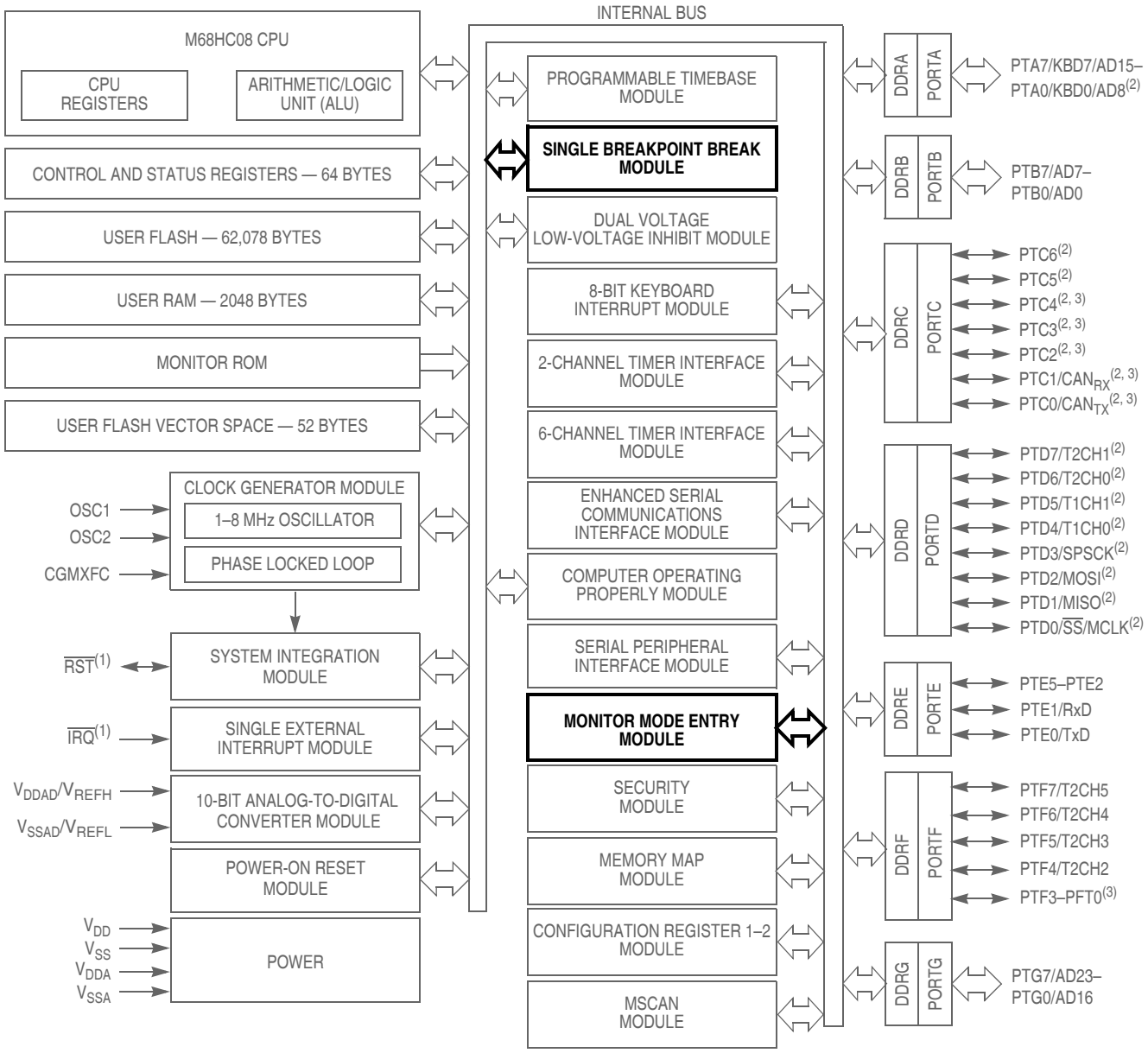
The following events can cause a break interrupt to occur:

- A CPU generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a 1 to the BRKA bit in the break status and control register.

When a CPU generated address matches the contents of the break address registers, the break interrupt is generated. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the microcontroller unit (MCU) to normal operation.

Figure 20-2 shows the structure of the break module.

Figure 20-3 provides a summary of the I/O registers.




1. Pin contains integrated pullup device.
2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.
3. Higher current drive port pins

**Figure 20-1. Block Diagram Highlighting BRK and MON Blocks**

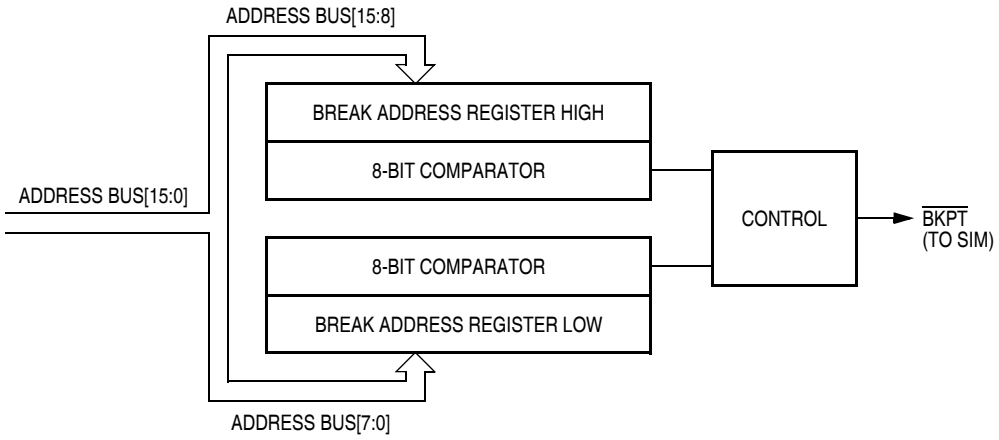



**Figure 20-2. Break Module Block Diagram**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FE00 | Break Status Register (BSR) See page 304. | Read: | R | R | R | R | R | R | SBSW | R |
| | | Write: | | | | | | | Note(1) | |
| | | Reset: | | | | | | | 0 | |
| $FE03 | Break Flag Control Register (BFCR) See page 304. | Read: | BCFE | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | | | | | | | |
| $FE09 | Break Address High Register (BRKH) See page 303. | Read: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE0A | Break Address Low Register (BRKL) See page 303. | Read: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE0B | Break Status and Control Register (BRKSCR) See page 303. | Read: | BRKE | BRKA | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. Writing a 0 clears SBSW.

= Unimplemented  R = Reserved

**Figure 20-3. Break I/O Register Summary**

When the internal address bus matches the value written in the break address registers or when software writes a 1 to the BRKA bit in the break status and control register, the CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with $FFFC and $FFFD ($FEFC and $FEFD in monitor mode)

The break interrupt timing is:

- When a break address is placed at the address of the instruction opcode, the instruction is not executed until after completion of the break interrupt routine.
- When a break address is placed at an address of an instruction operand, the instruction is executed before the break interrupt.
- When software writes a 1 to the BRKA bit, the break interrupt occurs just before the next instruction is executed.

By updating a break address and clearing the BRKA bit in a break interrupt routine, a break interrupt can be generated continuously.

***CAUTION***

*A break address should be placed at the address of the instruction opcode. When software does not change the break address and clears the BRKA bit in the first break interrupt routine, the next break interrupt will not be generated after exiting the interrupt routine even when the internal address bus matches the value written in the break address registers.*

#### *20.2.1.1 Flag Protection During Break Interrupts*

The system integration module (SIM) controls whether or not module status bits can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See 15.7.3 Break Flag Control Register and the **Break Interrupts** subsection for each module.

#### *20.2.1.2 TIM During Break Interrupts*

A break interrupt stops the timer counter and inhibits input captures.

#### *20.2.1.3 COP During Break Interrupts*

The COP is disabled during a break interrupt when $V_{TST}$ is present on the $\overline{RST}$ pin.

### 20.2.2 Break Module Registers

These registers control and monitor operation of the break module:

- Break status and control register (BRKSCR)
- Break address register high (BRKH)
- Break address register low (BRKL)
- Break status register (BSR)
- Break flag control register (BFCR)

### 20.2.2.1 Break Status and Control Register

The break status and control register (BRKSCR) contains break module enable and status bits.

Address: $FE0B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | BRKE | BRKA | 0 | 0 | 0 | 0 | 0 | 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 20-4. Break Status and Control Register (BRKSCR)**

**BRKE — Break Enable Bit**

This read/write bit enables breaks on break address register matches. Clear BRKE by writing a 0 to bit 7. Reset clears the BRKE bit.

1 = Breaks enabled on 16-bit address match
0 = Breaks disabled

**BRKA — Break Active Bit**

This read/write status and control bit is set when a break address match occurs. Writing a 1 to BRKA generates a break interrupt. Clear BRKA by writing a 0 to it before exiting the break routine. Reset clears the BRKA bit.

1 = Break address match
0 = No break address match

### 20.2.2.2 Break Address Registers

The break address registers (BRKH and BRKL) contain the high and low bytes of the desired breakpoint address. Reset clears the break address registers.

Address: $FE09

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-5. Break Address Register High (BRKH)**

Address: $FE0A

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-6. Break Address Register Low (BRKL)**

### 20.2.2.3 Break Status Register

The break status register (BSR) contains a flag to indicate that a break caused an exit from wait mode. This register is only used in emulation mode.

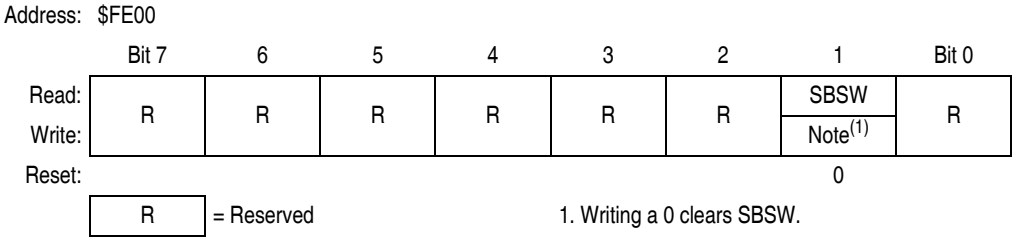



**Figure 20-7. Break Status Register (BSR)**

**SBSW — SIM Break Stop/Wait**

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it.

1 = Wait mode was exited by break interrupt
0 = Wait mode was not exited by break interrupt

### 20.2.2.4 Break Flag Control Register

The break control register (BFCR) contains a bit that enables software to clear status bits while the MCU is in a break state.

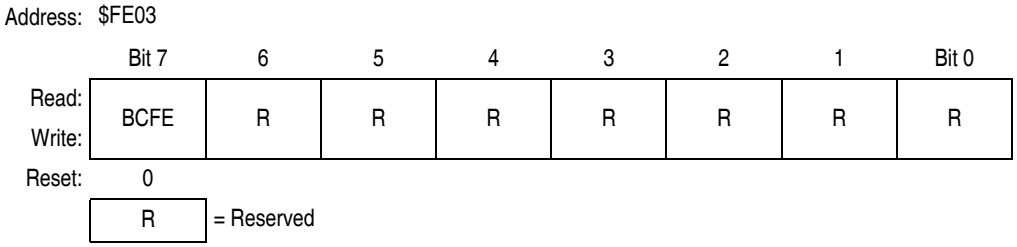



**Figure 20-8. Break Flag Control Register (BFCR)**

**BCFE — Break Clear Flag Enable Bit**

This read/write bit enables software to clear status bits by accessing status registers while the MCU is in a break state. To clear status bits during the break state, the BCFE bit must be set.

1 = Status bits clearable during break
0 = Status bits not clearable during break

## 20.2.3 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power- consumption standby modes. If enabled, the break module will remain enabled in wait and stop modes. However, since the internal address bus does not increment in these modes, a break interrupt will never be triggered.

## 20.3 Monitor Module (MON)

The monitor module allows debugging and programming of the microcontroller unit (MCU) through a single-wire interface with a host computer. Monitor mode entry can be achieved without use of the higher test voltage, $V_{TST}$, as long as vector addresses $FFFE and $FFFF are blank, thus reducing the hardware requirements for in-circuit programming.

Features of the monitor module include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between MCU and host computer
- Standard non-return-to-zero (NRZ) communication with host computer
- Standard communication baud rate (7200 @ 2-MHz bus frequency)
- Execution of code in random-access memory (RAM) or FLASH
- FLASH memory security feature(1)
- FLASH memory programming interface
- Monitor mode entry without high voltage, $V_{TST}$, if reset vector is blank ($FFFE and $FFFF contain $FF)
- Normal monitor mode entry if $V_{TST}$ is applied to $\overline{IRQ}$

### 20.3.1 Functional Description

Figure 20-9 shows a simplified diagram of the monitor mode.

The monitor module receives and executes commands from a host computer.
Figure 20-10 and Figure 20-11 show example circuits used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

Simple monitor commands can access any memory address. In monitor mode, the MCU can execute code downloaded into RAM by a host computer while most MCU pins retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.

Table 20-1 shows the pin conditions for entering monitor mode. As specified in the table, monitor mode may be entered after a power-on reset (POR) and will allow communication at 7200 baud provided one of the following sets of conditions is met:

- If $FFFE and $FFFF does not contain $FF (programmed state):
  - The external clock is 4.0 MHz (7200 baud)
  - PTB4 = low
  - $\overline{IRQ}$ = $V_{TST}$
- If $FFFE and $FFFF do not contain $FF (programmed state):
  - The external clock is 8.0 MHz (7200 baud)
  - PTB4 = high
  - IRQ = $V_{TST}$
- If $FFFE and $FFFF contain $FF (erased state):
  - The external clock is 8.0 MHz (7200 baud)
  - $\overline{IRQ}$ = $V_{DD}$ (this can be implemented through the internal $\overline{IRQ}$ pullup) or $V_{SS}$

1. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH difficult for unauthorized users.

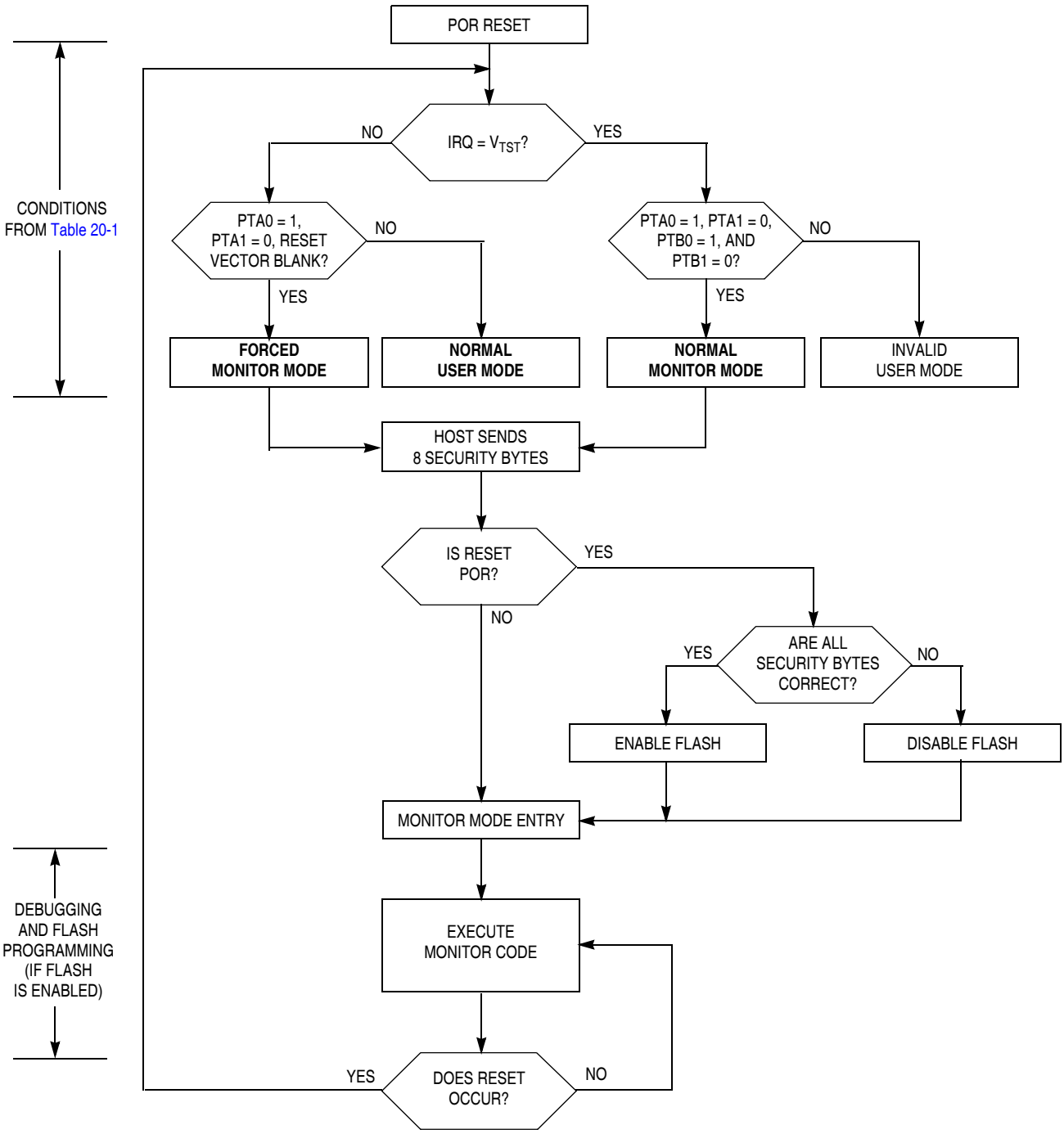



**Figure 20-9. Simplified Monitor Mode Entry Flowchart**

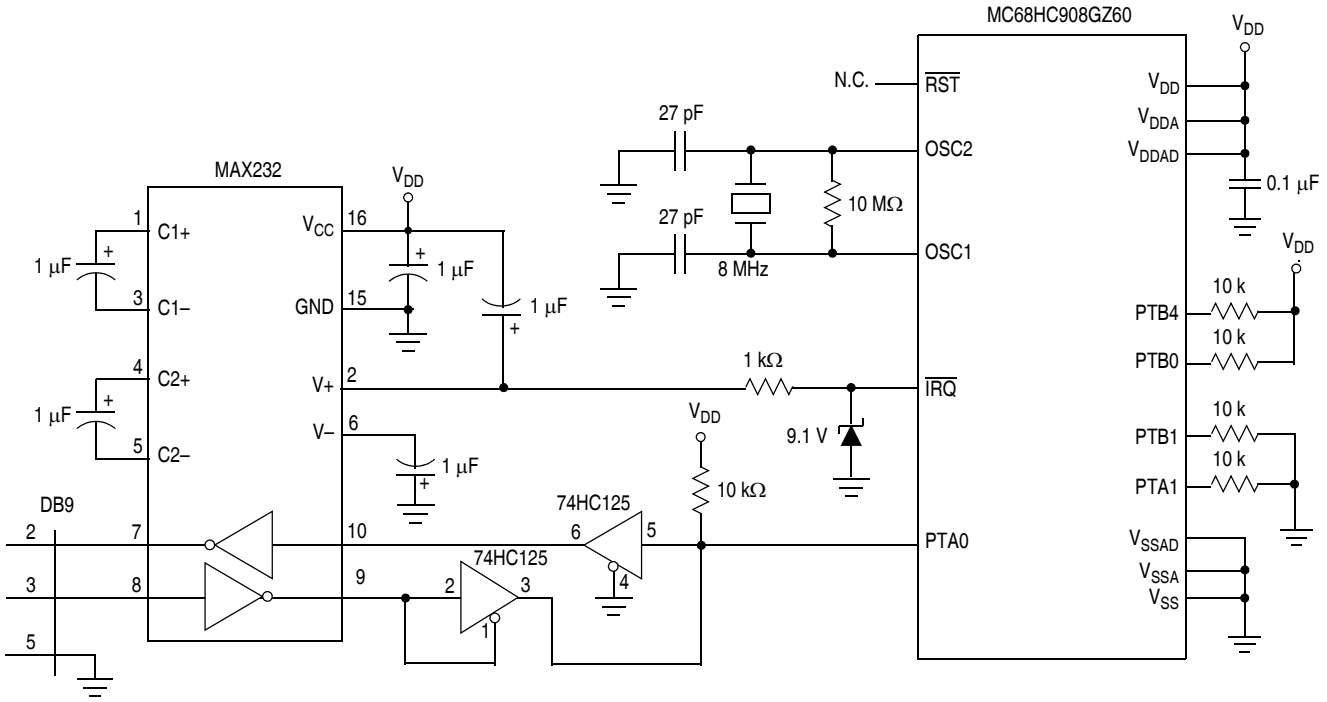



**Figure 20-10. Normal Monitor Mode Circuit**

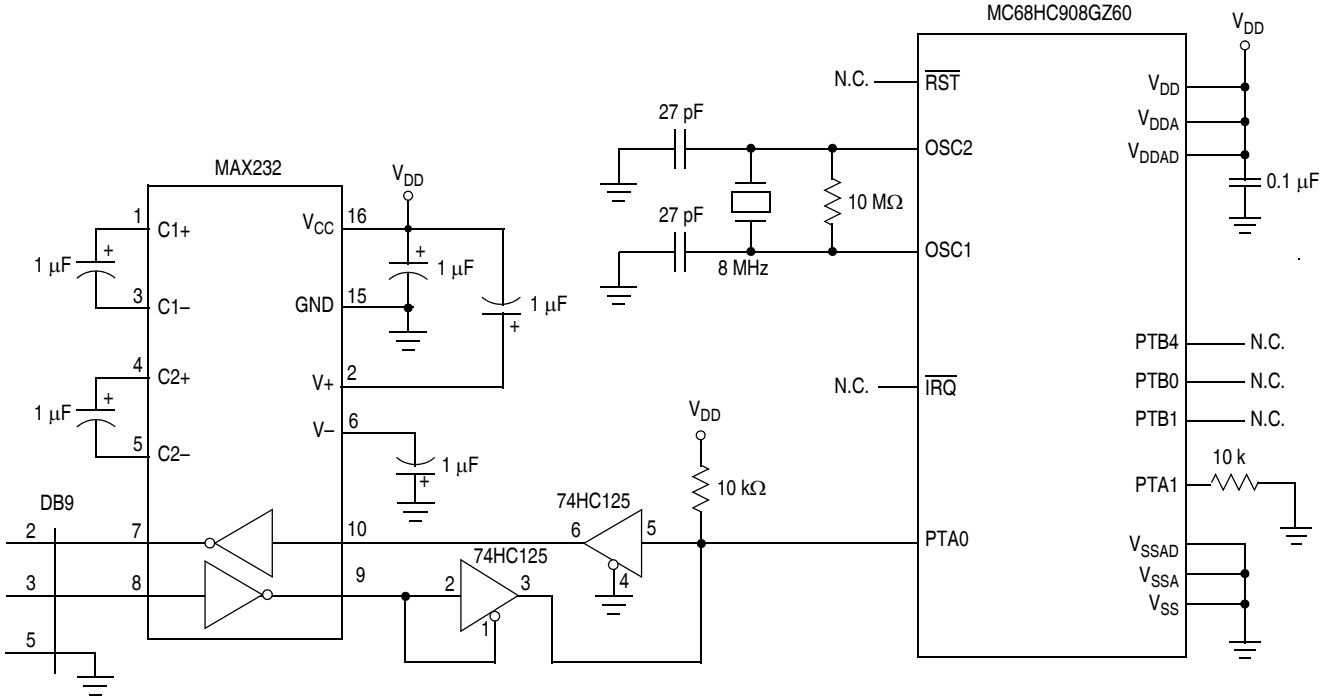



**Figure 20-11. Forced Monitor Mode**

**Table 20-1. Monitor Mode Signal Requirements and Options**

| Mode | $\overline{IRQ}$ | $\overline{RST}$ | Reset Vector | Serial Communication | | Mode Selection | | Divider | PLL | COP | Communication Speed | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | PTA0 | PTA1 | PTB0 | PTB1 | PTB4 | | | External Clock | Bus Frequency | Baud Rate |
| Normal Monitor | $V_{TST}$ | $V_{DD}$ or $V_{TST}$ | X | 1 | 0 | 1 | 0 | 0 | OFF | Disabled | 4.0 MHz | 2.0 MHz | 7200 |
| | $V_{TST}$ | $V_{DD}$ or $V_{TST}$ | X | 1 | 0 | 1 | 0 | 1 | OFF | Disabled | 8.0 MHz | 2.0 MHz | 7200 |
| Forced Monitor | $V_{DD}$ or $V_{SS}$ | $V_{DD}$ | $FF (blank) | 1 | 0 | X | X | X | OFF | Disabled | 8.0 MHz | 2.0 MHz | 7200 |
| User | $V_{DD}$ or $V_{SS}$ | $V_{DD}$ or $V_{TST}$ | Not $FF | X | X | X | X | X | X | Enabled | X | X | X |
| MON08 Function [Pin No.] | $V_{TST}$ [6] | $\overline{RST}$ [4] | — | COM [8] | SSEL [10] | MOD0 [12] | MOD1 [14] | DIV4 [16] | — | — | OSC1 [13] | — | — |

1. PTA0 must have a pullup resistor to $V_{DD}$ in monitor mode.
2. Communication speed in the table is an example to obtain a baud rate of 7200. Baud rate using external oscillator is bus frequency / 278.
3. External clock is a 4.0 MHz or 8.0 MHz crystal on OSC1 and OSC2 or a canned oscillator on OSC1.
4. X = don't care
5. MON08 pin refers to P&E Microcomputer Systems' MON08-Cyclone 2 by 8-pin connector.

| | | | |
|---|---|---|---|
| NC | 1 | 2 | GND |
| NC | 3 | 4 | $\overline{RST}$ |
| NC | 5 | 6 | $\overline{IRQ}$ |
| NC | 7 | 8 | PTA0 |
| NC | 9 | 10 | PTA1 |
| NC | 11 | 12 | PTB0 |
| OSC1 | 13 | 14 | PTB1 |
| $V_{DD}$ | 15 | 16 | PTB4 |

Enter monitor mode with pin configuration shown in Table 20-1 by pulling $\overline{RST}$ low and then high. The rising edge of $\overline{RST}$ latches monitor mode. Once monitor mode is latched, the levels on the port pins except PTA0 can change.

Once out of reset, the MCU waits for the host to send eight security bytes (see 20.3.2 Security). After the security bytes, the MCU sends a break signal (10 consecutive 0s) to the host, indicating that it is ready to receive a command.

#### 20.3.1.1 Normal Monitor Mode

If $V_{TST}$ is applied to $\overline{IRQ}$ and PTB4 is low upon monitor mode entry, the bus frequency is a divide-by-two of the input clock. If PTB4 is high with $V_{TST}$ applied to $\overline{IRQ}$ upon monitor mode entry, the bus frequency will be a divide-by-four of the input clock. Holding the PTB4 pin low when entering monitor mode causes a bypass of a divide-by-two stage at the oscillator *only if $V_{TST}$ is applied to $\overline{IRQ}$*. In this event, the CGMOUT frequency is equal to the CGMXCLK frequency, and the OSC1 input directly generates internal bus clocks. In this case, the OSC1 signal must have a 50% duty cycle at maximum bus frequency.

When monitor mode was entered with $V_{TST}$ on $\overline{IRQ}$, the computer operating properly (COP) is disabled as long as $V_{TST}$ is applied to either $\overline{IRQ}$ or $\overline{RST}$.

This condition states that as long as $V_{TST}$ is maintained on the $\overline{IRQ}$ pin after entering monitor mode, or if $V_{TST}$ is applied to $\overline{RST}$ after the initial reset to get into monitor mode (when $V_{TST}$ was applied to $\overline{IRQ}$), then the COP will be disabled. In the latter situation, after $V_{TST}$ is applied to the $\overline{RST}$ pin, $V_{TST}$ can be removed from the $\overline{IRQ}$ pin in the interest of freeing the $\overline{IRQ}$ for normal functionality in monitor mode.

#### 20.3.1.2 Forced Monitor Mode

If entering monitor mode without high voltage on $\overline{IRQ}$, then all port B pin requirements and conditions, including the PTB4 frequency divisor selection, are not in effect. This is to reduce circuit requirements when performing in-circuit programming.

**NOTE**

*If the reset vector is blank and monitor mode is entered, the chip will see an additional reset cycle after the initial power-on reset (POR). Once the reset vector has been programmed, the traditional method of applying a voltage, $V_{TST}$, to $\overline{IRQ}$ must be used to enter monitor mode.*

An external oscillator of 8 MHz is required for a baud rate of 7200, as the internal bus frequency is automatically set to the external frequency divided by four.

When the forced monitor mode is entered the COP is always disabled regardless of the state of $\overline{IRQ}$ or $\overline{RST}$.

#### 20.3.1.3 Monitor Vectors

In monitor mode, the MCU uses different vectors for reset, SWI (software interrupt), and break interrupt than those for user mode. The alternate vectors are in the $FE page instead of the $FF page and allow code execution from the internal monitor firmware instead of user code.

Table 20-2 summarizes the differences between user mode and monitor mode.

**Table 20-2. Mode Differences**

| Modes | Functions | | | | | |
|---|---|---|---|---|---|---|
| | Reset Vector High | Reset Vector Low | Break Vector High | Break Vector Low | SWI Vector High | SWI Vector Low |
| User | $FFFE | $FFFF | $FFFC | $FFFD | $FFFC | $FFFD |
| Monitor | $FEFE | $FEFF | $FEFC | $FEFD | $FEFC | $FEFD |

### 20.3.1.4 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. Transmit and receive baud rates must be identical.

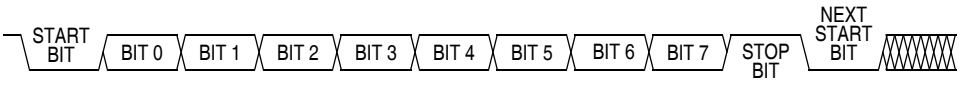



**Figure 20-12. Monitor Data Format**

### 20.3.1.5 Break Signal

A start bit (0) followed by nine 0 bits is a break signal. When the monitor receives a break signal, it drives the PTA0 pin high for the duration of approximately two bits and then echoes back the break signal.

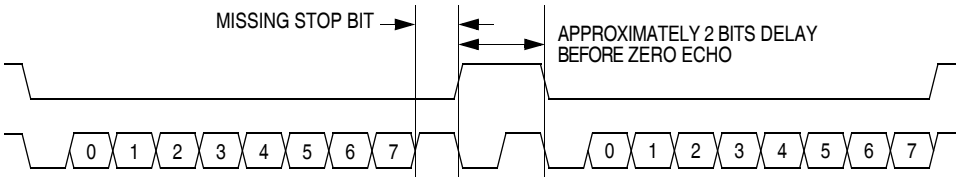



**Figure 20-13. Break Transaction**

### 20.3.1.6 Baud Rate

The communication baud rate is controlled by the crystal frequency or external clock and the state of the PTB4 pin (when $\overline{IRQ}$ is set to $V_{TST}$) upon entry into monitor mode. If monitor mode was entered with $V_{DD}$ on $\overline{IRQ}$ and the reset vector blank, then the baud rate is independent of PTB4.

Table 20-1 also lists external frequencies required to achieve a standard baud rate of 7200 bps. The effective baud rate is the bus frequency divided by 278. If using a crystal as the clock source, be aware of the upper frequency limit that the internal clock module can handle. See 21.7 5.0-Volt Control Timing or 21.8 3.3-Volt Control Timing for this limit.

### 20.3.1.7 Commands

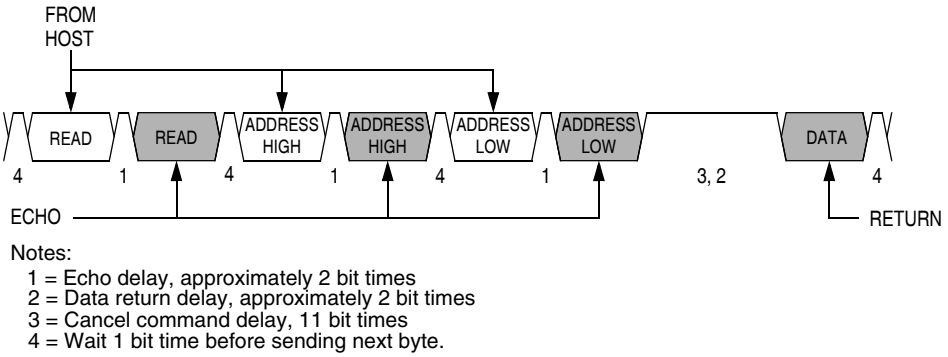
The monitor ROM firmware uses these commands:

- READ (read memory)
- WRITE (write memory)
- IREAD (indexed read)
- IWRITE (indexed write)
- READSP (read stack pointer)
- RUN (run user program)

The monitor ROM firmware echoes each received byte back to the PTA0 pin for error checking. An 11-bit delay at the end of each command allows the host to send a break character to cancel the command. A delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. The data returned by a read command appears after the echo of the last byte of the command.

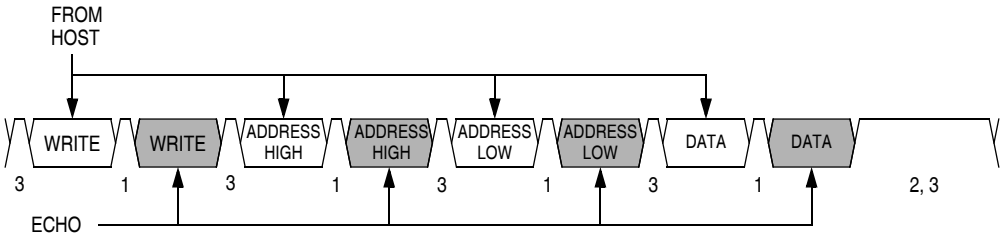
***NOTE***

*Wait one bit time after each echo before sending the next byte.*




Notes:
1 = Echo delay, approximately 2 bit times
2 = Data return delay, approximately 2 bit times
3 = Cancel command delay, 11 bit times
4 = Wait 1 bit time before sending next byte.

**Figure 20-14. Read Transaction**




Notes:
1 = Echo delay, approximately 2 bit times
2 = Cancel command delay, 11 bit times
3 = Wait 1 bit time before sending next byte.

**Figure 20-15. Write Transaction**

A brief description of each monitor mode command is given in Table 20-3 through Table 20-8.

**Table 20-3. READ (Read Memory) Command**

| | |
|---|---|
| Description | Read byte from memory |
| Operand | 2-byte address in high-byte:low-byte order |
| Data Returned | Returns contents of specified address |
| Opcode | $4A |
| **Command Sequence** | |

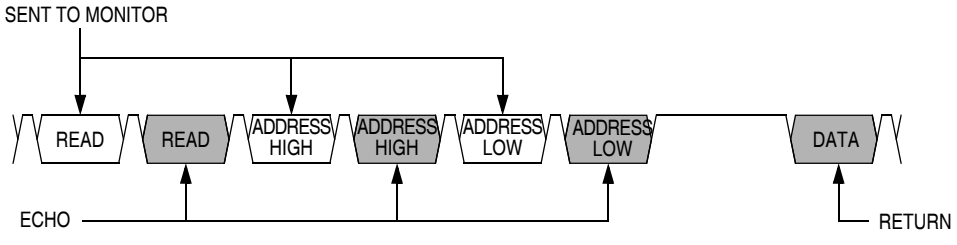


**Table 20-4. WRITE (Write Memory) Command**

| | |
|---|---|
| Description | Write byte to memory |
| Operand | 2-byte address in high-byte:low-byte order; low byte followed by data byte |
| Data Returned | None |
| Opcode | $49 |

**Command Sequence**

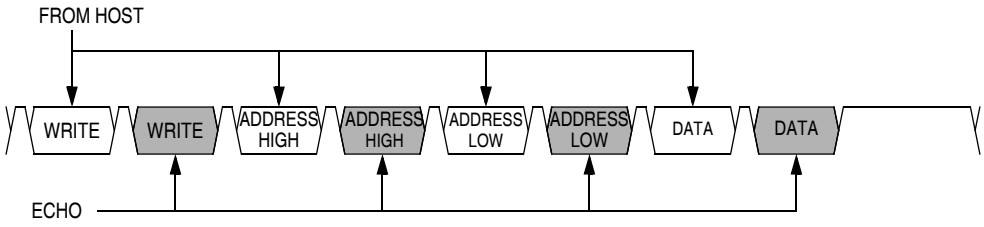



**Table 20-5. IREAD (Indexed Read) Command**

| | |
|---|---|
| Description | Read next 2 bytes in memory from last address accessed |
| Operand | None |
| Data Returned | Returns contents of next two addresses |
| Opcode | $1A |

**Command Sequence**

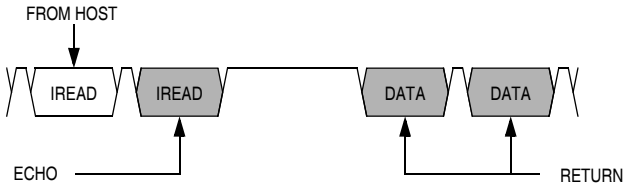
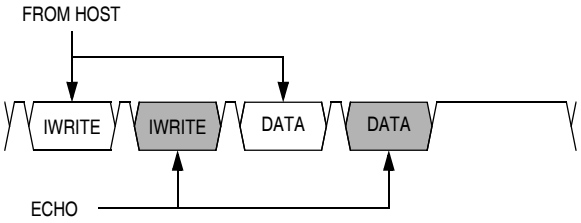



**Table 20-6. IWRITE (Indexed Write) Command**

| | |
|---|---|
| Description | Write to last address accessed + 1 |
| Operand | Single data byte |
| Data Returned | None |
| Opcode | $19 |

**Command Sequence**




A sequence of IREAD or IWRITE commands can access a block of memory sequentially over the full 64-Kbyte memory map.

**Table 20-7. READSP (Read Stack Pointer) Command**

| | |
|---|---|
| Description | Reads stack pointer |
| Operand | None |
| Data Returned | Returns incremented stack pointer value (SP + 1) in high-byte:low-byte order |
| Opcode | $0C |
| **Command Sequence** | |

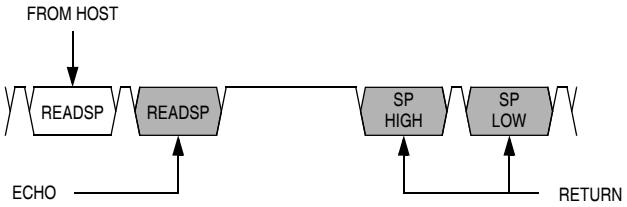
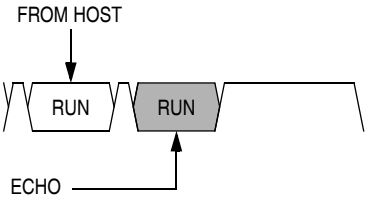



**Table 20-8. RUN (Run User Program) Command**

| | |
|---|---|
| Description | Executes PULH and RTI instructions |
| Operand | None |
| Data Returned | None |
| Opcode | $28 |
| **Command Sequence** | |




The MCU executes the SWI and PSHH instructions when it enters monitor mode. The RUN command tells the MCU to execute the PULH and RTI instructions. Before sending the RUN command, the host can modify the stacked CPU registers to prepare to run the host program. The READSP command returns the incremented stack pointer value, SP + 1. The high and low bytes of the program counter are at addresses SP + 5 and SP + 6.

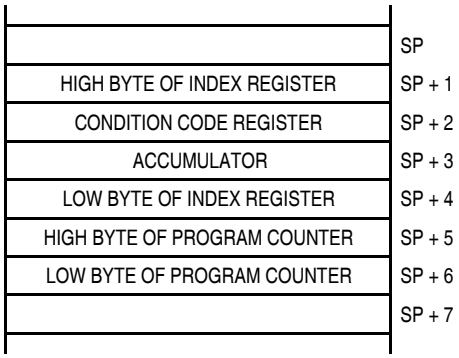



**Figure 20-16. Stack Pointer at Monitor Mode Entry**

### 20.3.2 Security

A security feature discourages unauthorized reading of FLASH locations while in monitor mode. The host can bypass the security feature at monitor mode entry by sending eight security bytes that match the bytes at locations $FFF6–$FFFD. Locations $FFF6–$FFFD contain user-defined data.

**NOTE**

*Do not leave locations $FFF6–$FFFD blank. For security reasons, program locations $FFF6–$FFFD even if they are not used for vectors.*

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PTA0. If the received bytes match those at locations $FFF6–$FFFD, the host bypasses the security feature and can read all FLASH locations and execute code from FLASH. Security remains bypassed until a power-on reset occurs. If the reset was not a power-on reset, security remains bypassed and security code entry is not required. See Figure 20-17.

Upon power-on reset, if the received bytes of the security code do not match the data at locations $FFF6–$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. After receiving the eight security bytes from the host, the MCU transmits a break character, signifying that it is ready to receive a command.

**NOTE**

*The MCU does not transmit a break character until after the host sends the eight security bytes.*

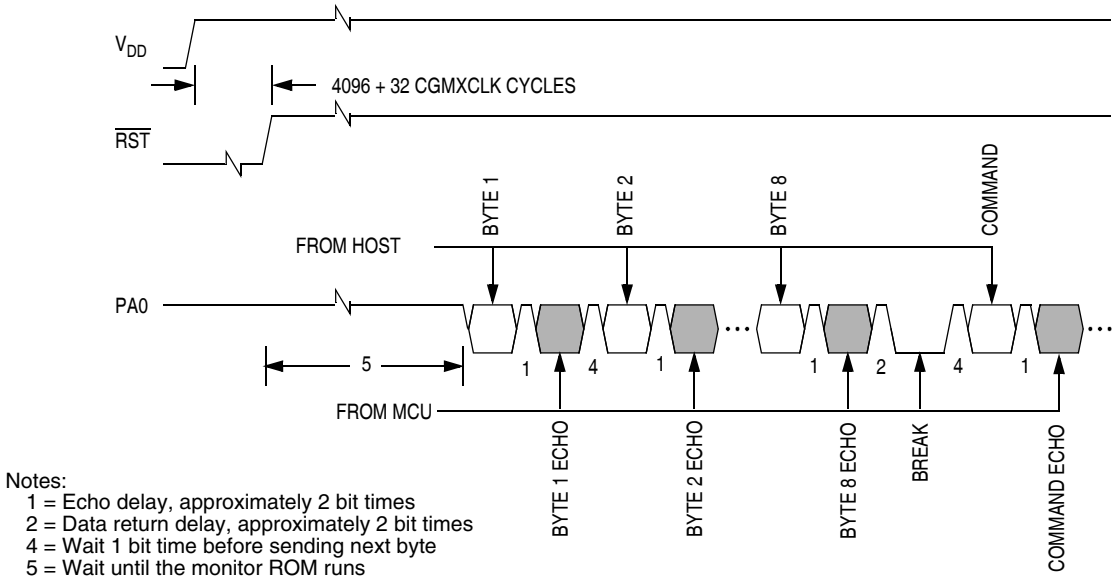



**Figure 20-17. Monitor Mode Entry Timing**

To determine whether the security code entered is correct, check to see if bit 6 of RAM address $40 is set. If it is, then the correct security code has been entered and FLASH can be accessed.

If the security sequence fails, the device should be reset by a power-on reset and brought up in monitor mode to attempt another entry. After failing the security sequence, the FLASH module can also be mass erased by executing an erase routine that was downloaded into internal RAM. The mass erase operation clears the security code locations so that all eight security bytes become $FF (blank).

# Chapter 21
# Electrical Specifications

## 21.1 Introduction

This section contains electrical and timing specifications.

## 21.2 Absolute Maximum Ratings

Maximum ratings are the extreme limits to which the microcontroller unit (MCU) can be exposed without permanently damaging it.

***NOTE***

*This device is not guaranteed to operate properly at the maximum ratings. Refer to 21.5 5.0-Vdc Electrical Characteristics for guaranteed operating conditions.*

| Characteristic[1] | Symbol | Value | Unit |
|---|---|---|---|
| Supply voltage | $V_{DD}$ | –0.3 to + 6.0 | V |
| Input voltage | $V_{In}$ | $V_{SS}$ – 0.3 to $V_{DD}$ + 0.3 | V |
| Maximum current per pin excluding those specified below | I | ± 15 | mA |
| Maximum current for pins PTC0–PTC4 | $I_{PTC0–PTC4}$ | ± 25 | mA |
| Maximum current into $V_{DD}$ | $I_{mvdd}$ | 150 | mA |
| Maximum current out of $V_{SS}$ | $I_{mvss}$ | 150 | mA |
| Storage temperature | $T_{stg}$ | –55 to +150 | °C |

1. Voltages referenced to $V_{SS}$

***NOTE***

*This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. For proper operation, it is recommended that $V_{In}$ and $V_{Out}$ be constrained to the range $V_{SS} \le (V_{In}$ or $V_{Out}) \le V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either $V_{SS}$ or $V_{DD}$).*

## 21.3 Functional Operating Range

| Characteristic | Symbol | Value | Unit |
|---|---|---|---|
| Operating temperature range | $T_A$ | –40 to +125 | °C |
| Operating voltage range | $V_{DD}$ | 5.0 ±10%<br>3.3 ±10% | V |

## 21.4 Thermal Characteristics

| Characteristic | Symbol | Value | Unit |
|---|---|---|---|
| Thermal resistance<br>32-pin LQFP<br>48-pin LQFP<br>64-pin QFP | $\theta_{JA}$ | <br>95<br>95<br>54 | °C/W |
| I/O pin power dissipation | $P_{I/O}$ | User determined | W |
| Power dissipation(1) | $P_D$ | $P_D = (I_{DD} \times V_{DD}) + P_{I/O} = K/(T_J + 273\ °C)$ | W |
| Constant(2) | K | $P_D \times (T_A + 273\ °C) + P_D^2 \times \theta_{JA}$ | W/°C |
| Average junction temperature | $T_J$ | $T_A + (P_D \times \theta_{JA})$ | °C |

1. Power dissipation is a function of temperature.
2. K is a constant unique to the device. K can be determined for a known $T_A$ and measured $P_D$. With this value of K, $P_D$ and $T_J$ can be determined for any value of $T_A$.

## 21.5 5.0-Vdc Electrical Characteristics

| Characteristic[1] | Symbol | Min | Typ[2] | Max | Unit |
|---|---|---|---|---|---|
| Output high voltage | | | | | |
| ($I_{Load}$ = –2.0 mA) all I/O pins | $V_{OH}$ | $V_{DD}$ – 0.8 | — | — | V |
| ($I_{Load}$ = –10.0 mA) all I/O pins | $V_{OH}$ | $V_{DD}$ – 1.5 | — | — | V |
| ($I_{Load}$ = –20.0 mA) pins PTC0–PTC4, PTF0–PTF3 only | $V_{OH}$ | $V_{DD}$ – 1.5 | — | — | V |
| Maximum combined $I_{OH}$ for port PTA7–PTA3, port PTC0–PTC1, port E, port PTD0–PTD3, port PTF0–PTF3, port PTG4–PTG7 | $I_{OH1}$ | — | — | 50 | mA |
| Maximum combined $I_{OH}$ for port PTA2–PTA0, port B, port PTC2–PTC6, port PTD4–PTD7, port PTF4–PTF7, port PTG0–PTG3 | $I_{OH2}$ | — | — | 50 | mA |
| Maximum total $I_{OH}$ for all port pins | $I_{OHT}$ | — | — | 100 | mA |
| Output low voltage | | | | | |
| ($I_{Load}$ = 1.6 mA) all I/O pins | $V_{OL}$ | — | — | 0.4 | V |
| ($I_{Load}$ = 10 mA) all I/O pins | $V_{OL}$ | — | — | 1.5 | V |
| ($I_{Load}$ = 20 mA) pins PTC0–PTC4, PTF0–PTF3 only | $V_{OL}$ | — | — | 1.5 | V |
| Maximum combined $I_{OH}$ for port PTA7–PTA3, port PTC0–PTC1, port E, port PTD0–PTD3, port PTF0–PTF3, port PTG4–PTG7 | $I_{OL1}$ | — | — | 50 | mA |
| Maximum combined $I_{OH}$ for port PTA2–PTA0, port B, port PTC2–PTC6, port PTD4–PTD7, port PTF4–PTF7, port PTG0–PTG3 | $I_{OL2}$ | — | — | 50 | mA |
| Maximum total $I_{OL}$ for all port pins | $I_{OLT}$ | — | — | 100 | mA |
| Input high voltage<br>All ports, $\overline{IRQ}$, $\overline{RST}$, OSC1 | $V_{IH}$ | 0.7 × $V_{DD}$ | — | $V_{DD}$ | V |
| Input low voltage<br>All ports, $\overline{IRQ}$, $\overline{RST}$, OSC1 | $V_{IL}$ | $V_{SS}$ | — | 0.2 × $V_{DD}$ | V |
| $V_{DD}$ supply current | | | | | |
| Run[3] | $I_{DD}$ | — | 20 | 30 | mA |
| Wait[4] | | — | 6 | 12 | mA |
| Stop[5] | | — | 0.6 | 10 | µA |
| Stop with TBM enabled[6] | | — | 1 | 1.25 | mA |
| Stop with LVI and TBM enabled[6] | | — | 1.25 | 1.6 | mA |
| Stop with LVI | | — | 250 | 350 | µA |
| DC injection current[7] [8] [9] [10] | | | | | |
| Single pin limit | | | | | |
| $V_{in}$ > $V_{DD}$ | $I_{IC}$ | 0 | — | 2 | mA |
| $V_{in}$ < $V_{SS}$ | | 0 | — | –0.2 | |
| Total MCU limit, includes sum of all stressed pins | | | | | |
| $V_{in}$ > $V_{DD}$ | | 0 | — | 25 | |
| $V_{in}$ < $V_{SS}$ | | 0 | — | –5 | |
| I/O ports Hi-Z leakage current[11] | $I_{IL}$ | 0 | — | ±1 | µA |

Continued on next page

| Characteristic[1] | Symbol | Min | Typ[2] | Max | Unit |
|---|---|---|---|---|---|
| Pullup/pulldown resistors (as input only)<br>Ports PTA7/KBD7–PTA0/KBD0, PTC6–PTC0/$CAN_{TX}$, PTD7/T2CH1–PTD0/$\overline{SS}$ | $R_{PU}$ | 20 | 45 | 65 | kΩ |
| Capacitance<br>Ports (as input or output) | $C_{Out}$<br>$C_{In}$ | —<br>— | —<br>— | 12<br>8 | pF |
| Monitor mode entry voltage | $V_{TST}$ | $V_{DD}$ + 2.5 | — | $V_{DD}$ + 4.0 | V |
| Low-voltage inhibit, trip falling voltage | $V_{TRIPF}$ | 3.90 | 4.25 | 4.50 | V |
| Low-voltage inhibit, trip rising voltage | $V_{TRIPR}$ | 4.0 | 4.35 | 4.60 | V |
| Low-voltage inhibit reset/recover hysteresis ($V_{TRIPF}$ + $V_{HYS}$ = $V_{TRIPR}$) | $V_{HYS}$ | — | 100 | — | mV |
| POR rearm voltage[12] | $V_{POR}$ | 0 | — | 100 | mV |
| POR reset voltage[13] | $V_{PORRST}$ | 0 | 700 | 800 | mV |
| POR rise time ramp rate[14] | $R_{POR}$ | 0.035 | — | — | V/ms |

1. $V_{DD}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A$ = $T_A$ (min) to $T_A$ (max), unless otherwise noted
2. Typical values reflect average measurements at midpoint of voltage range, 25°C only.
3. Run (operating) $I_{DD}$ measured using external square wave clock source ($f_{OSC}$ = 32 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. $C_L$ = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run $I_{DD}$. Measured with all modules enabled.
4. Wait $I_{DD}$ measured using external square wave clock source ($f_{OSC}$ = 32 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. $C_L$ = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait $I_{DD}$. Measured with CGM and LVI enabled.
5. Stop $I_{DD}$ is measured with OSC1 = $V_{SS}$. All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. All ports configured as inputs. Typical values at midpoint of voltage range, 25°C only.
6. Stop $I_{DD}$ with TBM enabled is measured using an external square wave clock source ($f_{OSC}$ = 8 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. All inputs configured as inputs.
7. This parameter is characterized and not tested on each device.
8. All functional non-supply pins are internally clamped to $V_{SS}$ and $V_{DD}$.
9. 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.
10. 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).
11. Pullups and pulldowns are disabled. Port B leakage is specified in 21.10 5.0-Volt ADC Characteristics.
12. Maximum is highest voltage that POR is guaranteed.
13. Maximum is highest voltage that POR is possible.
14. If minimum $V_{DD}$ is not reached before the internal POR reset is released, $\overline{RST}$ must be driven low externally until minimum $V_{DD}$ is reached.

## 21.6 3.3-Vdc Electrical Characteristics

| Characteristic[1] | Symbol | Min | Typ[2] | Max | Unit |
|---|---|---|---|---|---|
| Output high voltage | | | | | |
| ($I_{Load}$ = –0.6 mA) all I/O pins | $V_{OH}$ | $V_{DD}$ – 0.3 | — | — | V |
| ($I_{Load}$ = –4.0 mA) all I/O pins | $V_{OH}$ | $V_{DD}$ – 1.0 | — | — | V |
| ($I_{Load}$ = –10.0 mA) pins PTC0–PTC4, PTF0–PTF3 only | $V_{OH}$ | $V_{DD}$ – 1.0 | — | — | V |
| Maximum combined $I_{OH}$ for port PTA7–PTA3, port PTC0–PTC1, port E, port PTD0–PTD3, port PTF0–PTF3, port PTG4–PTG7 | $I_{OH1}$ | — | — | 30 | mA |
| Maximum combined $I_{OH}$ for port PTA2–PTA0, port B, port PTC2–PTC6, port PTD4–PTD7 port PTF4–PTF7, port PTG0–PTG3 | $I_{OH2}$ | — | — | 30 | mA |
| Maximum total $I_{OH}$ for all port pins | $I_{OHT}$ | — | — | 60 | mA |
| Output low voltage | | | | | |
| ($I_{Load}$ = 0.5 mA) all I/O pins | $V_{OL}$ | — | — | 0.3 | V |
| ($I_{Load}$ = 5 mA) all I/O pins | $V_{OL}$ | — | — | 1.0 | V |
| ($I_{Load}$ = 10 mA) pins PTC0–PTC4, PTF0–PTF3 only | $V_{OL}$ | — | — | 0.8 | V |
| Maximum combined $I_{OH}$ for port PTA7–PTA3, port PTC0–PTC1, port E, port PTD0–PTD3 port PTF0–PTF3, port PTG4–PTG7 | $I_{OL1}$ | — | — | 30 | mA |
| Maximum combined $I_{OH}$ for port PTA2–PTA0, port B, port PTC2–PTC6, port PTD4–PTD7 port PTF4–PTF7, port PTG0–PTG3 | $I_{OL2}$ | — | — | 30 | mA |
| Maximum total $I_{OL}$ for all port pins | $I_{OLT}$ | — | — | 60 | mA |
| Input high voltage All ports, $\overline{IRQ}$, $\overline{RST}$, OSC1 | $V_{IH}$ | 0.7 × $V_{DD}$ | — | $V_{DD}$ | V |
| Input low voltage All ports, $\overline{IRQ}$, $\overline{RST}$, OSC1 | $V_{IL}$ | $V_{SS}$ | — | 0.3 × $V_{DD}$ | V |
| $V_{DD}$ supply current | | | | | |
| Run[3] | $I_{DD}$ | — | 8 | 12 | mA |
| Wait[4] | | — | 3 | 6 | mA |
| Stop[5] | | — | 0.5 | 6 | µA |
| Stop with TBM enabled[6] | | — | 500 | 700 | µA |
| Stop with LVI and TBM enabled[6] | | — | 700 | 900 | µA |
| Stop with LVI | | — | 200 | 300 | µA |
| DC injection current[7] [8] [9] [10] | | | | | |
| Single pin limit | | | | | |
| $V_{in}$ > $V_{DD}$ | $I_{IC}$ | 0 | — | 2 | mA |
| $V_{in}$ < $V_{SS}$ | | 0 | — | –0.2 | |
| Total MCU limit, includes sum of all stressed pins | | | | | |
| $V_{in}$ > $V_{DD}$ | | 0 | — | 25 | |
| $V_{in}$ < $V_{SS}$ | | 0 | — | –5 | |
| I/O ports Hi-Z leakage current[11] | $I_{IL}$ | 0 | — | ±1 | µA |

Continued on next page

| Characteristic[1] | Symbol | Min | Typ[2] | Max | Unit |
|---|---|---|---|---|---|
| Pullup/pulldown resistors (as input only)<br>Ports PTA7/KBD7–PTA0/KBD0, PTC6–PTC0, PTD7/T2CH1–PTD0/$\overline{SS}$ | $R_{PU}$ | 20 | 45 | 65 | kΩ |
| Capacitance<br>Ports (as input or output) | $C_{Out}$<br>$C_{In}$ | —<br>— | —<br>— | 12<br>8 | pF |
| Monitor mode entry voltage | $V_{TST}$ | $V_{DD}$ + 2.5 | — | $V_{DD}$ + 4.0 | V |
| Low-voltage inhibit, trip falling voltage | $V_{TRIPF}$ | 2.35 | 2.6 | 2.8 | V |
| Low-voltage inhibit, trip rising voltage | $V_{TRIPR}$ | 2.4 | 2.66 | 2.9 | V |
| Low-voltage inhibit reset/recover hysteresis ($V_{TRIPF}$ + $V_{HYS}$ = $V_{TRIPR}$) | $V_{HYS}$ | — | 100 | — | mV |
| POR rearm voltage[12] | $V_{POR}$ | 0 | — | 100 | mV |
| POR reset voltage[13] | $V_{PORRST}$ | 0 | 700 | 800 | mV |
| POR rise time ramp rate[14] | $R_{POR}$ | 0.02 | — | — | V/ms |

1. $V_{DD}$ = 3.3 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A$ = $T_A$ (min) to $T_A$ (max), unless otherwise noted
2. Typical values reflect average measurements at midpoint of voltage range, 25°C only.
3. Run (operating) $I_{DD}$ measured using external square wave clock source ($f_{OSC}$ = 16 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. $C_L$ = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run $I_{DD}$. Measured with all modules enabled.
4. Wait $I_{DD}$ measured using external square wave clock source ($f_{OSC}$ = 16 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. $C_L$ = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait $I_{DD}$. Measured with CGM and LVI enabled.
5. Stop $I_{DD}$ is measured with OSC1 = $V_{SS}$. All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. All ports configured as inputs. Typical values at midpoint of voltage range, 25°C only.
6. Stop $I_{DD}$ with TBM enabled is measured using an external square wave clock source ($f_{OSC}$ = 4 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. All inputs configured as inputs.
7. This parameter is characterized and not tested on each device.
8. All functional non-supply pins are internally clamped to $V_{SS}$ and $V_{DD}$.
9. 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.
10. 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).
11. Pullups and pulldowns are disabled.
12. Maximum is highest voltage that POR is guaranteed.
13. Maximum is highest voltage that POR is possible.
14. If minimum $V_{DD}$ is not reached before the internal POR reset is released, $\overline{RST}$ must be driven low externally until minimum $V_{DD}$ is reached.

## 21.7 5.0-Volt Control Timing

| Characteristic[1] | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Frequency of operation<br>Crystal option<br>External clock option[2] | $f_{OSC}$ | <br>1<br>dc | <br>8<br>32 | MHz |
| Internal operating frequency | $f_{OP}$ ($f_{Bus}$) | — | 8 | MHz |
| Internal clock period ($1/f_{OP}$) | $t_{CYC}$ | 125 | — | ns |
| RESET input pulse width low | $t_{RL}$ | 100 | — | ns |
| $\overline{IRQ}$ interrupt pulse width low (edge-triggered) | $t_{ILIH}$ | 100 | — | ns |
| $\overline{IRQ}$ interrupt pulse period[3] | $t_{ILIL}$ | Note 3 | — | $t_{CYC}$ |

1. $V_{SS}$ = 0 Vdc; timing shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ unless otherwise noted.
2. No more than 10% duty cycle deviation from 50%.
3. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 $t_{CYC}$.

## 21.8 3.3-Volt Control Timing

| Characteristic[1] | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Frequency of operation<br>Crystal option<br>External clock option[2] | $f_{OSC}$ | <br>1<br>dc | <br>8<br>16 | MHz |
| Internal operating frequency | $f_{OP}$ ($f_{Bus}$) | — | 4 | MHz |
| Internal clock period ($1/f_{OP}$) | $t_{CYC}$ | 250 | — | ns |
| RESET input pulse width low | $t_{RL}$ | 200 | — | ns |
| $\overline{IRQ}$ interrupt pulse width low (edge-triggered) | $t_{ILIH}$ | 200 | — | ns |
| $\overline{IRQ}$ interrupt pulse period[3] | $t_{ILIL}$ | Note 3 | — | $t_{CYC}$ |

1. $V_{SS}$ = 0 Vdc; timing shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ unless otherwise noted.
2. No more than 10% duty cycle deviation from 50%.
3. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 $t_{CYC}$.

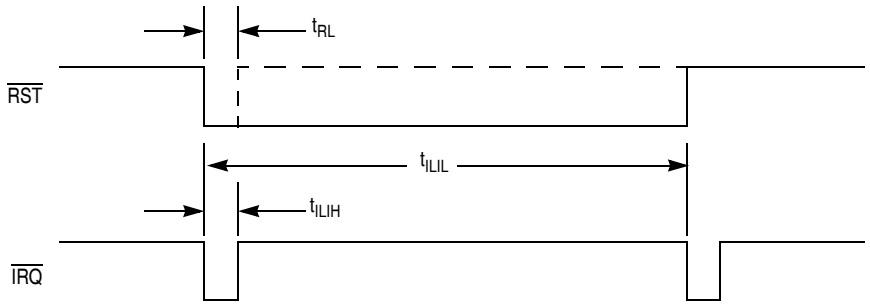



**Figure 21-1. $\overline{RST}$ and $\overline{IRQ}$ Timing**

# 21.9 Clock Generation Module (CGM) Characteristics

## 21.9.1 CGM Operating Conditions

| Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Operating voltage | $V_{DDA}$<br>$V_{SSA}$ | $V_{DD}$ – 0.3<br>$V_{SS}$ – 0.3 | —<br>— | $V_{DD}$ + 0.3<br>$V_{SS}$ + 0.3 | V |
| Crystal reference frequency | $f_{RCLK}$ | 1 | — | 8 | MHz |
| Input clock frequency (PLL off)(1) | $f_{XCLK}$ | — | — | 32 | MHz |
| Range nominal multiplier | $f_{NOM}$ | — | 71.42 | — | kHz |
| VCO center-of-range frequency(2) | $f_{VRS}$ | 71.42k | — | 40M | Hz |
| VCO operating frequency(3) | $f_{VCLK}$ | 71.42k | — | 32M | Hz |

1. External square wave applied to OSC1. Voltage levels must be rail-to-rail and duty cycle must be 50%.
2. Range of frequencies that VCO can produce to generate input clock to frequency divider during acquisition and tracking modes.
3. Allowed VCO operating range.

## 21.9.2 CGM Component Information

| Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Crystal frequency | $f_{XCLK}$ | 1 | — | 8 | MHz |
| Crystal load capacitance(1) | $C_L$ | — | 20 | — | pF |
| Crystal fixed capacitance(2) | $C_1$ | — | $(2 \times C_L) - 5$ | 47 | pF |
| Crystal tuning capacitance(3) | $C_2$ | — | $(2 \times C_L) - 5$ | 47 | pF |
| Feedback bias resistor | $R_B$ | — | 1 | 10 | MΩ |
| Series damping resistor | $R_S$ | 0 | 0 | — | kΩ |
| $V_{DDA}/V_{SSA}$ bypass capacitor | $C_{byp}$ | — | 0.1 | — | µF |
| CGMXFC filter values | See Table 4-5. Example Filter Component Values | | | | |

1. Consult crystal manufacturer's specification.
2. Capacitor on OSC1 pin. Does not include parasitic capacitance due to package, pin, and board.
3. Capacitor on OSC2 pin. Does not include parasitic capacitance due to package, pin, and board.

### 21.9.3 CGM Acquisition/Lock Time Information

| Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Acquisition mode entry frequency tolerance(1) | $\Delta_{ACQ}$ | ± 3.6 | — | ± 7.2 | % |
| Tracking mode entry frequency tolerance(2) | $\Delta_{TRK}$ | 0 | — | ± 3.6 | % |
| LOCK entry frequency tolerance(3) | $\Delta_{LOCK}$ | 0 | — | ± 0.9 | % |
| LOCK exit frequency tolerance(4) | $\Delta_{UNL}$ | ± 0.9 | — | ± 1.8 | % |
| Reference cycles per acquisition mode period | $n_{ACQ}$ | — | 32 | — | |
| Reference cycles per tracking mode period | $n_{TRK}$ | — | 128 | — | |
| Automatic mode time to stable | $t_{ACQ}$ | $n_{ACQ}/f_{RCLK}$ | See note(5) | — | s |
| Automatic stable to lock time | $t_{AL}$ | $n_{TRK}/f_{RCLK}$ | See note(6) | — | s |
| Automatic lock time ($t_{ACQ} + t_{AL}$)(7) | $t_{LOCK}$ | — | 5 | 25 | ms |
| PLL jitter, deviation of average bus frequency over 2 ms period | $f_J$ | 0 | — | $f_{RCLK}$ x 0.025% x N/4 | Hz |

1. Deviation between VCO frequency and desired frequency to enter PLL acquisition mode.
2. Deviation between VCO frequency and desired frequency to enter PLL tracking mode (stable).
3. Deviation between VCO frequency and desired frequency to enter locked mode.
4. Deviation between VCO frequency and desired frequency to exit locked mode.
5. Acquisition time is an integer multiple of reference cycles divided by reference clock.
6. Stable to lock time is an integer multiple of reference cycles divided by reference clock.
7. Maximum lock time depends on CGMXFC filter components, power supply filtering, and reference clock stability. PLL may not lock if improper components or poor filtering and layout are used.

## 21.10 5.0-Volt ADC Characteristics

| Characteristic[1] | Symbol | Min | Max | Unit | Comments |
|---|---|---|---|---|---|
| Supply voltage | $V_{DDAD}$ | 4.5 | 5.5 | V | $V_{DDAD}$ should be tied to the same potential as $V_{DD}$ via separate traces. |
| Input voltages | $V_{ADIN}$ | 0 | $V_{DDAD}$ | V | $V_{ADIN} <= V_{DDAD}$ |
| Resolution | $B_{AD}$ | 10 | 10 | Bits | |
| Absolute accuracy | $A_{AD}$ | –4 | +4 | LSB | Includes quantization |
| ADC internal clock | $f_{ADIC}$ | 500 k | 1.048 M | Hz | $t_{AIC} = 1/f_{ADIC}$ |
| Conversion range | $R_{AD}$ | $V_{SSAD}$ | $V_{DDAD}$ | V | |
| Power-up time | $t_{ADPU}$ | 16 | — | $t_{AIC}$ cycles | |
| Conversion time | $t_{ADC}$ | 16 | 17 | $t_{AIC}$ cycles | |
| Sample time | $t_{ADS}$ | 5 | — | $t_{AIC}$ cycles | |
| Monotonicity | $M_{AD}$ | Guaranteed | | | |
| Zero input reading | $Z_{ADI}$ | 000 | 003 | Hex | $V_{ADIN} = V_{SSA}$ |
| Full-scale reading | $F_{ADI}$ | 3FC | 3FF | Hex | $V_{ADIN} = V_{DDA}$ |
| Input capacitance | $C_{ADI}$ | — | 30 | pF | Not tested |
| $V_{DDAD}/V_{REFH}$ current | $I_{VREF}$ | — | 1.6 | mA | |
| Absolute accuracy (8-bit truncation mode) | $A_{AD}$ | –1 | +1 | LSB | Includes quantization |
| Quantization error (8-bit truncation mode) | — | –1/8 | +7/8 | LSB | |

1. $V_{DD}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $V_{DDAD}/V_{REFH}$ = 5.0 Vdc ± 10%, $V_{SSAD}/V_{REFL}$ = 0 Vdc

## 21.11 3.3-Volt ADC Characteristics

| Characteristic[1] | Symbol | Min | Max | Unit | Comments |
|---|---|---|---|---|---|
| Supply voltage | $V_{DDAD}$ | 3.0 | 3.6 | V | $V_{DDAD}$ should be tied to the same potential as $V_{DD}$ via separate traces. |
| Input voltages | $V_{ADIN}$ | 0 | $V_{DDAD}$ | V | $V_{ADIN} <= V_{DDAD}$ |
| Resolution | $B_{AD}$ | 10 | 10 | Bits | |
| Absolute accuracy | $A_{AD}$ | –6 | +6 | LSB | Includes quantization |
| ADC internal clock | $f_{ADIC}$ | 500 k | 1.048 M | Hz | $t_{AIC} = 1/f_{ADIC}$ |
| Conversion range | $R_{AD}$ | $V_{SSAD}$ | $V_{DDAD}$ | V | |
| Power-up time | $t_{ADPU}$ | 16 | — | $t_{AIC}$ cycles | |
| Conversion time | $t_{ADC}$ | 16 | 17 | $t_{AIC}$ cycles | |
| Sample time | $t_{ADS}$ | 5 | — | $t_{AIC}$ cycles | |
| Monotonicity | $M_{AD}$ | Guaranteed | | | |
| Zero input reading | $Z_{ADI}$ | 000 | 005 | Hex | $V_{ADIN} = V_{SSA}$ |
| Full-scale reading | $F_{ADI}$ | 3FA | 3FF | Hex | $V_{ADIN} = V_{DDA}$ |
| Input capacitance | $C_{ADI}$ | — | 30 | pF | Not tested |
| $V_{DDAD}/V_{REFH}$ current | $I_{VREF}$ | — | 1.2 | mA | |
| Absolute accuracy (8-bit truncation mode) | $A_{AD}$ | –1 | +1 | LSB | Includes quantization |
| Quantization error (8-bit truncation mode) | — | –1/8 | +7/8 | LSB | |

1. $V_{DD}$ = 3.3 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $V_{DDAD}/V_{REFH}$ = 3.3 Vdc ± 10%, $V_{SSAD}/V_{REFL}$ = 0 Vdc

## 21.12 5.0-Volt SPI Characteristics

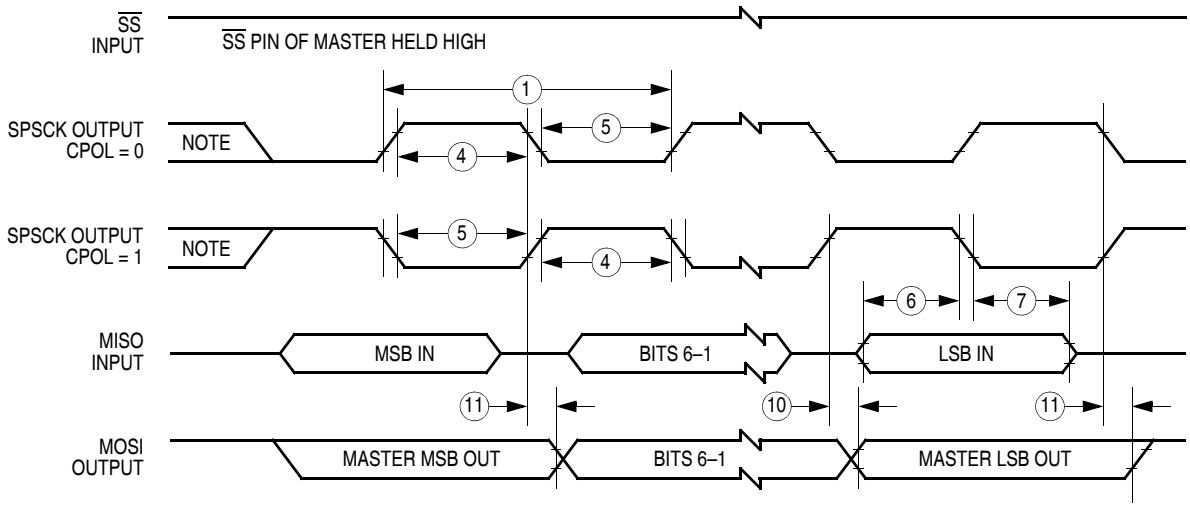
| Diagram Number[1] | Characteristic[2] | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| | Operating frequency<br>Master<br>Slave | $f_{OP(M)}$<br>$f_{OP(S)}$ | $f_{OP}/128$<br>dc | $f_{OP}/2$<br>$f_{OP}$ | MHz<br>MHz |
| 1 | Cycle time<br>Master<br>Slave | $t_{CYC(M)}$<br>$t_{CYC(S)}$ | 2<br>1 | 128<br>— | $t_{CYC}$<br>$t_{CYC}$ |
| 2 | Enable lead time | $t_{Lead(S)}$ | 1 | — | $t_{CYC}$ |
| 3 | Enable lag time | $t_{Lag(S)}$ | 1 | — | $t_{CYC}$ |
| 4 | Clock (SPSCK) high time<br>Master<br>Slave | $t_{SCKH(M)}$<br>$t_{SCKH(S)}$ | $t_{CYC}$ –25<br>1/2 $t_{CYC}$ –25 | 64 $t_{CYC}$<br>— | ns<br>ns |
| 5 | Clock (SPSCK) low time<br>Master<br>Slave | $t_{SCKL(M)}$<br>$t_{SCKL(S)}$ | $t_{CYC}$ –25<br>1/2 $t_{CYC}$ –25 | 64 $t_{CYC}$<br>— | ns<br>ns |
| 6 | Data setup time (inputs)<br>Master<br>Slave | $t_{SU(M)}$<br>$t_{SU(S)}$ | 30<br>30 | —<br>— | ns<br>ns |
| 7 | Data hold time (inputs)<br>Master<br>Slave | $t_{H(M)}$<br>$t_{H(S)}$ | 30<br>30 | —<br>— | ns<br>ns |
| 8 | Access time, slave[3]<br>CPHA = 0<br>CPHA = 1 | $t_{A(CP0)}$<br>$t_{A(CP1)}$ | 0<br>0 | 40<br>40 | ns<br>ns |
| 9 | Disable time, slave[4] | $t_{DIS(S)}$ | — | 40 | ns |
| 10 | Data valid time, after enable edge<br>Master<br>Slave[5] | $t_{V(M)}$<br>$t_{V(S)}$ | —<br>— | 50<br>50 | ns<br>ns |
| 11 | Data hold time, outputs, after enable edge<br>Master<br>Slave | $t_{HO(M)}$<br>$t_{HO(S)}$ | 0<br>0 | —<br>— | ns<br>ns |

1. Numbers refer to dimensions in Figure 21-2 and Figure 21-3.
2. All timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$, unless noted; 100 pF load on all SPI pins.
3. Time to data active from high-impedance state
4. Hold time to high-impedance state
5. With 100 pF on all SPI pins

## 21.13 3.3-Volt SPI Characteristics

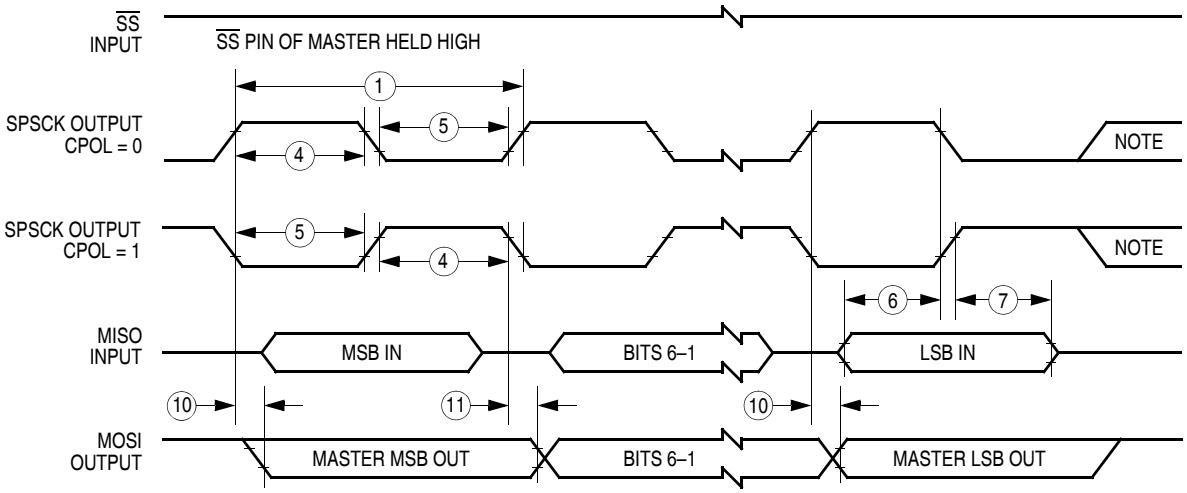
| Diagram Number[1] | Characteristic[2] | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| | Operating frequency<br>Master<br>Slave | $f_{OP(M)}$<br>$f_{OP(S)}$ | $f_{OP}/128$<br>DC | $f_{OP}/2$<br>$f_{OP}$ | MHz<br>MHz |
| 1 | Cycle time<br>Master<br>Slave | $t_{CYC(M)}$<br>$t_{CYC(S)}$ | 2<br>1 | 128<br>— | $t_{cyc}$<br>$t_{cyc}$ |
| 2 | Enable lead time | $t_{Lead(S)}$ | 1 | — | $t_{cyc}$ |
| 3 | Enable lag time | $t_{Lag(S)}$ | 1 | — | $t_{cyc}$ |
| 4 | Clock (SPSCK) high time<br>Master<br>Slave | $t_{SCKH(M)}$<br>$t_{SCKH(S)}$ | $t_{cyc}$ –35<br>1/2 $t_{cyc}$ –35 | 64 $t_{cyc}$<br>— | ns<br>ns |
| 5 | Clock (SPSCK) low time<br>Master<br>Slave | $t_{SCKL(M)}$<br>$t_{SCKL(S)}$ | $t_{cyc}$ –35<br>1/2 $t_{cyc}$ –35 | 64 $t_{cyc}$<br>— | ns<br>ns |
| 6 | Data setup time (inputs)<br>Master<br>Slave | $t_{SU(M)}$<br>$t_{SU(S)}$ | 40<br>40 | —<br>— | ns<br>ns |
| 7 | Data hold time (inputs)<br>Master<br>Slave | $t_{H(M)}$<br>$t_{H(S)}$ | 40<br>40 | —<br>— | ns<br>ns |
| 8 | Access time, slave[3]<br>CPHA = 0<br>CPHA = 1 | $t_{A(CP0)}$<br>$t_{A(CP1)}$ | 0<br>0 | 50<br>50 | ns<br>ns |
| 9 | Disable time, slave[4] | $t_{DIS(S)}$ | — | 50 | ns |
| 10 | Data valid time, after enable edge<br>Master<br>Slave[5] | $t_{V(M)}$<br>$t_{V(S)}$ | —<br>— | 60<br>60 | ns<br>ns |
| 11 | Data hold time, outputs, after enable edge<br>Master<br>Slave | $t_{HO(M)}$<br>$t_{HO(S)}$ | 0<br>0 | —<br>— | ns<br>ns |

1. Numbers refer to dimensions in Figure 21-2 and Figure 21-3.
2. All timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$, unless noted; 100 pF load on all SPI pins.
3. Time to data active from high-impedance state
4. Hold time to high-impedance state
5. With 100 pF on all SPI pins




Note: This first clock edge is generated internally, but is not seen at the SPSCK pin.

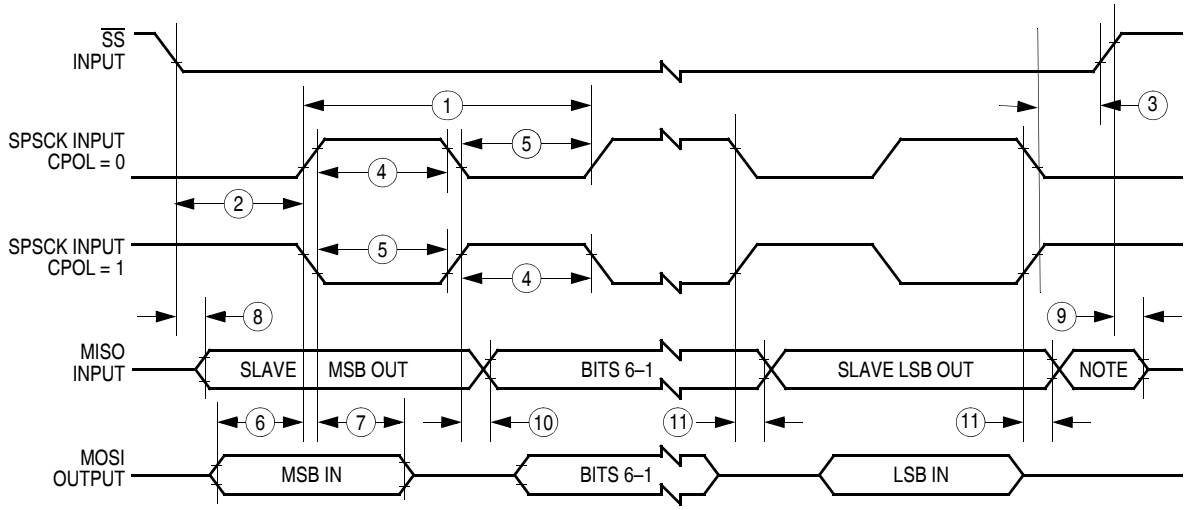
**a) SPI Master Timing (CPHA = 0)**




Note: This last clock edge is generated internally, but is not seen at the SPSCK pin.

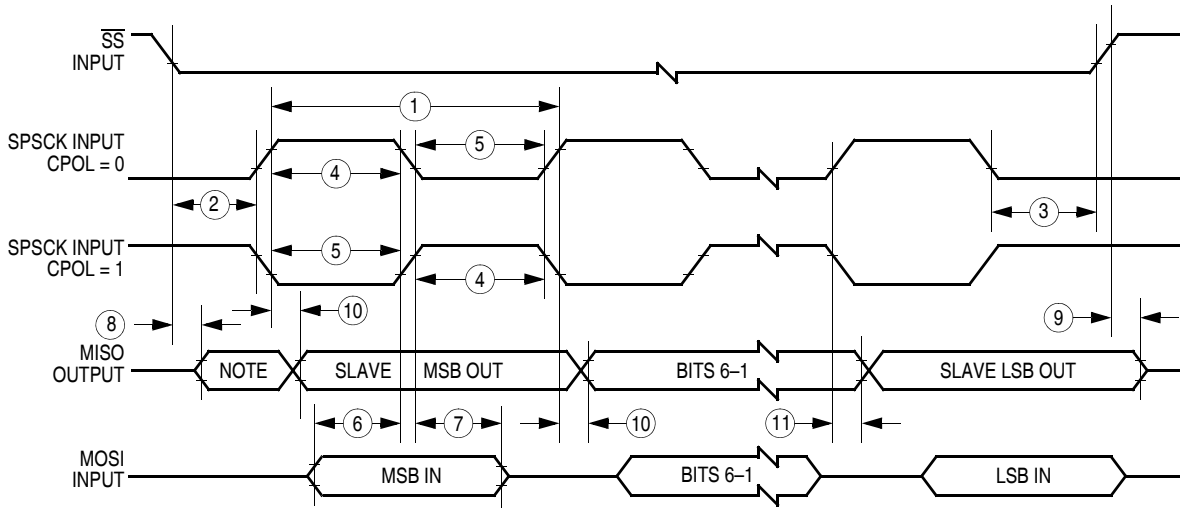
**b) SPI Master Timing (CPHA = 1)**

**Figure 21-2. SPI Master Timing**




Note: Not defined but normally MSB of character just received

**a) SPI Slave Timing (CPHA = 0)**




Note: Not defined but normally LSB of character previously transmitted

**b) SPI Slave Timing (CPHA = 1)**

**Figure 21-3. SPI Slave Timing**

## 21.14 Timer Interface Module Characteristics

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Timer input capture pulse width | $t_{TH}$, $t_{TL}$ | 2 | — | $t_{cyc}$ |
| Timer input capture period | $t_{TLTL}$ | Note(1) | — | $t_{cyc}$ |
| Timer input clock pulse width | $t_{TCL}$, $t_{TCH}$ | $t_{cyc}$ + 5 | — | ns |

1. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 $t_{cyc}$.

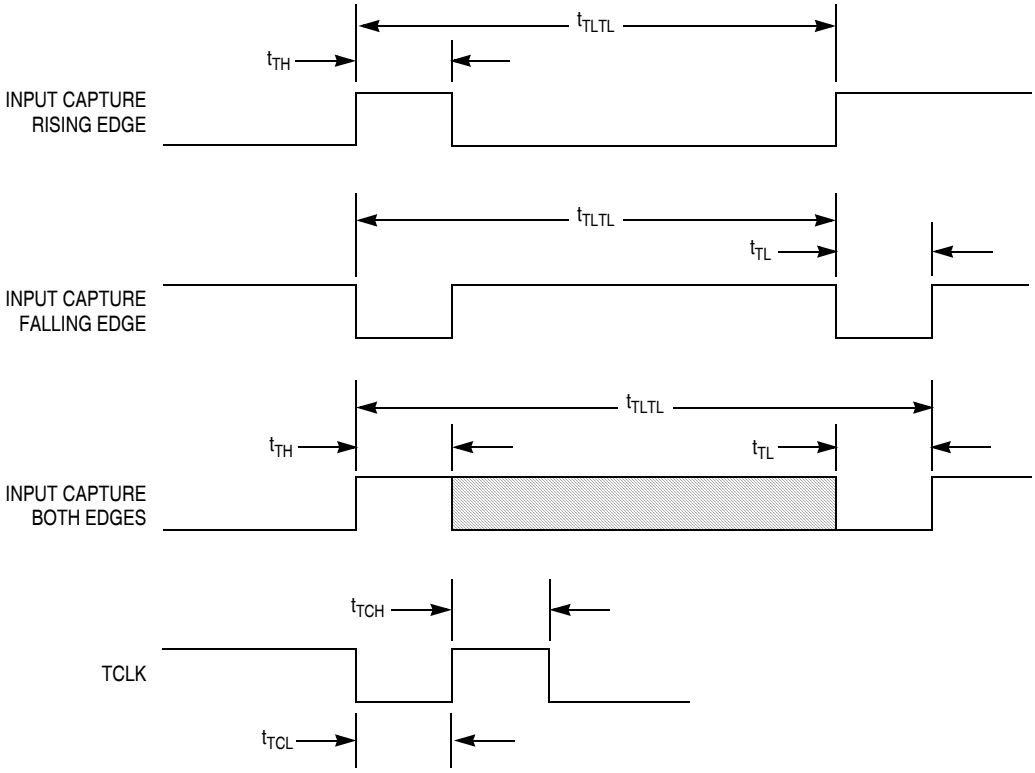



**Figure 21-4. Timer Input Timing**

## 21.15 Memory Characteristics

| Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| RAM data retention voltage | $V_{RDR}$ | 1.3 | — | — | V |
| FLASH program bus clock frequency | — | 1 | — | — | MHz |
| FLASH read bus clock frequency | $f_{Read}$ (1) | 0 | — | 8 M | Hz |
| FLASH page erase time<br><1 k cycles<br>>1 k cycles | $t_{Erase}$ | <br>0.9<br>3.6 | <br>1<br>4 | <br>1.1<br>5.5 | ms |
| FLASH mass erase time | $t_{MErase}$ | 4 | — | — | ms |
| FLASH PGM/ERASE to HVEN setup time | $t_{NVS}$ | 10 | — | — | µs |
| FLASH high-voltage hold time | $t_{NVH}$ | 5 | — | — | µs |
| FLASH high-voltage hold time (mass erase) | $t_{NVHL}$ | 100 | — | — | µs |
| FLASH program hold time | $t_{PGS}$ | 5 | — | — | µs |
| FLASH program time | $t_{PROG}$ | 30 | — | 40 | µs |
| FLASH return to read time | $t_{RCV}$ (2) | 1 | — | — | µs |
| FLASH cumulative program HV period | $t_{HV}$ (3) | — | — | 4 | ms |
| FLASH endurance(4) | — | 10 k | 100 k | — | Cycles |
| FLASH data retention time(5) | — | 15 | 100 | — | Years |

1. $f_{Read}$ is defined as the frequency range for which the FLASH memory can be read.
2. $t_{RCV}$ is defined as the time it needs before the FLASH can be read after turning off the high voltage charge pump, by clearing HVEN to 0.
3. $t_{HV}$ is defined as the cumulative high voltage programming time to the same row before next erase. $t_{HV}$ must satisfy this condition: $t_{NVS} + t_{NVH} + t_{PGS} + (t_{PROG} \times 32) \leq t_{HV}$ maximum.
4. Typical endurance was evaluated for this product family. For additional information on how Freescale defines *Typical Endurance*, please refer to Engineering Bulletin EB619.
5. Typical data retention values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25°C using the Arrhenius equation. For additional information on how Freescale defines *Typical Data Retention*, please refer to Engineering Bulletin EB618.

# Chapter 22
# Ordering Information and Mechanical Specifications

## 22.1 Introduction

This section contains ordering numbers for the MC68HC908GZ60 and gives the dimensions for:

- 32-pin low-profile quad flat pack (case 873A)
- 48-pin low-profile quad flat pack (case 932-03)
- 64-pin quad flat pack (case 840B)

The following figures show the latest package drawings at the time of this publication. To make sure that you have the latest package specifications, contact your local Freescale Sales Office.

## 22.2 MC Order Numbers

**Table 22-1. MC Order Numbers**

| MC Order Number | Operating Temperature Range | Package |
|---|---|---|
| MC908GZ60CFJ | –40°C to +85°C | 32-pin low-profile quad flat package (LQFP) |
| MC908GZ60VFJ | –40°C to +105°C | |
| MC908GZ60MFJ | –40°C to +125°C | |
| MC908GZ60CFA | –40°C to +85°C | 48-pin low-profile quad flat package (LQFP) |
| MC908GZ60VFA | –40°C to +105°C | |
| MC908GZ60MFA | –40°C to +125°C | |
| MC908GZ60CFU | –40°C to +85°C | 64-pin quad flat package (QFP) |
| MC908GZ60VFU | –40°C to +105°C | |
| MC908GZ60MFU | –40°C to +125°C | |

Temperature designators:
C = –40°C to +85°C
V = –40°C to +105°C
M = –40°C to +125°C

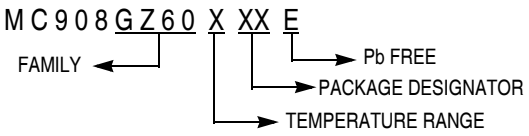
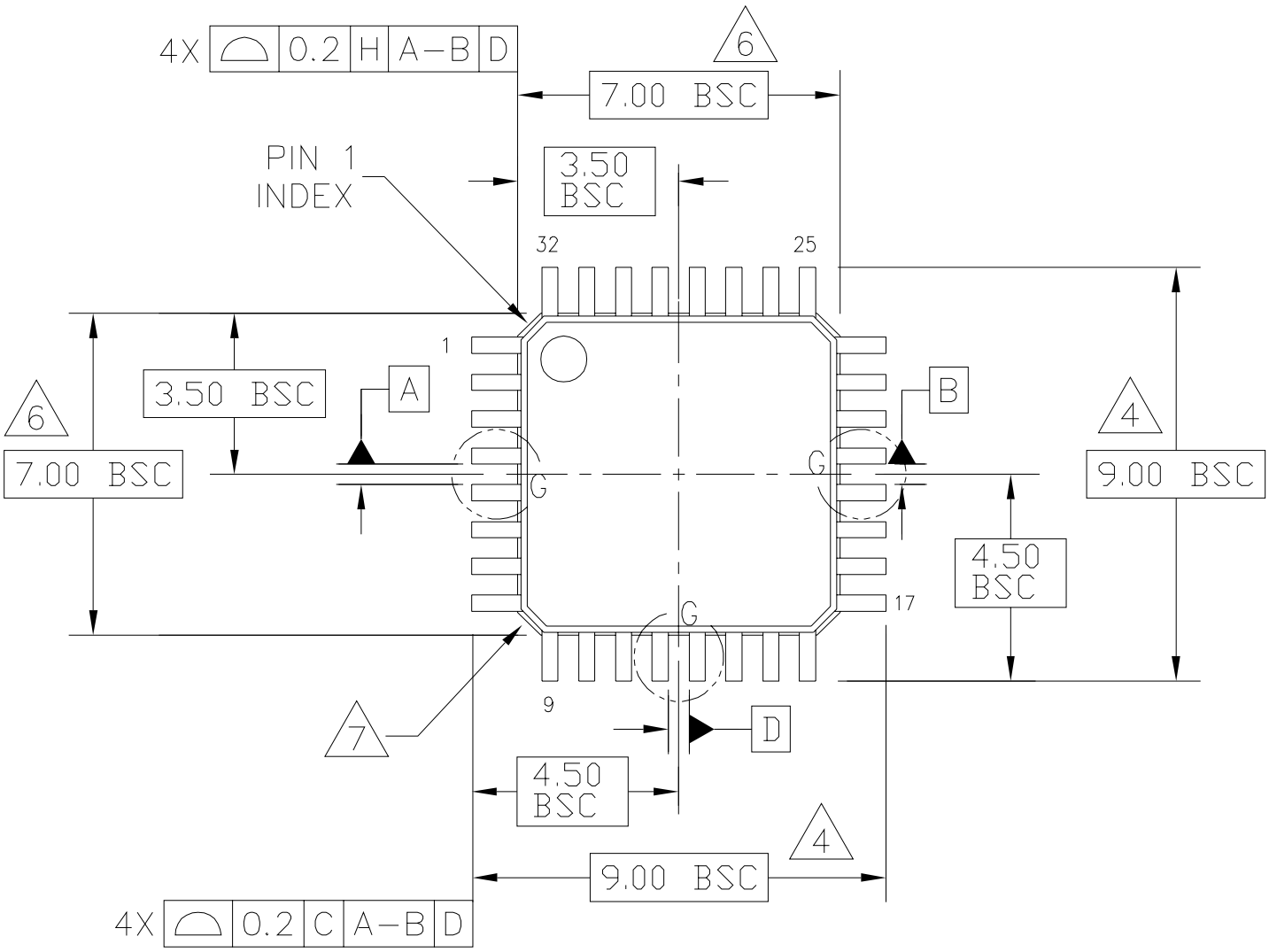
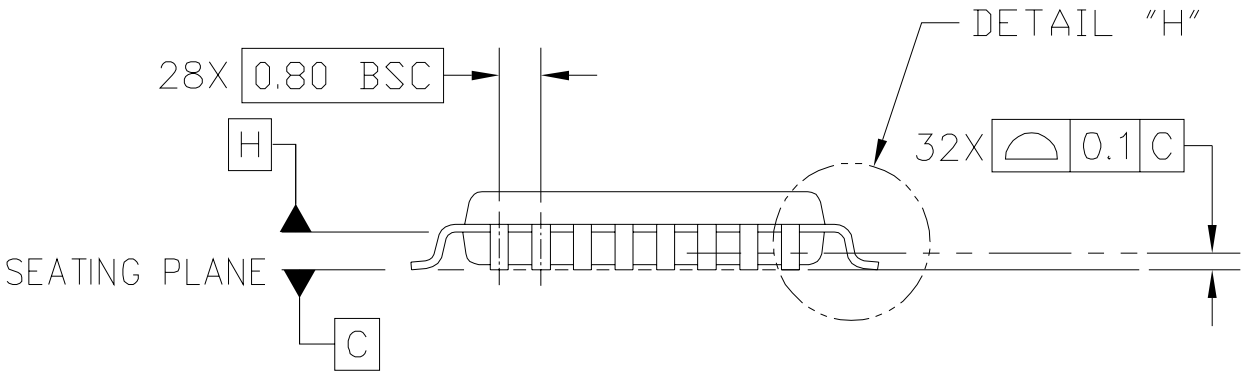



**Figure 22-1. Device Numbering System**

## 22.3 Package Dimensions

Refer to the following pages for detailed package dimensions.

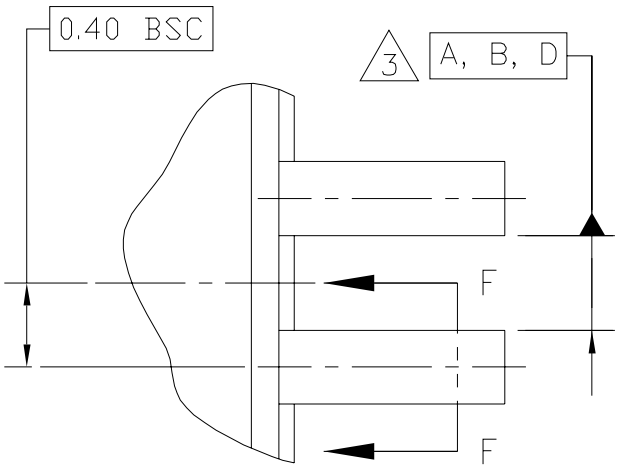


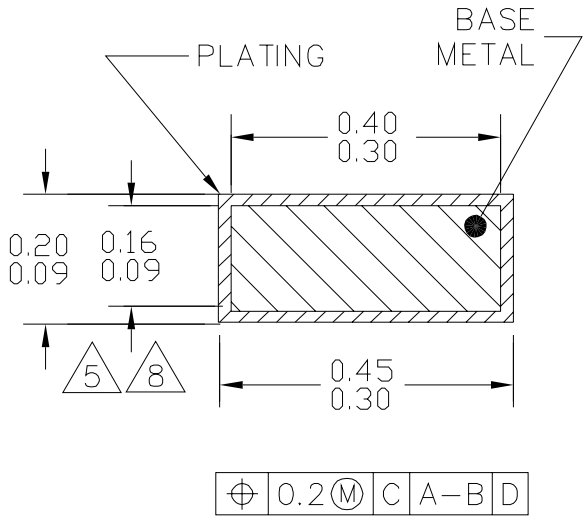

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

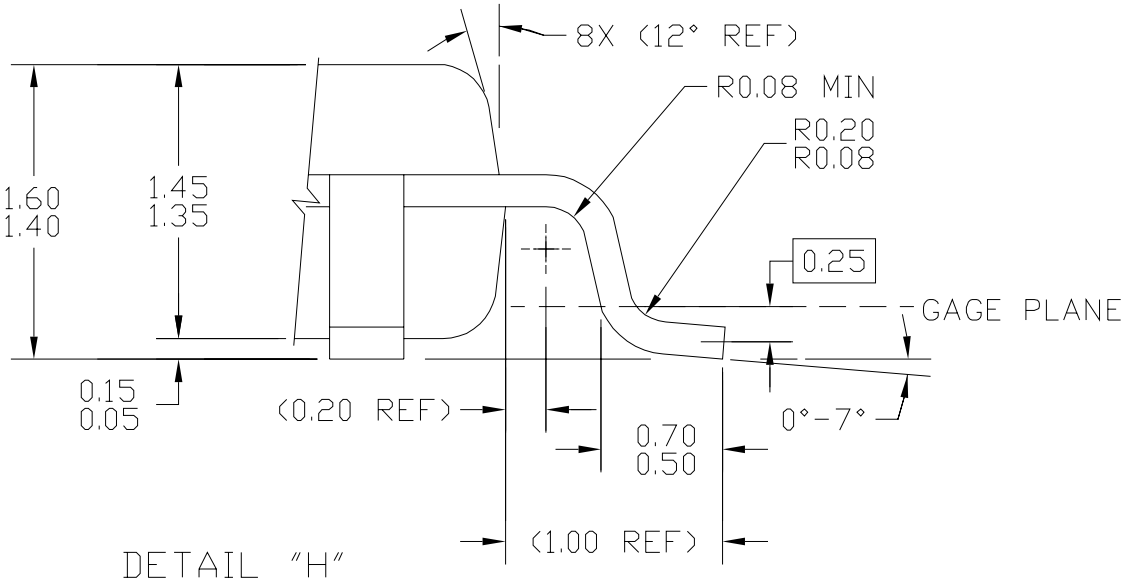
| TITLE: | DOCUMENT NO: 98ASH70029A | REV: C |
|---|---|---|
| LOW PROFILE QUAD FLAT PACK (LQFP)<br>32 LEAD, 0.8 PITCH (7 X 7 X 1.4) | CASE NUMBER: 873A-04 | 01 APR 2005 |
| | STANDARD: JEDEC MS-026 BBA | |




DETAIL G




SECTION F–F
ROTATED 90°CW
32 PLACES




DETAIL "H"

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASH70029A | REV: C |
|---|---|---|
| LOW PROFILE QUAD FLAT PACK (LQFP)<br>32 LEAD, 0.8 PITCH (7 X 7 X 1.4) | CASE NUMBER: 873A-04 | 01 APR 2005 |
| | STANDARD: JEDEC MS-026 BBA | |

NOTES:

1. DIMENSIONS ARE IN MILLIMETERS.

2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5–1994.

3. DATUMS A, B, AND D TO BE DETERMINED AT DATUM PLANE H.

4. DIMENSIONS TO BE DETERMINED AT SEATING PLANE DATUM C.

5. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM DIMENSION BY MORE THAN 0.08 MM. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND ADJACENT LEAD OR PROTRUSION: 0.07 MM.

6. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 MM PER SIDE. DIMENSIONS ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.

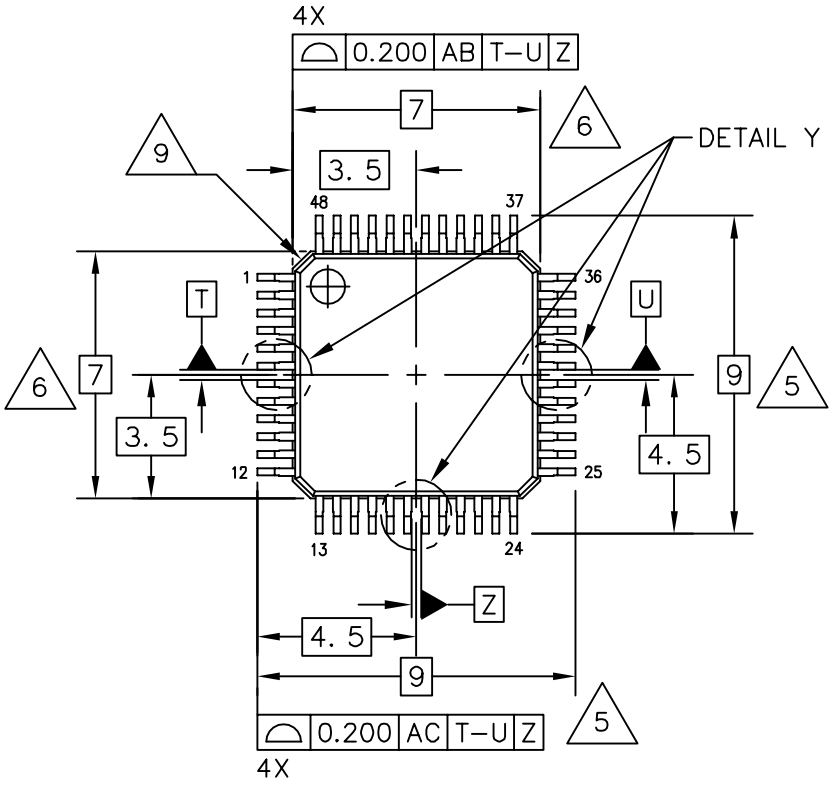
7. EXACT SHAPE OF EACH CORNER IS OPTIONAL.

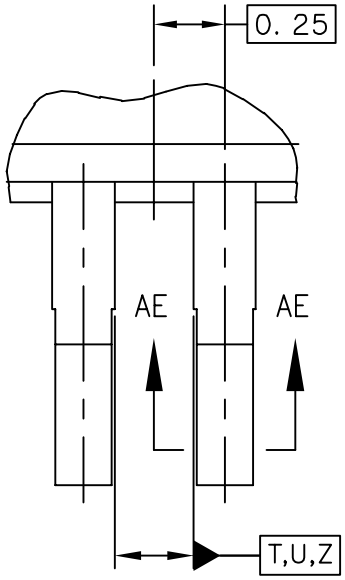
8. THESE DIMENSIONS APPLY TO THE FLAT SECTION OF THE LEAD BETWEEN 0.1 MM AND 0.25 MM FROM THE LEAD TIP.

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

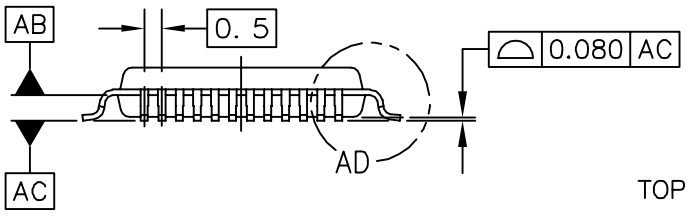
| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

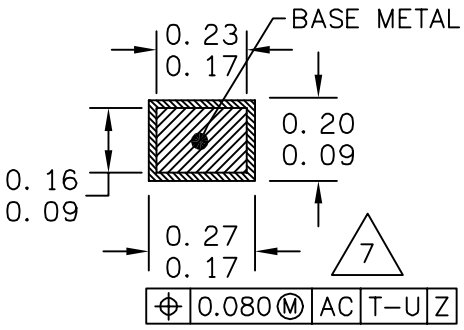
| TITLE: | DOCUMENT NO: 98ASH70029A | REV: C |
|---|---|---|
| LOW PROFILE QUAD FLAT PACK (LQFP)<br>32 LEAD, 0.8 PITCH (7 X 7 X 1.4) | CASE NUMBER: 873A-04 | 01 APR 2005 |
| | STANDARD: JEDEC MS-026 BBA | |



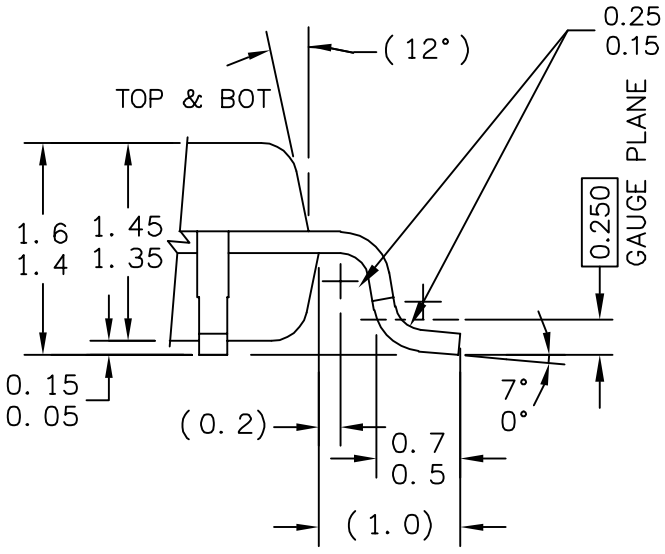




DETAIL Y




SECTION AE–AE




DETAIL AD

| © FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED. | **MECHANICAL OUTLINE** | PRINT VERSION NOT TO SCALE |
|---|---|---|
| TITLE: LQFP, 48 LEAD, 0.50 PITCH (7.0 X 7.0 X 1.4) | DOCUMENT NO: 98ASH00962A | REV: G |
| | CASE NUMBER: 932-03 | 14 APR 2005 |
| | STANDARD: JEDEC MS-026-BBC | |

NOTES:

1. DIMENSIONS AND TOLERANCING PER ASME Y14.5M–1994.

2. CONTROLLING DIMENSION: MILLIMETER.

3. DATUM PLANE AB IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.

4. DATUMS T, U, AND Z TO BE DETERMINED AT DATUM PLANE AB.

5. DIMENSIONS TO BE DETERMINED AT SEATING PLANE AC.

6. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.250 PER SIDE. DIMENSIONS DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE AB.

7. THIS DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED 0.350.

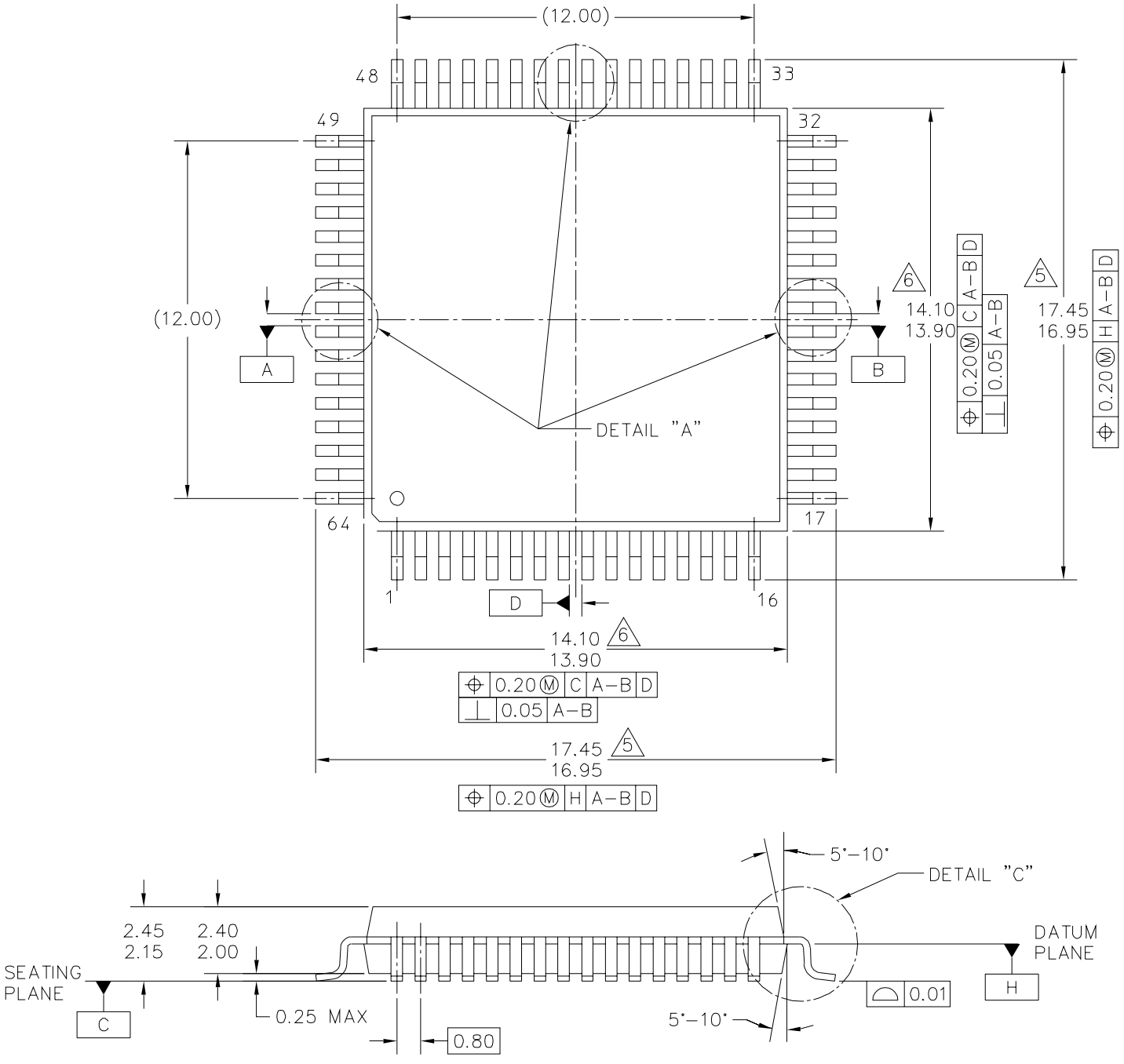
8. MINIMUM SOLDER PLATE THICKNESS SHALL BE 0.0076.

9. EXACT SHAPE OF EACH CORNER IS OPTIONAL.

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

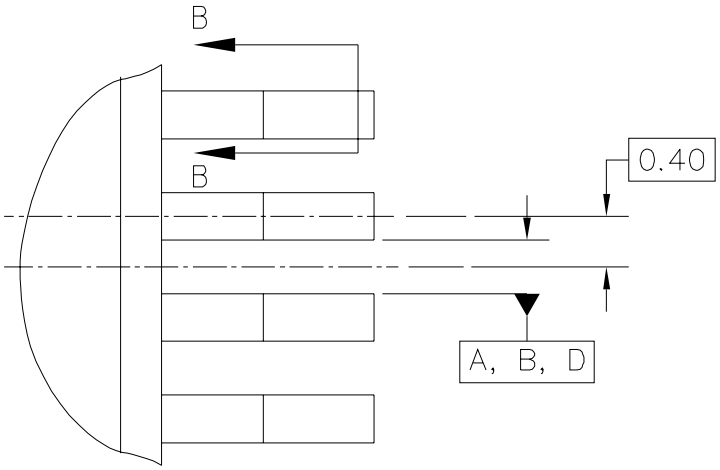
| TITLE: | DOCUMENT NO: 98ASH00962A | REV: G |
|---|---|---|
| LQFP, 48 LEAD, 0.50 PITCH (7.0 X 7.0 X 1.4) | CASE NUMBER: 932-03 | 14 APR 2005 |
| | STANDARD: JEDEC MS-026-BBC | |



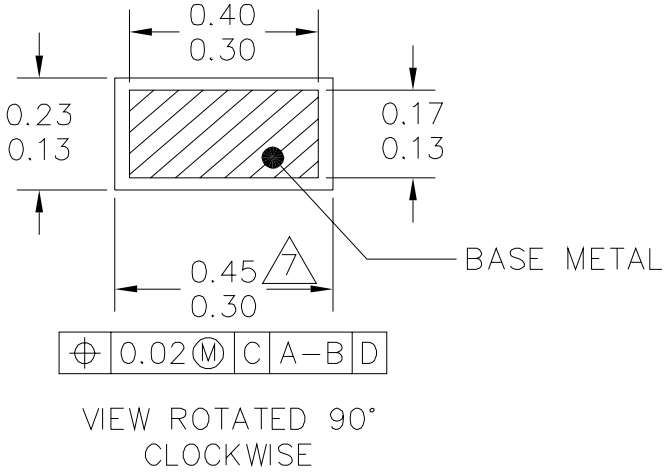

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASB42844B | REV: A |
|---|---|---|
| 64LD QFP (14 X 14) | CASE NUMBER: 840B-02 | 06 APR 2005 |
| | STANDARD: NON-JEDEC | |

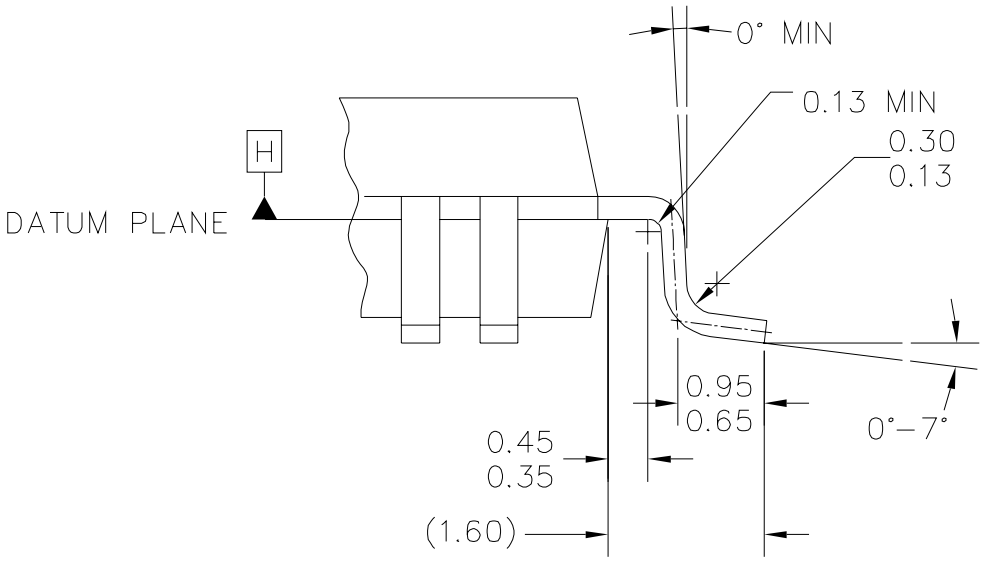



DETAIL "A"




VIEW ROTATED 90° CLOCKWISE

SECTION B–B




DETAIL "C"

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASB42844B | REV: A |
|---|---|---|
| 64LD QFP (14 X 14) | CASE NUMBER: 840B-02 | 06 APR 2005 |
| | STANDARD: NON-JEDEC | |

NOTES:

1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.

2. CONTROLLING DIMENSION: MILLIMETER.

3. DATUM PLANE −H− IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE
PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.

4. DATUMS A−B AND −D− TO BE DETERMINED AT DATUM PLANE −H−.

5. DIMENSIONS TO BE DETERMINED AT SEATING PLANE −C−.

6. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. DIMENSIONS DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE −H−.

7. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08mm TOTAL IN EXCESS OF THE DIMENSION AT MAXIMUM MATERIAL CONDICTION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

© FREESCALE SEMICONDUCTOR, INC.
ALL RIGHTS RESERVED.

| MECHANICAL OUTLINE | PRINT VERSION NOT TO SCALE |
|---|---|

| TITLE: | DOCUMENT NO: 98ASB42844B | REV: A |
|---|---|---|
| 64LD QFP (14 X 14) | CASE NUMBER: 840B-02 | 06 APR 2005 |
| | STANDARD: NON-JEDEC | |

# Appendix A
# MC68HC908GZ48

## A.1 Introduction

The MC68HC908GZ48 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

The information contained in this document pertains to the MC68HC908GZ48 with the exceptions shown in this appendix.

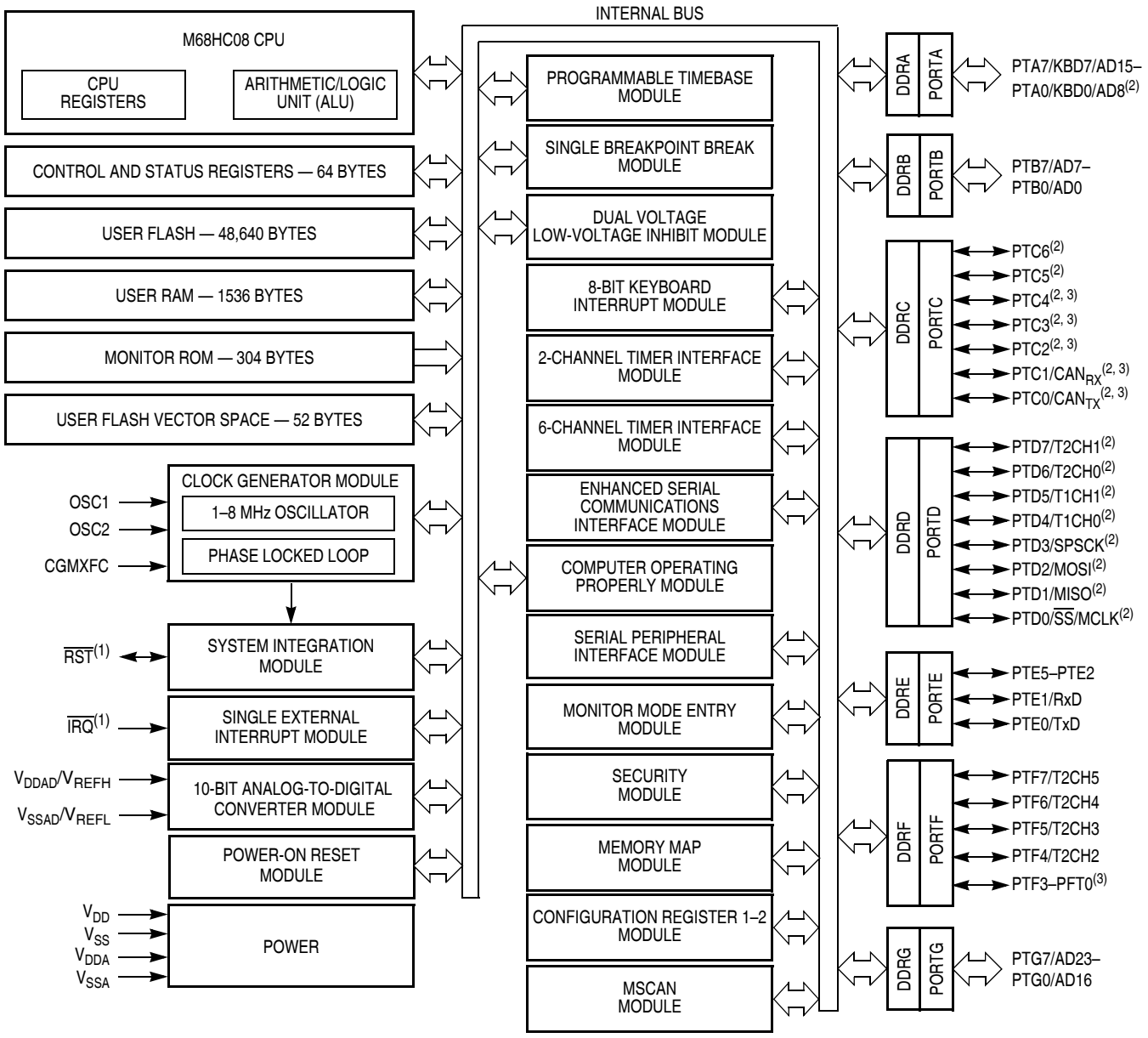
## A.2 Block Diagram

See Figure A-1.

## A.3 Memory

The MC68HC908GZ48 can address 48 Kbytes of memory space. The memory map, shown in Figure A-2, includes:

- 48 Kbytes of user FLASH memory
- 1536 bytes of random-access memory (RAM)
- 52 bytes of user-defined vectors




1. Pin contains integrated pullup device.
2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.
3. Higher current drive port pins

**Figure A-1. MC68HC908GZ48 Block Diagram**

1. $FFF6–$FFFD used for eight security bytes

**Figure A-2. MC68HC908GZ48 Memory Map**

## A.4 Ordering Information

**Table A-1. MC Order Numbers**

| MC Order Number | Operating Temperature Range | Package |
|---|---|---|
| MC908GZ48CFJ | –40°C to +85°C | 32-pin low-profile quad flat package (LQFP) |
| MC908GZ48VFJ | –40°C to +105°C | |
| MC908GZ48MFJ | –40°C to +125°C | |
| MC908GZ48CFA | –40°C to +85°C | 48-pin low-profile quad flat package (LQFP) |
| MC908GZ48VFA | –40°C to +105°C | |
| MC908GZ48MFA | –40°C to +125°C | |
| MC908GZ48CFU | –40°C to +85°C | 64-pin quad flat package (QFP) |
| MC908GZ48VFU | –40°C to +105°C | |
| MC908GZ48MFU | –40°C to +125°C | |

Temperature designators:
C = –40°C to +85°C
V = –40°C to +105°C
M = –40°C to +125°C

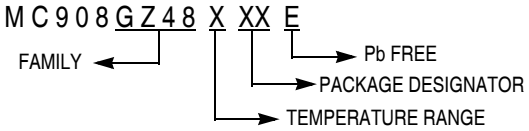



**Figure A-3. Device Numbering System**

# Appendix B
# MC68HC908GZ32

## B.1 Introduction

The MC68HC908GZ32 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

The information contained in this document pertains to the MC68HC908GZ32 with the exceptions shown in this appendix.

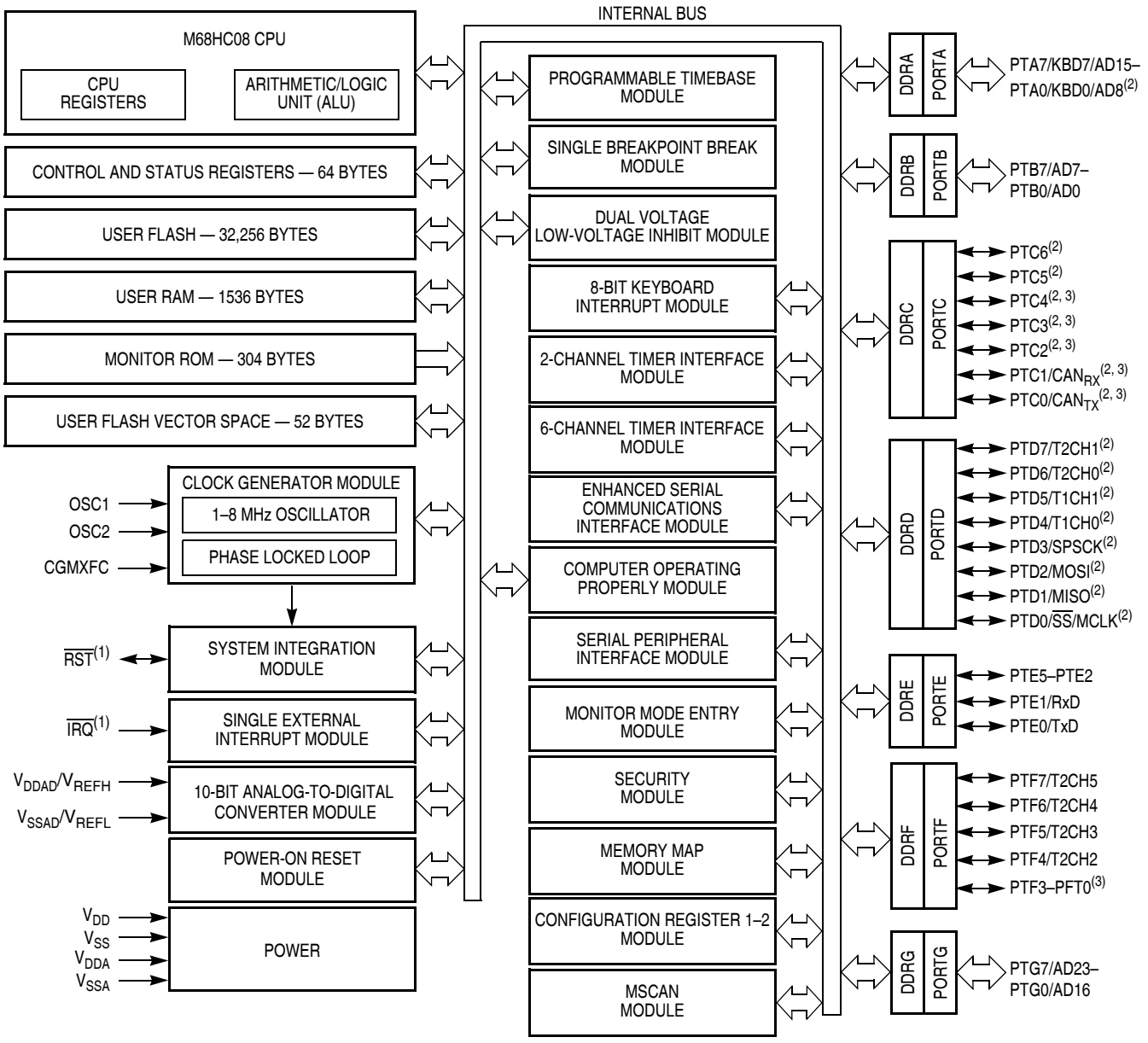
## B.2 Block Diagram

See Figure B-1.

## B.3 Memory

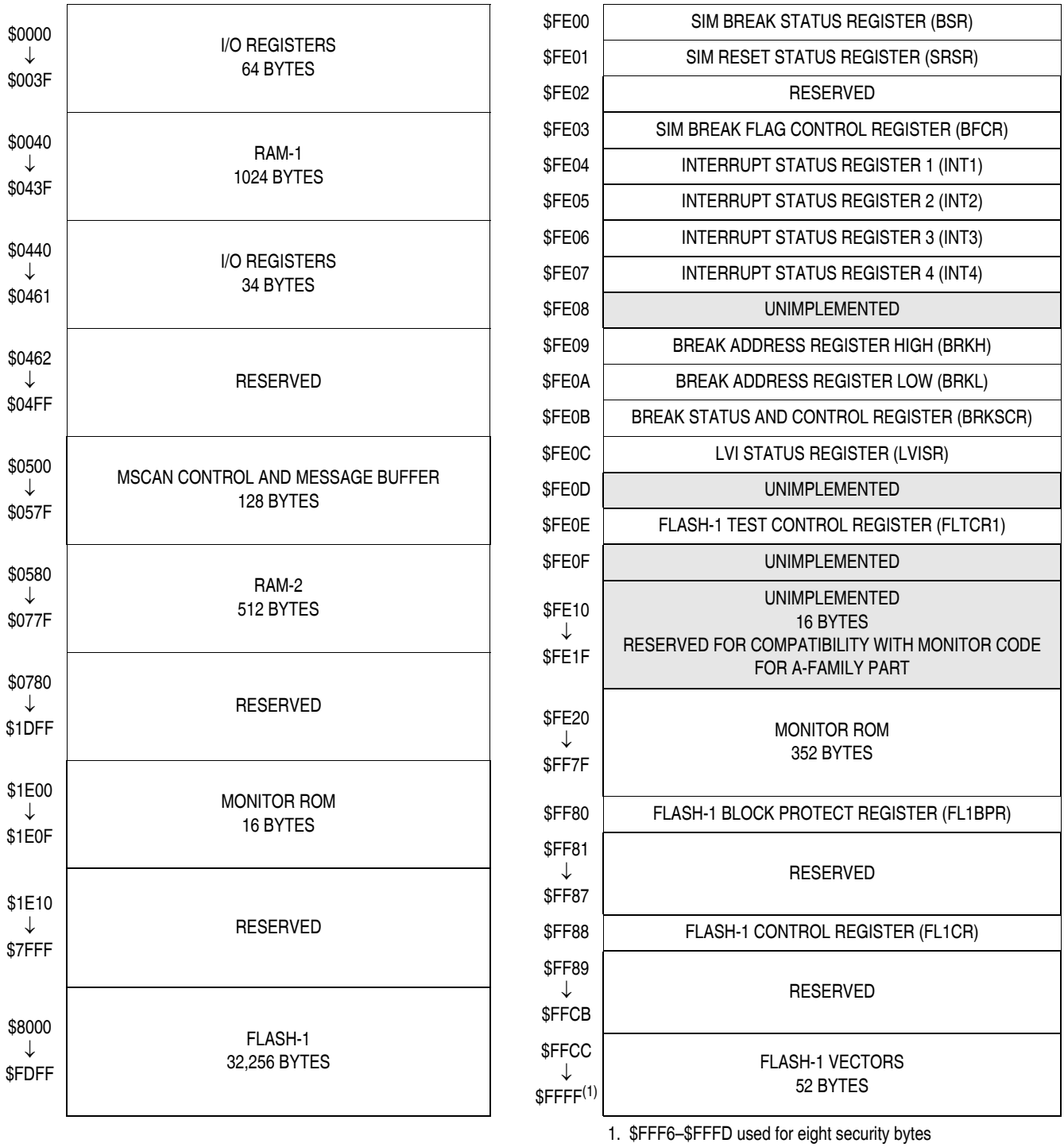
The MC68HC908GZ32 can address 32 Kbytes of memory space. The memory map, shown in Figure B-2, includes:

- 32 Kbytes of user FLASH memory
- 1536 bytes of random-access memory (RAM)
- 52 bytes of user-defined vectors




1. Pin contains integrated pullup device.
2. Ports are software configurable with pullup device if input port or pullup/pulldown device for keyboard input.
3. Higher current drive port pins

**Figure B-1. MC68HC908GZ32 Block Diagram**




1. $FFF6–$FFFD used for eight security bytes

**Figure B-2. MC68HC908GZ32 Memory Map**

## B.4 Ordering Information

**Table B-1. MC Order Numbers**

| MC Order Number | Operating Temperature Range | Package |
|---|---|---|
| MC908GZ32CFJ | –40°C to +85°C | 32-pin low-profile quad flat package (LQFP) |
| MC908GZ32VFJ | –40°C to +105°C | |
| MC908GZ32MFJ | –40°C to +125°C | |
| MC908GZ32CFA | –40°C to +85°C | 48-pin low-profile quad flat package (LQFP) |
| MC908GZ32VFA | –40°C to +105°C | |
| MC908GZ32MFA | –40°C to +125°C | |
| MC908GZ32CFU | –40°C to +85°C | 64-pin quad flat package (QFP) |
| MC908GZ32VFU | –40°C to +105°C | |
| MC908GZ32MFU | –40°C to +125°C | |

Temperature designators:
C = –40°C to +85°C
V = –40°C to +105°C
M = –40°C to +125°C

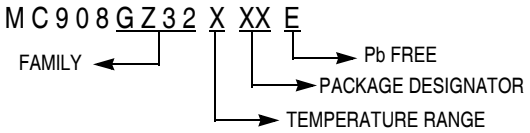



**Figure B-3. Device Numbering System**

## *How to Reach Us:*


**Home Page:**
www.freescale.com

**E-mail:**
support@freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
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 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com


RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics of 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.


Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2005, 2006, 2007. All rights reserved.



MC68HC908GZ60
Rev. 6, 04/2007


freescale™
semiconductor

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View MC68HC908GZ60CFA 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