# THE DATASHEET OF

# MC908EY16AMFJE

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

# Addendum to MC68HC908EY16A, rev. 2

This addendum introduces a change to this data sheet.

### Chapter 19 Development Support, Section 19.3.2 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.*

Changes to:

### Chapter 19 Development Support, Section 19.3.2 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. An improved security function denies monitor mode entry if five or more of the eight security bytes are $00 (zero bytes).*

This page intentionally blank.

# MC68HC908EY16A
# MC68HC908EY8A

Data Sheet

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

MC68HC908EY16A
Rev. 2
09/2010

freescale.com

freescale™
semiconductor

# MC68HC908EY16A
# MC68HC908EY8A

**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://www.freescale.com/

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

## Revision History

| Date | Revision Level | Description | Page Number(s) |
|---|---|---|---|
| April, 2006 | 0 | Initial release | N/A |
| September, 2006 | 1 | 21.2 Ordering Information — Separated automotive and consumer/industrial part numbers. | 279 |
| | | A.4 Ordering Information — Separated automotive and consumer/industrial part numbers. | 286 |

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

© Freescale Semiconductor, Inc., 2006, 2010. All rights reserved.

## Revision History (Continued)

| Date | Revision Level | Description | Page Number(s) |
|---|---|---|---|
| September 2010 | 2 | 20.3 Functional Operating Range — Changed maximum temperature specification from 135 °C to 125 °C. | 262 |
| | | 20.5 5V DC Electrical Characteristics — Changed maximum temperature specification from 135 °C to 125 °C. | 262 |
| | | 20.7 3V DC Electrical Characteristics — Changed maximum temperature specification from 135 °C to 125 °C. | 265 |
| | | 20.9 Internal Oscillator Characteristics — Changed maximum temperature specification from 135 °C to 125 °C. | 266 |
| | | 20.10 External Oscillator Characteristics — Changed maximum temperature specification from 135 °C to 125 °C. | 267 |
| | | 20.11 Trimmed Accuracy of the Internal Clock Generator — Changed maximum temperature specification from 135 °C to 125 °C. | 268 |
| | | 20.16 Memory Characteristics — Removed note referring to flash memory behavior between 125 °C to 135 °C. | 275 |
| | | 21.2 Ordering Information — Removed S908EY16AKFJE and MC908EY16AKFJE part numbers. | 279 |
| | | A.4 Ordering Information — Removed S908EY16AKFJE and MC908EY16AKFJE part numbers. | 286 |

# List of Chapters


Chapter 1
General Description . . . . . 19

Chapter 2
Memory . . . . . 29

Chapter 3
Analog-to-Digital Converter (ADC10) Module . . . . . 47

Chapter 4
BEMF Counter Module (BEMF) . . . . . 61

Chapter 5
Configuration Registers (CONFIG1, CONFIG2, CONFIG3) . . . . . 63

Chapter 6
Computer Operating Properly . . . . . 69

Chapter 7
Central Processor Unit (CPU) . . . . . 73

Chapter 8
Internal Clock Generator (ICG) Module . . . . . 85

Chapter 9
External Interrupt (IRQ) . . . . . 109

Chapter 10
Keyboard Interrupt (KBI) Module . . . . . 113

Chapter 11
Low-Voltage Inhibit (LVI) Module . . . . . 119

Chapter 12
Input/Output (I/O) Ports (PORTS) . . . . . 123

Chapter 13
Enhanced Serial Communications Interface (ESCI) Module . . . . . 133

Chapter 14
System Integration Module (SIM) . . . . . 163

Chapter 15
Serial Peripheral Interface (SPI) Module . . . . . 179

Chapter 16

Timebase Module (TBM) . . . . . 199
Chapter 17
Timer Interface A (TIMA) Module . . . . . 203
Chapter 18
Timer Interface B (TIMB) Module . . . . . 219
Chapter 19
Development Support . . . . . 235
Chapter 20
Electrical Specifications . . . . . 261
Chapter 21
Ordering Information and Mechanical Specifications . . . . . 279
Appendix A
MC68HC908EY8A . . . . . 283
Appendix B
Differences Between 908EY16A and 908EY16 . . . . . 287

# Table of Contents

## Chapter 1
## General Description


1.1 Introduction . . . 19
1.2 Features . . . 19
1.3 MCU Block Diagram . . . 20
1.4 Pin Assignments . . . 22
1.5 Pin Functions . . . 22
1.5.1 Power Supply Pins ($V_{DD}$ and $V_{SS}$) . . . 22
1.5.2 Oscillator Pins (PTC4/OSC1 and PTC3/OSC2) . . . 23
1.5.3 External Reset Pin ($\overline{RST}$) . . . 23
1.5.4 External Interrupt Pin ($\overline{IRQ}$) . . . 23
1.5.5 Analog Power Supply/Reference Pins ($V_{DDA}$, $V_{REFH}$, $V_{SSA}$, and $V_{REFL}$) . . . 23
1.5.6 Port A I/O Pins (PTA6/$\overline{SS}$, PTA5/SPSCK, PTA4/KBD4, PTA3/KBD3/RxD, PTA2/KBD2/TxD, PTA1/KBD1, and PTA0/KBD0) . . . 24
1.5.7 Port B I/O Pins (PTB7/AD7/TBCH1, PTB6/AD6/TBCH0, PTB5/AD5/SPSCK, PTB4/AD4/MOSI, PTB3/AD3/MISO, PTB2/AD2–PTB0/AD0) . . . 24
1.5.8 Port C I/O Pins (PTC4/OSC1, PTC3/OSC2, PTC2/MCLK/$\overline{SS}$, PTC1/MOSI, PTC0/MISO) . . . 24
1.5.9 Port D I/O Pins (PTD1/TACH1–PTD0/TACH0) . . . 24
1.5.10 Port E I/O Pins (PTE1/RxD–PTE0/TxD) . . . 24
1.6 Pin Summary . . . 25
1.7 Priority of Shared Pins . . . 27


## Chapter 2
## Memory


2.1 Introduction . . . 29
2.2 Unimplemented Memory Locations . . . 29
2.3 Reserved Memory Locations . . . 29
2.4 Input/Output (I/O) Section . . . 29
2.5 Random Access Memory (RAM) . . . 39
2.6 FLASH Memory (FLASH) . . . 39
2.6.1 FLASH Control Register . . . 40
2.6.2 FLASH Page Erase Operation . . . 41
2.6.3 FLASH Mass Erase Operation . . . 42
2.6.4 FLASH Program/Read Operation . . . 43
2.6.5 FLASH Block Protection . . . 45
2.6.6 FLASH Block Protect Register . . . 45
2.6.7 Wait Mode . . . 46
2.6.8 Stop Mode . . . 46


## Chapter 3

## Analog-to-Digital Converter (ADC10) Module


3.1 Introduction . . . 47
3.2 Features . . . 47
3.3 Functional Description . . . 47
3.3.1 Clock Select and Divide Circuit . . . 49
3.3.2 Input Select and Pin Control . . . 50
3.3.3 Conversion Control . . . 50
3.3.3.1 Initiating Conversions . . . 50
3.3.3.2 Completing Conversions . . . 50
3.3.3.3 Aborting Conversions . . . 50
3.3.3.4 Total Conversion Time . . . 51
3.3.4 Sources of Error . . . 52
3.3.4.1 Sampling Error . . . 52
3.3.4.2 Pin Leakage Error . . . 52
3.3.4.3 Noise-Induced Errors . . . 52
3.3.4.4 Code Width and Quantization Error . . . 53
3.3.4.5 Linearity Errors . . . 53
3.3.4.6 Code Jitter, Non-Monotonicity and Missing Codes . . . 53
3.4 Interrupts . . . 54
3.5 Low-Power Modes . . . 54
3.5.1 Wait Mode . . . 54
3.5.2 Stop Mode . . . 54
3.6 ADC10 During Break Interrupts . . . 54
3.7 I/O Signals . . . 55
3.7.1 ADC10 Analog Power Pin (VDDA) . . . 55
3.7.2 ADC10 Analog Ground Pin (VSSA) . . . 55
3.7.3 ADC10 Voltage Reference High Pin (VREFH) . . . 55
3.7.4 ADC10 Voltage Reference Low Pin (VREFL) . . . 55
3.7.5 ADC10 Channel Pins (ADn) . . . 56
3.8 Registers . . . 56
3.8.1 ADC10 Status and Control Register . . . 56
3.8.2 ADC10 Result High Register (ADRH) . . . 58
3.8.3 ADC10 Result Low Register (ADRL) . . . 58
3.8.4 ADC10 Clock Register (ADCLK) . . . 59


## Chapter 4
## BEMF Counter Module (BEMF)


4.1 Introduction . . . 61
4.2 Functional Description . . . 61
4.3 BEMF Register . . . 61
4.4 Input Signal . . . 61
4.5 Low Power Modes . . . 61
4.5.1 Wait Mode . . . 61
4.5.2 Stop Mode . . . 62


## Chapter 5
## Configuration Registers (CONFIG1, CONFIG2, CONFIG3)

5.1 Introduction . . . 63
5.2 Functional Description . . . 63

## Chapter 6
## Computer Operating Properly

6.1 Introduction . . . 69
6.2 Functional Description . . . 69
6.3 I/O Signals . . . 70
6.3.1 CGMXCLK . . . 70
6.3.2 STOP Instruction . . . 70
6.3.3 COPCTL Write . . . 70
6.3.4 Power-On Reset . . . 70
6.3.5 Internal Reset . . . 70
6.3.6 Reset Vector Fetch . . . 70
6.3.7 COPD . . . 70
6.3.8 COPRS . . . 71
6.4 COP Control Register . . . 71
6.5 Interrupts . . . 71
6.6 Monitor Mode . . . 71
6.7 Low-Power Modes . . . 71
6.7.1 Wait Mode . . . 71
6.7.2 Stop Mode . . . 71
6.8 COP Module During Break Interrupts . . . 71

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

7.1 Introduction . . . 73
7.2 Features . . . 73
7.3 CPU Registers . . . 73
7.3.1 Accumulator . . . 74
7.3.2 Index Register . . . 74
7.3.3 Stack Pointer . . . 75
7.3.4 Program Counter . . . 75
7.3.5 Condition Code Register . . . 76
7.4 Arithmetic/Logic Unit (ALU) . . . 77
7.5 Low-Power Modes . . . 77
7.5.1 Wait Mode . . . 77
7.5.2 Stop Mode . . . 77
7.6 CPU During Break Interrupts . . . 77
7.7 Instruction Set Summary . . . 78
7.8 Opcode Map . . . 83

## Chapter 8
## Internal Clock Generator (ICG) Module

8.1 Introduction . . . 85
8.2 Features . . . 85

8.3 Functional Description . . . 85
8.3.1 Clock Enable Circuit . . . 88
8.3.2 Internal Clock Generator . . . 88
8.3.2.1 Digitally Controlled Oscillator . . . 89
8.3.2.2 Modulo N Divider . . . 89
8.3.2.3 Frequency Comparator . . . 90
8.3.2.4 Digital Loop Filter . . . 90
8.3.3 External Clock Generator . . . 90
8.3.3.1 External Oscillator Amplifier . . . 91
8.3.3.2 External Clock Input Path . . . 92
8.3.4 Clock Monitor Circuit . . . 92
8.3.4.1 Clock Monitor Reference Generator . . . 93
8.3.4.2 Internal Clock Activity Detector . . . 93
8.3.4.3 External Clock Activity Detector . . . 93
8.3.5 Clock Selection Circuit . . . 95
8.3.5.1 Clock Selection Switches . . . 95
8.3.5.2 Clock Switching Circuit . . . 95
8.4 Usage Notes . . . 96
8.4.1 Switching Clock Sources . . . 96
8.4.2 Enabling the Clock Monitor . . . 96
8.4.3 Using Clock Monitor Interrupts . . . 97
8.4.4 Quantization Error in DCO Output . . . 97
8.4.4.1 Digitally Controlled Oscillator . . . 98
8.4.4.2 Binary Weighted Divider . . . 98
8.4.4.3 Variable-Delay Ring Oscillator . . . 98
8.4.4.4 Ring Oscillator Fine-Adjust Circuit . . . 99
8.4.5 Switching Internal Clock Frequencies . . . 99
8.4.6 Nominal Frequency Settling Time . . . 99
8.4.6.1 Settling to Within 15 Percent . . . 100
8.4.6.2 Settling to Within 5 Percent . . . 100
8.4.6.3 Total Settling Time . . . 100
8.4.7 Trimming Frequency on the Internal Clock Generator . . . 101
8.5 Low-Power Modes . . . 101
8.5.1 Wait Mode . . . 101
8.5.2 Stop Mode . . . 102
8.6 CONFIG Options . . . 102
8.6.1 External Clock Enable (EXTCLKEN) . . . 102
8.6.2 External Crystal Enable (EXTXTALEN) . . . 102
8.6.3 Slow External Clock (EXTSLOW) . . . 103
8.6.4 Oscillator Enable In Stop (OSCENINSTOP) . . . 103
8.7 Input/Output (I/O) Registers . . . 103
8.7.1 ICG Control Register . . . 104
8.7.2 ICG Multiplier Register . . . 106
8.7.3 ICG Trim Register . . . 106
8.7.4 ICG 5-Volt Trim Value . . . 107
8.7.5 ICG 3-Volt Trim Value . . . 107
8.7.6 ICG DCO Divider Register . . . 107
8.7.7 ICG DCO Stage Register . . . 108

## Chapter 9
## External Interrupt (IRQ)


9.1 Introduction . . . 109
9.2 Features . . . 109
9.3 Functional Description . . . 109
9.4 $\overline{\text{IRQ}}$ Pin . . . 110
9.5 IRQ Module During Break Interrupts . . . 112
9.6 IRQ Status and Control Register . . . 112


## Chapter 10
## Keyboard Interrupt (KBI) Module


10.1 Introduction . . . 113
10.2 Features . . . 113
10.3 Functional Description . . . 113
10.3.1 Keyboard Operation . . . 113
10.3.1.1 MODEK = 1 . . . 115
10.3.1.2 MODEK = 0 . . . 115
10.3.2 Keyboard Initialization . . . 116
10.4 Interrupts . . . 116
10.5 Low-Power Modes . . . 116
10.5.1 Wait Mode . . . 116
10.5.2 Stop Mode . . . 116
10.6 KBI During Break Interrupts . . . 116
10.7 I/O Signals . . . 117
10.7.1 KBI Input Pins (KBI7:KBI0) . . . 117
10.8 Registers . . . 117
10.8.1 Keyboard Status and Control Register (KBSCR) . . . 117
10.8.2 Keyboard Interrupt Enable Register (KBIER) . . . 118
10.8.3 Keyboard Interrupt Polarity Register (KBIPR) . . . 118


## Chapter 11
## Low-Voltage Inhibit (LVI) Module


11.1 Introduction . . . 119
11.2 Features . . . 119
11.3 Functional Description . . . 119
11.3.1 Polled LVI Operation . . . 120
11.3.2 Forced Reset Operation . . . 120
11.3.3 False Reset Protection . . . 120
11.3.4 LVI Status Register . . . 120
11.4 LVI Interrupts . . . 121
11.5 Low-Power Modes . . . 121
11.5.1 Wait Mode . . . 121
11.5.2 Stop Mode . . . 121


## Chapter 12
## Input/Output (I/O) Ports (PORTS)

12.1 Introduction . . . 123
12.2 Port A . . . 123
12.2.1 Port A Data Register . . . 123
12.2.2 Data Direction Register A . . . 123
12.3 Port B . . . 125
12.3.1 Port B Data Register . . . 125
12.3.2 Data Direction Register B . . . 125
12.4 Port C . . . 126
12.4.1 Port C Data Register . . . 126
12.4.2 Data Direction Register C . . . 127
12.5 Port D . . . 128
12.5.1 Port D Data Register . . . 128
12.5.2 Data Direction Register D . . . 128
12.6 Port E . . . 130
12.6.1 Port E Data Register . . . 130
12.6.2 Data Direction Register E . . . 130

## Chapter 13
## Enhanced Serial Communications Interface (ESCI) Module

13.1 Introduction . . . 133
13.2 Features . . . 133
13.3 Pin Name Conventions . . . 133
13.4 Functional Description . . . 135
13.4.1 Data Format . . . 136
13.4.2 Transmitter . . . 136
13.4.2.1 Character Length . . . 137
13.4.2.2 Character Transmission . . . 137
13.4.2.3 Break Characters . . . 137
13.4.2.4 Idle Characters . . . 138
13.4.2.5 Inversion of Transmitted Output . . . 138
13.4.2.6 Transmitter Interrupts . . . 138
13.4.3 Receiver . . . 138
13.4.3.1 Character Length . . . 138
13.4.3.2 Character Reception . . . 139
13.4.3.3 Data Sampling . . . 140
13.4.3.4 Framing Errors . . . 141
13.4.3.5 Baud Rate Tolerance . . . 141
13.4.3.6 Receiver Wakeup . . . 143
13.4.3.7 Receiver Interrupts . . . 144
13.4.3.8 Error Interrupts . . . 144
13.5 Low-Power Modes . . . 144
13.5.1 Wait Mode . . . 144
13.5.2 Stop Mode . . . 145
13.6 ESCI During Break Module Interrupts . . . 145
13.7 I/O Signals . . . 145
13.7.1 PTE0/TxD (Transmit Data) . . . 145
13.7.2 PTE1/RxD (Receive Data) . . . 145

13.8 I/O Registers . . . 146
13.8.1 ESCI Control Register 1 . . . 146
13.8.2 ESCI Control Register 2 . . . 148
13.8.3 ESCI Control Register 3 . . . 150
13.8.4 ESCI Status Register 1 . . . 151
13.8.5 ESCI Status Register 2 . . . 153
13.8.6 ESCI Data Register . . . 154
13.8.7 ESCI Baud Rate Register . . . 154
13.8.8 ESCI Prescaler Register . . . 156
13.9 ESCI Arbiter . . . 159
13.9.1 ESCI Arbiter Control Register . . . 159
13.9.2 ESCI Arbiter Data Register . . . 160
13.9.3 Bit Time Measurement . . . 161
13.9.4 Arbitration Mode . . . 162

## Chapter 14
## System Integration Module (SIM)

14.1 Introduction . . . 163
14.2 SIM Bus Clock Control and Generation . . . 163
14.2.1 Bus Timing . . . 165
14.2.2 Clock Startup from POR or LVI Reset . . . 165
14.2.3 Clocks in Stop Mode and Wait Mode . . . 165
14.3 Reset and System Initialization . . . 165
14.3.1 External Pin Reset . . . 165
14.3.2 Active Resets from Internal Sources . . . 166
14.3.2.1 Power-On Reset . . . 166
14.3.2.2 Computer Operating Properly (COP) Reset . . . 167
14.3.2.3 Illegal Opcode Reset . . . 167
14.3.2.4 Illegal Address Reset . . . 167
14.3.2.5 Forced Monitor Mode Entry Reset (MENRST) . . . 167
14.3.2.6 Low-Voltage Inhibit (LVI) Reset . . . 168
14.4 SIM Counter . . . 168
14.4.1 SIM Counter During Power-On Reset . . . 168
14.4.2 SIM Counter During Stop Mode Recovery . . . 168
14.4.3 SIM Counter and Reset States . . . 168
14.5 Program Exception Control . . . 168
14.5.1 Interrupts . . . 169
14.5.1.1 Hardware Interrupts . . . 170
14.5.1.2 SWI Instruction . . . 171
14.6 Interrupt Status Registers . . . 172
14.6.1 Interrupt Status Register 1 . . . 172
14.6.2 Interrupt Status Register 2 . . . 173
14.6.3 Interrupt Status Register 3 . . . 173
14.6.4 Reset . . . 173
14.6.5 Break Interrupts . . . 173
14.6.6 Status Flag Protection in Break Mode . . . 174
14.7 Low-Power Modes . . . 174

14.7.1 Wait Mode . . . 174
14.7.2 Stop Mode . . . 175
14.8 SIM Registers . . . 176
14.8.1 SIM Break Status Register . . . 176
14.8.2 SIM Reset Status Register . . . 177
14.8.3 SIM Break Flag Control Register . . . 178

## Chapter 15
## Serial Peripheral Interface (SPI) Module

15.1 Introduction . . . 179
15.2 Features . . . 179
15.3 Pin Name and Register Name Conventions . . . 179
15.4 Functional Description . . . 181
15.4.1 Master Mode . . . 182
15.4.2 Slave Mode . . . 182
15.5 Transmission Formats . . . 183
15.5.1 Clock Phase and Polarity Controls . . . 183
15.5.2 Transmission Format When CPHA = 0 . . . 183
15.5.3 Transmission Format When CPHA = 1 . . . 184
15.5.4 Transmission Initiation Latency . . . 185
15.6 Error Conditions . . . 185
15.6.1 Overflow Error . . . 186
15.6.2 Mode Fault Error . . . 187
15.7 Interrupts . . . 189
15.8 Queuing Transmission Data . . . 190
15.9 Resetting the SPI . . . 191
15.10 Low-Power Modes . . . 191
15.10.1 Wait Mode . . . 191
15.10.2 Stop Mode . . . 191
15.11 SPI During Break Interrupts . . . 191
15.12 SPI I/O Signals . . . 192
15.12.1 MISO (Master In/Slave Out) . . . 192
15.12.2 MOSI (Master Out/Slave In) . . . 192
15.12.3 SPSCK (Serial Clock) . . . 193
15.12.4 $\overline{SS}$ (Slave Select) . . . 193
15.12.5 $V_{SS}$ (Clock Ground) . . . 193
15.13 I/O Registers . . . 194
15.13.1 SPI Control Register . . . 194
15.13.2 SPI Status and Control Register . . . 195
15.13.3 SPI Data Register . . . 197

## Chapter 16
## Timebase Module (TBM)

16.1 Introduction . . . 199
16.2 Features . . . 199
16.3 Functional Description . . . 199

16.4 Interrupts . . . 199
16.5 TBM Interrupt Rate . . . 200
16.6 Low-Power Modes . . . 201
16.6.1 Wait Mode . . . 201
16.6.2 Stop Mode . . . 201
16.7 Timebase Control Register . . . 202

## Chapter 17
## Timer Interface A (TIMA) Module

17.1 Introduction . . . 203
17.2 Features . . . 203
17.3 Functional Description . . . 203
17.3.1 TIMA Counter Prescaler . . . 203
17.3.2 Input Capture . . . 205
17.3.3 Output Compare . . . 206
17.3.3.1 Unbuffered Output Compare . . . 206
17.3.3.2 Buffered Output Compare . . . 206
17.3.4 Pulse Width Modulation (PWM) . . . 207
17.3.4.1 Unbuffered PWM Signal Generation . . . 208
17.3.4.2 Buffered PWM Signal Generation . . . 208
17.3.4.3 PWM Initialization . . . 209
17.4 Interrupts . . . 209
17.5 Low-Power Modes . . . 210
17.5.1 Wait Mode . . . 210
17.5.2 Stop Mode . . . 210
17.6 TIMA During Break Interrupts . . . 210
17.7 I/O Signals . . . 210
17.7.1 TIMA Channel I/O Pins (PTD0/TACH0, PTD1/TACH1) . . . 210
17.8 I/O Registers . . . 211
17.8.1 TIMA Status and Control Register . . . 211
17.8.2 TIMA Counter Registers . . . 213
17.8.3 TIMA Counter Modulo Registers . . . 213
17.8.4 TIMA Channel Status and Control Registers . . . 214
17.8.5 TIMA Channel Registers . . . 217

## Chapter 18
## Timer Interface B (TIMB) Module

18.1 Introduction . . . 219
18.2 Features . . . 219
18.3 Functional Description . . . 219
18.3.1 TIMB Counter Prescaler . . . 219
18.3.2 Input Capture . . . 221
18.3.3 Output Compare . . . 222
18.3.3.1 Unbuffered Output Compare . . . 222
18.3.3.2 Buffered Output Compare . . . 223
18.3.4 Pulse Width Modulation (PWM) . . . 223
18.3.4.1 Unbuffered PWM Signal Generation . . . 224

18.3.4.2 Buffered PWM Signal Generation . . . . . 224
18.3.4.3 PWM Initialization . . . . . 225
18.4 Interrupts . . . . . 225
18.5 Low-Power Modes . . . . . 226
18.5.1 Wait Mode . . . . . 226
18.5.2 Stop Mode . . . . . 226
18.6 TIMB During Break Interrupts . . . . . 226
18.7 I/O Signals . . . . . 226
18.7.1 TIMB Channel I/O Pins (PTB7/AD7/TBCH1–PTB6/AD6/TBCH0) . . . . . 226
18.8 I/O Registers . . . . . 227
18.8.1 TIMB Status and Control Register . . . . . 227
18.8.2 TIMB Counter Registers . . . . . 229
18.8.3 TIMB Counter Modulo Registers . . . . . 229
18.8.4 TIMB Channel Status and Control Registers . . . . . 230
18.8.5 TIMB Channel Registers . . . . . 233

## Chapter 19
## Development Support

19.1 Introduction . . . . . 235
19.2 Break Module (BRK) . . . . . 235
19.2.1 Functional Description . . . . . 235
19.2.1.1 Flag Protection During Break Interrupts . . . . . 237
19.2.1.2 TIM During Break Interrupts . . . . . 237
19.2.1.3 COP During Break Interrupts . . . . . 237
19.2.2 Break Module Registers . . . . . 237
19.2.2.1 Break Status and Control Register . . . . . 238
19.2.2.2 Break Address Registers . . . . . 238
19.2.2.3 Break Status Register . . . . . 239
19.2.2.4 Break Flag Control Register . . . . . 239
19.2.3 Low-Power Modes . . . . . 239
19.3 Monitor Module (MON) . . . . . 240
19.3.1 Functional Description . . . . . 240
19.3.1.1 Normal Monitor Mode . . . . . 243
19.3.1.2 Forced Monitor Mode . . . . . 244
19.3.1.3 Monitor Vectors . . . . . 245
19.3.1.4 Data Format . . . . . 245
19.3.1.5 Break Signal . . . . . 245
19.3.1.6 Baud Rate . . . . . 246
19.3.1.7 Commands . . . . . 246
19.3.2 Security . . . . . 249
19.3.3 Extended Security . . . . . 250
19.4 Routines Supported in ROM . . . . . 250
19.4.1 Variables Used in the Routines . . . . . 251
19.4.2 How to Use the Routines . . . . . 251
19.4.2.1 GetByte . . . . . 253
19.4.2.2 PutByte . . . . . 254
19.4.2.3 Verify . . . . . 254

19.4.2.4 fProgram . . . . . 257
19.4.2.5 fErase . . . . . 259

## Chapter 20
## Electrical Specifications

20.1 Introduction . . . . . 261
20.2 Absolute Maximum Ratings . . . . . 261
20.3 Functional Operating Range . . . . . 262
20.4 Thermal Characteristics . . . . . 262
20.5 5V DC Electrical Characteristics . . . . . 262
20.6 5V Control Timing . . . . . 264
20.7 3V DC Electrical Characteristics . . . . . 265
20.8 3V Control Timing . . . . . 266
20.9 Internal Oscillator Characteristics . . . . . 266
20.10 External Oscillator Characteristics . . . . . 267
20.11 Trimmed Accuracy of the Internal Clock Generator . . . . . 268
20.11.1 Trimmed Internal Clock Generator Characteristics . . . . . 268
20.12 ADC10 Characteristics . . . . . 268
20.13 5V SPI Characteristics . . . . . 270
20.14 3V SPI Characteristics . . . . . 271
20.15 Timer Interface Module Characteristics . . . . . 274
20.16 Memory Characteristics . . . . . 275
20.17 EMC Performance . . . . . 276
20.17.1 Radiated Emissions . . . . . 276
20.17.2 Conducted Transient Susceptibility . . . . . 277

## Chapter 21
## Ordering Information and Mechanical Specifications

21.1 Introduction . . . . . 279
21.2 Ordering Information . . . . . 279
21.3 Package Dimensions . . . . . 279

## Appendix A
## MC68HC908EY8A

A.1 Introduction . . . . . 283
A.2 Block Diagram . . . . . 283
A.3 Memory . . . . . 283
A.4 Ordering Information . . . . . 286

## Appendix B
## Differences Between 908EY16A and 908EY16

B.1 Introduction . . . . . 287
B.2 Configuration . . . . . 287
B.2.1 Enhanced Serial Communications Interface Module (ESCI) . . . . . 287
B.2.2 Serial Peripheral Interface Module (SPI) . . . . . 288

B.2.3 Internal Clock Generator Module (ICG) . . . . . 288
B.2.4 Keyboard Interface Module (KBI) . . . . . 288
B.2.5 Analog-to-Digital Converter Module (ADC) . . . . . 288
B.3 Monitor Mode . . . . . 289
B.3.1 Monitor Extended Security . . . . . 289
B.3.2 Zeroes in Security Bytes . . . . . 289
B.3.3 Forced Monitor Mode Baud Rate . . . . . 289
B.4 Monitor ROM FLASH Programming Routines . . . . . 289
B.4.1 Erase . . . . . 289
B.4.2 Program . . . . . 290

# Chapter 1
# General Description

## 1.1 Introduction

The MC68HC908EY16A 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 MC68HC908EY8A with the exceptions noted in Appendix A MC68HC908EY8A.

## 1.2 Features

For convenience, features have been organized to reflect:

- Standard features of the MC68HC908EY16A
- Features of the CPU08

Standard features of the MC68HC908EY16A 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 at 5V
- Internal oscillator requiring no external components:
  - Software selectable bus frequencies
  - 25 percent accuracy with a trimming capability of better than 1 percent
  - Clock monitor
  - Option to allow use of external clock source or external crystal/ceramic resonator
- 15,872 bytes of on-chip FLASH memory with in-circuit programming
- FLASH program memory security[1]
- 512 bytes of on-chip random-access memory (RAM)
- Low voltage inhibit (LVI) module
- Internal clock generator module (ICG)
- Two 16-bit, 2-channel timer (TIMA and TIMB) interface modules with selectable input capture, output compare, and pulse-width modulation (PWM) capability on each channel
- 8-channel, 10-bit successive approximation analog-to-digital converter (ADC)
- Enhanced serial communications interface module (ESCI) for local interconnect network (LIN) connectivity
- Serial peripheral interface (SPI)

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

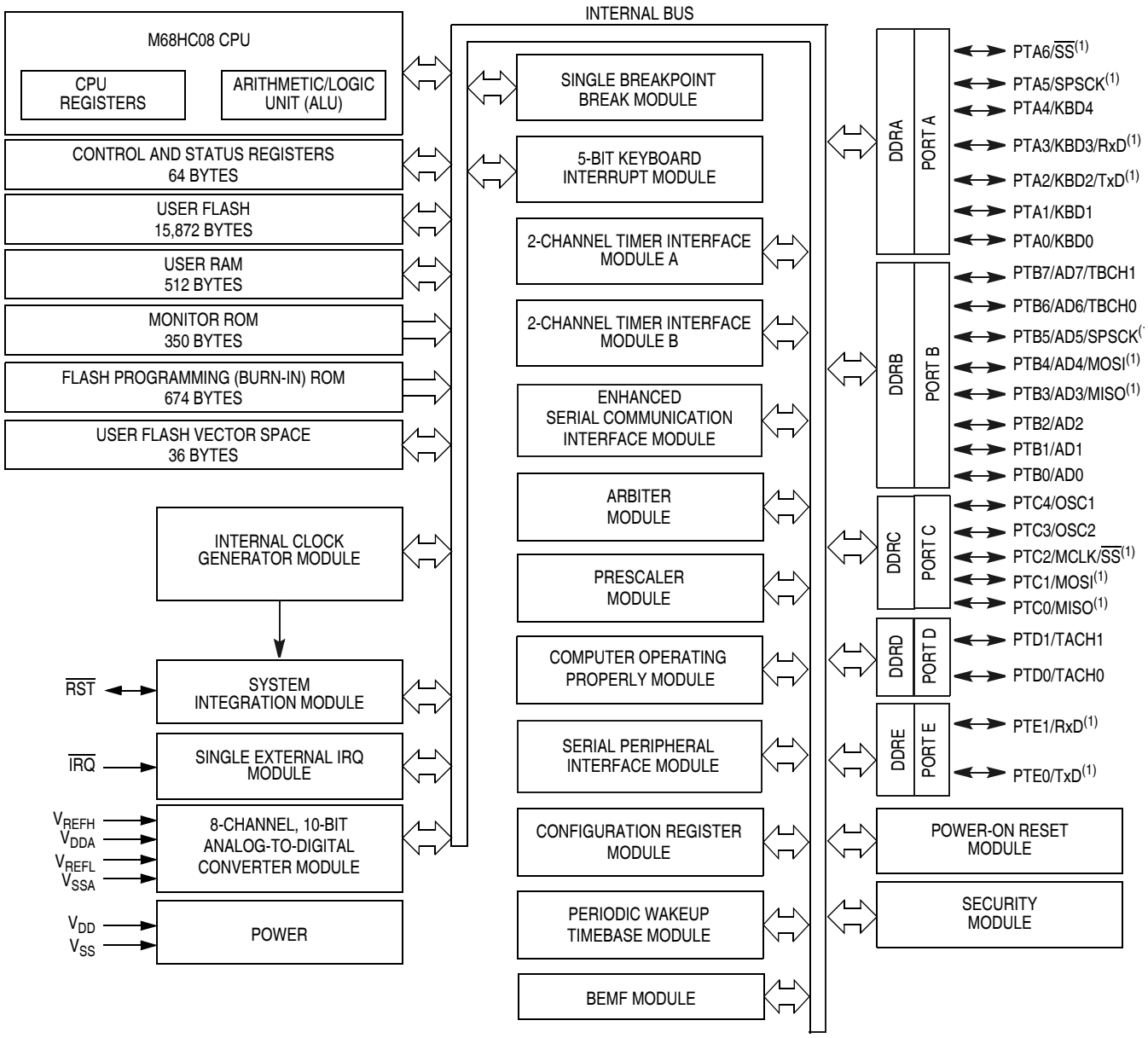
- Timebase Module (TBM)
- 5-bit keyboard wakeup port with software selectable rising or falling edge detect, as well as high- or low-level detection
  – Programmable for rising/falling edge or high/low level detection
- 24 general-purpose input/output (I/O) pins
- External asynchronous interrupt pin with internal pullup ($\overline{\text{IRQ}}$)
- System protection features:
  – Optional computer operating properly (COP) reset
  – Illegal opcode detection with reset
  – Illegal address detection with reset
- 32-pin quad flat pack (QFP) package
- Low-power design; fully static with stop and wait modes
- Internal pullups on $\overline{\text{IRQ}}$ and $\overline{\text{RST}}$ to reduce customer system cost
- Standard low-power modes of operation:
  – Wait mode
  – Stop mode
- Master reset pin ($\overline{\text{RST}}$) and power-on reset (POR)
- BREAK module (BRK) to allow single breakpoint setting during in-circuit debugging
- Higher current source capability on nine port lines for LED drive (PTA6/$\overline{\text{SS}}$, PTA5/SPSCK, PTA4/KBD4, PTA3/KBD3/RxD, PTA2/KBD2/TxD, PTA1/KBD1, PTA0/KBD0, PTC1/MOSI, and PTC0/MISO)

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
- Third party C language support

## 1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908EY16A.




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

**Figure 1-1. MCU Block Diagram**

## 1.4 Pin Assignments

Figure 1-2 shows the pin assignments for the MC68HC908EY16A.

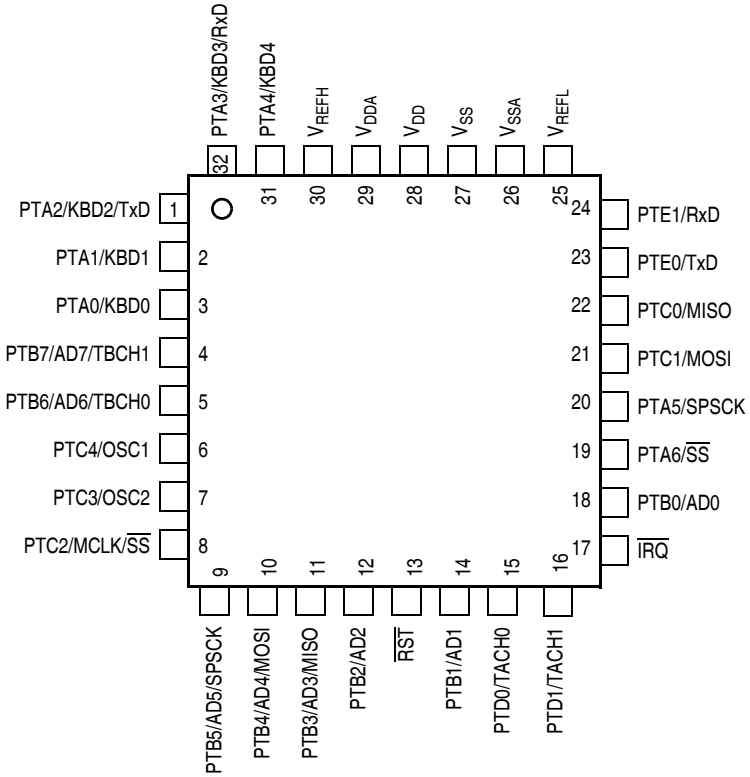



**Figure 1-2. Pin Assignments**

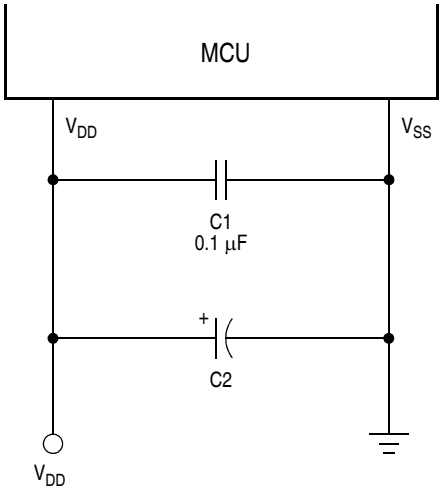
## 1.5 Pin Functions

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 1-3 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 1-3. Power Supply Bypassing**

### 1.5.2 Oscillator Pins (PTC4/OSC1 and PTC3/OSC2)

The OSC1 and OSC2 pins are available through programming options in the configuration register. These pins then become the connections to an external clock source or crystal/ceramic resonator.

When selecting PTC4 and PTC3 as I/O, OSC1 and OSC2 functions are not available.

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

A logic 0 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 that is always activated, even when the reset pin is pulled low. See Chapter 14 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 that is always activated, even when the $\overline{IRQ}$ pin is pulled low. See Chapter 9 External Interrupt (IRQ).

### 1.5.5 Analog Power Supply/Reference Pins ($V_{DDA}$, $V_{REFH}$, $V_{SSA}$, and $V_{REFL}$)

$V_{DDA}$ and $V_{SSA}$ are the power supply pins for the analog-to-digital converter (ADC). Decoupling of these pins should be as per the digital supply.

***NOTE***

*$V_{REFH}$ is the high reference supply for the ADC. $V_{DDA}$ should be tied to the same potential as $V_{DD}$ via separate traces.*

*$V_{REFL}$ is the low reference supply for the ADC. $V_{SSA}$ should be tied to the same potential as $V_{SS}$ via separate traces.*

See Chapter 3 Analog-to-Digital Converter (ADC10) Module.

### 1.5.6 Port A I/O Pins (PTA6/$\overline{SS}$, PTA5/SPSCK, PTA4/KBD4, PTA3/KBD3/RxD, PTA2/KBD2/TxD, PTA1/KBD1, and PTA0/KBD0)

Port A input/output (I/O) pins (PTA6/$\overline{SS}$, PTA5/SPSCK, PTA4/KBD4, PTA3/KBD3/RxD, PTA2/KBD2/TxD, PTA1/KBD1, and PTA0/KBD0) are special-function, bidirectional I/O port pins. PTA5 and PTA6 are shared with the serial peripheral interface (SPI). PTA4-PTA0 can be programmed to serve as keyboard interrupt pins. PTA2 and PTA3 can be programmed to serve as the ESCI transmit and receive data pins.

See Chapter 12 Input/Output (I/O) Ports (PORTS), Chapter 15 Serial Peripheral Interface (SPI) Module, Chapter 10 Keyboard Interrupt (KBI) Module, and Chapter 13 Enhanced Serial Communications Interface (ESCI) Module.

### 1.5.7 Port B I/O Pins (PTB7/AD7/TBCH1, PTB6/AD6/TBCH0, PTB5/AD5/SPSCK, PTB4/AD4/MOSI, PTB3/AD3/MISO, PTB2/AD2–PTB0/AD0)

PTB7/AD7/TBCH1, PTB6/AD6/TBCH0, PTB5/AD5/SPSCK, PTB4/AD4/MOSI, PTB3/AD3/MISO, PTB2/AD2–PTB0/AD0 are special-function, bidirectional I/O port pins that can also be used for ADC inputs. PTB7/AD7/TBCH1 and PTB6/AD6/TBCH0 are special function bidirectional I/O port pins that can also be used for timer interface pins. PTB5/AD5/SPSCK, PTB4/AD4/MOSI, and PTB3/AD3/MISO can be programmed to serve as SPI clock and data pins.

See Chapter 12 Input/Output (I/O) Ports (PORTS), Chapter 3 Analog-to-Digital Converter (ADC10) Module, Chapter 18 Timer Interface B (TIMB) Module, and Chapter 15 Serial Peripheral Interface (SPI) Module.

### 1.5.8 Port C I/O Pins (PTC4/OSC1, PTC3/OSC2, PTC2/MCLK/$\overline{SS}$, PTC1/MOSI, PTC0/MISO)

PTC4/OSC1, PTC3/OSC2, PTC2/MCLK/SS, PTC1/MOSI, PTC0/MISO are special-function, bidirectional I/O port pins. See Chapter 12 Input/Output (I/O) Ports (PORTS). PTC3/OSC2 and PTC4/OSC1 are shared with the on-chip oscillator circuit through configuration options. See Chapter 8 Internal Clock Generator (ICG) Module.

When applications require:

- PTC3/OSC2 can be programmed to be OSC2
- PTC4/OSC1 can be programmed to be OSC1

PTC2/MCLK/$\overline{SS}$ is software selectable to be MCLK, or bus clock out. PTC1/MOSI can be programmed to be the MOSI signal for the SPI. PTC0/MISO can be programmed to be the MISO signal for the SPI. See Chapter 15 Serial Peripheral Interface (SPI) Module.

### 1.5.9 Port D I/O Pins (PTD1/TACH1–PTD0/TACH0)

PTD1/TACH1–PTD0/TACH0 are special-function, bidirectional I/O port pins that can also be programmed to be timer pins.

See Chapter 12 Input/Output (I/O) Ports (PORTS) and Chapter 17 Timer Interface A (TIMA) Module.

### 1.5.10 Port E I/O Pins (PTE1/RxD–PTE0/TxD)

PTE1/RxD–PTE0/TxD are special-function, bidirectional I/O port pins that can also be programmed to be enhanced serial communication interface (ESCI) pins.

See Chapter 12 Input/Output (I/O) Ports (PORTS) and Chapter 13 Enhanced Serial Communications Interface (ESCI) Module.

***NOTE***

*Any unused inputs and I/O ports should be tied to an appropriate logic level (either $V_{DD}$ or $V_{SS}$). Although the I/O ports of the MC68HC908EY16A do not require termination, termination is recommended to reduce the possibility of electro-static discharge damage.*

## 1.6 Pin Summary

**Table 1-1. External Pin Summary**

| Pin Name | Function | Driver Type | Hysteresis[1] | Reset State |
|---|---|---|---|---|
| PTA6/$\overline{SS}$ | General-Purpose I/O<br>SPI Slave Select | Dual State | Yes | Input Hi-Z |
| PTA5/SPSCK | General-Purpose I/O<br>SPI Clock | Dual State | Yes | Input Hi-Z |
| PTA4/KBD4 | General-Purpose I/O<br>Keyboard Wakeup Pin | Dual State | Yes | Input Hi-Z |
| PTA3/KBD3/RxD | General-Purpose I/O<br>Keyboard Wakeup Pin<br>SCI Receive Data | Dual State | Yes | Input Hi-Z |
| PTA2/KBD2/TxD | General-Purpose I/O<br>Keyboard Wakeup Pin<br>SCI Transmit Data | Dual State | Yes | Input Hi-Z |
| PTA1/KBD1 | General-Purpose I/O<br>Keyboard Wakeup Pin | Dual State | Yes | Input Hi-Z |
| PTA0/KBD0 | General-Purpose I/O<br>Keyboard Wakeup Pin | Dual State | Yes | Input Hi-Z |
| PTB7/ATD7/TBCH1 | General-Purpose I/O<br>ADC Channel<br>Timer B Channel 1 | Dual State | Yes | Input Hi-Z |
| PTB6/ATD6/TBCH0 | General-Purpose I/O<br>ADC Channel<br>Timer B Channel 0 | Dual State | Yes | Input Hi-Z |
| PTB5/ATD5/SPSCK | General-Purpose I/O<br>ADC Channel<br>SPI Clock | Dual State | Yes | Input Hi-Z |
| PTB4/ATD4/MOSI | General-Purpose I/O<br>ADC Channel<br>SPI Data Path | Dual State | Yes | Input Hi-Z |
| PTB3/ATD3/MISO | General-Purpose I/O<br>ADC Channel<br>SPI Data Path | Dual State | Yes | Input Hi-Z |
| PTB2/ATD2 | General-Purpose I/O<br>ADC Channel | Dual State | Yes | Input Hi-Z |

## Table 1-1. External Pin Summary (Continued)

| Pin Name | Function | Driver Type | Hysteresis[1] | Reset State |
|---|---|---|---|---|
| PTB1/ATD1 | General-Purpose I/O<br>ADC Channel | Dual State | Yes | Input Hi-Z |
| PTB0/ATD0 | General-Purpose I/O<br>ADC Channel | Dual State | Yes | Input Hi-Z |
| PTC4/OSC1 | General-Purpose I/O<br>External Clock In | Dual State | Yes | Input Hi-Z |
| PTC3/OSC2 | General-Purpose I/O | Dual State | Yes | Input Hi-Z |
| PTC2/MCLK/$\overline{SS}$ | General-Purpose I/O<br>MCLK output<br>SPI Slave Select | Dual State | Yes | Input Hi-Z |
| PTC1/MOSI | General-Purpose I/O<br>SPI Data Path | Dual State | Yes | Input Hi-Z |
| PTC0/MISO | General-Purpose I/O<br>SPI Data Path | Dual State | Yes | Input Hi-Z |
| PTD1/TACH1 | General Purpose I/O<br>Timer A Channel 1 | Dual State | Yes | Input Hi-Z |
| PTD0/TACH0 | General Purpose I/O<br>Timer A Channel 0 | Dual State | Yes | Input Hi-Z |
| PTE1/RxD | General-Purpose I/O<br>SCI Receive Data | Dual State | Yes | Input Hi-Z |
| PTE0/TxD | General-Purpose I/O<br>SCI Transmit Data | Dual State | Yes | Input Hi-Z |
| $V_{DD}$ | Chip Power Supply | N/A | N/A | N/A |
| $V_{SS}$ | Chip Ground | N/A | N/A | N/A |
| $V_{DDA}$ | CGM Analog Power Supply | N/A | N/A | N/A |
| $V_{SSA}$ | CGM Analog Ground | N/A | N/A | N/A |
| $V_{REFH}$ | ADC Reference High Voltage | N/A | N/A | N/A |
| $V_{REFL}$ | ADC Reference Low Voltage | N/A | N/A | N/A |
| $\overline{IRQ}$ | External Interrupt Request | N/A | Yes | Input Hi-Z |
| $\overline{RST}$ | External Reset | Open Drain | Yes | Output Low |

1. Hysteresis is not 100% tested but is typically a minimum of 300mV.

## 1.7 Priority of Shared Pins

**Table 1-2. Priority of Shared Pins**

| Port Number | Priority 1 | Priority 2 | Priority 3 |
|---|---|---|---|
| PTA6 | SSB[1] (in) | PTA6 (in/out) | |
| PTA5 | SPSCK[1] (in/out) | PTA5 (in/out) | |
| PTA4 | KBD4 (in) | PTA4 (in/out) | |
| PTA3 | RX[1] (in) | KBD3 (in) | PTA3 (in/out) |
| PTA2 | TX[1] (out) | KBD2 (in) | PTA2 (in/out) |
| PTA1 | KBD1 (in) | PTA1 (in/out) | |
| PTA0 | KBD0 (in) | PTA0 (in/out) | |
| PTB7 | AD7 (in) | TBCH1 (in/out) | PTB7 (in/out) |
| PTB6 | AD6 (in) | TBCH0 (in/out) | PTB6 (in/out) |
| PTB5 | AD5 (in) | SPSCK[1] (in/out) | PTB5 (in/out) |
| PTB4 | AD4 (in) | MOSI[1] (in/out) | PTB4 (in/out) |
| PTB3 | AD3 (in) | MISO[1] (in/out) | PTB3 (in/out) |
| PTB2 | AD2 (in) | PTB2 (in/out) | |
| PTB1 | AD1 (in) | PTB1 (in/out) | |
| PTB0 | AD0 (in) | PTB0 (in/out) | |
| PTC4 | OSC1 (in) | PTC4 (in/out) | |
| PTC3 | OSC2 (out) | PTC3 (in/out) | |
| PTC2 | MCLK (out) | SSB[1] (in) | PTC2 (in/out) |
| PTC1 | MOSI[1] (in/out) | PTC1 (in/out) | |
| PTC0 | MISO[1] (in/out) | PTC0 (in/out) | |
| PTD1 | TACH1 (in/out) | PTD1 (in/out) | |
| PTD0 | TACH0 (in/out) | PTD0 (in/out) | |
| PTE1 | RX[1] (in) | PTE1 (in/out) | |
| PTE0 | TX[1] (out) | PTE0 (in/out) | |

1. Pin location can be changed using CONFIG3 register bits.

# Chapter 2
# Memory

## 2.1 Introduction

The M68HC08 central processor unit (CPU08) can address 64 Kbytes of memory space. The memory map, shown in Figure 2-1, includes:

- 16 Kbytes of FLASH memory, 15,872 bytes of user space
- 512 bytes of random-access memory (RAM)
- 36 bytes of user-defined vectors
- 350 bytes of monitor routines in read-only memory (ROM)
- 674 bytes of integrated FLASH burn-in routines in ROM

## 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 unit (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. Additional I/O registers have these addresses:

- $FE00; SIM break status register, SBSR
- $FE01; SIM reset status register, SRSR
- $FE03; SIM break flag control register, SBFCR
- $FE04; interrupt status register 1, INT1
- $FE05; interrupt status register 2, INT2
- $FE06; interrupt status register 3, INT3
- $FE08; FLASH control register, FLCR
- $FE09; break address register high, BRKH
- $FE0A; break address register low, BRKL
- $FE0B; break status and control register, BRKSCR
- $FE0C; LVI status register, LVISR
- $FF7E; FLASH block protect register, FLBPR
- $FF80; 5V internal oscillator trim value (optional), ICGT5V
- $FF81; 3V internal oscillator trim value (optional), ICGT3V

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

| | |
|---|---|
| $0000<br>↓<br>$003F | I/O Registers<br>64 Bytes |
| $0040<br>↓<br>$023F | RAM<br>512 Bytes |
| $0240<br>↓<br>$0FFF | Unimplemented<br>3520 Bytes |
| $1000<br>↓<br>$101F | Jump Table for FLASH Routines<br>32 Bytes |
| $1020<br>↓<br>$121F | Integrated FLASH Program<br>and Erase Routines<br>512 Bytes |
| $1220<br>↓<br>$137D | Unimplemented<br>350 Bytes |
| $137E<br>↓<br>$13FF | Integrated FLASH Program<br>and Erase Routines<br>130 Bytes |
| $1400<br>↓<br>$BFFF | Unimplemented<br>44,032 Bytes |
| $C000<br>↓<br>$FDFF | FLASH Memory<br>15,872 Bytes |
| $FDFF | Extended Security Byte |
| $FE00 | SIM Break Status Register (SBSR) |
| $FE01 | SIM Reset Status Register (SRSR) |
| $FE02 | Reserved |
| $FE03 | SIM Break Flag Control Register (SBFCR) |
| $FE04 | Interrupt Status Register 1 (INT1) |
| $FE05 | Interrupt Status Register 2 (INT2) |
| $FE06 | Interrupt Status Register 3 (INT3) |
| $FE07 | Reserved |
| $FE08 | FLASH Control Register (FLCR) |
| $FE09 | Break Address Register High (BRKH) |
| $FE0A | Break Address Register Low (BRKL) |
| $FE0B | Break Status and Control Register (BRKSCR) |
| $FE0C | LVI Status Register (LVISR) |
| $FE0D<br>↓<br>$FE1F | Reserved<br>19 Bytes |
| $FE20<br>↓<br>FF7D | Monitor ROM 350 Bytes |
| $FF7E | FLASH Block Protect Register (FLBPR) |
| $FF7F | Unimplemented |
| $FF80 | 5V ICG Trim Value (Optional) (ICGT5V) |
| $FF81 | 3V ICG Trim Value (Optional) (ICGT3V) |
| $FF82<br>↓<br>$FFDB | Unimplemented<br>90 Bytes |
| $FFDC<br>↓<br>$FFFF | FLASH Vectors<br>36 Bytes |

Note:
Locations $FFF6–$FFFD are used for the eight security bytes.

**Figure 2-1. Memory Map**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0000 | Port A Data Register (PTA) See page 123. | Read: | 0 | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0001 | Port B Data Register (PTB) See page 125. | Read: | PTB7 | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1 | PTB0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0002 | Port C Data Register (PTC) See page 126. | Read: | 0 | 0 | 0 | PTC4 | PTC3 | PTC2 | PTC1 | PTC0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0003 | Port D Data Register (PTD) See page 128. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | PTD1 | PTD0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0004 | Data Direction Register A (DDRA) See page 123. | Read: | 0 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0005 | Data Direction Register B (DDRB) See page 125. | Read: | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0006 | Data Direction Register C (DDRC) See page 127. | Read: | MCLKEN | 0 | 0 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0007 | Data Direction Register D (DDRD) See page 128. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | DDRD1 | DDRD0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0008 | Port E Data Register (PTE) See page 130. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | PTE1 | PTE0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0009 | Configuration Register 3 (CONFIG3) See page 67. | Read: | | RNGSEL | ESCISRE | SPISRE | MCLKSRE | PORTSRE | ESCISEL | SPISEL |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000A | Data Direction Register E (DDRE) See page 130. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | DDRE1 | DDRE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000B | BEMF Register (BEMF) See page 61. | Read: | BEMF7 | BEMF6 | BEMF5 | BEMF4 | BEMF3 | BEMF2 | BEMF1 | BEMF0 |
| | | 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 7)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $000C | KBI Polarity Register (KBIPR) See page 118. | Read: | 0 | 0 | 0 | KBIP4 | KBIP3 | KBIP2 | KBIP1 | KBIP0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $000D | SPI Control Register (SPCR) See page 194. | Read: | SPRIE | R | SPMSTR | CPOL | CPHA | SPWOM | SPE | SPTIE |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| $000E | SPI Status and Control Register (SPSCR) See page 195. | Read: | SPRF | ERRIE | OVRF | MODF | SPTE | MODFEN | SPR1 | SPR0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $000F | SPI Data Register (SPDR) See page 197. | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0010 | ESCI Control Register 1 (SCC1) See page 146. | Read: | LOOPS | ENSCI | TXINV | M | WAKE | ILTY | PEN | PTY |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0011 | ESCI Control Register 2 (SCC2) See page 148. | Read: | SCTIE | TCIE | SCRIE | ILIE | TE | RE | RWU | SBK |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0012 | ESCI Control Register 3 (SCC3) See page 150. | Read: | R8 | T8 | R | R | ORIE | NEIE | FEIE | PEIE |
| | | Write: | | | | | | | | |
| | | Reset: | U | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0013 | ESCI Status Register 1 (SCS1) See page 151. | Read: | SCTE | TC | SCRF | IDLE | OR | NF | FE | PE |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0014 | ESCI Status Register 2 (SCS2) See page 153. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | BKF | RPF |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0015 | ESCI Data Register (SCDR) See page 154. | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $0016 | ESCI Baud Rate Register (SCBR) See page 154. | Read: | LINT | LINR | SCP1 | SCP0 | R | SCR2 | SCR1 | SCR0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0017 | ESCI Prescale Register (SCPSC) See page 156. | Read: | PDS2 | PDS1 | PDS0 | PSSB4 | PSSB3 | PSSB2 | PSSB1 | PSSB0 |
| | | 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 7)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0018 | ESCII Arbiter Control Register (SCIACTL) See page 159. | Read: | AM1 | ALOST | AM0 | ACLK | AFIN | ARUN | AROVFL | ARD8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0019 | ESCI Arbiter Data Register (SCIACTL) See page 160. | Read: | ARD7 | ARD6 | ARD5 | ARD4 | ARD3 | ARD2 | ARD1 | ARD0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $001A | Keyboard Status and Control Register (KBSCR) See page 117. | 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 (KBIER) See page 118. | Read: | 0 | 0 | 0 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $001C | Timebase Control Register (TBCR) See page 202. | 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 112. | 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) See page 65. | Read: | R | ESCI BDSRC | EXT-XTALEN | EXT-SLOW | EXT-CLKEN | TMB-CLKSEL | OSCENIN-STOP | SSB-PUENB |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| $001F | Configuration Register 1 (CONFIG1) See page 64. | Read: | COPRS | LVISTOP | LVIRSTD | LVIPWRD | LVI5OR3(1) | SSREC | STOP | COPD |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0020 | Timer A Status and Control Register (TASC) See page 211. | Read: | TOF | TOIE | TSTOP | 0 | R | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $0021 | Timer A Counter Register High (TACNTH) See page 213. | 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 | Timer A Counter Register Low (TACNTL) See page 213. | 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 | Timer A Counter Modulo Register High (TAMODH) See page 213. | 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 |

1. The LVI5OR3 bit is cleared only by a power-on reset (POR).

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 7)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0024 | Timer A Counter Modulo Register Low (TAMODL) See page 213. | 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 |
| $0025 | Timer A Channel 0 Status and Control Register (TASC0) See page 214. | Read: | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0026 | Timer A Channel 0 Register High (TACH0H) See page 217. | Read: Write: | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0027 | Timer A Channel 0 Register Low (TACH0L) See page 217. | Read: Write: | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0028 | Timer A Channel 1 Status and Control Register (TASC1) See page 214. | Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| | | Write: | 0 | | R | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0029 | Timer A Channel 1 Register High (TACH1H) See page 217. | Read: Write: | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $002A | Timer A Channel 1 Register Low (TACH1L) See page 217. | Read: Write: | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $002B | Timer B Status and Control Register (TBSC) See page 227. | Read: | TOF | TOIE | TSTOP | 0 | R | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $002C | Timer B Counter Register High (TBCNTH) See page 229. | 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 |
| $002D | Timer B Counter Register Low (TBCNTL) See page 229. | 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 |
| $002E | Timer B Counter Modulo Register High (TBMODH) See page 229. | 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 |
| $002F | Timer B Counter Modulo Register Low (TBMODL) See page 229. | 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 |

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 7)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0030 | Timer B Channel 0 Status and Control Register (TBSC0) See page 230. | Read: | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| | | Write: | 0 | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0031 | Timer B Channel 0 Register High (TBCH0H) See page 233. | Read: | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0032 | Timer B Channel 0 Register Low (TBCH0L) See page 233. | Read: | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0033 | Timer B Channel 1 Status and Control Register (TBSC1) See page 230. | Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| | | Write: | 0 | | R | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0034 | Timer B Channel 1 Register High (TBCH1H) See page 233. | Read: | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0035 | Timer B Channel 1 Register Low (TBCH1L) See page 233. | Read: | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| | | Write: | | | | | | | | |
| | | Reset: | Indeterminate after reset | | | | | | | |
| $0036 | ICG Control Register (ICGCR) See page 104. | Read: | CMIE | CMF | CMON | CS | ICGON | ICGS | ECGON | ECGS |
| | | Write: | | 0 | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $0037 | ICG Multiplier Register (ICGMR) See page 106. | Read: | | N6 | N5 | N4 | N3 | N2 | N1 | N0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| $0038 | ICG Trim Register (ICGTR) See page 106. | Read: | TRIM7 | TRIM6 | TRIM5 | TRIM4 | TRIM3 | TRIM2 | TRIM1 | TRIM0 |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $0039 | ICG Divider Control Register (ICGDVR) See page 107. | Read: | | | | | DDIV3 | DDIV2 | DDIV1 | DDIV0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | U | U | U | U |
| $003A | ICG DCO Stage Control Register (ICGDSR) See page 108. | Read: | DSTG7 | DSTG6 | DSTG5 | DSTG4 | DDSTG3 | DSTG2 | DSTG1 | DSTG0 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | U | U | U | U | U | U | U | U |
| $003B | Reserved | | R | R | R | R | R | R | R | R |
| $003C | ADC10 Status and Control Register (ADSCR) See page 56. | Read: | COCO | AIEN | ADCO | ADCH4 | ADCH3 | ADCH2 | ADCH1 | ADCH0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 7)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $003D | ADC10 Data Register High (ADRH) See page 58. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | AD9 | AD8 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $003E | Analog-to-Digital Data Register Low (ADRL) See page 58. | Read: | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset | | | | | | | |
| $003F | Analog-to-Digital Clock Register (ADCLK) See page 59. | Read: | ADLPC | ADIV1 | ADIV0 | ADICLK | MODE1 | MODE0 | ADLSMP | ACLKEN |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $FE00 | SIM Break Status Register (SBSR) See page 176. | Read: | R | R | R | R | R | R | SBSW | R |
| | | Write: | | | | | | | NOTE | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Note: Writing a 0 clears SBSW. | | | | | | | | | |
| $FE01 | SIM Reset Status Register (SRSR) See page 177. | Read: | POR | PIN | COP | ILOP | ILAD | MENRST | LVI | 0 |
| | | Write: | | | | | | | | |
| | | POR: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE02 | Reserved | | | | | | | | | |
| $FE03 | SIM Break Flag Control Register (SBFCR) | | BCFE | R | R | R | R | R | R | R |
| $FE04 | Interrupt Status Register 1 (INT1) See See page 172. | | IF6 | IF5 | IF4 | IF3 | IF2 | IF1 | 0 | 0 |
| | | | R | R | R | R | R | R | R | R |
| | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE05 | Interrupt Status Register 2 (INT2) See See page 173. | | IF14 | IF13 | IF12 | IF11 | IF10 | IF9 | IF8 | IF7 |
| | | | R | R | R | R | R | R | R | R |
| | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE06 | Interrupt Status Register 3 (INT3) See See page 173. | | 0 | 0 | 0 | 0 | 0 | 0 | IF16 | IF15 |
| | | | R | R | R | R | R | R | R | R |
| | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE07 | Reserved | | | | | | | | | |
| $FE08 | FLASH Control Register (FLCR) See page 40. | 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 238. | 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 |

= Unimplemented R = Reserved U = Unaffected

**Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 7)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FE0A | Break Address Register Low (BRKL) See page 238. | Read:<br>Write: | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE0B | Break Status and Control Register (BSCR) See page 239. | 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 120. | Read: | LVIOUT | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FF7E | FLASH Block Protect Register (FLBPR)(1) See page 45. | Read:<br>Write: | BPR7 | BPR6 | BPR5 | BPR4 | BPR3 | BPR2 | BPR1 | BPR0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $FF80 | 5V Internal Oscillator Trim Value (Optional) (ICGT5V)(1) | Read:<br>Write: | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| $FF81 | 3V Internal Oscillator Trim Value (Optional) (ICGT3V)(1) | Read:<br>Write: | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| | | Reset: | Unaffected by reset | | | | | | | |

1. Non-volatile FLASH register

| | | | |
|---|---|---|---|
| $FFFF | COP Control Register (COPCTL) See page 71. | Read: | Low byte of reset vector |
| | | Write: | Writing clears COP counter (any value) |
| | | Reset: | Unaffected by reset |

= Unimplemented R = Reserved U = Unaffected

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

## Table 2-1. Vector Addresses

| Vector Priority | Vector | Address | Vector |
|---|---|---|---|
| Lowest | IF16 | $FFDC | Timebase interrupt vector (high) |
| | | $FFDD | Timebase interrupt vector (low) |
| | IF15 | $FFDE | SPI transmit vector (high) |
| | | $FFDF | SPI transmit vector (low) |
| | IF14 | $FFE0 | SPI receive vector (high) |
| | | $FFE1 | SPI receive vector (low) |
| | IF13 | $FFE2 | ADC conversion complete vector (high) |
| | | $FFE3 | ADC conversion complete vector (low) |
| | IF12 | $FFE4 | Keyboard vector (high) |
| | | $FFE5 | Keyboard vector (low) |
| | IF11 | $FFE6 | ESCI transmit vector (high) |
| | | $FFE7 | ESCI transmit vector (low) |
| | IF10 | $FFE8 | ESCI receive vector (high) |
| | | $FFE9 | ESCI receive vector (low) |
| | IF9 | $FFEA | ESCI error vector (high) |
| | | $FFEB | ESCI error vector (low) |
| | IF8 | $FFEC | TIMB overflow vector (high) |
| | | $FFED | TIMB overflow vector (low) |
| | IF7 | $FFEE | TIMB channel 1 vector (high) |
| | | $FFEF | TIMB channel 1 vector (low) |
| | IF6 | $FFF0 | TIMB channel 0 vector (high) |
| | | $FFF1 | TIMB channel 0 vector (low) |
| | IF5 | $FFF2 | TIMA overflow vector (high) |
| | | $FFF3 | TIMA overflow vector (low) |
| | IF4 | $FFF4 | TIMA channel 1 vector (high) |
| | | $FFF5 | TIMA channel 1 vector (low) |
| | IF3 | $FFF6 | TIMA channel 0 vector (high) |
| | | $FFF7 | TIMA channel 0 vector (low) |
| | IF2 | $FFF8 | CMIREQ (high) |
| | | $FFF9 | CMIREQ (low) |
| | IF1 | $FFFA | IRQ vector (high) |
| | | $FFFB | IRQ vector (low) |
| | — | $FFFC | SWI vector (high) |
| | | $FFFD | SWI vector (low) |
| | — | $FFFE | Reset vector (high) |
| Highest | | $FFFF | Reset vector (low) |

## 2.5 Random Access Memory (RAM)

Addresses $0040–$00FF and $0100–$023F are RAM locations. The location of the stack RAM is programmable with the reset stack pointer instruction (RSP). The 16-bit stack pointer allows the stack RAM to be anywhere in the 64K-byte 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 input/output (I/O) control and user data or code. When the stack pointer is moved from its reset location at $00FF, direct addressing mode instructions can access all page zero RAM locations efficiently. 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 central processor unit (CPU) registers.

**NOTE**

*For M6805, M146805, and M68HC05 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 could overwrite data in the RAM during a subroutine or during the interrupt stacking operation.*

## 2.6 FLASH Memory (FLASH)

The FLASH memory is an array of 15,872 bytes with an additional 36 bytes of user vectors and one byte used for block protection.

**NOTE**

*An erased bit reads as 1 and a programmed bit reads as 0.*

The program and erase operations are facilitated through control bits in the FLASH control register (FLCR). See 2.6.1 FLASH Control Register.

The FLASH is organized internally as an 16,384-word by 8-bit complementary metal-oxide semiconductor (CMOS) page erase, byte (8-bit) program embedded FLASH memory. Each page consists of 64 bytes. The page erase operation erases all words within a page. A page is composed of two adjacent rows.

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

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

### 2.6.1 FLASH Control Register

The FLASH control register (FLCR) controls FLASH program and erase operations.

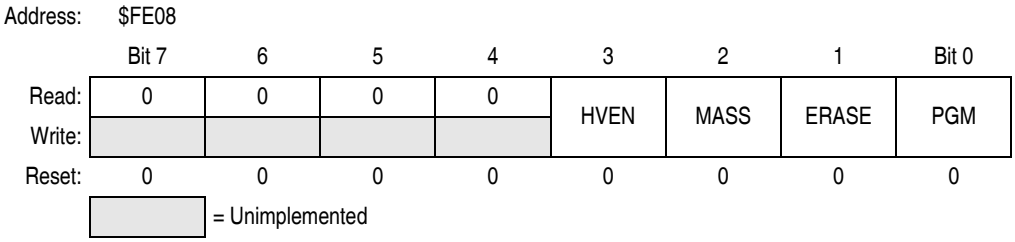



**Figure 2-3. FLASH Control Register (FLCR)**

**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 be set only 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 16-Kbyte FLASH 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 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

### 2.6.2 FLASH Page Erase Operation

Use this step-by-step procedure to erase a page (64 bytes) of FLASH memory to read as logic 1:

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

***NOTE***

*While these operations must be performed in the order shown, other unrelated operations may occur between the steps.*

***NOTE***

*Due to the security feature (see 19.3 Monitor Module (MON)) the last page of the FLASH (0xFFDC–0xFFFF), which contains the security bytes, cannot be erased by Page Erase Operation. It can only be erased with the Mass Erase Operation.*

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.

### 2.6.3 FLASH Mass Erase Operation

Use this step-by-step procedure to erase entire FLASH memory to read as logic 1:

1. Set both the ERASE bit and the MASS bit in the FLASH control register.
2. Read the FLASH block protect register.
3. Write any data to any FLASH address[1] within the FLASH memory address range.
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.

**NOTE**

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

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

**NOTE**

*Programming and erasing of FLASH locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order as shown, but other unrelated operations may occur between the steps.*

1. When in monitor mode, with security sequence failed (see 19.3.2 Security), write to the FLASH block protect register instead of any FLASH address.

### 2.6.4 FLASH Program/Read Operation

Programming of the FLASH memory is done on a row basis. A row consists of 32 consecutive bytes starting from addresses $XX00, $XX20, $XX40, $XX60, $XX80, $XXA0, $XXC0, and $XXE0. Use this step-by-step procedure to program a row of FLASH memory (Figure 2-4 is a flowchart representation).

**NOTE**

*To avoid program disturbs, the row must be erased before any byte on that row is programmed.*

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

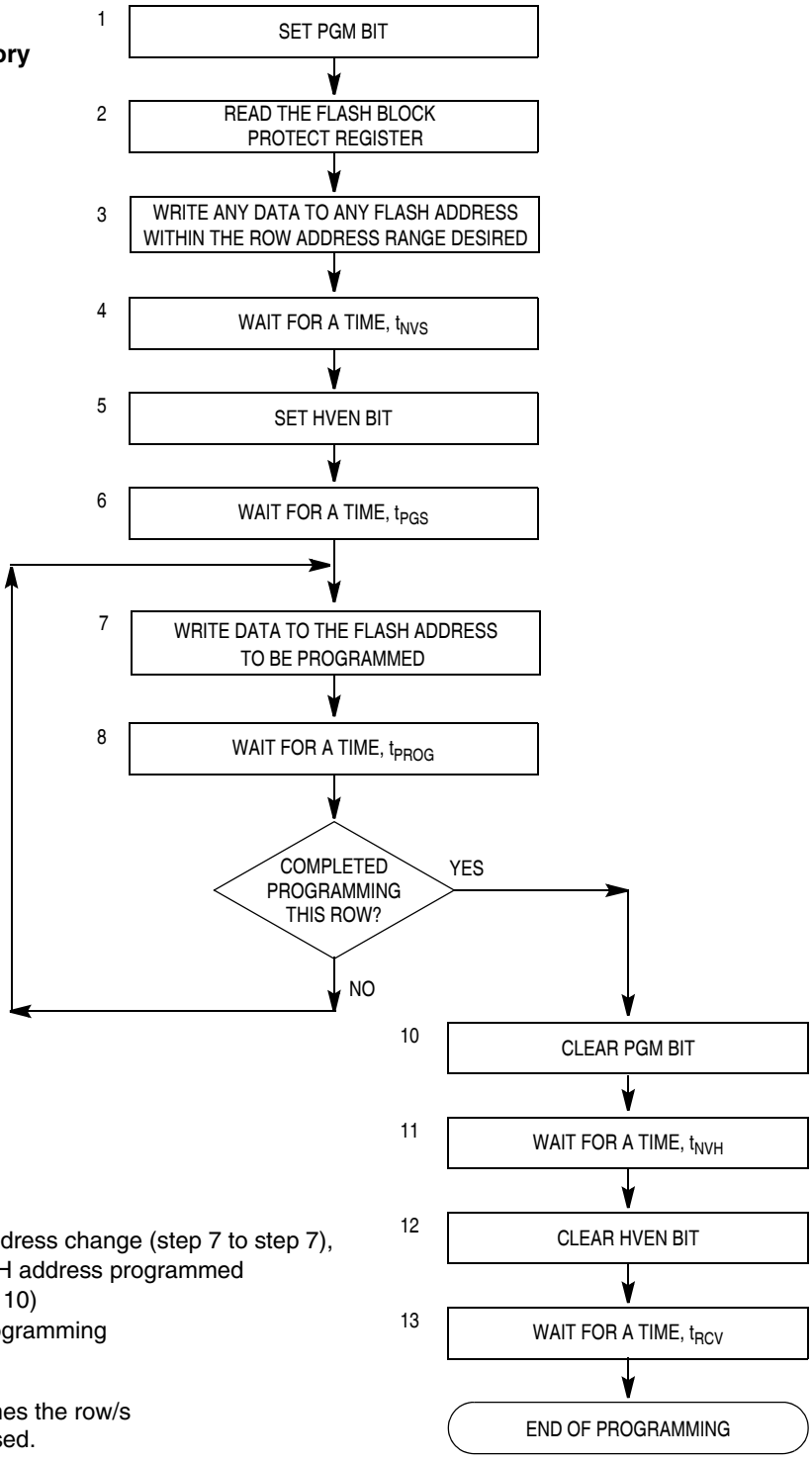
This program sequence is repeated throughout the memory until all data is programmed.

**NOTE**

*Programming and erasing of FLASH locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Do not exceed $t_{PROG}$ maximum.*

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

**Algorithm for Programming a Row (32 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 step 10) 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 2-4. FLASH Programming Flowchart**

### 2.6.5 FLASH 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 a block of memory from unintentional erase or program operations due to system malfunction. This protection is done by using the FLASH block protect register (FLBPR). The FLBPR determines the range of the FLASH memory which is to be protected. The range of the protected area starts from a location defined by FLBPR and ends at the bottom of the FLASH memory ($FFFF). When the memory is protected, the HVEN bit cannot be set in either erase or program operations.

***NOTE***

*In performing a program or erase operation, FLBPR must be read after setting the PGM or ERASE bit and before asserting the HVEN bit.*

When FLBPR is programmed with all 0s, the entire memory is protected from being programmed and erased. When all the bits are erased (all 1s), the entire memory is accessible for program and erase.

When bits within the FLBPR are programmed, they lock a block of memory address ranges as shown in 2.6.6 FLASH Block Protect Register. Once the FLBPR is programmed with a value other than $FF or $FE, any erase or program of the FLBPR or the protected block of FLASH memory is prohibited. Mass erase is disabled whenever any block is protected (FLBPR does not equal $FF). The presence of a $V_{TST}$ on the $\overline{IRQ}$ pin will bypass the block protection so that all of the memory included in the block protect register is open for program and erase operations.

### 2.6.6 FLASH Block Protect Register

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

Address: $FF7E

| | 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. Initial value from factory is $FF. | | | | | | | |

Write to this register is by a programming sequence to the FLASH memory.

**Figure 2-5. FLASH Block Protect Register (FLBPR)**

**BPR7–BPR0 — FLASH Block Protect Bits**

These eight bits represent bits [13:6] of a 16-bit memory address. Bit 15 and Bit 14 are 1s and bits [5:0] are 0s.

The resultant 16-bit address is used for specifying the start address of the FLASH memory for block protection. The FLASH is protected from this start address to the end of FLASH memory, at $FFFF. With this mechanism, the protect start address can be $XX00, $XX40, $XX80, and $XXC0 (64 bytes page boundaries) within the FLASH memory.

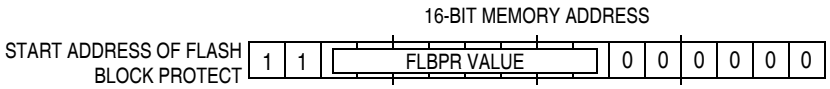



**Figure 2-6. FLASH Block Protect Start Address**

**Table 2-2. Examples of Protect Address Ranges**

| BPR[7:0] | Addresses of Protect Range |
|---|---|
| $00 | The entire FLASH memory is protected. |
| $01 (**0000 0001**) | $C040 (1**100 0000 01**00 0000) — $FFFF |
| $02 (**0000 0010**) | $C080 (1**100 0000 10**00 0000) — $FFFF |
| $03 (**0000 0011**) | $C0C0 (11**00 0000 11**00 0000) — $FFFF |
| $04 (**0000 0100**) | $C100 (11**00 0001 00**00 0000) — $FFFF |
| and so on... | |
| $FC (**1111 1100**) | $FF00 (11**11 1111 00**00 0000) — FFFF |
| $FD (**1111 1101**) | $FF40 (11**11 1111 01**00 0000) — $FFFF<br>FLBPR and vectors are protected |
| $FE (**1111 1110**) | $FF80 (1**111 1111 10**00 0000) — FFFF<br>Vectors are protected |
| $FF | The entire FLASH memory is not protected. |

### 2.6.7 Wait Mode

Putting the microcontroller unit (MCU) into wait mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly, but there will not be any memory activity since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH, or the operation will discontinue and the FLASH will be on standby mode.

### 2.6.8 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, but there will not be any memory activity since the CPU is inactive.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH, or the operation will discontinue and the FLASH will be on 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 at a minimum.*

# Chapter 3
# Analog-to-Digital Converter (ADC10) Module

## 3.1 Introduction

This section describes the 10-bit successive approximation analog-to-digital converter (ADC10).

The ADC10 module shares its pins with general-purpose input/output (I/O) port pins. See Figure 3-1 for port location of these shared pins. The ADC10 on this MCU uses $V_{DDA}$ and $V_{SSA}$ as its supply pins and $V_{REFH}$ and $V_{REFL}$ as its reference pins. This MCU uses CGMXCLK as its alternate clock source for the ADC. This MCU does not have a hardware conversion trigger.

## 3.2 Features

Features of the ADC10 module include:

- Linear successive approximation algorithm with 10-bit resolution
- Output formatted in 10- or 8-bit right-justified format
- Single or continuous conversion (automatic power-down in single conversion mode)
- Configurable sample time and conversion speed (to save power)
- Conversion complete flag and interrupt
- Input clock selectable from up to three sources
- Operation in wait and stop modes for lower noise operation

## 3.3 Functional Description

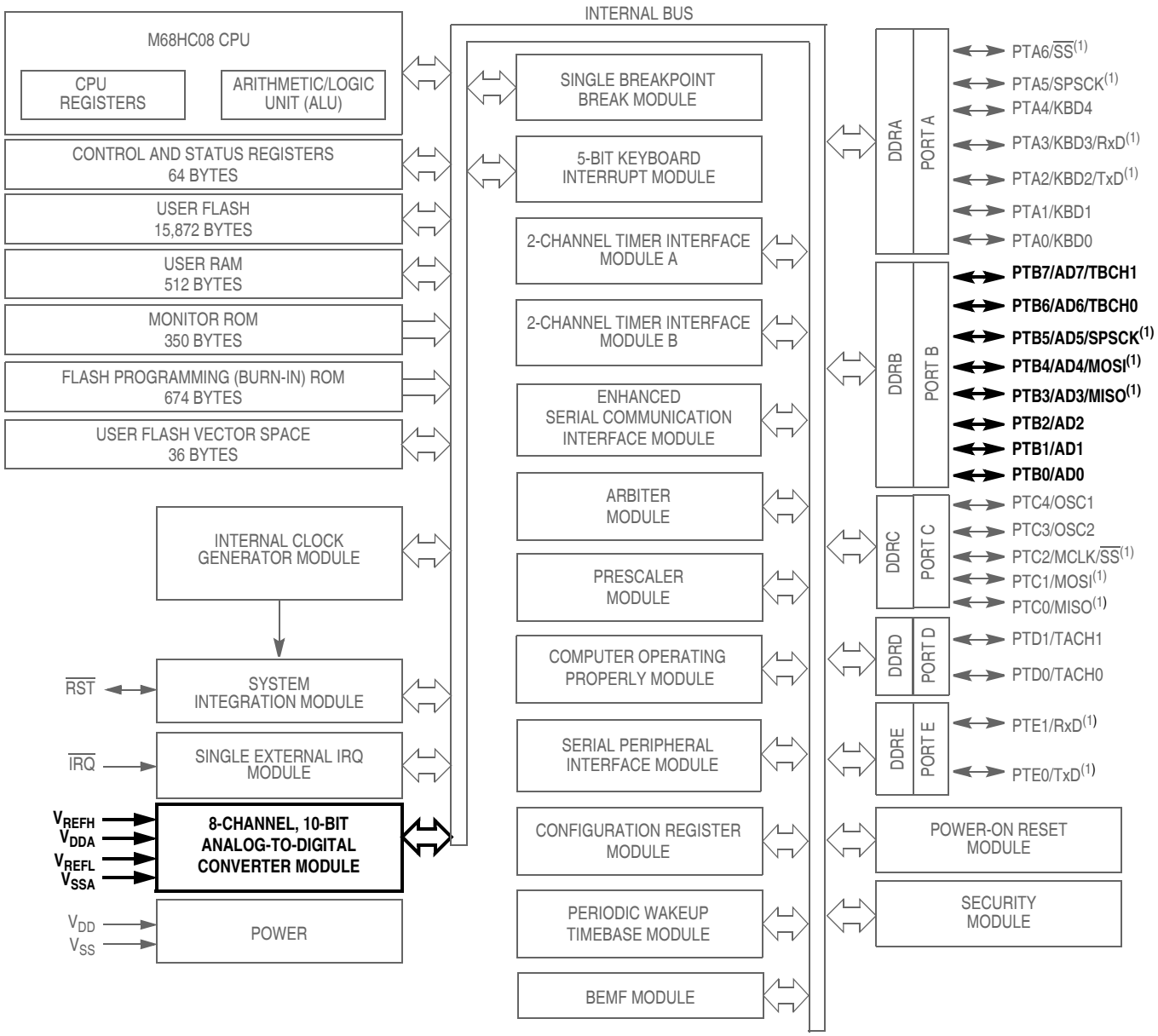
The ADC10 uses successive approximation to convert the input sample taken from ADVIN to a digital representation. The approximation is taken and then rounded to the nearest 10- or 8-bit value to provide greater accuracy and to provide a more robust mechanism for achieving the ideal code-transition voltage.

Figure 3-2 shows a block diagram of the ADC10.

For proper conversion, the voltage on ADVIN must fall between $V_{REFH}$ and $V_{REFL}$. If ADVIN is equal to or exceeds $V_{REFH}$, the converter circuit converts the signal to $3FF for a 10-bit representation or $FF for a 8-bit representation. If ADVIN is equal to or less than $V_{REFL}$, the converter circuit converts it to $000. Input voltages between $V_{REFH}$ and $V_{REFL}$ are straight-line linear conversions.

***NOTE***

*Input voltage must not exceed the analog supply voltages.*




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

**Figure 3-1. Block Diagram Highlighting ADC10 Block and Pins**

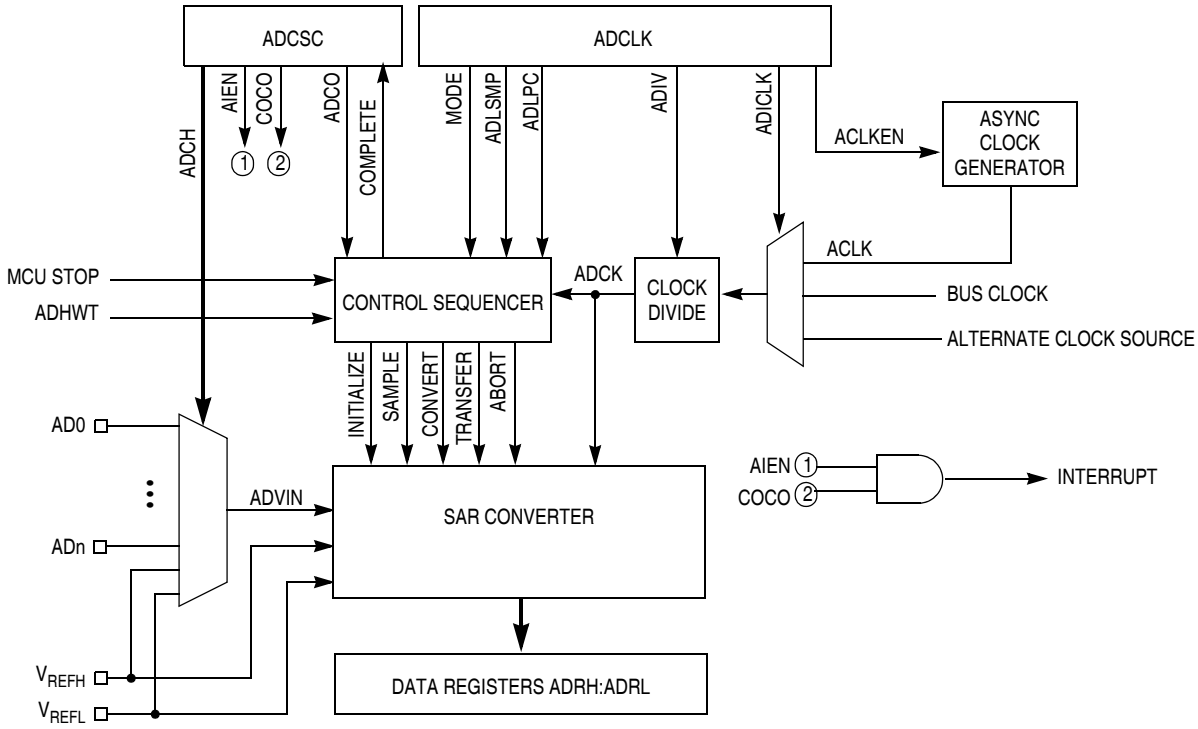



**Figure 3-2. ADC10 Block Diagram**

The ADC10 can perform an analog-to-digital conversion on one of the software selectable channels. The output of the input multiplexer (ADVIN) is converted by a successive approximation algorithm into a 10-bit digital result. When the conversion is completed, the result is placed in the data registers (ADRH and ADRL). In 8-bit mode, the result is rounded to 8 bits and placed in ADRL. The conversion complete flag is then set and an interrupt is generated if the interrupt has been enabled.

### 3.3.1 Clock Select and Divide Circuit

The clock select and divide circuit selects one of three clock sources and divides it by a configurable value to generate the input clock to the converter (ADCK). The clock can be selected from one of the following sources:

- The asynchronous clock source (ACLK) — This clock source is generated from a dedicated clock source which is enabled when the ADC10 is converting and the clock source is selected by setting the ACLKEN bit. When the ADLPC bit is clear, this clock operates from 1–2 MHz; when ADLPC is set it operates at 0.5–1 MHz. This clock is not disabled in STOP and allows conversions in stop mode for lower noise operation.
- Alternate clock source — This clock source is equal to the external oscillator clock or a four times the bus clock. The alternate clock source is MCU specific, see 3.1 Introduction to determine source and availability of this clock source option. This clock is selected when ADICLK and ACLKEN are both low.
- The bus clock — This clock source is equal to the bus frequency. This clock is selected when ADICLK is high and ACLKEN is low.

Whichever clock is selected, its frequency must fall within the acceptable frequency range for ADCK. If the available clocks are too slow, the ADC10 will not perform according to specifications. If the available clocks are too fast, then the clock must be divided to the appropriate frequency. This divider is specified by the ADIV[1:0] bits and can be divide-by 1, 2, 4, or 8.

### 3.3.2 Input Select and Pin Control

Only one analog input may be used for conversion at any given time. The channel select bits in ADCSC are used to select the input signal for conversion.

### 3.3.3 Conversion Control

Conversions can be performed in either 10-bit mode or 8-bit mode as determined by the MODE bits. Conversions can be initiated by either a software or hardware trigger. In addition, the ADC10 module can be configured for low power operation, long sample time, and continuous conversion.

#### *3.3.3.1 Initiating Conversions*

A conversion is initiated:

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

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

#### *3.3.3.2 Completing Conversions*

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

A blocking mechanism prevents a new result from overwriting previous data in ADRH and ADRL if the previous data is in the process of being read while in 10-bit mode (ADRH has been read but ADRL has not). In this case the data transfer is blocked, COCO is not set, and the new result is lost. When a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled). If single conversions are enabled, this could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

#### *3.3.3.3 Aborting Conversions*

Any conversion in progress will be aborted when:

- A write to ADCSC occurs (the current conversion will be aborted and a new conversion will be initiated, if ADCH are not all 1s).
- A write to ADCLK occurs.
- The MCU is reset.
- The MCU enters stop mode with ACLK not enabled.

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

Upon reset or when a conversion is otherwise aborted, the ADC10 module will enter a low power, inactive state. In this state, all internal clocks and references are disabled. This state is entered asynchronously and immediately upon aborting of a conversion.

#### *3.3.3.4 Total Conversion Time*

The total conversion time depends on many factors such as sample time, bus frequency, whether ACLKEN is set, and synchronization time. The total conversion time is summarized in Table 3-1.

**Table 3-1. Total Conversion Time versus Control Conditions**

| Conversion Mode | ACLKEN | Maximum Conversion Time |
|---|---|---|
| 8-Bit Mode (short sample — ADLSMP = 0): | | |
| Single or 1st continuous | 0 | 18 ADCK + 3 bus clock |
| Single or 1st continuous | 1 | 18 ADCK + 3 bus clock + 5 μs |
| Subsequent continuous ($f_{Bus} \geq f_{ADCK}$) | x | 16 ADCK |
| 8-Bit Mode (long sample — ADLSMP = 1): | | |
| Single or 1st continuous | 0 | 38 ADCK + 3 bus clock |
| Single or 1st continuous | 1 | 38 ADCK + 3 bus clock + 5 μs |
| Subsequent continuous ($f_{Bus} \geq f_{ADCK}$) | x | 36 ADCK |
| 10-Bit Mode (short sample — ADLSMP = 0): | | |
| Single or 1st continuous | 0 | 21 ADCK + 3 bus clock |
| Single or 1st continuous | 1 | 21 ADCK + 3 bus clock + 5 μs |
| Subsequent continuous ($f_{Bus} \geq f_{ADCK}$) | x | 19 ADCK |
| 10-Bit Mode (long sample — ADLSMP = 1): | | |
| Single or 1st continuous | 0 | 41 ADCK + 3 bus clock |
| Single or 1st continuous | 1 | 41 ADCK + 3 bus clock + 5 μs |
| Subsequent continuous ($f_{Bus} \geq f_{ADCK}$) | x | 39 ADCK |

The maximum total conversion time for a single conversion or the first conversion in continuous conversion mode is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK and ACLKEN bits, and the divide ratio is specified by the ADIV bits. For example, if the alternate clock source is 16 MHz and is selected as the input clock source, the input clock divide-by-8 ratio is selected and the bus frequency is 4 MHz, then the conversion time for a single 10-bit conversion is:

$$\text{Maximum Conversion time} = \frac{21 \text{ ADCK cycles}}{16 \text{ MHz}/8} + \frac{3 \text{ bus cycles}}{4 \text{ MHz}} = 11.25 \ \mu\text{s}$$

$$\text{Number of bus cycles} = 11.25 \ \mu\text{s} \times 4 \text{ MHz} = 45 \text{ cycles}$$

***NOTE***

*The ADCK frequency must be between $f_{ADCK}$ minimum and $f_{ADCK}$ maximum to meet A/D specifications.*

### 3.3.4 Sources of Error

Several sources of error exist for ADC conversions. These are discussed in the following sections.

#### *3.3.4.1 Sampling Error*

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 15 kΩ and input capacitance of approximately 10 pF, sampling to within 1/4LSB (at 10-bit resolution) can be achieved within the minimum sample window (3.5 cycles / 2 MHz maximum ADCK frequency) provided the resistance of the external analog source ($R_{AS}$) is kept below 10 kΩ. Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP (to increase the sample window to 23.5 cycles) or decreasing ADCK frequency to increase sample time.

#### *3.3.4.2 Pin Leakage Error*

Leakage on the I/O pins can cause conversion error if the external analog source resistance ($R_{AS}$) is high. If this error cannot be tolerated by the application, keep $R_{AS}$ lower than $V_{ADVIN}$ / (4096*$I_{Leak}$) for less than 1/4LSB leakage error (at 10-bit resolution).

#### *3.3.4.3 Noise-Induced Errors*

System noise which occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC10 accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1μF low-ESR capacitor from $V_{REFH}$ to $V_{REFL}$ (if available).
- There is a 0.1μF low-ESR capacitor from $V_{DDA}$ to $V_{SSA}$ (if available).
- If inductive isolation is used from the primary supply, an additional 1μF capacitor is placed from $V_{DDA}$ to $V_{SSA}$ (if available).
- $V_{SSA}$ and $V_{REFL}$ (if available) is connected to $V_{SS}$ at a quiet point in the ground plane.
- The MCU is placed in wait mode immediately after initiating the conversion (next instruction after write to ADCSC).
- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive $V_{DD}$ noise is coupled into the ADC10. In these cases, or when the MCU cannot be placed in wait or I/O activity cannot be halted, the following recommendations may reduce the effect of noise on the accuracy:

- Place a 0.01 μF capacitor on the selected input channel to $V_{REFL}$ or $V_{SSA}$ (if available). This will improve noise issues but will affect sample rate based on the external analog source resistance.
- Operate the ADC10 in stop mode by setting ACLKEN, selecting the channel in ADCSC, and executing a STOP instruction. This will reduce $V_{DD}$ noise but will increase effective conversion time due to stop recovery.
- Average the input by converting the output many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ACLKEN=1) and averaging. Noise that is synchronous to the ADCK cannot be averaged out.

### 3.3.4.4 Code Width and Quantization Error

The ADC10 quantizes the ideal straight-line transfer function into 1024 steps (in 10-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points from one code to the next. The ideal code width for an N bit converter (in this case N can be 8 or 10), defined as 1LSB, is:

$$1_{LSB} = (V_{REFH} - V_{REFL}) / 2^N$$

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

### 3.3.4.5 Linearity Errors

The ADC10 may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the user should be aware of them because they affect overall accuracy. These errors are:

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

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

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

- Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 LSB but will increase with noise.
- Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage.
- Missing codes are those which are never converted for any input value. In 8-bit or 10-bit mode, the ADC10 is guaranteed to be monotonic and to have no missing codes.

## 3.4 Interrupts

When AIEN is set, the ADC10 is capable of generating a CPU interrupt after each conversion. A CPU interrupt is generated when the conversion completes (indicated by COCO being set). COCO will set at the end of a conversion regardless of the state of AIEN.

## 3.5 Low-Power Modes

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

### 3.5.1 Wait Mode

The ADC10 will continue the conversion process and will generate an interrupt following a conversion if AIEN is set. If the ADC10 is not required to bring the MCU out of wait mode, ensure that the ADC10 is not in continuous conversion mode by clearing ADCO in the ADC10 status and control register before executing the WAIT instruction. In single conversion mode the ADC10 automatically enters a low-power state when the conversion is complete. It is not necessary to set the channel select bits (ADCH[4:0]) to all 1s to enter a low power state.

### 3.5.2 Stop Mode

If ACLKEN is clear, executing a STOP instruction will abort the current conversion and place the ADC10 in a low-power state. Upon return from stop mode, a write to ADCSC is required to resume conversions, and the result stored in ADRH and ADRL will represent the last completed conversion until the new conversion completes.

If ACLKEN is set, the ADC10 continues normal operation during stop mode. The ADC10 will continue the conversion process and will generate an interrupt following a conversion if AIEN is set. If the ADC10 is not required to bring the MCU out of stop mode, ensure that the ADC10 is not in continuous conversion mode by clearing ADCO in the ADC10 status and control register before executing the STOP instruction. In single conversion mode the ADC10 automatically enters a low-power state when the conversion is complete. It is not necessary to set the channel select bits (ADCH[4:0]) to all 1s to enter a low-power state.

If ACLKEN is set, a conversion can be initiated while in stop using the external hardware trigger ADEXTCO when in external convert mode. The ADC10 will operate in a low-power mode until the trigger is asserted, at which point it will perform a conversion and assert the interrupt when complete (if AIEN is set).

## 3.6 ADC10 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 break flag control register (BFCR) enables software to clear status bits during the break state. See BFCR in the SIM section of this data sheet.

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 cleared (its default state), software can read and write 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 cleared. After the break, doing the second step clears the status bit.

## 3.7 I/O Signals

The ADC10 module shares its pins with general-purpose input/output (I/O) port pins. See Figure 3-1 for port location of these shared pins. The ADC10 on this MCU uses $V_{DD}$ and $V_{SS}$ as its supply and reference pins. This MCU does not have an external trigger source.

### 3.7.1 ADC10 Analog Power Pin ($V_{DDA}$)

The ADC10 analog portion uses $V_{DDA}$ as its power pin. In some packages, $V_{DDA}$ is connected internally to $V_{DD}$. If externally available, connect the $V_{DDA}$ pin to the same voltage potential as $V_{DD}$. External filtering may be necessary to ensure clean $V_{DDA}$ for good results.

***NOTE***

*If externally available, route $V_{DDA}$ carefully for maximum noise immunity and place bypass capacitors as near as possible to the package.*

### 3.7.2 ADC10 Analog Ground Pin ($V_{SSA}$)

The ADC10 analog portion uses $V_{SSA}$ as its ground pin. In some packages, $V_{SSA}$ is connected internally to $V_{SS}$. If externally available, connect the $V_{SSA}$ pin to the same voltage potential as $V_{SS}$.

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

### 3.7.3 ADC10 Voltage Reference High Pin ($V_{REFH}$)

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

***NOTE***

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

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

### 3.7.4 ADC10 Voltage Reference Low Pin ($V_{REFL}$)

$V_{REFL}$ is the power supply for setting the low-reference voltage for the converter. In some packages, $V_{REFL}$ is connected internally to $V_{SSA}$. If externally available, connect the $V_{REFL}$ pin to the same voltage potential as $V_{SSA}$. There will be a brief current associated with $V_{REFL}$ when the sampling capacitor is

charging. If externally available, connect the $V_{REFL}$ pin to the same potential as $V_{SSA}$ at the single point ground location.

### 3.7.5 ADC10 Channel Pins (ADn)

The ADC10 has multiple input channels. Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. 0.01 μF capacitors with good high-frequency characteristics are sufficient. These capacitors are not necessary in all cases, but when used they must be placed as close as possible to the package pins and be referenced to $V_{SSA}$.

## 3.8 Registers

These registers control and monitor operation of the ADC10:

- ADC10 status and control register, ADCSC
- ADC10 data registers, ADRH and ADRL
- ADC10 clock register, ADCLK

### 3.8.1 ADC10 Status and Control Register

This section describes the function of the ADC10 status and control register (ADCSC). Writing ADCSC aborts the current conversion and initiates a new conversion (if the ADCH[4:0] bits are equal to a value other than all 1s).

Address: $003C

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

= Unimplemented

**Figure 3-3. ADC10 Status and Control Register (ADCSC)**

**COCO — Conversion Complete Bit**

COCO is a read-only bit which is set each time a conversion is completed. This bit is cleared whenever the status and control register is written or whenever the data register (low) is read.

1 = Conversion completed
0 = Conversion not completed

**AIEN — ADC10 Interrupt Enable Bit**

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

1 = ADC10 interrupt enabled
0 = ADC10 interrupt disabled

**ADCO — ADC10 Continuous Conversion Bit**

When this bit is set, the ADC10 will begin to convert samples continuously (continuous conversion mode) and update the result registers at the end of each conversion, provided the ADCH[4:0] bits do not decode to all 1s. The ADC10 will continue to convert until the MCU enters reset, the MCU enters stop mode (if ACLKEN is clear), ADCLK is written, or until ADCSC is written again. If stop is entered

(with ACLKEN low), continuous conversions will cease and can be restarted only with a write to ADCSC. Any write to ADCSC with ADCO set and the ADCH bits not all 1s will abort the current conversion and begin continuous conversions.

If the bus frequency is less than the ADCK frequency, precise sample time for continuous conversions cannot be guaranteed in short-sample mode (ADLSMP = 0). If the bus frequency is less than 1/11th of the ADCK frequency, precise sample time for continuous conversions cannot be guaranteed in long-sample mode (ADLSMP = 1).

When clear, the ADC10 will perform a single conversion (single conversion mode) each time ADCSC is written (assuming the ADCH[4:0] bits do not decode all 1s).

1 = Continuous conversion following a write to ADCSC
0 = One conversion following a write to ADCSC

**ADCH[4:0] — Channel Select Bits**

The ADCH[4:0] bits form a 5-bit field that is used to select one of the input channels. The input channels are detailed in Table 3-2. The successive approximation converter subsystem is turned off when the channel select bits are all set to 1. This feature allows explicit disabling of the ADC10 and isolation of the input channel from the I/O pad. Terminating continuous conversion mode this way will prevent an additional, single conversion from being performed. It is not necessary to set the channel select bits to all 1s to place the ADC10 in a low-power state, however, because the module is automatically placed in a low-power state when a conversion completes.

**Table 3-2. Input Channel Select**

| ADCH4 | ADCH3 | ADCH2 | ADCH1 | ADCH0 | Input Select[1] |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | PTB0 |
| 0 | 0 | 0 | 0 | 1 | PTB1 |
| 0 | 0 | 0 | 1 | 0 | PTB2 |
| 0 | 0 | 0 | 1 | 1 | PTB3 |
| 0 | 0 | 1 | 0 | 0 | PTB4 |
| 0 | 0 | 1 | 0 | 1 | PTB5 |
| 0 | 0 | 1 | 1 | 0 | PTB6 |
| 0 | 0 | 1 | 1 | 1 | PTB7 |
| 0 | 1 | 0 | 0 | 0 | Unused |
| Continuing through | | | | | Unused |
| 1 | 1 | 0 | 1 | 0 | Unused |
| 1 | 1 | 0 | 1 | 1 | Reserved |
| 1 | 1 | 1 | 0 | 0 | Reserved |
| 1 | 1 | 1 | 0 | 1 | $V_{REFH}$ |
| 1 | 1 | 1 | 1 | 0 | $V_{REFL}$ |
| 1 | 1 | 1 | 1 | 1 | Low-power state |

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

### 3.8.2 ADC10 Result High Register (ADRH)

This register holds the MSBs of the result and is updated each time a conversion completes. All other bits read as 0s. Reading ADRH prevents the ADC10 from transferring subsequent conversion results into the result registers until ADRL is read. If ADRL is not read until the after next conversion is completed, then the intermediate conversion result will be lost. In 8-bit mode, this register contains no interlocking with ADRL.

Address: $003D

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 3-4. ADC10 Data Register High (ADRH), 8-Bit Mode**

Address: $003D

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | 0 | 0 | AD9 | AD8 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 3-5. ADC10 Data Register High (ADRH), 10-Bit Mode**

### 3.8.3 ADC10 Result Low Register (ADRL)

This register holds the LSBs of the result. This register is updated each time a conversion completes. Reading ADRH prevents the ADC10 from transferring subsequent conversion results into the result registers until ADRL is read. If ADRL is not read until the after next conversion is completed, then the intermediate conversion result will be lost. In 8-bit mode, there is no interlocking with ADRH.

Address: $003E

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 3-6. ADC10 Data Register Low (ADRL)**

### 3.8.4 ADC10 Clock Register (ADCLK)

This register selects the clock frequency for the ADC10 and the modes of operation.

Address: $003F

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

**Figure 3-7. ADC10 Clock Register (ADCLK)**

**ADLPC — ADC10 Low-Power Configuration Bit**

ADLPC controls the speed and power configuration of the successive approximation converter. This is used to optimize power consumption when higher sample rates are not required.

1 = Low-power configuration: The power is reduced at the expense of maximum clock speed.
0 = High-speed configuration

**ADIV[1:0] — ADC10 Clock Divider Bits**

ADIV1 and ADIV0 select the divide ratio used by the ADC10 to generate the internal clock ADCK. Table 3-3 shows the available clock configurations.

**Table 3-3. ADC10 Clock Divide Ratio**

| ADIV1 | ADIV0 | Divide Ratio (ADIV) | Clock Rate |
|---|---|---|---|
| 0 | 0 | 1 | Input clock ÷ 1 |
| 0 | 1 | 2 | Input clock ÷ 2 |
| 1 | 0 | 4 | Input clock ÷ 4 |
| 1 | 1 | 8 | Input clock ÷ 8 |

**ADICLK — Input Clock Select Bit**

If ACLKEN is clear, ADICLK selects either the bus clock or an alternate clock source as the input clock source to generate the internal clock ADCK. If the alternate clock source is less than the minimum clock speed, use the internally-generated bus clock as the clock source. As long as the internal clock ADCK, which is equal to the selected input clock divided by ADIV, is at a frequency ($f_{ADCK}$) between the minimum and maximum clock speeds (considering ALPC), correct operation can be guaranteed.

1 = The internal bus clock is selected as the input clock source
0 = The alternate clock source IS SELECTED

**MODE[1:0] — 10- or 8-Bit Mode Selection**

These bits select 10- or 8-bit operation. The successive approximation converter generates a result that is rounded to 8- or 10-bit value based on the mode selection. This rounding process sets the transfer function to transition at the midpoint between the ideal code voltages, causing a quantization error of ± 1/2LSB.

Reset returns 8-bit mode.

00 = 8-bit, right-justified, ADCSC software triggered mode enabled
01 = 10-bit, right-justified, ADCSC software triggered mode enabled
10 = Reserved
11 = Reserved

**ADLSMP — Long Sample Time Configuration**

This bit configures the sample time of the ADC10 to either 3.5 or 23.5 ADCK clock cycles. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption in continuous conversion mode if high conversion rates are not required.

1 = Long sample time (23.5 cycles)

0 = Short sample time (3.5 cycles)

**ACLKEN — Asynchronous Clock Source Enable**

This bit enables the asynchronous clock source as the input clock to generate the internal clock ADCK, and allows operation in stop mode. The asynchronous clock source will operate between 1 MHz and 2 MHz if ADLPC is clear, and between 0.5 MHz and 1 MHz if ADLPC is set.

1 = The asynchronous clock is selected as the input clock source (the clock generator is only enabled during the conversion)

0 = ADICLK specifies the input clock source and conversions will not continue in stop mode

# Chapter 4
# BEMF Counter Module (BEMF)

## 4.1 Introduction

This section describes the BEMF module. The BEMF counter integrates over time, while the PTD0/TACH0 pin is active. This function is useful for measuring recirculation currents in motors occurring on switching of inductive loads.

BEMF is the abbreviation for **B**ack **E**lectro**M**agnetic **F**orce.

## 4.2 Functional Description

The 8-bit BEMF counter runs at the internal bus frequency divided by 64. Whenever PTD0/TACH0 is a logic 1, the counter increments by 1 with each period.

## 4.3 BEMF Register

The BEMF register contains the eight read-only bits of the BEMF counter, showing its actual value. A read access to the BEMF register resets all counter bits to 0.

Address: $000B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | BEMF7 | BEMF6 | BEMF5 | BEMF4 | BEMF3 | BEMF2 | BEMF1 | BEMF0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 4-1. BEMF Register (BEMF)**

## 4.4 Input Signal

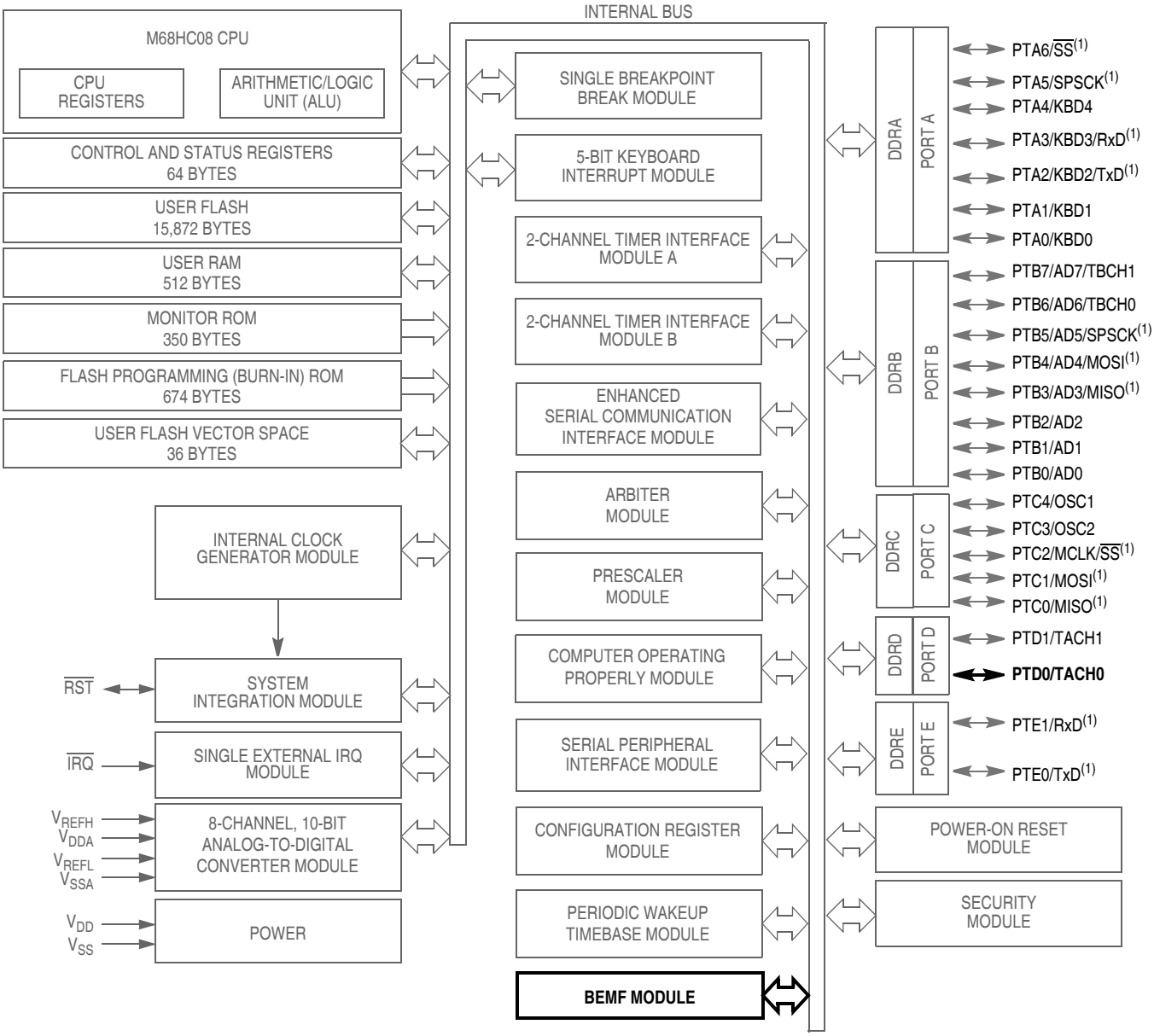
Port D shares the PTD0/TACH0 pin with the BEMF module. To measure an external signal with the BEMF module, PTD0/TACH0 must be configured as an input (DDRD0 = 0).

## 4.5 Low Power Modes

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

### 4.5.1 Wait Mode

The BEMF module remains active after execution of the WAIT instruction. In wait mode the BEMF register is not accessible by the CPU.




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

**Figure 4-2. Block Diagram Highlighting BEMF Block and Pins**

### 4.5.2 Stop Mode

The BEMF module is inactive after execution of the STOP instruction. In stop mode the BEMF register is not accessible by the CPU.

# Chapter 5
# Configuration Registers (CONFIG1, CONFIG2, CONFIG3)

## 5.1 Introduction

This section describes the configuration registers, CONFIG1, CONFIG2, and CONFIG3.

The configuration registers control these options:

- Stop mode recovery time, 32 CGMXCLK cycles or 4096 CGMXCLK cycles
- Computer operating properly (COP) timeout period, 262,128 or 8176 CGMXCLK cycles
- STOP instruction
- Computer operating properly (COP) module
- Low-voltage inhibit (LVI) module control and voltage trip point selection
- Enable/disable the oscillator (OSC) during stop mode
- External clock/crystal source control
- Enhanced SCI clock source selection
- SPI pin selection
- ESCI pin selection
- External oscillator frequency range selection
- Slew rate control for the ports, SPI, ESCI, and MCLK outputs

## 5.2 Functional Description

The configuration registers are used in the initialization of various options and 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 $0009, $001E, and $001F. For compatibility, a write to a read-only memory (ROM) version of the MCU at this location will have no effect. The configuration register may be read at anytime.

***NOTE***

*On a ROM device, the CONFIG module is known as an MOR (mask option register). On a ROM device, the options are fixed at the time of device fabrication and are neither writable nor changeable by the user.*

*On a FLASH device, the CONFIG registers 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, Figure 5-2, and Figure 5-3.*

| Address: | $001F | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read:<br>Write: | COPRS | LVISTOP | LVIRSTD | LVIPWRD | LVI5OR3[1] | SSREC | STOP | COPD |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. The LVI5OR3 bit is cleared only by a power-on reset (POR).

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

**COPRS — COP Rate Select Bit**

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

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) Module.

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) Module.

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) Module. The voltage mode selected for the LVI will typically be 5 V. However, users may choose to operate the LVI in 3-V mode if desired. See Chapter 20 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 CGMXCLCK cycles

***NOTE***

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

If the system clock source selected is the internal oscillator or the external crystal and the OSCENINSTOP configuration bit is not set, the oscillator will be disabled during stop mode. The short stop recovery does not provide enough time for oscillator stabilization and thus the SSREC bit should not be set.

The system stabilization time for power-on reset and long stop recovery (both 4096 CGMXCLK cycles) gives a delay longer than the LVI enable time for these startup scenarios. There is no period where the MCU is not protected from a low-power condition. However, when using the short stop recovery configuration option, the 32-CGMXCLK delay must be greater than the LVI's turn on time to avoid a period in startup where the LVI is not protecting the MCU.

**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.

1 = COP module disabled
0 = COP module enabled

Address: $001E

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | R | ESCIBDSRC | EXTXTALEN | EXTSLOW | EXTCLKEN | TMBCLKSEL | OSCENINSTOP | SSBPUENB |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

R = Reserved

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

**ESCIBDSRC — ESCI Baud Rate Clock Source Bit**

ESCIBDSRC controls the clock source used for the ESCI. The setting of the bit affects the frequency at which the ESCI operates.

1 = Internal data bus clock used as clock source for ESCI
0 = CGMXCLK used as clock source for ESCI

**EXTXTALEN — External Crystal Enable Bit**

EXTXTALEN enables the external oscillator circuits to be configured for a crystal configuration where the PTC4/OSC1 and PTC3/OSC2 pins are the connections for an external crystal.

***NOTE***

*This bit does not function without setting the EXTCLKEN bit also.*

Clearing the EXTXTALEN bit (default setting) allows the PTC3/OSC2 pin to function as a general-purpose I/O pin. Refer to Table 5-1 for configuration options for the external source. See Chapter 8 Internal Clock Generator (ICG) Module for a more detailed description of the external clock operation.

EXTXTALEN, when set, also configures the clock monitor to expect an external clock source in the valid range of crystals (30 kHz to 100 kHz or 1 MHz to 8 MHz). When EXTXTALEN is clear, the clock monitor will expect an external clock source in the valid range for externally generated clocks when using the clock monitor (60 Hz to 32 MHz).

EXTXTALEN, when set, also configures the external clock stabilization divider in the clock monitor for a 4096-cycle timeout to allow the proper stabilization time for a crystal. When EXTXTALEN is clear, the stabilization divider is configured to 16 cycles since an external clock source does not need a startup time.

1 = Allows PTC3/OSC2 to be an external crystal connection.
0 = PTC3/OSC2 functions as an I/O port pin (default).

**EXTSLOW — Slow External Crystal Enable Bit**

The EXTSLOW bit has two functions. It configures the ICG module for a fast (1 MHz to 32 MHz) or slow (30 kHz to 100 kHz) speed crystal. The option also configures the clock monitor operation in the ICG module to expect an external frequency higher (307.2 kHz to 32 MHz) or lower (60 Hz to 307.2 kHz) than the base frequency of the internal oscillator. See Chapter 8 Internal Clock Generator (ICG) Module.

1 = ICG set for slow external crystal operation
0 = ICG set for fast external crystal operation

**Table 5-1. External Clock Option Settings**

| External Clock Configuration Bits | | Pin Function | | Description |
|---|---|---|---|---|
| EXTCLKEN | EXTXTALEN | PTC4/OSC1 | PTC3/OSC2 | |
| 0 | 0 | PTC4 | PTC3 | Default setting — external oscillator disabled |
| 0 | 1 | PTC4 | PTC3 | External oscillator disabled since EXTCLKEN not set |
| 1 | 0 | OSC1 | PTC3 | External oscillator configured for an external clock source input (square wave) on OSC1 |
| 1 | 1 | OSC1 | OSC2 | External oscillator configured for an external crystal configuration on OSC1 and OSC2. System will also operate with square-wave clock source in OSC1. |

**EXTCLKEN — External Clock Enable Bit**

EXTCLKEN enables an external clock source or crystal/ceramic resonator to be used as a clock input. Setting this bit enables PTC4/OSC1 pin to be a clock input pin. Clearing this bit (default setting) allows the PTC4/OSC1 and PTC3/OSC2 pins to function as general-purpose input/output (I/O) pins. Refer to Table 5-1 for configuration options for the external source. See Chapter 8 Internal Clock Generator (ICG) Module for a more detailed description of the external clock operation.

1 = Allows PTC4/OSC1 to be an external clock connection
0 = PTC4/OSC1 and PTC3/OSC2 function as I/O port pins (default).

**TMBCLKSEL — Timebase Clock Select Bit**

TMBCLKSEL enables an enable the extra divide by 128 prescaler in the timebase module. Setting this bit enables the extra prescaler and clearing this bit disables it. Refer to Table 16-1 for timebase divider selection details.

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 internal clock generator module to continue to generate clocks (either internal, ICLK, or external, ECLK) in stop mode. See Chapter 8 Internal Clock Generator (ICG) Module. This function is used to keep the timebase running while the rest of the microcontroller stops. When clear, all clock generation will cease and both ICLK and ECLK will be forced low during stop mode. The default state for this option is clear, disabling the ICG in stop mode.

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

**SSBPUENB — $\overline{SS}$ Pullup Enable Bit**

Clearing SSBPUENB enables the $\overline{SS}$ pullup resistor.

1 = Disables $\overline{SS}$ pullup resistor.
0 = Enables $\overline{SS}$ pullup resistor.

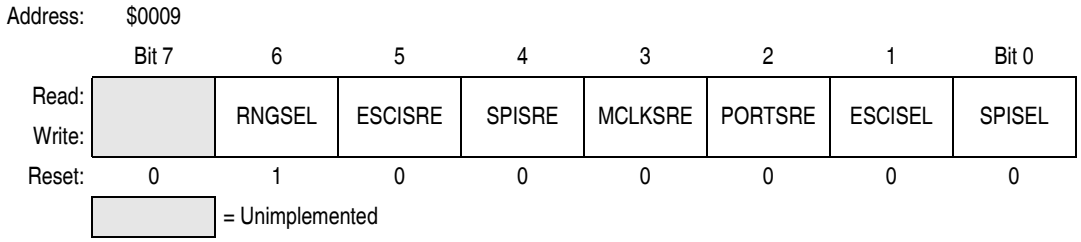



**Figure 5-3. Configuration Register 3 (CONFIG3)**

**RNGSEL — External Oscillator Frequency Range Select**

RNGSEL works in conjunction with EXTSLOW to enable the amplifiers for the crystal oscillator.

**Table 5-2. External Crystal Frequency Range Selection**

| EXTSLOW | RNGSEL | Frequency Range |
|---|---|---|
| 0 | 0 | 8–32 MHz |
| 0 | 1 | 1–8 MHz |
| 1 | 0 | 32–100 kHz |
| 1 | 1 | Reserved |

Setting EXTSLOW will force RNGSEL to a 0. RNGSEL cannot be written if EXTSLOW = 1.

**ESCISRE — Slew Rate Enable for ESCI**

1 = Slew rate controlled output for TxD
0 = Normal output

**SPISRE — Slew Rate Enable for SPI**

1 = Slew rate controlled output for MISO, MOSI, and SCK
0 = Normal outputs

**MCLKSRE — Slew Rate Enable for MCLK**

1 = Slew rate controlled output for MCLK
0 = Normal output

**PORTSRE — Slew Rate Enable for Ports**

1 = Slew rate controlled output for all ports
0 = Normal output

**ESCISEL — ESCI Pin Selection Bit**

ESCISEL is used to select the pins to be used as ESCI pins when the ESCI is enabled. For more information on the ESCI, see Chapter 13 Enhanced Serial Communications Interface (ESCI) Module.

1 = TxD on PTA2 — RxD on PTA3
0 = TxD on PTE0 — RxD on PTE1

**SPISEL — SPI Pin Selection Bit**

SPISEL is used to select the pins to be used as SPI pins when the SPI is enabled. For more information on the SPI, see Chapter 15 Serial Peripheral Interface (SPI) Module.

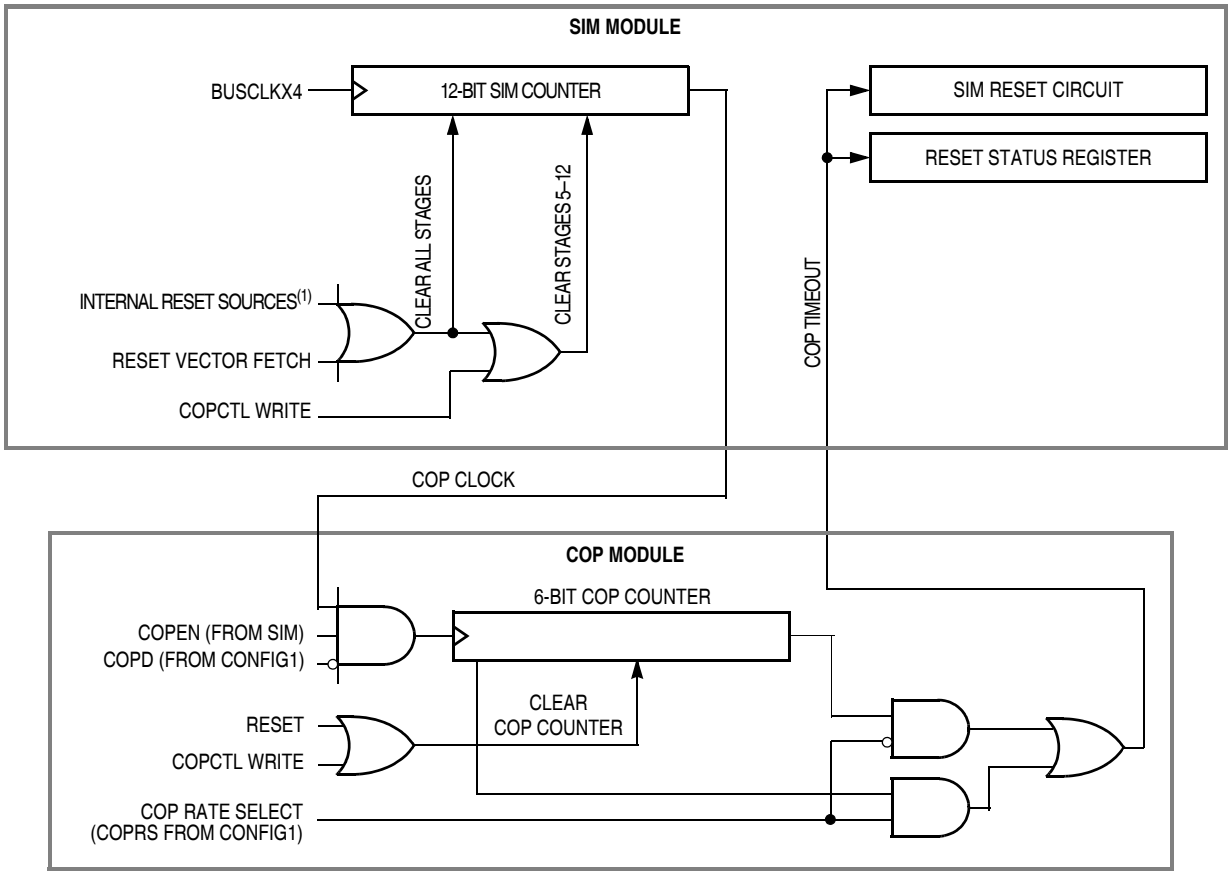
1 = MISO on PTB3 — MOSI on PTB4 — SPSCK on PTB5 — $\overline{SS}$ on PTC2
0 = MISO on PTC0 — MOSI on PTC1 — SPSCK on PTA5 — $\overline{SS}$ on PTA6

# Chapter 6
# Computer Operating Properly

## 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 periodically clearing the COP counter.

## 6.2 Functional Description




1. See Chapter 14 System Integration Module (SIM) for more details.

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

The COP counter is a free-running 6-bit counter preceded by a 12-bit prescaler. If not cleared by software, the COP counter overflows and generates an asynchronous reset after 8176 or 262,128 CGMXCLK cycles, depending on the state of the COP rate select bit, COPRS, in the CONFIG1. When COPRS = 0, 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 4–12 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 $\overline{\text{IRQ}}$ pin 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 signal clears the COP prescaler.

### 6.3.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see 6.4 COP Control Register) clears the COP counter and clears stages 12 through 4 of the COP prescaler. Reading the COP control register returns the reset vector.

### 6.3.4 Power-On Reset

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

### 6.3.5 Internal Reset

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

### 6.3.6 Reset Vector Fetch

A reset vector fetch occurs when the vector address appears on the data bus. A reset vector fetch clears the COP prescaler.

### 6.3.7 COPD

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register. See Chapter 5 Configuration Registers (CONFIG1, CONFIG2, CONFIG3).

### 6.3.8 COPRS

The COPRS signal reflects the state of the COP rate select bit (COPRS) in the configuration register. See Chapter 5 Configuration Registers (CONFIG1, CONFIG2, CONFIG3).

## 6.4 COP Control Register

The COP control register 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.

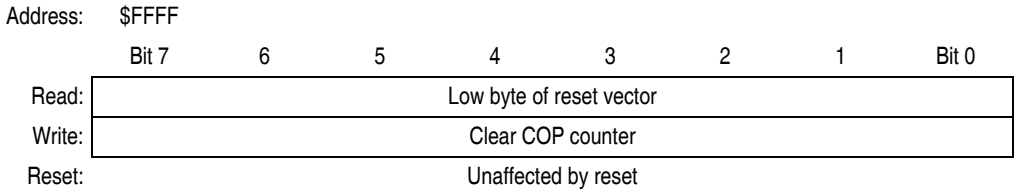



**Figure 6-2. COP Control Register (COPCTL)**

## 6.5 Interrupts

The COP does not generate CPU interrupt requests.

## 6.6 Monitor Mode

The COP is disabled in monitor mode when $V_{TST}$ is present on the $\overline{IRQ}$ pin or on the $\overline{RST}$ pin.

## 6.7 Low-Power Modes

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

### 6.7.1 Wait Mode

The COP remains active in wait mode. To prevent a COP reset during wait mode, periodically clear the COP counter in a CPU interrupt routine.

### 6.7.2 Stop Mode

Stop mode turns off the CGMXCLK input to the COP and clears the COP prescaler. 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 configuration register (CONFIG) enables the STOP instruction. To prevent inadvertently turning off the COP with a STOP instruction, disable the STOP instruction by clearing the STOP bit.

## 6.8 COP Module During Break Interrupts

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.

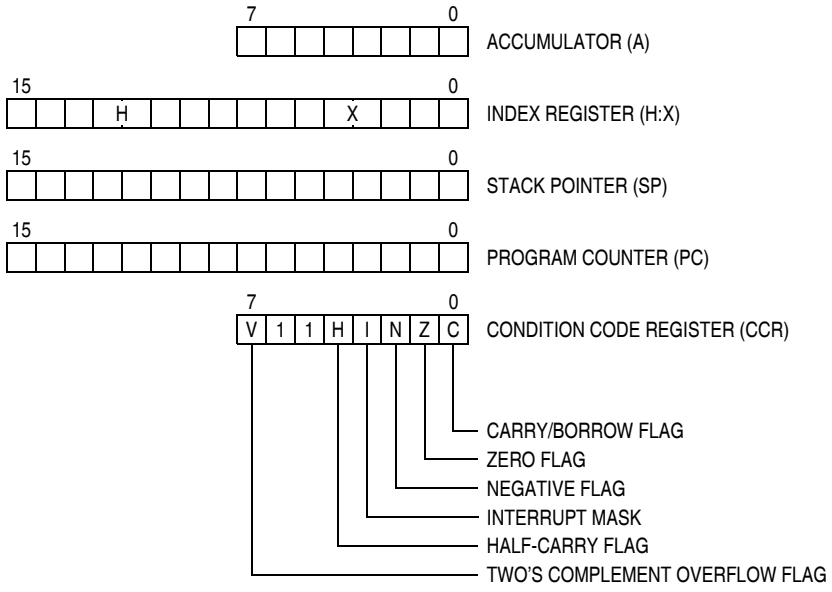



**Figure 7-1. CPU Registers**

### 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.

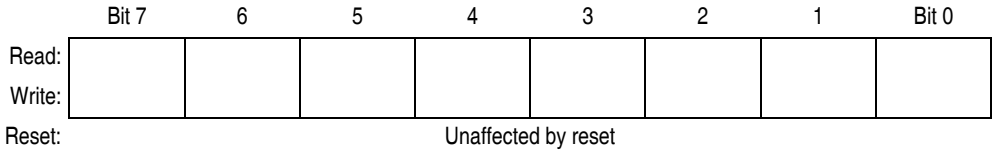



**Figure 7-2. Accumulator (A)**

### 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.

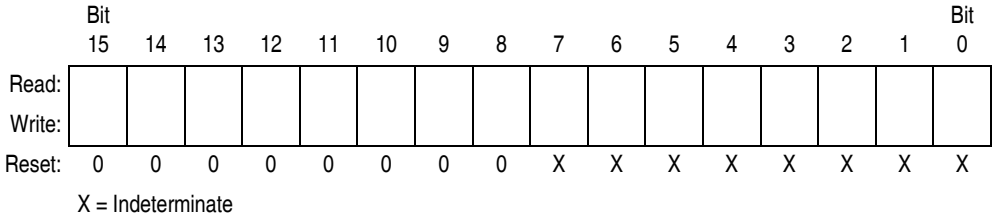



**Figure 7-3. Index Register (H:X)**

### 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.

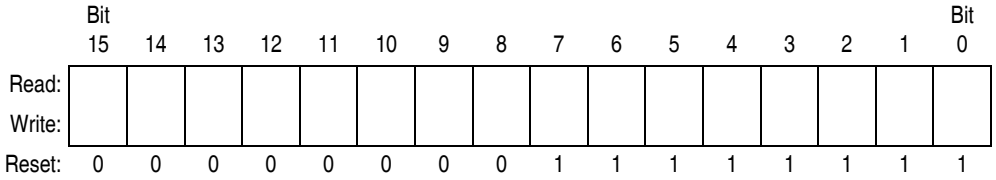



**Figure 7-4. Stack Pointer (SP)**

***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.

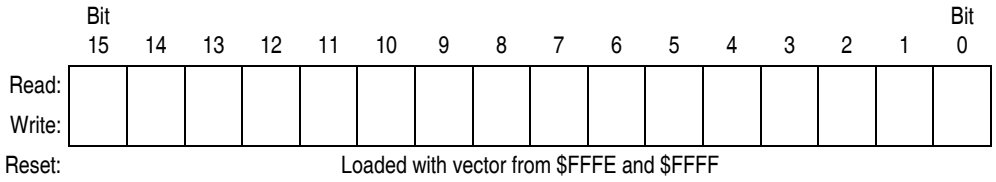



**Figure 7-5. Program Counter (PC)**

### 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

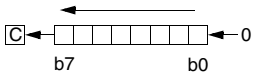
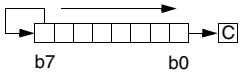
The break interrupt begins after completion of the CPU instruction in progress. If the break address register match occurs on the last cycle of a CPU instruction, the break interrupt begins immediately.

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 |

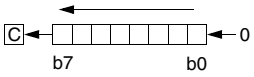
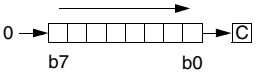
## Table 7-1. Instruction Set Summary (Sheet 2 of 6)

| 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 $\overline{IRQ}$ Pin High | PC ← (PC) + 2 + *rel* ? $\overline{IRQ}$ = 1 | – | – | – | – | – | – | REL | 2F | rr | 3 |
| BIL *rel* | Branch if $\overline{IRQ}$ Pin Low | PC ← (PC) + 2 + *rel* ? $\overline{IRQ}$ = 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 |

## 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 | | | | |
| CLI | Clear Interrupt Mask | I ← 0 | – | – | 0 | – | – | – | INH | 9A | | 2 |
| 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)

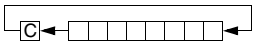
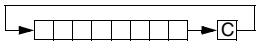
| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| PULA | Pull A from Stack | SP ← (SP + 1); Pull (A) | – | – | – | – | – | – | INH | 86 | | 2 |
| 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 | *n* | Any bit |
| C | Carry/borrow bit | *opr* | Operand (one or two bytes) |
| CCR | Condition code register | PC | Program counter |
| dd | Direct address of operand | PCH | Program counter high byte |
| dd rr | Direct address of operand and relative offset of branch instruction | PCL | Program counter low byte |
| DD | Direct to direct addressing mode | REL | Relative addressing mode |
| DIR | Direct addressing mode | *rel* | Relative program counter offset byte |
| DIX+ | Direct to indexed with post increment addressing mode | rr | Relative program counter offset byte |
| ee ff | High and low bytes of offset in indexed, 16-bit offset addressing | SP1 | Stack pointer, 8-bit offset addressing mode |
| EXT | Extended addressing mode | SP2 | Stack pointer 16-bit offset addressing mode |
| ff | Offset byte in indexed, 8-bit offset addressing | SP | Stack pointer |
| H | Half-carry bit | U | Undefined |
| H | Index register high byte | V | Overflow bit |
| hh ll | High and low bytes of operand address in extended addressing | X | Index register low byte |
| I | Interrupt mask | Z | Zero bit |
| ii | Immediate operand byte | & | Logical AND |
| IMD | Immediate source to direct destination addressing mode | \| | Logical OR |
| IMM | Immediate addressing mode | ⊕ | Logical EXCLUSIVE OR |
| INH | Inherent addressing mode | ( ) | Contents of |
| IX | Indexed, no offset addressing mode | –( ) | Negation (two's complement) |
| IX+ | Indexed, no offset, post increment addressing mode | # | Immediate value |
| IX+D | Indexed with post increment to direct addressing mode | « | Sign extend |
| IX1 | Indexed, 8-bit offset addressing mode | ← | Loaded with |
| IX1+ | Indexed, 8-bit offset, post increment addressing mode | ? | If |
| IX2 | Indexed, 16-bit offset addressing mode | : | Concatenated with |
| M | Memory location | ↕ | Set or cleared |
| N | Negative bit | — | 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 | CI 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 | ST 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 | OI 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
# Internal Clock Generator (ICG) Module

## 8.1 Introduction

The internal clock generator (ICG) module is used to create a stable clock source for the microcontroller without using any external components. The ICG generates the oscillator output clock (CGMXCLK), which is used by the computer operating properly (COP), low-voltage inhibit (LVI), and other modules. The ICG also generates the clock generator output (CGMOUT), which is fed to the system integration module (SIM) to create the bus clocks. The bus frequency will be one-fourth the frequency of CGMXCLK and one-half the frequency of CGMOUT. Finally, the ICG generates the timebase clock (TBMCLK), which is used in the timebase module (TBM).

## 8.2 Features

The ICG has these features:

- Selectable external clock generator, either 1-pin external source or 2-pin crystal, multiplexed with port pins
- Internal clock generator with programmable frequency output in integer multiples of a nominal frequency (307.2 kHz ± 25 percent)
- Internal oscillator trimmed accuracy of ±3.5 percent
- Bus clock software selectable from either internal or external clock (bus frequency range from 76.8 kHz ± 25 percent to 9.75 MHz ± 25 percent in 76.8-kHz increments)

***NOTE***

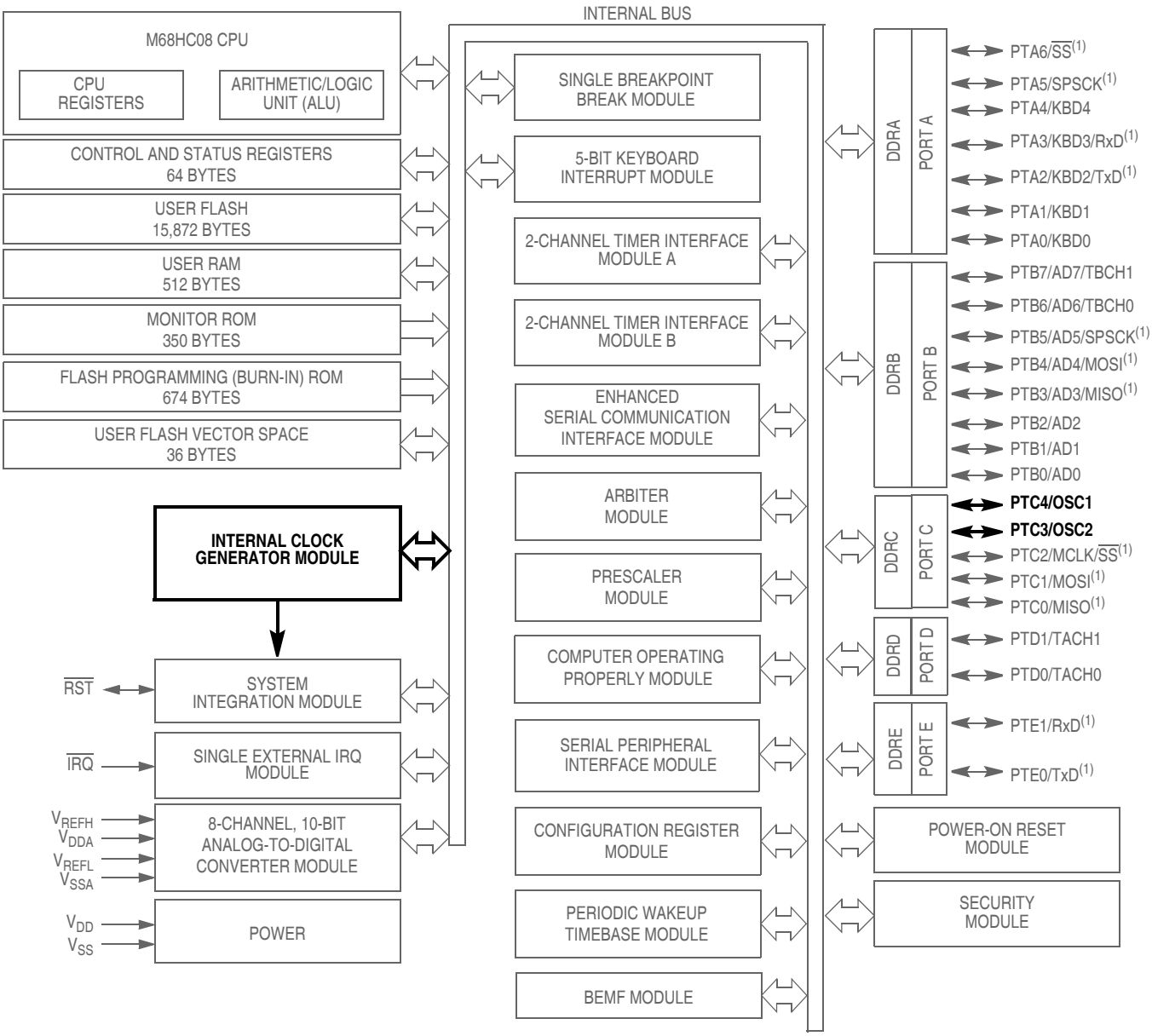
*For the MC68HC908EY16A, do not exceed the maximum bus frequency of 8 MHz at 5.0 V.*

- Timebase clock automatically selected from external clock if external clock is available
- Clock monitor for both internal and external clocks

## 8.3 Functional Description

The ICG, shown in Figure 8-2, contains these major submodules:

- Clock enable circuit
- Internal clock generator
- External clock generator
- Clock monitor circuit
- Clock selection circuit




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

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

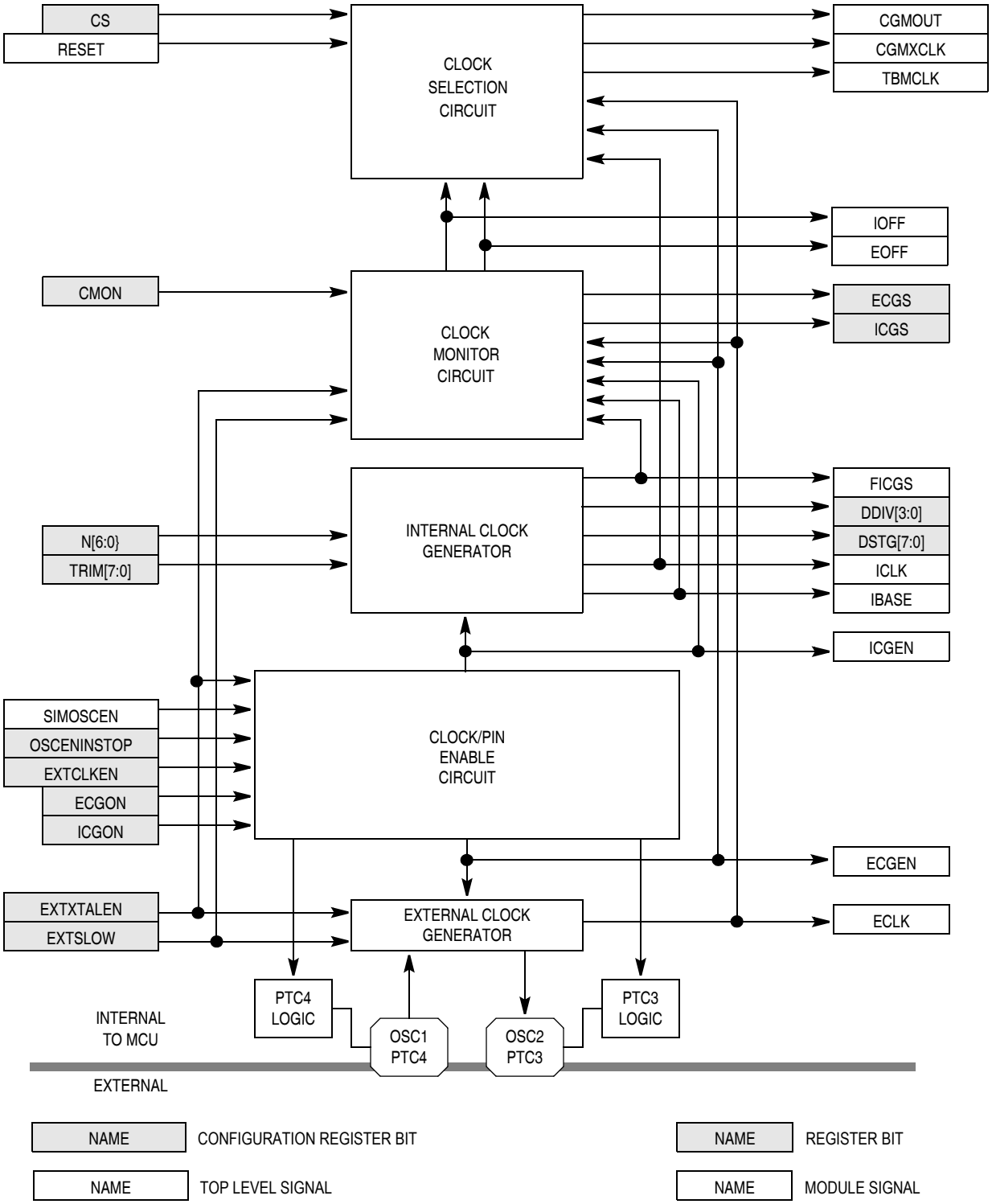



**Figure 8-2. ICG Module Block Diagram**

### 8.3.1 Clock Enable Circuit

The clock enable circuit is used to enable the internal clock (ICLK) or external clock (ECLK) and the port logic which is shared with the oscillator pins (OSC1 and OSC2). The clock enable circuit generates an ICG stop (ICGSTOP) signal which stops all clocks (ICLK, ECLK, and the low-frequency base clock, IBASE). ICGSTOP is set and the ICG is disabled in stop mode if the oscillator enable stop bit (OSCENINSTOP) in the configuration (CONFIG) register is clear. The ICG clocks will be enabled in stop mode if OSCENINSTOP is high.

The internal clock enable signal (ICGEN) turns on the internal clock generator which generates ICLK. ICGEN is set (active) whenever the ICGON bit is set and the ICGSTOP signal is clear. When ICGEN is clear, ICLK and IBASE are both low.

The external clock enable signal (ECGEN) turns on the external clock generator which generates ECLK. ECGEN is set (active) whenever the ECGON bit is set and the ICGSTOP signal is clear. ECGON cannot be set unless the external clock enable (EXTCLKEN) bit in the CONFIG is set. when ECGEN is clear, ECLK is low.

The port C4 enable signal (PC4EN) turns on the port C4 logic. Since port C4 is on the same pin as OSC1, this signal is only active (set) when the external clock function is not desired. Therefore, PC4EN is clear when ECGON is set. PC4EN is not gated with ICGSTOP, which means that if the ECGON bit is set, the port C4 logic will remain disabled in stop mode.

The port C3 enable signal (PC3EN) turns on the port C3 logic. Since port C3 is on the same pin as OSC2, this signal is only active (set) when 2-pin oscillator function is not desired. Therefore, PC3EN is clear when ECGON and the external crystal enable (EXTXTALEN) bit in the CONFIG are both set. PC4EN is not gated with ICGSTOP, which means that if ECGON and EXTXTALEN are set, the port C3 logic will remain disabled in stop mode.

### 8.3.2 Internal Clock Generator

The internal clock generator, shown in Figure 8-3, creates a low frequency base clock (IBASE), which operates at a nominal frequency ($f_{NOM}$) of 307.2 kHz $\pm$ 25 percent, and an internal clock (ICLK) which is an integer multiple of IBASE. This multiple is the ICG multiplier factor (N), which is programmed in the ICG multiplier register (ICGMR). The internal clock generator is turned off and the output clocks (IBASE and ICLK) are held low when the internal clock generator enable signal (ICGEN) is clear.

The internal clock generator contains:

- A digitally controlled oscillator
- A modulo N divider
- A frequency comparator, which contains voltage and current references, a frequency to voltage converter, and comparators
- A digital loop filter

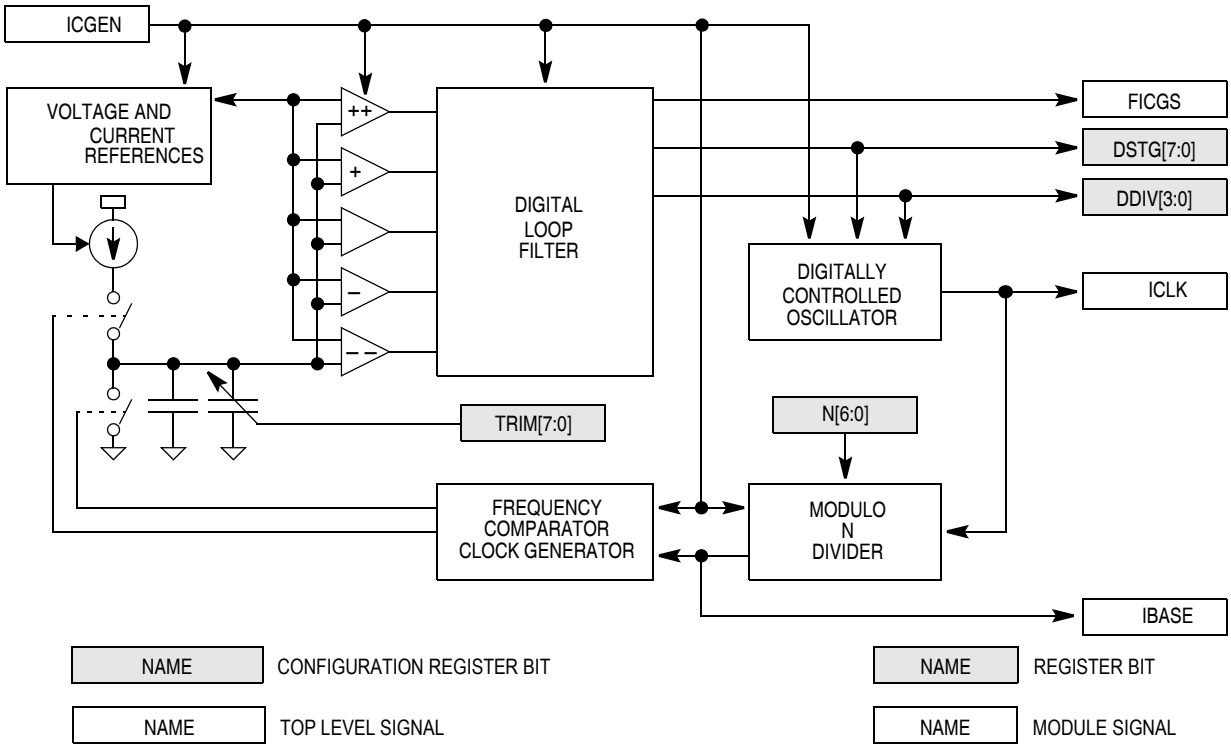



**Figure 8-3. Internal Clock Generator Block Diagram**

### *8.3.2.1 Digitally Controlled Oscillator*

The digitally controlled oscillator (DCO) is an inaccurate oscillator which generates the internal clock (ICLK). The clock period of ICLK is dependent on the digital loop filter outputs (DSTG[7:0] and DDIV[3:0]). Because of only a limited number of bits in DDIV and DSTG, the precision of the output (ICLK) is restricted to a precision of approximately ±0.202 percent to ±0.368 percent when measured over several cycles (of the desired frequency). Additionally, since the propagation delays of the devices used in the DCO ring oscillator are a measurable fraction of the bus clock period, reaching the long-term precision may require alternately running faster and slower than desired, making the worst case cycle-to-cycle frequency variation ±6.45 percent to ±11.8 percent (of the desired frequency). The valid values of DDIV:DSTG range from $000 to $9FF. For more information on the quantization error in the DCO, see 8.4.4 Quantization Error in DCO Output.

### *8.3.2.2 Modulo N Divider*

The modulo N divider creates the low-frequency base clock (IBASE) by dividing the internal clock (ICLK) by the ICG multiplier factor (N), contained in the ICG multiplier register (ICGMR). When N is programmed to a $01 or $00, the divider is disabled and ICLK is passed through to IBASE undivided. When the internal clock generator is stable, the frequency of IBASE will be equal to the nominal frequency ($f_{NOM}$) of 307.2 kHz ± 25 percent.

#### 8.3.2.3 Frequency Comparator

The frequency comparator effectively compares the low-frequency base clock (IBASE) to a nominal frequency, $f_{NOM}$. First, the frequency comparator converts IBASE to a voltage by charging a known capacitor with a current reference for a period dependent on IBASE. This voltage is compared to a voltage reference with comparators, whose outputs are fed to the digital loop filter. The dependence of these outputs on the capacitor size, current reference, and voltage reference causes up to ±25 percent error in $f_{NOM}$.

#### 8.3.2.4 Digital Loop Filter

The digital loop filter (DLF) uses the outputs of the frequency comparator to adjust the internal clock (ICLK) clock period. The DLF generates the DCO divider control bits (DDIV[3:0]) and the DCO stage control bits (DSTG[7:0]), which are fed to the DCO. The DLF first concatenates the DDIV and DSTG registers (DDIV[3:0]:DSTG[7:0]) and then adds or subtracts a value dependent on the relative error in the low-frequency base clock's period, as shown in Table 8-1. In some extreme error conditions, such as operating at a $V_{DD}$ level which is out of specification, the DLF may attempt to use a value above the maximum ($9FF) or below the minimum ($000). In both cases, the value for DDIV will be between $A and $F. In this range, the DDIV value will be interpreted the same as $9 (the slowest condition). Recovering from this condition requires subtracting (increasing frequency) in the normal fashion until the value is again below $9FF. (If the desired value is $9xx, the value may settle at $Axx through $Fxx. This is an acceptable operating condition.) If the error is less than ±5 percent, the internal clock generator's filter stable indicator (FICGS) is set, indicating relative frequency accuracy to the clock monitor.

**Table 8-1. Correction Sizes from DLF to DCO**

| Frequency Error of IBASE Compared to $f_{NOM}$ | DDVI[3:0]:DSTG[7:0] Correction | Current to New DDIV[3:0]:DSTG[7:0](1) | | Relative Correction in DCO | |
|---|---|---|---|---|---|
| IBASE < 0.85 $f_{NOM}$ | –32 (–$020) | Minimum | $xFF to $xDF | –2/31 | –6.45% |
| | | Maximum | $x20 to $x00 | –2/19 | –10.5% |
| 0.85 $f_{NOM}$ < IBASE IBASE < 0.95 $f_{NOM}$ | –8 (–$008) | Minimum | $xFF to $xF7 | –0.5/31 | –1.61% |
| | | Maximum | $x08 to $x00 | –0.5/17.5 | –2.86% |
| 0.95 $f_{NOM}$ < IBASE IBASE < $f_{NOM}$ | –1 (–$001) | Minimum | $xFF to $xFE | –0.0625/31 | –0.202% |
| | | Maximum | $x01 to $x00 | –0.0625/17.0625 | –0.366% |
| $f_{NOM}$ < IBASE IBASE < 1.05 $f_{NOM}$ | +1 (+$001) | Minimum | $xFE to $xFF | +0.0625/30.9375 | +0.202% |
| | | Maximum | $x00 to $x01 | +0.0625/17 | +0.368% |
| 1.05 $f_{NOM}$ < IBASE IBASE < 1.15 $f_{NOM}$ | +8 (+$008) | Minimum | $xF7 to $xFF | +0.5/30.5 | +1.64% |
| | | Maximum | $x00 to $x08 | +0.5/17 | +2.94% |
| 1.15 $f_{NOM}$ < IBASE | +32 (+$020) | Minimum | $xDF to $xFF | +2/29 | +6.90% |
| | | Maximum | $x00 to $x20 | +2/17 | +11.8% |

1. x = Maximum error is independent of value in DDIV[3:0]. DDIV increments or decrements when an addition to DSTG[7:0] carries or borrows.

### 8.3.3 External Clock Generator

The ICG also provides for an external oscillator or external clock source, if desired. The external clock generator, shown in Figure 8-4, contains an external oscillator amplifier and an external clock input path.

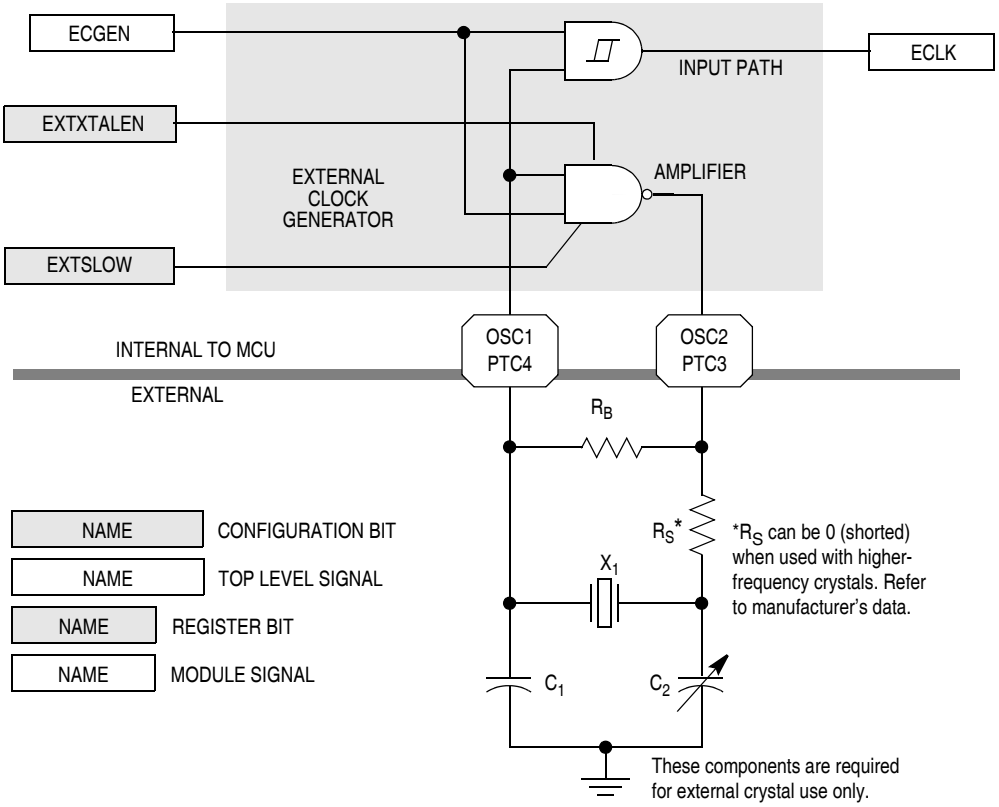



**Figure 8-4. External Clock Generator Block Diagram**

### *8.3.3.1 External Oscillator Amplifier*

The external oscillator amplifier provides the gain required by an external crystal connected in a Pierce oscillator configuration. The amount of this gain is controlled by the slow external (EXTSLOW) bit in the CONFIG. When EXTSLOW is set, the amplifier gain is reduced for operating low-frequency crystals (32 kHz to 100 kHz). When EXTSLOW is clear, the amplifier gain will be sufficient for 1-MHz to 8-MHz crystals. EXTSLOW must be configured correctly for the given crystal or the circuit may not operate.

The amplifier is enabled when the external clock generator enable (ECGEN) signal is set and when the external crystal enable (EXTXTALEN) bit in the CONFIG is set. ECGEN is controlled by the clock enable circuit (see 8.3.1 Clock Enable Circuit) and indicates that the external clock function is desired. When enabled, the amplifier will be connected between the PTC4/OSC1 and PTC3/OSC2 pins. Otherwise, the PTC3/OSC2 pin reverts to its port function.

In its typical configuration, the external oscillator requires five external components:

1. Crystal, $X_1$
2. Fixed capacitor, $C_1$
3. Tuning capacitor, $C_2$ (can also be a fixed capacitor)
4. Feedback resistor, RB
5. Series resistor, $R_S$ (included in Figure 8-4 to follow strict Pierce oscillator guidelines and may not be required for all ranges of operation, especially with high frequency crystals. Refer to the crystal manufacturer's data for more information.)

### *8.3.3.2 External Clock Input Path*

The external clock input path is the means by which the microcontroller uses an external clock source. The input to the path is the PTC4/OSC1 pin and the output is the external clock (ECLK). The path, which contains input buffering, is enabled when the external clock generator enable signal (ECGEN) is set. When not enabled, the PTC4/OSC1 pin reverts to its port function.

## 8.3.4 Clock Monitor Circuit

The ICG contains a clock monitor circuit which, when enabled, will continuously monitor both the external clock (ECLK) and the internal clock (ICLK) to determine if either clock source has been corrupted. The clock monitor circuit, shown in Figure 8-5, contains these blocks:

- Clock monitor reference generator
- Internal clock activity detector
- External clock activity detector

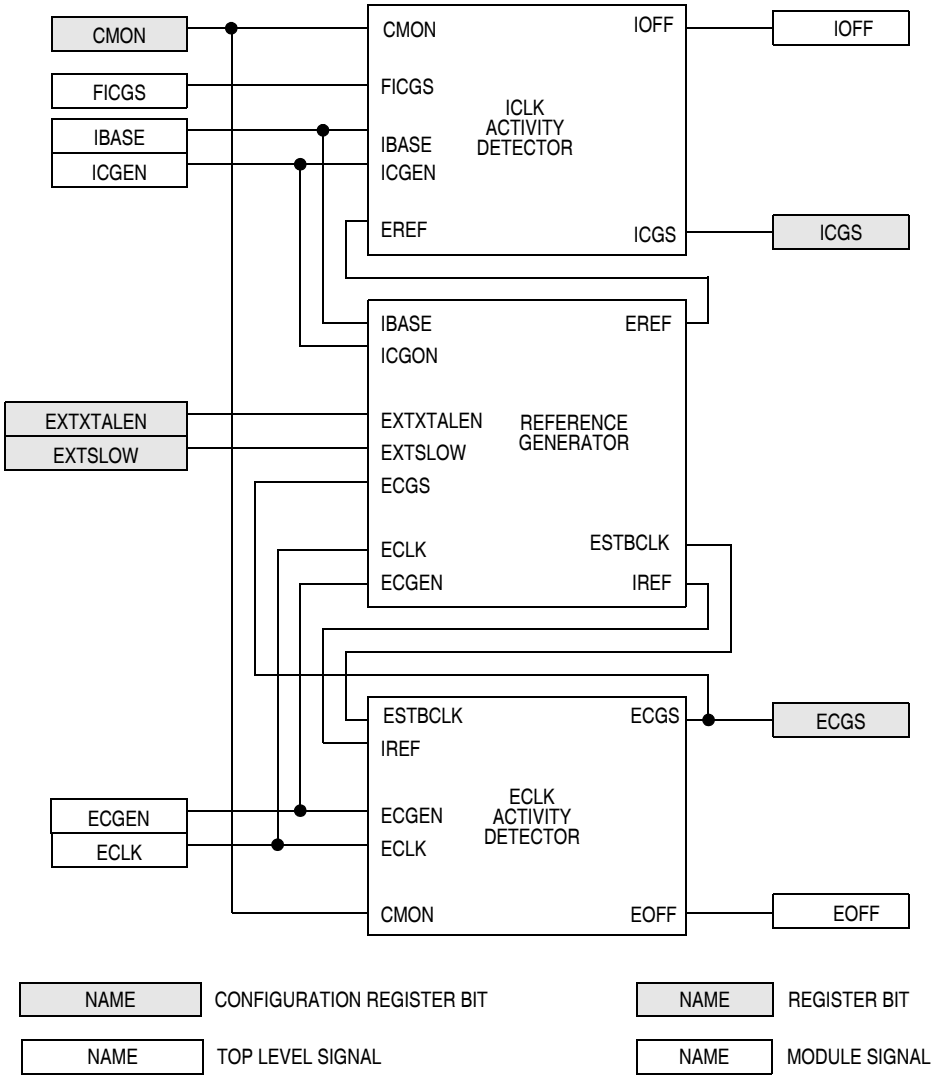



**Figure 8-5. Clock Monitor Block Diagram**

#### 8.3.4.1 Clock Monitor Reference Generator

The clock monitor uses a reference based on one clock source to monitor the other clock source. The clock monitor reference generator generates the external reference clock (EREF) based on the external clock (ECLK) and the internal reference clock (IREF) based on the internal clock (ICLK). To simplify the circuit, the low-frequency base clock (IBASE) is used in place of ICLK because it always operates at or near 307.2 kHz. For proper operation, EREF must be at least twice as slow as IBASE and IREF must be at least twice as slow as ECLK.

To guarantee that IREF is slower than ECLK and EREF is slower than IBASE, one of the signals is divided down. Which signal is divided and by how much is determined by the external slow (EXTSLOW) and external crystal enable (EXTXTALEN) bits in the CONFIG, according to the rules in Table 8-2.

**NOTE**

*Each signal (IBASE and ECLK) is always divided by four. A longer divider is used on either IBASE or ECLK based on the EXTSLOW bit.*

To conserve size, the long divider (divide by 4096) is also used as an external crystal stabilization divider. The divider is reset when the external clock generator is turned off or in stop mode (ECGEN is clear). When the external clock generator is first turned on, the external clock generator stable bit (ECGS) will be clear. This condition automatically selects ECLK as the input to the long divider. The external stabilization clock (ESTBCLK) will be ECLK divided by 16 when EXTXTALEN is low or 4096 when EXTXTALEN is high. This timeout allows the crystal to stabilize. The falling edge of ESTBCLK is used to set ECGS, which will set after a full 16 or 4096 cycles. When ECGS is set, the divider returns to its normal function. ESTBCLK may be generated by either IBASE or ECLK, but any clocking will only reinforce the set condition. If ECGS is cleared because the clock monitor determined that ECLK was inactive, the divider will revert to a stabilization divider. Since this will change the EREF and IREF divide ratios, it is important to turn the clock monitor off (CMON = 0) after inactivity is detected to ensure valid recovery.

#### 8.3.4.2 Internal Clock Activity Detector

The internal clock activity detector, shown in Figure 8-6, looks for at least one falling edge on the low-frequency base clock (IBASE) every time the external reference (EREF) is low. Since EREF is less than half the frequency of IBASE, this should occur every time. If it does not occur two consecutive times, the internal clock inactivity indicator (IOFF) is set. IOFF will be cleared the next time there is a falling edge of IBASE while EREF is low.

The internal clock stable bit (ICGS) is also generated in the internal clock activity detector. ICGS is set when the internal clock generator's filter stable signal (FICGS) indicates that IBASE is within about 5 percent of the target 307.2 kHz $\pm$ 25 percent for two consecutive measurements. ICGS is cleared when FICGS is clear, the internal clock generator is turned off or is in stop mode (ICGEN is clear), or when IOFF is set.

#### 8.3.4.3 External Clock Activity Detector

The external clock activity detector, shown in Figure 8-7, looks for at least one falling edge on the external clock (ECLK) every time the internal reference (IREF) is low. Since IREF is less than half the frequency of ECLK, this should occur every time. If it does not occur two consecutive times, the external clock inactivity indicator (EOFF) is set. EOFF will be cleared the next time there is a falling edge of ECLK while IREF is low.

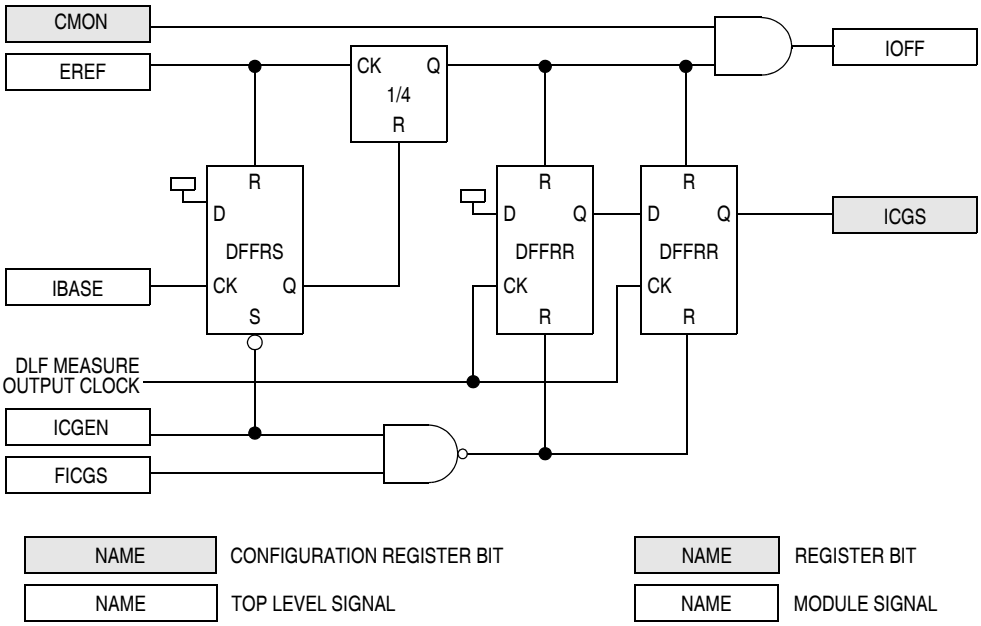



**Figure 8-6. Internal Clock Activity Detector**

The external clock stable bit (ECGS) is also generated in the external clock activity detector. ECGS is set on a falling edge of the external stabilization clock (ESTBCLK). This will be 4096 ECLK cycles after the external clock generator on bit is set, or the MCU exits stop mode (ECGEN = 1) if the external crystal enable (EXTXTALEN) in the CONFIG is set, or 16 cycles when EXTXTALEN is clear. ECGS is cleared when the external clock generator is turned off or in stop mode (ECGEN is clear) or when EOFF is set.

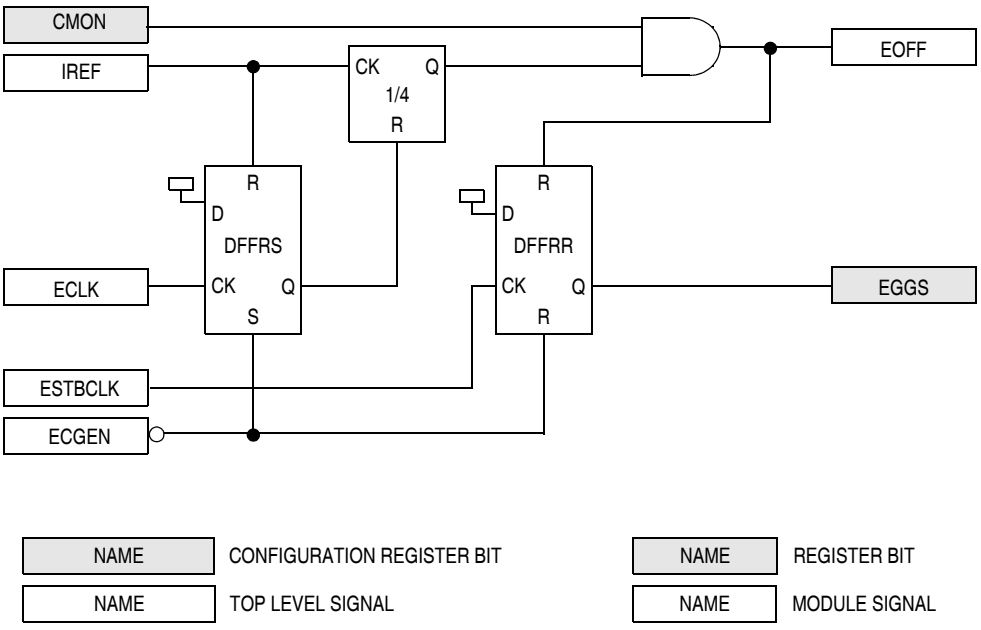



**Figure 8-7. External Clock Activity Detector**

### 8.3.5 Clock Selection Circuit

The clock selection circuit, shown in Figure 8-8, contains two clock switches which generate the oscillator output clock (CGMXCLK) and the timebase clock (TBMCLK) from either the internal clock (ICLK) or the external clock (ECLK). The clock selection circuit also contains a divide-by-two circuit which creates the clock generator output clock (CGMOUT), which generates the bus clocks.

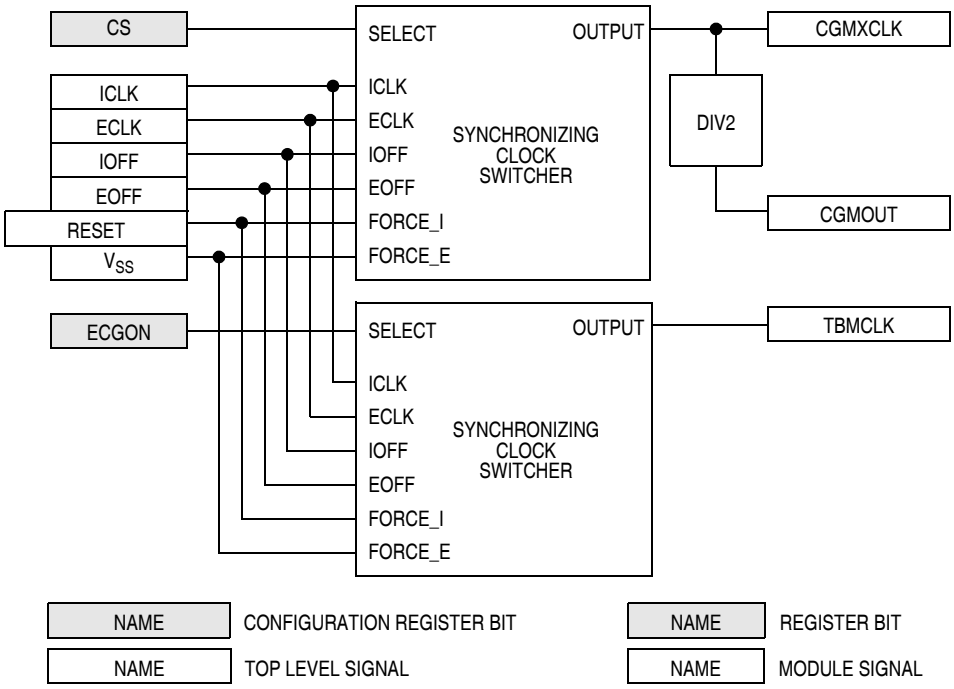



**Figure 8-8. Clock Selection Circuit Block Diagram**

#### *8.3.5.1 Clock Selection Switches*

The first switch creates the oscillator output clock (CGMXCLK) from either the internal clock (ICLK) or the external clock (ECLK), based on the clock select bit (CS; set selects ECLK, clear selects ICLK). When switching the CS bit, both ICLK and ECLK must be on (ICGON and ECGON set). The clock being switched to also must be stable (ICGS or ECGS set).

The second switch creates the timebase clock (TBMCLK) from ICLK or ECLK based on the external clock on bit. When ECGON is set, the switch automatically selects the external clock, regardless of the state of the ECGS bit.

#### *8.3.5.2 Clock Switching Circuit*

To robustly switch between the internal clock (ICLK) and the external clock (ECLK), the switch assumes the clocks are completely asynchronous, so a synchronizing circuit is required to make the transition. When the select input (the clock select bit for the oscillator output clock switch or the external clock on bit for the timebase clock switch) is changed, the switch will continue to operate off the original clock for between one and two cycles as the select input is transitioned through one side of the synchronizer. Next, the output will be held low for between one and two cycles of the new clock as the select input transitions through the other side. Then the output starts switching at the new clock's frequency. This transition guarantees that no glitches will be seen on the output even though the select input may change

asynchronously to the clocks. The unpredictably of the transition period is a necessary result of the asynchronicity.

The switch automatically selects ICLK during reset. When the clock monitor is on (CMON is set) and it determines one of the clock sources is inactive (as indicated by the IOFF or EOFF signals), the circuit is forced to select the active clock. There are no clocks for the inactive side of the synchronizer to properly operate, so that side is forced deselected. However, the active side will not be selected until one to two clock cycles after the IOFF or EOFF signal transitions.

## 8.4 Usage Notes

The ICG has several features which can provide protection to the microcontroller if properly used. Other features can greatly simplify usage of the ICG if certain techniques are employed. This section describes several possible ways to use the ICG and its features. These techniques are not the only ways to use the ICG and may not be optimum for all environments. In any case, these techniques should be used only as a template, and the user should modify them according to the application's requirements.

These notes include:

- Switching clock sources
- Enabling the clock monitor
- Using clock monitor interrupts
- Quantization error in digitally controlled oscillator (DCO) output
- Switching internal clock frequencies
- Nominal frequency settling time
- Improving frequency settling time
- Trimming frequency

### 8.4.1 Switching Clock Sources

Switching from one clock source to another requires both clock sources to be enabled and stable. A simple flow requires:

- Enable desired clock source
- Wait for it to become stable
- Switch clocks
- Disable previous clock source

The key point to remember in this flow is that the clock source cannot be switched (CS cannot be written) unless the desired clock is on and stable.

### 8.4.2 Enabling the Clock Monitor

Many applications require the clock monitor to determine if one of the clock sources has become inactive, so the other can be used to recover from a potentially dangerous situation. Using the clock monitor requires both clocks to be active (ECGON and ICGON both set). To enable the clock monitor, both clocks also must be stable (ECGS and ICGS both set). This is to prevent the use of the clock monitor when a clock is first turned on and potentially unstable.

Enabling the clock monitor and clock monitor interrupts requires a flow similar to this:

- Enable the alternate clock source
- Wait for both clock sources to be stable
- Switch to the desired clock source if necessary
- Enable the clock monitor
- Enable clock monitor interrupts

These events must happen in sequence.

### 8.4.3 Using Clock Monitor Interrupts

The clock monitor circuit can be used to recover from perilous situations such as crystal loss. To use the clock monitor effectively, these points should be observed:

- Enable the clock monitor and clock monitor interrupts.
- The first statement in the clock monitor interrupt service routine (CMISR) should be a read to the ICG control register (ICGCR) to verify that the clock monitor flag (CMF) is set. This is also the first step in clearing the CMF bit.
- The second statement in the CMISR should be a write to the ICGCR to clear the CMF bit (write the bit low). Writing the bit high will not affect it. This statement does not need to immediately follow the first, but must be contained in the CMISR.
- The third statement in the CMISR should be to clear the CMON bit. This is required to ensure proper reconfiguration of the reference dividers. This statement also must be contained in the CMISR.
- Although the clock monitor can be enabled only when both clocks are stable (ICGS is set or ECGS is set), it will remain set if one of the clocks goes unstable.
- The clock monitor only works if the external slow (EXTSLOW) bit in the CONFIG is set to the correct value.
- The internal and external clocks must both be enabled and running to use the clock monitor.
- When the clock monitor detects inactivity, the inactive clock is automatically deselected and the active clock selected as the source for CGMXCLK and TBMCLK. The CMISR can use the state of the CS bit to check which clock is inactive.
- When the clock monitor detects inactivity, the application may have been subjected to extreme conditions which may have affected other circuits. The CMISR should take any appropriate precautions.

### 8.4.4 Quantization Error in DCO Output

The digitally controlled oscillator (DCO) is comprised of three major sub-blocks:

1. Binary weighted divider
2. Variable-delay ring oscillator
3. Ring oscillator fine-adjust circuit

Each of these blocks affects the clock period of the internal clock (ICLK). Since these blocks are controlled by the digital loop filter (DLF) outputs DDIV and DSTG, the output of the DCO can change only in quantized steps as the DLF increments or decrements its output. The following sections describe how each block will affect the output frequency.

#### 8.4.4.1 Digitally Controlled Oscillator

The digitally controlled oscillator (DCO) is an inaccurate oscillator which generates the internal clock (ICLK), whose clock period is dependent on the digital loop filter outputs (DSTG[7:0] and DDIV[3:0]). Because of the digital nature of the DCO, the clock period of ICLK will change in quantized steps. This will create a clock period difference or quantization error (Q-ERR) from one cycle to the next. Over several cycles or for longer periods, this error is divided out until it reaches a minimum error of 0.202 percent to 0.368 percent. The dependence of this error on the DDIV[3:0] value and the number of cycles the error is measured over is shown in Table 8-2.

#### 8.4.4.2 Binary Weighted Divider

The binary weighted divider divides the output of the ring oscillator by a power of two, specified by the DCO divider control bits (DDIV[3:0]). DDIV maximizes at %1001 (values of %1010 through %1111 are interpreted as %1001), which corresponds to a divide by 512. When DDIV is %0000, the ring oscillator's output is divided by 1. Incrementing DDIV by one will double the period; decrementing DDIV will halve the period. The DLF cannot directly increment or decrement DDIV; DDIV is only incremented or decremented when an addition or subtraction to DSTG carries or borrows.

**Table 8-2. Quantization Error in ICLK**

| DDIV[3:0] | ICLK Cycles | Bus Cycles | $\tau_{ICLK}$ Q-ERR |
|---|---|---|---|
| %0000 (min) | 1 | NA | 6.45%–11.8% |
| %0000 (min) | 4 | 1 | 1.61%–2.94% |
| %0000 (min) | ≥ 32 | ≥ 8 | 0.202%–0.368% |
| %0001 | 1 | NA | 3.23%–5.88% |
| %0001 | 4 | 1 | 0.806%–1.47% |
| %0001 | ≥ 16 | ≥ 4 | 0.202%–0.368% |
| %0010 | 1 | NA | 1.61%–2.94% |
| %0010 | 4 | 1 | 0.403%–0.735% |
| %0010 | ≥ 8 | ≥ 2 | 0.202%–0.368% |
| %0011 | 1 | NA | 0.806%–1.47% |
| %0011 | ≥ 4 | ≥ 1 | 0.202%–0.368% |
| %0100 | 1 | NA | 0.403%–0.735% |
| %0100 | ≥ 2 | ≥ 1 | 0.202%–0.368% |
| %0101–%1001 (max) | ≥ 1 | ≥ 1 | 0.202%–0.368% |

#### 8.4.4.3 Variable-Delay Ring Oscillator

The variable-delay ring oscillator's period is adjustable from 17 to 31 stage delays, in increments of two, based on the upper three DCO stage control bits (DSTG[7:5]). A DSTG[7:5] of %000 corresponds to 17 stage delays; DSTG[7:5] of %111 corresponds to 31 stage delays. Adjusting the DSTG[5] bit has a 6.45 percent to 11.8 percent effect on the output frequency. This also corresponds to the size correction made when the frequency error is greater than ±15 percent. The value of the binary weighted divider does not affect the relative change in output clock period for a given change in DSTG[7:5].

#### *8.4.4.4 Ring Oscillator Fine-Adjust Circuit*

The ring oscillator fine-adjust circuit causes the ring oscillator to effectively operate at non-integer numbers of stage delays by operating at two different points for a variable number of cycles specified by the lower five DCO stage control bits (DSTG[4:0]). For example:

- When DSTG[7:5] is %011, the ring oscillator nominally operates at 23 stage delays.
- When DSTG[4:0] is %00000, the ring will always operate at 23 stage delays.
- When DSTG[4:0] is %00001, the ring will operate at 25 stage delays for one of 32 cycles and at 23 stage delays for 31 of 32 cycles.
- Likewise, when DSTG[4:0] is %11111, the ring operates at 25 stage delays for 31 of 32 cycles and at 23 stage delays for one of 32 cycles.
- When DSTG[7:5] is %111, similar results are achieved by including a variable divide-by-two, so the ring operates at 31 stages for some cycles and at 17 stage delays, with a divide-by-two for an effective 34 stage delays, for the remainder of the cycles.

Adjusting the DSTG[0] bit has a 0.202 percent to 0.368 percent effect on the output clock period. This corresponds to the minimum size correction made by the DLF, and the inherent, long-term quantization error in the output frequency.

### 8.4.5 Switching Internal Clock Frequencies

The frequency of the internal clock (ICLK) may need to be changed for some applications. For example, if the reset condition does not provide the correct frequency, or if the clock is slowed down for a low-power mode (or sped up after a low-power mode), the frequency must be changed by programming the internal clock multiplier factor (N). The frequency of ICLK is N times the frequency of IBASE, which is 307.2 kHz ±25 percent.

Before switching frequencies by changing the N value, the clock monitor must be disabled. This is because when N is changed, the frequency of the low-frequency base clock (IBASE) will change proportionally until the digital loop filter has corrected the error. Since the clock monitor uses IBASE, it could erroneously detect an inactive clock. The clock monitor cannot be re-enabled until the internal clock is stable again (ICGS is set).

The following flow is an example of how to change the clock frequency:

- Verify there is no clock monitor interrupt by reading the CMF bit.
- Turn off the clock monitor.
- If desired, switch to the external clock (see 8.4.1 Switching Clock Sources).
- Change the value of N.
- Switch back to internal (see 8.4.1 Switching Clock Sources), if desired.
- Turn on the clock monitor (see 8.4.2 Enabling the Clock Monitor), if desired.

### 8.4.6 Nominal Frequency Settling Time

Because the clock period of the internal clock (ICLK) is dependent on the digital loop filter outputs (DDIV and DSTG) which cannot change instantaneously, ICLK temporarily will operate at an incorrect clock period when any operating condition changes. This happens whenever the part is reset, the ICG multiply factor (N) is changed, the ICG trim factor (TRIM) is changed, or the internal clock is enabled after inactivity (stop mode or disabled operation). The time that the ICLK takes to adjust to the correct period is known as the settling time.

Settling time depends primarily on how many corrections it takes to change the clock period and the period of each correction. Since the corrections require four periods of the low-frequency base clock ($4*\tau_{IBASE}$), and since ICLK is N (the ICG multiply factor for the desired frequency) times faster than IBASE, each correction takes $4*N*\tau_{ICLK}$. The period of ICLK, however, will vary as the corrections occur.

#### 8.4.6.1 Settling to Within 15 Percent

When the error is greater than 15 percent, the filter takes eight corrections to double or halve the clock period. Due to how the DCO increases or decreases the clock period, the total period of these eight corrections is approximately 11 times the period of the fastest correction. (If the corrections were perfectly linear, the total period would be 11.5 times the minimum period; however, the ring must be slightly nonlinear.) Therefore, the total time it takes to double or halve the clock period is $44*N*\tau_{ICLKFAST}$.

If the clock period needs more than doubled or halved, the same relationship applies, only for each time the clock period needs doubled, the total number of cycles doubles. That is, when transitioning from fast to slow, going from the initial speed to half speed takes $44*N*\tau_{ICLKFAST}$; from half speed to quarter speed takes $88*N*\tau_{ICLKFAST}$; going from quarter speed to eighth speed takes $176*N*\tau_{ICLKFAST}$; and so on. This series can be expressed as $(2^X–1)*44*N*\tau_{ICLKFAST}$, where x is the number of times the speed needs doubled or halved. Since $2^X$ happens to be equal to $\tau_{ICLKSLOW}/\tau_{ICLKFAST}$, the equation reduces to $44*N*(\tau_{ICLKSLOW}–\tau_{ICLKFAST})$.

Note that increasing speed takes much longer than decreasing speed since N is higher. This can be expressed in terms of the initial clock period ($\tau_1$) minus the final clock period ($\tau_2$) as such:

$$\tau_{15} = \text{abs}[44N(\tau_1 - \tau_2)]$$

#### 8.4.6.2 Settling to Within 5 Percent

Once the clock period is within 15 percent of the desired clock period, the filter starts making smaller adjustments. When between 15 percent and 5 percent error, each correction will adjust the clock period between 1.61 percent and 2.94 percent. In this mode, a maximum of eight corrections will be required to get to less than 5 percent error. Since the clock period is relatively close to desired, each correction takes approximately the same period of time, or $4*\tau_{IBASE}$. At this point, the internal clock stable bit (ICGS) will be set and the clock frequency is usable, although the error will be as high as 5 percent. The total time to this point is:

$$\tau_5 = \text{abs}[44N(\tau_1 - \tau_2)] + 32\tau_{IBASE}$$

#### 8.4.6.3 Total Settling Time

Once the clock period is within 5 percent of the desired clock period, the filter starts making minimum adjustments. In this mode, each correction will adjust the frequency between 0.202 percent and 0.368 percent. A maximum of 24 corrections will be required to get to the minimum error. Each correction takes approximately the same period of time, or $4*\tau_{IBASE}$. Added to the corrections for 15 percent to 5 percent, this makes 32 corrections ($128*\tau_{IBASE}$) to get from 15 percent to the minimum error. The total time to the minimum error is:

$$\tau_{tot} = \text{abs}[44N(\tau_1 - \tau_2)] + 128\tau_{IBASE}$$

The equations for $\tau_{15}$, $\tau_5$, and $\tau_{tot}$ are dependent on the actual initial and final clock periods $\tau_1$ and $\tau_2$, not the nominal. This means the variability in the ICLK frequency due to process, temperature, and voltage must be considered. Additionally, other process factors and noise can affect the actual tolerances of the points at which the filter changes modes. This means a worst case adjustment of up to 35 percent (ICLK

clock period tolerance plus 10 percent) must be added. This adjustment can be reduced with trimming. Table 8-3 shows some typical values for settling time.

**Table 8-3. Typical Settling Time Examples**

| $\tau_1$ | $\tau_2$ | N | $\tau_{15}$ | $\tau_5$ | $\tau_{tot}$ |
|---|---|---|---|---|---|
| 1/ (6.45 MHz) | 1/ (25.8 MHz) | 84 | 430 µs | 535 µs | 850 µs |
| 1/ (25.8 MHz) | 1/ (6.45 MHz) | 21 | 107 µs | 212 µs | 525 µs |
| 1/ (25.8 MHz) | 1/ (307.2 kHz) | 1 | 141 µs | 246 µs | 560 µs |
| 1/ (307.2 kHz) | 1/ (25.8 MHz) | 84 | 11.9 ms | 12.0 ms | 12.3 ms |

### 8.4.7 Trimming Frequency on the Internal Clock Generator

The unadjusted frequency of the low-frequency base clock (IBASE), when the comparators in the frequency comparator indicate zero error, will vary as much as ±25 percent due to process, temperature, and voltage dependencies. These dependencies are in the voltage and current references, the offset of the comparators, and the internal capacitor.

The method of changing the unadjusted operating point is by changing the size of the capacitor. This capacitor is designed with 639 equally sized units. Of that number, 384 of these units are always connected. The remaining 255 units are put in by adjusting the ICG trim factor (TRIM). The default value for TRIM is $80, or 128 units, making the default capacitor size 512. Each unit added or removed will adjust the output frequency by about ±0.195 percent of the unadjusted frequency (adding to TRIM will decrease frequency). Therefore, the frequency of IBASE can be changed to ±25 percent of its unadjusted value, which is enough to cancel the process variability mentioned before.

The best way to trim the internal clock is to use the timer to measure the width of an input pulse on an input capture pin (this pulse must be supplied by the application and should be as long or wide as possible). Considering the prescale value of the timer and the theoretical (zero error) frequency of the bus (307.2 kHz *N/4), the error can be calculated. This error, expressed as a percentage, can be divided by 0.195 percent and the resultant factor added or subtracted from TRIM. This process should be repeated to eliminate any residual error.

## 8.5 Low-Power Modes

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

### 8.5.1 Wait Mode

The ICG remains active in wait mode. If enabled, the ICG interrupt to the CPU can bring the MCU out of wait mode.

In some applications, low power-consumption is desired in wait mode and a high-frequency clock is not needed. In these applications, reduce power consumption by either selecting a low-frequency external clock and turn the internal clock generator off or reduce the bus frequency by minimizing the ICG multiplier factor (N) before executing the WAIT instruction.

### 8.5.2 Stop Mode

The value of the oscillator enable in stop (OSCENINSTOP) bit in the CONFIG determines the behavior of the ICG in stop mode. If OSCENINSTOP is low, the ICG is disabled in stop and, upon execution of the STOP instruction, all ICG activity will cease and the output clocks (CGMXCLK, CGMOUT, and TBMCLK) will be held low. Power consumption will be minimal.

If OSCENINSTOP is high, the ICG is enabled in stop and activity will continue. This is useful if the timebase module (TBM) is required to bring the MCU out of stop mode. ICG interrupts will not bring the MCU out of stop mode in this case.

During stop mode, if OSCENINSTOP is low, several functions in the ICG are affected. The stable bits (ECGS and ICGS) are cleared, which will enable the external clock stabilization divider upon recovery. The clock monitor is disabled (CMON = 0) which will also clear the clock monitor interrupt enable (CMIE) and clock monitor flag (CMF) bits. The CS, ICGON, ECGON, N, TRIM, DDIV, and DSTG bits are unaffected.

## 8.6 CONFIG Options

Four CONFIG options affect the functionality of the ICG. These options are:

1. EXTCLKEN, external clock enable
2. EXTXTALEN, external crystal enable
3. EXTSLOW, slow external clock
4. OSCENINSTOP, oscillator enable in stop

All CONFIG options will have a default setting. Refer to Chapter 5 Configuration Registers (CONFIG1, CONFIG2, CONFIG3) on how the CONFIG is used.

### 8.6.1 External Clock Enable (EXTCLKEN)

External clock enable (EXTCLKEN), when set, enables the ECGON bit to be set. ECGON turns on the external clock input path through the PTC4/OSC1 pin. When EXTCLKEN is clear, ECGON cannot be set and PTC4/OSC1 will always perform the PTC4 function.

The default state for this option is clear.

### 8.6.2 External Crystal Enable (EXTXTALEN)

External crystal enable (EXTXTALEN), when set, will enable an amplifier to drive the PTC3/OSC2 pin from the PTC4/OSC1 pin. The amplifier will drive only if the external clock enable (EXTCLKEN) bit and the ECGON bit are also set. If EXTCLKEN or ECGON are clear, PTC3/OSC2 will perform the PTC3 function. When EXTXTALEN is clear, PTC3/OSC2 will always perform the PTC3 function.

EXTXTALEN, when set, also configures the clock monitor to expect an external clock source in the valid range of crystals (30 kHz to 100 kHz or 1 MHz to 32 MHz). When EXTXTALEN is clear, the clock monitor will expect an external clock source in the valid range for externally generated clocks when using the clock monitor (60 Hz to 32 MHz).

EXTXTALEN, when set, also configures the external clock stabilization divider in the clock monitor for a 4096 cycle timeout to allow the proper stabilization time for a crystal. When EXTXTALEN is clear, the stabilization divider is configured to 16 cycles since an external clock source does not need a startup time.

The default state for this option is clear.

### 8.6.3 Slow External Clock (EXTSLOW)

Slow external clock (EXTSLOW), when set, will decrease the drive strength of the oscillator amplifier, enabling low-frequency crystal operation (30 kHz–100 kHz) if properly enabled with the external clock enable (EXTCLKEN) and external crystal enable (EXTXTALEN) bits. When clear, EXTSLOW enables high-frequency crystal operation (1 MHz to 8 MHz or 8 MHz to 32 MHz depending on RNGSEL).

EXTSLOW, when set, also configures the clock monitor to expect an external clock source that is slower than the low-frequency base clock (60 Hz to 307.2 kHz). When EXTSLOW is clear, the clock monitor will expect an external clock faster than the low-frequency base clock (307.2 kHz to 32 MHz).

The default state for this option is clear.

### 8.6.4 Oscillator Enable In Stop (OSCENINSTOP)

Oscillator enable in stop (OSCENINSTOP), when set, will enable the ICG to continue to generate clocks (either CGMXCLK, CGMOUT, or TBMCLK) in stop mode. This function is used to keep the timebase running while the rest of the microcontroller stops. When OSCENINSTOP is clear, all clock generation will cease and CGMXCLK, CGMOUT, and TBMCLK will be forced low during stop mode.

The default state for this option is clear.

## 8.7 Input/Output (I/O) Registers

The ICG contains five registers. These registers are:

1. ICG control register, ICGCR
2. ICG multiplier register, ICGMR
3. ICG trim register, ICGTR
4. ICG DCO divider control register, ICGDVR
5. ICG DCO stage control register, ICGDSR

Several of the bits in these registers have interaction where the state of one bit may force another bit to a particular state or prevent another bit from being set or cleared. A summary of this interaction is shown in Table 8-4.

**Table 8-4. ICG Module Register Bit Interaction Summary**

| Condition | Register Bit Results for Given Condition | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CMIE | CMF | CMON | CS | ICGON | ICGS | ECQON | ECQ | N[6:0] | TRIM[7:0] | DDIV[3:0] | DSTQ[7:0] |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | $15 | $80 | — | — |
| OSCENINSTOP = 0, STOP = 1 | 0 | 0 | 0 | — | — | 0 | — | 0 | — | — | — | — |
| EXTCLKEN = 0 | 0 | 0 | 0 | 0 | 1 | — | 0 | 0 | — | — | uw | uw |
| CMF = 1 | — | (1) | 1 | — | 1 | — | 1 | — | uw | uw | uw | uw |
| CMON = 0 | 0 | 0 | (0) | — | — | — | — | — | — | — | — | — |
| CMON = 1 | — | — | (1) | — | 1 | — | 1 | — | uw | uw | uw | uw |
| CS = 0 | — | — | — | (0) | 1 | — | — | — | — | — | uw | uw |
| CS = 1 | — | — | — | (1) | — | — | 1 | — | — | — | — | — |
| ICGON = 0 | 0 | 0 | 0 | 1 | (0) | 0 | 1 | — | — | — | — | — |
| ICGON = 1 | — | — | — | — | (1) | — | — | — | — | — | uw | uw |
| ICGS = 0 | us | — | us | uc | — | (0) | — | — | — | — | — | — |
| ECGON = 0 | 0 | 0 | 0 | 0 | 1 | — | (0) | 0 | — | — | uw | uw |
| ECGS = 0 | us | — | us | us | — | — | — | (0) | — | — | — | — |
| IOFF = 1 | — | 1* | (1) | 1 | (1) | 0 | (1) | — | uw | uw | uw | uw |
| EOFF = 1 | — | 1* | (1) | 0 | (1) | — | (1) | 0 | uw | uw | uw | uw |
| N = written | (0) | (0) | (0) | — | — | 0* | — | — | — | — | — | — |
| TRIM = written | (0) | (0) | (0) | — | — | 0* | — | — | — | — | — | — |

| | |
|---|---|
| — | Register bit is unaffected by the given condition. |
| 0, 1 | Register bit is forced clear or set (respectively) in the given condition. |
| 0*, 1* | Register bit is temporarily forced clear or set (respectively) in the given condition. |
| (0), (1) | Register bit must be clear or set (respectively) for the given condition to occur. |
| us, uc, uw | Register bit cannot be set, cleared, or written (respectively) in the given condition. |

## 8.7.1 ICG Control Register

The ICG control register (ICGCR) contains the control and status bits for the internal clock generator, external clock generator, and clock monitor as well as the clock select and interrupt enable bits.

Address: $0036

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | CMIE | CMF | CMON | CS | ICGON | ICGS | ECGON | ECGS |
| Write: | | 0(1) | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

1. See CMF bit description for method of clearing CMF bit.

= Unimplemented

**Figure 8-9. ICG Control Register (ICGCR)**

**CMIE — Clock Monitor Interrupt Enable Bit**

This read/write bit enables clock monitor interrupts. An interrupt will occur when both CMIE and CMF are set. CMIE can be set when the CMON bit has been set for at least one cycle. CMIE is forced clear when CMON is clear or during reset.

1 = Clock monitor interrupts enabled
0 = Clock monitor interrupts disabled

**CMF — Clock Monitor Interrupt Flag**

This read-only bit is set when the clock monitor determines that either ICLK or ECLK becomes inactive and the CMON bit is set. This bit is cleared by first reading the bit while it is set, followed by writing the bit low. This bit is forced clear when CMON is clear or during reset.

1 = Either ICLK or ECLK has become inactive.
0 = ICLK and ECLK have not become inactive since the last read of the ICGCR, or the clock monitor is disabled.

**CMON — Clock Monitor On Bit**

This read/write bit enables the clock monitor. CMON can be set when both ICLK and ECLK have been on and stable for at least one bus cycle. (ICGON, ECGON, ICGS, and ECGS are all set.) CMON is forced set when CMF is set, to avoid inadvertent clearing of CMF. CMON is forced clear when either ICGON or ECGON is clear, during stop mode with OSCENINSTOP low, or during reset.

1 = Clock monitor output enabled
0 = Clock monitor output disabled

**CS — Clock Select Bit**

This read/write bit determines which clock will generate the oscillator output clock (CGMXCLK). This bit can be set when ECGON and ECGS have been set for at least one bus cycle and can be cleared when ICGON and ICGS have been set for at least one bus cycle. This bit is forced set when the clock monitor determines the internal clock (ICLK) is inactive or when ICGON is clear. This bit is forced clear when the clock monitor determines that the external clock (ECLK) is inactive, when ECGON is clear, or during reset.

1 = External clock (ECLK) sources CGMXCLK
0 = Internal clock (ICLK) sources CGMXCLK

**ICGON — Internal Clock Generator On Bit**

This read/write bit enables the internal clock generator. ICGON can be cleared when the CS bit has been set and the CMON bit has been clear for at least one bus cycle. ICGON is forced set when the CMON bit is set, the CS bit is clear, or during reset.

1 = Internal clock generator enabled
0 = Internal clock generator disabled

**ICGS — Internal Clock Generator Stable Bit**

This read-only bit indicates when the internal clock generator has determined that the internal clock (ICLK) is within about 5 percent of the desired value. This bit is forced clear when the clock monitor determines the ICLK is inactive, when ICGON is clear, when the ICG multiplier register (ICGMR) is written, when the ICG TRIM register (ICGTR) is written, during stop mode with OSCENINSTOP low, or during reset.

1 = Internal clock is within 5 percent of the desired value.
0 = Internal clock may not be within 5 percent of the desired value.

**ECGON — External Clock Generator On Bit**

This read/write bit enables the external clock generator. ECGON can be cleared when the CS and CMON bits have been clear for at least one bus cycle. ECGON is forced set when the CMON bit or the CS bit is set. ECGON is forced clear during reset.

1 = External clock generator enabled
0 = External clock generator disabled

**ECGS — External Clock Generator Stable Bit**

This read-only bit indicates when at least 4096 external clock (ECLK) cycles have elapsed since the external clock generator was enabled. This is not an assurance of the stability of ECLK but is meant to provide a startup delay. This bit is forced clear when the clock monitor determines ECLK is inactive, when ECGON is clear, during stop mode with OSCENINSTOP low, or during reset.

1 = 4096 ECLK cycles have elapsed since ECGON was set.
0 = External clock is unstable, inactive, or disabled.

## 8.7.2 ICG Multiplier Register

Address: $0037

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: / Write: | | N6 | N5 | N4 | N3 | N2 | N1 | N0 |
| Reset: | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |

= Unimplemented

**Figure 8-10. ICG Multiplier Register (ICGMR)**

**N6:N0 — ICG Multiplier Factor Bits**

These read/write bits change the multiplier used by the internal clock generator. The internal clock (ICLK) will be:

(307.2 kHz ± 25 percent) * N

A value of $00 in this register is interpreted the same as a value of $01. This register cannot be written when the CMON bit is set. Reset sets this factor to $15 (decimal 21) for default frequency of 6.45 MHz ± 25 percent (1.613 MHz ± 25 percent bus).

## 8.7.3 ICG Trim Register

Address: $0038

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: / Write: | TRIM7 | TRIM6 | TRIM5 | TRIM4 | TRIM3 | TRIM2 | TRIM1 | TRIM0 |
| Reset: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-11. ICG Trim Register (ICGTR)**

**TRIM7:TRIM0 — ICG Trim Factor Bits**

These read/write bits change the size of the internal capacitor used by the internal clock generator. By testing the frequency of the internal clock and incrementing or decrementing this factor accordingly, the accuracy of the internal clock can be improved (see 20.11.1 Trimmed Internal Clock Generator Characteristics). Incrementing this register by one decreases the frequency by 0.195 percent of the

unadjusted value. Decrementing this register by one increases the frequency by 0.195 percent. This register cannot be written when the CMON bit is set. Reset sets these bits to $80, centering the range of possible adjustment.

***NOTE***

*Optional storage for 5V and 3V trim values is provided in non-volatile memory at addresses $FF80 and $FF81. See 8.7.4 ICG 5-Volt Trim Value and 8.7.5 ICG 3-Volt Trim Value.*

### 8.7.4 ICG 5-Volt Trim Value

Address: $FF80

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | TRIM7 | TRIM6 | TRIM5 | TRIM4 | TRIM3 | TRIM2 | TRIM1 | TRIM0 |
| Reset: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-12. 5V Internal Oscillator Trim Value (ICGT5V)**

This register provides non-volatile storage for an optional 5V oscillator trim value, which can be transferred by the user software to the ICG Trim Register (ICGTR) when the device comes out of reset. (See 8.7.3 ICG Trim Register.)

### 8.7.5 ICG 3-Volt Trim Value

Address: $FF81

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | TRIM7 | TRIM6 | TRIM5 | TRIM4 | TRIM3 | TRIM2 | TRIM1 | TRIM0 |
| Reset: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 8-13. 3V Internal Oscillator Trim Value (ICGT3V)**

This register provides non-volatile storage for an optional 3V oscillator trim value, which can be transferred by the user software to the ICG Trim Register (ICGTR) when the device comes out of reset. (See 8.7.3 ICG Trim Register.)

### 8.7.6 ICG DCO Divider Register

Address: $0039

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | | | | | DDIV3 | DDIV2 | DDIV1 | DDIV0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | U | U | U | U |

= Unimplemented  U = Unaffected

**Figure 8-14. ICG DCO Divider Control Register (ICGDVR)**

**DDIV3:DDIV0 — ICG DCO Divider Control Bits**

These bits indicate the number of divide-by-twos (DDIV) that follow the digitally controlled oscillator. When ICGON is set, DDIV is controlled by the digital loop filter. The range of valid values for DDIV is from $0 to $9. Values of $A through $F are interpreted the same as $9. Since the DCO is active during reset, reset has no effect on DSTG and the value may vary.

## 8.7.7 ICG DCO Stage Register

Address: $003A

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | DSTG7 | DSTG6 | DSTG5 | DSTG4 | DSTG3 | DSTG2 | DSTG1 | DSTG0 |
| Write: | R | R | R | R | R | R | R | R |
| Reset: | Unaffected by reset | | | | | | | |

R = Reserved

**Figure 8-15. ICG DCO Stage Control Register (ICGDSR)**

**DSTG7:DSTG0 — ICG DCO Stage Control Bits**

These bits indicate the number of stages (above the minimum) in the digitally controlled oscillator. The total number of stages is approximately equal to $1FF, so changing DSTG from $00 to $FF will approximately double the period. Incrementing DSTG will increase the period (decrease the frequency) by 0.202 percent to 0.368 percent (decrementing has the opposite effect). DSTG cannot be written when ICGON is set to prevent inadvertent frequency shifting. When ICGON is set, DSTG is controlled by the digital loop filter. Since the DCO is active during reset, reset has no effect on DSTG and the value may vary.

# Chapter 9
# External Interrupt (IRQ)

## 9.1 Introduction

This section describes the non-maskable external interrupt ($\overline{IRQ}$) input.

## 9.2 Features

Features include:

- Dedicated external interrupt pin ($\overline{IRQ}$)
- Hysteresis buffer
- Programmable edge-only or edge- and level-interrupt sensitivity
- Automatic interrupt acknowledge

## 9.3 Functional Description

A logic 0 applied to the external interrupt pin can latch a central processor unit (CPU) interrupt request. Figure 9-1 shows the structure of the IRQ module.

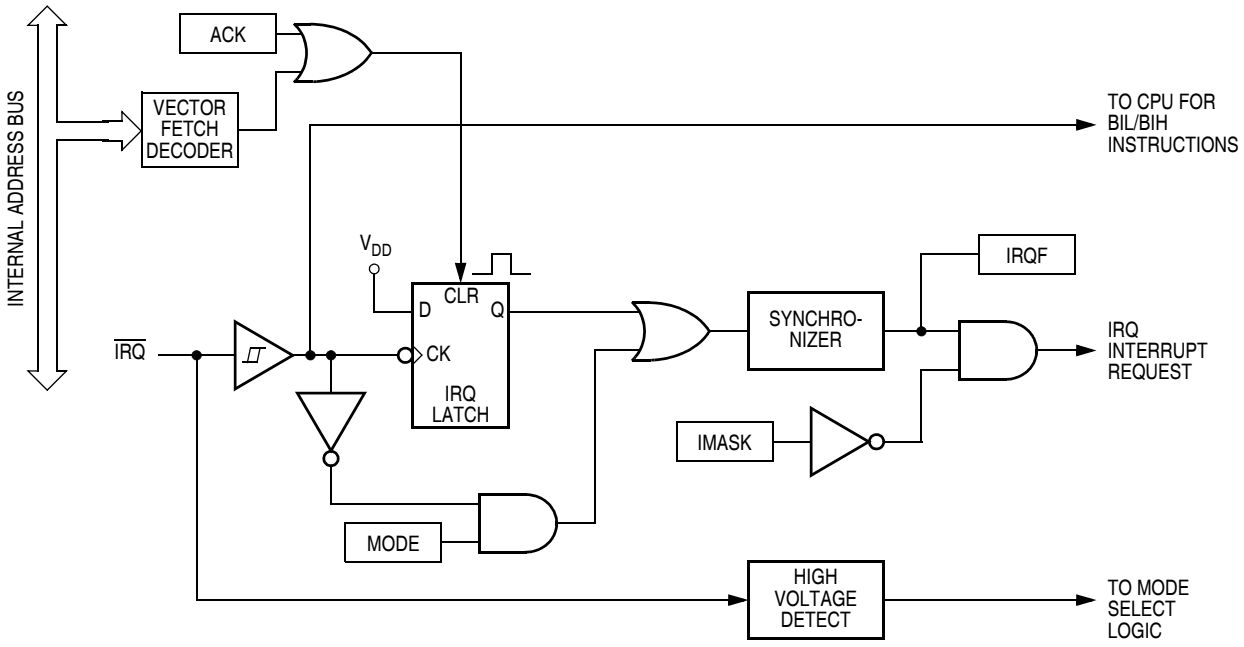



**Figure 9-1. IRQ Block Diagram**

Interrupt signals on the $\overline{\text{IRQ}}$ pin are latched into the IRQ latch. An interrupt latch remains set until one of these 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 (ISCR). Writing a 1 to the ACK bit clears the IRQ latch.
- Reset — A reset automatically clears both interrupt latches.

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

When an interrupt pin is edge-triggered only, the interrupt latch remains set until a vector fetch, software clear, or reset occurs.

When an interrupt pin is both falling-edge and low-level-triggered, the interrupt latch remains set until both of these occur:

- Vector fetch or software clear
- Return of the interrupt pin to logic 1

The vector fetch or software clear may occur before or after the interrupt pin returns to logic 1. 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 ISCR masks all external interrupt requests. A latched interrupt request is not presented to the interrupt priority logic unless the corresponding IMASK bit is clear.

***NOTE***

*The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. See Figure 9-2.*

## 9.4 $\overline{\text{IRQ}}$ Pin

A logic 0 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 these actions must occur to clear the IRQ latch:

- 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 (ISCR). 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 can also prevent spurious interrupts due to noise. Setting ACK does not affect subsequent transitions on the $\overline{\text{IRQ}}$ pin. A falling edge on $\overline{\text{IRQ}}$ 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 logic 1 — As long as the $\overline{\text{IRQ}}$ pin is at logic 0, the IRQ latch remains set.

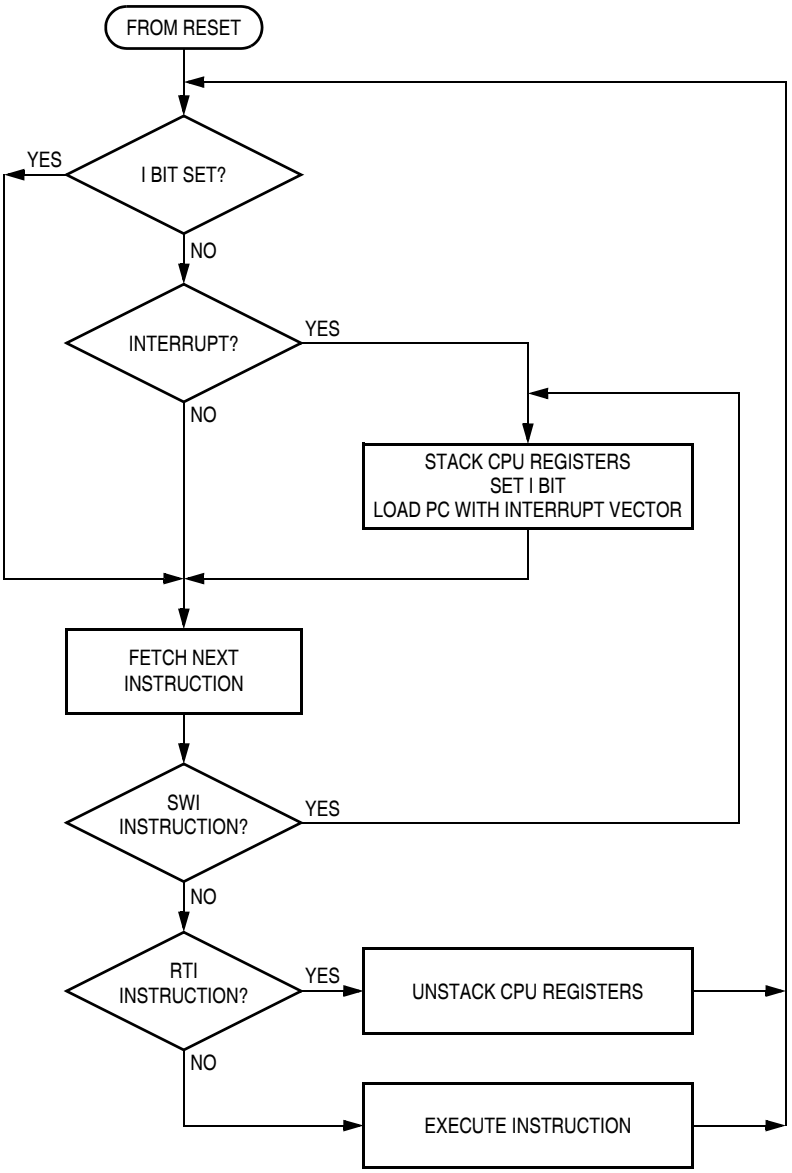



**Figure 9-2. IRQ Interrupt Flowchart**

The vector fetch or software clear and the return of the $\overline{\text{IRQ}}$ pin to logic 1 can occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ}}$ pin is at logic 0. 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 ISCR 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.*

## 9.5 IRQ Module During Break Interrupts

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

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 the latch 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 latch.

## 9.6 IRQ Status and Control Register

The IRQ status and control register (ISCR) controls and monitors operation of the IRQ module. The ISCR has these functions:

- Shows the state of the IRQ interrupt flag
- Clears the IRQ interrupt 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 9-3. IRQ Status and Control Register (ISCR)**

**IRQF — $\overline{\text{IRQ}}$ 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 — $\overline{\text{IRQ}}$ 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 — $\overline{\text{IRQ}}$ Interrupt Mask Bit**

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

1 = $\overline{\text{IRQ}}$ interrupt requests disabled
0 = $\overline{\text{IRQ}}$ interrupt requests enabled

**MODE — $\overline{\text{IRQ}}$ 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 10
# Keyboard Interrupt (KBI) Module

## 10.1 Introduction

The keyboard interrupt module (KBI) provides independently maskable external interrupts.

The KBI shares its pins with general-purpose input/output (I/O) port pins. See Figure 10-1 for port location of these shared pins.

## 10.2 Features

Features of the keyboard interrupt module include:

- Keyboard interrupt pins with separate keyboard interrupt enable bits and one keyboard interrupt mask
- Programmable edge-only or edge and level interrupt sensitivity
- Edge sensitivity programmable for rising or falling edge
- Level sensitivity programmable for high or low level
- Pullup or pulldown device automatically enabled based on the polarity of edge or level detect
- Exit from low-power modes

## 10.3 Functional Description

The keyboard interrupt module controls the enabling/disabling of interrupt functions on the KBI pins. These pins can be enabled/disabled independently of each other.

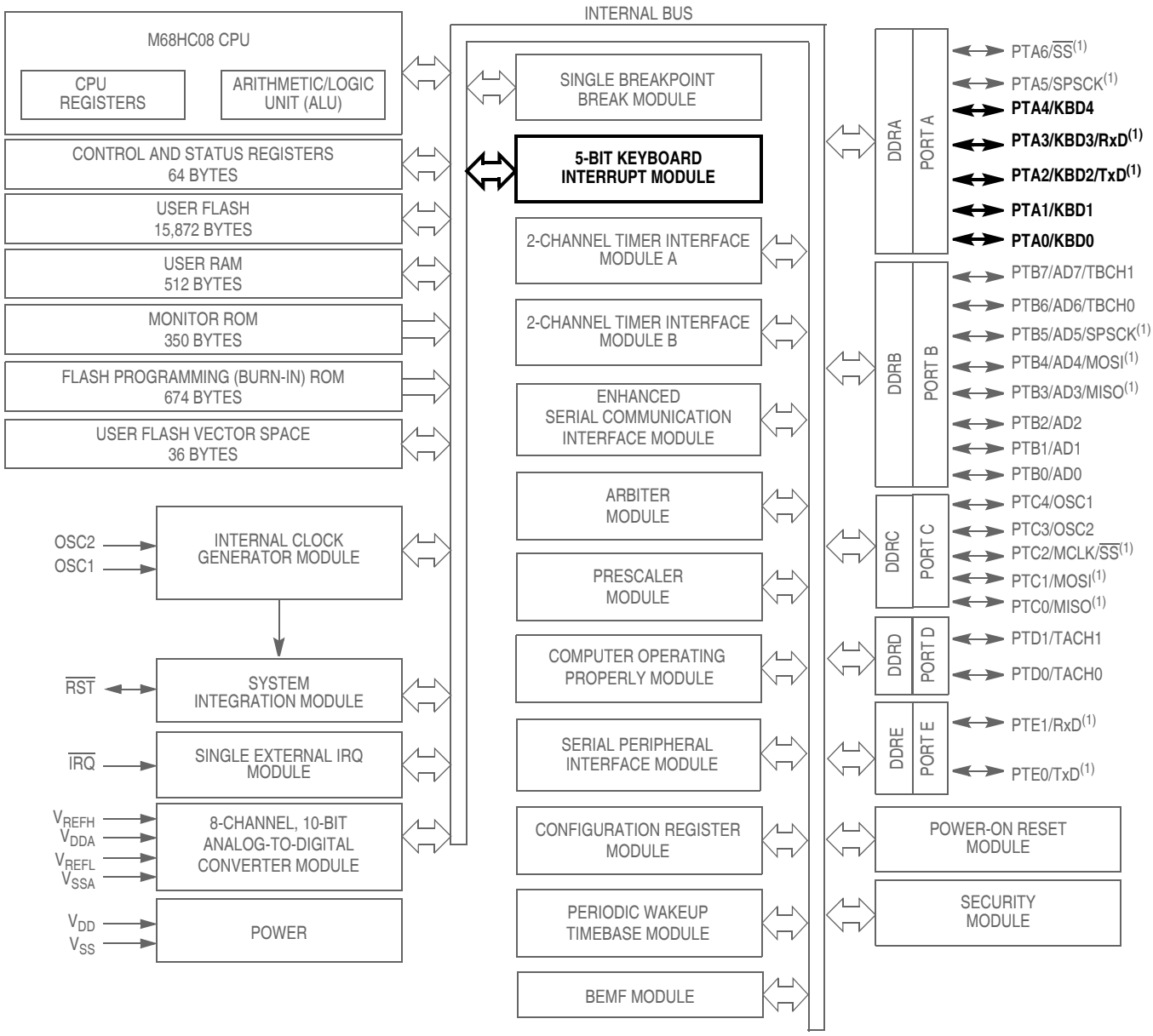
### 10.3.1 Keyboard Operation

Writing to the KBIEx bits in the keyboard interrupt enable register (KBIER) independently enables or disables each KBI pin. The polarity of the keyboard interrupt is controlled using the KBIPx bits in the keyboard interrupt polarity register (KBIPR). Edge-only or edge and level sensitivity is controlled using the MODEK bit in the keyboard status and control register (KBISCR).

Enabling a keyboard interrupt pin also enables its internal pullup or pulldown device based on the polarity enabled. On falling edge or low level detection, a pullup device is configured. On rising edge or high level detection, a pulldown device is configured.

The keyboard interrupt latch is set when one or more enabled keyboard interrupt inputs are asserted.

- If the keyboard interrupt sensitivity is edge-only, for KBIPx = 0, a falling (for KBIPx = 1, a rising) edge on a keyboard interrupt input does not latch an interrupt request if another enabled keyboard pin is already asserted. To prevent losing an interrupt request on one input because another input remains asserted, software can disable the latter input while it is asserted.
- If the keyboard interrupt is edge and level sensitive, an interrupt request is present as long as any enabled keyboard interrupt input is asserted.




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

**Figure 10-1. Block Diagram Highlighting KBI Block and Pins**

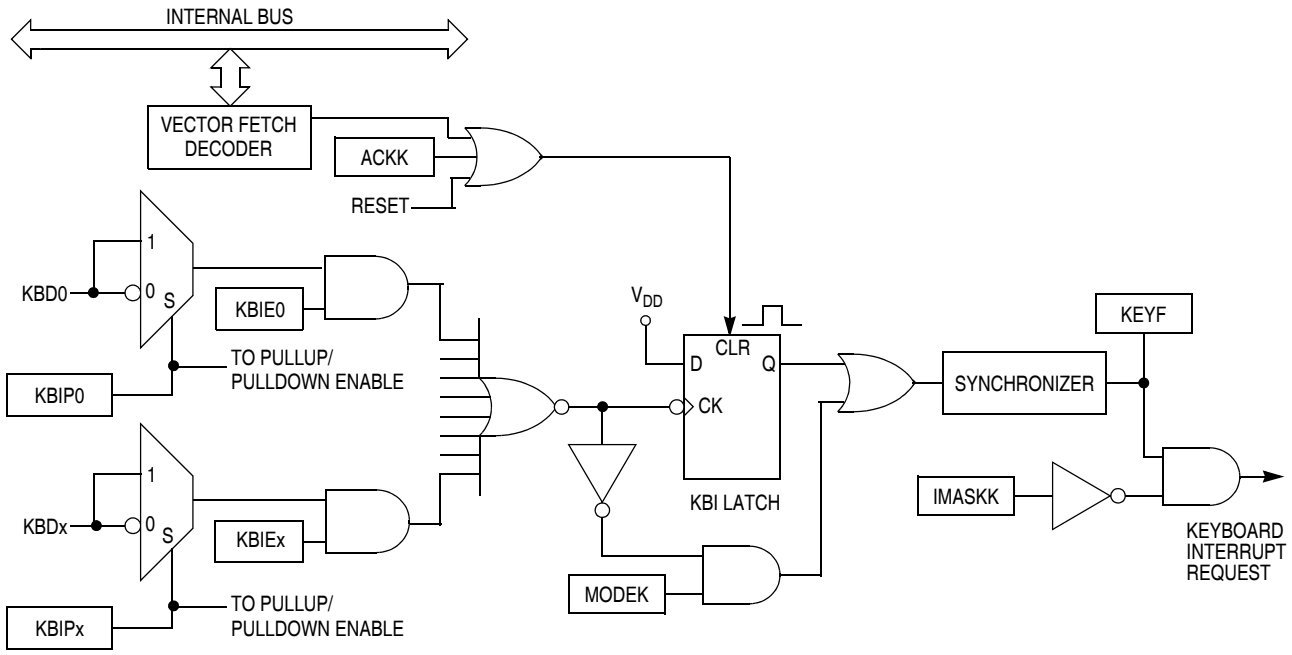



**Figure 10-2. Keyboard Interrupt Block Diagram**

### *10.3.1.1 MODEK = 1*

If the MODEK bit is set, the keyboard interrupt inputs are both edge and level sensitive. The KBIPx bit will determine whether a edge sensitive pin detects rising or falling edges and on level sensitive pins whether the pin detects low or high levels. With MODEK set, both of the following actions must occur to clear a keyboard interrupt request:

- Return of all enabled keyboard interrupt inputs to a deasserted level. As long as any enabled keyboard interrupt pin is asserted, the keyboard interrupt remains active.
- Vector fetch or software clear. A KBI vector fetch generates an interrupt acknowledge signal to clear the KBI latch. Software generates the interrupt acknowledge signal by writing a 1 to ACKK in KBSCR. The ACKK bit is useful in applications that poll the keyboard interrupt inputs and require software to clear the KBI latch. Writing to ACKK 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 inputs. An edge detect that occurs after writing to ACKK latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the KBI vector address.

The KBI vector fetch or software clear and the return of all enabled keyboard interrupt pins to a deasserted level may occur in any order.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt input stays asserted.

### *10.3.1.2 MODEK = 0*

If the MODEK bit is clear, the keyboard interrupt inputs are edge sensitive. The KBIPx bit will determine whether an edge sensitive pin detects rising or falling edges. A KBI vector fetch or software clear immediately clears the KBI latch.

The keyboard flag bit (KEYF) in KBSCR can be read to check for pending interrupts. The KEYF bit is not affected by IMASKK, which makes it useful in applications where polling is preferred.

**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.*

### 10.3.2 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup or pulldown device to pull the pin to its deasserted level. 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 IMASKK in KBSCR.
2. Enable the KBI polarity by setting the appropriate KBIPx bits in KBIPR.
3. Enable the KBI pins by setting the appropriate KBIEx bits in KBIER.
4. Write to ACKK in KBSCR to clear any false interrupts.
5. Clear IMASKK.

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

## 10.4 Interrupts

The following KBI source can generate interrupt requests:

- Keyboard flag (KEYF) — The KEYF bit is set when any enabled KBI pin is asserted based on the KBI mode and pin polarity. The keyboard interrupt mask bit, IMASKK, is used to enable or disable KBI interrupt requests.

## 10.5 Low-Power Modes

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

### 10.5.1 Wait Mode

The KBI module remains active in wait mode. Clearing IMASKK in KBSCR enables keyboard interrupt requests to bring the MCU out of wait mode.

### 10.5.2 Stop Mode

The KBI module remains active in stop mode. Clearing IMASKK in KBSCR enables keyboard interrupt requests to bring the MCU out of stop mode.

## 10.6 KBI 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 break flag control register (BFCR) enables software to clear status bits during the break state. See BFCR in the SIM section of this data sheet.

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 cleared (its default state), software can read and write 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 cleared. After the break, doing the second step clears the status bit.

## 10.7 I/O Signals

The KBI module can share its pins with the general-purpose I/O pins. See Figure 10-1 for the port pins that are shared.

### 10.7.1 KBI Input Pins (KBI7:KBI0)

Each KBI pin is independently programmable as an external interrupt source. KBI pin polarity can be controlled independently. Each KBI pin when enabled will automatically configure the appropriate pullup/pulldown device based on polarity.

## 10.8 Registers

The following registers control and monitor operation of the KBI module:

- KBSCR (keyboard interrupt status and control register)
- KBIER (keyboard interrupt enable register)
- KBIPR (keyboard interrupt polarity register)

### 10.8.1 Keyboard Status and Control Register (KBSCR)

Features of the KBSCR:

- 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 10-3. Keyboard Status and Control Register (KBSCR)**

**Bits 7–4 — Not used**

**KEYF — Keyboard Flag Bit**

This read-only bit is set when a keyboard interrupt is pending.

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

**ACKK — Keyboard Acknowledge Bit**

Writing a 1 to this write-only bit clears the KBI request. ACKK always reads 0.

**IMASKK— Keyboard Interrupt Mask Bit**

Writing a 1 to this read/write bit prevents the output of the KBI latch from generating interrupt requests.

1 = Keyboard interrupt requests disabled
0 = Keyboard interrupt requests enabled

**MODEK — Keyboard Triggering Sensitivity Bit**

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

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

## 10.8.2 Keyboard Interrupt Enable Register (KBIER)

KBIER enables or disables each keyboard interrupt pin.

Address: $001B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 10-4. Keyboard Interrupt Enable Register (KBIER)**

**KBIE4–KBIE0 — Keyboard Interrupt Enable Bits**

Each of these read/write bits enables the corresponding keyboard interrupt pin to latch KBI interrupt requests.

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

## 10.8.3 Keyboard Interrupt Polarity Register (KBIPR)

KBIPR determines the polarity of the enabled keyboard interrupt pin and enables the appropriate pullup or pulldown device.

Address: $000C

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | KBIP4 | KBIP3 | KBIP2 | KBIP1 | KBIP0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 10-5. Keyboard Interrupt Polarity Register (KBIPR)**

**KBIP4–KBIP0 — Keyboard Interrupt Polarity Bits**

Each of these read/write bits selects 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 11
# Low-Voltage Inhibit (LVI) Module

## 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 to the LVI trip voltage.

## 11.2 Features

Features include:

- Programmable LVI reset
- Programmable power consumption
- 3 V or 5 V selectable trip point

## 11.3 Functional Description

Figure 11-1 shows the structure of the LVI module. The LVI is enabled out of reset. The following bits located in the configuration register can alter the default conditions:

- Setting the LVI power disable bit, LVIPWRD, disables the LVI
- Setting the LVI reset disable bit, LVIRSTD, prevents the LVI module from generating a reset.
- Setting the LVI enable in stop mode bit, LVISTOP, enables the LVI to continue monitoring the voltage level on $V_{DD}$ while in stop mode.

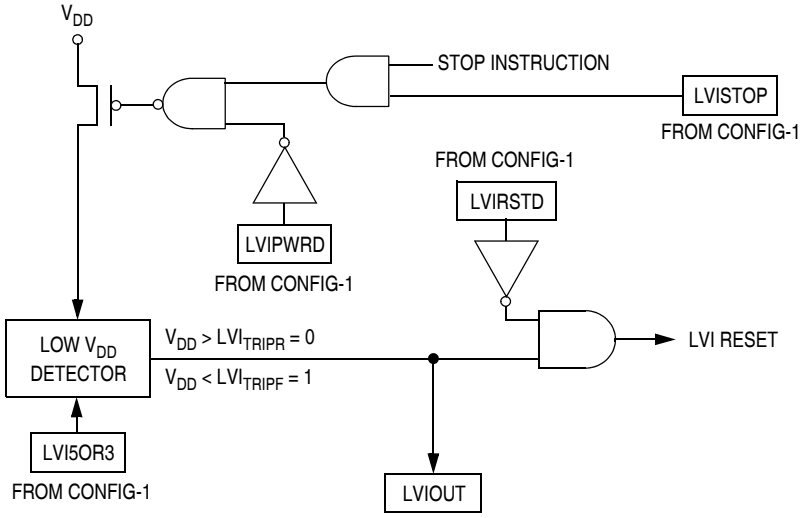



**Figure 11-1. LVI Module Block Diagram**

Once an LVI reset occurs, the MCU remains in reset until $V_{DD}$ rises above a voltage, $LVI_{TRIPR}$. $V_{DD}$ must be above $LVI_{TRIPR}$ for only one CPU cycle to bring the MCU out of reset (see 11.3.2 Forced Reset Operation). 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.

### 11.3.1 Polled LVI Operation

In applications that can operate at $V_{DD}$ levels below the $LVI_{TRIPF}$ level, software can monitor $V_{DD}$ by polling the LVIOUT bit. In the configuration register, the LVIPWRD bit must be at 0 to enable the LVI module, and the LVIRSTD bit must be at 1 to disable LVI resets.

### 11.3.2 Forced Reset Operation

In applications that require $V_{DD}$ to remain above the $LVI_{TRIPF}$ level, enabling LVI resets allows the LVI module to reset the MCU when $V_{DD}$ falls to the $LVI_{TRIPF}$ level. In the configuration register, the LVIPWRD and LVIRSTD bits must be at 0 to enable the LVI module and to enable LVI resets.

### 11.3.3 False Reset Protection

False reset protection is provided by the hysteresis in the LVI trip circuit (refer to Table 11-1). Please refer to 20.5 5V DC Electrical Characteristics for hysteresis value (VHYS) and rising and falling LVI trip values.

### 11.3.4 LVI Status Register

The LVI status register flags $V_{DD}$ voltages below the $LVI_{TRIPF}$ 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 11-2. LVI Status Register (LVISR)**

**LVIOUT — LVI Output Bit**

This read-only flag becomes set when the $V_{DD}$ voltage falls below the $LVI_{TRIPF}$ voltage. (See Table 11-1.) Reset clears the LVIOUT bit.

**Table 11-1. LVIOUT Bit Indication**

| $V_{DD}$ At Level: | LVIOUT |
|---|---|
| $V_{DD} > LVI_{TRIPR}$ | 0 |
| $V_{DD} < LVI_{TRIPF}$ | 1 |
| $LVI_{TRIPF} < V_{DD} < LVI_{TRIPR}$ | Previous Value |

## 11.4 LVI Interrupts

The LVI module does not generate interrupt requests.

## 11.5 Low-Power Modes

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

### 11.5.1 Wait Mode

With the LVIPWRD bit in the configuration register programmed to 0, the LVI module is active after a WAIT instruction.

With the LVIRSTD bit in the configuration register programmed to 0, the LVI module can generate a reset and bring the MCU out of wait mode.

### 11.5.2 Stop Mode

With the LVISTOP bit in the configuration register programmed to a 1 and the LVIPWRD bit programmed to 0, the LVI module will be active after a STOP instruction.

With the LVIPWRD bit in the configuration register programmed to 1 and the LVISTOP bit at a 0, the LVI module will be inactive after a STOP instruction.

# Chapter 12
# Input/Output (I/O) Ports (PORTS)

## 12.1 Introduction

Twenty-four bidirectional input/output (I/O) pins form five parallel ports. All I/O pins are programmable as inputs or outputs.

***NOTE***

*Connect any unused I/O pins to an appropriate logic level, either $V_{DD}$ or $V_{SS}$. Although the I/O ports do not require termination for proper operation, termination reduces excess current consumption and the possibility of electrostatic damage.*

## 12.2 Port A

Port A is a 7-bit general-purpose bidirectional I/O port that shares pin functions with the serial peripheral interface (SPI) and keyboard (KBD) modules.

### 12.2.1 Port A Data Register

The port A data register contains a data latch for each of the seven port A pins.

| Address: | $0000 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | 0 | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |
| Alternative Function: | | $\overline{SS}$ | SPSCK | KBD4 | KBD3 | KBD2 | KBD1 | KBD0 |
| Alternative Function: | | | | | RxD | TxD | | |

= Unimplemented

**Figure 12-1. Port A Data Register (PTA)**

**PTA[6:0] — 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.

### 12.2.2 Data Direction Register A

Data direction register A 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: | 0 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 12-2. Data Direction Register A (DDRA)**

**DDRA[6:0] — Data Direction Register A Bits**

These read/write bits control port A data direction. Reset clears DDRA[6:0], 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 12-3 shows the port A I/O logic.

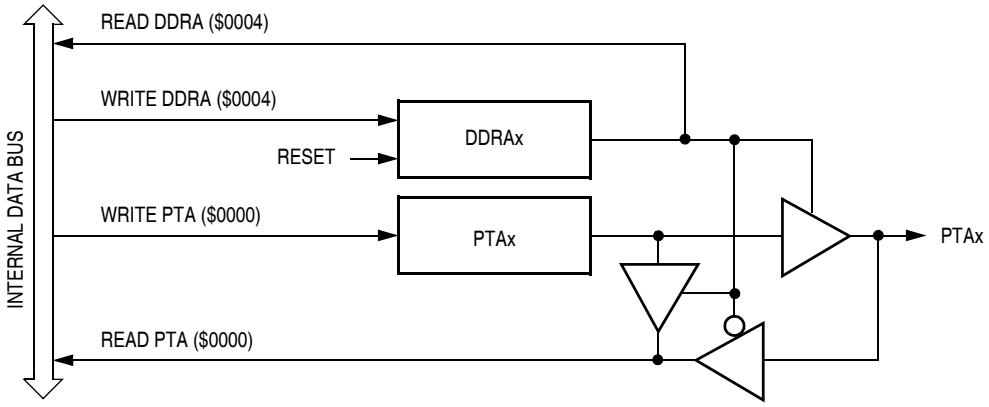



**Figure 12-3. Port A I/O Circuit**

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 12-1 summarizes the operation of the port A pins.

**Table 12-1. Port A Pin Functions**

| DDRA Bit | PTA Bit | I/O Pin Mode | Accesses to DDRA | Accesses to PTA | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, Hi-Z | DDRA[6:0] | Pin | PTA[6:0](1) |
| 1 | X | Output | DDRA[6:0] | PTA[6:0] | PTA[6:0] |

X = don't care
Hi-Z = high impedance
1. Writing affects data register, but does not affect input.

## 12.3 Port B

Port B is an 8-bit special-function port that shares all of its pins with the analog-to-digital converter (ADC) and some pin functions with TIMB.

Port B is designed so that the ADC function will take priority over the timer functionality on PTB6 and PTB7. If the ADC is selected for a conversion on a previously enabled timer pin, the port pin will be connected to the ADC and disconnected from the timer. If both the timer input capture and ADC functions are being used on the same port pin, it is recommended that the timer channel be disabled before the pin is enabled as an ADC input to avoid glitches. If both the timer output compare (or PWM) and ADC functions are being used on the same port pin, it is recommended that the timer channel be disabled before the pin is enabled as an ADC input.

### 12.3.1 Port B Data Register

The port B data register contains a data latch for each of the eight port B 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 | | | | | | | |
| Alternative Function: | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 |
| Alternative Function: | TBCH1 | TBCH0 | SPSCK | MOSI | MISO | | | |

**Figure 12-4. Port B Data Register (PTB)**

**PTB[7:0] — 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.

### 12.3.2 Data Direction Register B

Data direction register B 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 12-5. Data Direction Register B (DDRB)**

**DDRB[7:0] — Data Direction Register B Bits**

These read/write bits control port B data direction. Reset clears DDRB[7:0], 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 12-6 shows the port B I/O logic.

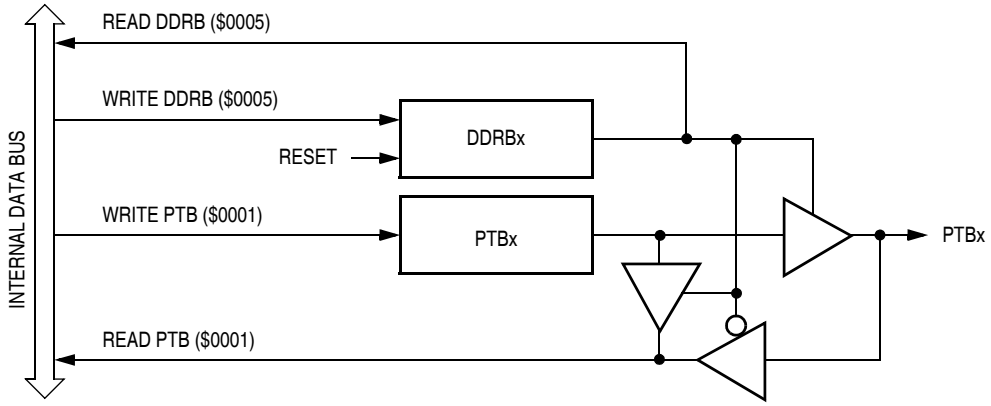



**Figure 12-6. Port B I/O Circuit**

When DDRBx is a 1, reading address $0001 reads the PTBx data latch. When 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 12-2 summarizes the operation of the port B pins.

**Table 12-2. Port B Pin Functions**

| DDRB Bit | PTB Bit | I/O Pin Mode | Accesses to DDRB | Accesses to PTB | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, Hi-Z | DDRB[7:0] | Pin | PTB[7:0](1) |
| 1 | X | Output | DDRB[7:0] | PTB[7:0] | PTB[7:0] |

X = don't care
Hi-Z = high impedance
1. Writing affects data register, but does not affect input.

## 12.4 Port C

Port C is an 5-bit general-purpose bidirectional I/O port that shares pin functions with the internal clock generator (ICG) and serial peripheral interface (SPI) modules.

### 12.4.1 Port C Data Register

The port C data register contains a data latch for each of the five port C pins.

Address: $0002

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | PTC4 | PTC3 | PTC2 | PTC1 | PTC0 |
| Write: | | | | | | | | |
| Reset: | Unaffected by reset | | | | | | | |
| Alternative Function: | | | | OSC1 | OSC2 | MCLK | MOSI | MISO |
| Alternative Function: | | | | | | $\overline{SS}$ | | |

= Unimplemented

**Figure 12-7. Port C Data Register (PTC)**

**PTC[4:0] — 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.

## 12.4.2 Data Direction Register C

Data direction register C 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: | MCLKEN | 0 | 0 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 12-8. Data Direction Register C (DDRC)**

**MCLKEN — MCLK Enable Bit**

This read/write bit enables MCLK, a bus clock frequency clock signal, to be an output signal on PTC2. If MCLK is enabled, PTC2 is under the control of MCLKEN. Reset clears this bit.

1 = MCLK output enabled
0 = MCLK output disabled

**DDRC[4:0] — Data Direction Register C Bits**

These read/write bits control port C data direction. Reset clears DDRC[4:0] and MCLKEN, 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 12-9 shows the port C I/O logic.

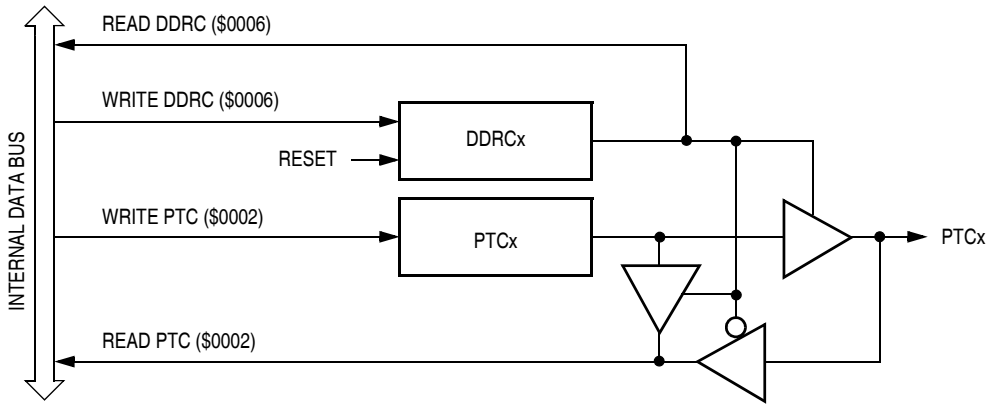



**Figure 12-9. Port C I/O Circuit**

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 12-3 summarizes the operation of the port C pins.

**Table 12-3. Port C Pin Functions**

| DDRC Bit | PTC Bit | I/O Pin Mode | Accesses to DDRC | Accesses to PTC | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | 2 | Input, Hi-Z | DDRC[7] | Pin | PTC2 |
| 1 | 2 | Output | DDRC[7] | 0 | — |
| 0 | X | Input, Hi-Z | DDRC[4:0] | Pin | PTC[4:0](1) |
| 1 | X | Output | DDRC[4:0] | PTC[4:0] | PTC[4:0] |

X = don't care
Hi-Z = high impedance
1. Writing affects data register, but does not affect input.

## 12.5 Port D

Port D is a 2-bit special function port that shares its pins with the timer interface module (TIMA).

### 12.5.1 Port D Data Register

The port D data register contains a data latch for each of the two port D pins.

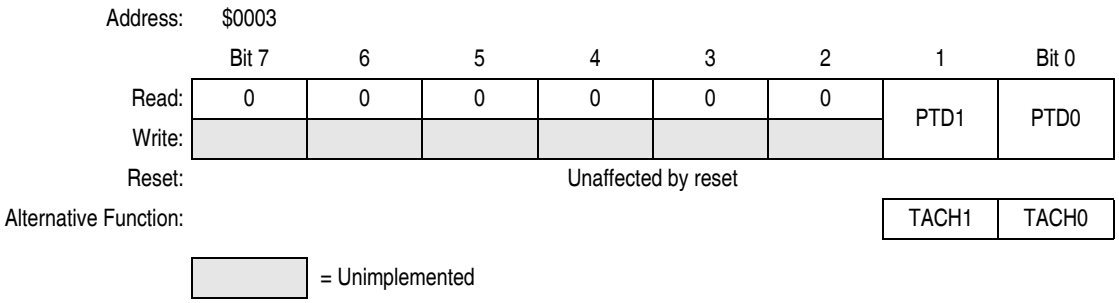



**Figure 12-10. Port D Data Register (PTD)**

**PTD[1:0] — Port D Data Bits**

PTD[1:0] are read/write, software programmable bits. Data direction of each port D pin is under the control of the corresponding bit in data direction register D.

### 12.5.2 Data Direction Register D

Data direction register D 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: | 0 | 0 | 0 | 0 | 0 | 0 | DDRD1 | DDRD0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 12-11. Data Direction Register D (DDRD)**

**DDRD[1:0] — Data Direction Register D Bits**

These read/write bits control port D data direction. Reset clears DDRD[1:0], 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 12-12 shows the port D I/O logic.

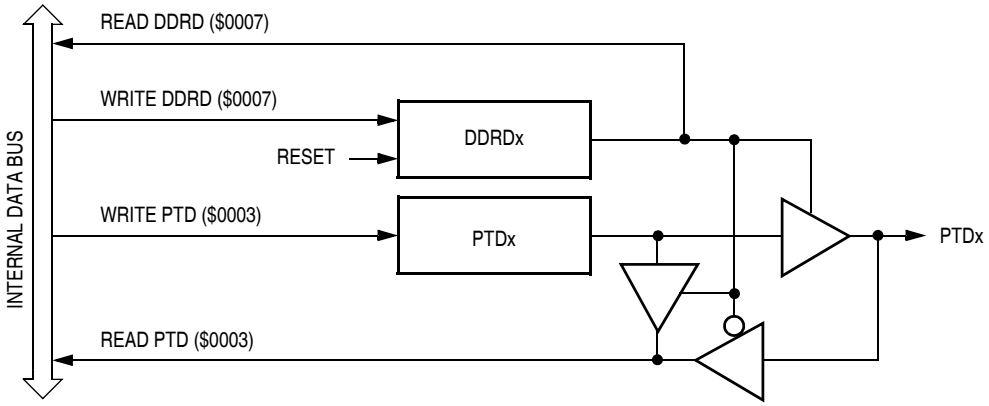



**Figure 12-12. Port D I/O Circuit**

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 12-4 summarizes the operation of the port D pins.

**Table 12-4. Port D Pin Functions**

| DDRD Bit | PTD Bit | I/O Pin Mode | Accesses to DDRD | Accesses to PTD | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, Hi-Z | DDRD[1:0] | Pin | PTD[1:0](1) |
| 1 | X | Output | DDRD[1:0] | PTD[1:0] | PTD[1:0] |

X = don't care
Hi-Z = high impedance
1. Writing affects data register, but does not affect input.

## 12.6 Port E

Port E is a 2-bit special function port that shares its pins with the enhanced serial communications interface module (ESCI).

### 12.6.1 Port E Data Register

The port E data register contains a data latch for each of the port E pins.

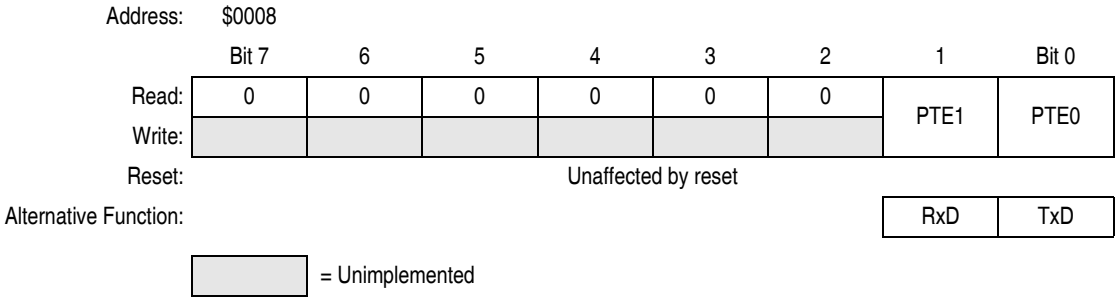



**Figure 12-13. Port E Data Register (PTE)**

**PTE[1:0] — 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 PTE[1:0].

### 12.6.2 Data Direction Register E

Data direction register E 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.

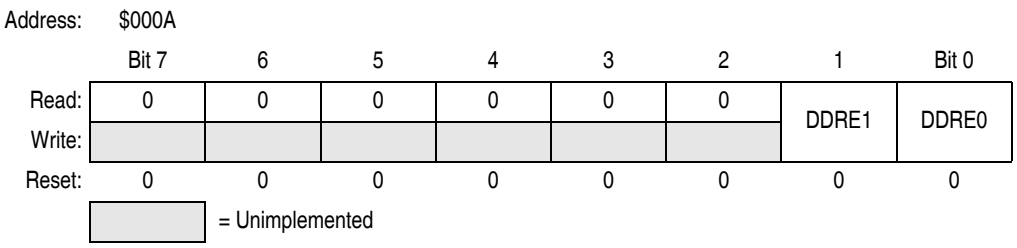



**Figure 12-14. Data Direction Register E (DDRE)**

**DDRE[1:0] — Data Direction Register E Bits**

These read/write bits control port E data direction. Reset clears DDRE[1:0], 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 12-15 shows the port E I/O logic.

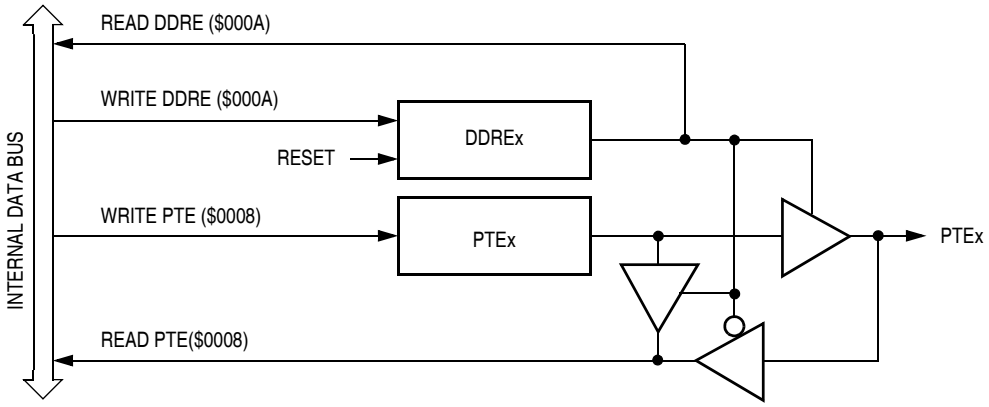



**Figure 12-15. Port E I/O Circuit**

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 12-5 summarizes the operation of the port E pins.

**Table 12-5. Port E Pin Functions**

| DDRE Bit | PTE Bit | I/O Pin Mode | Accesses to DDRE | Accesses to PTE | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, Hi-Z | DDRE[1:0] | Pin | PTE[1:0](1) |
| 1 | X | Output | DDRE[1:0] | PTE[1:0] | PTE[1:0] |

X = don't care
Hi-Z = high impedance
1. Writing affects data register, but does not affect input.

# Chapter 13
# Enhanced Serial Communications Interface (ESCI) Module

## 13.1 Introduction

The enhanced serial communications interface (ESCI) module allows asynchronous communications with peripheral devices and other microcontroller units (MCU).

## 13.2 Features

Features include:

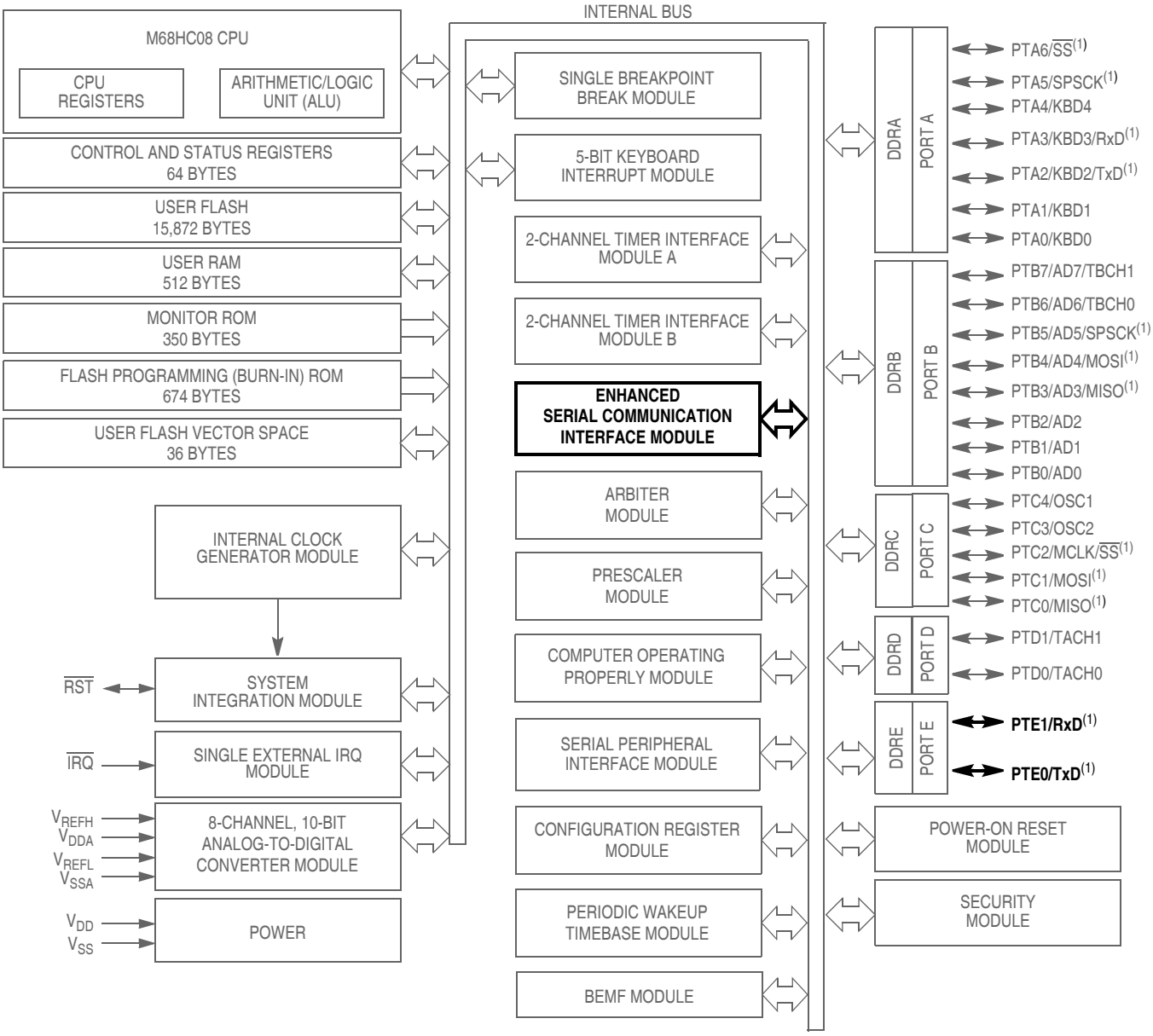
- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- Programmable baud rates
- Programmable 8-bit or 9-bit character length
- Separately enabled transmitter and receiver
- Separate receiver and transmitter central processor unit (CPU) interrupt requests
- Programmable transmitter 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

## 13.3 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 13-1 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.




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

**Figure 13-1. Block Diagram Highlighting ESCI Block and Pins**

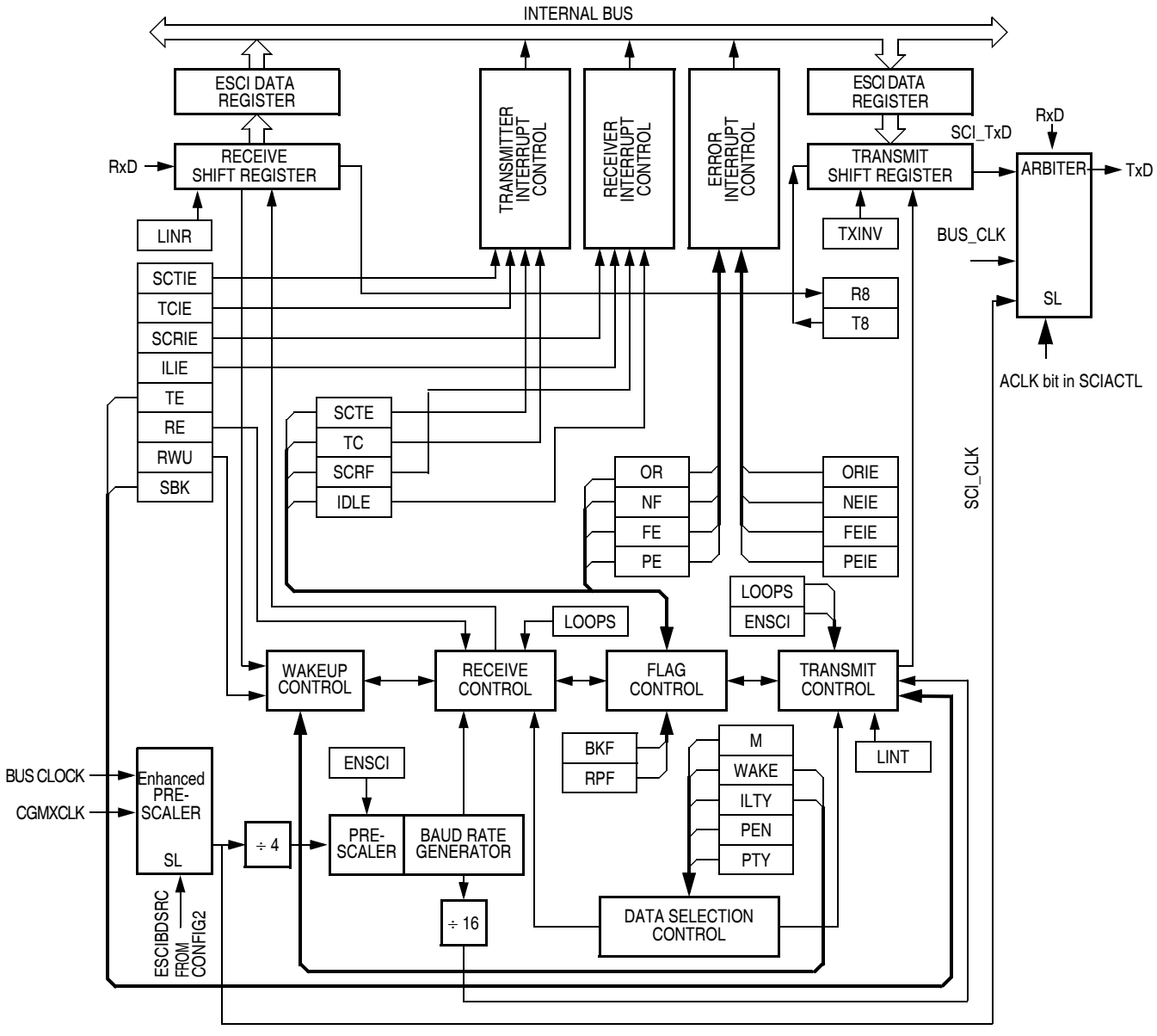
**Table 13-1. Pin Name Conventions**

| ESCI Generic Pin Name | RxD | TxD |
|---|---|---|
| Full Pin Name | PTE1/RxD | PTE0/TxD |
| Alternative Pin | PTA3 | PTA2 |

## 13.4 Functional Description

Figure 13-2 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, ESCIBDSRC, of the CONFIG2 register ($001E).




SL=1 -> SCI_CLK = BUSCLK
SL=0 -> SCI_CLK = CGMXCLK (4x BUSCLK)

**Figure 13-2. ESCI Module Block Diagram**

### 13.4.1 Data Format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in Figure 13-3.

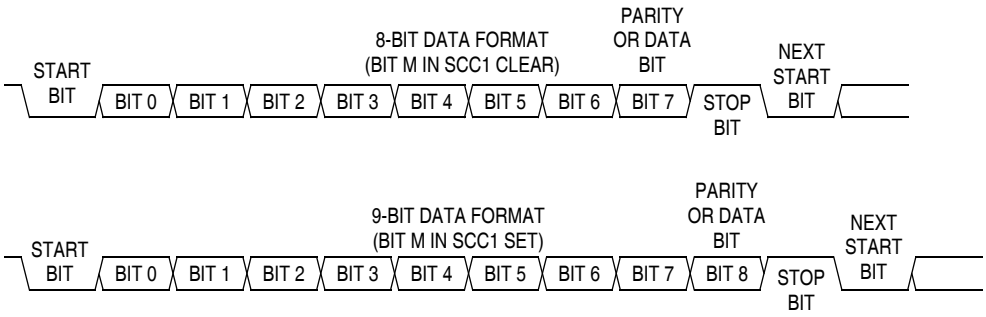



**Figure 13-3. SCI Data Formats**

### 13.4.2 Transmitter

Figure 13-4 shows the structure of the SCI transmitter. The baud rate clock source for the ESCI can be selected via the configuration bit, ESCIBDSRC.

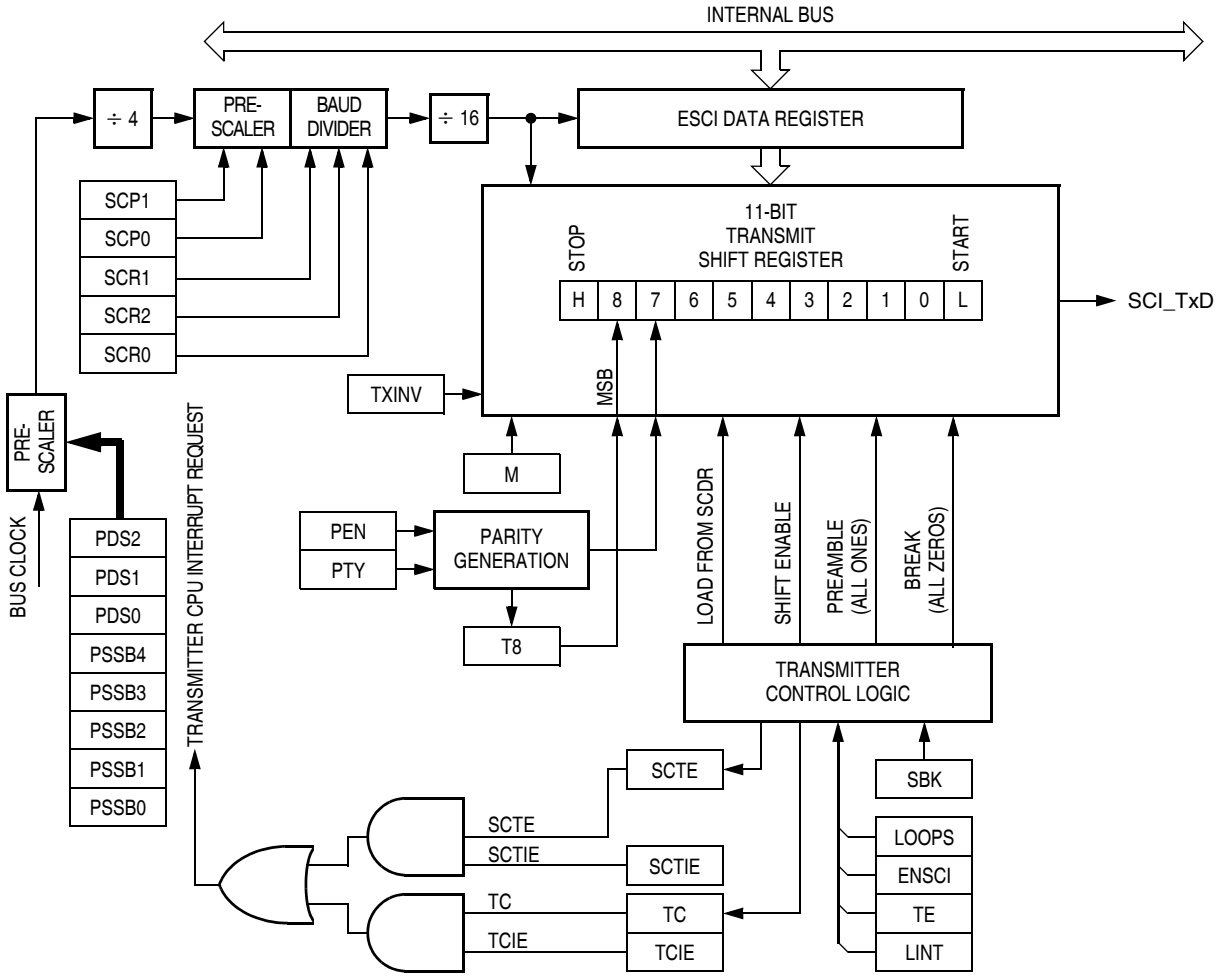



**Figure 13-4. ESCI Transmitter**

#### 13.4.2.1 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).

#### 13.4.2.2 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, logic 1. 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.

#### 13.4.2.3 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 consecutive 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

#### *13.4.2.4 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.

#### *13.4.2.5 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 13.8.1 ESCI Control Register 1.

#### *13.4.2.6 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.

### 13.4.3 Receiver

Figure 13-5 shows the structure of the ESCI receiver.

#### *13.4.3.1 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).

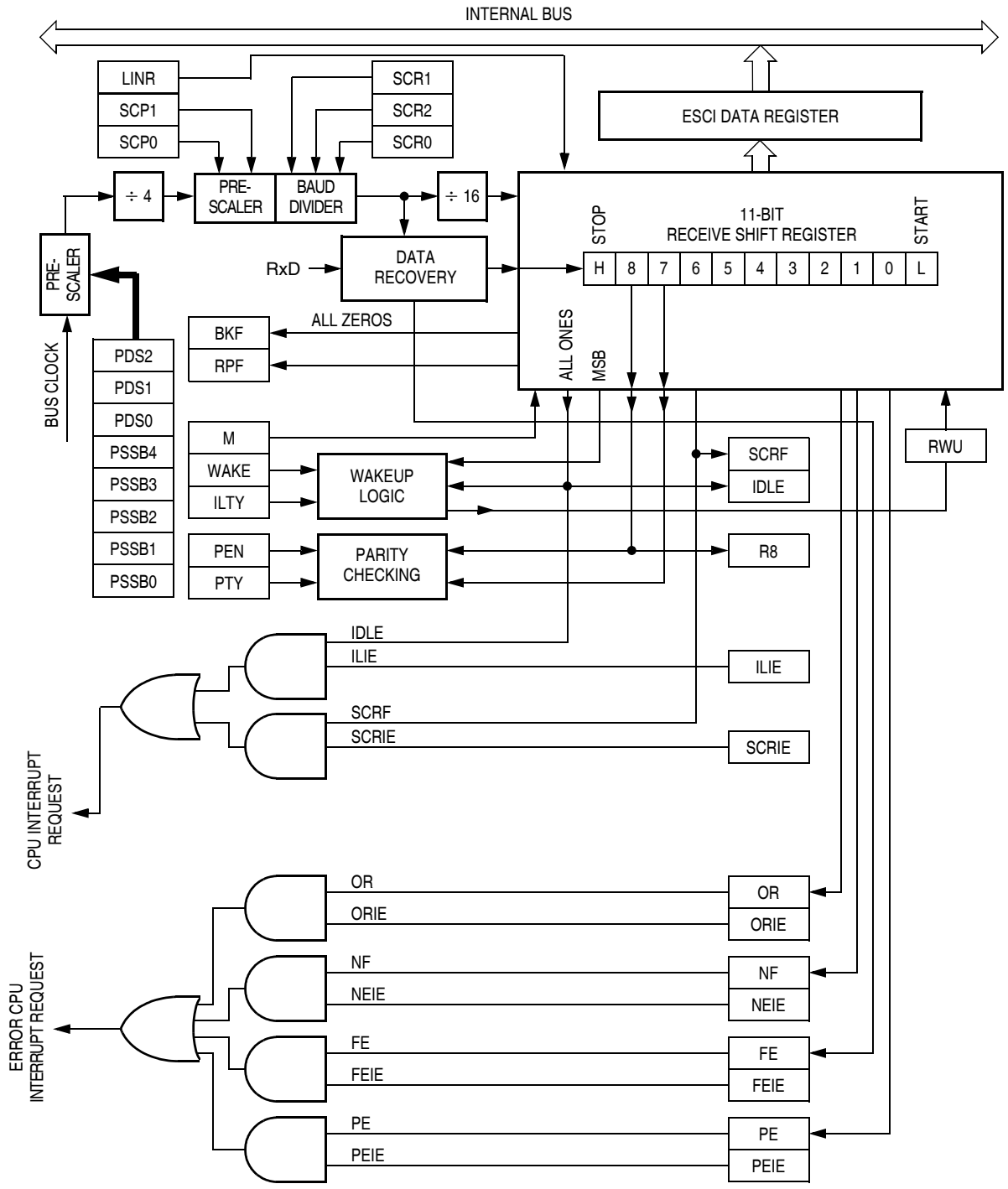



**Figure 13-5. ESCI Receiver Block Diagram**

### *13.4.3.2 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.

#### *13.4.3.3 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 13-6):

- 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.

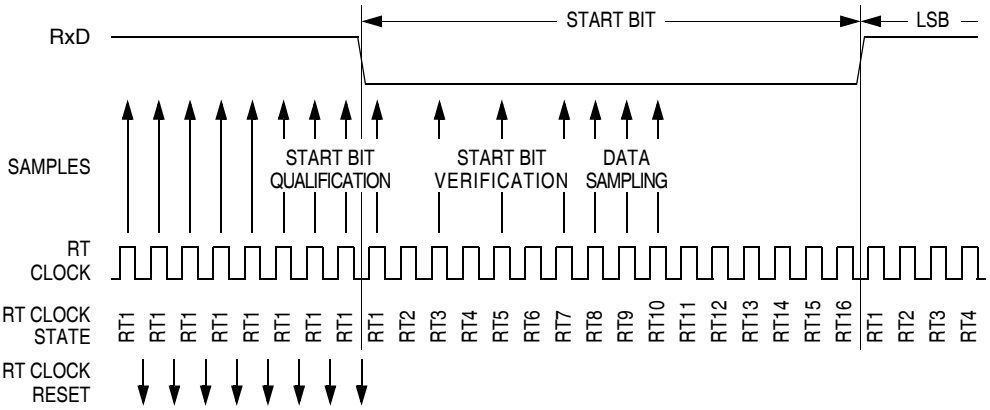



**Figure 13-6. Receiver Data Sampling**

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 13-2 summarizes the results of the start bit verification samples.

**Table 13-2. 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 13-3 summarizes the results of the data bit samples.

**Table 13-3. 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 13-4 summarizes the results of the stop bit samples.

**Table 13-4. 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 |

#### *13.4.3.4 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.

#### *13.4.3.5 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 13-7 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.

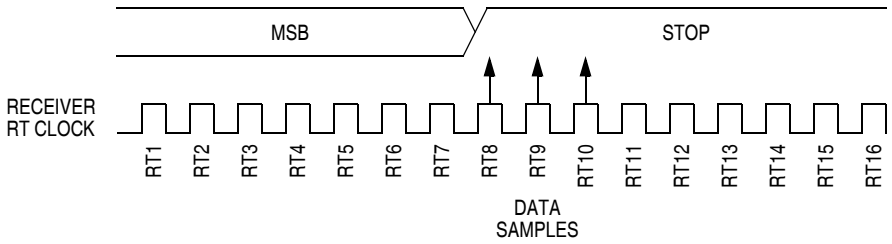



**Figure 13-7. Slow Data**

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 13-7, 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 13-7, 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 13-8 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.

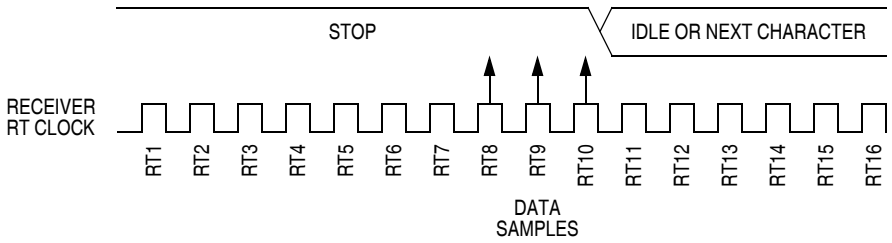



**Figure 13-8. Fast Data**

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 13-8, 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 13-8, 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:

$$\left|\frac{170 - 176}{170}\right| \times 100 = 3.53\%.$$

#### *13.4.3.6 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.*

#### 13.4.3.7 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.

#### 13.4.3.8 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.

## 13.5 Low-Power Modes

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

### 13.5.1 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.

### 13.5.2 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.

## 13.6 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 19.2 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.

## 13.7 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

### 13.7.1 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).

### 13.7.2 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).

## 13.8 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

### 13.8.1 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: $0010

| | 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 13-9. 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 13-5).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 13-5. 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 13-5). When enabled, the parity function inserts a parity bit in the MSB position (see Table 13-3). 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 13-5). 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.*

### 13.8.2 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: $0011

| | 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 13-10. 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 consecutive 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 (1). 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.*

### 13.8.3 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: $0012

| | 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 13-11. 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 error 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

### 13.8.4 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: $0013

| | 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 13-12. 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 13-13 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.

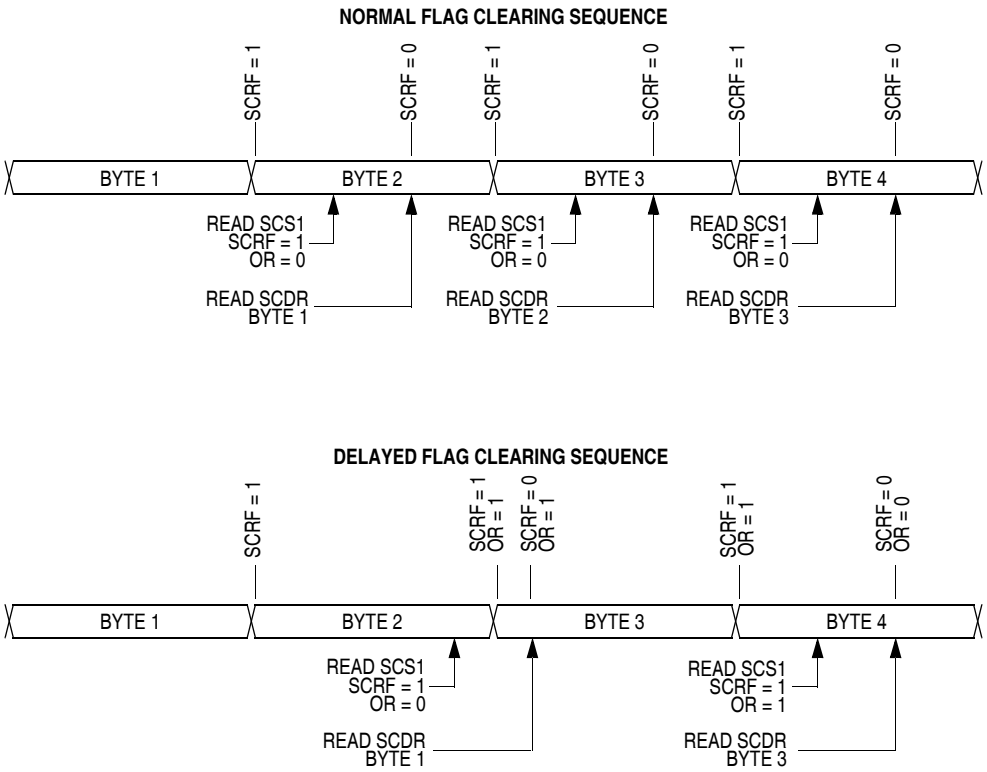



**Figure 13-13. Flag Clearing Sequence**

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

### 13.8.5 ESCI Status Register 2

ESCI status register 2 (SCS2) contains flags to signal these conditions:

- Break character detected
- Incoming data

Address: $0014

| | 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 13-14. 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

### 13.8.6 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: $0015

| | 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 13-15. 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.*

### 13.8.7 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: $0016

| | 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 13-16. ESCI Baud Rate Register (SCBR)**

**LINT — LIN Break Symbol Transmit Enable**

This read/write bit selects the enhanced ESCI features for master nodes in the local interconnect network (LIN) protocol (version 1.2) as shown in Table 13-6. Reset clears LINT.

### LINR — LIN Receiver Bit

This read/write bit selects the enhanced ESCI features for slave nodes in the local interconnect network (LIN) protocol as shown in Table 13-6. Reset clears LINR.

**Table 13-6. ESCI LIN Slave Node 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 13-7. Reset clears SCP1 and SCP0.

**Table 13-7. 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 13-8. Reset clears SCR2–SCR0.

**Table 13-8. 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 |

### 13.8.8 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: $0017

| | 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 13-17. ESCI Prescaler Register (SCPSC)**

**PDS2–PDS0 — Prescaler Divisor Select Bits**

These read/write bits select the prescaler divisor as shown in Table 13-9. 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 13-9. 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 13-10. Reset clears PSSB4–PSSB0.

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 ESCIBDSRC in the CONFIG2 register)

BPD = Baud rate register prescaler divisor

BD = Baud rate divisor

PD = Prescaler divisor

PDFA = Prescaler divisor fine adjust

## Table 13-10. 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 |
| 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 |

Table 13-11 shows the ESCI baud rates that can be generated with a 4.9152-MHz clock frequency.

**Table 13-11. 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 (ESCI Clock = 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 |

## 13.9 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).

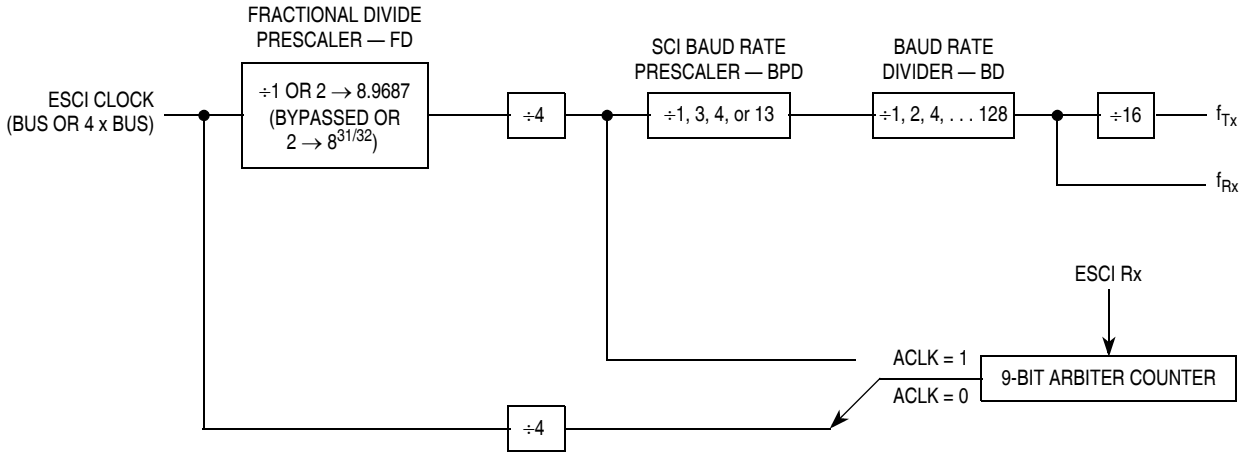



**Figure 13-18. ESCI Arbiter Counter Clock Selection**

### 13.9.1 ESCI Arbiter Control Register

Address: $0018

| | 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 13-19. ESCI Arbiter Control Register (SCIACTL)**

**AM1 and AM0 — Arbiter Mode Select Bits**

As shown in Table 13-12, these read/write bits select the mode of the arbiter module. Reset clears AM1 and AM0.

**Table 13-12. 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 quarter 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 ESCIBDSRC 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.

## 13.9.2 ESCI Arbiter Data Register

Address: $0019

| | 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 13-20. 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.

### 13.9.3 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 one quarter of the ESCI input clock (= bus or CGMXCLK, see Figure 5-2. Configuration Register 2 (CONFIG2) for a description of the ESCIBDSRC bit). 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 13-21.
2. ACLK = 1 — The counter is clocked with one quarter of the ESCI input clock divided by the ESCI prescaler. The counter is started when a logic 0 is detected on RxD (see Figure 13-22). A logic 0 on RxD on enabling the bit time measurement with ACLK = 1 leads to immediate start of the counter (see Figure 13-23). The counter will be stopped on the next rising edge of RxD. This mode is used to measure the length of a received break.

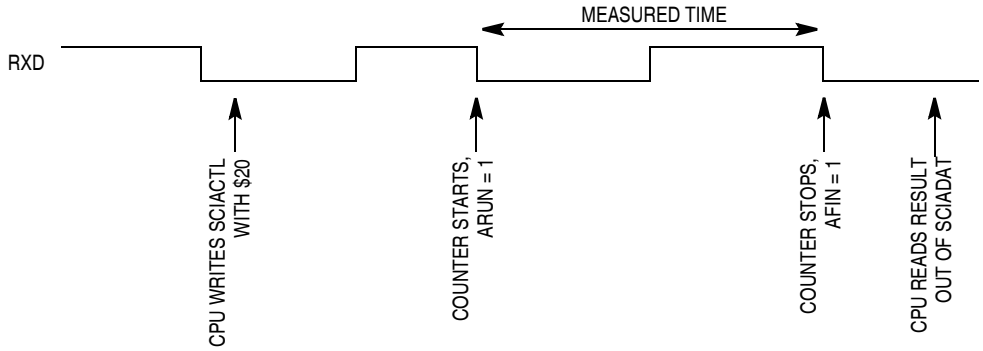



**Figure 13-21. Bit Time Measurement with ACLK = 0**

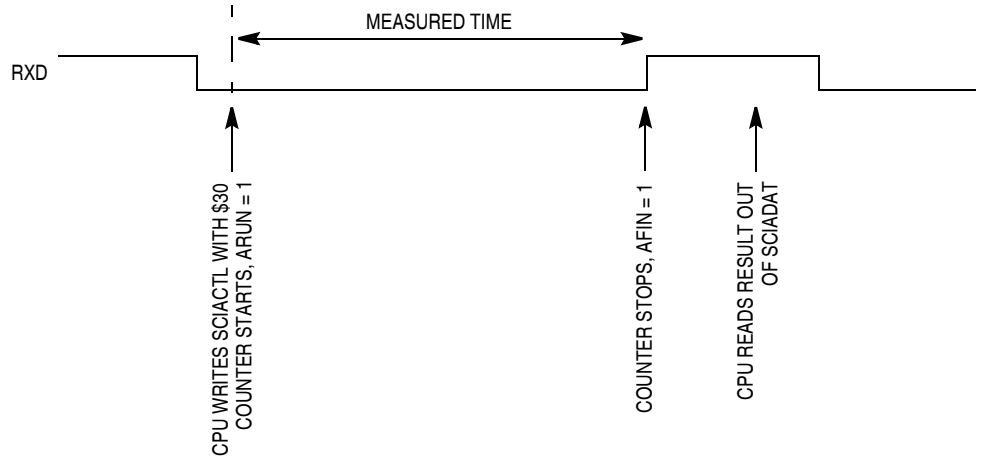



**Figure 13-22. Bit Time Measurement with ACLK = 1, Scenario A**

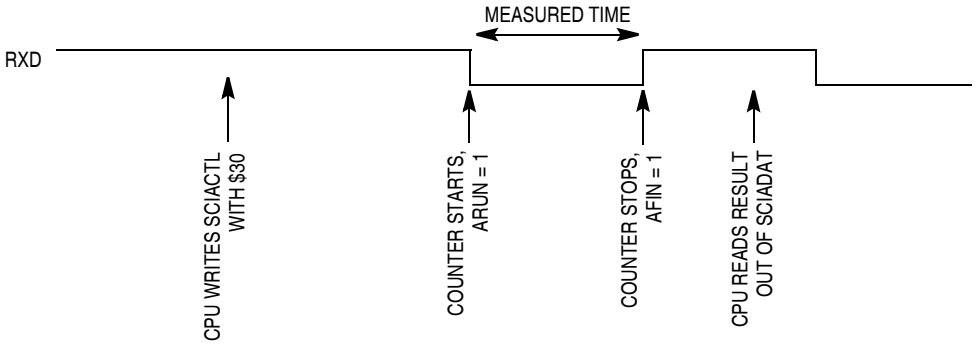



**Figure 13-23. Bit Time Measurement with ACLK = 1, Scenario B**

### 13.9.4 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 is sensed logic 0 without having sensed a logic 0 before on RxD, the counter will be reset, arbitration operation will be restarted after the next rising edge of SCI_TxD.

# Chapter 14
# System Integration Module (SIM)

## 14.1 Introduction

This section describes the system integration module (SIM), which supports up to 24 external and/or internal interrupts. The SIM is a system state controller that coordinates the central processor unit (CPU) and exception timing. Together with the CPU, the SIM controls all microcontroller unit (MCU) activities. A block diagram of the SIM is shown in Figure 14-1.

The SIM is responsible for:

- Bus clock generation and control for CPU and peripherals:
  - Stop/wait/reset entry and recovery
  - Internal clock control
- Master reset control, including power-on reset (POR) and computer operating properly (COP) timeout
- Interrupt control:
  - Acknowledge timing
  - Arbitration control timing
  - Vector address generation
- CPU enable/disable timing
- Modular architecture expandable to 128 interrupt sources

Table 14-1 shows the internal signal names used in this section.

**Table 14-1. Signal Name Conventions**

| Signal Name | Description |
|---|---|
| CGMXCLK | Selected clock source from internal clock generator module (ICG) |
| CGMOUT | Clock output from ICG module (bus clock = CGMOUT divided by two) |
| IAB | Internal address bus |
| IDB | Internal data bus |
| PORRST | Signal from the power-on reset (POR) module to the SIM |
| IRST | Internal reset signal |
| R/$\overline{W}$ | Read/write signal |

## 14.2 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 14-2. This clock originates from either an external oscillator or from the internal clock generator.

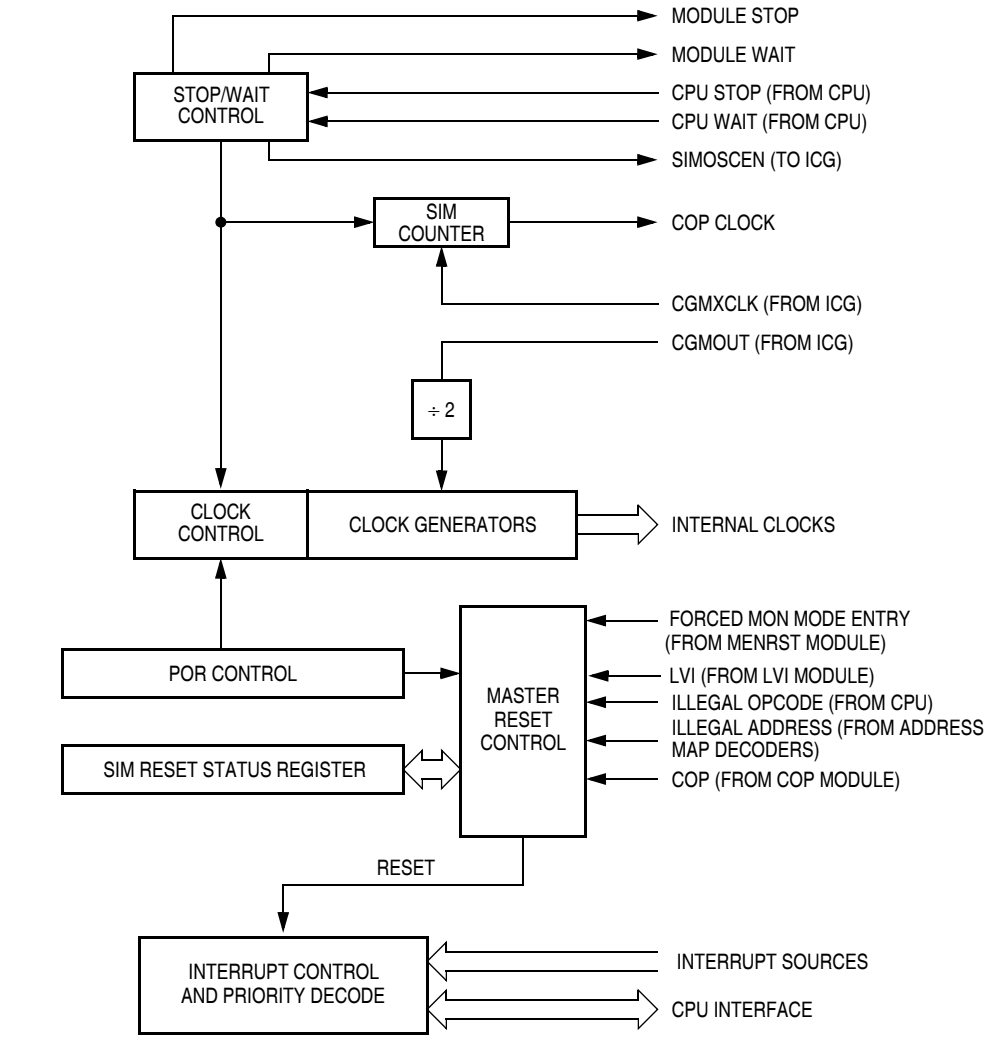



**Figure 14-1. SIM Block Diagram**

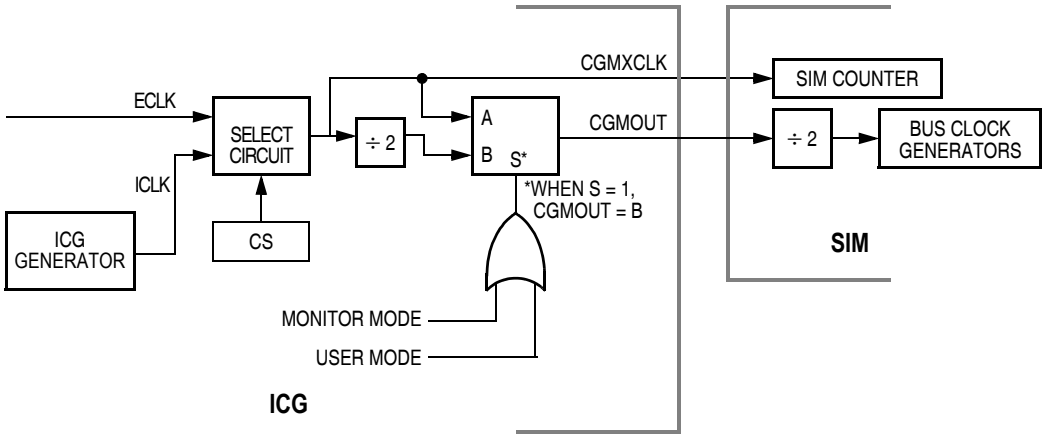



**Figure 14-2. System Clock Signals**

### 14.2.1 Bus Timing

In user mode, the internal bus frequency is the internal clock generator output (CGMXCLK) divided by four.

### 14.2.2 Clock Startup from POR or LVI Reset

When the power-on reset (POR) module or the low-voltage inhibit (LVI) module generates a reset, the clocks to the CPU and peripherals are inactive and held in an inactive phase until after 4096 CGMXCLK cycles. The MCU is held in reset by the SIM during this entire period. The bus clocks start upon completion of the timeout.

### 14.2.3 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. Stop mode recovery timing is discussed in detail in 14.7.2 Stop Mode.

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.

## 14.3 Reset and System Initialization

The MCU has these internal reset sources:

- Power-on reset (POR) module
- Computer operating properly (COP) module
- Low-voltage inhibit (LVI) module
- Illegal opcode
- Illegal address
- Forced monitor mode entry reset (MENRST) module

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.

These internal resets clear the SIM counter and set a corresponding bit in the SIM reset status register (SRSR). See 14.4 SIM Counter and 14.8.2 SIM Reset Status Register.

### 14.3.1 External Pin Reset

The $\overline{\text{RST}}$ pin circuits include an internal pullup device. Pulling the asynchronous $\overline{\text{RST}}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{\text{RST}}$ is held low for at least the minimum of $t_{ILR}$ time. Figure 14-3 shows the relative timing.

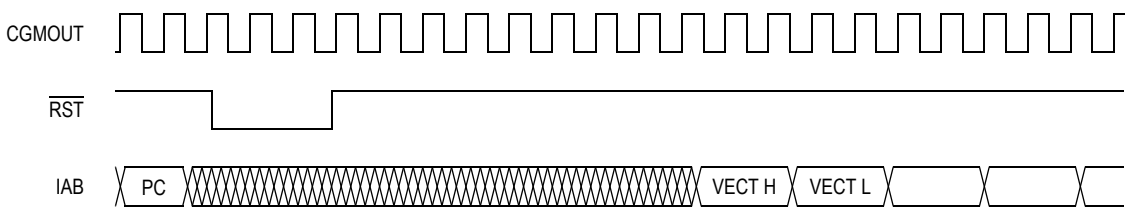



**Figure 14-3. External Reset Timing**

### 14.3.2 Active Resets from Internal Sources

An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, POR, or MENRST as shown in Figure 14-4.

***NOTE***

*For LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM asserts IRST. The internal reset signal then follows with the 64-cycle phase as shown in Figure 14-5.*

The COP reset is asynchronous to the bus clock.

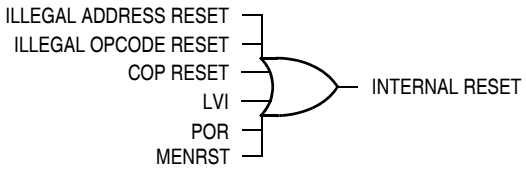



**Figure 14-4. Sources of Internal Reset**

**Table 14-2. Reset Recovery Timing**

| Reset Type | Actual Number of Cycles |
|---|---|
| POR/LVI | 4163 (4096 + 64 + 3) |
| All Others | 67 (64 + 3) |

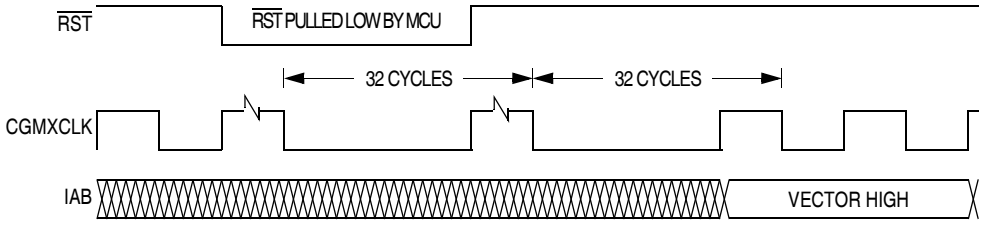



**Figure 14-5. Internal Reset Timing**

#### *14.3.2.1 Power-On Reset*

When power is first applied to the MCU, the power-on reset (POR) module generates a pulse to indicate that power-on has occurred. The MCU is held in reset while the SIM counter counts out 4096 CGMXCLK cycles. Another 64 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 internal clock generator.
- The POR bit of the SIM reset status register (SRSR) is set and all other bits in the register are cleared.

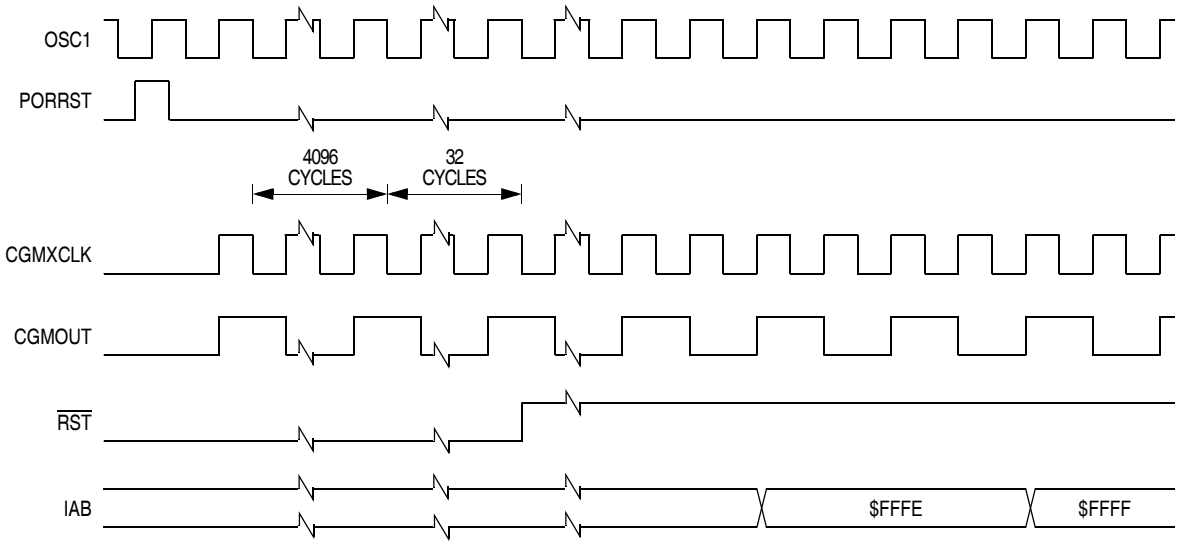



**Figure 14-6. POR Recovery**

#### *14.3.2.2 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 reset status register (SRSR).

To prevent a COP module timeout, write any value to location $FFFF. Writing to location $FFFF clears the COP counter and stages 12–5 of the SIM counter. The SIM counter output, which occurs at least every 4080 CGMXCLK cycles, drives the COP counter. The COP should be serviced as soon as possible out of reset to guarantee the maximum amount of time before the first timeout.

The COP module is disabled if the $\overline{\text{IRQ}}$ pin is held at $V_{TST}$ while the MCU is in monitor mode. The COP module can be disabled only through combinational logic conditioned with the high-voltage signal on the $\overline{\text{IRQ}}$ pin. This prevents the COP from becoming disabled as a result of external noise.

#### *14.3.2.3 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 configuration register (CONFIG1) is 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset.

#### *14.3.2.4 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.

#### *14.3.2.5 Forced Monitor Mode Entry Reset (MENRST)*

The MENRST module is monitoring the reset vector fetches and will assert an internal reset if it detects that the reset vectors are erased ($00). When the MCU comes out of reset, it is forced into monitor mode. See 19.3 Monitor Module (MON).

#### *14.3.2.6 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 a chip reset is asserted if the LVIPWRD and LVIRSTD bits in the CONFIG register are at 0. The MCU is held in reset until $V_{DD}$ rises above $V_{TRIPR.}$ The MCU remains in reset until the SIM counts 4096 CGMXCLK to begin a reset recovery. Another 64 CGMXCLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. See Chapter 11 Low-Voltage Inhibit (LVI) Module.

## 14.4 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 (IBUS) clocks. The SIM counter also serves as a prescaler for the computer operating properly module (COP). The SIM counter overflow supplies the clock for the COP module. The SIM counter is 12 bits long and is clocked by the falling edge of CGMXCLK.

### 14.4.1 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 internal clock generator to drive the bus clock state machine.

### 14.4.2 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 or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the configuration 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.

### 14.4.3 SIM Counter and Reset States

The SIM counter is free-running after all reset states. See 14.3.2 Active Resets from Internal Sources for counter control and internal reset recovery sequences.

## 14.5 Program Exception Control

Normal, sequential program execution can be changed in two ways:

1. Interrupts
   a. Maskable hardware CPU interrupts
   b. Non-maskable software interrupt instruction (SWI)
2. Reset

### 14.5.1 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 return-from-interrupt (RTI) instruction recovers the CPU register contents from the stack so that normal processing can resume. Figure 14-7 shows interrupt entry timing. Figure 14-8 shows interrupt recovery timing.

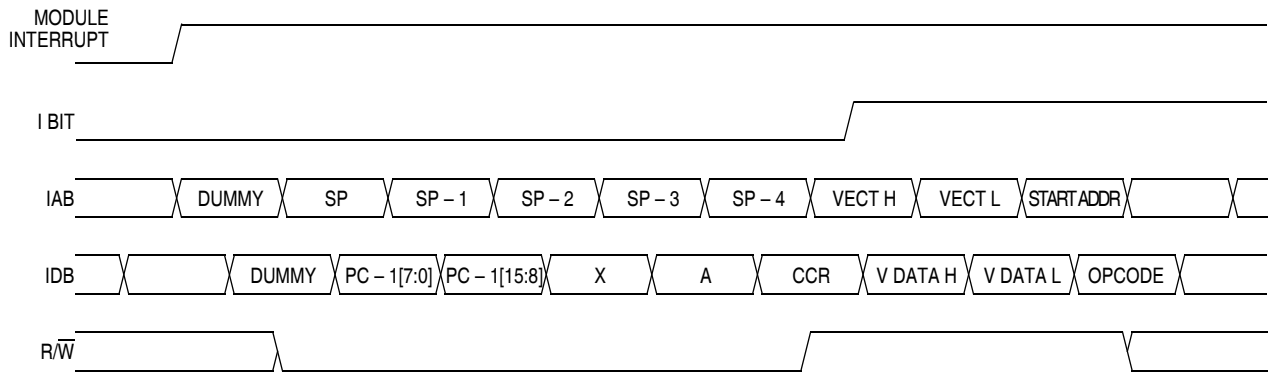



**Figure 14-7. Interrupt Entry**

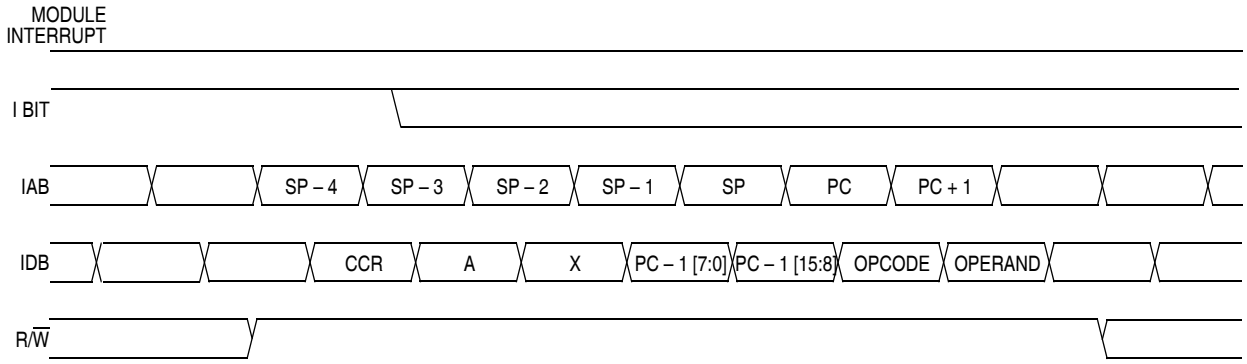



**Figure 14-8. Interrupt Recovery**

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. As shown in Figure 14-9, 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.

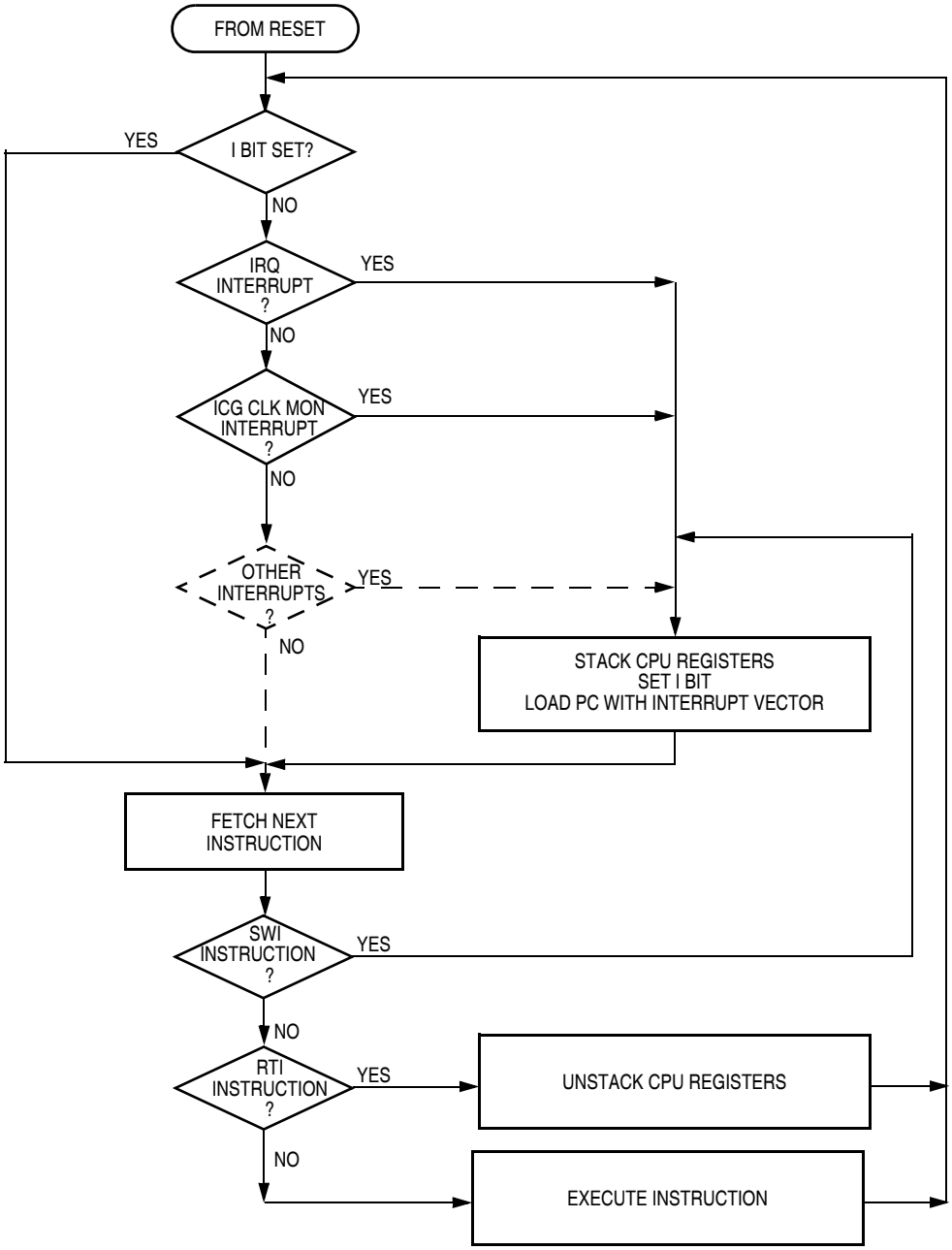



**Figure 14-9. Interrupt Processing**

### *14.5.1.1 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 14-10 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 load-accumulator- from-memory (LDA) instruction is executed.

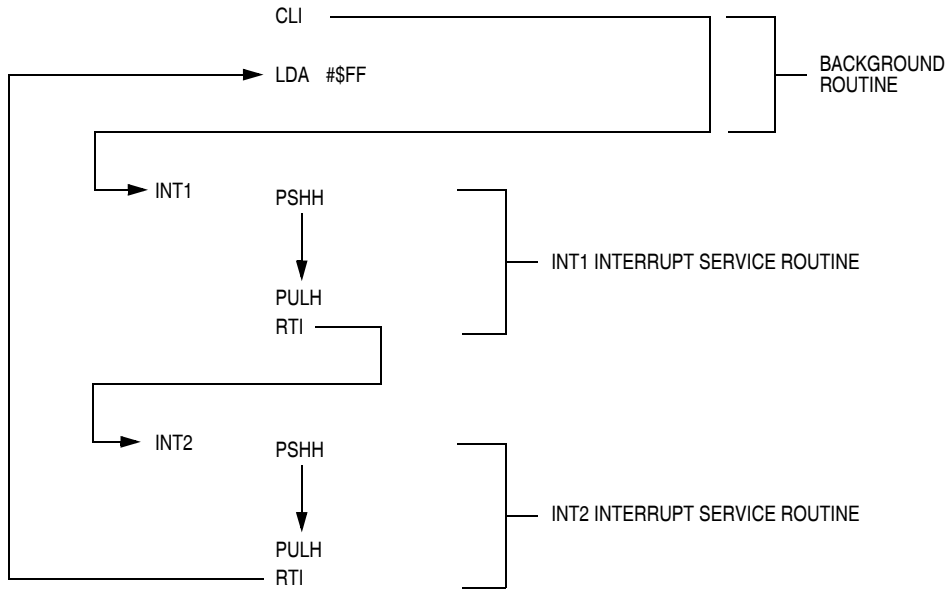



**Figure 14-10. Interrupt Recognition Example**

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, M146805, and MC68HC05 Families 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.*

#### *14.5.1.2 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.*

## 14.6 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. Table 14-3 summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

**Table 14-3. Interrupt Sources**

| Priority | Interrupt Source | Interrupt Status Register Flag |
|---|---|---|
| Highest | Reset | — |
| | SWI instruction | — |
| | $\overline{\text{IRQ}}$ pin | I1 |
| | CGM clock monitor | I2 |
| | TIM1 channel 0 | I3 |
| | TIM1 channel 1 | I4 |
| | TIM1 overflow | I5 |
| | TIM2 channel 0 | I6 |
| | TIM2 channel 1 | I7 |
| | TIM2 overflow | I8 |
| | SCI Error | I9 |
| | SCI receiver | I10 |
| | SCI transmitter | I11 |
| | Keyboard | I12 |
| | ADC conversion complete | I13 |
| | SPI receiver | I14 |
| | SPI transmitter | I15 |
| Lowest | Timebase module | I16 |

### 14.6.1 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 14-11. Interrupt Status Register 1 (INT1)**

**IF6–IF1 — Interrupt Flags 6–1**

These flags indicate the presence of interrupt requests from the sources shown in Table 14-3.

1 = Interrupt request present
0 = No interrupt request present

**Bit 1 and Bit 0 — Always read 0**

### 14.6.2 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 14-12. 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 14-3.

1 = Interrupt request present
0 = No interrupt request present

### 14.6.3 Interrupt Status Register 3

Address: $FE06

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | 0 | 0 | IF16 | IF15 |
| Write: | R | R | R | R | R | R | R | R |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 14-13. Interrupt Status Register 3 (INT3)**

**IF16–IF15 — Interrupt Flags 16–15**

This flag indicates the presence of an interrupt request from the source shown in Table 14-3.

1 = Interrupt request present
0 = No interrupt request present

**Bits 7–2 — Always read 0**

### 14.6.4 Reset

All reset sources always have higher priority than interrupts and cannot be arbitrated.

### 14.6.5 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. See 19.2 Break Module (BRK). 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.

### 14.6.6 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 (SBFCR).

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 two-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.

## 14.7 Low-Power Modes

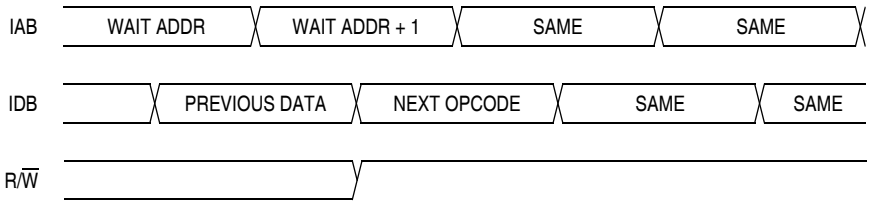
Executing the WAIT or STOP instruction puts the MCU in a low power- consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur. Low-power modes are exited via an interrupt or reset.

### 14.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while one set of peripheral clocks continues to run. Figure 14-14 shows the timing for wait mode entry.

A module that is active during wait mode can wake up 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. 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.

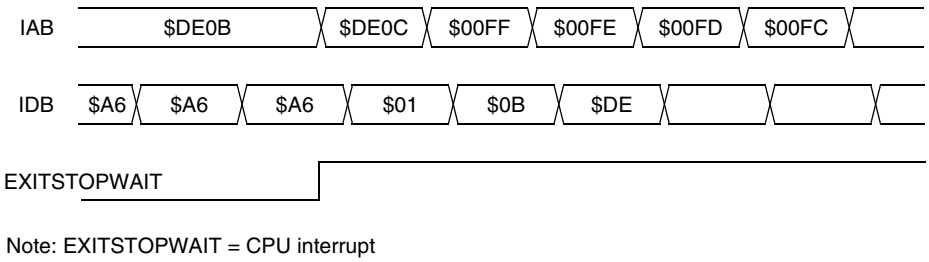
Wait mode can also be exited by a reset. If the COP disable bit, COPD, in the configuration register is 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.




Note: Previous data can be operand data or the WAIT opcode, depending on the last instruction.

**Figure 14-14. Wait Mode Entry Timing**

Figure 14-15 and Figure 14-16 show the timing for WAIT recovery.




Note: EXITSTOPWAIT = CPU interrupt

**Figure 14-15. Wait Recovery from Interrupt**

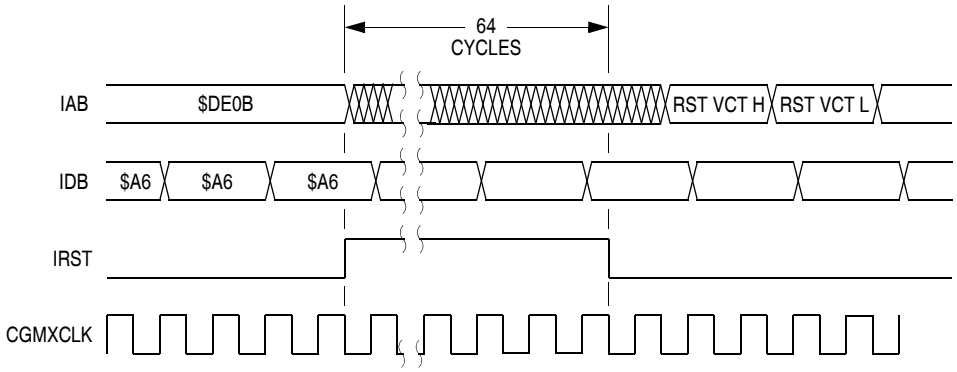



**Figure 14-16. Wait Recovery from Internal Reset**

### 14.7.2 Stop Mode

In stop mode, the SIM counter is held in reset and the CPU and peripheral clocks are held inactive. If the STOPOSCEN bit in the configuration register is not enabled, the SIM also disables the internal clock generator module outputs (CGMOUT and CGMXCLK).

The CPU and peripheral clocks do not become active until after the stop delay timeout. Stop mode is exited via an interrupt request from a module that is still active in stop mode or from a system reset.

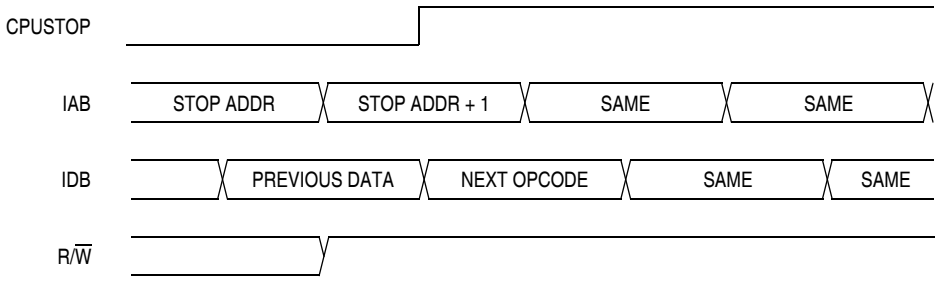
An interrupt request from a module that is still active in stop mode can cause an exit from stop mode. Stop recovery time is selectable using the SSREC bit in the configuration register. If SSREC is set, stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32. Stacking for interrupts begins after the selected stop recovery time has elapsed.

When stop mode is exited due to a reset condition, the SIM forces a long stop recovery time of 4096 CGMXCLK cycles.

***NOTE***

*Short stop recovery is ideal for applications using canned oscillators that do not require long startup times for stop mode. External crystal applications should use the full stop recovery time by clearing the SSREC bit.*

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 14-17 shows stop mode entry timing.




Note: Previous data can be operand data or the STOP opcode, depending on the last instruction.

**Figure 14-17. Stop Mode Entry Timing**

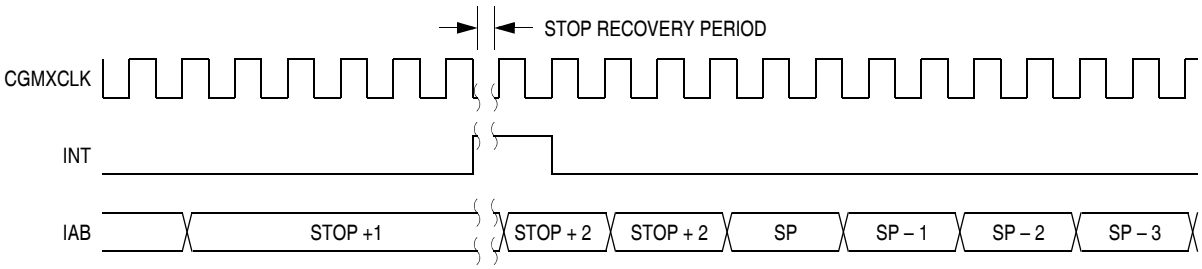



**Figure 14-18. Stop Mode Recovery from Interrupt**

## 14.8 SIM Registers

The SIM has three memory mapped registers. Table 14-4 shows the mapping of these registers.

**Table 14-4. SIM Registers**

| Address | Register | Access Mode |
|---|---|---|
| $FE00 | SBSR | User |
| $FE01 | SRSR | User |
| $FE03 | SBFCR | User |

### 14.8.1 SIM Break Status Register

The SIM break status register (SBSR) contains a flag to indicate that a break caused an exit from stop or wait 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

Note: 1. Writing a 0 clears SBSW

**Figure 14-19. SIM Break Status Register (SBSR)**

**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

### 14.8.2 SIM Reset Status Register

The SRSR register contains flags that show the source of the last reset. The status register will automatically clear after reading it. A power-on reset sets the POR bit and clears all other bits in the register. All other reset sources set the individual flag bits but do not clear the register. More than one reset source can be flagged at any time depending on the conditions at the time of the internal or external reset. For example, the POR and LVI bits can both be set if the power supply has a slow rise time.

Address: $FE01

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | POR | PIN | COP | ILOP | ILAD | MENRST | LVI | 0 |
| Write: | | | | | | | | |
| POR: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 14-20. 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 $\overline{\text{RST}}$
0 = POR or read of SPSR

**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

**MENRST — Forced Monitor Mode Entry Reset Bit**

1 = Last reset was caused by the MENRST circuit
0 = POR or read of SRSR

**LVI — Low-Voltage Inhibit Reset Bit**

1 = Last reset was caused by the LVI circuit
0 = POR or read of SRSR

### 14.8.3 SIM Break Flag Control Register

The SIM break flag control register (SBFCR) contains a bit that enables software to clear status bits while the MCU is in a break state.

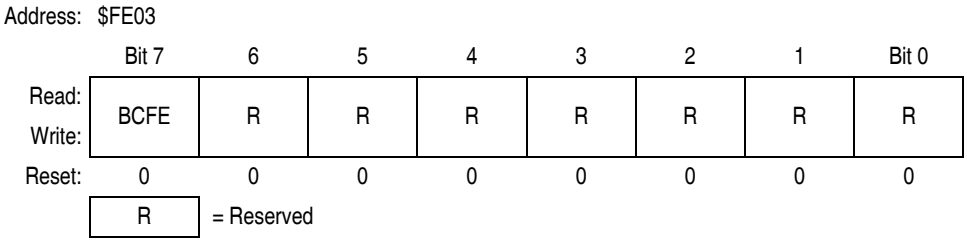



**Figure 14-21. SIM Break Flag Control Register (SBFCR)**

**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 15
# Serial Peripheral Interface (SPI) Module

## 15.1 Introduction

This section describes the serial peripheral interface (SPI) module, which allows full-duplex, synchronous, serial communications with peripheral devices.

## 15.2 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 with CPU service:
  - 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
- $I^2C$ (inter-integrated circuit) compatibility

## 15.3 Pin Name and Register Name Conventions

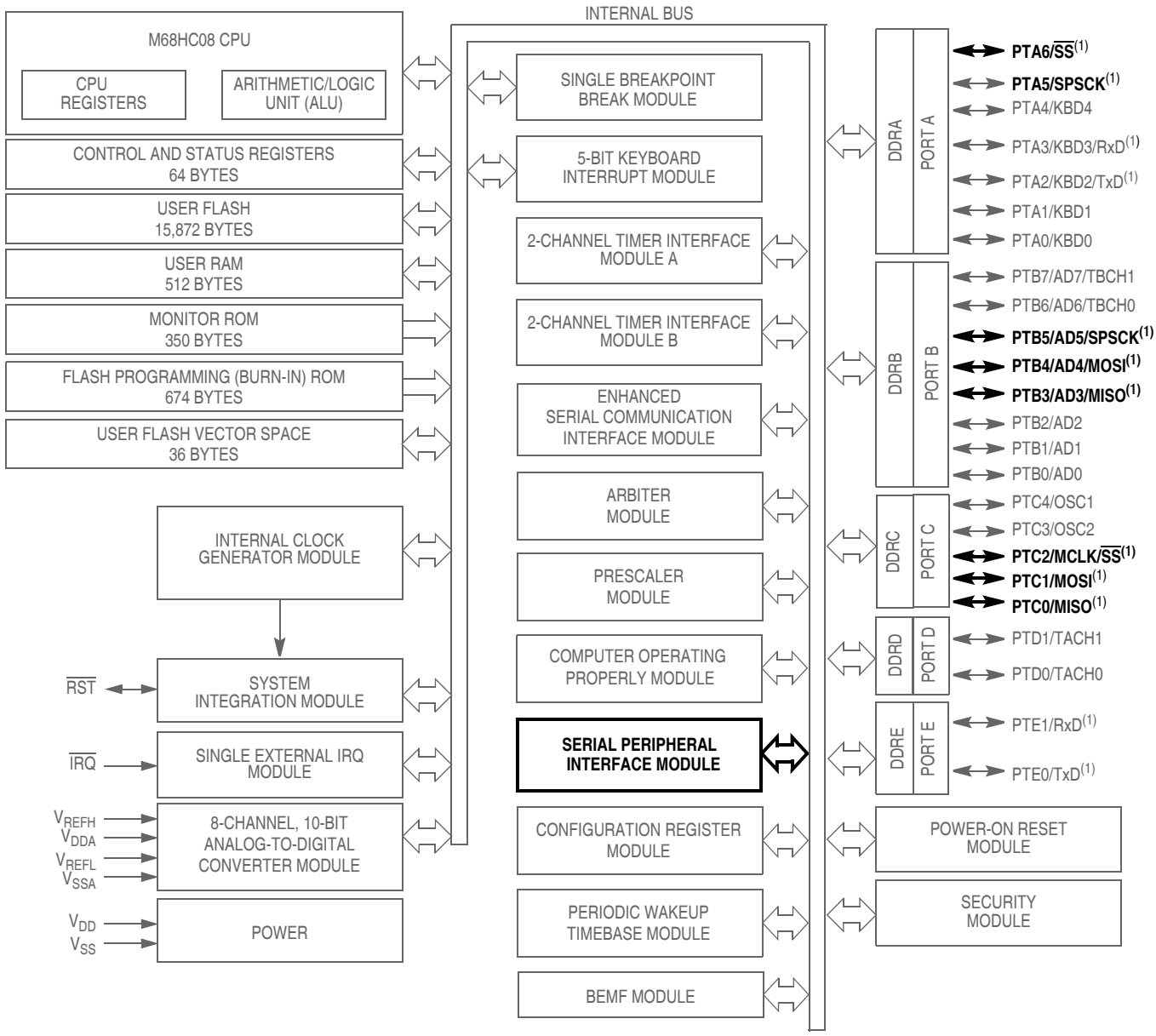
The generic names of the SPI input/output (I/O) pins are:

- $\overline{SS}$ (slave select)
- SPSCK (SPI serial clock)
- MOSI (master out slave in)
- MISO (master in slave out)

The SPI shares four I/O pins with a parallel I/O port. The full name of an SPI pin reflects the name of the shared port pin. Table 15-1 shows the full names of the SPI I/O pins. The generic pin names appear in the text that follows.

**Table 15-1. Pin Name Conventions**

| SPI Generic Pin Name | MISO | MOSI | $\overline{SS}$ | SPSCK |
|---|---|---|---|---|
| Full SPI Pin Name | PTC0/MISO | PTC1/MOSI | PTA6/$\overline{SS}$ | PTA5/SPSCK |
| Alternative Pin | PTB3 | PTB4 | PTC2 | PTB5 |




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

**Figure 15-1. Block Diagram Highlighting SPI Block and Pins**

## 15.4 Functional Description

Figure 15-2 shows the structure of the SPI module.

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. All SPI interrupts can be serviced by the CPU.

The following paragraphs describe the operation of the SPI module.

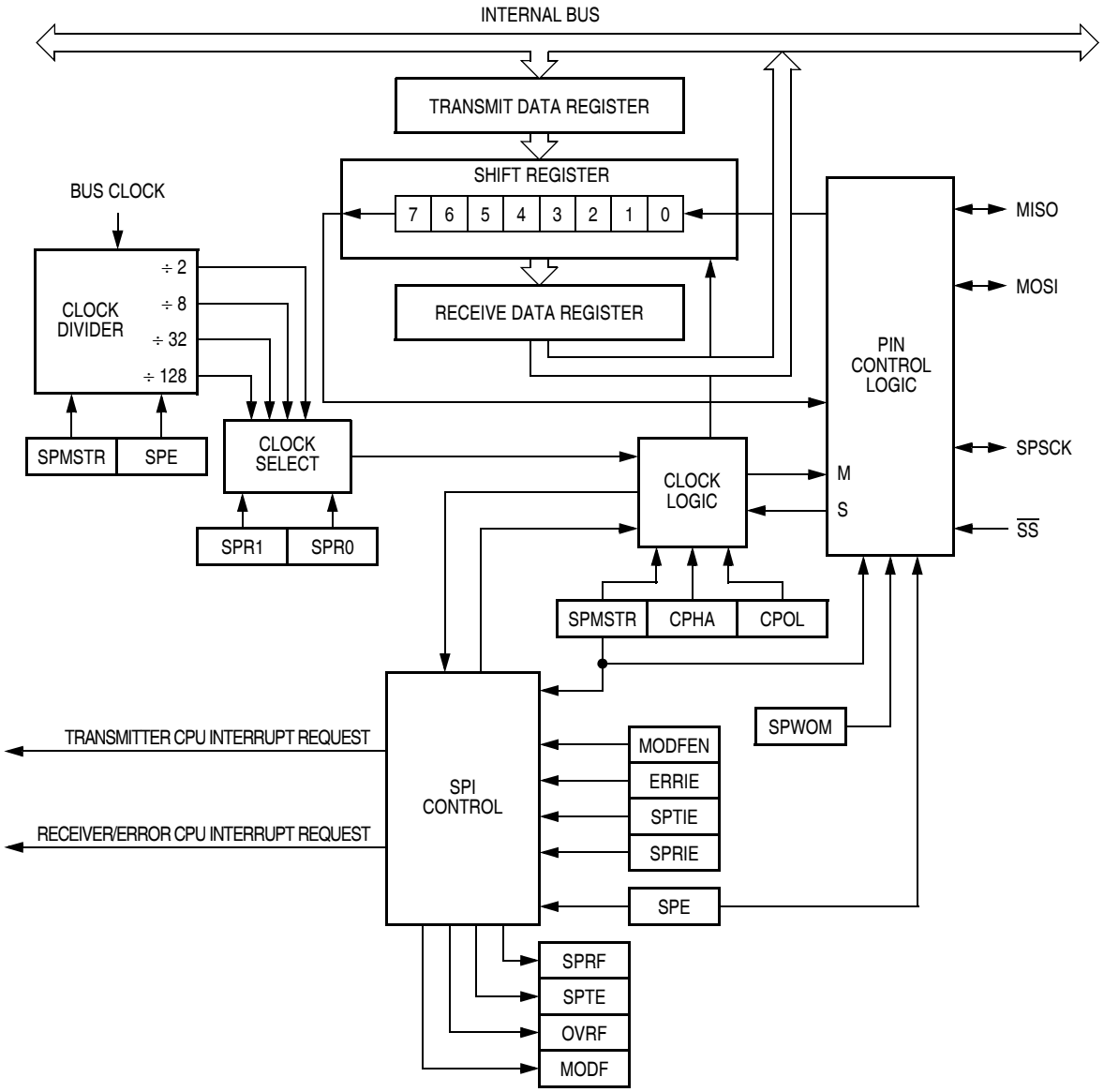



**Figure 15-2. SPI Module Block Diagram**

### 15.4.1 Master Mode

The SPI operates in master mode when the SPI master bit, SPMSTR (SPCR $0010), is set.

***NOTE***

*Configure the SPI modules as master and slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. See 15.13.1 SPI Control Register.*

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the SPI data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE (SPSCR $0011). The byte begins shifting out on the MOSI pin under the control of the serial clock. (See Table 15-2).

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See 15.13.2 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.

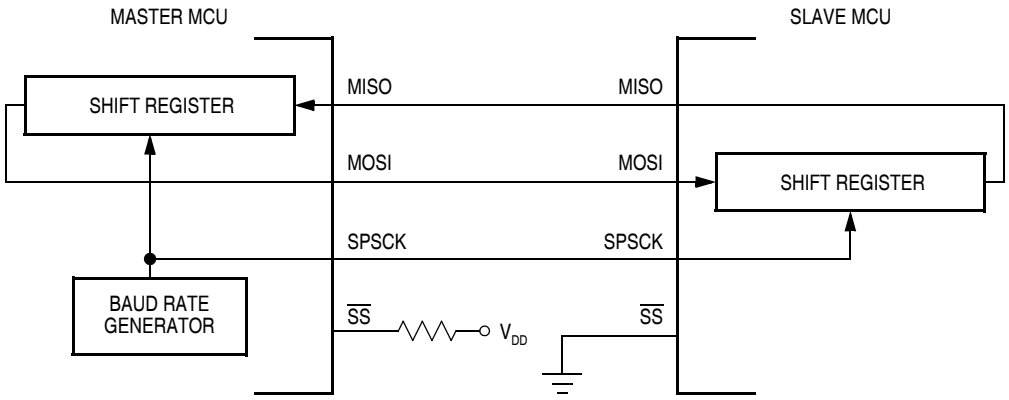



**Figure 15-3. Full-Duplex Master-Slave Connections**

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 (SPSCR), 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 and then reading the SPI data register. Writing to the SPI data register clears the SPTIE bit.

### 15.4.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit (SPCR, $0010) 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 MCU must be at logic 0. $\overline{SS}$ must remain low until the transmission is complete. (See 15.6.2 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 is transferred to the receive data register, and the SPRF bit (SPSCR) is set. To prevent an overflow condition, slave software then must read the SPI data register before another 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 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 15.5 Transmission Formats.)

If the write to the data register is late, the SPI transmits the data already in the shift register from the previous transmission.

***NOTE***

*To prevent SPSCK from appearing as a clock edge, SPSCK must be in the proper idle state before the slave is enabled.*

## 15.5 Transmission Formats

During an SPI transmission, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling on the two serial data lines. A slave select line allows individual selection of a 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 be used optionally to indicate a multiple-master bus contention.

### 15.5.1 Clock Phase and Polarity Controls

Software can select any of four combinations of serial clock (SCK) 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 (SPCR) 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 (SPCR), disable the SPI by clearing the SPI enable bit (SPE).*

### 15.5.2 Transmission Format When CPHA = 0

Figure 15-4 shows an SPI transmission in which CPHA (SPCR) is  0. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: 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 (SCK), 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 at logic 0, 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 15.6.2 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 transmission. The $\overline{SS}$ pin must be toggled high and then low again between each byte transmitted.

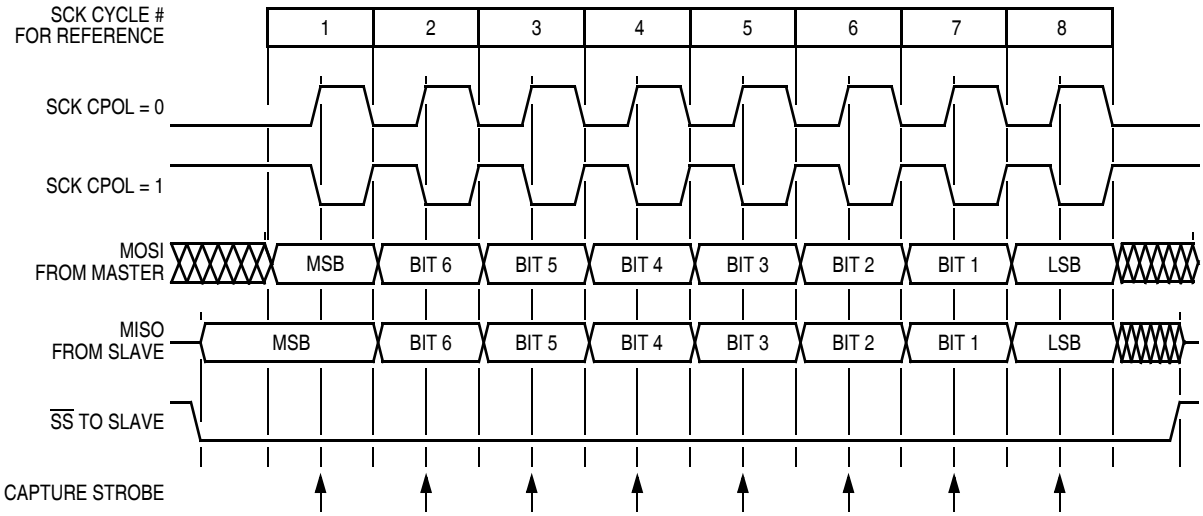



**Figure 15-4. Transmission Format (CPHA = 0)**

### 15.5.3 Transmission Format When CPHA = 1

Figure 15-5 shows an SPI transmission in which CPHA (SPCR) is 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: 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 (SCK), 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 at logic 0, 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 15.6.2 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.

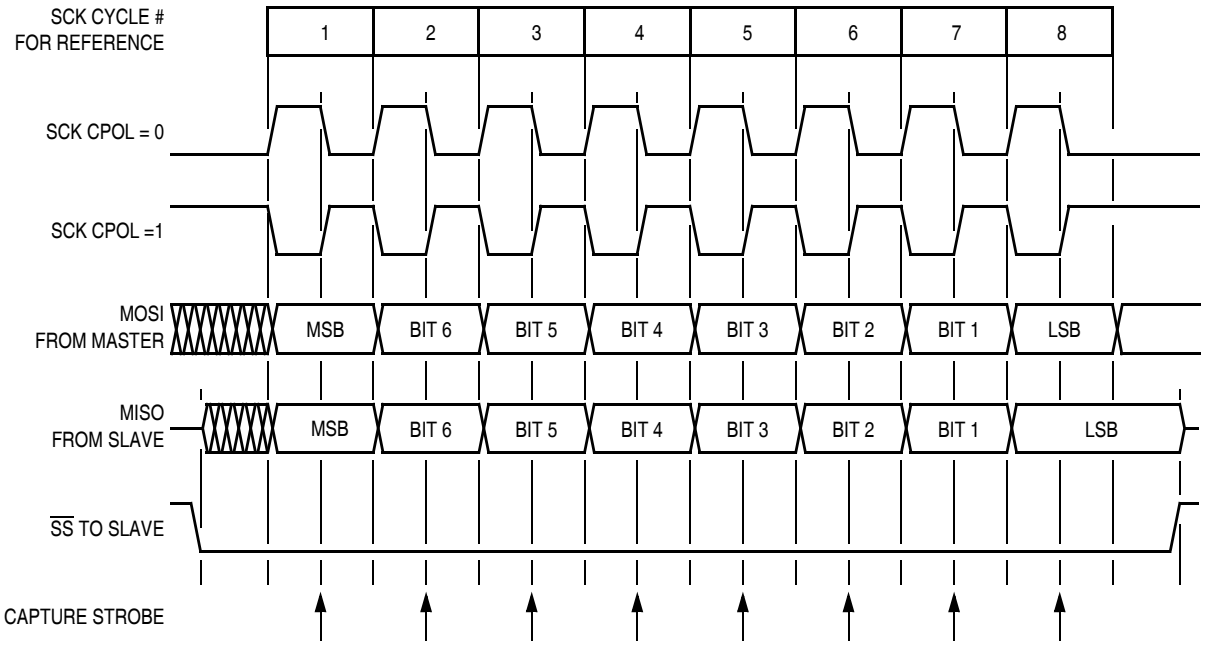



**Figure 15-5. Transmission Format (CPHA = 1)**

### 15.5.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), transmissions are started by a software write to the SPDR ($0012). CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SCK signal. When CPHA = 0, the SCK signal remains inactive for the first half of the first SCK cycle. When CPHA = 1, the first SCK cycle begins with an edge on the SCK 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 15-6.) The internal SPI clock in the master is a free-running derivative of the internal MCU clock. It is only enabled when both the SPE and SPMSTR bits (SPCR) are set to conserve power. SCK edges occur half way through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR will occur relative to the slower SCK. This uncertainty causes the variation in the initiation delay shown in Figure 15-6. This delay will be no longer than a single SPI bit time. That is, the maximum delay between the write to SPDR and the start of the SPI transmission 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.

## 15.6 Error Conditions

Two flags signal SPI error conditions:

1. Overflow (OVRF in SPSCR) — Failing to read the SPI data register before the next 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 by accessing the SPI data register. OVRF is in the SPI status and control register.
2. Mode fault error (MODF in SPSCR) — 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.

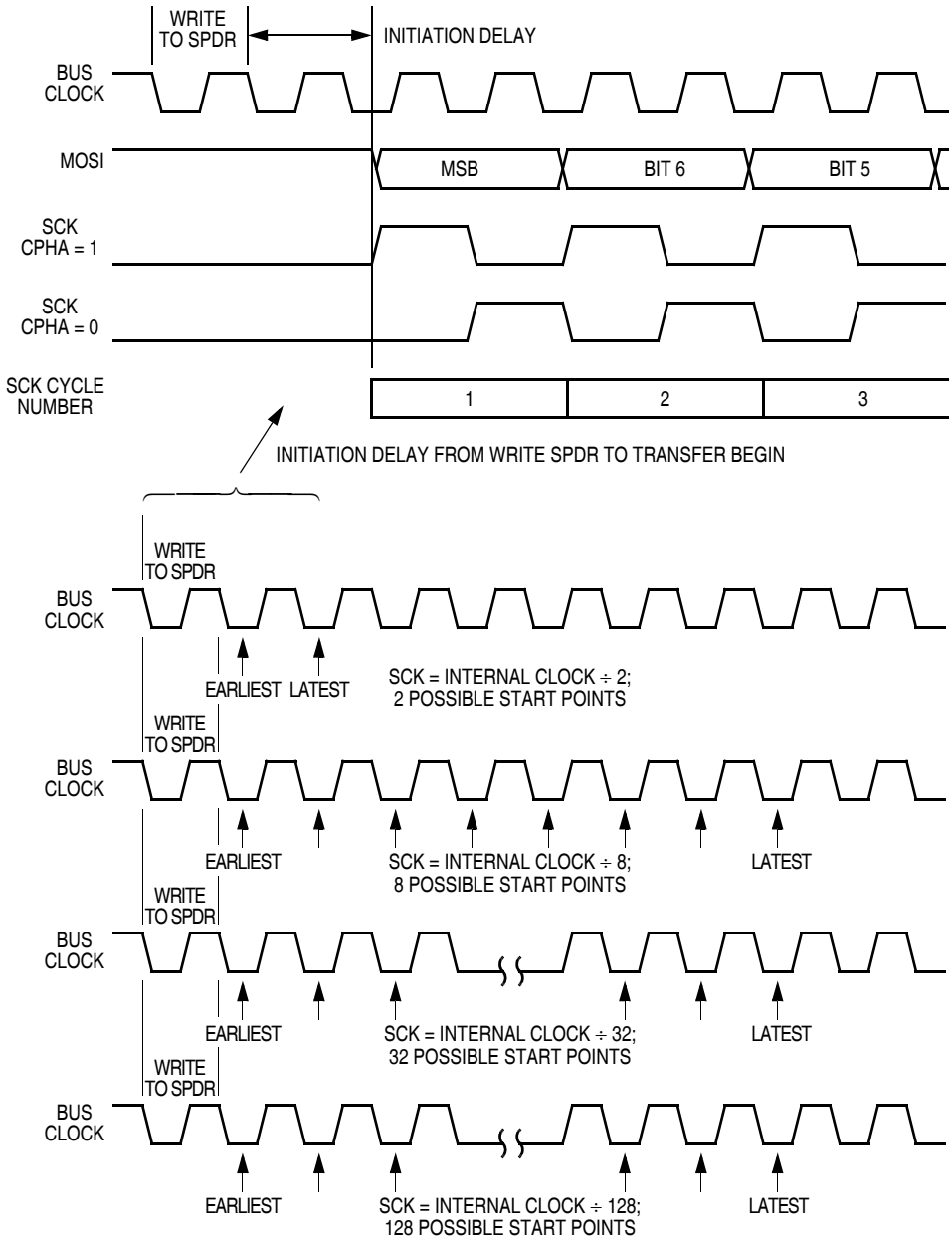



**Figure 15-6. Transmission Start Delay (Master)**

### 15.6.1 Overflow Error

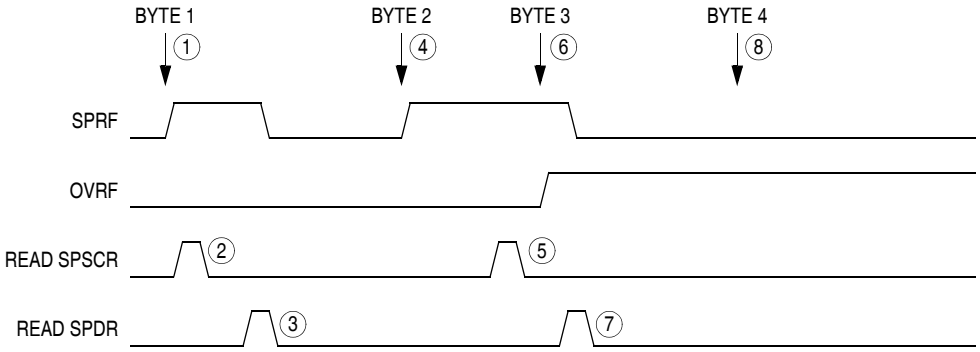
The overflow flag (OVRF in SPSCR) becomes set if the SPI receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. (See Figure 15-4 and Figure 15-5.) If an overflow occurs, the data being received is not transferred to the receive data register so that the unread data can still be read. Therefore, an overflow error always indicates the loss of data.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE in SPSCR) is also set. MODF and OVRF can generate a receiver/error CPU interrupt request. (See Figure 15-9.) It

is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

If an end-of-block transmission interrupt was meant to pull the MCU out of wait, having an overflow condition without overflow interrupts enabled causes the MCU to hang in wait mode. If the OVRF is enabled to generate an interrupt, it can pull the MCU out of wait mode instead.

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. Figure 15-7 shows how it is possible to miss an overflow.




1. BYTE 1 SETS SPRF BIT.
2. CPU READS SPSCR WITH SPRF BIT SET AND OVRF BIT CLEAR.
3. CPU READS BYTE 1 IN SPDR, CLEARING SPRF BIT.
4. BYTE 2 SETS SPRF BIT.
5. CPU READS SPSCRW WITH SPRF BIT SET AND OVRF BIT CLEAR.
6. BYTE 3 SETS OVRF BIT. BYTE 3 IS LOST.
7. CPU READS BYTE 2 IN SPDR, CLEARING SPRF BIT, BUT NOT OVRF BIT.
8. BYTE 4 FAILS TO SET SPRF BIT BECAUSE OVRF BIT IS SET. BYTE 4 IS LOST.

**Figure 15-7. Missed Read of Overflow Condition**

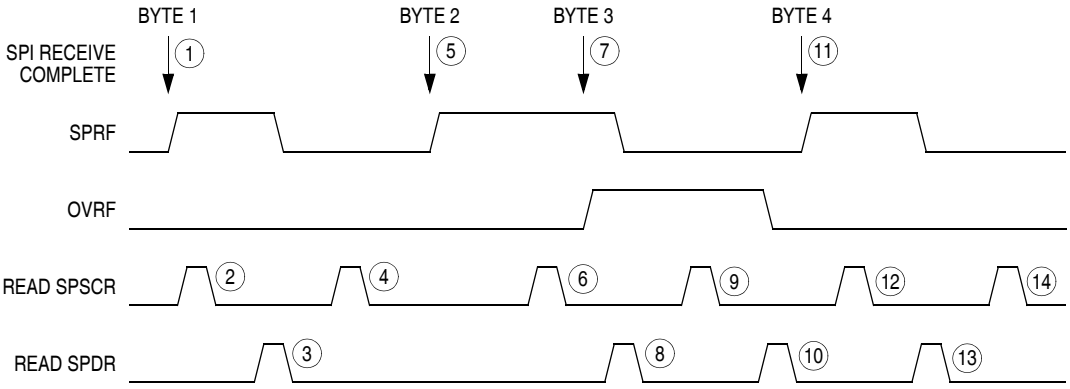
The first part of Figure 15-7 shows how to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF flag can be set in between the time that SPSCR and SPDR are read.

In this case, an overflow can be easily missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it will not be 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 after the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions will complete with an SPRF interrupt. Figure 15-8 illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit (SPSCR).

### 15.6.2 Mode Fault Error

For the MODF flag (in SPSCR) to be set, the mode fault error enable bit (MODFEN in SPSCR) 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 in SPSCR) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. MODF and OVRF can generate a receiver/error CPU interrupt request. (See Figure 15-9). It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.




1. BYTE 1 SETS SPRF BIT.
2. CPU READS SPSCR WITH SPRF BIT SET AND OVRF BIT CLEAR.
3. CPU READS BYTE 1 IN SPDR, CLEARING SPRF BIT.
4. CPU READS SPSCR AGAIN TO CHECK OVRF BIT.
5. BYTE 2 SETS SPRF BIT.
6. CPU READS SPSCR WITH SPRF BIT SET AND OVRF BIT CLEAR.
7. BYTE 3 SETS OVRF BIT. BYTE 3 IS LOST.
8. CPU READS BYTE 2 IN SPDR, CLEARING SPRF BIT.
9. CPU READS SPSCR AGAIN TO CHECK OVRF BIT.
10. CPU READS BYTE 2 SPDR, CLEARING OVRF BIT.
11. BYTE 4 SETS SPRF BIT.
12. CPU READS SPSCR.
13. CPU READS BYTE 4 IN SPDR, CLEARING SPRF BIT.
14. CPU READS SPSCR AGAIN TO CHECK OVRF BIT.

**Figure 15-8. Clearing SPRF When OVRF Interrupt Is Not Enabled**

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes to logic 0. 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 data direction register (DDR) bits associated with the SPI shared port pins.*

***NOTE***

*Setting the MODF flag (SPSCR) does not clear the SPMSTR bit. Reading SPMSTR when MODF = 1 will indicate a MODE fault error occurred in either master mode or slave mode.*

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 returns to its idle level after 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 after the shift of the last data bit. (See 15.5 Transmission Formats.)

***NOTE***

*When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is at logic 0) and later deselected ($\overline{SS}$ is at logic 1) even if no SPSCK is sent to that slave. This happens because $\overline{SS}$ at logic 0 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 deselected with no transmission*

*occurring. Therefore, MODF does not occur since a transmission was never begun.*

In a slave SPI (MSTR = 0), the MODF bit 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 toggling the SPE bit of the slave.

***NOTE***

*A logic 1 voltage 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 a transmission has begun.*

To clear the MODF flag, read the SPSCR and then write to the SPCR register. This entire clearing procedure must occur with no MODF condition existing or else the flag will not be cleared.

## 15.7 Interrupts

Four SPI status flags can be enabled to generate CPU interrupt requests:

**Table 15-2. SPI Interrupts**

| Flag | Request |
|---|---|
| SPTE (Transmitter Empty) | SPI Transmitter CPU Interrupt Request (SPTIE = 1) |
| SPRF (Receiver Full) | SPI Receiver CPU Interrupt Request (SPRIE = 1) |
| OVRF (Overflow) | SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1) |
| MODF (Mode Fault) | SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1, MODFEN = 1) |

The SPI transmitter interrupt enable bit (SPTIE) enables the SPTE flag to generate transmitter CPU interrupt requests.

The SPI receiver interrupt enable bit (SPRIE) enables the SPRF bit to generate receiver CPU interrupt, provided that the SPI is enabled (SPE = 1).

The error interrupt enable bit (ERRIE) enables both the MODF and OVRF flags 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 flag is enabled to generate receiver/error CPU interrupt requests.

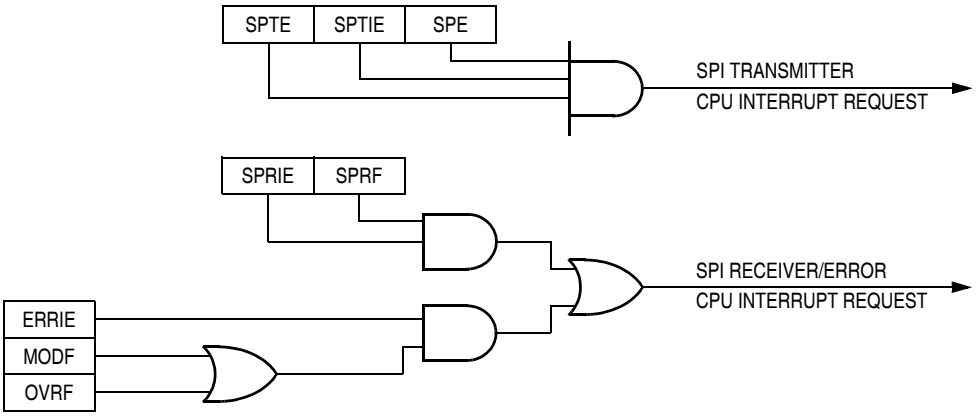



**Figure 15-9. SPI Interrupt Request Generation**

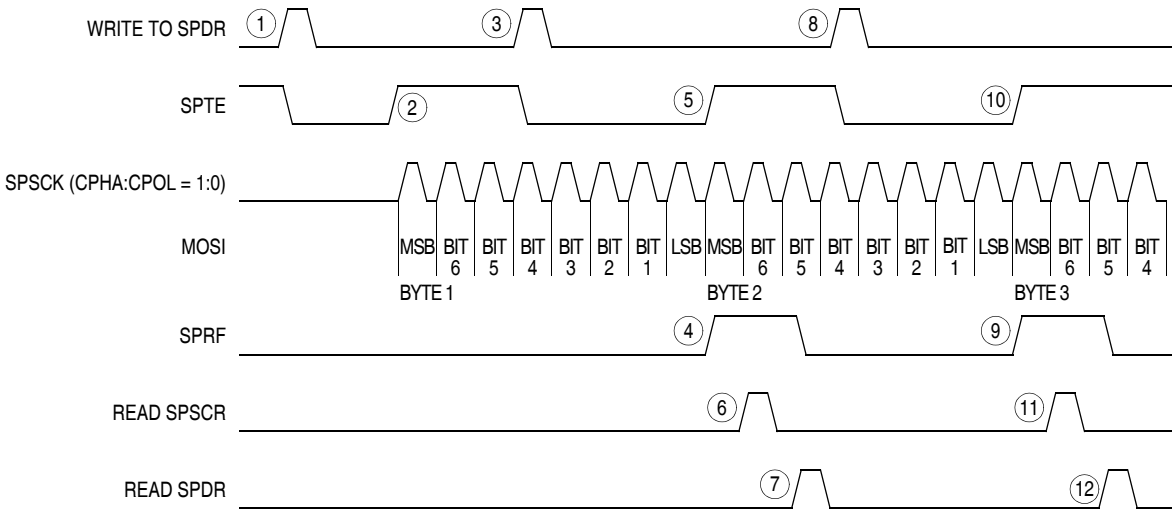
Two sources in the SPI status and control register can generate CPU interrupt requests:

1. SPI receiver full bit (SPRF) — The SPRF bit 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 can generate an SPI receiver/error CPU interrupt request.
2. SPI transmitter empty (SPTE) — The SPTE bit 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 can generate an SPTE CPU interrupt request.

## 15.8 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 in SPSCR) indicates when the transmit data buffer is ready to accept new data. Write to the SPI data register only when the SPTE bit is high. Figure 15-10 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA:CPOL = 1:0).

For a slave, the transmit data buffer allows back-to-back transmissions to occur without the slave having to time the write of its data between the transmissions. Also, if no new data is written to the data buffer, the last value contained in the shift register will be the next data word transmitted.




1. CPU WRITES BYTE 1 TO SPDR, CLEARING SPTE BIT.
2. BYTE 1 TRANSFERS FROM TRANSMIT DATA REGISTER TO SHIFT REGISTER, SETTING SPTE BIT.
3. CPU WRITES BYTE 2 TO SPDR, QUEUEING BYTE 2 AND CLEARING SPTE BIT.
4. FIRST INCOMING BYTE TRANSFERS FROM SHIFT REGISTER TO RECEIVE DATA REGISTER, SETTING SPRF BIT.
5. BYTE 2 TRANSFERS FROM TRANSMIT DATA REGISTER TO SHIFT REGISTER, SETTING SPTE BIT.
6. CPU READS SPSCR WITH SPRF BIT SET.
7. CPU READS SPDR, CLEARING SPRF BIT.
8. CPU WRITES BYTE 3 TO SPDR, QUEUEING BYTE 3 AND CLEARING SPTE BIT.
9. SECOND INCOMING BYTE TRANSFERS FROM SHIFT REGISTER TO RECEIVE DATA REGISTER, SETTING SPRF BIT.
10. BYTE 3 TRANSFERS FROM TRANSMIT DATA REGISTER TO SHIFT REGISTER, SETTING SPTE BIT.
11. CPU READS SPSCR WITH SPRF BIT SET.
12. CPU READS SPDR, CLEARING SPRF BIT.

**Figure 15-10. SPRF/SPTE CPU Interrupt Timing**

## 15.9 Resetting the SPI

Any system reset completely resets the SPI. Partial reset occurs whenever the SPI enable bit (SPE) is low. Whenever SPE is low, 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.

The following additional 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 reset all control bits when SPE is set back to 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 a 0 to the SPE bit. The SPI also can be disabled by a mode fault occurring in an SPI that was configured as a master with the MODFEN bit set.

## 15.10 Low-Power Modes

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

### 15.10.1 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 15.7 Interrupts.)

### 15.10.2 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 the MCU exits stop mode. If stop mode is exited by reset, any transfer in progress is aborted and the SPI is reset.

## 15.11 SPI 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, $FE03) enables software to clear status bits during the break state. (See 19.2.1.1 Flag Protection During Break Interrupts.)

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.

Since the SPTE bit cannot be cleared during a break with the BCFE bit cleared, a write to the data register in break mode will not initiate a transmission nor will this data be transferred into the shift register. Therefore, a write to the SPDR in break mode with the BCFE bit cleared has no effect.

## 15.12 SPI I/O Signals

The SPI module has four I/O pins and shares three of them with a parallel I/O port.

- MISO — Data received
- MOSI — Data transmitted
- SPSCK — Serial clock
- $\overline{SS}$ — Slave select
- $V_{SS}$ — Clock ground

The SPI has limited inter-integrated circuit ($I^2C$) capability (requiring software support) as a master in a single-master environment. To communicate with $I^2C$ peripherals, MOSI becomes an open-drain output when the SPWOM bit in the SPI control register is set. In $I^2C$ communication, the MOSI and MISO pins are connected to a bidirectional pin from the $I^2C$ peripheral and through a pullup resistor to $V_{DD}$.

### 15.12.1 MISO (Master In/Slave Out)

MISO is one of the two SPI module pins that transmit 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 at logic 0. To support a multiple-slave system, a logic 1 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.

### 15.12.2 MOSI (Master Out/Slave In)

MOSI is one of the two SPI module pins that transmit 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.

### 15.12.3 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.

### 15.12.4 $\overline{SS}$ (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 15.5 Transmission Formats.) Since it is used to indicate the start of a transmission, the $\overline{SS}$ must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low throughout the transmission for the CPHA = 1 format. See Figure 15-11.

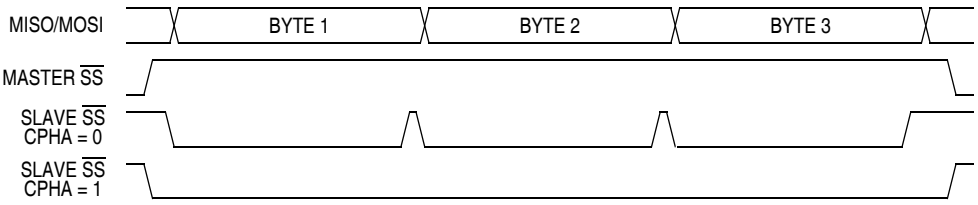



**Figure 15-11. CPHA/$\overline{SS}$ Timing**

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 the $\overline{SS}$ from creating a MODF error. (See 15.13.2 SPI Status and Control Register.)

***NOTE***

*A logic 1 voltage 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 a transmission already has begun.*

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 15.6.2 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 the MODFEN bit is low 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. With MODFEN high, it 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 data register. (See Table 15-2.)

### 15.12.5 $V_{SS}$ (Clock Ground)

$V_{SS}$ is the ground return for the serial clock pin, SPSCK, and the ground for the port output buffers. To reduce the ground return path loop and minimize radio frequency (RF) emissions, connect the ground pin of the slave to the $V_{SS}$ pin.

## 15.13 I/O Registers

Three registers control and monitor SPI operation:

- SPI control register (SPCR $0010)
- SPI status and control register (SPSCR $0011)
- SPI data register (SPDR $0012)

### 15.13.1 SPI Control Register

The SPI control register:

- Enables SPI module interrupt requests
- Selects CPU 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

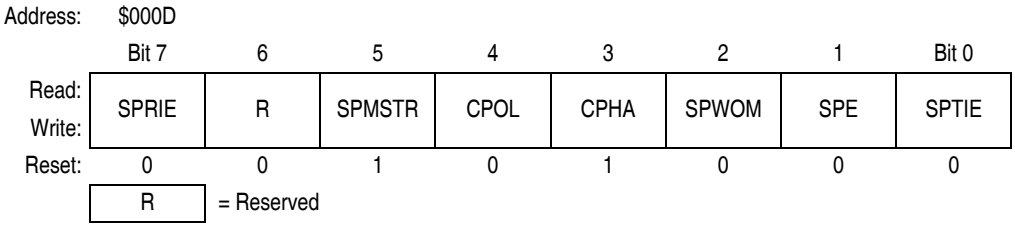



**Figure 15-12. SPI Control Register (SPCR)**

**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 15-4 and Figure 15-5.) To transmit data between SPI modules, the SPI modules must have identical CPOL bits. 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 15-4 and Figure 15-5.) To transmit data between SPI modules, the SPI modules must have identical CPHA bits. When CPHA = 0, the $\overline{SS}$ pin of the slave SPI module must be set to logic 1 between bytes. (See Figure 15-11). Reset sets the CPHA bit.

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 data register. Therefore,

the slave data register must be loaded with the desired transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the data register and transferred to the shift register at the current transmission.

When CPHA = 1 for a slave, the first edge of the SPSCK indicates the beginning of the transmission. The same applies when $\overline{SS}$ is high for a slave. The MISO pin is held in a high-impedance state, and the incoming SPSCK is ignored. In certain cases, it may also cause the MODF flag to be set. (See 15.6.2 Mode Fault Error). A logic 1 on the $\overline{SS}$ pin does not in any way affect the state of the SPI state machine.

**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 Bit**

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI (see 15.9 Resetting the SPI). Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

**SPTIE — SPI Transmit Interrupt Enable Bit**

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

### 15.13.2 SPI Status and Control Register

The SPI status and control register contains flags to signal the following 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: $000E

| | 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 15-13. 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. Any read of the SPI data register clears the SPRF bit.

Reset clears the SPRF bit.

1 = Receive data register full
0 = Receive data register not full

**ERRIE — Error Interrupt Enable Bit**

This read-only bit enables the MODF and OVRF flags 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 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 SPI data register. Reset clears the OVRF flag.

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. In a master SPI, the MODF flag is set if the $\overline{SS}$ pin goes low at any time. Clear the MODF bit by reading the SPI status and control register with MODF set and then writing to the 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 the SPTIE bit in the SPI control register is set also.

***NOTE***

*Do not write to the SPI data register unless the SPTE bit is high.*

For an idle master or idle slave that has no data loaded into its transmit buffer, the SPTE will be set again within two bus cycles since 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. The SPTE indicates when the next write can occur.

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 to 1, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag. If the SPI is enabled as a master and the MODFEN bit is low, then the $\overline{SS}$ pin is available as a general-purpose I/O.

If the MODFEN bit is set, then this 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 15.12.4 SS (Slave Select)).

If the MODFEN bit is low, 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 15.6.2 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 15-3. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

**Table 15-3. SPI Master Baud Rate Selection**

| SPR1:SPR0 | Baud Rate Divisor (BD) |
|---|---|
| 00 | 2 |
| 01 | 8 |
| 10 | 32 |
| 11 | 128 |

Use this formula to calculate the SPI baud rate:

$$\text{Baud rate} = \frac{\text{CGMOUT}}{2 \times \text{BD}}$$

where:

CGMOUT = base clock output of the internal clock generator module (ICG), see Chapter 8 Internal Clock Generator (ICG) Module.

BD = baud rate divisor

### 15.13.3 SPI Data Register

The SPI data register is the read/write buffer for the receive data register and the 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 buffers that can contain different values. See Figure 15-2

Address: $000F

| | 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: | Indeterminate after Reset | | | | | | | |

**Figure 15-14. 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 buffer read is not the same as the buffer written.*

# Chapter 16
# Timebase Module (TBM)

## 16.1 Introduction

This section describes the timebase module (TBM). The TBM will generate periodic interrupts at user selectable rates using a counter clocked by either the internal or external clock sources. This TBM version uses 15 divider stages, eight of which are user selectable.

## 16.2 Features

Features of the TBM module include:

- Software configurable periodic interrupts with divide-by-8, 16, 32, 64, 128, 1024, 2048, 4096, 8192, 16384, 32768, 262,144, 1,048,576, and 4,194,304 taps of the selected clock source
- Configurable for operation during stop mode to allow periodic wake up from stop

## 16.3 Functional Description

This module can generate a periodic interrupt by dividing the clock source supplied from the internal clock generator module, TBMCLK. Note that this clock source is the external clock ECLK when the ECGON bit in the ICG control register (ICGCR) is set. Otherwise, TBMCLK is driven at the internally generated clock frequency (ICLK). In other words, if the external clock is enabled it will be used as the TBMCLK, even if the MCU bus clock is based on the internal clock.

The counter is initialized to all 0s when TBON bit is cleared. The counter, shown in Figure 16-1, 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 internal clock generator 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.

## 16.4 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.

Interrupts must be acknowledged by writing a 1 to the TACK bit.

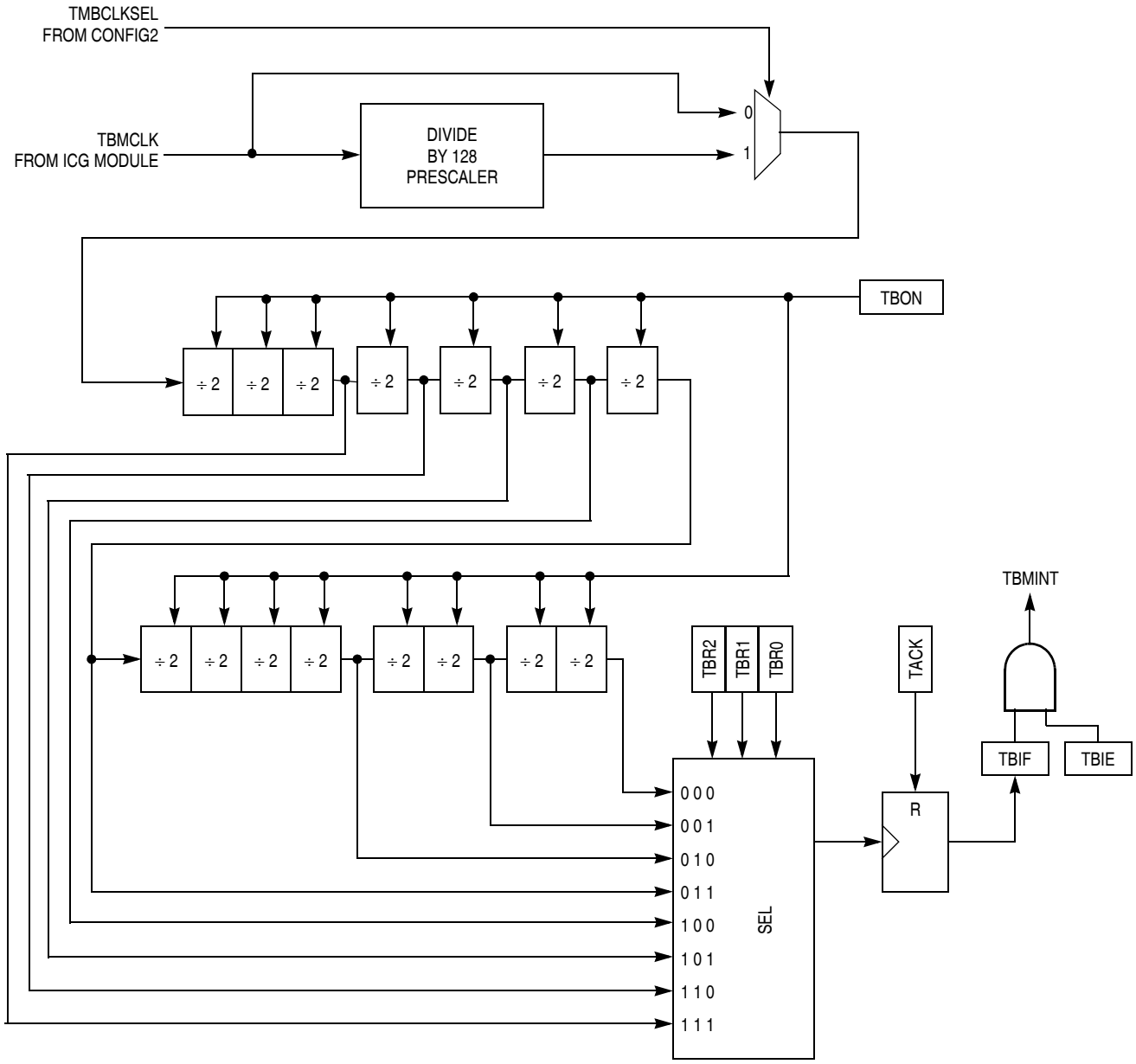



**Figure 16-1. Timebase Block Diagram**

## 16.5 TBM Interrupt Rate

The interrupt rate is determined by the equation:

$$t_{TBMRATE} = \frac{1}{f_{TBMRATE}} = \frac{Divider}{f_{TBMCLK}}$$

where:

$f_{TBMCLK}$ = Frequency supplied from the internal clock generator (ICG) module

Divider = Divider value as determined by TBR2–TBR0 settings. See Table 16-1.

As an example, a clock source of 4.9152 MHz and the TBR2–TBR0 set to {011}, the divider tap is 128 and the interrupt rate calculates to 128/4.9152 x $10^6$ = 26 μs.

**Table 16-1. Timebase Divider Selection**

| TBR2[1] | TBR1[1] | TBR0[1] | Divider Tap TMBCLKSEL | |
|---|---|---|---|---|
| | | | 0 | 1 |
| 0 | 0 | 0 | 32,768 | 4,194,304 |
| 0 | 0 | 1 | 8192 | 1,048,576 |
| 0 | 1 | 0 | 2048 | 262,144 |
| 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 |

1. Do not change TBR2–TBR0 bits while the timebase is enabled (TBON = 1).

## 16.6 Low-Power Modes

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

### 16.6.1 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.

### 16.6.2 Stop Mode

The timebase module may remain active after execution of the STOP instruction if the internal clock generator 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 wake up from stop mode.

If the internal clock generator 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.

## 16.7 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 16-2. 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 16-1.

***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.

***NOTE***

*Clearing TBON has no effect on the TBIF flag.*

# Chapter 17
# Timer Interface A (TIMA) Module

## 17.1 Introduction

This section describes the timer interface A module (TIMA). The TIMA is a 2-channel timer that provides a timing reference with input capture, output compare, and pulse width modulation (PWM) functions. Figure 17-2 is a block diagram of the TIMA.

For further information regarding timers on M68HC08 family devices, please consult the *HC08 Timer Reference Manual*, Freescale document order number TIM08RM/AD.

## 17.2 Features

Features include:

- 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 PWM signal generation
- Programmable TIMA clock input
  - 7-frequency internal bus clock prescaler selection
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMA counter stop and reset bits

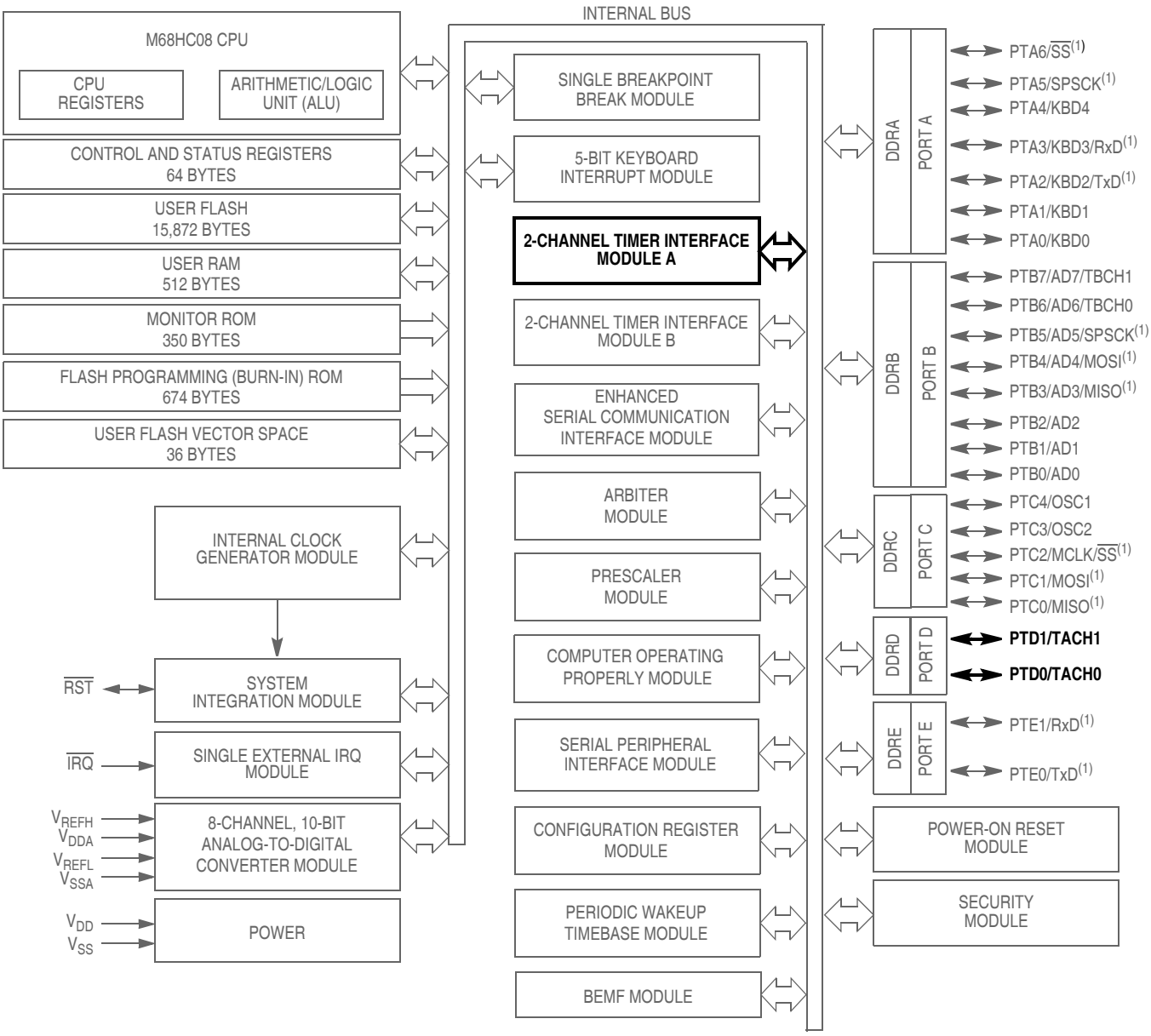
## 17.3 Functional Description

Figure 17-2 shows the TIMA structure. The central component of the TIMA is the 16-bit TIMA counter that can operate as a free-running counter or a modulo up-counter. The TIMA counter provides the timing reference for the input capture and output compare functions. The TIMA counter modulo registers, TAMODH–TAMODL, control the modulo value of the TIMA counter. Software can read the TIMA counter value at any time without affecting the counting sequence.

The two TIMA channels are programmable independently as input capture or output compare channels.

### 17.3.1 TIMA Counter Prescaler

The TIMA clock source can be 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 TIMA status and control register select the TIMA clock source.




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

**Figure 17-1. Block Diagram Highlighting TIMA Block and Pins**

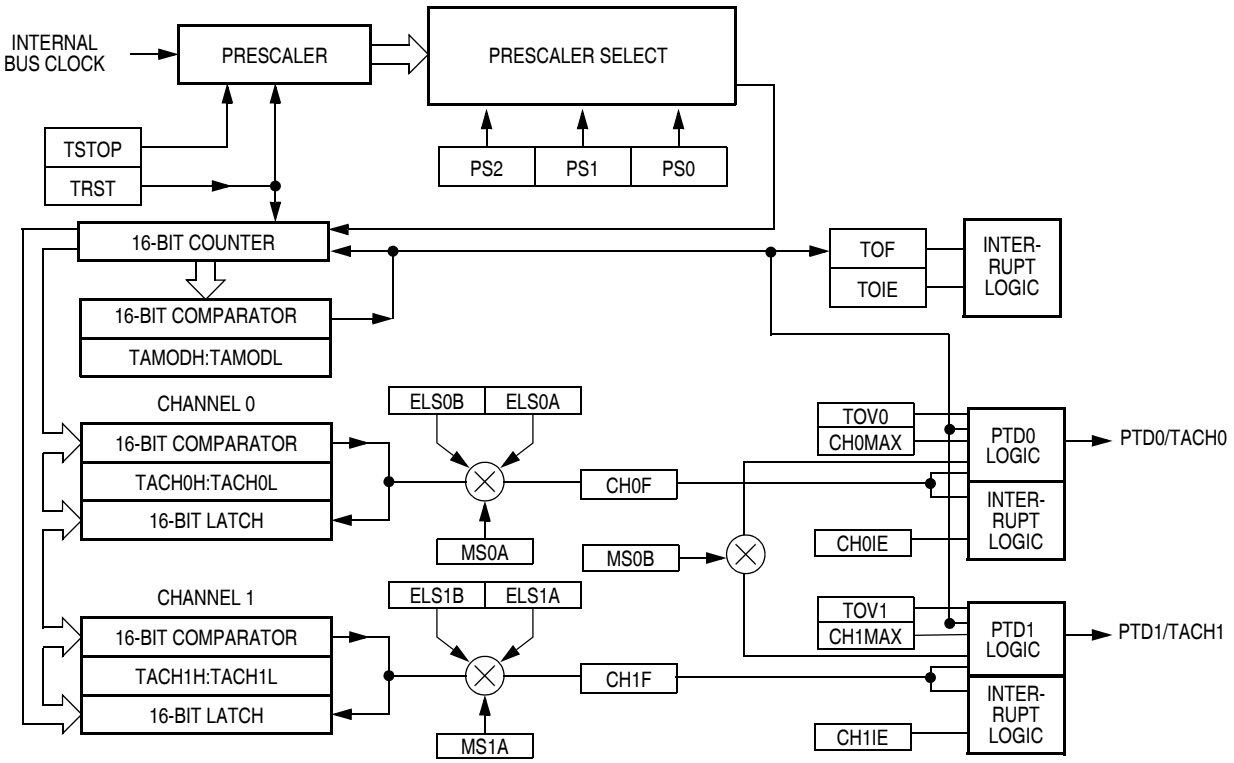



**Figure 17-2. TIMA Block Diagram**

### 17.3.2 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 TASC0 through TASC1 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMA latches the contents of the TIMA counter into the TIMA channel registers, TACHxH–TACHxL. Input captures can generate TIMA 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 TIMA channel status and control register (TACHxH–TACHxL, see 17.8.5 TIMA Channel Registers) on each proper signal transition regardless of whether the TIMA channel flag (CH0F–CH1F in TASC0–TASC1 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 2 bus cycles after 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 17.8.5 TIMA 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 register (TACHxH–TACHxL).

### 17.3.3 Output Compare

With the output compare function, the TIMA 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 TIMA can set, clear, or toggle the channel pin. Output compares can generate TIMA CPU interrupt requests.

#### *17.3.3.1 Unbuffered Output Compare*

Any output compare channel can generate unbuffered output compare pulses as described in 17.3.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMA channel registers.

An unsynchronized write to the TIMA 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 TIMA overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIMA may pass the new value before it is written.

Use these 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 TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA 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.

#### *17.3.3.2 Buffered Output Compare*

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTD0/TACH0 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The output compare value in the TIMA channel 0 registers initially controls the output on the PTD0/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to

synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, **PTD1/TACH1**, 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.*

### 17.3.4 Pulse Width Modulation (PWM)

By using the toggle-on-overflow feature with an output compare channel, the TIMA can generate a PWM signal. The value in the TIMA counter modulo registers determines the period of the PWM signal. The channel pin toggles when the counter reaches the value in the TIMA counter modulo registers. The time between overflows is the period of the PWM signal.

As Figure 17-3 shows, the output compare value in the TIMA channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMA to clear the channel pin on output compare if the state of the PWM pulse is logic 1. Program the TIMA to set the pin if the state of the PWM pulse is logic 0.

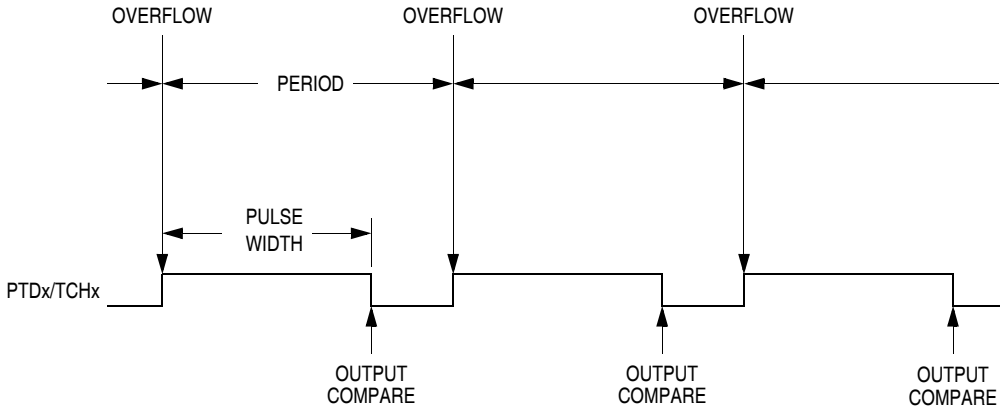



**Figure 17-3. PWM Period and Pulse Width**

The value in the TIMA 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 TIMA counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000 (see 17.8.1 TIMA Status and Control Register).

The value in the TIMA 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 TIMA channel registers produces a duty cycle of 128/256 or 50%.

#### 17.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in 17.3.4 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 TIMA channel registers.

An unsynchronized write to the TIMA 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 TIMA overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIMA may pass the new value before it is written to the TIMA channel registers.

Use these 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 TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA 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.*

#### 17.3.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTD0/TACH0 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The TIMA channel 0 registers initially control the pulse width on the PTD0/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the pulse width are the ones written to last. TASC0 controls and monitors the buffered PWM function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTD1/TACH1, 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.*

#### 17.3.4.3 PWM Initialization

To ensure correct operation when generating unbuffered or buffered PWM signals, use this initialization procedure:

1. In the TIMA status and control register (TASC):
   a. Stop the TIMA counter by setting the TIMA stop bit, TSTOP.
   b. Reset the TIMA counter prescaler by setting the TIMA reset bit, TRST.
2. In the TIMA counter modulo registers (TAMODH–TAMODL), write the value for the required PWM period.
3. In the TIMA channel x registers (TACHxH–TACHxL), write the value for the required pulse width.
4. In TIMA channel x status and control register (TASCx):
   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 17-2.
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (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 17-2.

   **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 TIMA status control register (TASC), clear the TIMA stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMA channel 0 registers (TACH0H–TACH0L) initially control the buffered PWM output. TIMA status control register 0 (TASC0) 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 TIMA 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 17.8.4 TIMA Channel Status and Control Registers.

## 17.4 Interrupts

These TIMA sources can generate interrupt requests:

- TIMA overflow flag (TOF) — The TOF bit is set when the TIM counter reaches the modulo value programmed in the TIMA counter modulo registers. The TIMA overflow interrupt enable bit, TOIE, enables TIMA overflow CPU interrupt requests. TOF and TOIE are in the TIMA status and control register.
- TIMA channel flags (CH1F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMA CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

## 17.5 Low-Power Modes

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

### 17.5.1 Wait Mode

The TIMA remains active after the execution of a WAIT instruction. In wait mode, the TIMA registers are not accessible by the CPU. Any enabled CPU interrupt request from the TIMA can bring the MCU out of wait mode.

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

### 17.5.2 Stop Mode

The TIMA is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions or the state of the TIMA counter. TIMA operation resumes when the MCU exits stop mode.

## 17.6 TIMA During Break Interrupts

A break interrupt stops the TIMA 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.

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.

## 17.7 I/O Signals

Port D shares two of its pins with the TIMA. There is no external clock input to the TIMA prescaler. The two TIMA channel I/O pins are PTD0/TACH0 and PTD1/TACH1. See Chapter 12 Input/Output (I/O) Ports (PORTS).

### 17.7.1 TIMA Channel I/O Pins (PTD0/TACH0, PTD1/TACH1)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTD0/TACH0 and PTD1/TACH1 can be configured as buffered output compare or buffered PWM pins.

## 17.8 I/O Registers

These I/O registers control and monitor TIMA operation:

- TIMA status and control register, TASC
- TIMA control registers, TACNTH–TACNTL
- TIMA counter modulo registers, TAMODH–TAMODL
- TIMA channel status and control registers, TASC0 and TASC1
- TIMA channel registers, TACH0H–TACH0L and TACH1H–TACH1L

### 17.8.1 TIMA Status and Control Register

The TIMA status and control register (TASC):

- Enables TIMA overflow interrupts
- Flags TIMA overflows
- Stops the TIMA counter
- Resets the TIMA counter
- Prescales the TIMA counter clock

Address: $0020

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | TOF | TOIE | TSTOP | 0 | R | PS2 | PS1 | PS0 |
| Write: | 0 | | | TRST | | | | |
| Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 17-4. TIMA Status and Control Register (TASC)**

**TOF — TIMA Overflow Flag Bit**

This read/write flag is set when the TIMA counter reaches the modulo value programmed in the TIMA counter modulo registers. Clear TOF by reading the TIMA status and control register when TOF is set and then writing a 0 to TOF. If another TIMA 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 = TIMA counter has reached modulo value
0 = TIMA counter has not reached modulo value

**TOIE — TIMA Overflow Interrupt Enable Bit**

This read/write bit enables TIMA overflow interrupts when the TOF bit becomes set. Reset clears the TOIE bit.

1 = TIMA overflow interrupts enabled
0 = TIMA overflow interrupts disabled

### TSTOP — TIMA Stop Bit

This read/write bit stops the TIMA counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the TIMA counter until software clears the TSTOP bit.

1 = TIMA counter stopped
0 = TIMA counter active

***NOTE***

*Do not set the TSTOP bit before entering wait mode if the TIMA 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 TSTOP is cleared.*

*When using TSTOP to stop the timer counter, see if any timer flags are set. If a timer flag is set, it must be cleared by clearing TSTOP, then clearing the flag, then setting TSTOP again.*

### TRST — TIMA Reset Bit

Setting this write-only bit resets the TIMA counter and the TIMA prescaler. Setting TRST has no effect on any other registers. Counting resumes from $0000. TRST is cleared automatically after the TIMA counter is reset and always reads as 0. Reset clears the TRST bit.

1 = Prescaler and TIMA counter cleared
0 = No effect

***NOTE***

*Setting the TSTOP and TRST bits simultaneously stops the TIMA 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 TIMA counter as Table 17-1 shows. Reset clears the PS[2:0] bits.

**Table 17-1. Prescaler Selection**

| PS[2:0] | TIMA 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 | Unused |

### 17.8.2 TIMA Counter Registers

The two read-only TIMA counter registers contain the high and low bytes of the value in the TIMA counter. Reading the high byte (TACNTH) latches the contents of the low byte (TACNTL) into a buffer. Subsequent reads of TACNTH do not affect the latched TACNTL value until TACNTL is read. Reset clears the TIMA counter registers. Setting the TIMA reset bit (TRST) also clears the TIMA counter registers.

***NOTE***

*If TACNTH is read during a break interrupt, be sure to unlatch TACNTL by reading TACNTL before exiting the break interrupt. Otherwise, TACNTL retains the value latched during the break.*

Register Name and Address: TACNTH — $0021

| | 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 |

Register Name and Address: TACNTL — $0022

| | 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 17-5. TIMA Counter Registers (TACNTH and TACNTL)**

### 17.8.3 TIMA Counter Modulo Registers

The read/write TIMA modulo registers contain the modulo value for the TIMA counter. When the TIMA counter reaches the modulo value, the overflow flag (TOF) becomes set, and the TIMA counter resumes counting from $0000 at the next timer clock. Writing to the high byte (TMODH) inhibits the TOF bit and overflow interrupts until the low byte (TMODL) is written. Reset sets the TIMA counter modulo registers.

Register Name and Address: TAMODH — $0023

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>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 |

Register Name and Address: TAMODL — $0024

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>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 17-6. TIMA Counter Modulo Registers (TMODH and TMODL)**

***NOTE***

*Reset the TIMA counter before writing to the TIMA counter modulo registers.*

### 17.8.4 TIMA Channel Status and Control Registers

Each of the TIMA 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 TIMA overflow
- Selects 0% and 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Register Name and Address: TASC0 — $0025

| | 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 |

Register Name and Address: TASC1 — $0028

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| Write: | 0 | | R | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 17-7. TIMA Channel Status and Control Register (TASC0–TASC1)**

**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 TIMA counter registers matches the value in the TIMA channel x registers.

When CHxIE = 1, clear CHxF by reading TIMA 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 TIMA 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 TIMA channel 0.

Setting MS0B disables the channel 1 status and control register and reverts TACH1 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 17-2.

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 once PWM, input capture, or output compare operation is enabled (see Table 17-2). 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 TIMA status and control register (TASC).*

**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, and pin PTD0/TACH0 or pin PTD1/TACH1 is available as a general-purpose I/O pin. However, channel x is at a state determined by these bits and becomes transparent to the respective pin when PWM, input capture, or output compare mode is enabled. Table 17-2 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

**Table 17-2. 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 |

**NOTE**

*Before enabling a TIMA channel register for input capture operation, make sure that the PTDx/TACHx pin is stable for at least two bus clocks.*

**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 TIMA 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 TIMA counter overflow.

0 = Channel x pin does not toggle on TIMA counter overflow.

**NOTE**

*When TOVx is set, a TIMA 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 and clear output on compare is selected, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100 percent. As Figure 17-8 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at 100 percent duty cycle level until the cycle after CHxMAX is cleared.

**NOTE**

*The PWM 100 percent duty cycle is defined as output high all of the time. To generate the 100 percent duty cycle, use the CHxMAX bit in the TSCx register. The PWM 0 percent duty cycle is defined as output low all of the time. To generate the 0 percent duty cycle, select clear output on compare and then clear the TOVx bit (CHxMAX = 0).*

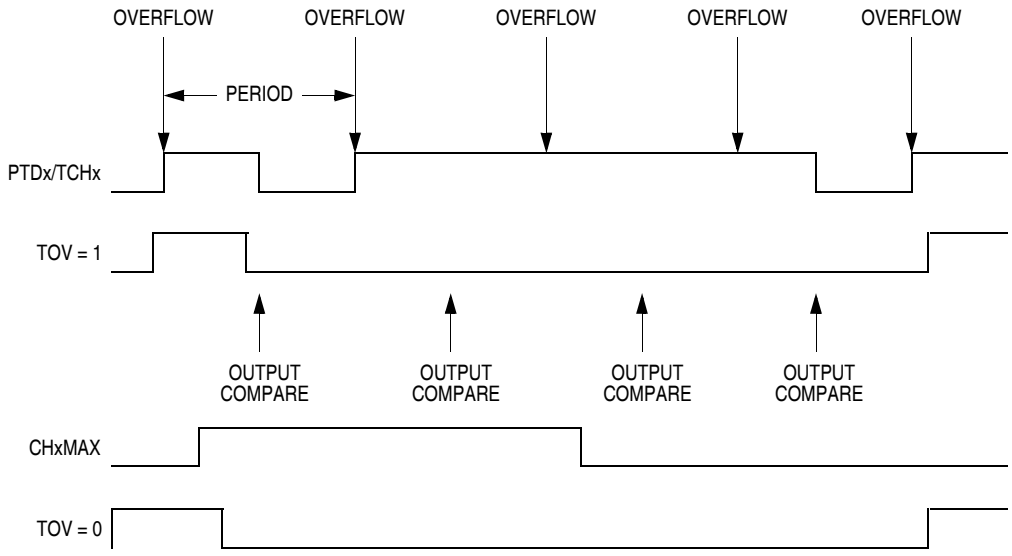



**Figure 17-8. CHxMAX Latency**

### 17.8.5 TIMA Channel Registers

These read/write registers contain the captured TIMA counter value of the input capture function or the output compare value of the output compare function. The state of the TIMA channel registers after reset is unknown.

In input capture mode (MSxB–MSxA = 0:0), reading the high byte of the TIMA channel x registers (TACHxH) inhibits input captures until the low byte (TACHxL) is read.

In output compare mode (MSxB–MSxA ≠ 0:0), writing to the high byte of the TIMA channel x registers (TACHxH) inhibits output compares and the CHxF bit until the low byte (TACHxL) is written.

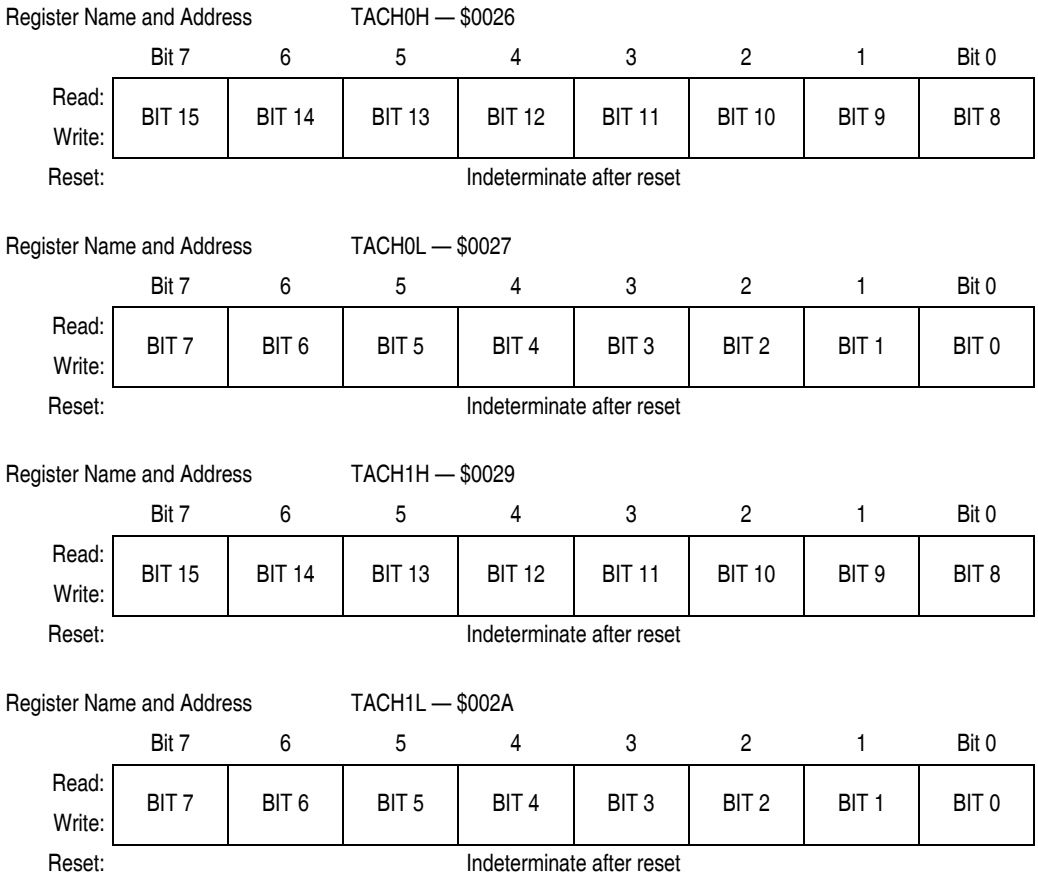



**Figure 17-9. TIMA Channel Registers (TACH0H/L–TACH1H/L)**

# Chapter 18
# Timer Interface B (TIMB) Module

## 18.1 Introduction

This section describes the timer interface B module (TIMB). The TIMB is a 2-channel timer that provides a timing reference with input capture, output compare, and pulse width modulation (PWM) functions. Figure 18-2 is a block diagram of the TIMB.

For further information regarding timers on M68HC08 family devices, please consult the *HC08 Timer Reference Manual*, Freescale document order number TIM08RM/AD.

## 18.2 Features

Features include:

- 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 PWM signal generation
- Programmable TIMB clock input
  - 7-frequency internal bus clock prescaler selection
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMB counter stop and reset bits

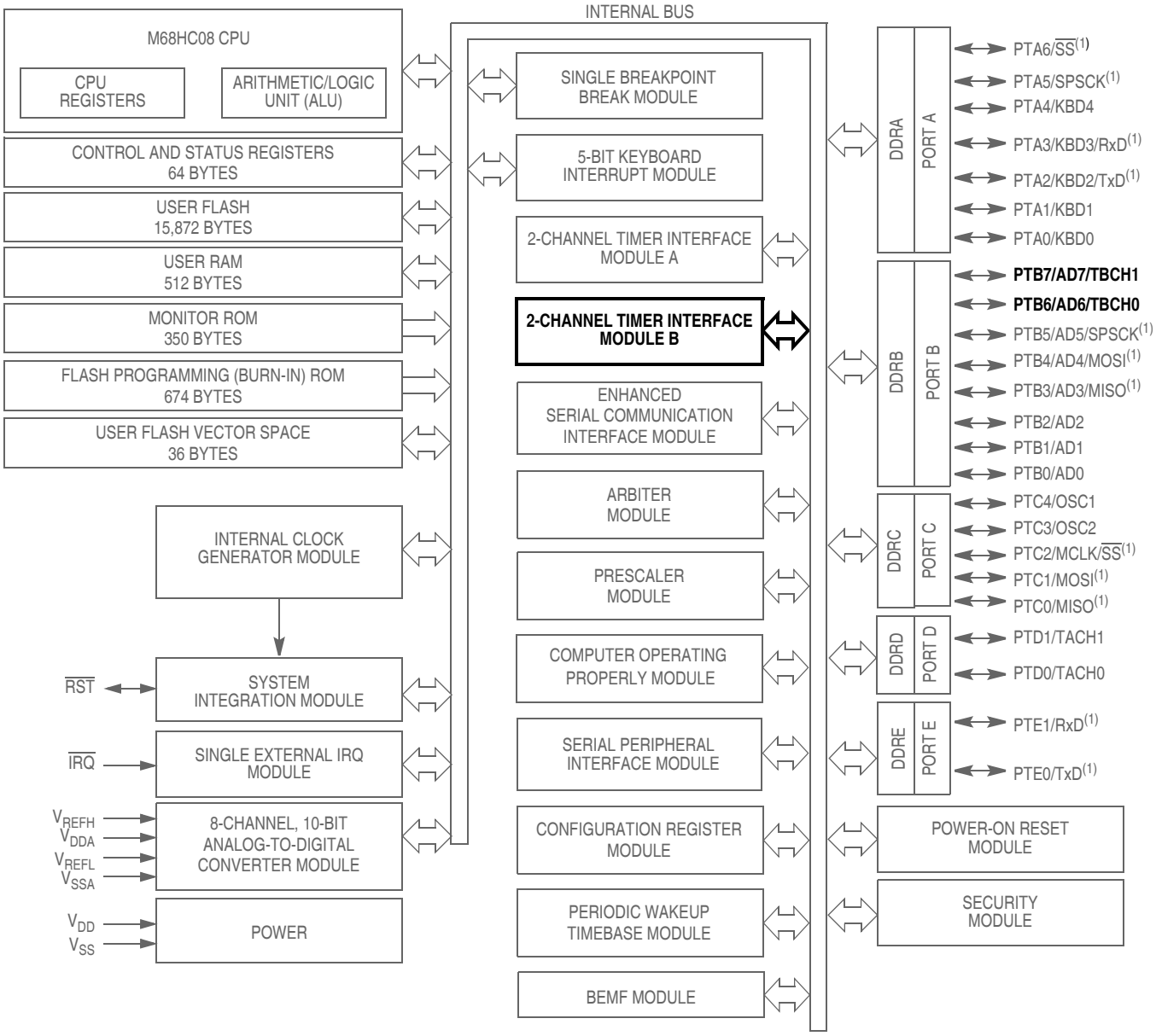
## 18.3 Functional Description

Figure 18-2 shows the TIMB structure. The central component of the TIMB is the 16-bit TIMB counter that can operate as a free-running counter or a modulo up-counter. The TIMB counter provides the timing reference for the input capture and output compare functions. The TIMB counter modulo registers, TBMODH–TBMODL, control the modulo value of the TIMB counter. Software can read the TIMB counter value at any time without affecting the counting sequence.

The two TIMB channels are programmable independently as input capture or output compare channels.

### 18.3.1 TIMB Counter Prescaler

The TIMB clock source can be 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 TIMB status and control register select the TIMB clock source.




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

**Figure 18-1. Block Diagram Highlighting TIMB Block and Pins**

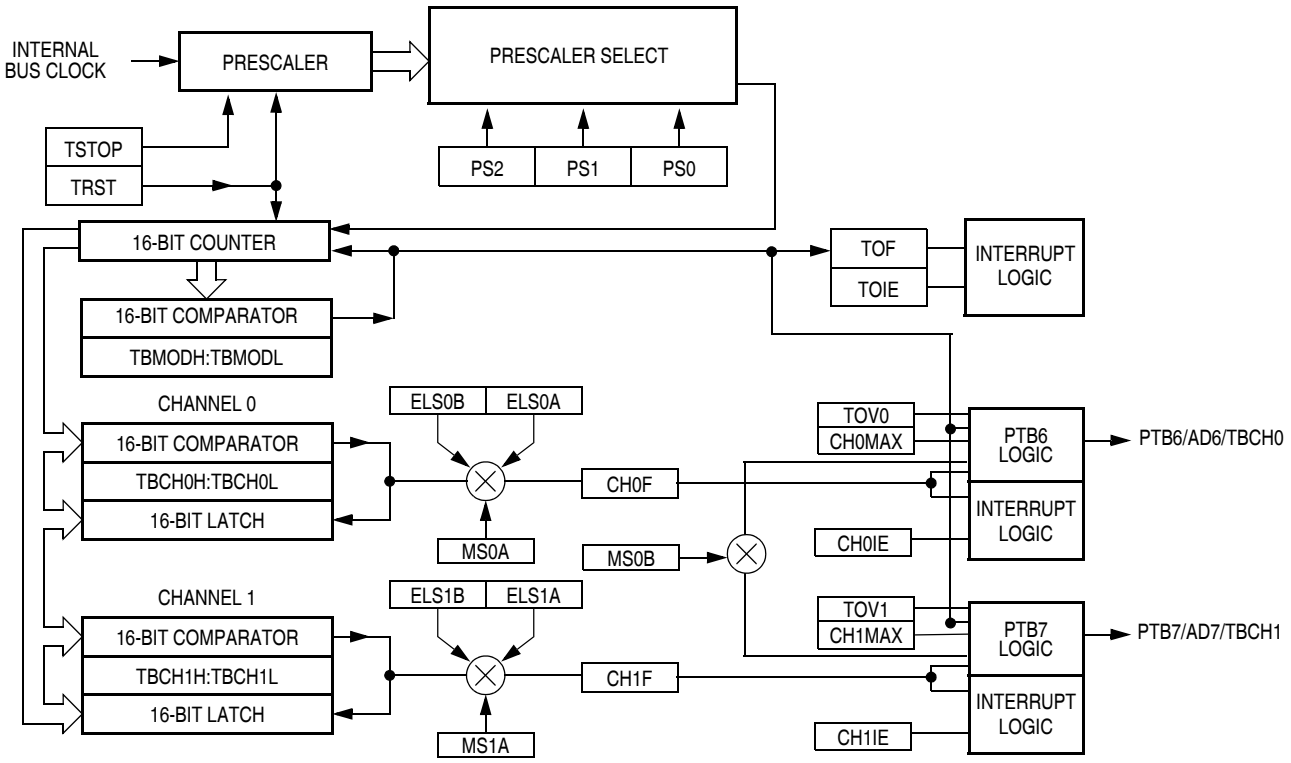



**Figure 18-2. TIMB Block Diagram**

### 18.3.2 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 TBSC0 through TBSC1 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMB latches the contents of the TIMB counter into the TIMB channel registers, TBCHxH–TBCHxL. Input captures can generate TIMB 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 TIMB channel status and control register (TBCHxH–TBCHxL, see 18.8.5 TIMB Channel Registers) on each proper signal transition regardless of whether the TIMB channel flag (CH0F–CH1F in TBSC0–TBSC1 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 2 bus cycles after 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 18.8.5 TIMB 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 register (TBCHxH–TBCHxL).

### 18.3.3 Output Compare

With the output compare function, the TIMB 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 TIMB can set, clear, or toggle the channel pin. Output compares can generate TIMB 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 TIMB channel registers.

An unsynchronized write to the TIMB 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 TIMB overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIMB may pass the new value before it is written.

Use these 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 TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB 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 PTB6/AD6/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The output compare value in the TIMB channel 0 registers initially controls the output on the PTB6/AD6/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the output after the TIMB overflows. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTB7/AD7/TBCH1, 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 TIMB can generate a PWM signal. The value in the TIMB counter modulo registers determines the period of the PWM signal. The channel pin toggles when the counter reaches the value in the TIMB counter modulo registers. The time between overflows is the period of the PWM signal.

As Figure 18-3 shows, the output compare value in the TIMB channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMB to clear the channel pin on output compare if the state of the PWM pulse is logic 1. Program the TIMB to set the pin if the state of the PWM pulse is logic 0.

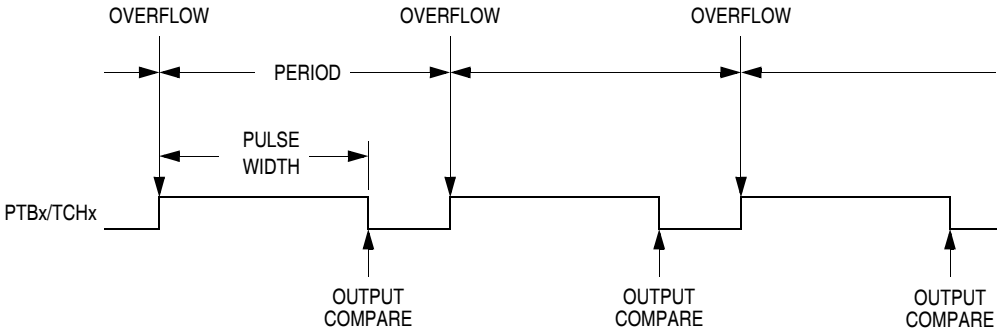



**Figure 18-3. PWM Period and Pulse Width**

The value in the TIMB 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 TIMB 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 TIMB Status and Control Register).

The value in the TIMB 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 TIMB channel registers produces a duty cycle of 128/256 or 50%.

#### 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 value currently in the TIMB channel registers.

An unsynchronized write to the TIMB 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 TIMB overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIMB may pass the new value before it is written to the TIMB channel registers.

Use these 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 TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB 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 PTB6/AD6/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The TIMB channel 0 registers initially control the pulse width on the PTB6/AD6/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the pulse width are the ones written to last. TBSC0 controls and monitors the buffered PWM function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTB7/AD7/TBCH1, 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 this initialization procedure:

1. In the TIMB status and control register (TBSC):
   a. Stop the TIMB counter by setting the TIMB stop bit, TSTOP.
   b. Reset the TIMB counter prescaler by setting the TIMB reset bit, TRST.
2. In the TIMB counter modulo registers (TBMODH–TBMODL), write the value for the required PWM period.
3. In the TIMB channel x registers (TBCHxH–TBCHxL), write the value for the required pulse width.
4. In TIMB channel x status and control register (TBSCx):
   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 (to clear output on compare) or 1:1 (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 TIMB status control register (TBSC), clear the TIMB stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMB channel 0 registers (TBCH0H–TBCH0L) initially control the buffered PWM output. TIMB status control register 0 (TBSC0) 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 TIMB 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 TIMB Channel Status and Control Registers.

## 18.4 Interrupts

These TIMB sources can generate interrupt requests:

- TIMB overflow flag (TOF) — The TOF bit is set when the TIMB counter reaches the modulo value programmed in the TIMB counter modulo registers. The TIMB overflow interrupt enable bit, TOIE, enables TIMB overflow CPU interrupt requests. TOF and TOIE are in the TIMB status and control register.
- TIMB channel flags (CH1F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMB CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

## 18.5 Low-Power Modes

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

### 18.5.1 Wait Mode

The TIMB remains active after the execution of a WAIT instruction. In wait mode, the TIMB registers are not accessible by the CPU. Any enabled CPU interrupt request from the TIMB can bring the MCU out of wait mode.

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

### 18.5.2 Stop Mode

The TIMB is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions or the state of the TIMB counter. TIMB operation resumes when the MCU exits stop mode.

## 18.6 TIMB During Break Interrupts

A break interrupt stops the TIMB 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.

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.

## 18.7 I/O Signals

Port B shares two of its pins with the TIMB. There is no external clock input to the TIMB prescaler. The two TIMB channel I/O pins are PTB6/AD6/TBCH0 and PTB7/AD7/TBCH1. See Chapter 12 Input/Output (I/O) Ports (PORTS).

### 18.7.1 TIMB Channel I/O Pins (PTB7/AD7/TBCH1–PTB6/AD6/TBCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTB6/AD6/TBCH0 and PTB7/AD7/TBCH1 can be configured as buffered output compare or buffered PWM pins.

## 18.8 I/O Registers

These I/O registers control and monitor TIMB operation:

- TIMB status and control register, TBSC
- TIMB control registers, TBCNTH–TBCNTL
- TIMB counter modulo registers, TBMODH–TBMODL
- TIMB channel status and control registers, TBSC0 and TBSC1
- TIMB channel registers, TBCH0H–TBCH0L and TBCH1H–TBCH1L

### 18.8.1 TIMB Status and Control Register

The TIMB status and control register:

- Enables TIMB overflow interrupts
- Flags TIMB overflows
- Stops the TIMB counter
- Resets the TIMB counter
- Prescales the TIMB counter clock

Address: $002B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | TOF | TOIE | TSTOP | 0 | R | PS2 | PS1 | PS0 |
| Write: | 0 | | | TRST | | | | |
| Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 18-4. TIMB Status and Control Register (TBSC)**

**TOF — TIMB Overflow Flag Bit**

This read/write flag is set when the TIMB counter reaches the modulo value programmed in the TIMB counter modulo registers. Clear TOF by reading the TIMB status and control register when TOF is set and then writing a 0 to TOF. If another TIMB 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 = TIMB counter has reached modulo value
0 = TIMB counter has not reached modulo value

**TOIE — TIMB Overflow Interrupt Enable Bit**

This read/write bit enables TIMB overflow interrupts when the TOF bit becomes set. Reset clears the TOIE bit.

1 = TIMB overflow interrupts enabled
0 = TIMB overflow interrupts disabled

### TSTOP — TIMB Stop Bit

This read/write bit stops the TIMB counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the TIMB counter until software clears the TSTOP bit.

1 = TIMB counter stopped
0 = TIMB counter active

***NOTE***

*Do not set the TSTOP bit before entering wait mode if the TIMB 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 TSTOP is cleared.*

*When using TSTOP to stop the timer counter, see if any timer flags are set. If a timer flag is set, it must be cleared by clearing TSTOP, then clearing the flag, then setting TSTOP again.*

### TRST — TIMB Reset Bit

Setting this write-only bit resets the TIMB counter and the TIMB prescaler. Setting TRST has no effect on any other registers. Counting resumes from $0000. TRST is cleared automatically after the TIMB counter is reset and always reads as 0. Reset clears the TRST bit.

1 = Prescaler and TIMB counter cleared
0 = No effect

***NOTE***

*Setting the TSTOP and TRST bits simultaneously stops the TIMB 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 TIMB counter as Table 18-1 shows. Reset clears the PS[2:0] bits.

**Table 18-1. Prescaler Selection**

| PS[2:0] | TIMB 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 | Unused |

### 18.8.2 TIMB Counter Registers

The two read-only TIMB counter registers contain the high and low bytes of the value in the TIMB counter. Reading the high byte (TBCNTH) latches the contents of the low byte (TBCNTL) into a buffer. Subsequent reads of TBCNTH do not affect the latched TBCNTL value until TBCNTL is read. Reset clears the TIMB counter registers. Setting the TIMB reset bit (TRST) also clears the TIMB counter registers.

***NOTE***

*If TBCNTH is read during a break interrupt, be sure to unlatch TBCNTL by reading TBCNTL before exiting the break interrupt. Otherwise, TBCNTL retains the value latched during the break.*

Register Name and Address: TBCNTH — $002C

| | 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 |

Register Name and Address: TBCNTL — $002D

| | 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 18-5. TIMB Counter Registers (TBCNTH and TBCNTL)**

### 18.8.3 TIMB Counter Modulo Registers

The read/write TIMB modulo registers contain the modulo value for the TIMB counter. When the TIMB counter reaches the modulo value, the overflow flag (TOF) becomes set, and the TIMB counter resumes counting from $0000 at the next timer clock. Writing to the high byte (TMODH) inhibits the TOF bit and overflow interrupts until the low byte (TMODL) is written. Reset sets the TIMB counter modulo registers.

Register Name and Address: TBMODH — $002E

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>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 |

Register Name and Address: TBMODL — $002F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>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 18-6. TIMB Counter Modulo Registers (TMODH and TMODL)**

***NOTE***

*Reset the TIMB counter before writing to the TIMB counter modulo registers.*

### 18.8.4 TIMB Channel Status and Control Registers

Each of the TIMB 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 TIMB overflow
- Selects 0% and 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Register Name and Address TBSC0 — $0030

| | 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 |

Register Name and Address TBSC1 — $0033

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| Write: | 0 | | R | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 18-7. TIMB Channel Status and Control Registers (TBSC0–TBSC1)**

**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 TIMB counter registers matches the value in the TIMB channel x registers.

When CHxIE = 1, clear CHxF by reading TIMB 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 TIMB 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 TIMB channel 0.

Setting MS0B disables the channel 1 status and control register and reverts TBCH1 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 once PWM, input capture, or output compare operation is enabled. 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 TIMB status and control register (TBSC).*

### 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 B, and pin PTBx/TBCHx is available as a general-purpose I/O pin. However, channel x is at a state determined by these bits and becomes transparent to the respective pin when PWM, input capture, or output compare mode is enabled. Table 18-2 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

**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 |

**NOTE**

*Before enabling a TIMB channel register for input capture operation, make sure that the* PTBx/TBCHx *pin is stable for at least two bus clocks.*

**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 TIMB 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 TIMB counter overflow.

0 = Channel x pin does not toggle on TIMB counter overflow.

**NOTE**

*When TOVx is set, a TIMB 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 and clear output on compare is selected, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100 percent. As Figure 18-8 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at 100 percent duty cycle level until the cycle after CHxMAX is cleared.

**NOTE**

*The PWM 100 percent duty cycle is defined as output high all of the time. To generate the 100 percent duty cycle, use the CHxMAX bit in the TSCx register. The PWM 0 percent duty cycle is defined as output low all of the time. To generate the 0 percent duty cycle, select clear output on compare and then clear the TOVx bit (CHxMAX = 0).*

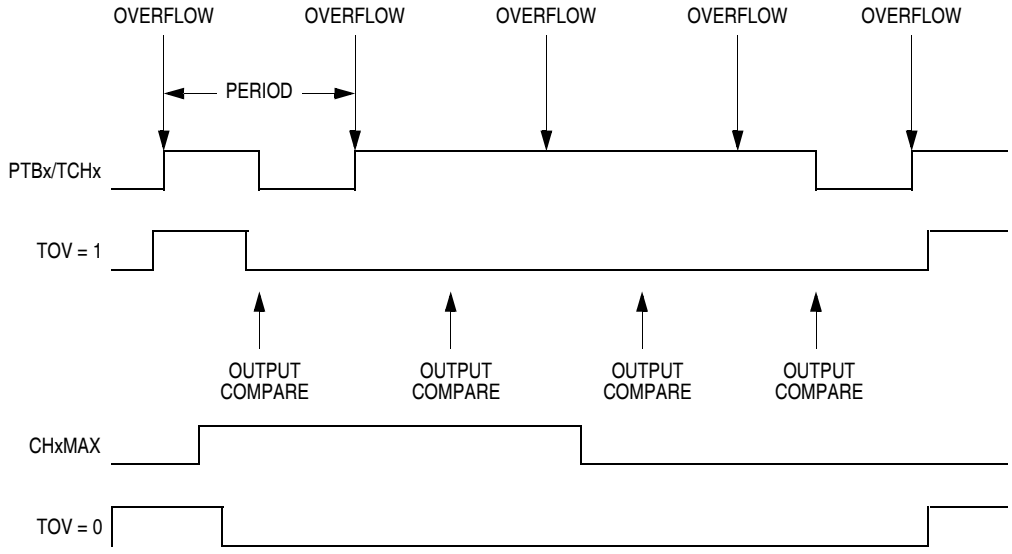



**Figure 18-8. CHxMAX Latency**

### 18.8.5 TIMB Channel Registers

These read/write registers contain the captured TIMB counter value of the input capture function or the output compare value of the output compare function. The state of the TIMB channel registers after reset is unknown.

In input capture mode (MSxB–MSxA = 0:0), reading the high byte of the TIMB channel x registers (TBCHxH) inhibits input captures until the low byte (TBCHxL) is read.

In output compare mode (MSxB–MSxA ≠ 0:0), writing to the high byte of the TIMB channel x registers (TBCHxH) inhibits output compares and the CHxF bit until the low byte (TBCHxL) is written.

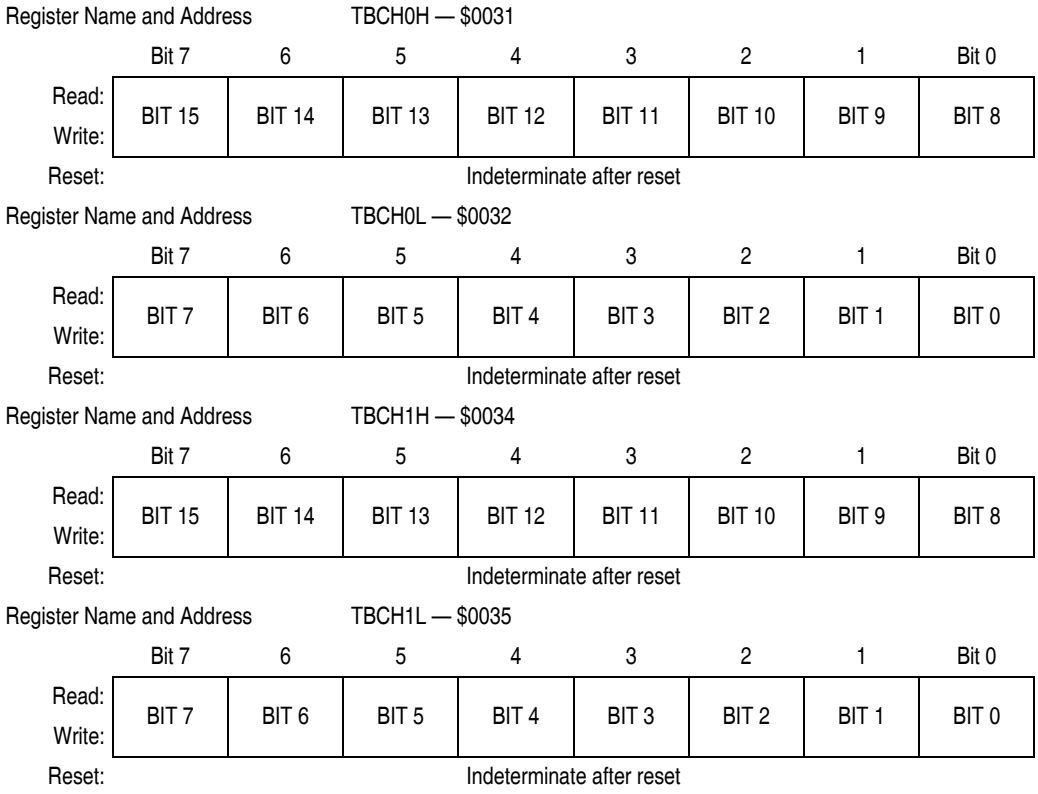



**Figure 18-9. TIMB Channel Registers (TBCH0H/L–TBCH1H/L)**

# Chapter 19
# Development Support

## 19.1 Introduction

This section describes the break module, the monitor module (MON), and the monitor mode entry methods.

## 19.2 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

### 19.2.1 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:

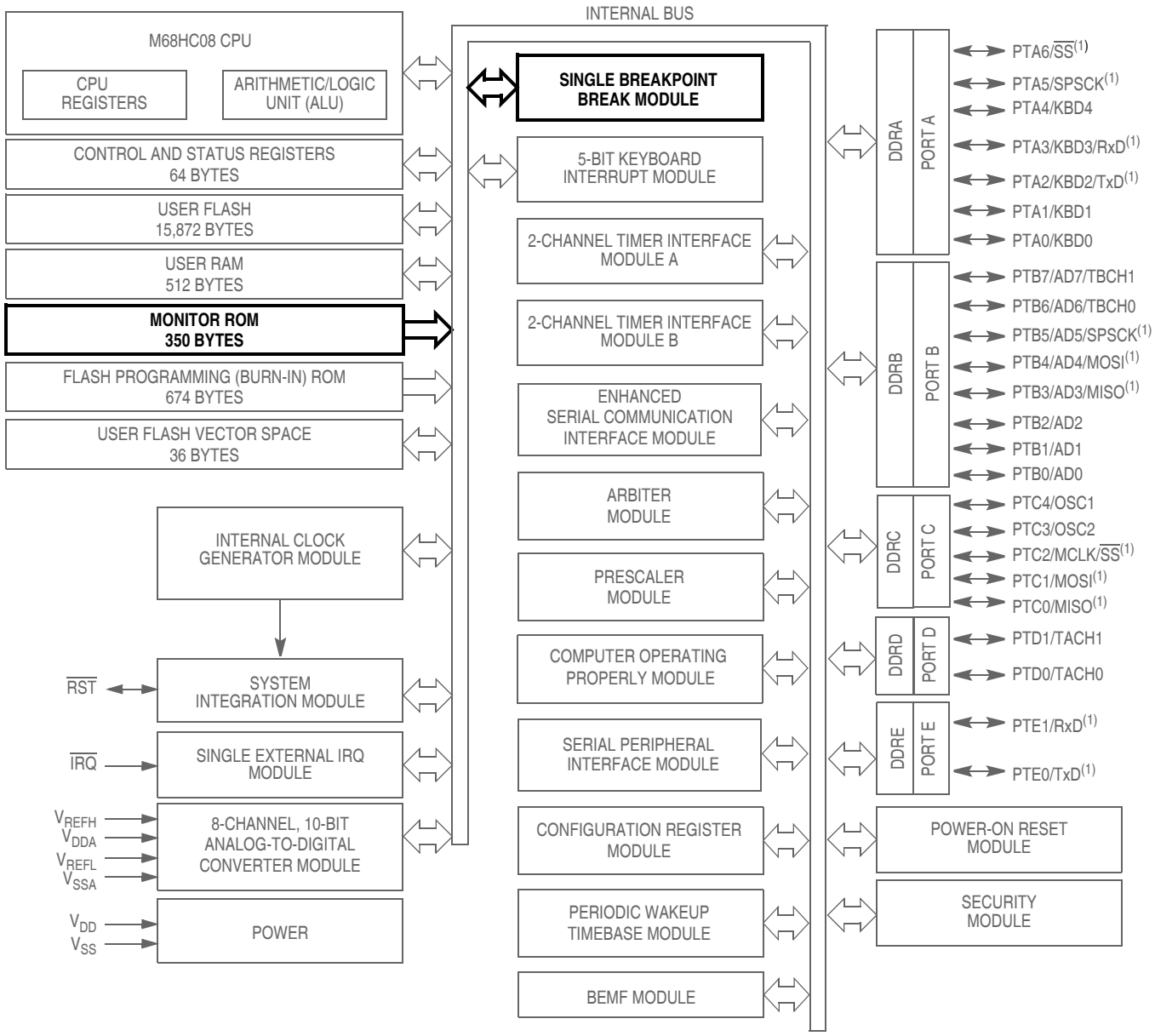
- A CPU generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a 1 to the BRKA bit in the break status and control register.

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 19-2 shows the structure of the break module.

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)




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

**Figure 19-1. Block Diagram Highlighting BRK and MON Blocks**

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.

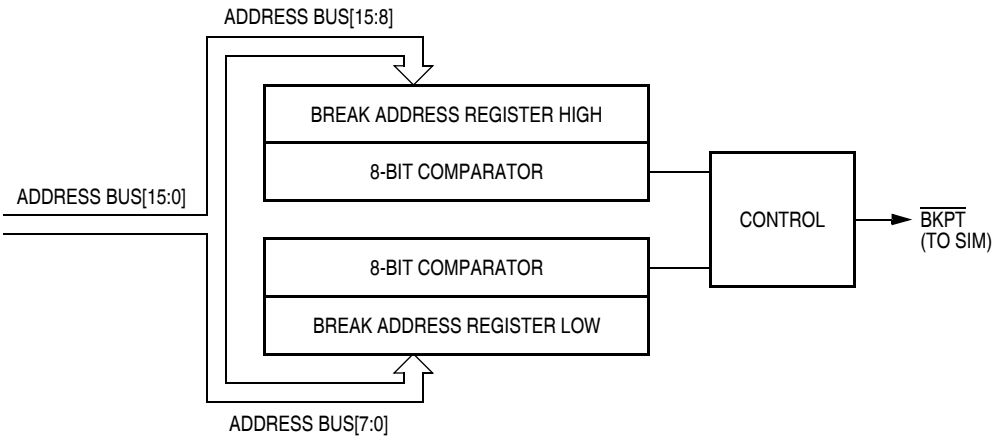



**Figure 19-2. Break Module Block Diagram**

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.*

#### *19.2.1.1 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 (SBFCR) enables software to clear status bits during the break state. See 14.8.3 SIM Break Flag Control Register and the “Break Interrupts” subsection for each module.

#### *19.2.1.2 TIM During Break Interrupts*

A break interrupt stops the timer counter and inhibits input captures.

#### *19.2.1.3 COP During Break Interrupts*

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

### 19.2.2 Break Module Registers

These registers control and monitor operation of the break module:

- Break status and control register (BSCR)
- Break address register high (BRKH)
- Break address register low (BRKL)
- Break status register (SBSR)
- Break flag control register (SBFCR)

### 19.2.2.1 Break Status and Control Register

The break status and control register (BSCR) 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 19-3. Break Status and Control Register (BSCR)**

**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

### 19.2.2.2 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 19-4. 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 19-5. Break Address Register Low (BRKL)**

### 19.2.2.3 Break Status Register

The break status register (SBSR) contains a flag to indicate that a break caused an exit from wait mode. This register is only used in emulation mode.

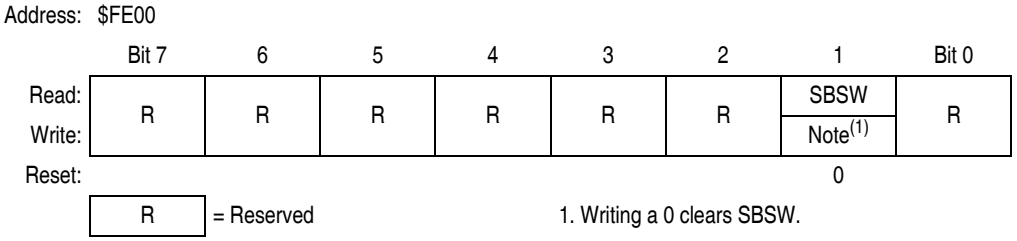



**Figure 19-6. Break Status Register (SBSR)**

**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

### 19.2.2.4 Break Flag Control Register

The break control register (SBFCR) contains a bit that enables software to clear status bits while the MCU is in a break state.

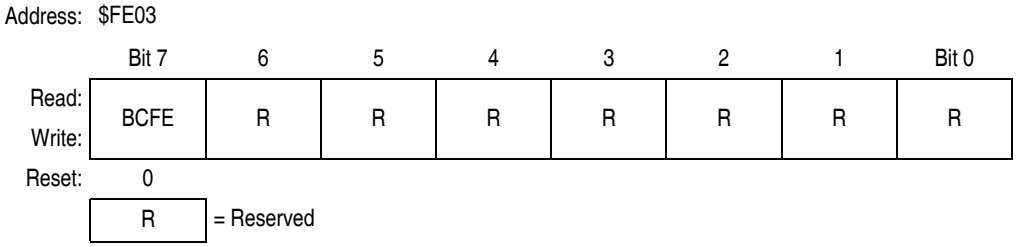



**Figure 19-7. Break Flag Control Register (SBFCR)**

**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

## 19.2.3 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.

## 19.3 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 (9600 @ 2.4576-MHz internal operating frequency)
- Execution of code in random-access memory (RAM) or FLASH
- FLASH memory security feature(1)
- FLASH memory programming interface
- Use of external 9.8304 MHz oscillator or ICG to generate internal operating frequency of 2.4576 MHz
- 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 $\overline{IRQ}$

### 19.3.1 Functional Description

Figure 19-8 shows a simplified diagram of the monitor mode.

The monitor module receives and executes commands from a host computer. Figure 19-9, Figure 19-10, and Figure 19-11 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 19-1 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 9600 baud provided one of the following sets of conditions is met:

- If $FFFE and $FFFF are erased or programmed:
  - The external clock is 9.8304 MHz (9600 baud)
  - $\overline{IRQ} = V_{TST}$
- If $FFFE and $FFFF contain $FF (erased state):
  - The external clock is 9.8304 MHz (9600 baud)
  - $\overline{IRQ} = V_{DD}$ (this can be implemented through the internal $\overline{IRQ}$ pullup)
- If $FFFE and $FFFF contain $FF (erased state):
  - The ICG clock is nominal 2.45 MHz (nominal 9600 baud)
  - $\overline{IRQ} = V_{SS}$

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

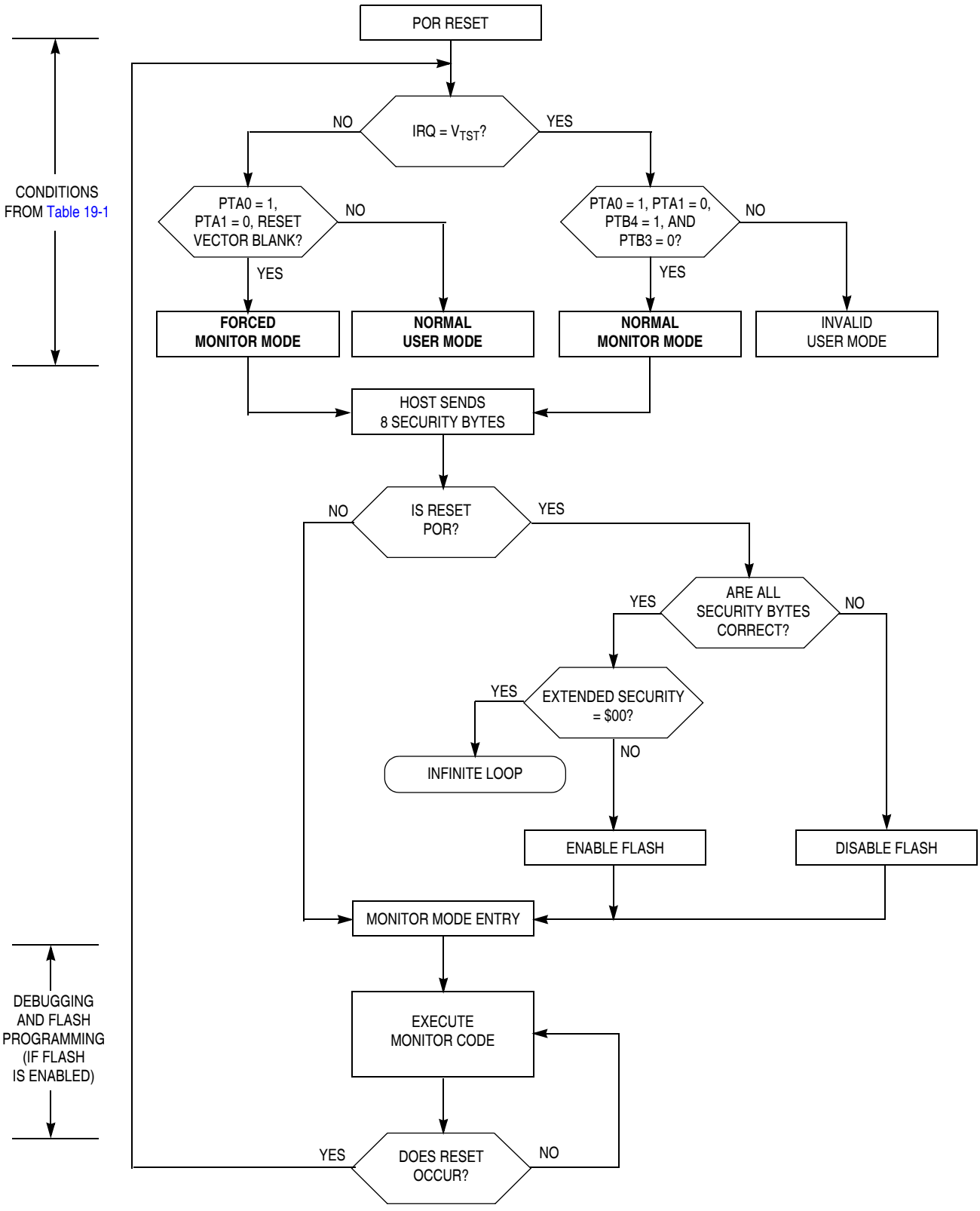



**Figure 19-8. Simplified Monitor Mode Entry Flowchart**

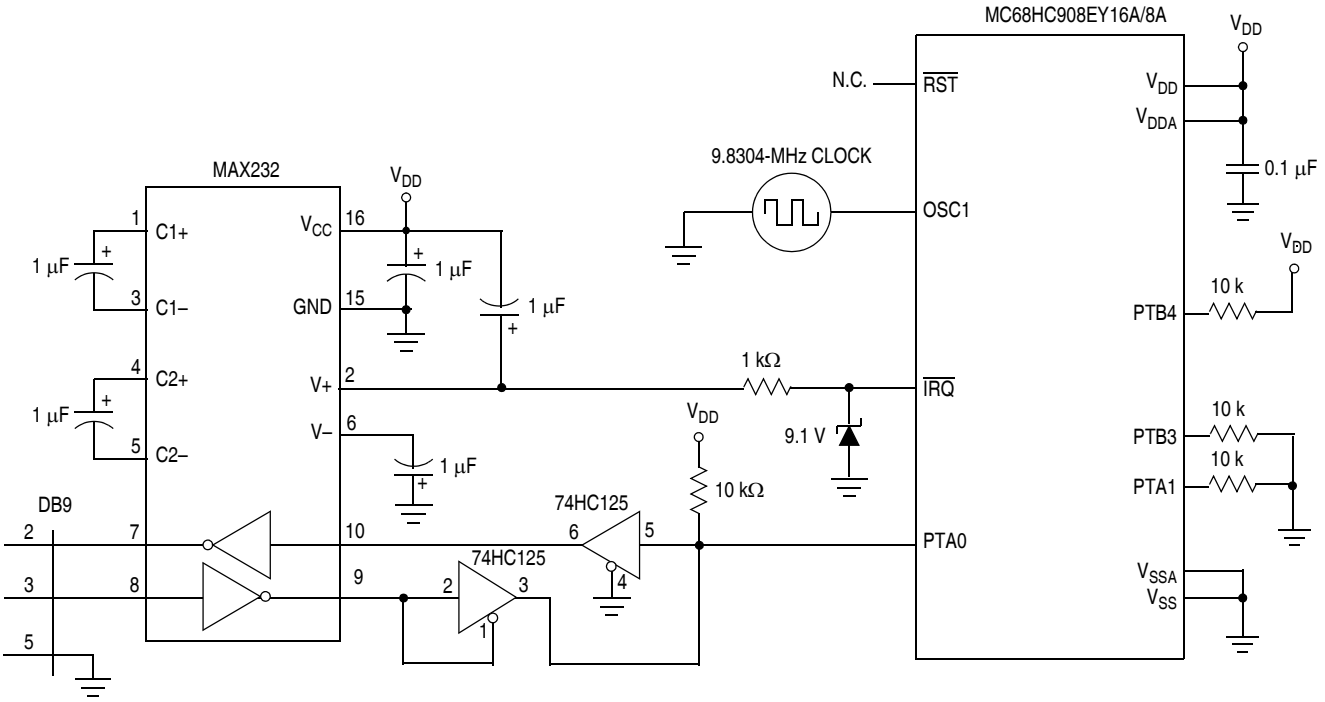



**Figure 19-9. Normal Monitor Mode Circuit**

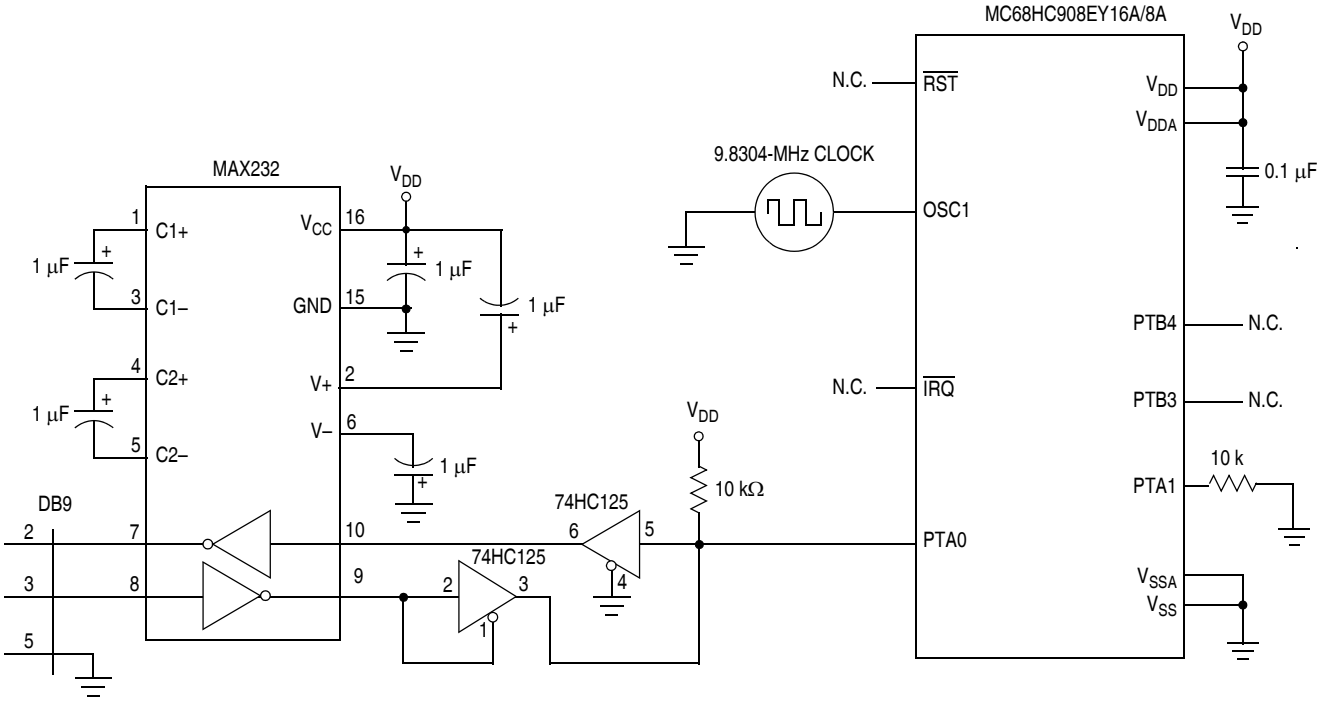



**Figure 19-10. Forced Monitor Mode ($V_{\overline{IRQ}} = V_{DD}$)**

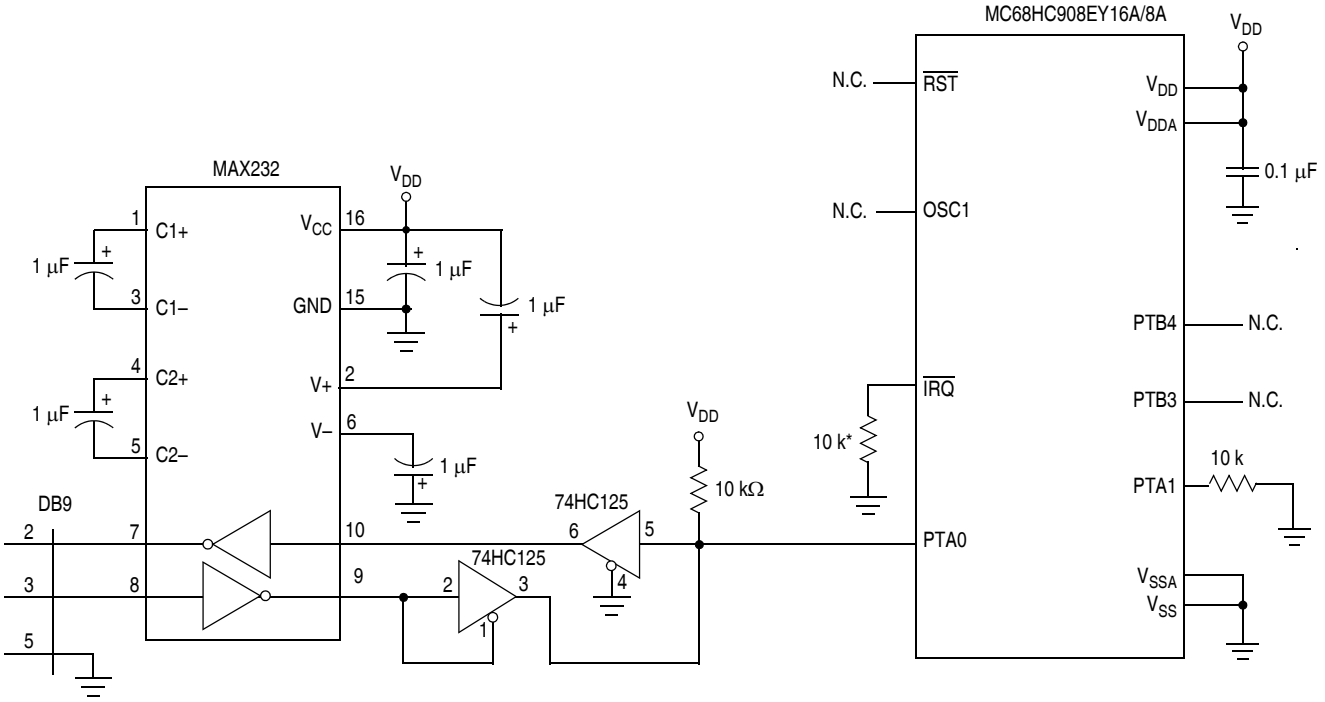



**Figure 19-11. Forced Monitor Mode ($V_{\overline{IRQ}} = V_{SS}$)**

Enter monitor mode with pin configuration shown in Table 19-1 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 19.3.2 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.

### *19.3.1.1 Normal Monitor Mode*

If $V_{TST}$ is applied to $\overline{IRQ}$ upon monitor mode entry, the internal operating frequency is a divide-by-four of the input clock.

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.

**NOTE**

*While the voltage on $\overline{IRQ}$ is at $V_{TST}$, the ICG module is bypassed and the external square-wave clock becomes the clock source. Dropping $\overline{IRQ}$ to below $V_{TST}$ will remove the bypass and the MCU will revert to the clock source selected by the ICG (a determined by the settings in the ICG registers).*

**Table 19-1. Monitor Mode Signal Requirements and Options**

| Mode | $\overline{IRQ}$ | $\overline{RST}$ | Reset Vector | Serial Communication | | Mode Selection | | ICG | COP | Communication Speed | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | PTA0 | PTA1 | PTB4 | PTB3 | | | External Clock | $f_{OP}$ | Baud Rate |
| Normal Monitor | $V_{TST}$ | $V_{DD}$ or $V_{TST}$ | X | 1 | 0 | 1 | 0 | OFF | Disabled | 9.8304 MHz | 2.4576 MHz | 9600 |
| Forced Monitor | $V_{DD}$ | $V_{DD}$ | $FFFF (blank) | 1 | 0 | X | X | OFF | Disabled | 9.8304 MHz | 2.4576 MHz | 9600 |
| | $V_{SS}$ | $V_{DD}$ | | 1 | 0 | X | X | ON | Disabled | — | Nominal 2.45 MHz | Nominal 9600 |
| User | $V_{DD}$ or $V_{SS}$ | $V_{DD}$ or $V_{TST}$ | Not $FFFF | X | X | X | X | ON | Enabled | — | Nominal 1.6 MHz | X |
| MON08 Function [Pin No.] | $V_{TST}$ [6] | $\overline{RST}$ [4] | — | COM [8] | SSEL [10] | MOD0 [12] | MOD1 [14] | — | — | 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 9600 except the forced monitor $\overline{IRQ} = V_{SS}$ case. Baud rate using external oscillator is bus frequency / 256.
4. X = don't care
5. $\overline{RST}$ column indicates the state of $\overline{RST}$ after the monitor entry.
6. 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 | PTB3 |
| OSC1 | 13 | 14 | PTB4 |
| $V_{DD}$ | 15 | 16 | NC |

### *19.3.1.2 Forced Monitor Mode*

If entering monitor mode without high voltage on $\overline{IRQ}$, then PTA1, PTB3, and PTB4 pin requirements and conditions are not in effect. This is to reduce circuit requirements when performing in-circuit programming.

If the reset vector is blank and monitor mode is entered without $V_{TST}$ on $\overline{IRQ}$, the MCU will see an additional reset cycle after the initial power-on reset (POR). The MCU will initially come out of reset in user mode. Internal circuitry monitors the reset vector fetches and will assert an internal reset if it detects the reset vector is erased ($FFFF).

Once the MCU enters this mode any reset other than a POR will automatically force the MCU to come back to the forced monitor mode. Exiting the forced monitor mode requires a POR. Pulling $\overline{RST}$ low will

not exit monitor mode in this situation. Once the reset vector has been programmed, the traditional method of applying a voltage, $V_{TST,}$ to $\overline{IRQ}$ must be used to re-enter monitor mode after the next POR.

When the forced monitor mode is entered the COP is always disabled regardless of the state of $\overline{IRQ}$ or $\overline{RST}$.

With $V_{DD}$ on $\overline{IRQ}$, an external oscillator of 9.8034 MHz is required for a baud rate of 9600, as the internal internal operating frequency is automatically set to the external frequency divided by four.

With $V_{SS}$ on $\overline{IRQ}$ at the monitor entry, the ICG is on. In this case, the internal operating frequency is a nominal 2.45 MHz and the baud rate is a nominal 9600.

#### *19.3.1.3 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 19-2 summarizes the differences between user mode and monitor mode.

**Table 19-2. 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 |

#### *19.3.1.4 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.

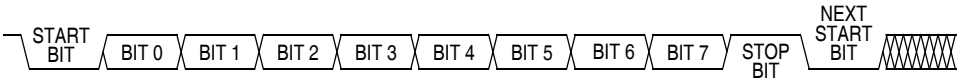



**Figure 19-12. Monitor Data Format**

#### *19.3.1.5 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.

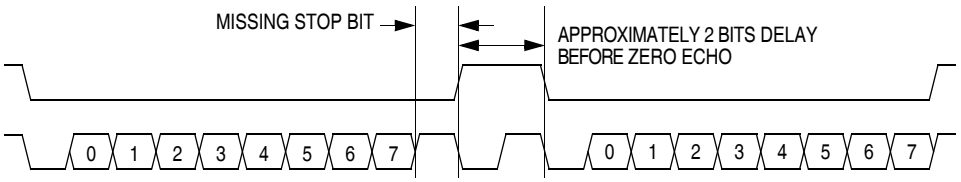



**Figure 19-13. Break Transaction**

### 19.3.1.6 Baud Rate

The monitor communication baud rate is controlled by the frequency of the external or internal oscillator and the state of the appropriate pins as shown in Table 19-1.

Table 19-1 also lists external frequencies required to achieve a standard baud rate of 9600 bps. The effective baud rate is the internal operating frequency divided by 256. If using a crystal as the clock source, be aware of the upper frequency limit that the internal clock module can handle. See 20.6 5V Control Timing for this limit.

### 19.3.1.7 Commands

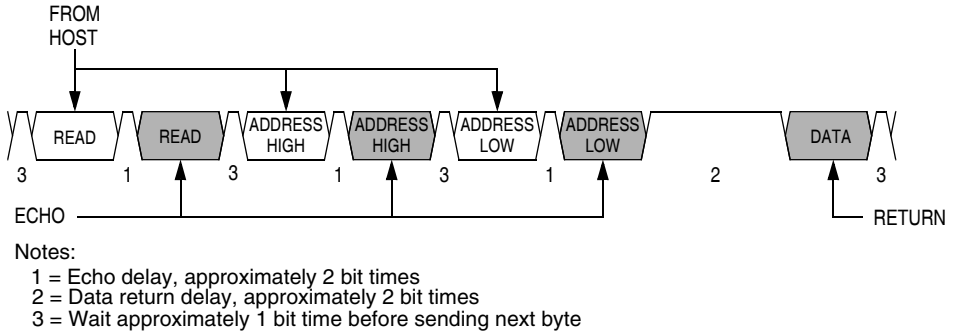
The monitor ROM firmware uses these commands:

- 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. 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.

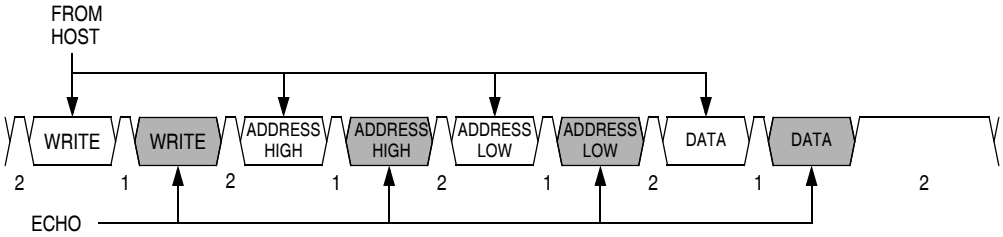
**NOTE**

*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 = Wait approximately 1 bit time before sending next byte

**Figure 19-14. Read Transaction**




Notes:
1 = Echo delay, approximately 2 bit times
2 = Wait approximately 1 bit time before sending next byte

**Figure 19-15. Write Transaction**

A brief description of each monitor mode command is given in Table 19-3 through Table 19-8.

**Table 19-3. 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**

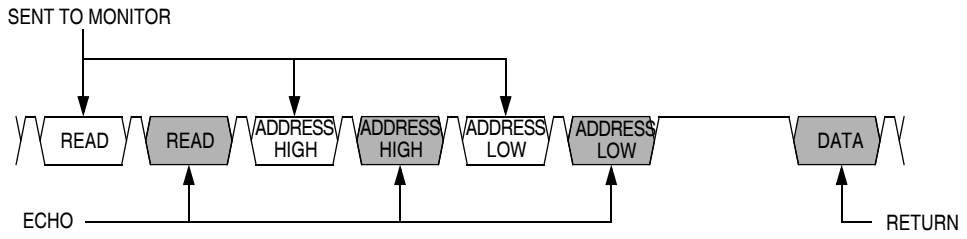



**Table 19-4. WRITE (Write Memory) Command**

| | |
|---|---|
| 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**

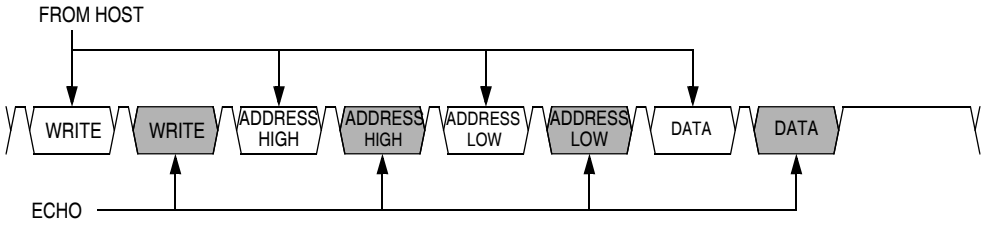



**Table 19-5. IREAD (Indexed Read) Command**

| | |
|---|---|
| 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**

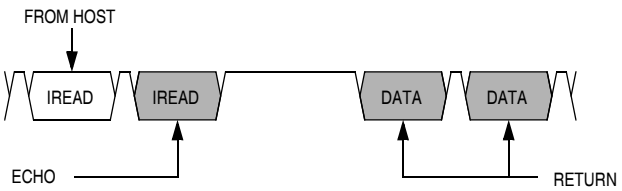
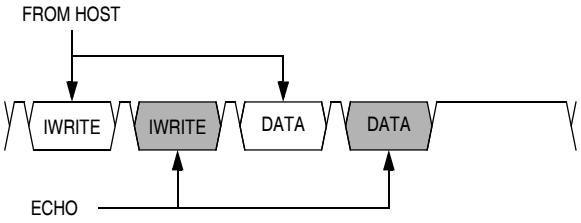


**Table 19-6. IWRITE (Indexed Write) Command**

| | |
|---|---|
| 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 19-7. 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** | |

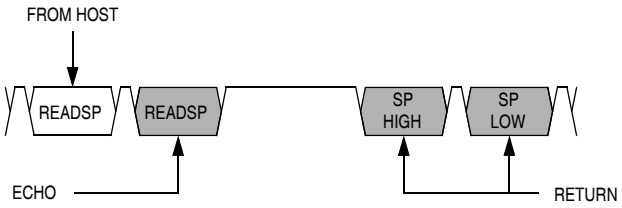
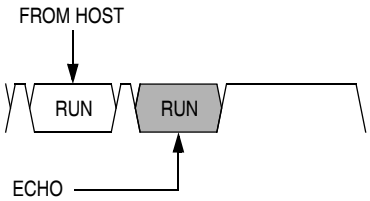



**Table 19-8. RUN (Run User Program) Command**

| | |
|---|---|
| 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.

| | |
|---|---|
| | SP |
| HIGH BYTE OF INDEX REGISTER | SP + 1 |
| CONDITION CODE REGISTER | SP + 2 |
| ACCUMULATOR | SP + 3 |
| LOW BYTE OF INDEX REGISTER | SP + 4 |
| HIGH BYTE OF PROGRAM COUNTER | SP + 5 |
| LOW BYTE OF PROGRAM COUNTER | SP + 6 |
| | SP + 7 |

**Figure 19-16. Stack Pointer at Monitor Mode Entry**

### 19.3.2 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 19-17.

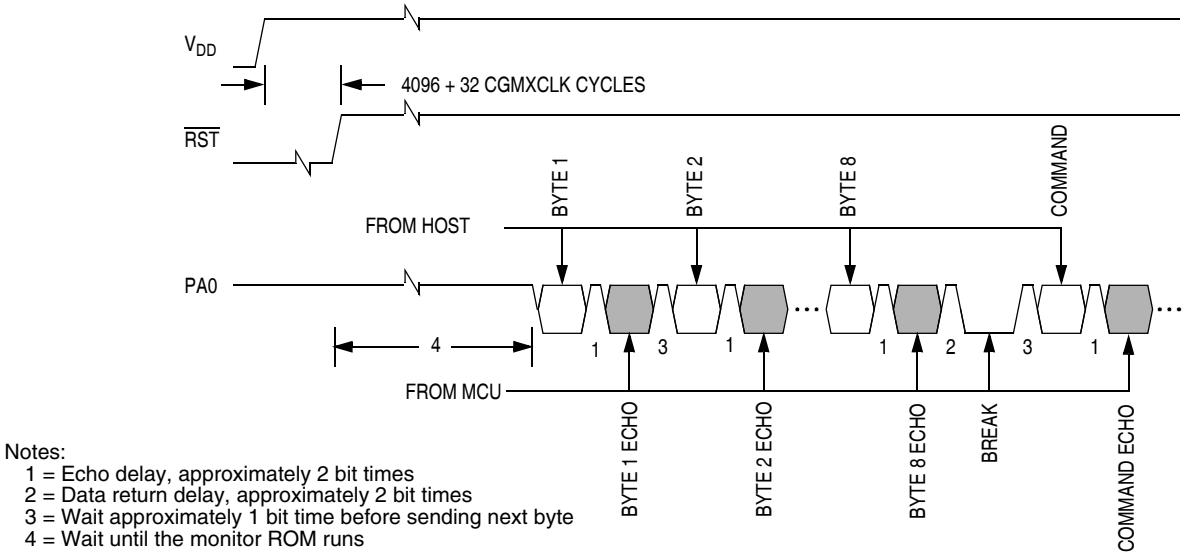



**Figure 19-17. Monitor Mode Entry Timing**

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.*

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).

### 19.3.3 Extended Security

In addition to the above security, a more secure feature called extended security is implemented in the MCU to further protect FLASH contents. Once this extended security is enabled, the MCU does not allow any user to enter the monitor mode even when all 8 security bytes are matched correctly. The extended security feature can be enabled by programming address $FDFF located in the user FLASH memory with data $00.

To unlock the extended security feature, the MCU must enter the monitor mode by failing the 8 byte security check. Then the FLASH must be mass-erased. This unlock process will erase the FLASH contents completely.

***NOTE***

*To avoid enabling the extended security unintentionally, the user must make sure that the user software does not contain data $00 at address $FDFF.*

## 19.4 Routines Supported in ROM

In the ROM, five routines are supported. Because the ROM has a jump table, the user does not call the routines with direct addresses. Therefore, the calling addresses will not change—even when the ROM code is updated in the future.

This section introduces each routine briefly. Details are discussed in later sections.

- **GetByte** — This routine is used to receive a byte serially on the general-purpose I/O PTA0. The receiving baud rate is the same as the baud rate used in monitor mode. In the GetByte routine, the GetBit routine is called to generate baud rates.
- **PutByte** — This routine is used to send a byte serially on the general-purpose I/O PTA0. The sending baud rate is the same as the baud rate specified in monitor mode.
- **Verify** — This routine is used to perform one of two options. Using the send-out option, this routine reads FLASH locations and sends the data out serially on the general-purpose I/O PTA0. Using the compare option, this routine compares the FLASH data against data in a specific RAM location, which is referred to as a DATA array. The DATA array locations and the variable locations required for this routine are at fixed memory addresses.

- **fProgram** — This routine is used to program a contiguous range of FLASH locations. Programming data is first loaded into the DATA array. The DATA array locations and the variable locations required for this routine are at fixed memory addresses. fProgram can be used when the internal operating frequency ($f_{op}$) is between 1.0 MHz and 8.0 MHz.
- **fErase** — This routine is used to erase either a page (64 bytes) or the whole array of FLASH. The variable locations required for this routine are at fixed memory addresses. This routine can be used when the internal operating frequency ($f_{op}$) is between 1.0 MHz and 8.0 MHz.

### 19.4.1 Variables Used in the Routines

The Verify, fProgram, and fErase routines require certain registers and/or RAM locations to be initialized before calling the routines in the user software. Table 19-9 shows variables used in the routines and their locations.

**Table 19-9. Variables and Their Locations**

| Location | Variable Name | Size (Bytes) | Description |
|---|---|---|---|
| $0040–$0048 | Reserved | 9 | Reserved for future use |
| $0049 | CPUSPD | 1 | CPU speed — the nearest integer of $f_{op}$ (in MHz) × 4; for example, if $f_{op}$ = 2.4576 MHz, CPUSPD = 10 |
| $004A:$004B | LADDR | 2 | Last address of a 16-bit range |
| $004C | DATA | Varies | First location of DATA array; DATA array size must match a programming or verifying range |

- CPUSPD — To set up proper delays used in the fProgram and fErase routines, a value indicating the internal operating frequency ($f_{op}$) must be stored at CPUSPD, which is located at RAM address $0049. The CPUSPD value is the nearest integer of $f_{op}$ (in MHz) times 4. For example, if $f_{op}$ is 4.2 MHz, the CPUSPD value is 17. If $f_{op}$ is 2.1 MHz, the CPUSPD value is 8. Setting a correct CPUSPD value is very important to program or erase the FLASH successfully.
- LADDR — A range specifies the FLASH locations to be read, verified, or programmed. The 16-bit value in RAM addresses $004A and $004B holds the last address of a range. The addresses $004A and $004B are the high and low bytes of the last address, respectively. LADDR is used for Verify and fProgram routines.
- DATA — DATA is the first location of the DATA array and is located at RAM address $004C. The array is used for loading program or verify data. The DATA array must be in the zero page and its size must match the size of the range to be programmed or verified.
- Registers H:X — In the Verify and fProgram routines, registers H and X are initialized with a 16-bit value representing the first address of a range. High and low bytes of the address are stored to registers H and X, respectively. In the fErase routine, registers H and X are initialized with an address which is within the page to be erased or with the address of the block protect register (FLBPR) if the entire array to be erased.

### 19.4.2 How to Use the Routines

This section describes the details of each routine. Table 19-10 provides necessary addresses used in the on-chip FLASH routines for each MCU type and summarizes the five routines.

## Table 19-10. Summary of On-Chip FLASH Support Routines

| | GetByte | PutByte | Verify | fProgram | fErase |
|---|---|---|---|---|---|
| **Jump Table Address** | $1000 | $100C | $1003 | $1009 | $1006 |
| **Routine Description** | Get a data byte serially through PTA0 | Send a data byte serially through PTA0 | Read and/or compare a FLASH range | Program a FLASH range | Erase a PAGE or entire array |
| **Internal Operating Frequency ($f_{op}$)** | N/A | N/A | N/A | 1.0 MHz to 8.0 MHz | 1.0 MHz to 8.0 MHz |
| **Hardware Requirement** | Pullup on PTA0 | Pullup on PTA0 | For send-out option, pullup on PTA0 | N/A | N/A |
| **Entry Conditions** | PTA0: Input (DDRA0 = 0) | PTA0: Input and 0 data bit (DDRA0 = 0, PTA0 = 0)<br>A: data to be sent | H:X: First address of range<br>LADDR: Last address of range<br>A: A = $00 for send-out option or A ≠ $00 for compare option.<br>For send-out option, PTA0: Input and 0 data bit (DDRA0 = 0, PTA0 = 0)<br>For compare option, DATA array: Load data to be compared against FLASH read data | H:X: First address of range<br>LADDR: Last address of range<br>CPUSPD: the nearest integer $f_{op}$ (in MHz) times 4<br>Data array: Load data to be programmed | H:X: Page erase — an address within the page<br>Mass erase = FLBPR<br>CPUSPD: the nearest integer $f_{op}$ (in MHz) times 4 |
| **Exit Conditions** | A: Data received through PTA0<br>C-bit: Framing error indicator (error: C = 0) | A, X: No change<br>PTA0: Input and 0 data bit (DDRA0 = 0, PTA0 = 0) | A: Checksum<br>H:X: Next FLASH address<br>C-bit: Verify result indicator (success: C = 1)<br>DATA array: Data replaced with FLASH read data (compare option) | H:X: Next FLASH address | H:X: No change |
| **I Bit** | I bit is preserved | I bit is preserved | I bit is preserved | I bit is set, then restored to entry condition on exit | I bit is set, then restored to entry condition on exit |
| **COP** | Not Serviced | Not Serviced | Serviced | Serviced | Serviced |

Continued on next page

**Table 19-10. Summary of On-Chip FLASH Support Routines (Continued) (Continued)**

| | GetByte | PutByte | Verify | fProgram | fErase |
|---|---|---|---|---|---|
| **Subroutines Called** | (None) | (None) | PutByte for send-out option | (None) | (None) |
| **RAM Variable** | N/A | N/A | LADDR (2 bytes), DATA array (no size limitation as long as in the zero page) | CPUSPD, LADDR (2 bytes), DATA array (maximum 32 bytes) | CPUSPD |
| **Stack Used (Including the Routine's Call)** | 6 bytes | 6 bytes | 9 bytes for verify option<br>13 bytes for send-out option | 10 bytes | 8 bytes |

#### *19.4.2.1 GetByte*

GetByte is a routine that receives a byte on the general-purpose I/O PTA0, and the received value is returned to the calling routine in the accumulator (A). This routine is also used in monitor mode so that it expects the same non-return-to-zero (NRZ) communication protocol and baud rates.

This routine detects a framing error when a STOP bit is not detected. If the carry (C) bit of the condition control register (CCR) is cleared after returning from this routine, a framing error occurred during the data receiving process. Therefore, the data in A is not reliable. The user software is responsible for handling such errors.

Interrupts are not masked (the I bit is not set) and the COP is not serviced in the GetByte routine. User software should ensure that interrupts are blocked during character reception.

The baud rate is defined by $f_{op}$ divided by a constant value. In the case of the MC68HC908EY16A, the baud rate is $f_{op}$ divided by 256. When the internal operating frequency is 2.4576 MHz, the baud rate is 2.4576 MHz/256 = 9600.

To use this routine, some hardware setup is required. The general-purpose I/O PTA0 must be pulled up. For more information, refer to 19.3 Monitor Module (MON).

**Entry Condition**

PTA0 must be configured as an input and pulled up in hardware.

**Exit Condition**

A — Contains data received from PTA0.

C bit — Usually the C bit is set, indicating proper reception of the STOP bit. However, if the C bit is clear, a framing error occurred. Therefore, the received byte in A is not reliable. Example 19-1 shows how to receive a byte serially on PTA0.

**Example 19-1. Receiving a Byte Serially**

```
GetByte:  equ  $1000          ;EY16A/8A GetByte jump address

          bclr 0,DDRA         ;Configure port A bit 0 as an input

          jsr  GetByte        ;Call GetByte routine
          bcc  FrameError     ;If C bit is clear, framing error
                              ; occurred. Take a proper action
```

***NOTE***

*After GetByte is called, the program will remain in this routine until a START bit (0) is detected and a complete character is received.*

#### *19.4.2.2 PutByte*

PutByte is a routine that receives a byte on the general-purpose I/O PTA0. The sent value must be loaded into the accumulator (A) before calling this routine. This routine is also used in the monitor mode. Therefore, it uses the same non-return-to-zero (NRZ) communication protocol. The communication baud rates are the same as those described in GetByte.

To use this routine, some hardware setup is required. The general-purpose I/O PTA0 must be pulled up and configured as an input and the PTA0 data bit must be initialized to 0.

Interrupts are not masked and the COP is not serviced in the PutByte routine. User software should ensure that interrupts are blocked during character transmission.

**Entry Condition**

A — Contains data sent from PTA0

PTA0 — This pin must be configured as an input and pulled up in hardware and the PTA0 data bit must be initialized to 0.

**Exit Condition**

A and X are restored with entry values.

Example 19-2 shows how to send a byte ($55) serially on PTA0.

**Example 19-2. Sending a Byte Serially**

```
PutByte:  equ  $100C          ;EY16A/8A PutByte jump address

          bclr 0,DDRA         ;Configure port A bit 0 as an input
          bclr 0,PTA          ;Initialize data bit to zero PTA0=0
          lda  #$55           ;Load sent data $55 to A
          jsr  PutByte        ;Call PutByte routine
```

#### *19.4.2.3 Verify*

When using the Verify routine, the user must select one of the function options summarized in Table 19-11. See Verify Routine Options for details.

**Table 19-11. Verify Routine Options**

| Option | Description |
|---|---|
| Send-out option | Used to read a range of FLASH locations and to send the read data to a host through PTA0 by using the PutByte routine. |
| Compare option | Used to read a range of FLASH locations and to compare the read data against the DATA array |

### Verify Routine Options

- Send-Out Option — If the accumulator (A) is initialized with $00 at the routine entry, the read data will be sent out serially through PTA0. The communication baud rate is the same as the baud rate described in the PutByte routine. When this option is selected, the PTA0 must be pulled up and configured as an input and the PTA0 data bit must be initialized to 0.
- Compare Option — If A is initialized with a non-zero value, the read data is compared against the DATA array for each byte of FLASH and the DATA array is replaced by the data read from FLASH. If the data does not match the corresponding value, the data read from FLASH can be confirmed in the DATA array. All data in the DATA array must be in the zero page, but a range can be beyond a row size or a page size.

### Carry (C) Bit and Checksum

The first and last addresses of the range to be read and/or compared are specified as parameters in registers H:X and LADDR, respectively. In the compare option, the carry (C) bit of the condition code register (CCR) is set if the data in the specified range is verified successfully against the data in the DATA array. However when the send-out option is selected, the status of the C bit is meaningless because this function does not include the compare operation. Both options calculate a checksum on data read in the range. This checksum, which is the LSB of the sum of all bytes in the entire data collection, is stored in A upon return from the function.

Interrupts are not masked. The COP is serviced in Verify. The first COP is serviced at 24 bus cycles after this routine is called in the user software. However, the COP timeout might still occur in the send-out option if the COP is configured for a short timeout period.

### Entry Condition

H:X — Contains the beginning address in a range.

LADDR — Contains the last address in a range.

A — When A contains $00, read data is sent out via PTA0 (send-out option is selected). When A contains a non-zero value, read data is verified against the DATA array (compare option is selected).

DATA array — Contains data to be verified against FLASH data. For the send-out option, the DATA array is not used.

PTA0 — When the send-out option is selected, this pin must be configured as an input and pulled up in hardware and PTA0 must be initialized to 0.

### Exit Condition

A — Contains a checksum value.

H:X — Contains the address of the next byte immediately after the range read.

C bit — Indicates the Verify result (only applies to the compare option).

- When the C bit is set, the verify succeeded.
- When the C bit is cleared, the verify failed.

DATA array — Replaced with data read from FLASH when the compare option is selected.

Example 19-3 shows how to use the compare option. Example 19-4 shows how to use the send-out option.

**Example 19-3. Compare Option**

```
Verify:    equ  $1003          ;EY16A/8A Verify jump address

LADDR:     equ  $004A          ;Define LADDR address (2 bytes)
DATA:      equ  $004C          ;Define DATA start address

           ldhx #$0000         ;Index offset into DATA array
           lda  #$AA           ;Initial data value to store in array
Data_load:
           coma
           sta  DATA,x         ;Fill DATA array, 32 bytes data,
                               ; to compare against programmed FLASH
           aix  #1             ; data (In this example comparing data
           cphx #$20           ; is $55, $AA, $55, $AA....)
           bne  Data_load

           ldhx #$C01F         ;Load last address of range to
           sthx LADDR          ; LADDR
           ldhx #$C000         ;Load beginning address of range
                               ; to H:X
           lda  #$55           ;Write non-zero value to A to select
                               ; the compare option
           jsr  Verify         ;Call Verify routine
           bcc  Error          ;If bit C is cleared, compare failed
                               ; Take a proper action
                               ; A contains a checksum value
```

**Example 19-4. Send-Out Option**

```
Verify:    equ  $1003          ;EY16A/8A Verify jump address

DATA:      equ  $004C          ;Define DATA start address

           bclr 0,DDRA         ;Configure Port A bit 0 as an input
           bclr 0,PTA          ;Initialize data bit to zero PTA0=0
           ldhx #$C025         ;Load last address of range to
           sthx LADDR          ; LADDR
           ldhx #$C010         ;Load beginning address of range
                               ; to H:X
           clra                ;A=0 to select send-out option
           jsr  Verify         ;Call Verify routine
                               ; A contains a checksum value
```

#### 19.4.2.4 fProgram

fProgram is used to program a range of FLASH locations with data loaded into the DATA array. All bytes that will be programmed must be in the same row. Programming data is passed to fProgram in the DATA array. All data in the DATA array must be in the zero page. The size of the DATA array must match the size of a specified programming range. This routine supports an internal operating frequency between 1.0 MHz and 8.0 MHz.

For this split-gate FLASH, the programming algorithm requires a programming time ($t_{prog}$) between 30 μs and 40 μs. Table 19-12 shows how $t_{prog}$ is adjusted by a CPUSPD value in this routine. The CPUSPD value is the nearest integer of $f_{op}$ (in MHz) multiplied by 4. For example, if $f_{op}$ is 2.4576 MHz, the CPUSPD value is 10 ($0A). If $f_{op}$ is 8.0 MHz, the CPUSPD value is 32 ($20).

**Table 19-12. $t_{prog}$ vs. Internal Operating Frequency**

| | Internal Operating Freq. ($f_{op}$) | CPUSPD | $t_{prog}$ (Cycles) | $t_{prog}$ |
|---|---|---|---|---|
| **Case 1** | 1.0 MHz $\leq f_{op} <$ 1.125 MHz | 4 | 38 | 33.8 μs $< t_{prog} \leq$ 38.0 μs |
| **Case 2** | 1.125 MHz $\leq f_{op} \leq$ 8.0 MHz | 5 to 32 | 8 x CPUSPD + 5 | 32.6 μs $\leq t_{prog} \leq$ 40.0 μs |

All programming is done using one programming algorithm. The algorithm allows for programming a single byte in each pass through it (one-byte programming method). Or, a whole row may be programmed by looping within the algorithm to write all the values in the row (row programming method).

- When the COPD bit in CONFIG1 is cleared and COP is therefore enabled, care must be taken to keep any programming operation from interfering with the servicing of the COP. In this case, each FLASH byte in the range is programmed using the one-byte programming method. Therefore, there are no limitations on range size and row/page boundary, but the total time to program multiple bytes is longer than the row programming method.
- When COPD bit is set (COP is disabled) and all programming addresses are in the same row, all FLASH bytes can be programmed at the same time using the row programming method. In this way, the FLASH can be programmed quickly.
- When COPD bit is set (COP is disabled) and a program range extends beyond a row or page, each FLASH byte in the range is programmed with the one-byte programming method until the beginning of the last row is reached. Then the bytes in the last row are programmed using the row programming method. However in this case, a programming range can not cross a boundary from $xxFF to $xx00. If a range crosses this boundary, programming data will not be guaranteed.

In fProgram, the high programming voltage time is enabled for less than 125 μs when programming a single byte at any internal operating frequency between 1.0 MHz and 8.0 MHz. Therefore, even when a row is programmed by 32 separate single-byte programming operations, the cumulative high voltage programming time is less than the maximum $t_{HV}$ (4 ms). The $t_{HV}$ is defined as the cumulative high voltage programming time to the same row before the next erase. For more information, refer to the memory characteristics in Chapter 20 Electrical Specifications.

This routine does not confirm that all bytes in the specified range are erased prior to programming. Nor does this routine perform a verification after programming, so there is no return confirmation that programming was successful. To program data successfully, the user software is responsible for these verifying operations. The Verify routine can be used to compare a programmed FLASH range against the DATA array.

Interrupts are masked (I bit is set) during a programming operation. When returning from this routine, I bit is restored to the entry condition. If the COP is enabled (COPD = 0), the COP is serviced in this routine. The first COP is serviced at 42 bus cycles after this routine is called in the user software. When the COP is disabled (COPD = 1), this row programming method is the fastest way to program the FLASH.

### Entry Condition

H:X — Contains the beginning address in a range.

LADDR — Contains the last address in a range.

CPUSPD — Contains the nearest integer value of $f_{op}$ (in MHz) times 4.

DATA array — Contains the data values to be programmed into FLASH.

### Exit Condition

H:X — Contains the address of the next byte after the range just programmed.

Example 19-5 shows how to program one full 32-byte row:

**Example 19-5. Programming a Row**

```
fProgram: equ  $1009          ;EY16A/8A fProgram jump address

CPUSPD:   equ  $0049          ;Define CPUSPD addrss
LADDR:    equ  $004A          ;Define LADDR address (2 bytes)
DATA:     equ  $004C          ;Define DATA start address

          ldhx #$0000         ;Index offset into DATA array
          lda  #$AA           ;Initial data value (inverted)
Data_load:
          coma                ;Alternate between $55 and $AA
          sta  DATA,x         ;Fill DATA array, 32 bytes data,
                              ; values to program into FLASH
          aix  #1             ; (ie. 55, AA, 55, AA....)
          cphx #$20
          bne  Data_load

          mov  #$0A,CPUSPD    ;fop = 2.4576MHz in this example
          ldhx #$C01F         ;Load last address of the row
          sthx LADDR          ; to LADDR
          ldhx #$C000         ;Load beginning address of the
                              ; row to H:X
          jsr  fProgram       ;Call fProgram routine
```

fProgram can be used to program a range less than 32 bytes. Example 19-6 shows how to program $55 and $AA at location $E004 and $E005, respectively.

**Example 19-6. Programming a Range Smaller than a Row**

```
fProgram: equ  $1009          ;EY16A/8A fProgram jump address

CPUSPD:   equ  $0049          ;Define CPUSPD addrss
LADDR:    equ  $004A          ;Define LADDR address (2 bytes)
DATA:     equ  $004C          ;Define DATA start address

          ldhx #$55AA
          sthx DATA

          mov  #$18,CPUSPD    ;fop = 6.0MHz in this example
          ldhx #$E005         ;Load last address to LADDR
          sthx LADDR
          ldhx #$E004         ;Load beginning address to H:X
          jsr  fProgram       ;Call fProgram routine
```

#### *19.4.2.5 fErase*

fErase can be called to erase a page (64 bytes) or a whole array of FLASH. When the address of the FLASH block protect register is passed to fErase, the entire array is erased (MASS). Any other valid FLASH address selects the page erase. This routine supports an internal operating frequency between 1.0 MHz and 8.0 MHz.

In this routine, both PAGE erase time ($t_{Erase}$) and MASS erase time ($t_{MErase}$) are set between 4 ms and 5.5 ms. The CPUSPD value is the nearest integer of $f_{op}$ (in MHz) times 4. For example if $f_{op}$ is 3.1 MHz, the CPUSPD is 12 ($0C). If $f_{op}$ is 4.9152 MHz, the CPUSPD is 20 ($14).

Interrupts are masked (I bit is set) during an erasing operation. When returning from this routine, I bit is restored to the entry condition, and the COP is serviced in ERARNGE. The first COP is serviced on (51+3xCPUSPD) bus cycles after this routine is called in the user software.

**Entry Condition**

H:X — Contains an address within a desired erase page or FLBPR for mass erase.

CPUSPD — Contains the nearest integer value of $f_{op}$ (in MHz) times 4.

**Exit Condition**

None

Example 19-7 shows how to erase an entire array.

**Example 19-7. Erasing an Entire Array**

```
fErase:    equ  $1006          ;EY16A/8A fErase jump address

CPUSPD:    equ  $0049          ;Define CPUSPD addrss

           mov  #$08,CPUSPD    ;fop = 2.0MHz in this example
           ldhx #FLBPR         ;Load FLBPR address to H:X
           jsr  fErase         ;Call fErase routine
```

Example 19-8 shows how to erase a page from $E100 through $E13F.

**Example 19-8. Erasing a Page**

```
fErase:    equ  $1006          ;EY16A/8A fErase jump address

CPUSPD:    equ  $0049          ;Define CPUSPD addrss

           mov  #$0E,CPUSPD    ;fop = 4.9152MHz in this example
           ldhx #$E121         ;Load any address within the
                               ; page to H:X
           jsr  fErase         ;Call fErase routine
```

If the FLASH locations that you want to erase are protected due to the value in the FLASH block protect register (FLBPR), the erase operation will not be successful. However when a high voltage ($V_{tst}$) is applied to the $\overline{IRQ}$ pin, the block protection is bypassed.

When the FLASH security check fails in the normal monitor mode, the FLASH can be re-accessed by erasing the entire FLASH array. To override the FLASH security mechanism and erase the FLASH array using this routine, registers H and X must contain the address of the FLASH block protect register (FLBPR).

# Chapter 20
# Electrical Specifications

## 20.1 Introduction

This section contains preliminary electrical and timing specifications.

## 20.2 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 20.5 5V DC 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 $V_{DD}$, $V_{SS}$, and PTA0–PTA6 and PTC0-PTC1 | I | ±15 | mA |
| Maximum current for pins PTA0–PTA6 and PTC0-PTC1 | $I_{PTA0}$–$I_{PTA6}$, $I_{PTC0}$–$I_{PTC1}$ | ±25 | mA |
| Maximum current out of $V_{SS}$ | $I_{MVSS}$ | 100 | mA |
| Maximum current into $V_{DD}$ | $I_{MVDD}$ | 100 | 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 $V_{DD}$).*

## 20.3 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>3.0 ± 10% | V |

## 20.4 Thermal Characteristics

| Characteristic | Symbol | Value | Unit |
|---|---|---|---|
| Thermal resistance<br>QFP (32 pins) | $\theta_{JA}$ | 100 | 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 $T_A$.

## 20.5 5V DC Electrical Characteristics

| Characteristic(1) | Symbol | Min | Typ(2) | Max | Unit |
|---|---|---|---|---|---|
| Output high voltage<br>$I_{Load}$ = –2.0 mA, all I/O pins<br>$I_{Load}$ = –5.0 mA, all I/O pins<br>$I_{Load}$ = –10.0 mA, all I/O pins<br>$I_{Load}$ = –15.0 mA, PTA0–PTA6/$\overline{SS}$ and PTC0–PTC1 only | $V_{OH}$ | <br>$V_{DD}$ –0.7<br>$V_{DD}$ –1.1<br>$V_{DD}$ –1.7<br>$V_{DD}$ –1.5 | <br>$V_{DD}$ –0.54<br>$V_{DD}$ –0.91<br>$V_{DD}$ –1.51<br>$V_{DD}$ –0.81 | <br>—<br>—<br>—<br>— | V |
| Output low voltage<br>$I_{Load}$ = 1.6 mA, all I/O pins<br>$I_{Load}$ = 5.0 mA, all I/O pins<br>$I_{Load}$ = 10.0 mA, all I/O pins<br>$I_{Load}$ = 15.0 mA, PTA0–PTA6/$\overline{SS}$ and PTC0–PTC1 only | $V_{OL}$ | <br>—<br>—<br>—<br>— | <br>0.31<br>0.56<br>0.99<br>1.44 | <br>0.4<br>1<br>1.5<br>1.8 | V |
| Input high voltage — all ports, $\overline{IRQ}$, $\overline{RST}$ | $V_{IH}$ | 0.7 x $V_{DD}$ | — | $V_{DD}$ + 0.3 | V |
| Input low voltage — all ports, $\overline{IRQ}$, $\overline{RST}$ | $V_{IL}$ | $V_{SS}$ | — | 0.3 x $V_{DD}$ | V |
| DC injection current, all ports(3) | $I_{INJ}$ | – 2.0 | — | + 2.0 | mA |
| Total DC current injection (sum of all I/O) | $I_{INJTOT}$ | – 25 | | + 25 | mA |

— Continued on next page

| Characteristic[1] | Symbol | Min | Typ[2] | Max | Unit |
|---|---|---|---|---|---|
| $V_{DD}$ + $V_{DDA}$ supply current<br>Run[4],[5]<br>Wait[5], [6]<br>Stop (LVI off) @ 25°C[7]<br>Stop (LVI on) @ 25°C<br>Stop (LVI off), –40°C to 125°C<br>Stop (LVI on), –40°C to 125°C | $I_{DD}$ | <br>—<br>—<br>—<br>—<br>—<br>— | <br>18<br>5.2<br>0.83<br>0.19<br>3.0<br>0.19 | <br>25<br>7.0<br>2.00<br>0.24<br>30<br>0.30 | <br>mA<br>mA<br>µA<br>mA<br>µA<br>mA |
| I/O ports Hi-Z leakage current[8] | $I_{IL}$ | –10 | — | +10 | µA |
| Input current – $\overline{RST}$, OSC1 | $I_{In}$ | –1 | — | +1 | µA |
| Capacitance<br>Ports (as input or output) | $C_{Out}$<br>$C_{In}$ | —<br>— | —<br>— | 12<br>8 | pF |
| POR rearm voltage[9] | $V_{POR}$ | 750 | — | — | mV |
| POR rise time ramp rate | $R_{POR}$ | 0.035 | — | — | V/ms |
| Monitor mode entry voltage | $V_{TST}$ | $V_{DD}$+ 3.5 | | 9.1 | V |
| Low-voltage inhibit reset, trip falling voltage[10] | $V_{TRIPF}$ | 3.90 | 4.30 | 4.50 | V |
| Low-voltage inhibit reset, trip rising voltage[11] | $V_{TRIPR}$ | 4.00 | 4.40 | 4.60 | V |
| Low-voltage inhibit reset/recover hysteresis[12] | $V_{HYS}$ | — | 90 | — | mV |
| Pullup resistor — PTA0–PTA6/$\overline{SS}$[13], $\overline{IRQ}$, $\overline{RST}$ | $R_{PU}$ | 24 | — | 48 | kΩ |
| Pulldown resistor — PTA0–PTA4[14] | $R_{PD}$ | 24 | 36 | 48 | kΩ |

1. $V_{DD}$ = 5.5 Vdc to 4.5 Vdc, $V_{SS}$ = 0 Vdc, $T_A$ = –40°C to +125°C, unless otherwise noted
2. Typical values reflect average measurements at midpoint of voltage range, 25°C only.
3. Some disturbance of the ADC accuracy is possible during any injection event and is dependent on board layout and power supply decoupling.
4. Run (operating) $I_{DD}$ measured using internal oscillator at its 32-MHz rate. $V_{DD}$ = 5.5 Vdc. All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. All ports configured as inputs. Measured with all modules enabled.
5. All measurements taken with LVI enabled.
6. Wait $I_{DD}$ measured using internal oscillator at its 1-MHz rate. All inputs 0.2 V from rail; no dc loads; less than 100 pF on all outputs. All ports configured as inputs.
7. Stop $I_{DD}$ is measured with no port pin sourcing current; all modules are disabled. OSCSTOPEN option is not selected.
8. Pullups and pulldowns are disabled.
9. Maximum is highest voltage that power-on reset (POR) is guaranteed.
10. These values assume the LVI is operating in 5-V mode (i.e. LVI5OR3 bit is set to 1).
11. These values assume the LVI is operating in 5-V mode (i.e. LVI5OR3 bit is set to 1).
12. These values assume the LVI is operating in 5-V mode (i.e. LVI5OR3 bit is set to 1).
13. PTA0–PTA4 pullup resistors are for interrupts only and are only enabled when the keyboard is in use.
14. Pulldown resistors available only when KBIx is enabled with KBIPx = 1.

## 20.6 5V Control Timing

| Characteristic[1] | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Frequency of operation [2]<br>Crystal option (EXTSLOW = 1, RNGSEL = 0)<br>Crystal option (EXTSLOW = 0, RNGSEL = 1)<br>Crystal option (EXTSLOW = 0, RNGSEL = 0)<br>External clock option (EXTSLOW = 1, RNGSEL = 0)[3]<br>External clock option (EXTSLOW = 0, RNGSEL = 1)<br>External clock option (EXTSLOW = 0, RNGSEL = 0) | $f_{OSC}$ | <br>32<br>1<br>8<br>dc[4]<br>1<br>8 | <br>100<br>8<br>32<br>100<br>8<br>32 | <br>kHz<br>MHz<br>MHz<br>kHz<br>MHz<br>MHz |
| Internal operating frequency | $f_{op}$ | — | 8 | MHz |
| Internal clock period ($1/f_{OP}$) | $t_{cyc}$ | 125 | — | ns |
| $\overline{RST}$ input pulse width low[5] | $t_{IRL}$ | 50 | — | ns |
| $\overline{IRQ}$ interrupt pulse width low[6] (edge-triggered) | $t_{ILIH}$ | 50 | — | ns |
| $\overline{IRQ}$ interrupt pulse period | $t_{ILIL}$ | Note 8 | — | $t_{cyc}$ |
| 16-bit timer[7]<br>Input capture pulse width<br>Input capture period | <br>$t_{TH}$,$t_{TL}$<br>$t_{TLTL}$ | <br>Note 8 | <br>—<br>— | <br>ns<br>$t_{cyc}$ |

1. $V_{SS}$ = 0 Vdc; timing shown with respect to 20% $V_{DD}$ and 70% $V_{SS}$ unless otherwise noted.
2. See Chapter 8 Internal Clock Generator (ICG) Module for more information.
3. No more than 10% duty cycle deviation from 50%
4. Some modules may require a minimum frequency greater than dc for proper operation. See appropriate table for this information.
5. Minimum pulse width reset is guaranteed to be recognized. It is possible for a smaller pulse width to cause a reset.
6. Minimum pulse width is for guaranteed interrupt. It is possible for a smaller pulse width to be recognized.
7. Minimum pulse width is for guaranteed interrupt. It is possible for a smaller pulse width to be recognized.
8. The minimum period, $t_{ILIL}$ or $t_{TLTL}$, should not be less than the number of cycles it takes to execute the interrupt service routine plus $t_{cyc}$.

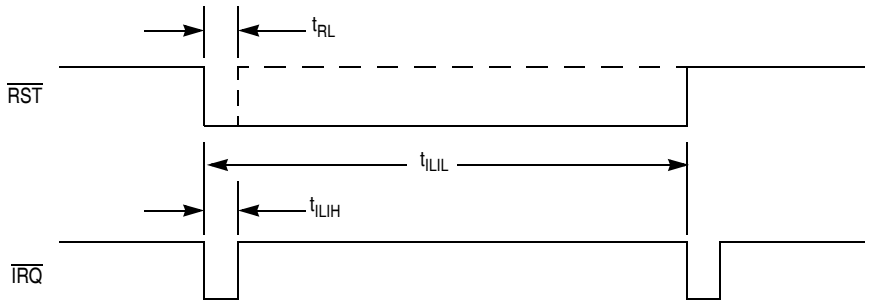



**Figure 20-1. $\overline{RST}$ and $\overline{IRQ}$ Timing**

## 20.7 3V DC Electrical Characteristics

| Characteristic[1] | Symbol | Min | Typ[2] | Max | Unit |
|---|---|---|---|---|---|
| Output high voltage<br>$I_{Load}$ = –0.6 mA, all I/O pins<br>$I_{Load}$ = –4.0 mA, all I/O pins<br>$I_{Load}$ = –10.0 mA, PTA0–PTA6/$\overline{SS}$, PTC0–PTC1 only | $V_{OH}$ | <br>$V_{DD}$–0.3<br>$V_{DD}$–1.0<br>$V_{DD}$–0.8 | <br>—<br>—<br>— | <br>—<br>—<br>— | V |
| Output low voltage<br>$I_{Load}$ = 0.5 mA, all I/O pins<br>$I_{Load}$ = 6.0 mA, all I/O pins<br>$I_{Load}$ = 10.0 mA, PTA0–PTA6/$\overline{SS}$, PTC0–PTC1 only | $V_{OL}$ | <br>—<br>—<br>— | <br>—<br>—<br>— | <br>0.3<br>1.0<br>0.8 | V |
| Input high voltage — all ports, $\overline{IRQ}$, $\overline{RST}$ | $V_{IH}$ | 0.7 x $V_{DD}$ | — | $V_{DD}$ + 0.3 | V |
| Input low voltage — all ports, $\overline{IRQ}$, $\overline{RST}$ | $V_{IL}$ | $V_{SS}$ | — | 0.3 x $V_{DD}$ | V |
| DC injection current, all ports | $I_{INJ}$ | –2 | — | +2 | mA |
| Total dc current injection (sum of all I/O) | $I_{INJTOT}$ | –25 | — | +25 | mA |
| $V_{DD}$ + $V_{DDA}$ supply current<br>Run[3],[4]<br>Wait[4], [5]<br>Stop (LVI off) @ 25°C[4], [6]<br>Stop (LVI on) @ 25°C<br>Stop (LVI off), –40°C to 125°C<br>Stop (LVI on), –40°C to 125°C | $I_{DD}$ | <br>—<br>—<br>—<br>—<br>—<br>— | <br>5.7<br>1.8<br>0.52<br>0.15<br>1.6<br>0.15 | <br>TBD<br>TBD<br>TBD<br>TBD<br>TBD<br>TBD | <br>mA<br>mA<br>µA<br>mA<br>µA<br>mA |
| Ports Hi-Z leakage current | $I_{IL}$ | –1 | ±0.1 | +1 | mA |
| Input current – $\overline{RST}$, OSC1 | $I_{In}$ | –1 | — | +1 | µA |
| Capacitance<br>Ports (as input or output) | $C_{IN}$<br>$C_{OUT}$ | —<br>— | —<br>— | 12<br>8 | pF |
| POR rearm voltage | $V_{POR}$ | 750 | — | — | mV |
| POR rise time ramp rate | $R_{POR}$ | 0.035 | — | — | V/ms |
| Monitor mode entry voltage | $V_{TST}$ | $V_{DD}$ + 2.5 | — | 9.1 | V |
| Low-voltage inhibit reset, trip falling voltage[7] | $V_{TRIPF}$ | 2.35 | 2.60 | 2.70 | V |
| Low-voltage inhibit reset, trip rising voltage[7] | $V_{TRIPR}$ | 2.45 | 2.66 | 2.80 | V |
| Low-voltage inhibit reset/recover hysteresis[7] | $V_{HYS}$ | — | 60 | — | mV |
| Pullup resistor — PTA0–PTA6/$\overline{SS}$, $\overline{IRQ}$, $\overline{RST}$[8] | $R_{PU}$ | 16 | 26 | 36 | kΩ |
| Pulldown resistor — PTA0–PTA4[9] | $R_{PD}$ | 16 | 26 | 36 | kΩ |

1. $V_{DD}$ = 2.7 to 3.3 Vdc, $V_{SS}$ = 0 Vdc, $T_A$ = $T_L$ to $T_H$, unless otherwise noted.
2. Typical values reflect average measurements at midpoint of voltage range, 25•C only.
3. Run (operating) $I_{DD}$ measured using internal oscillator at its 16-MHz rate. $V_{DD}$ = 3.0 Vdc. All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. All ports configured as inputs. Measured with all modules enabled.
4. All measurements taken with LVI enabled.
5. Wait $I_{DD}$ measured using internal oscillator at its 16-MHz rate. All inputs 0.2 V from rail; no dc loads; less than 100 pF on all outputs. All ports configured as inputs.
6. Stop $I_{DD}$ is measured with no port pin sourcing current; all modules are disabled. OSCSTOPEN option is not selected.
7. These values assume the LVI is operating in 3-V mode (i.e. LVI5OR3 bit is set to 0)
8. $R_{PU}$ is measured at $V_{DD}$ = 3.0 V.
9. Pulldown resistors available only when KBIx is enabled with KBIPx = 1.

## 20.8 3V Control Timing

| Characteristic[1] | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Frequency of operation [2]<br>Crystal option (EXTSLOW = 1, RNGSEL = 0)<br>Crystal option (EXTSLOW = 0, RNGSEL = 1)<br>Crystal option (EXTSLOW = 0, RNGSEL = 0)<br>External clock option (EXTSLOW = 1, RNGSEL = 0)[3]<br>External clock option (EXTSLOW = 0, RNGSEL = 1)<br>External clock option (EXTSLOW = 0, RNGSEL = 0) | $f_{OSC}$ | <br>32<br>1<br>8<br>dc[4]<br>1<br>8 | <br>100<br>8<br>16<br>100<br>8<br>16 | <br>kHz<br>MHz<br>MHz<br>kHz<br>MHz<br>MHz |
| Internal operating frequency | $f_{OP}$ | — | 4 | MHz |
| Internal operating period ($1/f_{OP}$) | $t_{CYC}$ | 250 | — | ns |
| $\overline{RST}$ input pulse width low[5] | $t_{RL}$ | 200 | — | ns |
| $\overline{IRQ}$ interrupt pulse width low[6] (edge-triggered) | $t_{ILIH}$ | 200 | — | ns |
| $\overline{IRQ}$ interrupt pulse period | $t_{ILIL}$ | Note[7] | — | $t_{CYC}$ |

1. $V_{DD}$ = 2.7 to 3.3 Vdc, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$; timing shown with respect to 20% $V_{DD}$ and 70% $V_{SS}$, unless otherwise noted.
2. See Chapter 8 Internal Clock Generator (ICG) Module for more information.
3. No more than 10% duty cycle deviation from 50%
4. Some modules may require a minimum frequency greater than dc for proper operation. See appropriate table for this information.
5. Minimum pulse width reset is guaranteed to be recognized. It is possible for a smaller pulse width to cause a reset.
6. Minimum pulse width is for guaranteed interrupt. It is possible for a smaller pulse width to be recognized.
7. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 $t_{CYC}$.

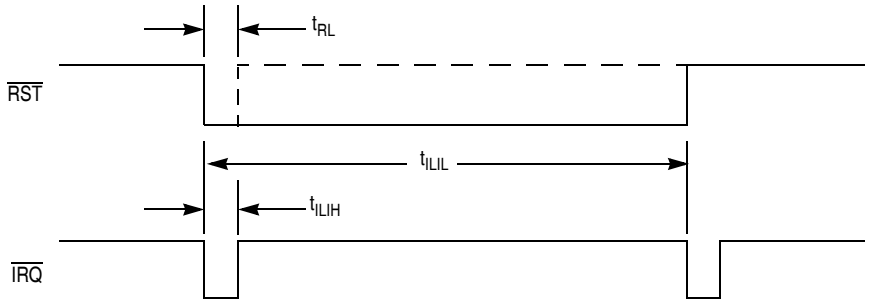



**Figure 20-2. $\overline{RST}$ and $\overline{IRQ}$ Timing**

## 20.9 Internal Oscillator Characteristics

| Characteristic[1] | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Internal oscillator base frequency[2], [3] | $f_{INTOSC}$ | 230.4 | 307.2 | 384 | kHz |
| Internal oscillator tolerance | $f_{OSC_TOL}$ | –25 | — | +25 | % |
| Internal oscillator multiplier[4] | N | 1 | — | 127 | — |

1. $V_{DD}$ = 4.5 to 5.5 Vdc, $V_{SS}$ = 0 Vdc, $T_A$ = –40°C to +125°C, unless otherwise noted
2. Internal oscillator is selectable through software for a maximum frequency. Actual frequency will be multiplier (N) x base frequency.
3. $f_{Bus} = (f_{INTOSC} / 4) \times N$ when internal clock source selected
4. Multiplier must be chosen to limit the maximum bus frequency of 8 MHz for 4.5-V operation.

## 20.10 External Oscillator Characteristics

| Characteristic[1] | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| External clock option[2][3]<br>With ICG clock disabled<br>With ICG clock enabled<br>External clock option (EXTSLOW = 1, RNGSEL = 0)[4]<br>External clock option (EXTSLOW = 0, RNGSEL = 1)[4]<br>External clock option (EXTSLOW = 0, RNGSEL = 0)[4] | $f_{EXTOSC}$ | <br>dc[5]<br><br>60<br>307.2<br>8 | <br>—<br><br>—<br>—<br>— | <br>32 [6]<br><br>307.2<br>8<br>32[6] | <br>MHz<br><br>kHz<br>MHz<br>MHz |
| External crystal options[7][8]<br>External cyrstal option (EXTSLOW = 1, RNGSEL = 0)[4]<br>External crystal option (EXTSLOW = 0, RNGSEL = 1)[4]<br>External crystal option (EXTSLOW = 0, RNGSEL = 0)[4] | $f_{EXTOSC}$ | <br>30<br>1<br>8 | <br>32.768<br>— | <br>100<br>8<br>32 | <br>kHz<br>MHz<br>MHz |
| Crystal load capacitance[9] | $C_L$ | — | 12.5 | — | pF |
| Crystal fixed capacitance[9] | $C_1$ | — | 15 | — | pF |
| Crystal tuning capacitance[9] | $C_2$ | — | 15 | — | pF |
| EXTSLOW = 1<br>Feedback bias resistor[9]<br>Series resistor [9] | <br>$R_B$<br>$R_s$ | <br>—<br>100 | <br>10<br>330 | <br>—<br>470 | <br>MΩ<br>kΩ |
| EXTSLOW = 0<br>Feedback bias resistor[9]<br>Series resistor [9][10]<br>$f_{OSCXCLK}$ = 1 MHz<br>$f_{OSCXCLK}$ = 4 MHz<br>$f_{OSCXCLK}$ = 8 to 25 MHz | <br>$R_B$<br><br>$R_s$<br>$R_s$<br>$R_s$ | <br>—<br><br>—<br>—<br>— | <br>1<br><br>20<br>10<br>0 | <br>—<br><br>—<br>—<br>— | <br>MΩ<br><br>kΩ<br>kΩ<br>kΩ |

1. $V_{DD}$ = 4.5 to 5.5 Vdc, $V_{SS}$ = 0 Vdc, $T_A$ = –40°C to +125°C, unless otherwise noted
2. Setting EXTCLKEN configuration option enables OSC1 pin for external clock square-wave input.
3. No more than 10% duty cycle deviation from 50%
4. EXTSLOW configuration option configures external oscillator for a slow speed crystal and sets the clock monitor circuits of the ICG module to expect an external clock frequency that is higher/lower than the internal oscillator base frequency, $f_{INTOSC}$.
5. Some modules may require a minimum frequency greater than dc for proper operation. See appropriate table for this information.
6. MCU speed derates from 32 MHz at $V_{DD}$ = 4.5 Vdc
7. Setting EXTCLKEN and EXTXTALEN configuration options enables OSC1 and OSC2 pins for external crystal option.
8. $f_{Bus} = (f_{EXTOSC} / 4)$ when external clock source is selected.
9. Crystal manufacturer's value.
10. Not required for high-frequency crystals

## 20.11 Trimmed Accuracy of the Internal Clock Generator

The unadjusted frequency of the low-frequency base clock (IBASE), when the comparators in the frequency comparator indicate zero error, can vary as much as ±25% due to process, temperature, and voltage. The trimming capability exists to compensate for process affects. The remaining variation in frequency is due to temperature, voltage, and change in target frequency (multiply register setting). These affects are designed to be minimal, however variation does occur. Better performance is seen with lower settings of N.

### 20.11.1 Trimmed Internal Clock Generator Characteristics

| Characteristic[1] | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Absolute trimmed internal oscillator tolerance[2],[3]<br>–40°C to 85°C<br>–40°C to 125°C | $F_{abs_tol}$ | —<br>— | ±2.0<br>±2.5 | ±3.5<br>±5.0 | % |
| Variation over temperature[3], [4] | $V_{ar_temp}$ | — | 0.05 | 0.08 | %/°C |
| Variation over voltage[3], [5]<br>25°C<br>–40°C to 85°C<br>–40°C to 125°C | $V_{ar_volt}$ | —<br>—<br>— | 1.0<br>1.0<br>1.0 | 2.0<br>2.0<br>2.0 | %/V |

1. These specifications concern long-term frequency variation. Each measurement is taken over a 1-ms period.
2. Absolute value of variation in ICG output frequency, trimmed at nominal $V_{DD}$ and temperature, as temperature and $V_{DD}$ are allowed to vary for a single given setting of N.
3. Specification is characterized but not tested.
4. Variation in ICG output frequency for a fixed N and voltage
5. Variation in ICG output frequency for a fixed N

## 20.12 ADC10 Characteristics

| Characteristic | Conditions | Symbol | Min | Typ[1] | Max | Unit | Comment |
|---|---|---|---|---|---|---|---|
| Supply voltage | Absolute | $V_{DD}$ | 2.7 | — | 5.5 | V | |
| Supply Current<br>ADLPC = 1<br>ADLSMP = 1<br>ADCO = 1 | $V_{DD} \leq 3.3$ V (3.0 V Typ) | $I_{DD}$[2] | — | 55 | — | µA | |
| | $V_{DD} \leq 5.5$ V (5.0 V Typ) | | — | 75 | — | | |
| Supply Current<br>ADLPC = 1<br>ADLSMP = 0<br>ADCO = 1 | $V_{DD} \leq 3.3$ V (3.0 V Typ) | $I_{DD}$[2] | — | 120 | — | µA | |
| | $V_{DD} \leq 5.5$ V (5.0 V Typ) | | — | 175 | — | | |
| Supply Current<br>ADLPC = 0<br>ADLSMP = 1<br>ADCO = 1 | $V_{DD} \leq 3.3$ V (3.0 V Typ) | $I_{DD}$[2] | — | 140 | — | µA | |
| | $V_{DD} \leq 5.5$ V (5.0 V Typ) | | — | 180 | — | | |
| Supply Current<br>ADLPC = 0<br>ADLSMP = 0<br>ADCO = 1 | $V_{DD} \leq 3.3$ V (3.0 V Typ) | $I_{DD}$[2] | — | 340 | — | µA | |
| | $V_{DD} \leq 5.5$ V (5.0 V Typ) | | — | 440 | 615 | | |

— Continued on next page

| Characteristic | Conditions | Symbol | Min | Typ(1) | Max | Unit | Comment |
|---|---|---|---|---|---|---|---|
| ADC internal clock | High speed (ADLPC = 0) | $f_{ADCK}$ | 0.40(3) | — | 2.00 | MHz | $t_{ADCK} = 1/f_{ADCK}$ |
| | Low power (ADLPC = 1) | | 0.40(3) | — | 1.00 | | |
| Conversion time (4) 10-bit Mode | Short sample (ADLSMP = 0) | $t_{ADC}$ | 19 | 19 | 20 | $t_{ADCK}$ cycles | |
| | Long sample (ADLSMP = 1) | | 39 | 39 | 40 | | |
| Conversion time (4) 8-bit Mode | Short sample (ADLSMP = 0) | $t_{ADC}$ | 16 | 16 | 17 | $t_{ADCK}$ cycles | |
| | Long sample (ADLSMP = 1) | | 36 | 36 | 37 | | |
| Sample time | Short sample (ADLSMP = 0) | $t_{ADS}$ | 4 | 4 | 4 | $t_{ADCK}$ cycles | |
| | Long sample (ADLSMP = 1) | | 24 | 24 | 24 | | |
| Input voltage | | $V_{ADIN}$ | $V_{SS}$ | — | $V_{DD}$ | V | |
| Input capacitance | | $C_{ADIN}$ | — | 7 | 10 | pF | Not tested |
| Input impedance | | $R_{ADIN}$ | — | 5 | 15 | kΩ | Not tested |
| Analog source impedance | | $R_{AS}$ | — | — | 10 | kΩ | External to MCU |
| Ideal resolution (1 LSB) | 10-bit mode | RES | 1.758 | 5 | 5.371 | mV | $V_{REFH}/2^N$ |
| | 8-bit mode | | 7.031 | 20 | 21.48 | | |
| Total unadjusted error | 10-bit mode | $E_{TUE}$ | 0 | ± 1.5 | ± 2.5 | LSB | Includes quantization |
| | 8-bit mode | | 0 | ± 0.7 | ± 1.0 | | |
| Differential non-linearity | 10-bit mode | DNL | 0 | ± 0.5 | — | LSB | |
| | 8-bit mode | | 0 | ± 0.3 | — | | |
| | Monotonicity and no-missing-codes guaranteed | | | | | | |
| Integral non-linearity | 10-bit mode | INL | 0 | ± 0.5 | — | LSB | |
| | 8-bit mode | | 0 | ± 0.3 | — | | |
| Zero-scale error | 10-bit mode | $E_{ZS}$ | 0 | ± 0.5 | — | LSB | $V_{ADIN} = V_{SS}$ |
| | 8-bit mode | | 0 | ± 0.3 | — | | |
| Full-scale error | 10-bit mode | $E_{FS}$ | 0 | ± 0.5 | — | LSB | $V_{ADIN} = V_{DD}$ |
| | 8-bit mode | | 0 | ± 0.3 | — | | |
| Quantization error | 10-bit mode | $E_Q$ | — | — | ± 0.5 | LSB | 8-bit mode is not truncated |
| | 8-bit mode | | — | — | ± 0.5 | | |
| Input leakage error | 10-bit mode | $E_{IL}$ | 0 | ± 0.2 | ± 5 | LSB | Pad leakage(5) * $R_{AS}$ |
| | 8-bit mode | | 0 | ± 0.1 | ± 1.2 | | |

1. Typical values assume $V_{DD}$ = 5.0 V, temperature = 25•C, $f_{ADCK}$ = 1.0 MHz unless otherwise stated. Typical values are for reference only and are not tested in production.
2. Incremental $I_{DD}$ added to MCU mode current.
3. Values are based on characterization results, not tested in production.
4. Reference the ADC module specification for more information on calculating conversion times.
5. Based on typical input pad leakage current.

## 20.13 5V SPI Characteristics

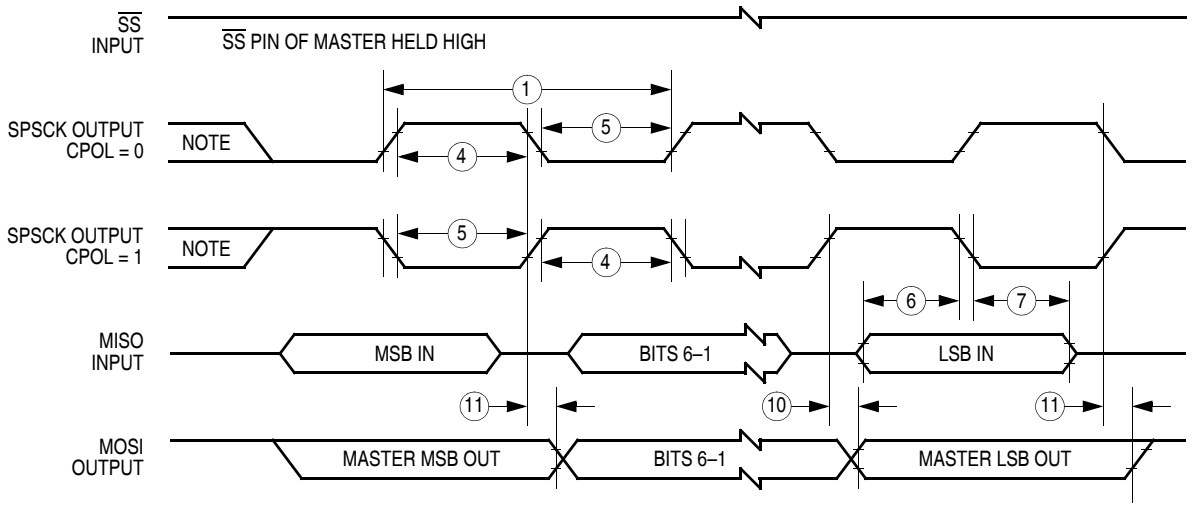
| Diagram Number[1] | Characteristic[2] | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| | Operating frequency<br>Master<br>Slave | <br>$f_{OP(M)}$<br>$f_{OP(S)}$ | <br>$f_{OP}/128$<br>dc | <br>$f_{OP}/2$<br>$f_{OP}$ | <br>MHz<br>MHz |
| 1 | Cycle time<br>Master<br>Slave | <br>$t_{CYC(M)}$<br>$t_{CYC(S)}$ | <br>2<br>1 | <br>128<br>— | <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 | <br>$t_{SCKH(M)}$<br>$t_{SCKH(S)}$ | <br>$t_{CYC}$ –25<br>1/2 $t_{CYC}$ –25 | <br>64 $t_{CYC}$<br>— | <br>ns<br>ns |
| 5 | Clock (SPSCK) low time<br>Master<br>Slave | <br>$t_{SCKL(M)}$<br>$t_{SCKL(S)}$ | <br>$t_{CYC}$ –25<br>1/2 $t_{CYC}$ –25 | <br>64 $t_{CYC}$<br>— | <br>ns<br>ns |
| 6 | Data setup time (inputs)<br>Master<br>Slave | <br>$t_{SU(M)}$<br>$t_{SU(S)}$ | <br>30<br>30 | <br>—<br>— | <br>ns<br>ns |
| 7 | Data hold time (inputs)<br>Master<br>Slave | <br>$t_{H(M)}$<br>$t_{H(S)}$ | <br>30<br>30 | <br>—<br>— | <br>ns<br>ns |
| 8 | Access time, slave[3]<br>CPHA = 0<br>CPHA = 1 | <br>$t_{A(CP0)}$<br>$t_{A(CP1)}$ | <br>0<br>0 | <br>40<br>40 | <br>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] | <br>$t_{V(M)}$<br>$t_{V(S)}$ | <br>—<br>— | <br>50<br>50 | <br>ns<br>ns |
| 11 | Data hold time, outputs, after enable edge<br>Master<br>Slave | <br>$t_{HO(M)}$<br>$t_{HO(S)}$ | <br>0<br>0 | <br>—<br>— | <br>ns<br>ns |

1. Numbers refer to dimensions in Figure 20-3 and Figure 20-4.
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

## 20.14 3V SPI Characteristics

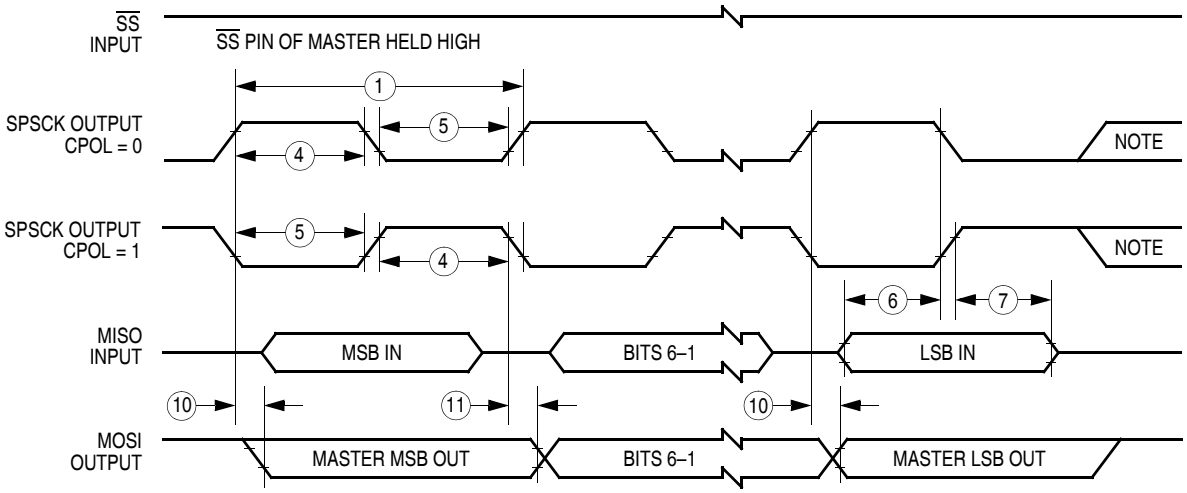
| Diagram Number[1] | Characteristic[2] | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| | Operating frequency<br>Master<br>Slave | <br>$f_{OP(M)}$<br>$f_{OP(S)}$ | <br>$f_{OP}/128$<br>DC | <br>$f_{OP}/2$<br>$f_{OP}$ | <br>MHz<br>MHz |
| 1 | Cycle time<br>Master<br>Slave | <br>$t_{CYC(M)}$<br>$t_{CYC(S)}$ | <br>2<br>1 | <br>128<br>— | <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 | <br>$t_{SCKH(M)}$<br>$t_{SCKH(S)}$ | <br>$t_{cyc}$ –35<br>1/2 $t_{cyc}$ –35 | <br>64 $t_{cyc}$<br>— | <br>ns<br>ns |
| 5 | Clock (SPSCK) low time<br>Master<br>Slave | <br>$t_{SCKL(M)}$<br>$t_{SCKL(S)}$ | <br>$t_{cyc}$ –35<br>1/2 $t_{cyc}$ –35 | <br>64 $t_{cyc}$<br>— | <br>ns<br>ns |
| 6 | Data setup time (inputs)<br>Master<br>Slave | <br>$t_{SU(M)}$<br>$t_{SU(S)}$ | <br>40<br>40 | <br>—<br>— | <br>ns<br>ns |
| 7 | Data hold time (inputs)<br>Master<br>Slave | <br>$t_{H(M)}$<br>$t_{H(S)}$ | <br>40<br>40 | <br>—<br>— | <br>ns<br>ns |
| 8 | Access time, slave[3]<br>CPHA = 0<br>CPHA = 1 | <br>$t_{A(CP0)}$<br>$t_{A(CP1)}$ | <br>0<br>0 | <br>50<br>50 | <br>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] | <br>$t_{V(M)}$<br>$t_{V(S)}$ | <br>—<br>— | <br>60<br>60 | <br>ns<br>ns |
| 11 | Data hold time, outputs, after enable edge<br>Master<br>Slave | <br>$t_{HO(M)}$<br>$t_{HO(S)}$ | <br>0<br>0 | <br>—<br>— | <br>ns<br>ns |

1. Numbers refer to dimensions in Figure 20-3 and Figure 20-4.
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.

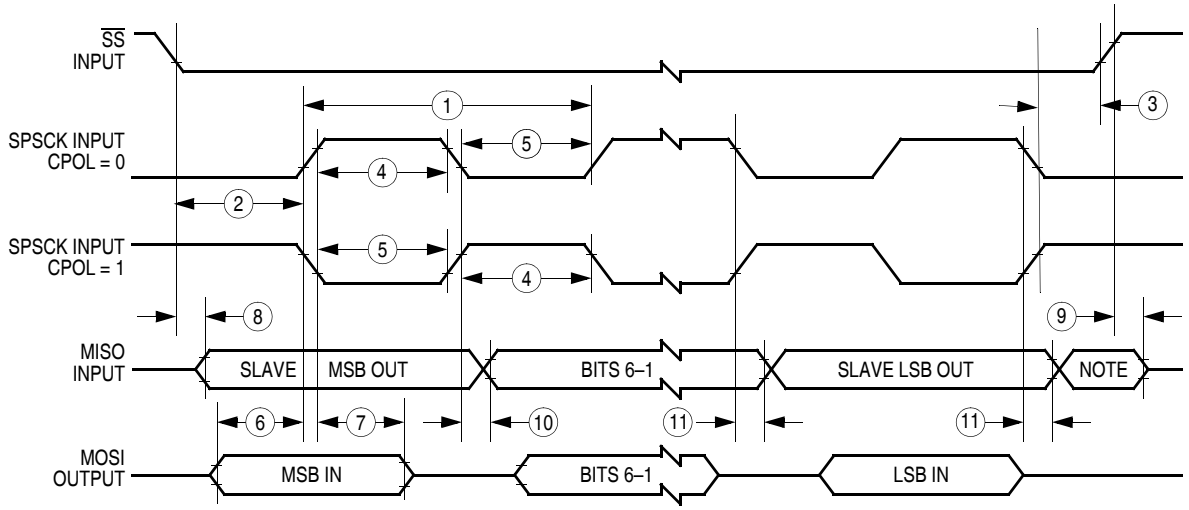
**a) SPI Master Timing (CPHA = 0)**




Note: This last clock edge is generated internally, but is not seen at the SPSCK pin.

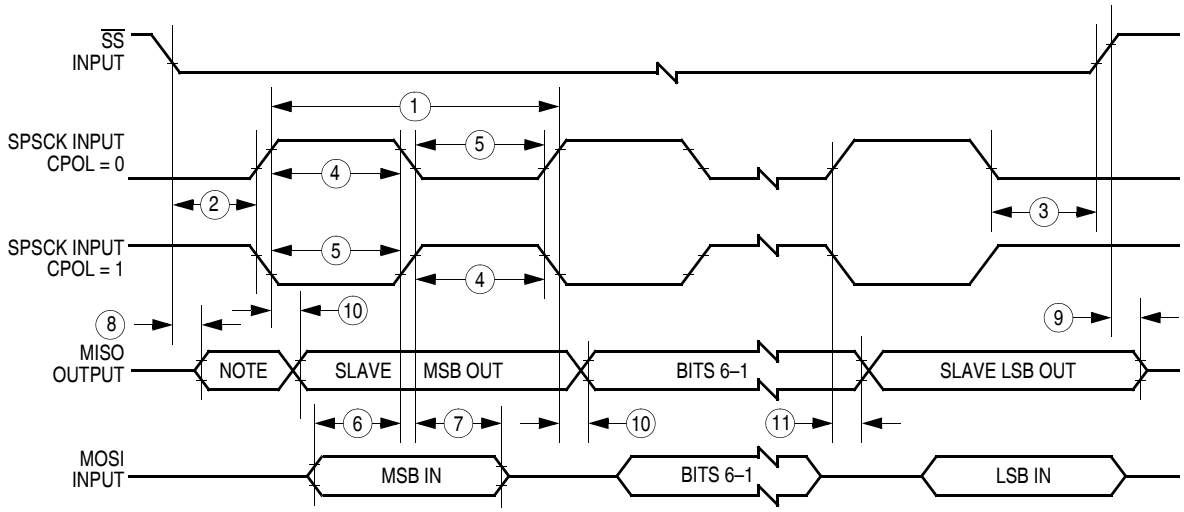
**b) SPI Master Timing (CPHA = 1)**

**Figure 20-3. 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 20-4. SPI Slave Timing**

## 20.15 Timer Interface Module Characteristics

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Timer input capture pulse width[1] | $t_{TH}$, $t_{TL}$ | 2 | — | $t_{cyc}$ |
| Timer input capture period | $t_{TLTL}$ | Note[2] | — | $t_{cyc}$ |
| Timer input clock pulse width[1] | $t_{TCL}$, $t_{TCH}$ | $t_{cyc}$ + 5 | — | ns |

1. Values are based on characterization results, not tested in production.
2. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 $t_{cyc}$.

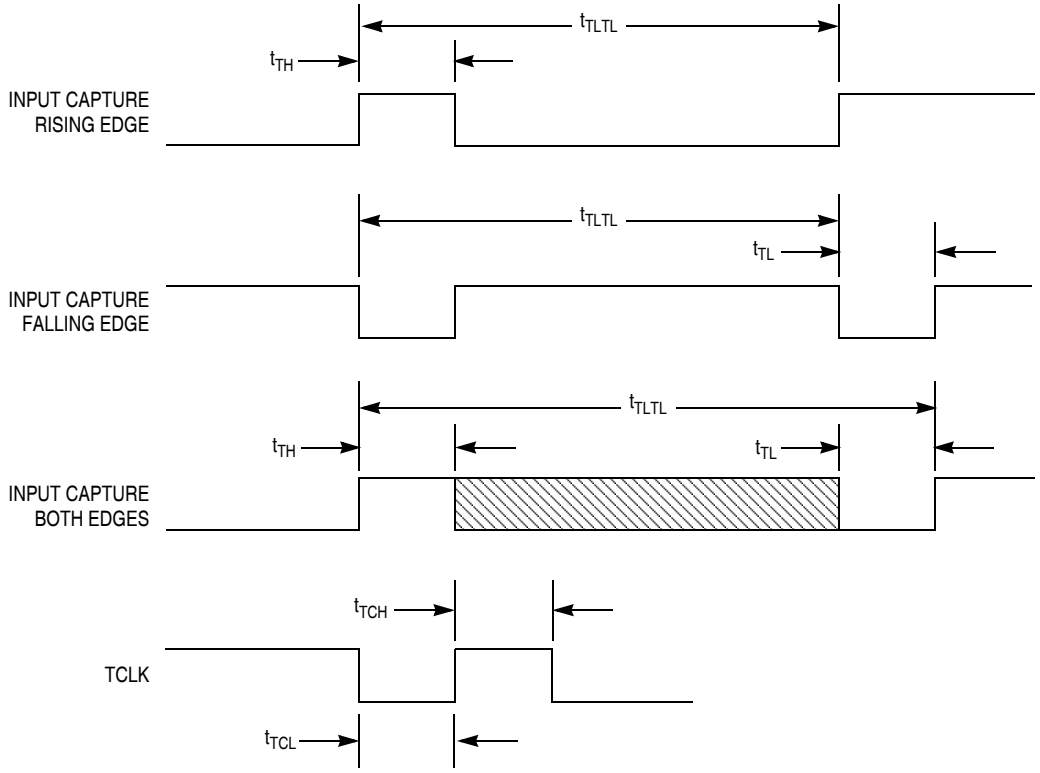



**Figure 20-5. Timer Input Timing**

## 20.16 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 PGM/ERASE supply voltage (VDD) | $V_{PGM/ERASE}$ | 2.7 | — | 5.5 | V |
| 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: $t_{NVS} + t_{NVH} + t_{PGS} + (t_{PROG} \times 64) \leq t_{HV}$ 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.

## 20.17 EMC Performance

EMC performance is highly dependant on the environment in which the MCU resides. Board design and layout, circuit topology choices, location and characteristics of external components as well as MCU software operation all play a significant role in EMC performance. The system designer should consult Freescale applications notes such as AN2321, AN1050, AN1263, AN2764, and AN1259 for advice and guidance specifically geared toward EMC performance.

### 20.17.1 Radiated Emissions

Microcontroller radiated RF emissions are measured from 150 kHz to 1 GHz using the TEM/GTEM Cell method in accordance with the IEC 61967-2 and SAE J1752/3 standards. The measurement is performed with the microcontroller installed on a custom EMC evaluation board including I/O pin loading and board layer usage while running specialized EMC test software all designed in compliance with the standards. The radiated emissions from the microcontroller are measured in a TEM cell in two package orientations (North and East). For more detailed information concerning the evaluation conditions and setup, please refer to the EMC Evaluation Report for this device.

The maximum radiated RF emissions of the tested configuration in all orientations are less than or equal to the reported emissions levels.

| Parameter | Symbol | Conditions | Frequency | $f_{osc}/f_{CPU}$ | Level[1] (Max) | Unit |
|---|---|---|---|---|---|---|
| Radiated emissions, electric field<br><br>Conditions — TBD | $V_{RE_TEM}$ | $V_{DD}$ = 5 V<br>$T_A$ = +25°C<br>32 QFP | 0.15 – 50 MHz | 4/8 | TBD | dBμV |
| | | | 50 – 150 MHz | | TBD | |
| | | | 150 – 500 MHz | | TBD | |
| | | | 500 – 1000 MHz | | TBD | |
| | | | IEC Level | | TBD | — |
| | | | SAE Level | | TBD | — |

1. Data based on qualification test results.

### 20.17.2 Conducted Transient Susceptibility

Microcontroller transient conducted susceptibility is measured in accordance with an internal Freescale test method. The measurement is performed with the microcontroller installed on a custom EMC evaluation board and running specialized EMC test software designed in compliance with the test method. The conducted susceptibility is determined by injecting the transient susceptibility signal on each pin of the microcontroller. The transient waveform and injection methodology are in accordance with IEC 61000-4-2 (ESD) and IEC 61000-4-4 (EFT/B). The transient voltage required to cause performance degradation on any pin in the tested configuration is greater than or equal to the reported levels unless otherwise indicated by footnotes below the table.

| Parameter | Symbol | Conditions | $f_{OSC}/f_{CPU}$ | Result | Amplitude(1) (Min) | Unit |
|---|---|---|---|---|---|---|
| Conducted susceptibility, electrical fast transient/burst (EFT/B) | $V_{CS_EFT}$ | $V_{DD}$ = 5 V<br>$T_A$ = +25°C<br>32 QFP | 4/8 | A | TBD | kV |
| | | | | B | TBD | |
| | | | | C | TBD | |
| | | | | D | TBD | |
| Conducted susceptibility, electrostatic discharge (ESD) | $V_{CS_ESD}$ | $V_{DD}$ = 5 V<br>$T_A$ = +25°C<br>32 QFP | 4/8 | A | TBD | kV |
| | | | | B | TBD | |
| | | | | C | TBD | |
| | | | | D | TBD | |

1. Data based on qualification test results. Not tested in production.
2. These pins demonstrate particularly low levels of performance:

The susceptibility performance classification is described in the following table.

| Result | Performance Criteria | |
|---|---|---|
| A | No failure | The MCU performs as designed during and after exposure. |
| B | Self-recovering failure | The MCU does not perform as designed during exposure. The MCU returns automatically to normal operation after exposure is removed. |
| C | Soft failure | The MCU does not perform as designed during exposure. The MCU does not return to normal operation until exposure is removed and the RESET pin is asserted. |
| D | Hard failure | The MCU does not perform as designed during exposure. The MCU does not return to normal operation until exposure is removed and the power to the MCU is cycled. |
| E | Damage | The MCU does not perform as designed during and after exposure. The MCU cannot be returned to proper operation due to physical damage or other permanent performance degradation. |

# Chapter 21
# Ordering Information and Mechanical Specifications

## 21.1 Introduction

This section contains ordering numbers for MC68HC908EY16. An example of the device numbering system is given in Figure 21-1. In addition, this section gives the package dimensions for the 32-pin LQFP.

## 21.2 Ordering Information

**Table 21-1. Ordering Numbers**

| Part Number[1] | Operating Temperature Range |
|---|---|
| **Automotive Part Numbers[2]** | |
| S908EY16AMFJE | –40•C to +125•C |
| S908EY16AVFJE | –40•C to +105•C |
| S908EY16ACFJE | –40•C to +85•C |
| **Consumer and Industrial Part Numbers** | |
| MC908EY16AMFJE | –40•C to +125•C |
| MC908EY16AVFJE | –40•C to +105•C |
| MC908EY16ACFJE | –40•C to +85•C |

1. FJ = 32-pin low-profile quad flat package
2. “S” part numbers are tested in accordance with the AEC-Q100 (Automotive Electronics Council) standard.

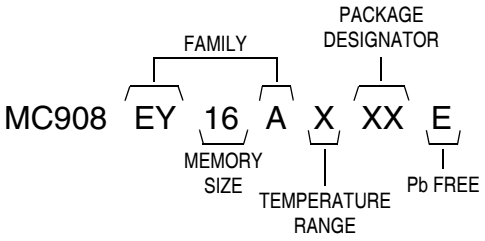



**Figure 21-1. Device Numbering System**

## 21.3 Package Dimensions

Refer to the following pages for detailed package dimensions.

Package drawing 98ASH70029A page 1 of 3

Package drawing 98ASH70029A page 2 of 3

Package drawing 98ASH70029A page 3 of 3

# Appendix A
# MC68HC908EY8A

## A.1 Introduction

The MC68HC908EY8A 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 MC68HC908EY8A with the exceptions shown in this appendix.

## A.2 Block Diagram

See Figure A-1.

## A.3 Memory

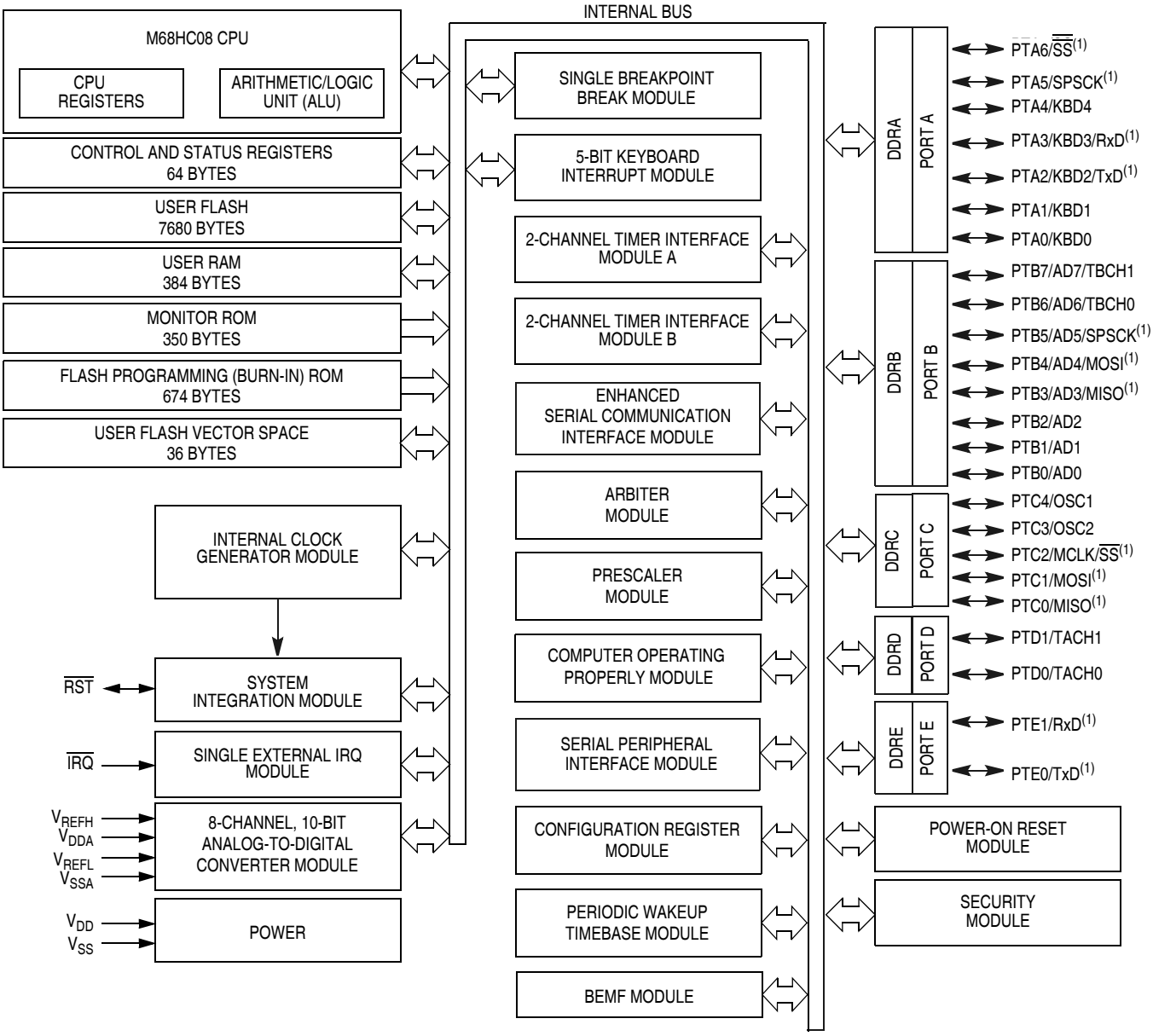
The memory map, shown in Figure A-2, includes:

- 8 Kbytes of FLASH memory, 7680 bytes of user space
- 384 bytes of random-access memory (RAM)
- 36 bytes of user-defined vectors
- 350 bytes of monitor routines in read-only memory (ROM)
- 674 bytes of integrated FLASH burn-in routines in ROM

The FLASH memory is an array of 7680 bytes with an additional 36 bytes of user vectors and one byte used for block protection. The FLASH is organized internally as an 8192-word by 8-bit complementary metal-oxide semiconductor (CMOS) page erase, byte (8-bit) program embedded FLASH memory. Each page consists of 64 bytes. The page erase operation erases all words within a page. A page is composed of two adjacent rows.

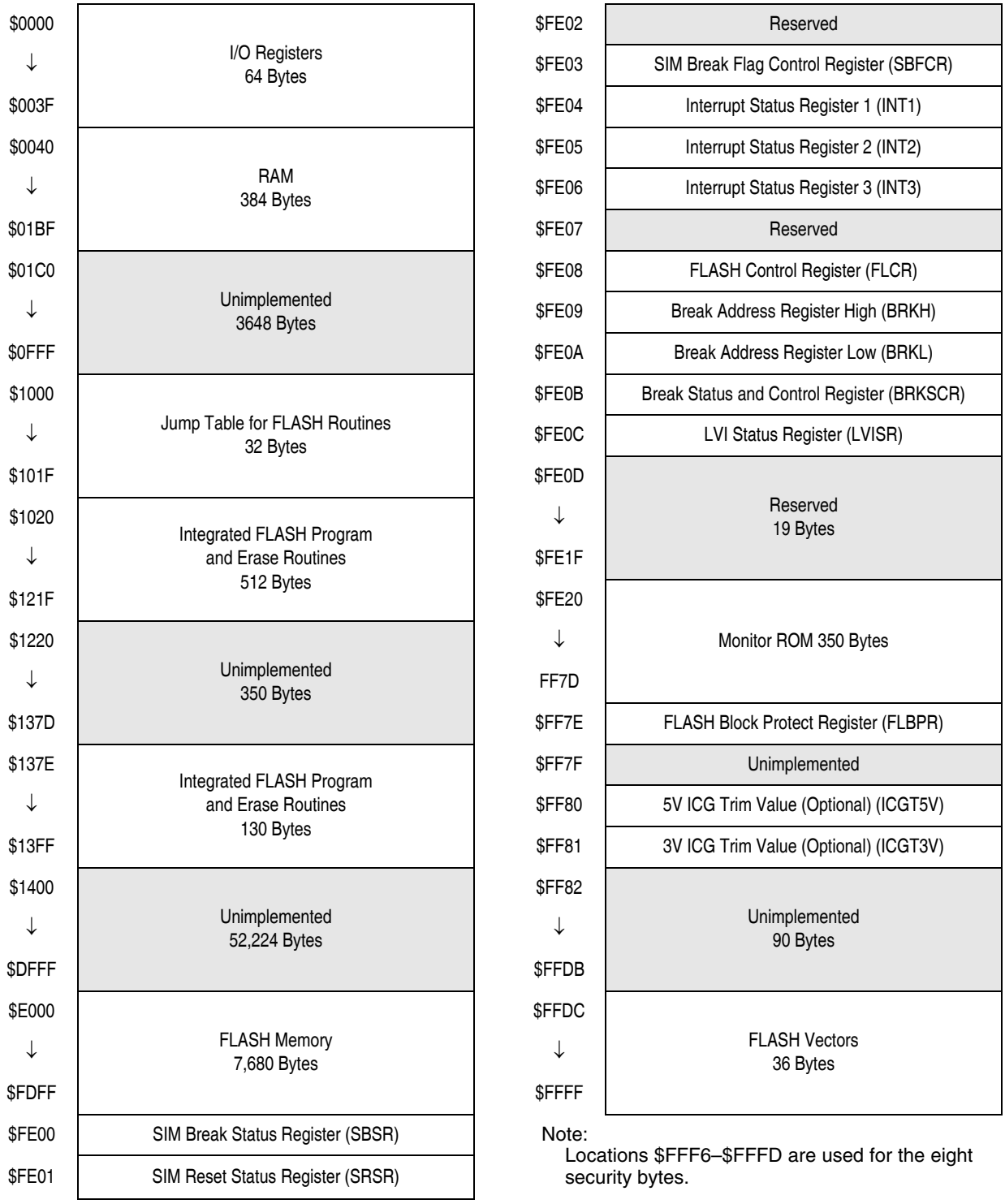
A security feature prevents viewing of the FLASH contents.(1)

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




NOTE:
1. The locations of the ESCI and SPI pins are user selectable using CONFIG3 option bits.

**Figure A-1. MC68HC908EY8A Block Diagram**




Note:
Locations $FFF6–$FFFD are used for the eight security bytes.

**Figure A-2. MC68HC908EY8A Memory Map**

## A.4 Ordering Information

**Table A-1. Ordering Numbers**

| Part Number[1] | Operating Temperature Range |
|---|---|
| **Automotive Part Numbers[2]** | |
| S908EY8AMFJE | –40•C to +125•C |
| S908EY8AVFJE | –40•C to +105•C |
| S908EY8ACFJE | –40•C to +85•C |
| **Consumer and Industrial Part Numbers** | |
| MC908EY8AMFJE | –40•C to +125•C |
| MC908EY8AVFJE | –40•C to +105•C |
| MC908EY8ACFJE | –40•C to +85•C |

1. FJ = 32-pin low-profile quad flat package
2. “S” part numbers are tested in accordance with the AEC-Q100 (Automotive Electronics Council) standard.

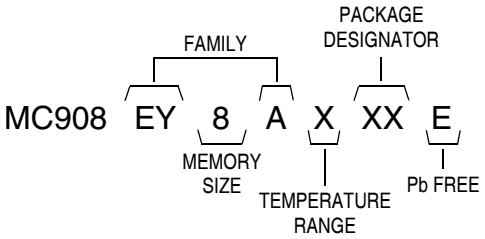



**Figure A-3. Device Numbering System**

# Appendix B
# Differences Between 908EY16A and 908EY16

## B.1 Introduction

The 908EY16A is a new revision of the existing 908EY16. The 908EY16A was designed using the latest HC08 design technology. Emphasis was placed on maintaining compatibility with the existing 908EY16 during the design stage. While this was largely realized, there are a few differences introduced by customer requested improvements and by the use of the latest, enhanced modules. The purpose of this appendix is to point out the differences between the two versions of the part.

## B.2 Configuration

A CONFIG3 register has been added to allow for new configuration features of the ESCI, SPI, and ICG. This replaces a reserved register location at address $0009.

Address: $0009

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 908EY16: | R | R | R | R | R | R | R | R |
| 908EY16A: | NA | RNGSEL | ESCISRE | SPISRE | MCLKSRE | PORTSRE | ESCISEL | SPISEL |
| Reset: | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure B-1. Configuration Register 3 (CONFIG3)**

### B.2.1 Enhanced Serial Communications Interface Module (ESCI)

Enhanced transmitter functions are available for the local interconnect network (LIN). The LINT bit has been added to ESCI Baud Rate register. (The bit location was previously reserved.)

Address: $0016

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 908EY16: | R | LINR | SCP1 | SCP0 | R | SCR2 | SCR1 | SCR0 |
| 908EY16A: | LINT | LINR | SCP1 | SCP0 | R | SCR2 | SCR1 | SCR0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure B-2. ESCI Baud Rate Register (SCBR)**

The transmit and receive pins can be remapped to the PTA2 and PTA3 pins. This is selected in the CONFIG3 register. The reset state of these bits maps the ESCI transmit and receive to the same pins as on the 908EY16.

### B.2.2 Serial Peripheral Interface Module (SPI)

The four pins associated with the SPI can be remapped to alternate assignments on PTB and PTC. The alternated assignment is selected in the CONFIG3 register. The reset state of these bits maps the SPI functions to the same pins as on the 908EY16.

### B.2.3 Internal Clock Generator Module (ICG)

An option to allow the use of high frequency (8 – 32 MHz) crystals for the external oscillator has been added. There is now a range select bit in the CONFIG3 register to select this high frequency range.

### B.2.4 Keyboard Interface Module (KBI)

The ability to select whether a keyboard interrupt is triggered by a rising or falling edge has been added. A reserved register has been replaced with the Keyboard Polarity register (KBIPR) at address $000C. While the falling edge trigger was the only mode available on the 908EY16, with the 908EY16A you can set either rising or falling edge and the reset state of the polarity bits will match the original state.

Address: $000C

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 908EY16: | R | R | R | R | R | R | R | R |
| 908EY16A: | 0/NA | 0/NA | 0/NA | KBIP4 | KBIP3 | KBIP2 | KBIP1 | KBIP0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure B-3. Keyboard Interrupt Polarity Register (KBIPR)**

### B.2.5 Analog-to-Digital Converter Module (ADC)

The original ADC module has been replaced with the improved ADC10 module. While the modules are extremely similar, there are some differences that need to be evaluated for impact on existing software.

- The conversion complete (COCO) bit now functions slightly differently. COCO is now always a read-only bit and will get set regardless of the state of the AIEN bit.
- The divide-by-6 clock selection has been removed. (ADIV2 bit replaced by ADLPC)
- The two left-justified modes for the ADC data format are no longer available.
- A long sample time option has been added to conserve power at the expense of longer conversion times. This option is selected using the new ADLSMP bit in the ADCLK register. (The bit location was previously reserved.)
- The ADC10 will now run in stop mode if the ACLKEN bit is set to enable the asynchronous clock inside the ADC10 module. Utilizing stop mode for an ADC conversion gives the quietest operating mode to get extremely accurate ADC readings. (The bit location now used by ACLKEN was unimplemented — it always read as a 0 and writes to that location had no affect.)
- The ADC10 conversion time is now anywhere from 21 ADC clock cycles to 44 ADC clock cycles depending on ACLKEN. The original ADC module in the 908EY16 had a conversion time of 16 to 17 ADC clock cycles. The bits that control these clocking operations are in the ADCLK register. These changes are shown in Figure B-4.

| Address: | $003F | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| 908EY16: | ADIV2 | ADIV1 | ADIV0 | ADICLK | MODE1 | MODE0 | R | 0/NA |
| 908EY16A: | ADLPC | ADIV1 | ADIV0 | ADICLK | MODE1 | MODE0 | ADLSMP | ACLKEN |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R | = Reserved | | | | | | |

**Figure B-4. ADC10 Clock Register (ADCLK)**

- Enabling an ADC channel no longer overrides the digital I/O function of the associated pin. To prevent the digital I/O from interfering with the ADC read of the pin, the data direction bit associated with the port pin must be set as input.

## B.3 Monitor Mode

### B.3.1 Monitor Extended Security

An extended security feature has been added to the 908EY16 monitor operation. When the extended security location is programmed with zero and all 8 byte security matches, the monitor is terminated in an infinite loop automatically. To unlock extended security, the part must enter monitor mode with failed security and then the FLASH must be mass-erased to erase the whole FLASH. The extended security location in the 908EY16A is located at address $FDFF. The user should check this location in their software to ensure that it will not cause unexpected operation.

### B.3.2 Zeroes in Security Bytes

An additional check has been added to the verification of the security bytes. The number of zero bytes used for the security bytes is limited to 5. More than 5 bytes of zero out of the 8 security bytes will cause the security check to fail. The user should check the values programmed into locations $FFF6–$FFFD to ensure that this requirement is not violated.

### B.3.3 Forced Monitor Mode Baud Rate

The baud rate used for the 908EY16 Forced Monitor Mode was set at ~6300 baud. In the 908EY16A, this has been changed to 9600 baud. In addition, the trim value stored in FLASH is used in this mode to ensure that the tolerance required for communicating at this rate is met.

## B.4 Monitor ROM FLASH Programming Routines

### B.4.1 Erase

The existing 908EY16 call for erase uses a RAM variable called CTRLBYT to determine whether the call is for page or mass erase. The 908EY16A routine uses the address passed to determine the page to be erased. (If ADDR = FLBPR, then a mass erase is performed.) If the parameters for the current 908EY16 routine are passed to the new 908EY16A routine for a page erase, it would work correctly. Using an existing 908EY16 call to mass erase the 908EY16A will not work.

A minor difference is that the new 908EY16A routine preserves the original state of the I bit while the existing 908EY16 erase routine sets the I bit and leaves it set on exit.

### B.4.2 Program

The existing 908EY16 routine uses the row programming method to program any range of addresses (starting address in H:X and ending address in RAM at LADDR, data to be programmed is in RAM starting after LADDR). This routine is interrupted every 6 bytes to service the COP.

The 908EY16A routine uses the same variables, so the call setup would be the same. In the 908EY16A, the COPD bit is checked. If COP servicing is required, then a byte-by-byte algorithm is used to program the range. If no COP servicing is required, then a combination of byte-by-byte and row programming is used for a faster algorithm. (In fact, the fastest programming would occur if the start address is the start of a row and the end address is the end of the same row. Then only the faster row programming method would be used.) To this point, the routines are compatible. However, there is a range limitation on this second algorithm. If a range is passed that crosses an xxFF to xx00 boundary, then it will fail. The byte-by-byte algorithm does not have this limitation.

**Table B-1. Programming Routine Comparison**

| Routine | 908EY16 | 908EY16A |
|---|---|---|
| Page erase | I bit set on exit | I bit restored to value at time of call |
| Mass erase | Selected with CTRBYT | Selected with ADDR = FLBPR |
| Program | No restrictions on range to be programmed | If COPD = 1, range cannot include xxFF/xx00 boundary |

The maximum number of bytes required for the stack for all FLASH programming routines is now 13 bytes. In the 908EY16, the maximum number of stack locations was 12 bytes. The user should check to verify that enough stack space is set aside to accommodate this one byte increase.

**How to Reach Us:**

**Home Page:**
www.freescale.com

**Web Support:**
http://www.freescale.com/support

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
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. 2006, 2010. All rights reserved.

MC68HC908EY16A
Rev. 2, 09/2010

freescale™
semiconductor

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View MC908EY16AMFJE 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