Addendum to MC68HC908QB8, rev. 3

This addendum introduces a change to this data sheet.

Chapter 17 Development Support, Section 17.3.2 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 17 Development Support, Section 17.3.2 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.

MC68HC908QB8
MC68HC908QB4
MC68HC908QY8

Data Sheet

M68HC08
Microcontrollers

MC68HC908QB8
Rev. 3
04/2010

freescale.com

freescale™
semiconductor

MC68HC908QB8
MC68HC908QB4
MC68HC908QY8

Data Sheet

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

http://freescale.com/

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)
September, 2005	1	Initial full release	N/A
April, 2007	2	Chapter 3 Analog-to-Digital Converter (ADC10) Module — Renamed ADCSC register to ADSCR to be consistent with development tools.	37
		Figure 4-1. Auto Wakeup Interrupt Request Generation Logic — Changed BUSCLK4 to BUSCLK2 and removed reference to CGMXCLK	51
		4.3 Functional Description — Reworked for clarity	52
		4.6.4 Configuration Register 2 — Changed BUSCLK4 to BUSCLK2	55
		4.6.5 Configuration Register 1 — Changed BUSCLK4 to BUSCLK2, added bit description for SSREC, and removed SSREC from the note	56
		Chapter 5 Configuration Register (CONFIG) — Changed CGMXCLK to BUSCLKX4	57
		Chapter 13 Enhanced Serial Communications Interface (ESCI) Module — Changed SCIBDSRC to ESCIBDSRC, CGMXCLK to BUSCLKX4, and BUS_CLK to BUS CLOCK	109
		Table 13-5. ESCI LIN Control Bits — Corrected Functionality column	130
		13.9.3 Bit Time Measurement — Corrected first sentence of listing number 1	136
		Figure 17-18. Monitor Mode Entry Timing — Changed CGMXCLK to BUSCLKX4	206
April, 2010	3	Clarify internal oscillator trim register information.	29, 36, 96, 101

List of Chapters

Chapter 1
General Description 17

Chapter 2
Memory 23

Chapter 3
Analog-to-Digital Converter (ADC10) Module 37

Chapter 4
Auto Wakeup Module (AWU) 51

Chapter 5
Configuration Register (CONFIG) 57

Chapter 6
Computer Operating Properly (COP) 61

Chapter 7
Central Processor Unit (CPU) 65

Chapter 8
External Interrupt (IRQ) 77

Chapter 9
Keyboard Interrupt Module (KBI) 83

Chapter 10
Low-Voltage Inhibit (LVI) 89

Chapter 11
Oscillator Module (OSC) 93

Chapter 12
Input/Output Ports (PORTS) 103

Chapter 13
Enhanced Serial Communications Interface (ESCI) Module 109

Chapter 14
System Integration Module (SIM) 139

Chapter 15
Serial Peripheral Interface (SPI) Module 155

Chapter 16
Timer Interface Module (TIM) 175

Chapter 17
Development Support 191

Chapter 18
Electrical Specifications 207

Chapter 19
Ordering Information and Mechanical Specifications 227

Table of Contents

Chapter 1
General Description

1.1 Introduction . . . 17
1.2 Features . . . 17
1.3 MCU Block Diagram . . . 19
1.4 Pin Assignments . . . 20
1.5 Pin Functions . . . 20
1.6 Pin Function Priority . . . 22

Chapter 2
Memory

2.1 Introduction . . . 23
2.2 Unimplemented Memory Locations . . . 23
2.3 Reserved Memory Locations . . . 23
2.4 Direct Page Registers . . . 23
2.5 Random-Access Memory (RAM) . . . 30
2.6 FLASH Memory (FLASH) . . . 31
2.6.1 FLASH Control Register . . . 31
2.6.2 FLASH Page Erase Operation . . . 32
2.6.3 FLASH Mass Erase Operation . . . 33
2.6.4 FLASH Program Operation . . . 33
2.6.5 FLASH Protection . . . 34
2.6.6 FLASH Block Protect Register . . . 36

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

3.1 Introduction . . . 37
3.2 Features . . . 37
3.3 Functional Description . . . 37
3.3.1 Clock Select and Divide Circuit . . . 39
3.3.2 Input Select and Pin Control . . . 40
3.3.3 Conversion Control . . . 40
3.3.3.1 Initiating Conversions . . . 40
3.3.3.2 Completing Conversions . . . 40
3.3.3.3 Aborting Conversions . . . 40
3.3.3.4 Total Conversion Time . . . 41
3.3.4 Sources of Error . . . 42
3.3.4.1 Sampling Error . . . 42

3.3.4.2 Pin Leakage Error . . . 42
3.3.4.3 Noise-Induced Errors . . . 42
3.3.4.4 Code Width and Quantization Error . . . 43
3.3.4.5 Linearity Errors . . . 43
3.3.4.6 Code Jitter, Non-Monotonicity and Missing Codes . . . 43
3.4 Interrupts . . . 44
3.5 Low-Power Modes . . . 44
3.5.1 Wait Mode . . . 44
3.5.2 Stop Mode . . . 44
3.6 ADC10 During Break Interrupts . . . 44
3.7 I/O Signals . . . 45
3.7.1 ADC10 Analog Power Pin (VDDA) . . . 45
3.7.2 ADC10 Analog Ground Pin (VSSA) . . . 45
3.7.3 ADC10 Voltage Reference High Pin (VREFH) . . . 45
3.7.4 ADC10 Voltage Reference Low Pin (VREFL) . . . 45
3.7.5 ADC10 Channel Pins (ADn) . . . 46
3.8 Registers . . . 46
3.8.1 ADC10 Status and Control Register . . . 46
3.8.2 ADC10 Result High Register (ADRH) . . . 48
3.8.3 ADC10 Result Low Register (ADRL) . . . 48
3.8.4 ADC10 Clock Register (ADCLK) . . . 49

Chapter 4
Auto Wakeup Module (AWU)

4.1 Introduction . . . 51
4.2 Features . . . 51
4.3 Functional Description . . . 52
4.4 Interrupts . . . 52
4.5 Low-Power Modes . . . 53
4.5.1 Wait Mode . . . 53
4.5.2 Stop Mode . . . 53
4.6 Registers . . . 53
4.6.1 Port A I/O Register . . . 53
4.6.2 Keyboard Status and Control Register . . . 54
4.6.3 Keyboard Interrupt Enable Register . . . 54
4.6.4 Configuration Register 2 . . . 55
4.6.5 Configuration Register 1 . . . 55

Chapter 5
Configuration Register (CONFIG)

5.1 Introduction . . . 57
5.2 Functional Description . . . 57

Chapter 6
Computer Operating Properly (COP)

6.1 Introduction . . . 61
6.2 Functional Description . . . 61
6.3 I/O Signals . . . 62
6.3.1 BUSCLKX4 . . . 62
6.3.2 STOP Instruction . . . 62
6.3.3 COPCTL Write . . . 62
6.3.4 Power-On Reset . . . 62
6.3.5 Internal Reset . . . 62
6.3.6 COPD (COP Disable) . . . 62
6.3.7 COPRS (COP Rate Select) . . . 63
6.4 Interrupts . . . 63
6.5 Monitor Mode . . . 63
6.6 Low-Power Modes . . . 63
6.6.1 Wait Mode . . . 63
6.6.2 Stop Mode . . . 63
6.7 COP Module During Break Mode . . . 63
6.8 Register . . . 63

Chapter 7
Central Processor Unit (CPU)

7.1 Introduction . . . 65
7.2 Features . . . 65
7.3 CPU Registers . . . 65
7.3.1 Accumulator . . . 66
7.3.2 Index Register . . . 66
7.3.3 Stack Pointer . . . 67
7.3.4 Program Counter . . . 67
7.3.5 Condition Code Register . . . 68
7.4 Arithmetic/Logic Unit (ALU) . . . 69
7.5 Low-Power Modes . . . 69
7.5.1 Wait Mode . . . 69
7.5.2 Stop Mode . . . 69
7.6 CPU During Break Interrupts . . . 69
7.7 Instruction Set Summary . . . 70
7.8 Opcode Map . . . 75

Chapter 8
External Interrupt (IRQ)

8.1 Introduction . . . 77
8.2 Features . . . 77
8.3 Functional Description . . . 77
8.3.1 MODE = 1 . . . 79
8.3.2 MODE = 0 . . . 79
8.4 Interrupts . . . 80

8.5 Low-Power Modes . . . 80
8.5.1 Wait Mode . . . 80
8.5.2 Stop Mode . . . 80
8.6 IRQ Module During Break Interrupts . . . 80
8.7 I/O Signals . . . 80
8.7.1 IRQ Input Pins (IRQ) . . . 80
8.8 Registers . . . 81

Chapter 9
Keyboard Interrupt Module (KBI)

9.1 Introduction . . . 83
9.2 Features . . . 83
9.3 Functional Description . . . 83
9.3.1 Keyboard Operation . . . 83
9.3.1.1 MODEK = 1 . . . 84
9.3.1.2 MODEK = 0 . . . 85
9.3.2 Keyboard Initialization . . . 86
9.4 Interrupts . . . 86
9.5 Low-Power Modes . . . 86
9.5.1 Wait Mode . . . 86
9.5.2 Stop Mode . . . 86
9.6 KBI During Break Interrupts . . . 86
9.7 I/O Signals . . . 87
9.7.1 KBI Input Pins (KBIx:KBI0) . . . 87
9.8 Registers . . . 87
9.8.1 Keyboard Status and Control Register (KBSCR) . . . 87
9.8.2 Keyboard Interrupt Enable Register (KBIER) . . . 88
9.8.3 Keyboard Interrupt Polarity Register (KBIPR) . . . 88

Chapter 10
Low-Voltage Inhibit (LVI)

10.1 Introduction . . . 89
10.2 Features . . . 89
10.3 Functional Description . . . 89
10.3.1 Polled LVI Operation . . . 90
10.3.2 Forced Reset Operation . . . 90
10.3.3 LVI Hysteresis . . . 90
10.3.4 LVI Trip Selection . . . 90
10.4 LVI Interrupts . . . 91
10.5 Low-Power Modes . . . 91
10.5.1 Wait Mode . . . 91
10.5.2 Stop Mode . . . 91
10.6 Registers . . . 91

Chapter 11
Oscillator Module (OSC)

11.1 Introduction . . . 93
11.2 Features . . . 93
11.3 Functional Description . . . 93
11.3.1 Internal Signal Definitions . . . 95
11.3.1.1 Oscillator Enable Signal (SIMOSCEN) . . . 95
11.3.1.2 XTAL Oscillator Clock (XTALCLK) . . . 95
11.3.1.3 RC Oscillator Clock (RCCLK) . . . 95
11.3.1.4 Internal Oscillator Clock (INTCLK) . . . 95
11.3.1.5 Bus Clock Times 4 (BUSCLKX4) . . . 95
11.3.1.6 Bus Clock Times 2 (BUSCLKX2) . . . 95
11.3.2 Internal Oscillator . . . 95
11.3.2.1 Internal Oscillator Trimming . . . 96
11.3.2.2 Internal to External Clock Switching . . . 96
11.3.2.3 External to Internal Clock Switching . . . 96
11.3.3 External Oscillator . . . 96
11.3.4 XTAL Oscillator . . . 97
11.3.5 RC Oscillator . . . 98
11.4 Interrupts . . . 98
11.5 Low-Power Modes . . . 98
11.5.1 Wait Mode . . . 98
11.5.2 Stop Mode . . . 98
11.6 OSC During Break Interrupts . . . 99
11.7 I/O Signals . . . 99
11.7.1 Oscillator Input Pin (OSC1) . . . 99
11.7.2 Oscillator Output Pin (OSC2) . . . 99
11.8 Registers . . . 100
11.8.1 Oscillator Status and Control Register . . . 100
11.8.2 Oscillator Trim Register (OSCTRIM) . . . 101

Chapter 12
Input/Output Ports (PORTS)

12.1 Introduction . . . 103
12.2 Port A . . . 103
12.2.1 Port A Data Register . . . 104
12.2.2 Data Direction Register A . . . 104
12.2.3 Port A Input Pullup Enable Register . . . 105
12.2.4 Port A Summary Table . . . 106
12.3 Port B . . . 106
12.3.1 Port B Data Register . . . 106
12.3.2 Data Direction Register B . . . 107
12.3.3 Port B Input Pullup Enable Register . . . 108
12.3.4 Port B Summary Table . . . 108

Chapter 13

Enhanced Serial Communications Interface (ESCI) Module

13.1 Introduction . . . 109
13.2 Features . . . 109
13.3 Functional Description . . . 111
13.3.1 Data Format . . . 112
13.3.2 Transmitter . . . 112
13.3.2.1 Character Length . . . 113
13.3.2.2 Character Transmission . . . 113
13.3.2.3 Break Characters . . . 113
13.3.2.4 Idle Characters . . . 114
13.3.2.5 Inversion of Transmitted Output . . . 114
13.3.3 Receiver . . . 114
13.3.3.1 Character Length . . . 114
13.3.3.2 Character Reception . . . 114
13.3.3.3 Data Sampling . . . 116
13.3.3.4 Framing Errors . . . 117
13.3.3.5 Baud Rate Tolerance . . . 117
13.3.3.6 Receiver Wakeup . . . 119
13.4 Interrupts . . . 119
13.4.1 Transmitter Interrupts . . . 120
13.4.2 Receiver Interrupts . . . 120
13.4.3 Error Interrupts . . . 120
13.5 Low-Power Modes . . . 120
13.5.1 Wait Mode . . . 120
13.5.2 Stop Mode . . . 121
13.6 ESCI During Break Interrupts . . . 121
13.7 I/O Signals . . . 121
13.7.1 ESCI Transmit Data (TxD) . . . 121
13.7.2 ESCI Receive Data (RxD) . . . 121
13.8 Registers . . . 121
13.8.1 ESCI Control Register 1 . . . 122
13.8.2 ESCI Control Register 2 . . . 123
13.8.3 ESCI Control Register 3 . . . 125
13.8.4 ESCI Status Register 1 . . . 126
13.8.5 ESCI Status Register 2 . . . 129
13.8.6 ESCI Data Register . . . 129
13.8.7 ESCI Baud Rate Register . . . 130
13.8.8 ESCI Prescaler Register . . . 131
13.9 ESCI Arbiter . . . 135
13.9.1 ESCI Arbiter Control Register . . . 135
13.9.2 ESCI Arbiter Data Register . . . 136
13.9.3 Bit Time Measurement . . . 136
13.9.4 Arbitration Mode . . . 138

Chapter 14
System Integration Module (SIM)

14.1 Introduction 139
14.2 RST and IRQ Pins Initialization 139
14.3 SIM Bus Clock Control and Generation 140
14.3.1 Bus Timing 141
14.3.2 Clock Start-Up from POR 141
14.3.3 Clocks in Stop Mode and Wait Mode 141
14.4 Reset and System Initialization 141
14.4.1 External Pin Reset 141
14.4.2 Active Resets from Internal Sources 142
14.4.2.1 Power-On Reset 143
14.4.2.2 Computer Operating Properly (COP) Reset 143
14.4.2.3 Illegal Opcode Reset 143
14.4.2.4 Illegal Address Reset 144
14.4.2.5 Low-Voltage Inhibit (LVI) Reset 144
14.5 SIM Counter 144
14.5.1 SIM Counter During Power-On Reset 144
14.5.2 SIM Counter During Stop Mode Recovery 144
14.5.3 SIM Counter and Reset States 144
14.6 Exception Control 145
14.6.1 Interrupts 145
14.6.1.1 Hardware Interrupts 145
14.6.1.2 SWI Instruction 148
14.6.2 Interrupt Status Registers 148
14.6.2.1 Interrupt Status Register 1 149
14.6.2.2 Interrupt Status Register 2 149
14.6.2.3 Interrupt Status Register 3 149
14.6.3 Reset 150
14.6.4 Break Interrupts 150
14.6.5 Status Flag Protection in Break Mode 150
14.7 Low-Power Modes 150
14.7.1 Wait Mode 150
14.7.2 Stop Mode 151
14.8 SIM Registers 152
14.8.1 SIM Reset Status Register 152
14.8.2 Break Flag Control Register 153

Chapter 15
Serial Peripheral Interface (SPI) Module

15.1 Introduction 155
15.2 Features 155
15.3 Functional Description 157
15.3.1 Master Mode 158
15.3.2 Slave Mode 158
15.3.3 Transmission Formats 159
15.3.3.1 Clock Phase and Polarity Controls 159
15.3.3.2 Transmission Format When CPHA = 0 159

15.3.3.3 Transmission Format When CPHA = 1 . . . 160
15.3.3.4 Transmission Initiation Latency . . . 161
15.3.4 Queuing Transmission Data . . . 163
15.3.5 Resetting the SPI . . . 164
15.3.6 Error Conditions . . . 164
15.3.6.1 Overflow Error . . . 164
15.3.6.2 Mode Fault Error . . . 166
15.4 Interrupts . . . 167
15.5 Low-Power Modes . . . 168
15.5.1 Wait Mode . . . 168
15.5.2 Stop Mode . . . 168
15.6 SPI During Break Interrupts . . . 168
15.7 I/O Signals . . . 169
15.7.1 MISO (Master In/Slave Out) . . . 169
15.7.2 MOSI (Master Out/Slave In) . . . 169
15.7.3 SPSCK (Serial Clock) . . . 169
15.7.4 $\overline{SS}$ (Slave Select) . . . 169
15.8 Registers . . . 170
15.8.1 SPI Control Register . . . 171
15.8.2 SPI Status and Control Register . . . 172
15.8.3 SPI Data Register . . . 174

Chapter 16
Timer Interface Module (TIM)

16.1 Introduction . . . 175
16.2 Features . . . 175
16.3 Functional Description . . . 175
16.3.1 TIM Counter Prescaler . . . 175
16.3.2 Input Capture . . . 176
16.3.3 Output Compare . . . 177
16.3.3.1 Unbuffered Output Compare . . . 178
16.3.3.2 Buffered Output Compare . . . 178
16.3.4 Pulse Width Modulation (PWM) . . . 179
16.3.4.1 Unbuffered PWM Signal Generation . . . 179
16.3.4.2 Buffered PWM Signal Generation . . . 180
16.3.4.3 PWM Initialization . . . 181
16.4 Interrupts . . . 182
16.5 Low-Power Modes . . . 182
16.5.1 Wait Mode . . . 182
16.5.2 Stop Mode . . . 182
16.6 TIM During Break Interrupts . . . 182
16.7 I/O Signals . . . 183
16.7.1 TIM Channel I/O Pins (TCH3:TCH0) . . . 183
16.7.2 TIM Clock Pin (TCLK) . . . 183
16.8 Registers . . . 183
16.8.1 TIM Status and Control Register . . . 183

16.8.2 TIM Counter Registers . . . 185
16.8.3 TIM Counter Modulo Registers . . . 185
16.8.4 TIM Channel Status and Control Registers . . . 186
16.8.5 TIM Channel Registers . . . 188

Chapter 17
Development Support

17.1 Introduction . . . 191
17.2 Break Module (BRK) . . . 191
17.2.1 Functional Description . . . 191
17.2.1.1 Flag Protection During Break Interrupts . . . 193
17.2.1.2 TIM During Break Interrupts . . . 193
17.2.1.3 COP During Break Interrupts . . . 193
17.2.2 Break Module Registers . . . 194
17.2.2.1 Break Status and Control Register . . . 194
17.2.2.2 Break Address Registers . . . 194
17.2.2.3 Break Auxiliary Register . . . 195
17.2.2.4 Break Status Register . . . 195
17.2.2.5 Break Flag Control Register . . . 195
17.2.3 Low-Power Modes . . . 196
17.3 Monitor Module (MON) . . . 196
17.3.1 Functional Description . . . 196
17.3.1.1 Normal Monitor Mode . . . 200
17.3.1.2 Forced Monitor Mode . . . 201
17.3.1.3 Monitor Vectors . . . 201
17.3.1.4 Data Format . . . 202
17.3.1.5 Break Signal . . . 202
17.3.1.6 Baud Rate . . . 202
17.3.1.7 Commands . . . 202
17.3.2 Security . . . 206

Chapter 18
Electrical Specifications

18.1 Introduction . . . 207
18.2 Absolute Maximum Ratings . . . 207
18.3 Functional Operating Range . . . 208
18.4 Thermal Characteristics . . . 208
18.5 5-V DC Electrical Characteristics . . . 209
18.6 Typical 5-V Output Drive Characteristics . . . 210
18.7 5-V Control Timing . . . 211
18.8 3-V DC Electrical Characteristics . . . 212
18.9 Typical 3-V Output Drive Characteristics . . . 213
18.10 3-V Control Timing . . . 214
18.11 Oscillator Characteristics . . . 215
18.12 Supply Current Characteristics . . . 217

18.13 ADC10 Characteristics 219
18.14 5.0-Volt SPI Characteristics 221
18.15 3.0-Volt SPI Characteristics 222
18.16 Timer Interface Module Characteristics 225
18.17 Memory Characteristics 226

Chapter 19
Ordering Information and Mechanical Specifications

19.1 Introduction 227
19.2 MC Order Numbers 227
19.3 Package Dimensions 227

Chapter 1
General Description

1.1 Introduction

The MC68HC908QB8 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.

Table 1-1. Summary of Device Variations

Device	FLASH/RAM Memory Size	ADC	16-Bit Timer Channels	ESCI	SPI	Pin Count
MC68HC908QB8	8K/256 bytes	10 channel, 10 bit	4	Yes	Yes	16 pins
MC68HC908QB4	4K/128 bytes	10 channel, 10 bit	4	Yes	Yes	16 pins
MC68HC908QY8	8K/256 bytes	4 channel, 10 bit	2	No	No	16 pins

1.2 Features

Features include:

- High-performance M68HC08 CPU core
- Fully upward-compatible object code with M68HC05 Family
- 5-V and 3-V operating voltages (V_{DD})
- 8-MHz internal bus operation at 5 V, 4-MHz at 3 V
- Trimmable internal oscillator
  - Software selectable 1 MHz, 2 MHz, or 3.2 MHz internal bus operation
  - 8-bit trim capability
  - ± 25% untrimmed
  - Trimmable to approximately 0.4%(1)
- Software selectable crystal oscillator range, 32–100 kHz, 1–8 MHz, and 8–32 MHz
- Software configurable input clock from either internal or external source
- Auto wakeup from STOP capability using dedicated internal 32-kHz RC or bus clock source
- On-chip in-application programmable FLASH memory
  - Internal program/erase voltage generation
  - Monitor ROM containing user callable program/erase routines
  - FLASH security(2)

1. See 18.11 Oscillator Characteristics for internal oscillator specifications
2. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH difficult for unauthorized users.

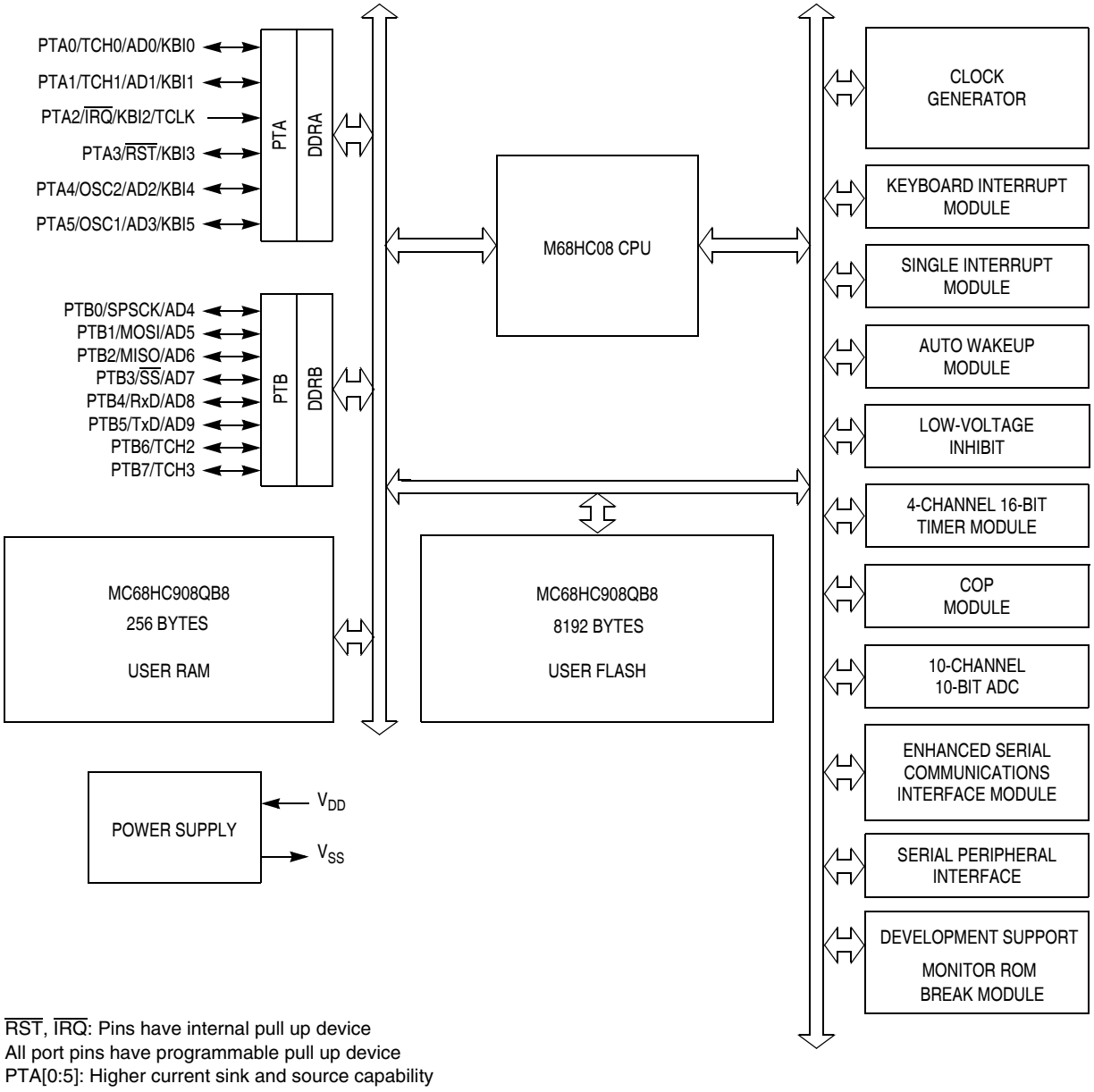
- On-chip random-access memory (RAM)
- Enhanced serial communications interface (ESCI) module
- Serial peripheral interface (SPI) module
- 4-channel, 16-bit timer interface (TIM) module
- 10-channel, 10-bit analog-to-digital converter (ADC) with internal bandgap reference channel (ADC10)
- Up to 13 bidirectional input/output (I/O) lines and one input only:
  - Six shared with KBI
  - Ten shared with ADC
  - Four shared with TIM
  - Two shared with ESCI
  - Four shared with SPI
  - One input only shared with IRQ
  - High current sink/source capability on all port pins
  - Selectable pullups on all ports, selectable on an individual bit basis
  - Three-state ability on all port pins
- 6-bit keyboard interrupt with wakeup feature (KBI)
  - Programmable for rising/falling or high/low level detect
- Low-voltage inhibit (LVI) module features:
  - Software selectable trip point
- System protection features:
  - Computer operating properly (COP) watchdog
  - Low-voltage detection with reset
  - Illegal opcode detection with reset
  - Illegal address detection with reset
- External asynchronous interrupt pin with internal pullup ($\overline{\text{IRQ}}$) shared with general-purpose input pin
- Master asynchronous reset pin with internal pullup ($\overline{\text{RST}}$) shared with general-purpose input/output (I/O) pin
- Memory mapped I/O registers
- Power saving stop and wait modes
- MC68HC908QB8, MC68HC908QB4 and MC68HC908QY8 are available in these packages:
  - 16-pin plastic dual in-line package (PDIP)
  - 16-pin small outline integrated circuit (SOIC) package
  - 16-pin thin shrink small outline packages (TSSOP)

Features of the CPU08 include the following:

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

1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908QB8.



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device
PTA[0:5]: Higher current sink and source capability

Figure 1-1. Block Diagram

1.4 Pin Assignments

The MC68HC908QB8, MC68HC908QB4, and MC68HC908QY8 are available in 16-pin packages. Figure 1-2 shows the pin assignment for these packages.

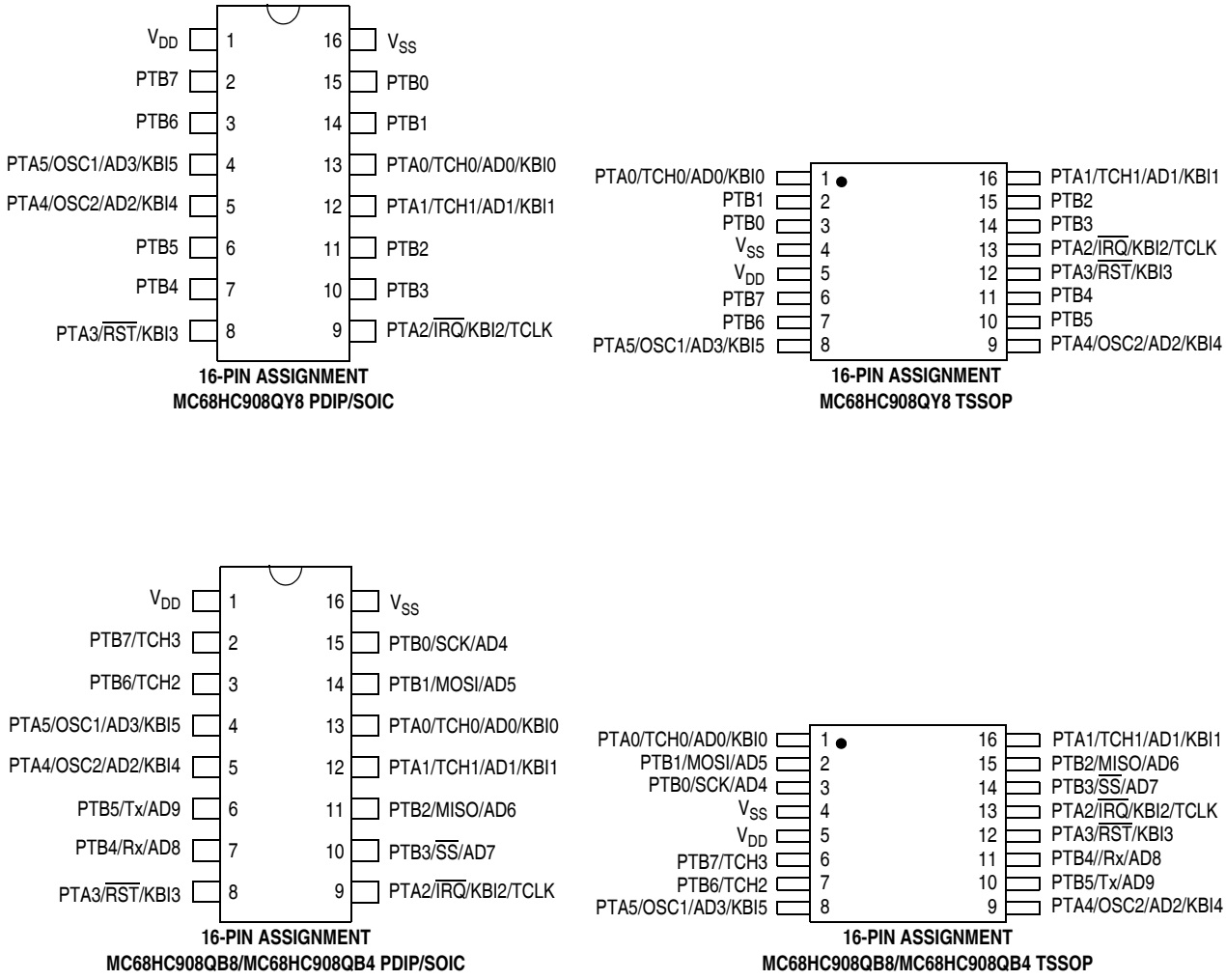


Figure 1-2. MCU Pin Assignments

1.5 Pin Functions

Table 1-2 provides a description of the pin functions.

Table 1-2. Pin Functions

Pin Name	Description	Input/Output
V_{DD}	Power supply	Power
V_{SS}	Power supply ground	Power
PTA0	PTA0 — General purpose I/O port	Input/Output
	TCH0 — Timer Channel 0 I/O	Input/Output
	AD0 — A/D channel 0 input	Input
	KBI0 — Keyboard interrupt input 0	Input
PTA1	PTA1 — General purpose I/O port	Input/Output
	TCH1 — Timer Channel 1 I/O	Input/Output
	AD1 — A/D channel 1 input	Input
	KBI1 — Keyboard interrupt input 1	Input
PTA2	PTA2 — General purpose input-only port	Input
	$\overline{IRQ}$ — External interrupt with programmable pullup and Schmitt trigger input	Input
	KBI2 — Keyboard interrupt input 2	Input
	TCLK — Timer clock input	Input
PTA3	PTA3 — General purpose I/O port	Input/Output
	$\overline{RST}$ — Reset input, active low with internal pullup and Schmitt trigger	Input
	KBI3 — Keyboard interrupt input 3	Input
PTA4	PTA4 — General purpose I/O port	Input/Output
	OSC2 —XTAL oscillator output (XTAL option only) RC or internal oscillator output (OSC2EN = 1 in PTAPUE register)	Output Output
	AD2 — A/D channel 2 input	Input
	KBI4 — Keyboard interrupt input 4	Input
PTA5	PTA5 — General purpose I/O port	Input/Output
	OSC1 — XTAL, RC, or external oscillator input	Input
	AD3 — A/D channel 3 input	Input
	KBI5 — Keyboard interrupt input 5	Input
PTB0	PTB0 — General-purpose I/O port	Input/Output
	SPSCK — SPI serial clock	Input/Output
	AD4 — A/D channel 4 input	Input
PTB1	PTB1 — General-purpose I/O port	Input/Output
	MOSI — SPI Master out Slave in	Input/Output
	AD5 — A/D channel 5 input	Input
PTB2	PTB2 — General-purpose I/O port	Input/Output
	MISO — SPI Master in Slave out	Input/Output
	AD6 — A/D channel 6 input	Input
PTB3	PTB3 — General-purpose I/O port	Input/Output
	$\overline{SS}$ — SPI slave select	Input
	AD7 — A/D channel 7 input	Input

— Continued on next page

Table 1-2. Pin Functions (Continued)

Pin Name	Description	Input/Output
PTB4	PTB4 — General-purpose I/O port	Input/Output
	RxD — ESCI receive data I/O	Input/Output
	AD8 — A/D channel 8 input	Input
PTB5	PTB5 — General-purpose I/O port	Input/Output
	TxD — ESCI transmit data I/O	Output
	AD9 — A/D channel 9 input	Input
PTB6	PTB6 — General-purpose I/O port	Input/Output
	TCH2 — Timer channel 2 I/O	Input/Output
PTB7	PTB7 — General-purpose I/O port	Input/Output
	TCH3 — Timer channel 3 I/O	Input/Output

1.6 Pin Function Priority

Table 1-3 is meant to resolve the priority if multiple functions are enabled on a single pin.

NOTE

Upon reset all pins come up as input ports regardless of the priority table.

Table 1-3. Function Priority in Shared Pins

Pin Name	Highest-to-Lowest Priority Sequence
PTA0[1]	AD0 → TCH0 → KBI0 → PTA0
PTA1[1]	AD1 → TCH1 → KBI1 → PTA1
PTA2	$\overline{IRQ}$ → TCLK → KBI2 → PTA2
PTA3	$\overline{RST}$ → KBI3 → PTA3
PTA4[1]	OSC2 → AD2 → KBI4 → PTA4
PTA5[1]	OSC1 → AD3 → KBI5 → PTA5
PTB0[1]	AD4 → SPSCK → PTB0
PTB1[1]	AD5 → MOSI → PTB1
PTB2[1]	AD6 → MISO → PTB2
PTB3[1]	AD7 → $\overline{SS}$ → PTB3
PTB4[1]	AD8 → RxD → PTB4
PTB5[1]	AD9 → TxD → PTB5
PTB6	TCH2 → PTB6
PTB7	TCH3 → PTB7

1. When a pin is to be used as an ADC pin, the I/O port function should be left as an input and all other shared modules should be disabled. The ADC does not override additional modules using the pin.

Chapter 2
Memory

2.1 Introduction

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

2.2 Unimplemented Memory Locations

Executing code from an unimplemented location will cause an illegal address reset. In Figure 2-1, unimplemented locations are shaded.

2.3 Reserved Memory Locations

Accessing a reserved location can have unpredictable effects on MCU operation. In Figure 2-1, register locations are marked with the word Reserved or with the letter R.

2.4 Direct Page Registers

Figure 2-2 shows the memory mapped registers of the MC68HC908QB8. Registers with addresses between $0000 and $00FF are considered direct page registers and all instructions including those with direct page addressing modes can access them. Registers between $0100 and $FFFF require non-direct page addressing modes. See Chapter 7 Central Processor Unit (CPU) for more information on addressing modes.

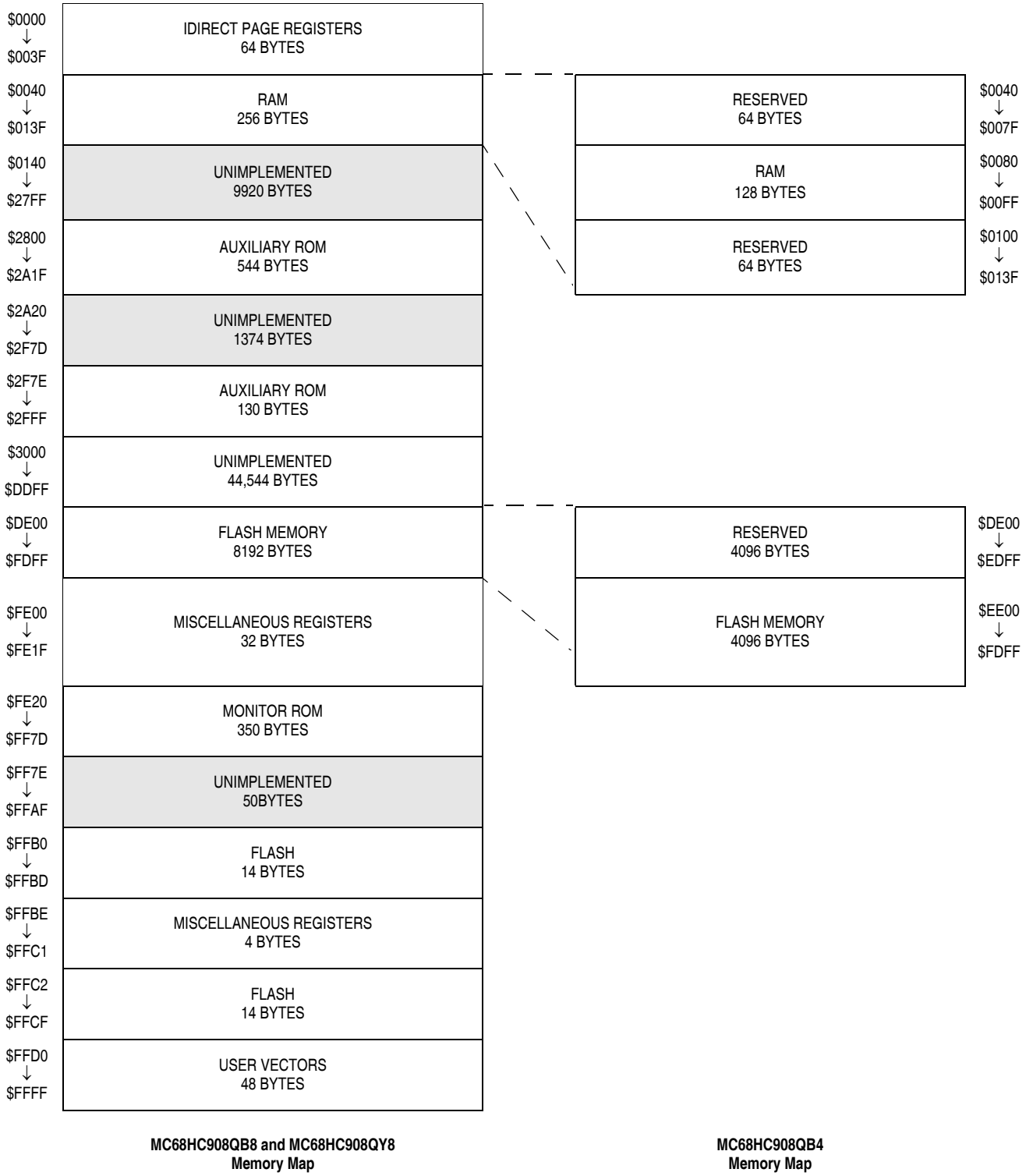


Figure 2-1. Memory Map

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA) See page 104.	Read:	R	AWUL	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB) See page 106.	Read:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Write:								
		Reset:	Unaffected by reset							
$0002 ↓ $0003	Reserved									
$0004	Data Direction Register A (DDRA) See page 104.	Read:	R	R	DDRA5	DDRA4	DDRA3	0	DDRA1	DDRA0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB) See page 107.	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0006 ↓ $000A	Reserved									
$000B	Port A Input Pullup Enable Register (PTAPUE) See page 105.	Read:	OSC2EN	0	PTAPUE5	PTAPUE4	PTAPUE3	PTAPUE2	PTAPUE1	PTAPUE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000C	Port B Input Pullup Enable Register (PTBPUE) See page 108.	Read:	PTBPUE7	PTBPUE6	PTBPUE5	PTBPUE4	PTBPUE3	PTBPUE2	PTBPUE1	PTBPUE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000D	SPI Control Register (SPCR) See page 171.	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 172.	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 174.	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0010	ESCI Control Register 1 (SCC1) See page 122.	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 124.	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 125.	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 126.	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 1 of 5)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0014	ESCI Status Register 2 (SCS2) See page 129.	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 129.	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 130.	Read: Write:	LINT	LINR	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Reset:	0	0	0	0	0	0	0	0
$0017	ESCI Prescaler Register (SCPSC) See page 131.	Read: Write:	PDS2	PDS1	PDS0	PSSB4	PSSB3	PSSB2	PSSB1	PSSB0
		Reset:	0	0	0	0	0	0	0	0
$0018	ESCI Arbiter Control Register (SCIACTL) See page 135.	Read:	AM1	ALOST	AM0	ACLK	AFIN	ARUN	AROVFL	ARD8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0019	ESCI Arbiter Data Register (SCIADAT) See page 136.	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 87.	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 88.	Read:	0	AWUIE	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001C	Keyboard Interrupt Polarity Register (KBIPR) See page 88.	Read:	0	0	KBIP5	KBIP4	KBIP3	KBIP2	KBIP1	KBIP0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001D	IRQ Status and Control Register (INTSCR) See page 81.	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0
$001E	Configuration Register 2 (CONFIG2)(1) See page 57.	Read:	IRQPUD	IRQEN	R	R	R	ESCIBDSRC	OSCENIN-STOP	RSTEN
		Write:								
		Reset:	0	0	0	0	0	0	0	0(2)

1. One-time writable register after each reset.
2. RSTEN reset to 0 by a power-on reset (POR) only.

$001F	Configuration Register 1 (CONFIG1)(1) See page 58.	Read:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVITRIP	SSREC	STOP	COPD
		Write:								
		Reset:	0	0	0	0	0(2)	0	0	0

1. One-time writable register after each reset.
2. LVITRIP reset to 0 by a power-on reset (POR) only.

$0020	TIM Status and Control Register (TSC) See page 183.	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0021	TIM Counter Register High (TCNTH) See page 185.	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 2 of 5)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0022	TIM Counter Register Low (TCNTL) See page 185.	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	TIM Counter Modulo Register High (TMODH) See page 185.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0024	TIM Counter Modulo Register Low (TMODL) See page 185.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	TIM Channel 0 Status and Control Register (TSC0) See page 186.	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0026	TIM Channel 0 Register High (TCH0H) See page 189.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0027	TIM Channel 0 Register Low (TCH0L) See page 189.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0028	TIM Channel 1 Status and Control Register (TSC1) See page 186.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0029	TIM Channel 1 Register High (TCH1H) See page 189.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002A	TIM Channel 1 Register Low (TCH1L) See page 189.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$002B ↓ $002F	Reserved									
$0030	TIM Channel 2 Status and Control Register (TSC2) See page 186.	Read:	CH2F	CH2IE	0	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0031	TIM Channel 2 Register High (TCH2H) See page 189.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0032	TIM Channel 2 Register Low (TCH2L) See page 189.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0033	TIM Channel 3 Status and Control Register (TSC3) See page 186.	Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0034	TIM Channel 3 Register High (TCH3H) See page 189.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0035	TIM Channel 3 Register Low (TCH3L) See page 189.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							

= Unimplemented R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 5)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0036	Oscillator Status and Control Register (OSCSC) See page 100.	Read:	OSCOPT1	OSCOPT0	ICFS1	ICFS0	ECFS1	ECFS0	ECGON	ECGST
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0037	Reserved									
$0038	Oscillator Trim Register (OSCTRIM) See page 101.	Read:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
		Write:								
		Reset:	1	0	0	0	0	0	0	0
$0039 ↓ $003B	Reserved									
$003C	ADC10 Status and Control Register (ADSCR) See page 46.	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:								
		Reset:	0	0	0	1	1	1	1	1
$003D	ADC10 Data Register High (ADRH) See page 48.	Read:	0	0	0	0	0	0	AD9	AD8
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$003E	ADC10 Data Register Low (ADRL) See page 48.	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$003F	ADC10 Clock Register (ADCLK) See page 49.	Read:	ADLPC	ADIV1	ADIV0	ADICLK	MODE1	MODE0	ADLSMP	ACLKEN
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE00	Break Status Register (BSR) See page 195.	Read:	R	R	R	R	R	R	SBSW	R
		Write:							0	
		Reset:							0	
$FE01	SIM Reset Status Register (SRSR) See page 152.	Read:	POR	PIN	COP	ILOP	ILAD	MODRST	LVI	0
		Write:								
		POR:	1	0	0	0	0	0	0	0
$FE02	Break Auxiliary Register (BRKAR) See page 195.	Read:	0	0	0	0	0	0	0	BDCOP
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE03	Break Flag Control Register (BFCR) See page 195.	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE04	Interrupt Status Register 1 (INT1) See page 149.	Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE05	Interrupt Status Register 2 (INT2) See page 149.	Read:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE06	Interrupt Status Register 3 (INT3) See page 149.	Read:	IF22	IF21	IF20	IF19	IF18	IF17	IF16	IF15
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 5)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE07	Reserved									
$FE08	FLASH Control Register (FLCR) See page 31.	Read:	0	0	0	0	HVEN	MASS	ERASE	PGM
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE09	Break Address High Register (BRKH) See page 194.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0A	Break Address low Register (BRKL) See page 194.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0B	Break Status and Control Register (BRKSCR) See page 194.	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 91.	Read:	LVIOUT	0	0	0	0	0	0	R
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0D ↓ $FE0F	Reserved									
$FFBE	FLASH Block Protect Register (FLBPR) See page 36.	Read:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
		Write:								
		Reset:	Unaffected by reset							
$FFBF	Reserved									
$FFC0	Internal Oscillator Trim (Factory Programmed, V_{DD} = 3.0 V)	Read:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
		Write:								
		Reset:	Resets to factory programmed value							
$FFC1	Reserved									
$FFFF	COP Control Register (COPCTL) See page 63.	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 5 of 5)

Table 2-1. Vector Addresses

Vector Priority	Vector	Address	Vector
Lowest	IF22–IF16	$FFD0–$FFDC	Not used
↑	IF15	$FFDE,F	ADC conversion complete vector
	IF14	$FFE0,1	Keyboard vector
	IF13	$FFE2,3	SPI transmit vector
	IF12	$FFE4,5	SPI receive vector
	IF11	$FFE6,7	ESCI transmit vector
	IF10	$FFE8,9	ESCI receive vector
	IF9	$FFEA,B	ESCI error vector
	IF8	—	Not used
	IF7	$FFEE,F	TIM1 Channel 3 vector
	IF6	$FFF0,1	TIM1 Channel 2 vector
	IF5	$FFF2,3	TIM1 overflow vector
	IF4	$FFF4,5	TIM1 Channel 1 vector
	IF3	$FFF6,7	TIM1 Channel 0 vector
	IF2	—	Not used
	IF1	$FFFA,B	$\overline{\text{IRQ}}$ vector
	—	$FFFC,D	SWI vector
Highest	—	$FFFE,F	Reset vector

2.5 Random-Access Memory (RAM)

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

The RAM retains data when the MCU is in low-power wait or stop mode. At power-on, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention.

For compatibility with older M68HC05 MCUs, the HC08 resets the stack pointer to $00FF. In the devices that have RAM above $00FF, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables. Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM).

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

2.6 FLASH Memory (FLASH)

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

This subsection describes the operation of the embedded FLASH memory. The FLASH memory can be read, programmed, and erased from the internal V_{DD} supply. The program and erase operations are enabled through the use of an internal charge pump.

The minimum size of FLASH memory that can be erased is 64 bytes; and the maximum size of FLASH memory that can be programmed in a program cycle is 32 bytes (a row). Program and erase operations are facilitated through control bits in the FLASH control register (FLCR). Details for these operations appear later in this section.

NOTE

An erased bit reads as a 1 and a programmed bit reads as a 0. A security feature prevents viewing of the FLASH contents.[1]

2.6.1 FLASH Control Register

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

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

= Unimplemented

Figure 2-3. FLASH Control Register (FLCR)

HVEN — High Voltage Enable Bit

This read/write bit enables high voltage from the charge pump to the memory for either program or erase operation. It can only be set if either PGM =1 or ERASE =1 and the proper sequence for program or erase is followed.

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

MASS — Mass Erase Control Bit

This read/write bit configures the memory for mass erase operation.

1 = Mass erase operation selected
0 = Mass erase operation unselected

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

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 the following procedure to erase a page of FLASH memory. A page consists of 64 consecutive bytes starting from addresses $XX00, $XX40, $XX80, or $XXC0. The 48-byte user interrupt vectors area also forms a page. Any FLASH memory page can be erased alone.

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}.
5. Set the HVEN bit.
6. Wait for a time, t_{Erase}.
7. Clear the ERASE bit.
8. Wait for a time, t_{NVH}.
9. Clear the HVEN bit.
10. After time, t_{RCV}, the memory can be accessed in read mode again.

NOTE

The COP register at location $FFFF should not be written between steps 5-9, when the HVEN bit is set. Since this register is located at a valid FLASH address, unpredictable behavior may occur if this location is written while HVEN is set.

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, other unrelated operations may occur between the steps.

CAUTION

A page erase of the vector page will erase the internal oscillator trim value at $FFC0.

2.6.3 FLASH Mass Erase Operation

Use the following procedure to erase the entire FLASH memory to read as a 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}.
5. Set the HVEN bit.
6. Wait for a time, t_{MErase}.
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}.
9. Clear the HVEN bit.
10. After time, t_{RCV}, 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, other unrelated operations may occur between the steps.

CAUTION

A mass erase will erase the internal oscillator trim value at $FFC0.

2.6.4 FLASH Program 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, or $XXE0. Use the following step-by-step procedure to program a row of FLASH memory

Figure 2-4 shows a flowchart of the programming algorithm.

NOTE

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

1. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
2. Read the FLASH block protect register.
3. Write any data to any FLASH location within the address range desired.
4. Wait for a time, t_{NVS}.
5. Set the HVEN bit.

1. When in monitor mode, with security sequence failed (see 17.3.2 Security), write to the FLASH block protect register instead of any FLASH address.

6. Wait for a time, t_{PGS}.
7. Write data to the FLASH address being programmed(1).
8. Wait for time, t_{PROG}.
9. Repeat step 7 and 8 until all desired bytes within the row are programmed.
10. Clear the PGM bit (1).
11. Wait for time, t_{NVH}.
12. Clear the HVEN bit.
13. After time, t_{RCV}, the memory can be accessed in read mode again.

NOTE

The COP register at location $FFFF should not be written between steps 5-12, when the HVEN bit is set. Since this register is located at a valid FLASH address, unpredictable behavior may occur if this location is written while HVEN is set.

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, see 18.17 Memory Characteristics.

2.6.5 FLASH 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 to protect blocks of memory from unintentional erase or program operations due to system malfunction. This protection is done by use of a 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 to 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

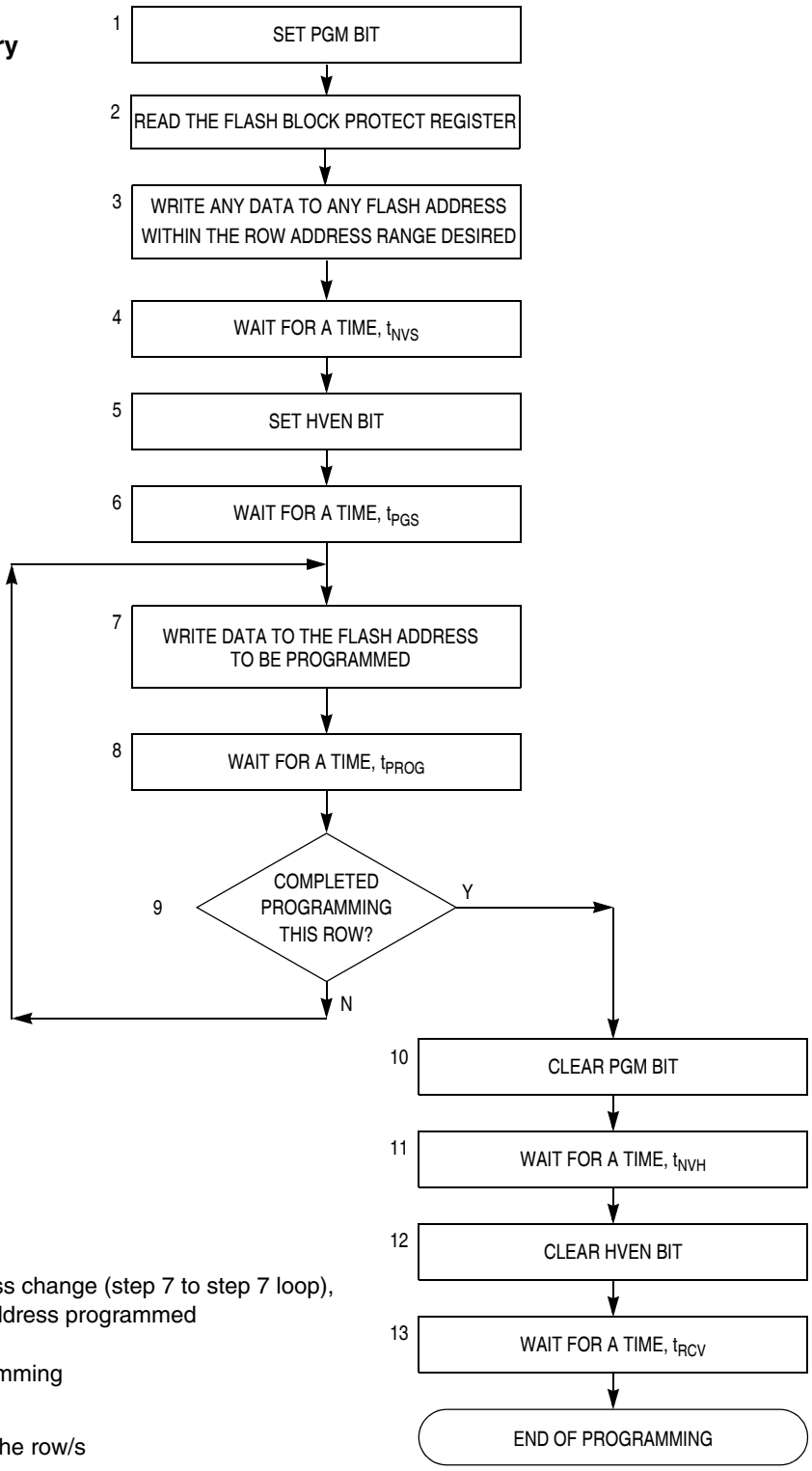
In performing a program or erase operation, the FLASH block protect register must be read after setting the PGM or ERASE bit and before asserting the HVEN bit.

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

When bits within the FLBPR are programmed, they lock a block of memory. The address ranges are shown in 2.6.6 FLASH Block Protect Register. Once the FLBPR is programmed with a value other than $FF, 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 FLBPR itself can be erased or programmed only with an external voltage, V_{TST}, present on the $\overline{IRQ}$ pin. This voltage also allows entry from reset into the monitor mode.

1. The time between each FLASH address change, or the time between the last FLASH address programmed to clearing 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 loop), 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} max.

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

Figure 2-4. FLASH Programming Flowchart

2.6.6 FLASH Block Protect Register

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

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
Reset:	Unaffected by reset. Initial value from factory is 1.							

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

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

BPR[7:0] — FLASH Protection Register Bits [7:0]

These eight bits in FLBPR represent bits [13:6] of a 16-bit memory address. Bits [15:14] 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, or XXC0 within the FLASH memory. See Figure 2-6 and Table 2-2.

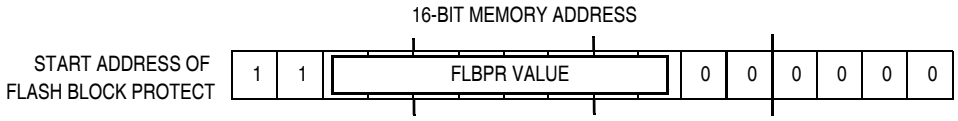


Figure 2-6. FLASH Block Protect Start Address

Table 2-2. Examples of Protect Start Address

BPR[7:0]	Start of Address of Protect Range
$00–$77	The entire FLASH memory is protected.
$78 (**0111 1000**)	$DE00 (11**01 1110 00**00 0000)
$79 (**0111 1001**)	$DE40 (11**01 1110 01**00 0000)
$7A (**0111 1010**)	$DE80 (11**01 1110 10**00 0000)
$7B (**0111 1011**)	$DFC0 (11**01 1110 11**00 0000)
and so on...	
$DE (**1101 1110**)	$F780 (11**11 0111 10**00 0000)
$DF (**1101 1111**)	$F7C0 (11**11 0111 11**00 0000)
$FE (**1111 1110**)	$FF80 (11**11 1111 10**00 0000) FLBPR, internal oscillator trim value, and vectors are protected
$FF	The entire FLASH memory is not protected.

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_{DD} and V_{SS} as its supply and reference pins. This MCU uses BUSCLKX4 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
- Selectable asynchronous hardware conversion trigger

3.3 Functional Description

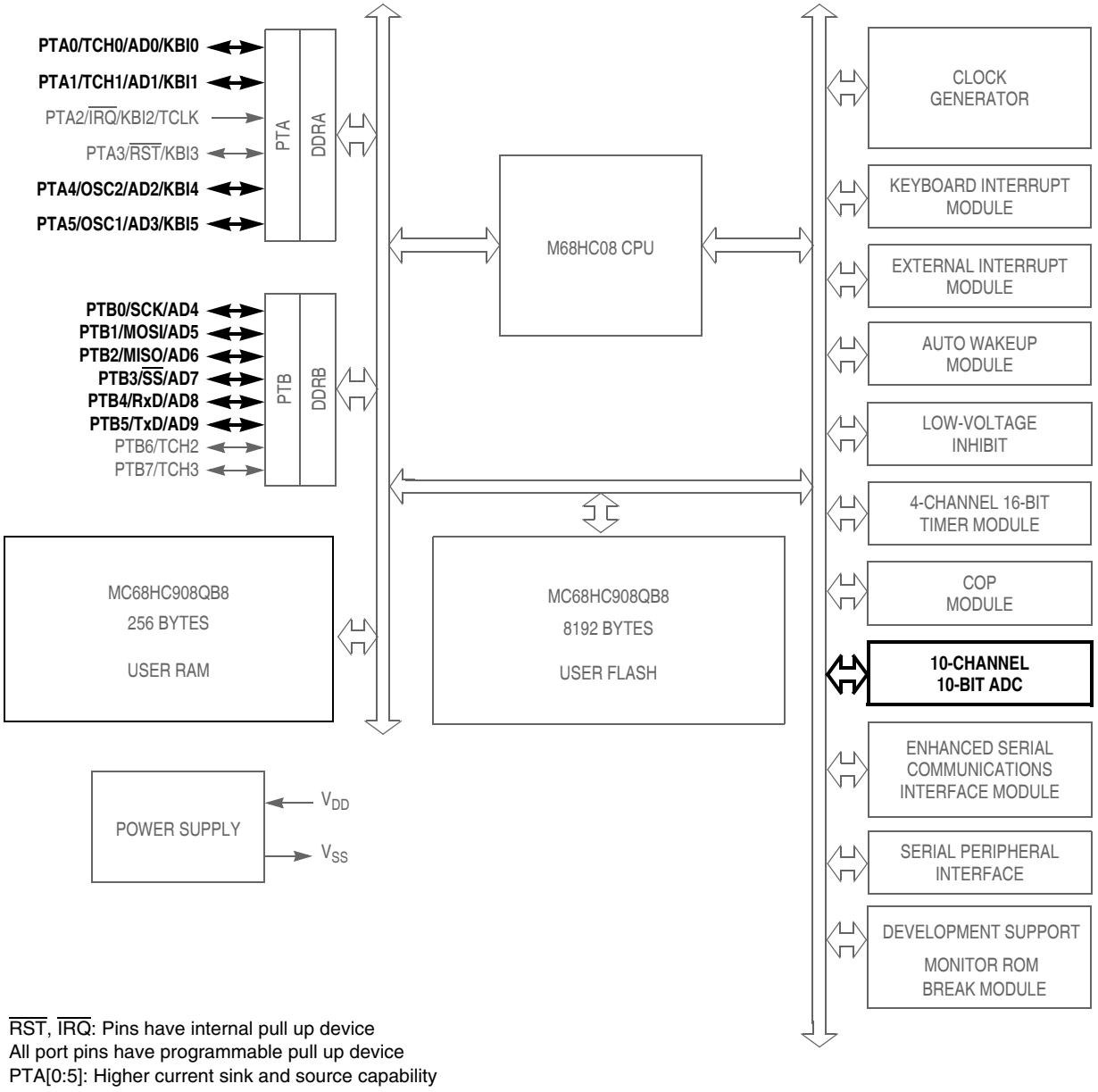
The ADC10 uses successive approximation to convert the input sample taken from ADVIN to a digital representation. The approximation is taken and then rounded to the nearest 10- or 8-bit value to provide greater accuracy and to provide a more robust mechanism for achieving the ideal code-transition voltage.

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.



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device
PTA[0:5]: Higher current sink and source capability

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

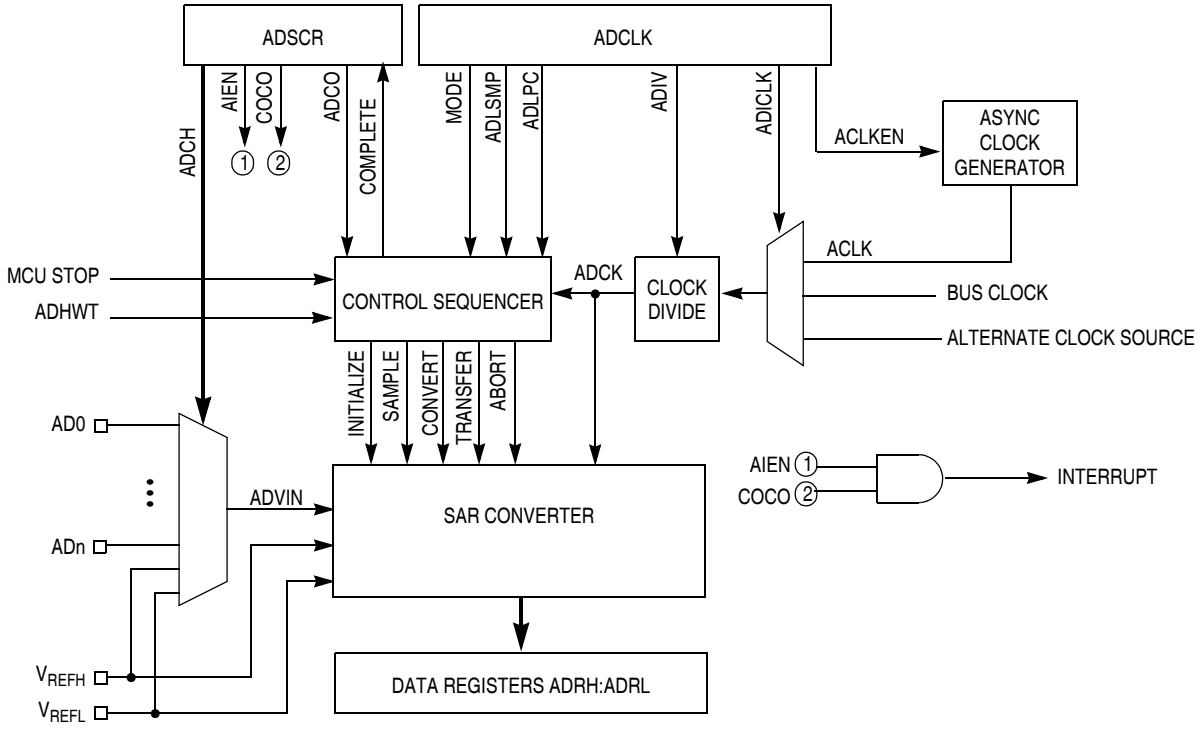


Figure 3-2. ADC10 Block Diagram

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 ADSCR 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 ADSCR (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 ADSCR 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 ADSCR 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 ADSCR).
- 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 ADSCR, 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 ±1/2LSB 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 ADSCR 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, ADSCR
- 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 (ADSCR). Writing ADSCR aborts the current conversion and initiates a new conversion (if the ADCH[4:0] bits are equal to a value other than all 1s).

	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 (ADSCR)

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 ADSCR is written again. If stop is entered

(with ACLKEN low), continuous conversions will cease and can be restarted only with a write to ADSCR. Any write to ADSCR 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 ADSCR is written (assuming the ADCH[4:0] bits do not decode all 1s).

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

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	AD0
0	0	0	0	1	AD1
0	0	0	1	0	AD2
0	0	0	1	1	AD3
0	0	1	0	0	AD4
0	0	1	0	1	AD5
0	0	1	1	0	AD6
0	0	1	1	1	AD7
0	1	0	0	0	Unused
Continuing through					Unused
1	0	1	1	1	Unused
1	1	0	0	0	AD8
1	1	0	0	1	AD9
1	1	0	1	0	BANDGAP REF[2]
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.
2. Requires LVI to be powered (LVIPWRD =0, in CONFIG1)

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.

	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

	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.

	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.

	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 or Hardware Triggered 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, ADSCR software triggered mode enabled
01 = 10-bit, right-justified, ADSCR software triggered mode enabled
10 = Reserved
11 = 10-bit, right-justified, hardware triggered mode enabled

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
Auto Wakeup Module (AWU)

4.1 Introduction

This section describes the auto wakeup module (AWU). The AWU generates a periodic interrupt during stop mode to wake the part up without requiring an external signal. Figure 4-1 is a block diagram of the AWU.

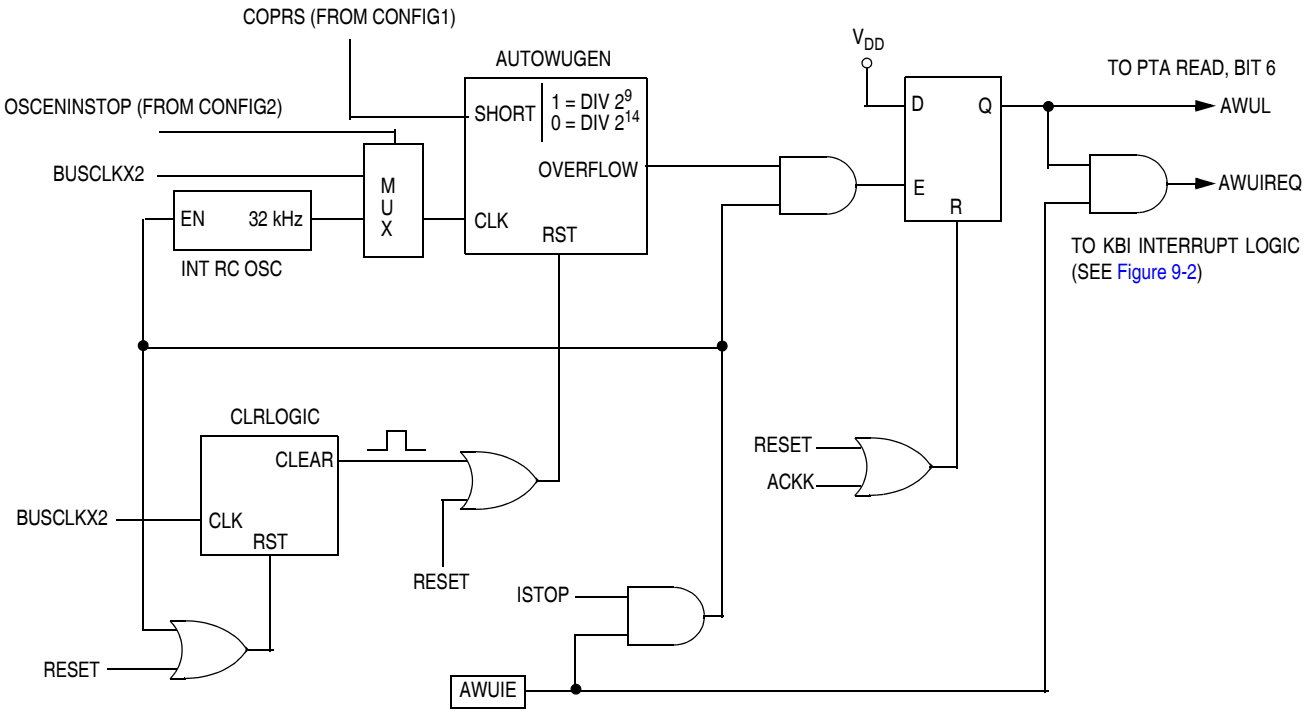


Figure 4-1. Auto Wakeup Interrupt Request Generation Logic

4.2 Features

Features of the auto wakeup module include:

- One internal interrupt with separate interrupt enable bit, sharing the same keyboard interrupt vector and keyboard interrupt mask bit
- Exit from low-power stop mode without external signals
- Selectable timeout periods
- Dedicated low-power internal oscillator separate from the main system clock sources
- Option to allow bus clock source to run the AWU if enabled in STOP

4.3 Functional Description

The function of the auto wakeup logic is to generate periodic wakeup requests to bring the microcontroller unit (MCU) out of stop mode. The wakeup requests are treated as regular keyboard interrupt requests, with the difference that instead of a pin, the interrupt signal is generated by an internal logic.

Entering stop mode will enable the auto wakeup generation logic. Writing the AWUIE bit in the keyboard interrupt enable register enables or disables the auto wakeup interrupt input (see Figure 4-1). A 1 applied to the AWUIREQ input with auto wakeup interrupt request enabled, latches an auto wakeup interrupt request.

Auto wakeup latch, AWUL, can be read directly from the bit 6 position of port A data register (PTA). This is a read-only bit which is occupying an empty bit position on PTA. No PTA associated registers, such as PTA6 data direction or PTA6 pullup exist for this bit.

There are two clock sources for the AWU. An internal RC oscillator (INTRCOSC, exclusive for the auto wakeup feature) drives the wakeup request generator provided the OSCENINSTOP bit in the CONFIG2 register Figure 4-1 is cleared. More accurate wakeup periods are possible using the BUSCLKX2 signal (from the oscillator module) which is selected by setting OSCENINSTOP.

Once the overflow count is reached in the generator counter, a wakeup request, AWUIREQ, is latched and sent to the KBI logic. See Figure 4-1.

Wakeup interrupt requests will only be serviced if the associated interrupt enable bit, AWUIE, in KBIER is set. The AWU shares the keyboard interrupt vector.

The overflow count can be selected from two options defined by the COPRS bit in CONFIG1. This bit was “borrowed” from the computer operating properly (COP) using the fact that the COP feature is idle (no MCU clock available) in stop mode. COPRS = 1 selects the short wakeup period while COPRS = 0 selects the long wakeup period.

The auto wakeup RC oscillator is highly dependent on operating voltage and temperature. This feature is not recommended for use as a time-keeping function.

The wakeup request is latched to allow the interrupt source identification. The latched value, AWUL, can be read directly from the bit 6 position of PTA data register. This is a read-only bit which is occupying an empty bit position on PTA. No PTA associated registers, such as PTA6 data, PTA6 direction, and PTA6 pullup exist for this bit. The latch can be cleared by writing to the ACKK bit in the KBSCR register. Reset also clears the latch. AWUIE bit in KBI interrupt enable register (see Figure 4-1) has no effect on AWUL reading.

The AWU oscillator and counters are inactive in normal operating mode and become active only upon entering stop mode.

4.4 Interrupts

The AWU can generate an interrupt requests:.

AWU Latch (AWUL) — The AWUL bit is set when the AWU counter overflows. The auto wakeup interrupt mask bit, AWUIE, is used to enable or disable AWU interrupt requests.

The AWU shares its interrupt with the KBI vector.

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 AWU module remains inactive in wait mode.

4.5.2 Stop Mode

When the AWU module is enabled (AWUIE = 1 in the keyboard interrupt enable register) it is activated automatically upon entering stop mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of stop mode. The AWU counters start from 0 each time stop mode is entered.

4.6 Registers

The AWU shares registers with the keyboard interrupt (KBI) module, the port A I/O module and configuration register 2. The following I/O registers control and monitor operation of the AWU:

- Port A data register (PTA)
- Keyboard interrupt status and control register (KBSCR)
- Keyboard interrupt enable register (KBIER)
- Configuration register 1 (CONFIG1)
- Configuration register 2 (CONFIG2)

4.6.1 Port A I/O Register

The port A data register (PTA) contains a data latch for the state of the AWU interrupt request, in addition to the data latches for port A.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	AWUL	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
Write:								
Reset:	0	0	Unaffected by reset					

= Unimplemented

Figure 4-2. Port A Data Register (PTA)

AWUL — Auto Wakeup Latch

This is a read-only bit which has the value of the auto wakeup interrupt request latch. The wakeup request signal is generated internally. There is no PTA6 port or any of the associated bits such as PTA6 data direction or pullup bits.

1 = Auto wakeup interrupt request is pending
0 = Auto wakeup interrupt request is not pending

NOTE

PTA5–PTA0 bits are not used in conjuction with the auto wakeup feature. To see a description of these bits, see 12.2.1 Port A Data Register.

4.6.2 Keyboard Status and Control Register

The keyboard status and control register (KBSCR):

- Flags keyboard/auto wakeup interrupt requests
- Acknowledges keyboard/auto wakeup interrupt requests
- Masks keyboard/auto wakeup interrupt requests

	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 4-3. Keyboard Status and Control Register (KBSCR)

Bits 7–4 — Not used

These read-only bits always read as 0s.

KEYF — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending on port A or auto wakeup. Reset clears the KEYF bit.

1 = Keyboard/auto wakeup interrupt pending
0 = No keyboard/auto wakeup interrupt pending

ACKK — Keyboard Acknowledge Bit

Writing a 1 to this write-only bit clears the keyboard/auto wakeup interrupt request on port A and auto wakeup logic. ACKK always reads as 0. Reset clears ACKK.

IMASKK— Keyboard Interrupt Mask Bit

Writing a 1 to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests on port A or auto wakeup. Reset clears the IMASKK bit.

1 = Keyboard/auto wakeup interrupt requests masked
0 = Keyboard/auto wakeup interrupt requests not masked

NOTE

MODEK is not used in conjuction with the auto wakeup feature. To see a description of this bit, see 9.8.1 Keyboard Status and Control Register (KBSCR).

4.6.3 Keyboard Interrupt Enable Register

The keyboard interrupt enable register (KBIER) enables or disables the auto wakeup to operate as a keyboard/auto wakeup interrupt input.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	AWUIE	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

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

AWUIE — Auto Wakeup Interrupt Enable Bit

This read/write bit enables the auto wakeup interrupt input to latch interrupt requests. Reset clears AWUIE.

1 = Auto wakeup enabled as interrupt input
0 = Auto wakeup not enabled as interrupt input

NOTE

KBIE5–KBIE0 bits are not used in conjuction with the auto wakeup feature. To see a description of these bits, see 9.8.2 Keyboard Interrupt Enable Register (KBIER).

4.6.4 Configuration Register 2

The configuration register 2 (CONFIG2), is used to allow the bus clock source to run in STOP. In this case, the clock, BUSCLKX2 will be used to drive the AWU request generator.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	IRQPUD	IRQEN	R	R	R	ESCI-BDSRC	OSCENINSTOP	RSTEN
Reset:	0	0	0	0	0	0	0	0

Figure 4-5. Configuration Register 2 (CONFIG2)

OSCENINSTOP — Oscillator Enable in Stop Mode Bit

OSCENINSTOP, when set, will allow the bus clock source (BUSCLKX2) to generate clocks for the AWU in stop mode. See *11.8.1 Oscillator Status and Control Register* for information on enabling the external clock sources.

1 = Oscillator enabled to operate during stop mode
0 = Oscillator disabled during stop mode

NOTE

IRQPUD, IRQEN, ESCIBDSRC, and RSTEN bits are not used in conjuction with the auto wakeup feature. To see a description of these bits, see Chapter 5 Configuration Register (CONFIG).

4.6.5 Configuration Register 1

The configuration register 1 (CONFIG1), is used to select the period for the AWU. The timeout will be based on the COPRS bit along with the clock source for the AWU.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVITRIP	SSREC	STOP	COPD
Reset: POR:	0 0	0 0	0 0	0 0	U 0	0 0	0 0	0 0

U = Unaffected

Figure 4-6. Configuration Register 1 (CONFIG1)

COPRS (In Stop Mode) — Auto Wakeup Period Selection Bit, depends on OSCSTOPEN in CONFIG2 and bus clock source (BUSCLKX2).

1 = Auto wakeup short cycle = 512 × (INTRCOSC or BUSCLKX2)
0 = Auto wakeup long cycle = 16,384 × (INTRCOSC or BUSCLKX2)

SSREC — Short Stop Recovery Bit

SSREC enables the CPU to exit stop mode with a delay of 32 BUSCLKX4 cycles instead of a 4096 BUSCLKX4 cycle delay.

1 = Stop mode recovery after 32 BUSCLKX4 cycles
0 = Stop mode recovery after 4096 BUSCLKX4 cycles

NOTE

LVISTOP, LVIRST, LVIPWRD, LVITRIP and COPD bits are not used in conjuction with the auto wakeup feature. To see a description of these bits, see Chapter 5 Configuration Register (CONFIG)

Chapter 5
Configuration Register (CONFIG)

5.1 Introduction

This section describes the configuration registers (CONFIG1 and CONFIG2). The configuration registers enable or disable the following options:

- Stop mode recovery time (32 × BUSCLKX4 cycles or 4096 × BUSCLKX4 cycles)
- STOP instruction
- Computer operating properly module (COP)
- COP reset period (COPRS): 8176 × BUSCLKX4 or 262,128 × BUSCLKX4
- Low-voltage inhibit (LVI) enable and trip voltage selection
- Auto wakeup timeout period
- Allow clock source to remain enabled in STOP
- Enable $\overline{\text{IRQ}}$ pin
- Disable $\overline{\text{IRQ}}$ pin pullup device
- Enable $\overline{\text{RST}}$ pin
- Clock source selection for the enhanced serial communication interface (ESCI) module

5.2 Functional Description

The configuration registers are used in the initialization of various options. The configuration registers can be written once after each reset. Most 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 this register be written immediately after reset. The configuration registers are located at $001E and $001F, and may be read at anytime.

NOTE

The CONFIG registers are one-time writable by the user after each reset. Upon a reset, the CONFIG registers default to predetermined settings as shown in Figure 5-1 and Figure 5-2.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	IRQPUD	IRQEN	R	R	R	ESCIBDSRC	OSCENINSTOP	RSTEN
Reset:	0	0	0	0	0	0	0	U
POR:	0	0	0	0	0	0	0	0

R = Reserved U = Unaffected

Figure 5-1. Configuration Register 2 (CONFIG2)

IRQPUD — $\overline{\text{IRQ}}$ Pin Pullup Control Bit

1 = Internal pullup is disconnected
0 = Internal pullup is connected between $\overline{\text{IRQ}}$ pin and V_{DD}

IRQEN — $\overline{\text{IRQ}}$ Pin Function Selection Bit

1 = Interrupt request function active in pin
0 = Interrupt request function inactive in pin

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 = BUSCLKX4 used as clock source for ESCI

OSCENINSTOP— Oscillator Enable in Stop Mode Bit

OSCENINSTOP, when set, will allow the clock source to continue to generate clocks in stop mode. This function can be used to keep the auto-wakeup running while the rest of the microcontroller stops. When clear, the clock source is disabled when the microcontroller enters stop mode.

1 = Oscillator enabled to operate during stop mode
0 = Oscillator disabled during stop mode

RSTEN — $\overline{\text{RST}}$ Pin Function Selection

1 = Reset function active in pin
0 = Reset function inactive in pin

NOTE

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

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVITRIP	SSREC	STOP	COPD
Reset:	0	0	0	0	U	0	0	0
POR:	0	0	0	0	0	0	0	0

U = Unaffected

Figure 5-2. Configuration Register 1 (CONFIG1)

COPRS (Out of Stop Mode) — COP Reset Period Selection Bit

1 = COP reset short cycle = 8176 × BUSCLKX4
0 = COP reset long cycle = 262,128 × BUSCLKX4

COPRS (In Stop Mode) — Auto Wakeup Period Selection Bit, depends on OSCSTOPEN in CONFIG2 and external clock source

1 = Auto wakeup short cycle = 512 × (INTRCOSC or BUSCLKX4)
0 = Auto wakeup long cycle = 16,384 × (INTRCOSC or BUSCLKX4)

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.

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

LVIPWRD — LVI Power Disable Bit

LVIPWRD disables the LVI module.

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

LVITRIP — LVI Trip Point Selection Bit

LVITRIP selects the voltage operating mode of the LVI module. The voltage mode selected for the LVI should match the operating V_{DD} for the LVI's voltage trip points for each of the modes.

1 = LVI operates for a 5-V protection
0 = LVI operates for a 3-V protection

NOTE

The LVITRIP 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 BUSCLKX4 cycles instead of a 4096 BUSCLKX4 cycle delay.

1 = Stop mode recovery after 32 BUSCLKX4 cycles
0 = Stop mode recovery after 4096 BUSCLKX4 cycles

NOTE

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

When using the LVI during normal operation but disabling during stop mode, the LVI will have an enable time of t_{EN}. The system stabilization time for power-on reset and long stop recovery (both 4096 BUSCLKX4 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 BUSCLKX4 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.

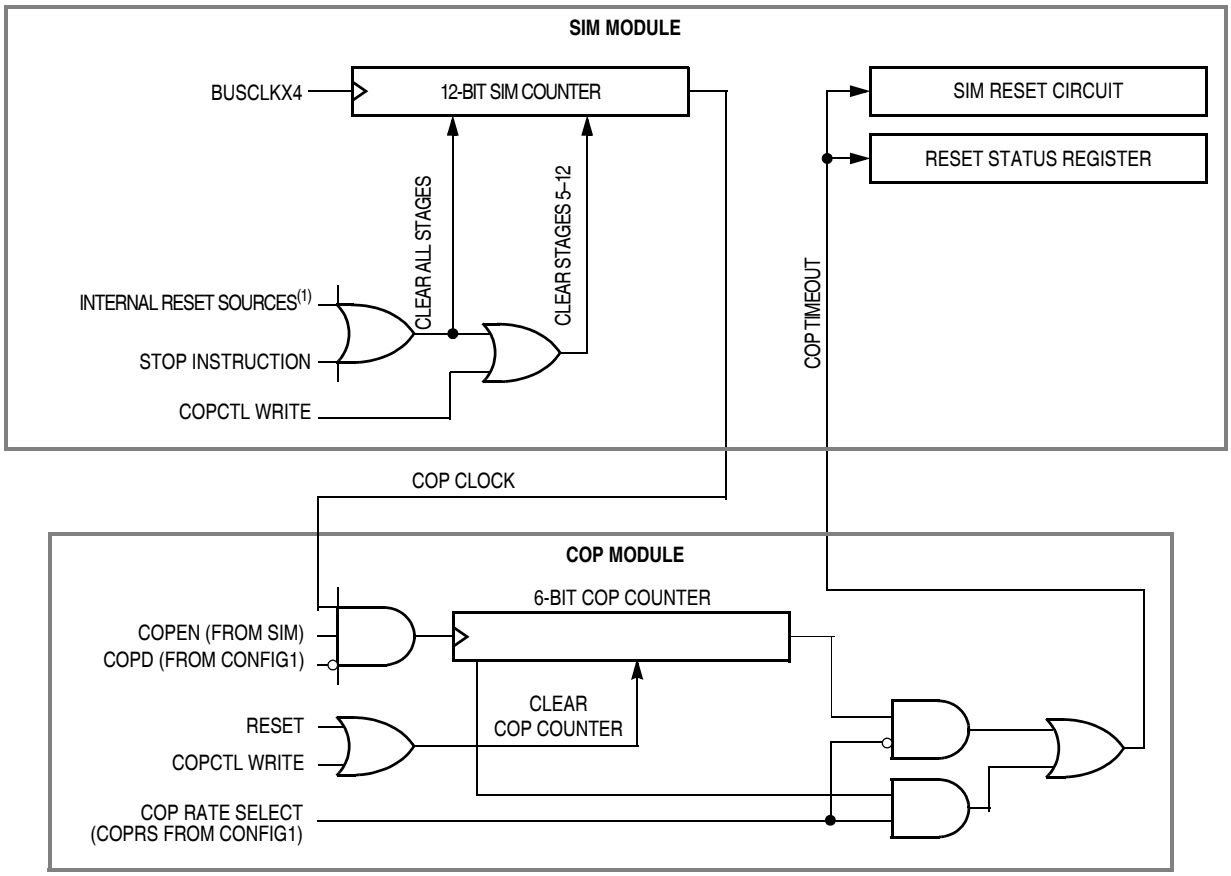
1 = COP module disabled
0 = COP module enabled

Chapter 6
Computer Operating Properly (COP)

6.1 Introduction

The computer operating properly (COP) module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by clearing the COP counter periodically. The COP module can be disabled through the COPD bit in the configuration 1 (CONFIG1) register.

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 the 12-bit system integration module (SIM) counter. If not cleared by software, the COP counter overflows and generates an asynchronous reset after 8176 or 262,128 BUSCLKX4 cycles; depending on the state of the COP rate select bit, COPRS, in configuration register 1. With a 262,128 BUSCLKX4 cycle overflow option, the internal 12.8-MHz oscillator gives a COP timeout period of 20.48 ms. Writing any value to location $FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 12–5 of the SIM counter.

NOTE

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

A COP reset pulls the $\overline{\text{RST}}$ pin low (if the RSTEN bit is set in the CONFIG1 register) for 32 × BUSCLKX4 cycles and sets the COP bit in the reset status register (RSR). See 14.8.1 SIM Reset Status Register.

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 BUSCLKX4

BUSCLKX4 is the oscillator output signal. BUSCLKX4 frequency is equal to the internal oscillator frequency, the crystal frequency, or the RC-oscillator frequency.

6.3.2 STOP Instruction

The STOP instruction clears the SIM counter.

6.3.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see Figure 6-2) clears the COP counter and clears stages 12–5 of the SIM counter. Reading the COP control register returns the low byte of the reset vector.

6.3.4 Power-On Reset

The power-on reset (POR) circuit in the SIM clears the SIM counter 4096 × BUSCLKX4 cycles after power up.

6.3.5 Internal Reset

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

6.3.6 COPD (COP Disable)

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

6.3.7 COPRS (COP Rate Select)

The COPRS signal reflects the state of the COP rate select bit (COPRS) in the configuration register 1 (CONFIG1). See Chapter 5 Configuration Register (CONFIG).

6.4 Interrupts

The COP does not generate CPU interrupt requests.

6.5 Monitor Mode

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

6.6 Low-Power Modes

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

6.6.1 Wait Mode

The COP continues to operate during wait mode. To prevent a COP reset during wait mode, periodically clear the COP counter.

6.6.2 Stop Mode

Stop mode turns off the BUSCLKX4 input to the COP and clears the SIM counter. Service the COP immediately before entering or after exiting stop mode to ensure a full COP timeout period after entering or exiting stop mode.

6.7 COP Module During Break Mode

The COP is disabled during a break interrupt with monitor mode when BDCOP bit is set in break auxiliary register (BRKAR).

6.8 Register

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

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LOW BYTE OF RESET VECTOR							
Write:	CLEAR COP COUNTER							
Reset:	Unaffected by reset							

Figure 6-2. COP Control Register (COPCTL)

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.

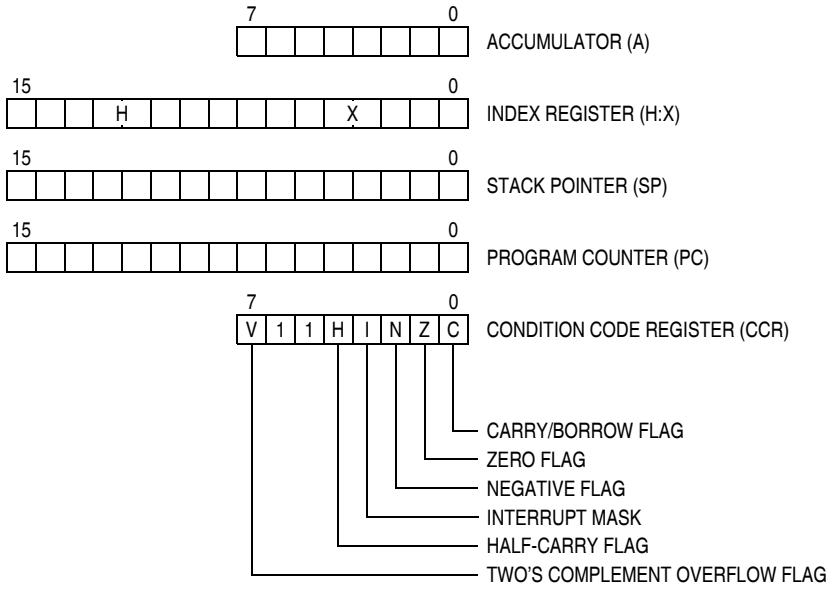


Figure 7-1. CPU Registers

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.

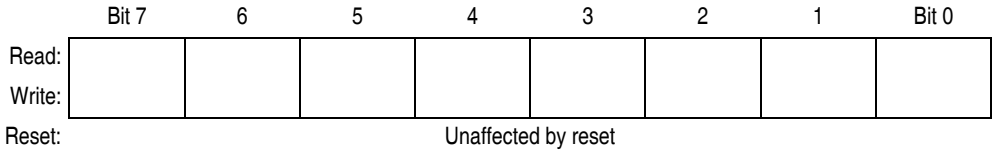


Figure 7-2. Accumulator (A)

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.

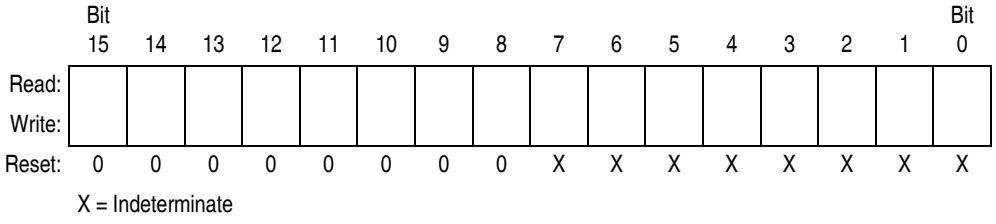


Figure 7-3. Index Register (H:X)

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.

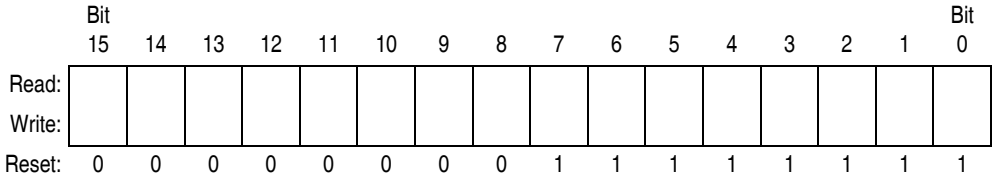


Figure 7-4. Stack Pointer (SP)

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.

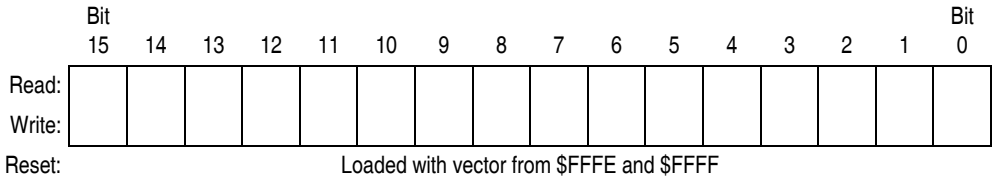


Figure 7-5. Program Counter (PC)

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: 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)

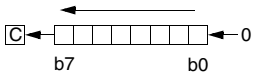
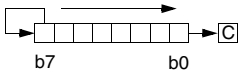
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X ADC *opr*,SP ADC *opr*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9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

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

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BHS *rel*	Branch if Higher or Same (Same as BCC)	PC ← (PC) + 2 + *rel* ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if $\overline{\text{IRQ}}$ Pin High	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if $\overline{\text{IRQ}}$ Pin Low	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #*opr* BIT *opr* BIT *opr* BIT *opr*,X BIT *opr*,X BIT ,X BIT *opr*,SP BIT *opr*,SP	Bit Test	(A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A5 B5 C5 D5 E5 F5 9EE5 9ED5	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 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) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 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) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n*,*opr*	Set Bit *n* in M	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	4
CBEQ *opr*,*rel* CBEQA #*opr*,*rel* CBEQX #*opr*,*rel* CBEQ *opr*,X+,*rel* CBEQ X+,*rel* CBEQ *opr*,SP,*rel*	Compare and Branch if Equal	PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (X) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 2 + rel ? (A) – (M) = $00 PC ← (PC) + 4 + rel ? (A) – (M) = $00	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 4 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	I ← 0	–	–	0	–	–	–	INH	9A		2

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

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

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

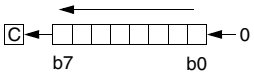
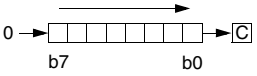
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR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

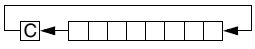
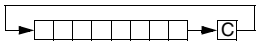
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–	–	INH	88		2
ROL *opr* ROLA ROLX ROL *opr*,X ROL ,X ROL *opr*,SP	Rotate Left through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	4 1 1 4 3 5
ROR *opr* RORA RORX ROR *opr*,X ROR ,X ROR *opr*,SP	Rotate Right through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	4 1 1 4 3 5
RSP	Reset Stack Pointer	SP ← $FF	–	–	–	–	–	–	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 1; Pull (CCR) SP ← (SP) + 1; Pull (A) SP ← (SP) + 1; Pull (X) SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	↕	↕	↕	↕	↕	↕	INH	80		7
RTS	Return from Subroutine	SP ← SP + 1; Pull (PCH) SP ← SP + 1; Pull (PCL)	–	–	–	–	–	–	INH	81		4
SBC #*opr* SBC *opr* SBC *opr* SBC *opr*,X SBC *opr*,X SBC ,X SBC *opr*,SP SBC *opr*,SP	Subtract with Carry	A ← (A) – (M) – (C)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A2 B2 C2 D2 E2 F2 9EE2 9ED2	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
SEC	Set Carry Bit	C ← 1	–	–	–	–	–	1	INH	99		1
SEI	Set Interrupt Mask	I ← 1	–	–	1	–	–	–	INH	9B		2
STA *opr* STA *opr* STA *opr*,X STA *opr*,X STA ,X STA *opr*,SP STA *opr*,SP	Store A in M	M ← (A)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	B7 C7 D7 E7 F7 9EE7 9ED7	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 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* STX *opr* STX *opr*,X STX *opr*,X STX ,X STX *opr*,SP STX *opr*,SP	Store X in M	M ← (X)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	BF CF DF EF FF 9EEF 9EDF	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 5
SUB #*opr* SUB *opr* SUB *opr* SUB *opr*,X SUB *opr*,X SUB ,X SUB *opr*,SP SUB *opr*,SP	Subtract	A ← (A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A0 B0 C0 D0 E0 F0 9EE0 9ED0	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 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) SP ← (SP) – 1; Push (PCH) SP ← (SP) – 1; Push (X) SP ← (SP) – 1; Push (A) SP ← (SP) – 1; Push (CCR) SP ← (SP) – 1; I ← 1 PCH ← Interrupt Vector High Byte 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* TSTA TSTX TST *opr*,X TST ,X TST *opr*,SP	Test for Negative or Zero	(A) – $00 or (X) – $00 or (M) – $00	0	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	3 1 1 3 2 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/Memory							
	DIR	DIR	REL	DIR	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX
MSB / LSB	0	1	2	3	4	5	6	9E6	7	8	9	A	B	C	D	9ED	E	9EE	F
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	5 SUB 4 SP2	3 SUB 2 IX1	4 SUB 3 SP1	2 SUB 1 IX
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	5 CMP 4 SP2	3 CMP 2 IX1	4 CMP 3 SP1	2 CMP 1 IX
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	5 SBC 4 SP2	3 SBC 2 IX1	4 SBC 3 SP1	2 SBC 1 IX
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	5 CPX 4 SP2	3 CPX 2 IX1	4 CPX 3 SP1	2 CPX 1 IX
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	5 AND 4 SP2	3 AND 2 IX1	4 AND 3 SP1	2 AND 1 IX
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	5 BIT 4 SP2	3 BIT 2 IX1	4 BIT 3 SP1	2 BIT 1 IX
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	5 LDA 4 SP2	3 LDA 2 IX1	4 LDA 3 SP1	2 LDA 1 IX
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	5 STA 4 SP2	3 STA 2 IX1	4 STA 3 SP1	2 STA 1 IX
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	5 EOR 4 SP2	3 EOR 2 IX1	4 EOR 3 SP1	2 EOR 1 IX
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	5 ADC 4 SP2	3 ADC 2 IX1	4 ADC 3 SP1	2 ADC 1 IX
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	5 ORA 4 SP2	3 ORA 2 IX1	4 ORA 3 SP1	2 ORA 1 IX
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	5 ADD 4 SP2	3 ADD 2 IX1	4 ADD 3 SP1	2 ADD 1 IX
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		3 JMP 2 IX1		2 JMP 1 IX
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		5 JSR 2 IX1		4 JSR 1 IX
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	5 LDX 4 SP2	3 LDX 2 IX1	4 LDX 3 SP1	2 LDX 1 IX
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	5 STX 4 SP2	3 STX 2 IX1	4 STX 3 SP1	2 STX 1 IX

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

MSB / LSB	0
0	5 BRSET0 3 DIR

Low Byte of Opcode in Hexadecimal

High Byte of Opcode in Hexadecimal

Cycles
Opcode Mnemonic
Number of Bytes / Addressing Mode

Chapter 8
External Interrupt (IRQ)

8.1 Introduction

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

$\overline{\text{IRQ}}$ functionality is enabled by setting configuration register 2 (CONFIG2) IRQEN bit accordingly. A zero disables the IRQ function and $\overline{\text{IRQ}}$ will assume the other shared functionalities. A one enables the IRQ function. See Chapter 5 Configuration Register (CONFIG) for more information on enabling the IRQ pin.

The IRQ pin shares its pin with general-purpose input/output (I/O) port pins. See Figure 8-1 for port location of this shared pin.

8.2 Features

Features of the IRQ module include:

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

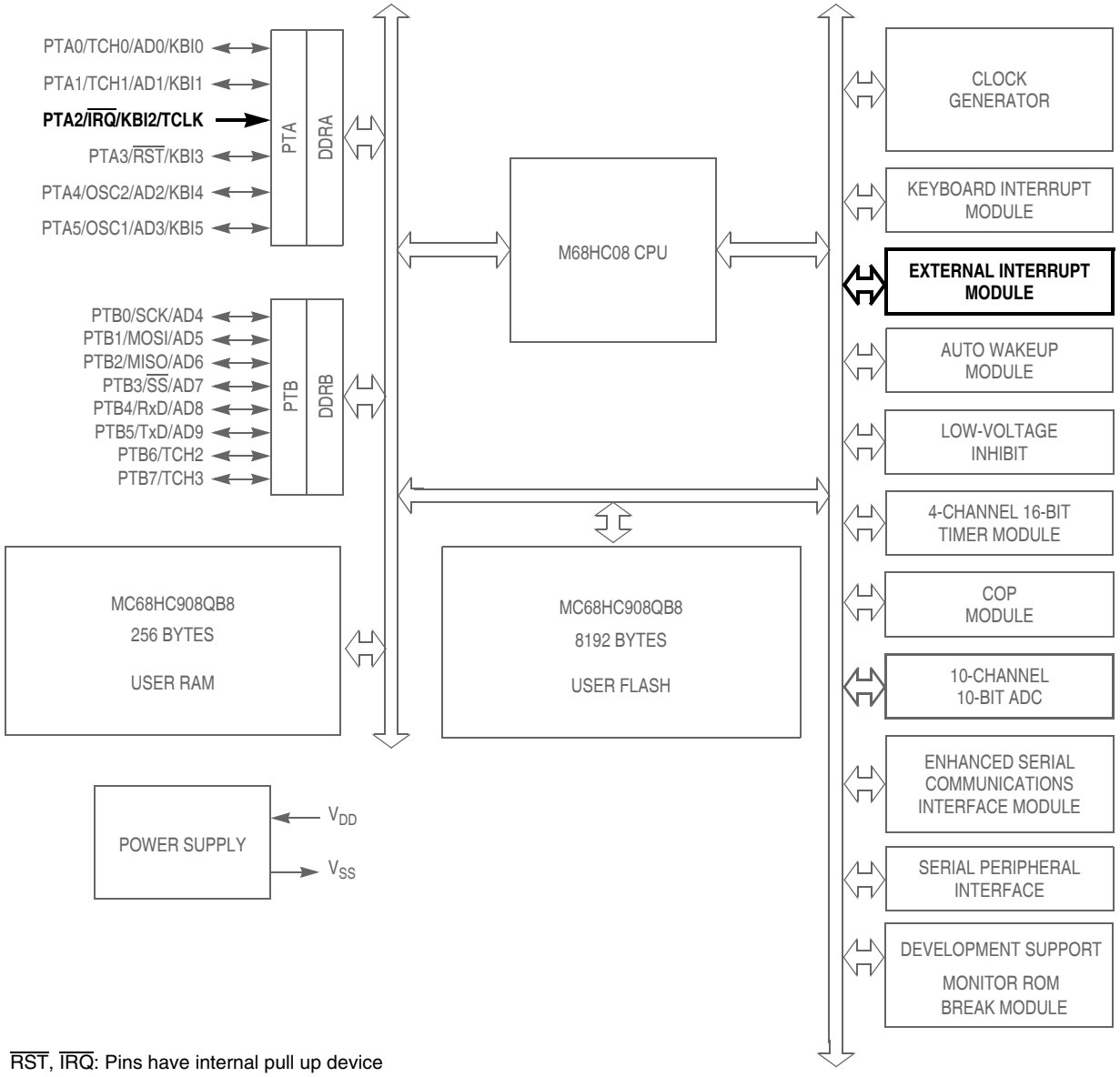
8.3 Functional Description

A low level applied to the external interrupt request ($\overline{\text{IRQ}}$) pin can latch a CPU interrupt request. Figure 8-2 shows the structure of the IRQ module.

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

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

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



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device
PTA[0:5]: Higher current sink and source capability

Figure 8-1. Block Diagram Highlighting IRQ Block and Pin

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

NOTE

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

A falling edge on the IRQ pin can latch an interrupt request into the IRQ latch. An IRQ vector fetch, software clear, or reset clears the IRQ latch.

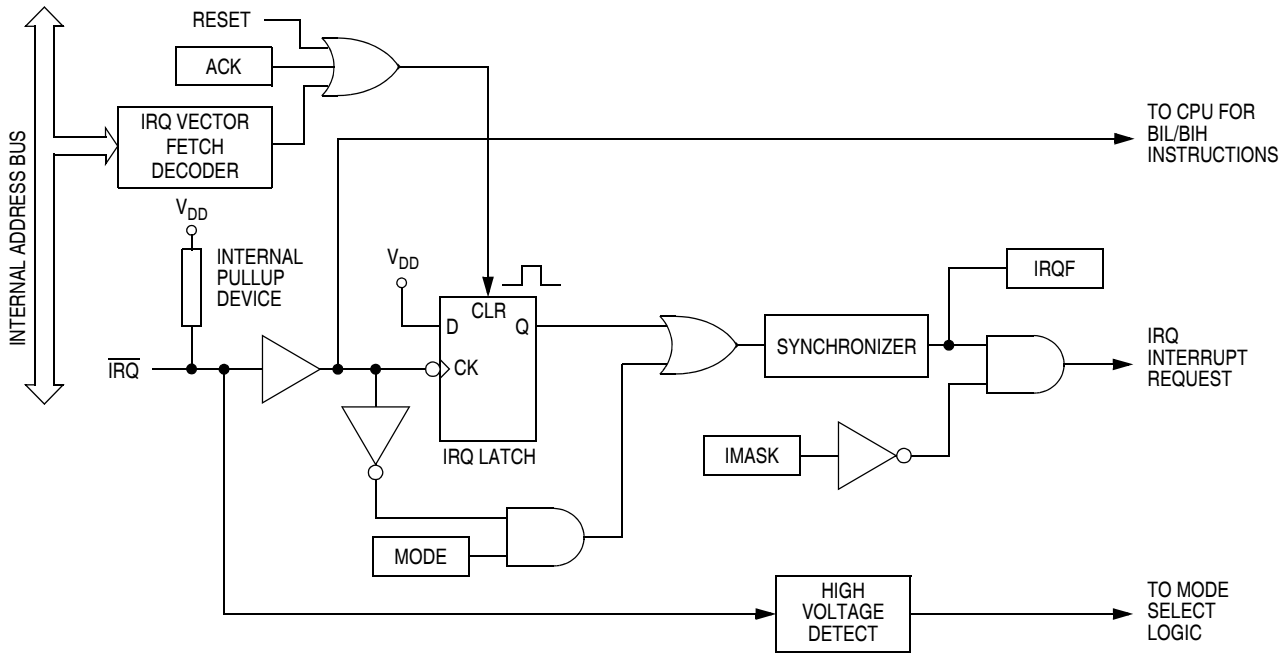


Figure 8-2. IRQ Module Block Diagram

8.3.1 MODE = 1

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

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

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

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

8.3.2 MODE = 0

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

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

NOTE

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

8.4 Interrupts

The following IRQ source can generate interrupt requests:

- Interrupt flag (IRQF) — The IRQF bit is set when the $\overline{IRQ}$ pin is asserted based on the IRQ mode.. The IRQ interrupt mask bit, IMASK, is used to enable or disable IRQ interrupt requests.

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 IRQ module remains active in wait mode. Clearing IMASK in INTSCR enables IRQ interrupt requests to bring the MCU out of wait mode.

8.5.2 Stop Mode

The IRQ module remains active in stop mode. Clearing IMASK in INTSCR enables IRQ interrupt requests to bring the MCU out of stop mode.

8.6 IRQ Module 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.

8.7 I/O Signals

The IRQ module does not share its pin with any module on this MCU.

8.7.1 IRQ Input Pins ($\overline{IRQ}$)

The $\overline{IRQ}$ pin provides a maskable external interrupt source. The $\overline{IRQ}$ pin contains an internal pullup device.

8.8 Registers

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

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

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

= Unimplemented

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

IRQF — IRQ Flag Bit

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

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

ACK — IRQ Interrupt Request Acknowledge Bit

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

IMASK — IRQ Interrupt Mask Bit

Writing a 1 to this read/write bit disables the IRQ interrupt request.

1 = IRQ interrupt request disabled
0 = IRQ interrupt request enabled

MODE — IRQ Edge/Level Select Bit

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

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

Chapter 9
Keyboard Interrupt Module (KBI)

9.1 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 9-1 for port location of these shared pins.

9.2 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

9.3 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. See Figure 9-2.

9.3.1 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.

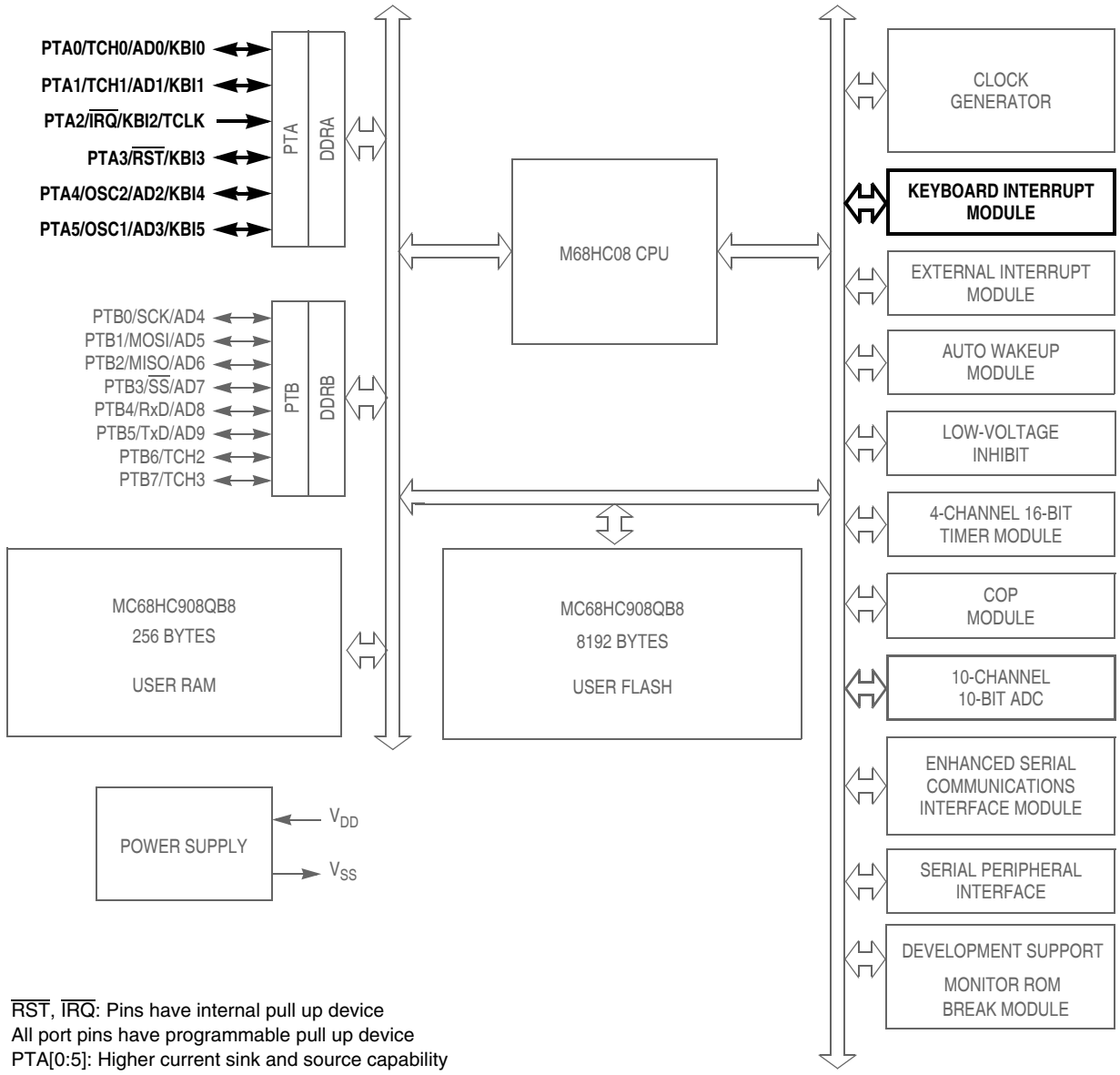


Figure 9-1. Block Diagram Highlighting KBI Block and Pins

9.3.1.1 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.

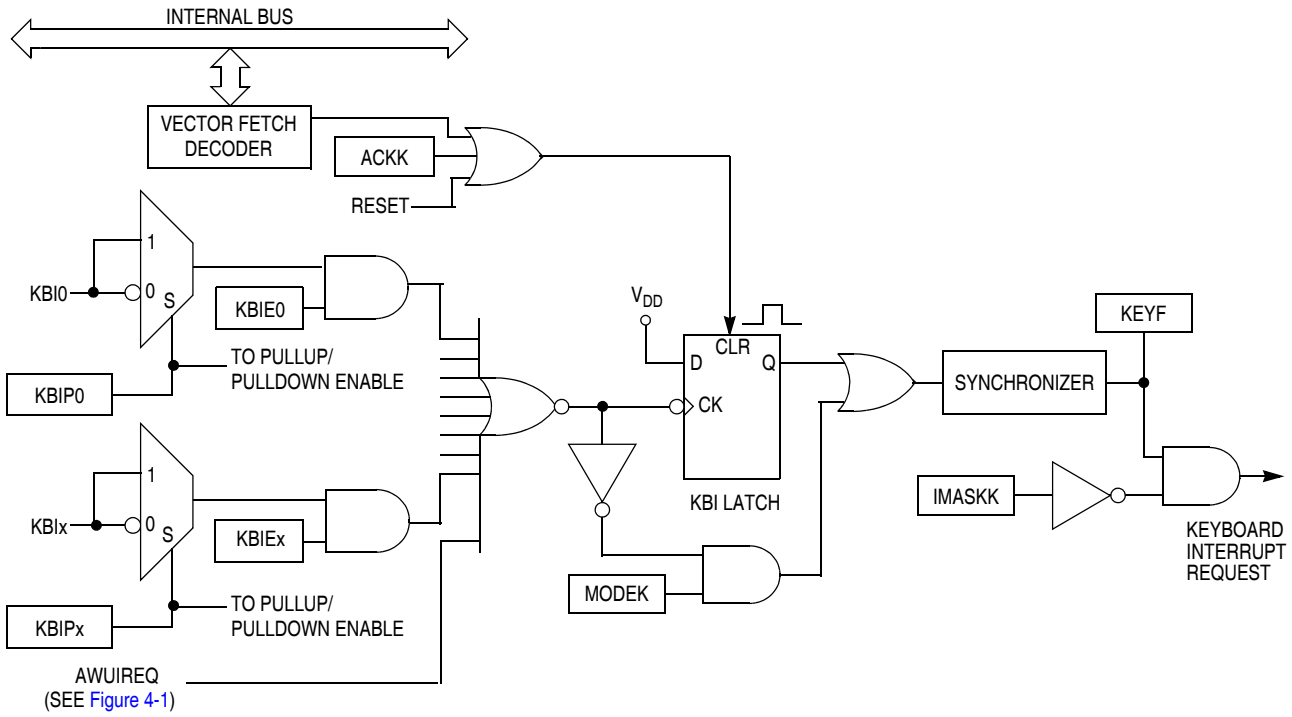


Figure 9-2. Keyboard Interrupt Block Diagram

- 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.

9.3.1.2 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.

9.3.2 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.

9.4 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.

9.5 Low-Power Modes

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

9.5.1 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.

9.5.2 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.

9.6 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.

9.7 I/O Signals

The KBI module can share its pins with the general-purpose I/O pins. See Figure 9-1 for the port pins that are shared.

9.7.1 KBI Input Pins (KBIx: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.

9.8 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)

9.8.1 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

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

= Unimplemented

Figure 9-3. 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

9.8.2 Keyboard Interrupt Enable Register (KBIER)

KBIER enables or disables each keyboard interrupt pin.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	AWUIE	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 9-4. Keyboard Interrupt Enable Register (KBIER)

KBIE5–KBIE0 — Keyboard Interrupt Enable Bits

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

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

NOTE

AWUIE bit is not used in conjunction with the keyboard interrupt feature. To see a description of this bit, see Chapter 4 Auto Wakeup Module (AWU)

9.8.3 Keyboard Interrupt Polarity Register (KBIPR)

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

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	KBIP5	KBIP4	KBIP3	KBIP2	KBIP1	KBIP0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 9-5. Keyboard Interrupt Polarity Register (KBIPR)

KBIP5–KBIP0 — Keyboard Interrupt Polarity Bits

Each of these read/write bits enables the polarity of the keyboard interrupt detection.

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

Chapter 10
Low-Voltage Inhibit (LVI)

10.1 Introduction

The low-voltage inhibit (LVI) module is provided as a system protection mechanism to prevent the MCU from operating below a certain operating supply voltage level. The module has several configuration options to allow functionality to be tailored to different system level demands.

The configuration registers (see Chapter 5 Configuration Register (CONFIG)) contain control bits for this module.

10.2 Features

Features of the LVI module include:

- Programmable LVI reset
- Selectable LVI trip voltage
- Programmable stop mode operation

10.3 Functional Description

Figure 10-1 shows the structure of the LVI module. LVISTOP, LVIPWRD, LVITRIP, and LVIRSTD are user selectable options found in the configuration register.

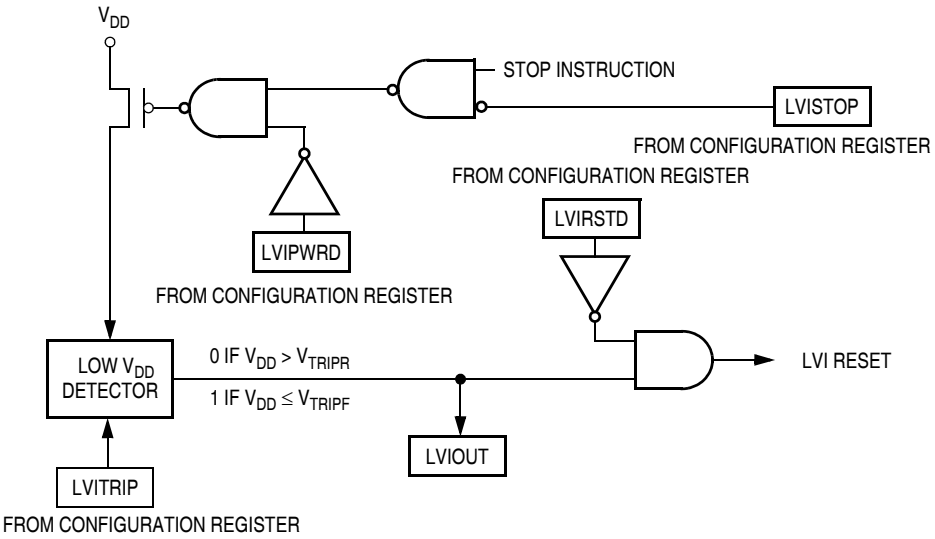


Figure 10-1. LVI Module Block Diagram

The LVI module contains a bandgap reference circuit and comparator. When the LVITRIP bit is cleared, the default state at power-on reset, V_{TRIPF} is configured for the lower V_{DD} operating range. The actual trip points are specified in 18.5 5-V DC Electrical Characteristics and 18.8 3-V DC Electrical Characteristics.

Because the default LVI trip point after power-on reset is configured for low voltage operation, a system requiring high voltage LVI operation must set the LVITRIP bit during system initialization. V_{DD} must be above the LVI trip rising voltage, V_{TRIPR}, for the high voltage operating range or the MCU will immediately go into LVI reset.

After an LVI reset occurs, the MCU remains in reset until V_{DD} rises above V_{TRIPR}. See Chapter 14 System Integration Module (SIM) for the reset recovery sequence.

The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR) and can be used for polling LVI operation when the LVI reset is disabled.

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 operate in stop mode.
- Setting the LVI trip point bit, LVITRIP, configures the trip point voltage (V_{TRIPF}) for the higher V_{DD} operating range.

10.3.1 Polled LVI Operation

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

10.3.2 Forced Reset Operation

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

10.3.3 LVI Hysteresis

The LVI has hysteresis to maintain a stable operating condition. After the LVI has triggered (by having V_{DD} fall below V_{TRIPF}), the MCU will remain in reset until V_{DD} rises above the rising trip point voltage, V_{TRIPR}. This prevents a condition in which the MCU is continually entering and exiting reset if V_{DD} is approximately equal to V_{TRIPF}. V_{TRIPR} is greater than V_{TRIPF} by the typical hysteresis voltage, V_{HYS}.

10.3.4 LVI Trip Selection

LVITRIP in the configuration register selects the LVI protection range. The default setting out of reset is for the low voltage range. Because LVITRIP is in a write-once configuration register, the protection range cannot be changed after initialization.

NOTE

The MCU is guaranteed to operate at a minimum supply voltage. The trip point (V_{TRIPF}) may be lower than this. See the Electrical Characteristics section for the actual trip point voltages.

10.4 LVI Interrupts

The LVI module does not generate interrupt requests.

10.5 Low-Power Modes

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

10.5.1 Wait Mode

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

10.5.2 Stop Mode

If the LVIPWRD bit in the configuration register is cleared and the LVISTOP bit in the configuration register is set, the LVI module remains active. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of stop mode.

10.6 Registers

The LVI status register (LVISR) contains a status bit that is useful when the LVI is enabled and LVI reset is disabled.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LVIOUT	0	0	0	0	0	0	R
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 10-2. LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

This read-only flag becomes set when the V_{DD} voltage falls below the V_{TRIPF} trip voltage and is cleared when V_{DD} voltage rises above V_{TRIPR}. (See Table 10-1).

Table 10-1. LVIOUT Bit Indication

V_{DD}	LVIOUT
$V_{DD} > V_{TRIPR}$	0
$V_{DD} < V_{TRIPF}$	1
$V_{TRIPF} < V_{DD} < V_{TRIPR}$	Previous value

Chapter 11
Oscillator Module (OSC)

11.1 Introduction

The oscillator (OSC) module is used to provide a stable clock source for the MCU system and bus.

The OSC shares its pins with general-purpose input/output (I/O) port pins. See Figure 11-1 for port location of these shared pins. The OSC2EN bit is located in the port A pull enable register (PTAPUEN) on this MCU. See Chapter 12 Input/Output Ports (PORTS) for information on PTAPUEN register.

11.2 Features

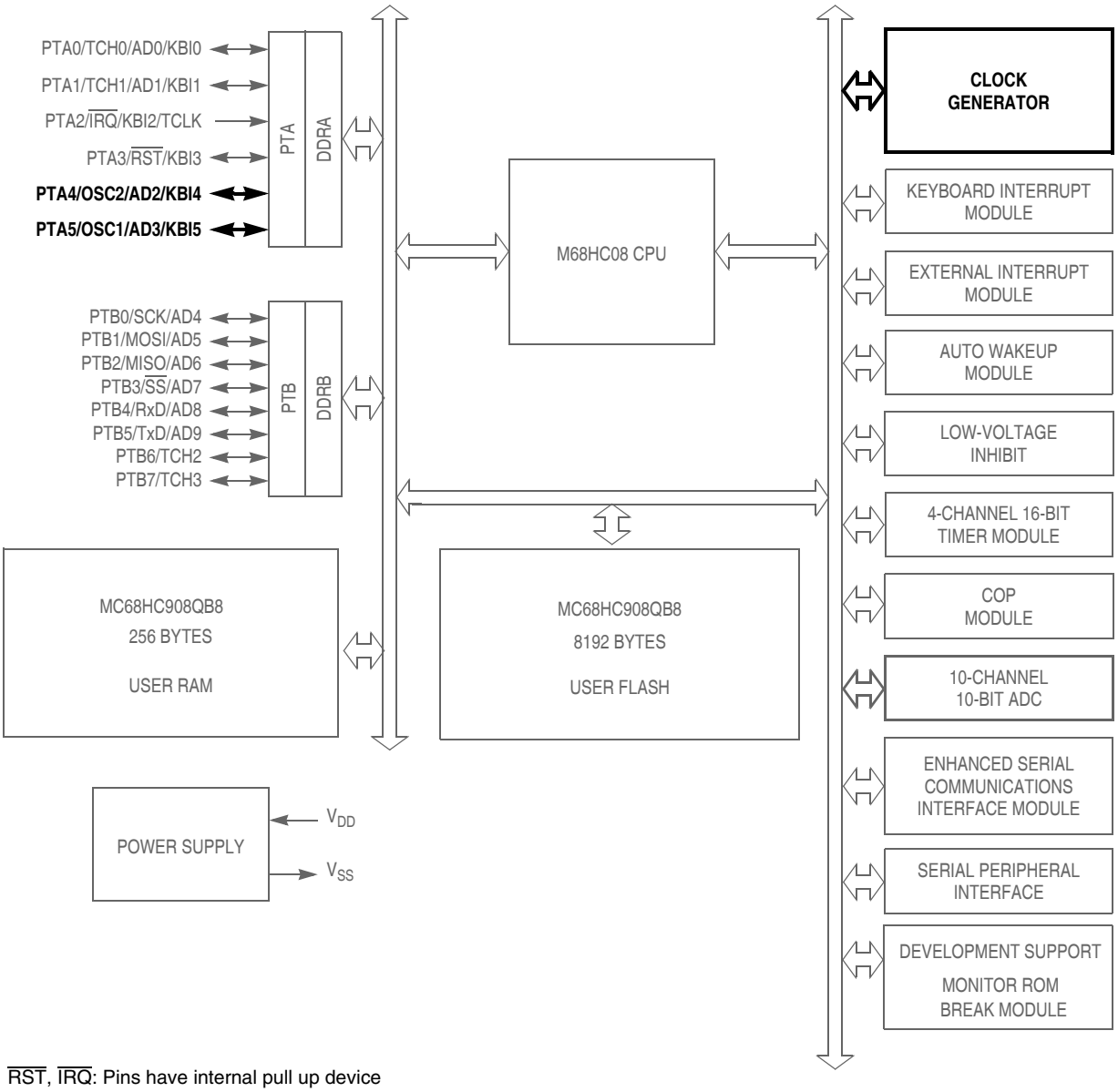
The bus clock frequency is one fourth of any of these clock source options:

1. Internal oscillator: An internally generated, fixed frequency clock, trimmable to ± 0.4%. There are three choices for the internal oscillator,12.8 MHz, 8 MHz, or 4 MHz. The 4-MHz internal oscillator is the default option out of reset.
2. External oscillator: An external clock that can be driven directly into OSC1.
3. External RC: A built-in oscillator module (RC oscillator) that requires an external R connection only. The capacitor is internal to the chip.
4. External crystal: A built-in XTAL oscillator that requires an external crystal or ceramic-resonator. There are three crystal frequency ranges supported, 8–32 MHz, 1–8 MHz, and 32–100 kHz.

11.3 Functional Description

The oscillator contains these major subsystems:

- Internal oscillator circuit
- Internal or external clock switch control
- External clock circuit
- External crystal circuit
- External RC clock circuit



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device
PTA[0:5]: Higher current sink and source capability

Figure 11-1. Block Diagram Highlighting OSC Block and Pins

11.3.1 Internal Signal Definitions

The following signals and clocks are used in the functional description and figures of the OSC module.

11.3.1.1 Oscillator Enable Signal (SIMOSCEN)

The SIMOSCEN signal comes from the system integration module (SIM) and disables the XTAL oscillator circuit, the RC oscillator, or the internal oscillator in stop mode. OSCENINSTOP in the configuration register can be used to override this signal.

11.3.1.2 XTAL Oscillator Clock (XTALCLK)

XTALCLK is the XTAL oscillator output signal. It runs at the full speed of the crystal (f_{XCLK}) and comes directly from the crystal oscillator circuit. Figure 11-2 shows only the logical relation of XTALCLK to OSC1 and OSC2 and may not represent the actual circuitry. The duty cycle of XTALCLK is unknown and may depend on the crystal and other external factors. The frequency of XTALCLK can be unstable at start up.

11.3.1.3 RC Oscillator Clock (RCCLK)

RCCLK is the RC oscillator output signal. Its frequency is directly proportional to the time constant of the external R (R_{EXT}) and internal C. Figure 11-3 shows only the logical relation of RCCLK to OSC1 and may not represent the actual circuitry.

11.3.1.4 Internal Oscillator Clock (INTCLK)

INTCLK is the internal oscillator output signal. INTCLK is software selectable to be nominally 12.8 MHz, 8.0 MHz, or 4.0 MHz. INTCLK can be digitally adjusted using the oscillator trimming feature of the OSCTRIM register (see 11.3.2.1 Internal Oscillator Trimming).

11.3.1.5 Bus Clock Times 4 (BUSCLKX4)

BUSCLKX4 is the same frequency as the input clock (XTALCLK, RCCLK, or INTCLK). This signal is driven to the SIM module and is used during recovery from reset and stop and is the clock source for the COP module.

11.3.1.6 Bus Clock Times 2 (BUSCLKX2)

The frequency of this signal is equal to half of the BUSCLKX4. This signal is driven to the SIM for generation of the bus clocks used by the CPU and other modules on the MCU. BUSCLKX2 will be divided by two in the SIM. The internal bus frequency is one fourth of the XTALCLK, RCCLK, or INTCLK frequency.

11.3.2 Internal Oscillator

The internal oscillator circuit is designed for use with no external components to provide a clock source with a tolerance of less than ±25% untrimmed. An 8-bit register (OSCTRIM) allows the digital adjustment to a tolerance of ACC_{INT}. See the oscillator characteristics in the Electrical section of this data sheet.

The internal oscillator is capable of generating clocks of 12.8 MHz, 8.0 MHz, or 4.0 MHz (INTCLK) resulting in a bus frequency (INTCLK divided by 4) of 3.2 MHz, 2.0 MHz, or 1.0 MHz respectively. The bus clock is software selectable and defaults to the 1.0-MHz bus out of reset. Users can increase the bus frequency based on the voltage range of their application.

Figure 11-3 shows how BUSCLKX4 is derived from INTCLK and OSC2 can output BUSCLKX4 by setting OSC2EN.

11.3.2.1 Internal Oscillator Trimming

OSCTRIM allows a clock period adjustment of +127 and –128 steps. Increasing the OSCTRIM value increases the clock period, which decreases the clock frequency. Trimming allows the internal clock frequency to be fine tuned to the target frequency.

All devices are factory programmed with a trim value that is stored in FLASH memory at location $FFC0. The trim value is not automatically loaded into the OSCTRIM register. User software must copy the trim value from $FFC0 into OSCTRIM if needed. The factory trim value provides the accuracy required for communication using forced monitor mode. Some production programmers erase the factory trim value, so confirm with your programmer vendor that the trim value at $FFC0 is preserved, or is re-trimmed. Trimming the device in the user application board will provide the most accurate trim value.

11.3.2.2 Internal to External Clock Switching

When external clock source (external OSC, RC, or XTAL) is desired, the user must perform the following steps:

1. For external crystal circuits only, configure OSCOPT[1:0] to external crystal. To help precharge an external crystal oscillator, momentarily configure OSC2 as an output and drive it high for several cycles. This can help the crystal circuit start more robustly.
2. Configure OSCOPT[1:0] and ECFS[1:0] according to 11.8.1 Oscillator Status and Control Register. The oscillator module control logic will then enable OSC1 as an external clock input and, if the external crystal option is selected, OSC2 will also be enabled as the clock output. If RC oscillator option is selected, enabling the OSC2 output may change the bus frequency.
3. Create a software delay to provide the stabilization time required for the selected clock source (crystal, resonator, RC). A good rule of thumb for crystal oscillators is to wait 4096 cycles of the crystal frequency; i.e., for a 4-MHz crystal, wait approximately 1 ms.
4. After the stabilization delay has elapsed, set ECGON.

After ECGON set is detected, the OSC module checks for oscillator activity by waiting two external clock rising edges. The OSC module then switches to the external clock. Logic provides a coherent transition. The OSC module first sets ECGST and then stops the internal oscillator.

11.3.2.3 External to Internal Clock Switching

After following the procedures to switch to an external clock source, it is possible to go back to the internal source. By clearing the OSCOPT[1:0] bits and clearing the ECGON bit, the external circuit will be disengaged. The bus clock will be derived from the selected internal clock source based on the ICFS[1:0] bits.

11.3.3 External Oscillator

The external oscillator option is designed for use when a clock signal is available in the application to provide a clock source to the MCU. The OSC1 pin is enabled as an input by the oscillator module. The clock signal is used directly to create BUSCLKX4 and also divided by two to create BUSCLKX2.

In this configuration, the OSC2 pin cannot output BUSCLKX4. The OSC2EN bit will be forced clear to enable alternative functions on the pin.

11.3.4 XTAL Oscillator

The XTAL oscillator circuit is designed for use with an external crystal or ceramic resonator to provide an accurate clock source. In this configuration, the OSC2 pin is dedicated to the external crystal circuit. The OSC2EN bit has no effect when this clock mode is selected.

In its typical configuration, the XTAL oscillator is connected in a Pierce oscillator configuration, as shown in Figure 11-2. This figure shows only the logical representation of the internal components and may not represent actual circuitry.

The oscillator configuration uses five components:

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

NOTE

The series resistor (R_S) is included in the diagram to follow strict Pierce oscillator guidelines and may not be required for all ranges of operation, especially with high frequency crystals. Refer to the oscillator characteristics table in the Electricals section for more information.

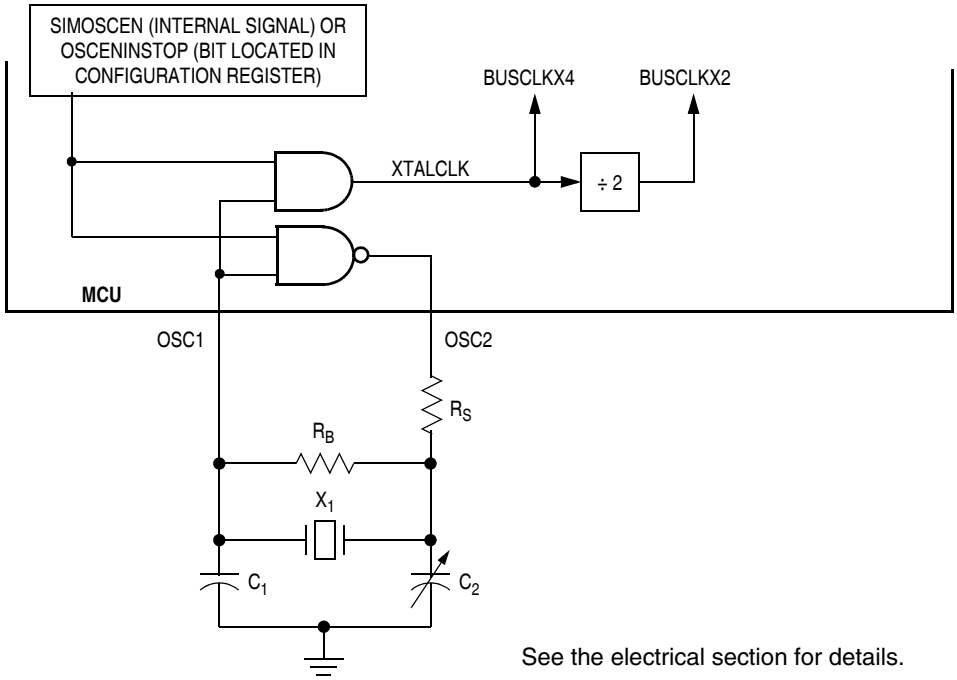


Figure 11-2. XTAL Oscillator External Connections

11.3.5 RC Oscillator

The RC oscillator circuit is designed for use with an external resistor (R_{EXT}) to provide a clock source with a tolerance within 25% of the expected frequency. See Figure 11-3.

The capacitor (C) for the RC oscillator is internal to the MCU. The R_{EXT} value must have a tolerance of 1% or less to minimize its effect on the frequency.

In this configuration, the OSC2 pin can be used as general-purpose input/output (I/O) port pins or other alternative pin function. The OSC2EN bit can be set to enable the OSC2 output function on the pin. Enabling the OSC2 output can affect the external RC oscillator frequency, f_{RCCLK}.

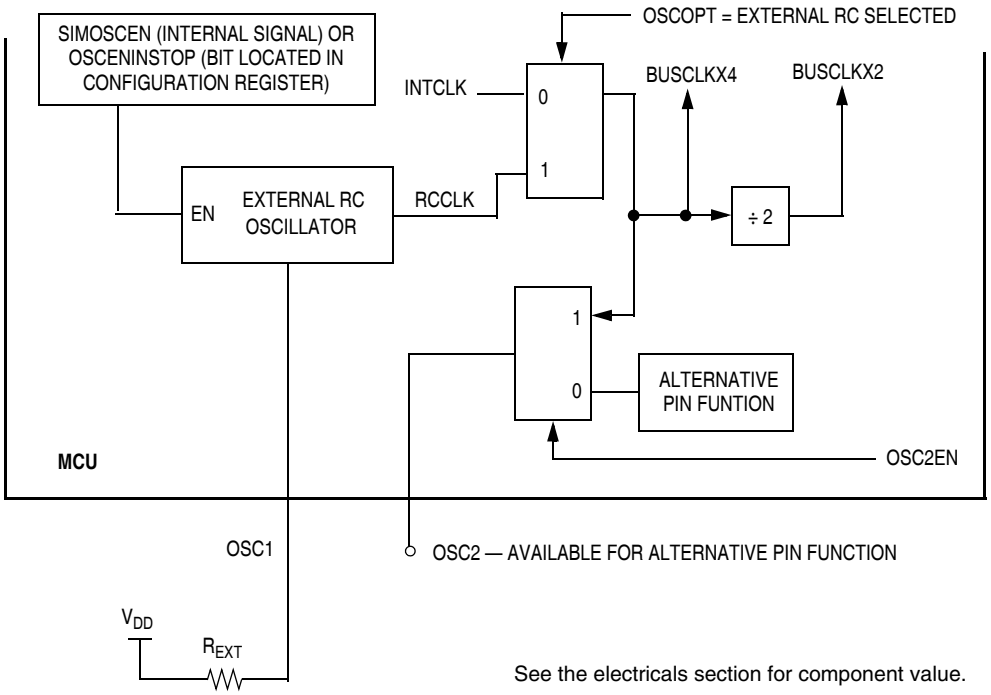


Figure 11-3. RC Oscillator External Connections

11.4 Interrupts

There are no interrupts associated with the OSC module.

11.5 Low-Power Modes

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

11.5.1 Wait Mode

The OSC module remains active in wait mode.

11.5.2 Stop Mode

The OSC module can be configured to remain active in stop mode by setting OSCENINSTOP located in a configuration register.

11.6 OSC During Break Interrupts

There are no status flags associated with the OSC module.

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.

11.7 I/O Signals

The OSC shares its pins with general-purpose input/output (I/O) port pins. See Figure 11-1 for port location of these shared pins.

11.7.1 Oscillator Input Pin (OSC1)

The OSC1 pin is an input to the crystal oscillator amplifier, an input to the RC oscillator circuit, or an input from an external clock source.

When the OSC is configured for internal oscillator, the OSC1 pin can be used as a general-purpose input/output (I/O) port pin or other alternative pin function.

11.7.2 Oscillator Output Pin (OSC2)

For the XTAL oscillator option, the OSC2 pin is the output of the crystal oscillator amplifier.

When the OSC is configured for internal oscillator, external clock, or RC, the OSC2 pin can be used as a general-purpose I/O port pin or other alternative pin function. When the oscillator is configured for internal or RC, the OSC2 pin can be used to output BUSCLKX4.

Table 11-1. OSC2 Pin Function

Option	OSC2 Pin Function
XTAL oscillator	Inverting OSC1
External clock	General-purpose I/O or alternative pin function
Internal oscillator or RC oscillator	Controlled by OSC2EN bit OSC2EN = 0: General-purpose I/O or alternative pin function OSC2EN = 1: BUSCLKX4 output

11.8 Registers

The oscillator module contains two registers:

- Oscillator status and control register (OSCSC)
- Oscillator trim register (OSCTRIM)

11.8.1 Oscillator Status and Control Register

The oscillator status and control register (OSCSC) contains the bits for switching between internal and external clock sources. If the application uses an external crystal, bits in this register are used to select the crystal oscillator amplifier necessary for the desired crystal. While running off the internal clock source, the user can use bits in this register to select the internal clock source frequency.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	OSCOPT1	OSCOPT0	ICFS1	ICFS0	ECFS1	ECFS0	ECGON	ECGST
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-4. Oscillator Status and Control Register (OSCSC)

OSCOPT1:OSCOPT0 — OSC Option Bits

These read/write bits allow the user to change the clock source for the MCU. The default reset condition has the bus clock being derived from the internal oscillator. See 11.3.2.2 Internal to External Clock Switching for information on changing clock sources.

OSCOPT1	OSCOPT0	Oscillator Modes
0	0	Internal oscillator (frequency selected using ICFSx bits)
0	1	External oscillator clock
1	0	External RC
1	1	External crystal (range selected using ECFSx bits)

ICFS1:ICFS0 — Internal Clock Frequency Select Bits

These read/write bits enable the frequency to be increased for applications requiring a faster bus clock when running off the internal oscillator. The WAIT instruction has no effect on the oscillator logic. BUSCLKX2 and BUSCLKX4 continue to drive to the SIM module.

ICFS1	ICFS0	Internal Clock Frequency
0	0	4.0 MHz — default reset condition
0	1	8.0 MHz
1	0	12.8 MHz
1	1	Reserved

ECFS1:ECFS0 — External Crystal Frequency Select Bits

These read/write bits enable the specific amplifier for the crystal frequency range. Refer to oscillator characteristics table in the Electricals section for information on maximum external clock frequency versus supply voltage.

ECFS1	ECFS0	External Crystal Frequency
0	0	8 MHz – 32 MHz
0	1	1 MHz – 8 MHz
1	0	32 kHz – 100 kHz
1	1	Reserved

ECGON — External Clock Generator On Bit

This read/write bit enables the OSC1 pin as the clock input to the MCU, so that the switching process can be initiated. This bit is cleared by reset. This bit is ignored in monitor mode with the internal oscillator bypassed.

1 = External clock enabled
0 = External clock disabled

ECGST — External Clock Status Bit

This read-only bit indicates whether an external clock source is engaged to drive the system clock.

1 = An external clock source engaged
0 = An external clock source disengaged

11.8.2 Oscillator Trim Register (OSCTRIM)

	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 11-5. Oscillator Trim Register (OSCTRIM)

TRIM7–TRIM0 — Internal Oscillator Trim Factor Bits

These read/write bits change the internal capacitance used by the internal oscillator. By measuring the period of the internal clock and adjusting this factor accordingly, the frequency of the internal clock can be fine tuned. Increasing (decreasing) this factor by one increases (decreases) the period by approximately 0.2% of the untrimmed oscillator period. The oscillator period is based on the oscillator frequency selected by the ICFS bits in OSCSC.

Applications using the internal oscillator should copy the internal oscillator trim value at location $FFC0 into this register to trim the clock source.

Chapter 12
Input/Output Ports (PORTS)

12.1 Introduction

The MC68HC908QB8, MC68HC908QB4 and MC68HC908QY8 have thirteen bidirectional pins and one input only pin.

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 an 6-bit special function port that shares its pins with the keyboard interrupt (KBI) module (see Chapter 9 Keyboard Interrupt Module (KBI), the 4-channel timer interface module (TIM) (see Chapter 16 Timer Interface Module (TIM)), the 10-bit ADC (see Chapter 3 Analog-to-Digital Converter (ADC10) Module), the external interrupt (IRQ) pin (see Chapter 8 External Interrupt (IRQ)), the reset (RST) pin enabled using a configuration register (see Chapter 5 Configuration Register (CONFIG)) and the oscillator pins (see Chapter 11 Oscillator Module (OSC)).

Each port A pin also has a software configurable pullup device if the corresponding port pin is configured as an input port.

NOTE

PTA2 is input only.

When the $\overline{IRQ}$ function is enabled in the configuration register 2 (CONFIG2), bit 2 of the port A data register (PTA) will always read a logic 0. In this case, the BIH and BIL instructions can be used to read the logic level on the PTA2 pin. When the $\overline{IRQ}$ function is disabled, these instructions will behave as if the PTA2 pin is a logic 1. However, reading bit 2 of PTA will read the actual logic level on the pin.

12.2.1 Port A Data Register

The port A data register (PTA) contains a data latch for each of the six port A pins.

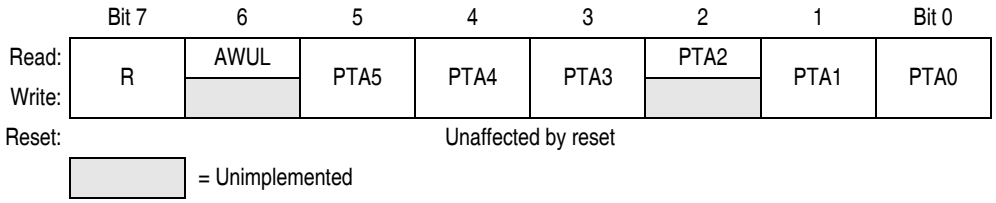


Figure 12-1. Port A Data Register (PTA)

PTA[5:0] — 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.

AWUL — Auto Wakeup Latch Data Bit

This is a read-only bit which has the value of the auto wakeup interrupt request latch. The wakeup request signal is generated internally (see Chapter 4 Auto Wakeup Module (AWU)). There is no PTA6 port nor any of the associated bits such as PTA6 data register, pullup enable or direction. .

12.2.2 Data Direction Register A

Data direction register A (DDRA) determines whether each port A pin is an input or an output. Writing a 1 to a DDRA bit enables the output buffer for the corresponding port A pin; a 0 disables the output buffer.

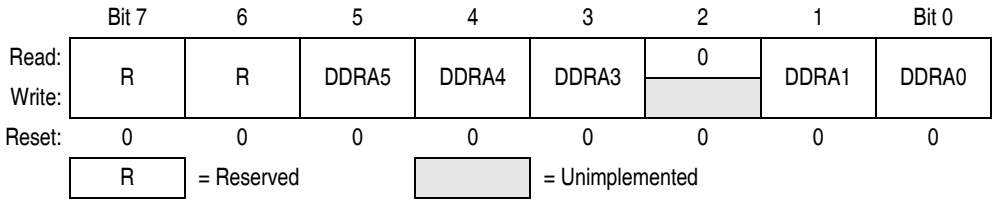


Figure 12-2. Data Direction Register A (DDRA)

DDRA[5:0] — Data Direction Register A Bits

These read/write bits control port A data direction. Reset clears DDRA[5:0], 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.

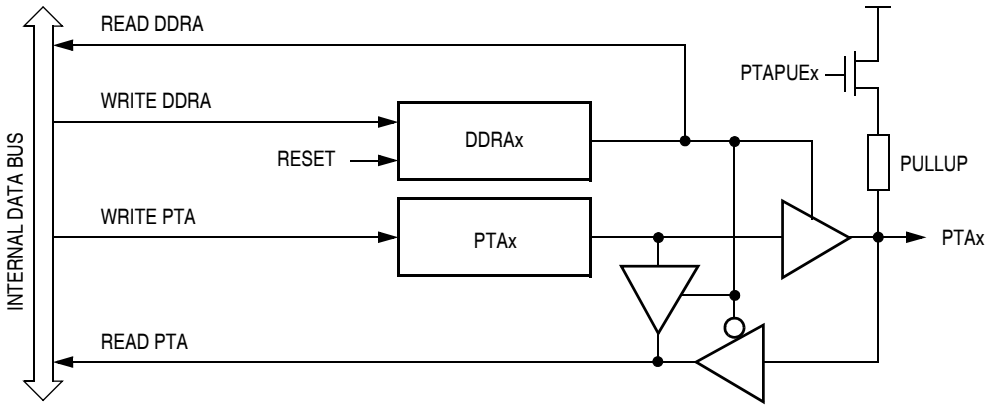


Figure 12-3. Port A I/O Circuit

NOTE

Figure 12-3 does not apply to PTA2

When DDRAx is a 1, reading PTA reads the PTAx data latch. When DDRAx is a 0, reading PTA reads the logic level on the PTAx pin. The data latch can always be written, regardless of the state of its data direction bit.

12.2.3 Port A Input Pullup Enable Register

The port A input pullup enable register (PTAPUE) contains a software configurable pullup device for each of the port A pins. Each bit is individually configurable and requires the corresponding data direction register, DDRAx, to be configured as input. Each pullup device is automatically and dynamically disabled when its corresponding DDRAx bit is configured as output.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OSC2EN		PTAPUE5	PTAPUE4	PTAPUE3	PTAPUE2	PTAPUE1	PTAPUE0
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 12-4. Port A Input Pullup Enable Register (PTAPUE)

OSC2EN — Enable PTA4 on OSC2 Pin

This read/write bit configures the OSC2 pin function when internal oscillator or RC oscillator option is selected. This bit has no effect for the XTAL or external oscillator options.

1 = OSC2 pin outputs the internal or RC oscillator clock (BUSCLKX4)

0 = OSC2 pin configured for PTA4 I/O, having all the interrupt and pullup functions

PTAPUE[5:0] — Port A Input Pullup Enable Bits

These read/write bits are software programmable to enable pullup devices on port A pins.

1 = Corresponding port A pin configured to have internal pullup if its DDRA bit is set to 0

0 = Pullup device is disconnected on the corresponding port A pin regardless of the state of its DDRA bit

12.2.4 Port A Summary Table

The following table summarizes the operation of the port A pins when used as a general-purpose input/output pins.

Table 12-1. Port A Pin Functions

PTAPUE Bit	DDRA Bit	PTA Bit	I/O Pin Mode	Accesses to DDRA	Accesses to PTA	
				Read/Write	Read	Write
1	0	X(1)	Input, V_{DD}(2)	DDRA5–DDRA0	Pin	PTA5–PTA0(3)
0	0	X	Input, Hi-Z(4)	DDRA5–DDRA0	Pin	PTA5–PTA0(3)
X	1	X	Output	DDRA5–DDRA0	PTA5–PTA0	PTA5–PTA0(5)

1. X = don't care
2. I/O pin pulled to V_{DD} by internal pullup.
3. Writing affects data register, but does not affect input.
4. Hi-Z = high impedance
5. Output does not apply to PTA2

12.3 Port B

Port B is an 8-bit special function port that shares its pins with the 4-channel timer interface module (TIM) (see Chapter 16 Timer Interface Module (TIM)), the 10-bit ADC (see Chapter 3 Analog-to-Digital Converter (ADC10) Module), the serial peripheral interface (SPI) module (see Chapter 15 Serial Peripheral Interface (SPI) Module) and the enhanced serial communications interface (ESCI) module (see Chapter 13 Enhanced Serial Communications Interface (ESCI) Module**).**

Each port B pin also has a software configurable pullup device if the corresponding port pin is configured as an input port.

12.3.1 Port B Data Register

The port B data register (PTB) contains a data latch for each of the port B pins.

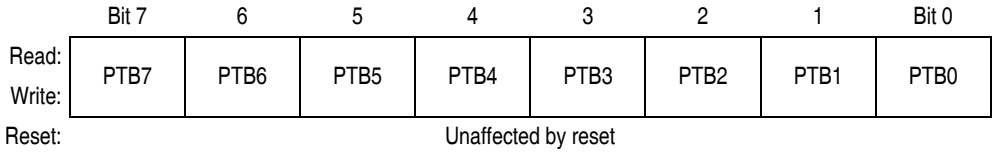


Figure 12-5. Port B Data Register (PTB)

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 (DDRB) determines whether each port B pin is an input or an output. Writing a 1 to a DDRB bit enables the output buffer for the corresponding port B pin; a 0 disables the output buffer.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Reset:	0	0	0	0	0	0	0	0

Figure 12-6. 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 12-7 shows the port B I/O logic.

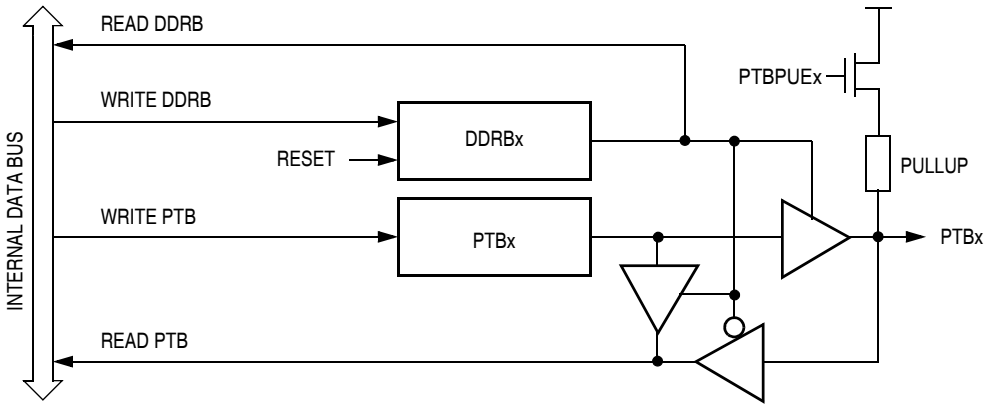


Figure 12-7. Port B I/O Circuit

When DDRBx is a 1, reading PTB reads the PTBx data latch. When DDRBx is a 0, reading PTB reads the logic level on the PTBx pin. The data latch can always be written, regardless of the state of its data direction bit.

12.3.3 Port B Input Pullup Enable Register

The port B input pullup enable register (PTBPUE) contains a software configurable pullup device for each of the eight port B pins. Each bit is individually configurable and requires the corresponding data direction register, DDRBx, be configured as input. Each pullup device is automatically and dynamically disabled when its corresponding DDRBx bit is configured as output.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTBPUE7	PTBPUE6	PTBPUE5	PTBPUE4	PTBPUE3	PTBPUE2	PTBPUE2	PTBPUE0
Reset:	0	0	0	0	0	0	0	0

Figure 12-8. Port B Input Pullup Enable Register (PTBPUE)

PTBPUE[7:0] — Port B Input Pullup Enable Bits

These read/write bits are software programmable to enable pullup devices on port B pins

1 = Corresponding port B pin configured to have internal pull if its DDRB bit is set to 0

0 = Pullup device is disconnected on the corresponding port B pin regardless of the state of its DDRB bit.

12.3.4 Port B Summary Table

Table 12-2 summarizes the operation of the port A pins when used as a general-purpose input/output 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(1)	Input, Hi-Z(2)	DDRB7–DDRB0	Pin	PTB7–PTB0(3)
1	X	Output	DDRB7–DDRB0	Pin	PTB7–PTB0

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

Chapter 13
Enhanced Serial Communications Interface (ESCI) Module

13.1 Introduction

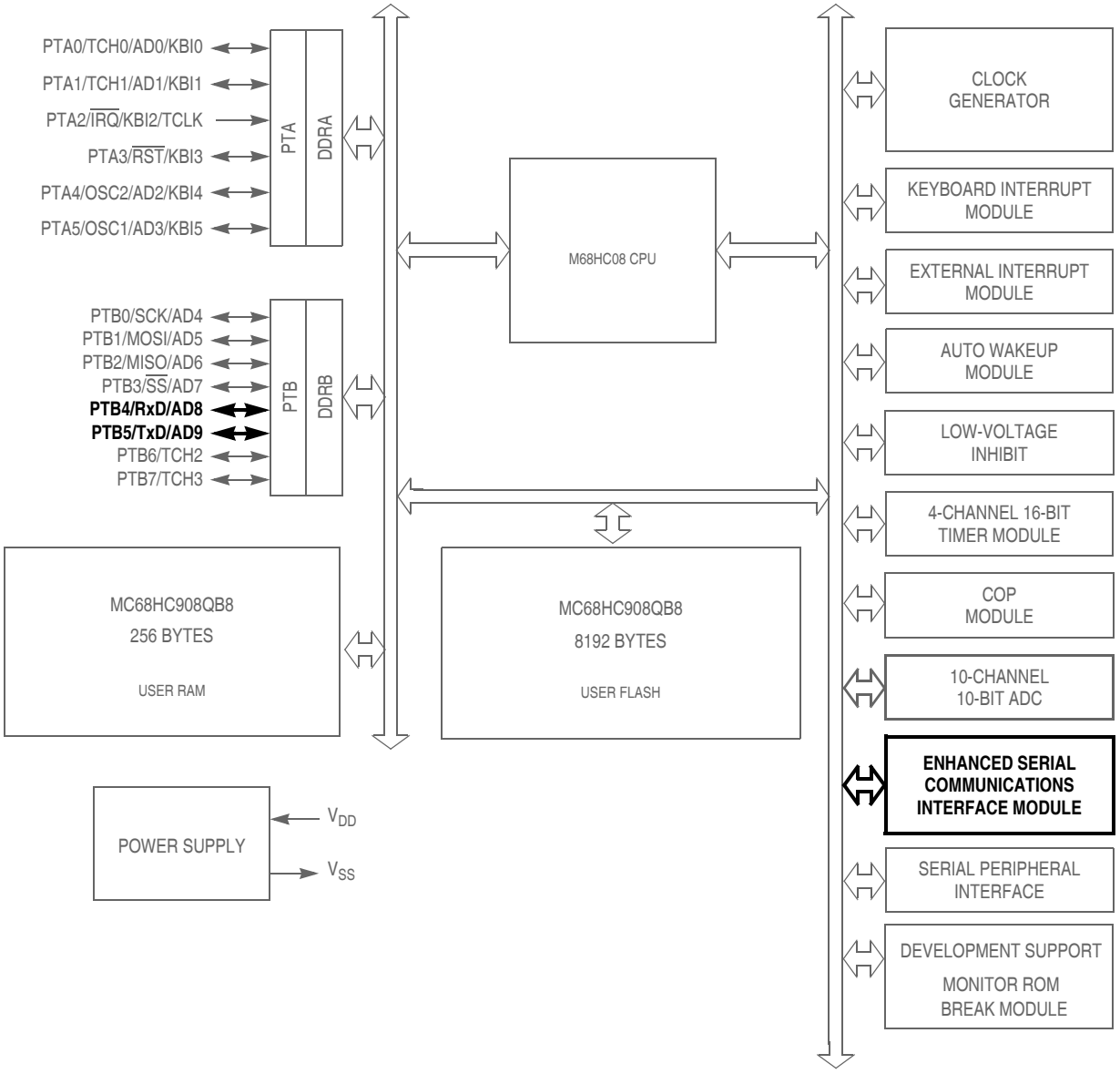
The enhanced serial communications interface (ESCI) module allows asynchronous communications with peripheral devices and other microcontroller units (MCU).

The ESCI module shares its pins with general-purpose input/output (I/O) port pins. See Figure 13-1 for port location of these shared pins. The ESCI baud rate clock source is controlled by a bit (ESCIBDSRC) located in the configuration register.

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 interrupt requests
- Programmable transmitter output polarity
- Receiver wakeup methods
  - Idle line
  - Address mark
- 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



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device
PTA[0:5]: Higher current sink and source capability

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

13.3 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.

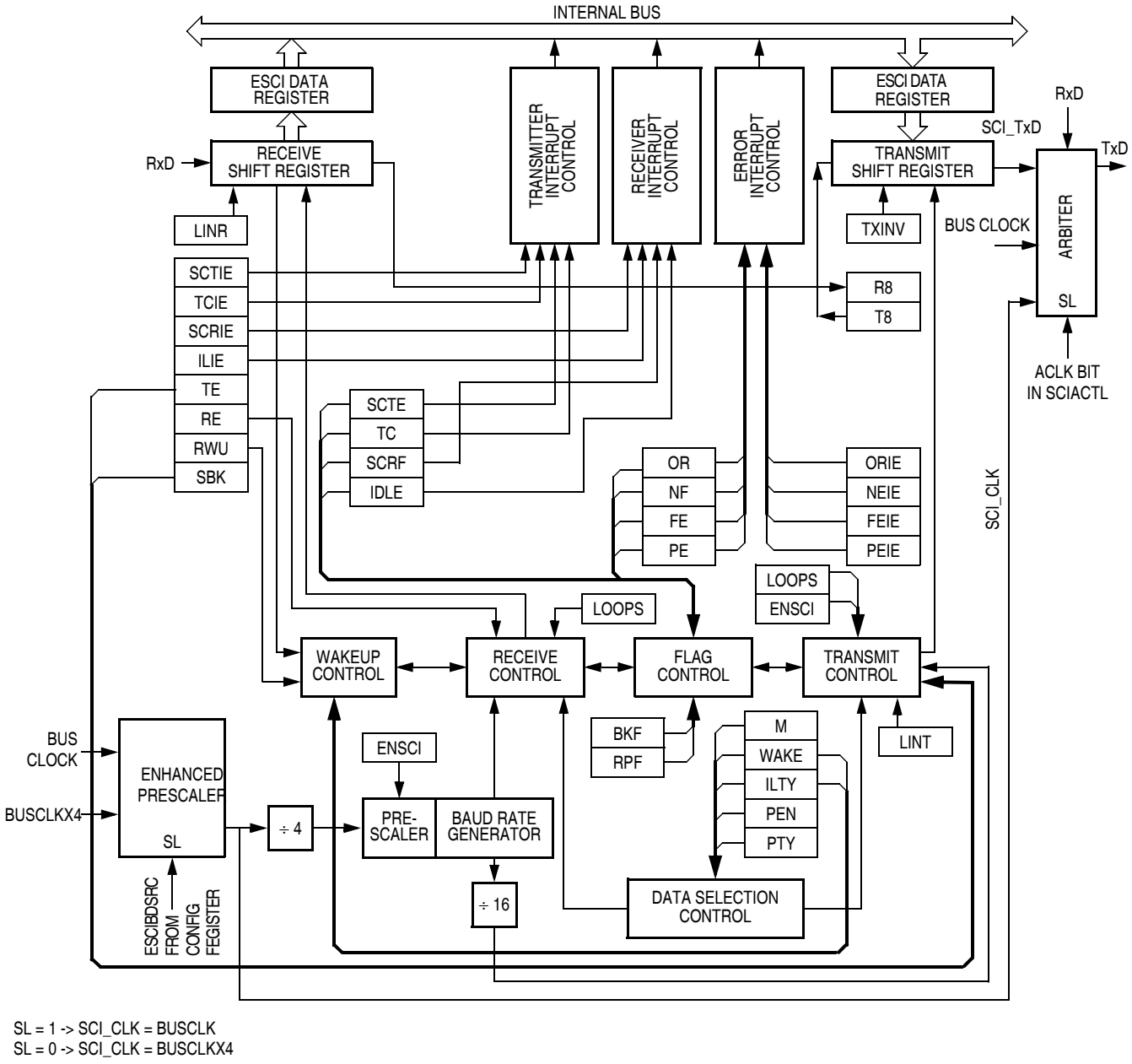


Figure 13-2. ESCI Module Block Diagram

13.3.1 Data Format

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

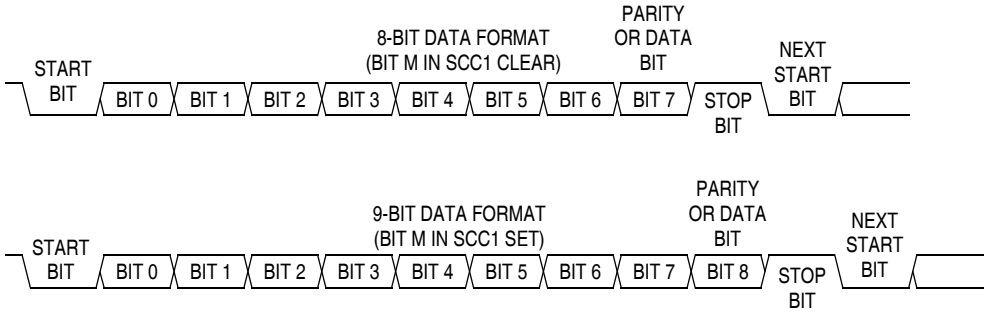


Figure 13-3. SCI Data Formats

13.3.2 Transmitter

Figure 13-4 shows the structure of the SCI transmitter.

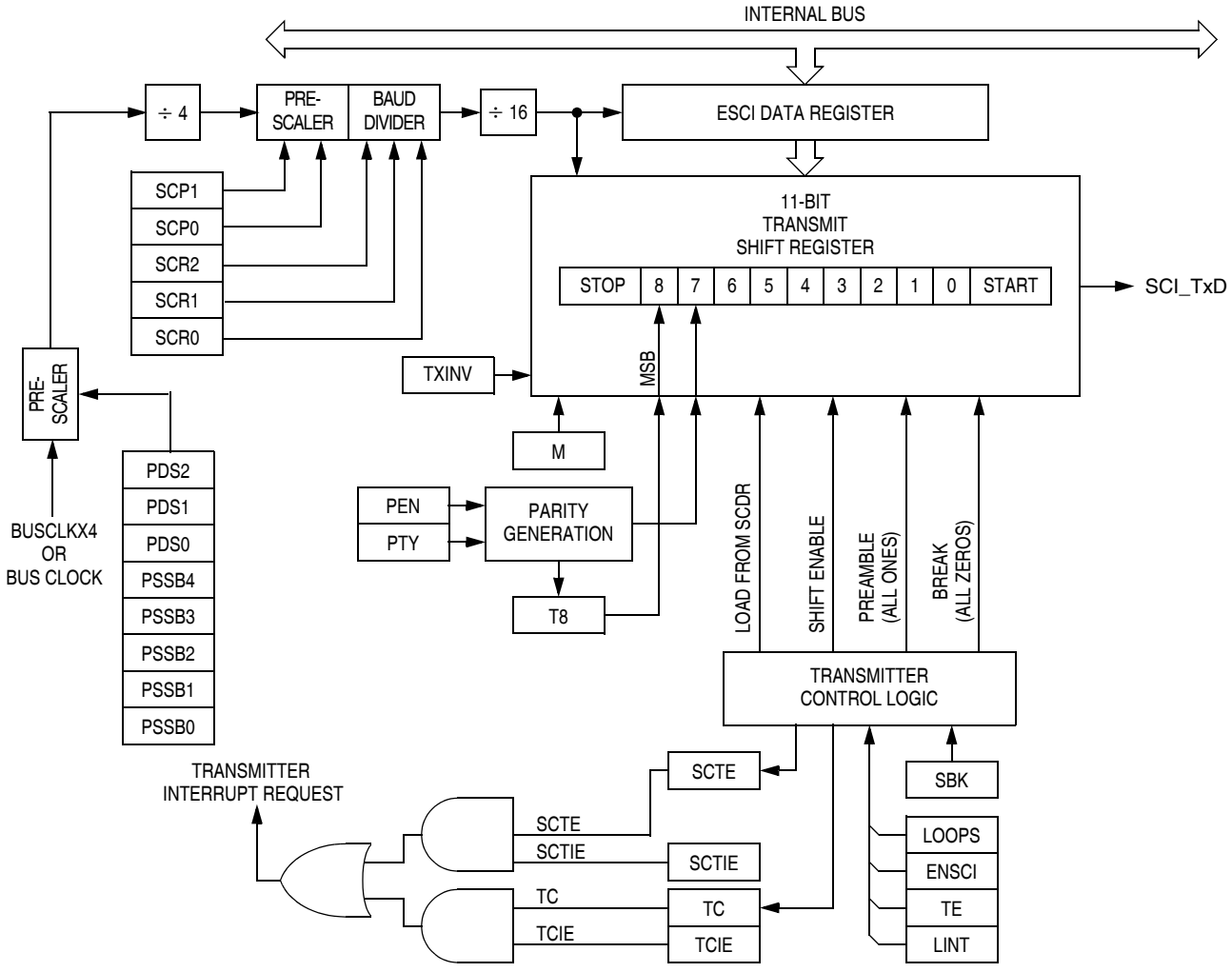


Figure 13-4. ESCI Transmitter

13.3.2.1 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).

13.3.2.2 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 interrupt request.

When the transmit shift register is not transmitting a character, the TxD pin goes to the idle condition, high. If at any time software clears the ENSCI bit in ESCI control register 1 (SCC1), the transmitter and receiver relinquish control of the port pins.

13.3.2.3 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 set, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one 1. The automatic 1 at the end of a break character guarantees the recognition of the start bit of the next character.

When LINR is cleared in SCBR, the ESCI recognizes a break character when a start bit is followed by eight or nine 0 data bits and a 0 where the stop bit should be, resulting in a total of 10 or 11 consecutive 0 data bits. When LINR is set in SCBR, the ESCI recognizes a break character when a start bit is followed by 9 or 10 0 data bits and a 0 where the stop bit should be, resulting in a total of 11 or 12 consecutive 0 data bits.

Receiving a break character has these effects on ESCI registers:

- Sets the framing error bit (FE) in SCS1
- Sets the ESCI receiver full bit (SCRF) in SCS1
- Clears the ESCI data register (SCDR)
- Clears the R8 bit in SCC3
- Sets the break flag bit (BKF) in SCS2
- May set the overrun (OR), noise flag (NF), parity error (PE), or reception in progress flag (RPF) bits

13.3.2.4 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.

13.3.2.5 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 set. See 13.8.1 ESCI Control Register 1.

13.3.3 Receiver

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

13.3.3.1 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).

13.3.3.2 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 interrupt request.

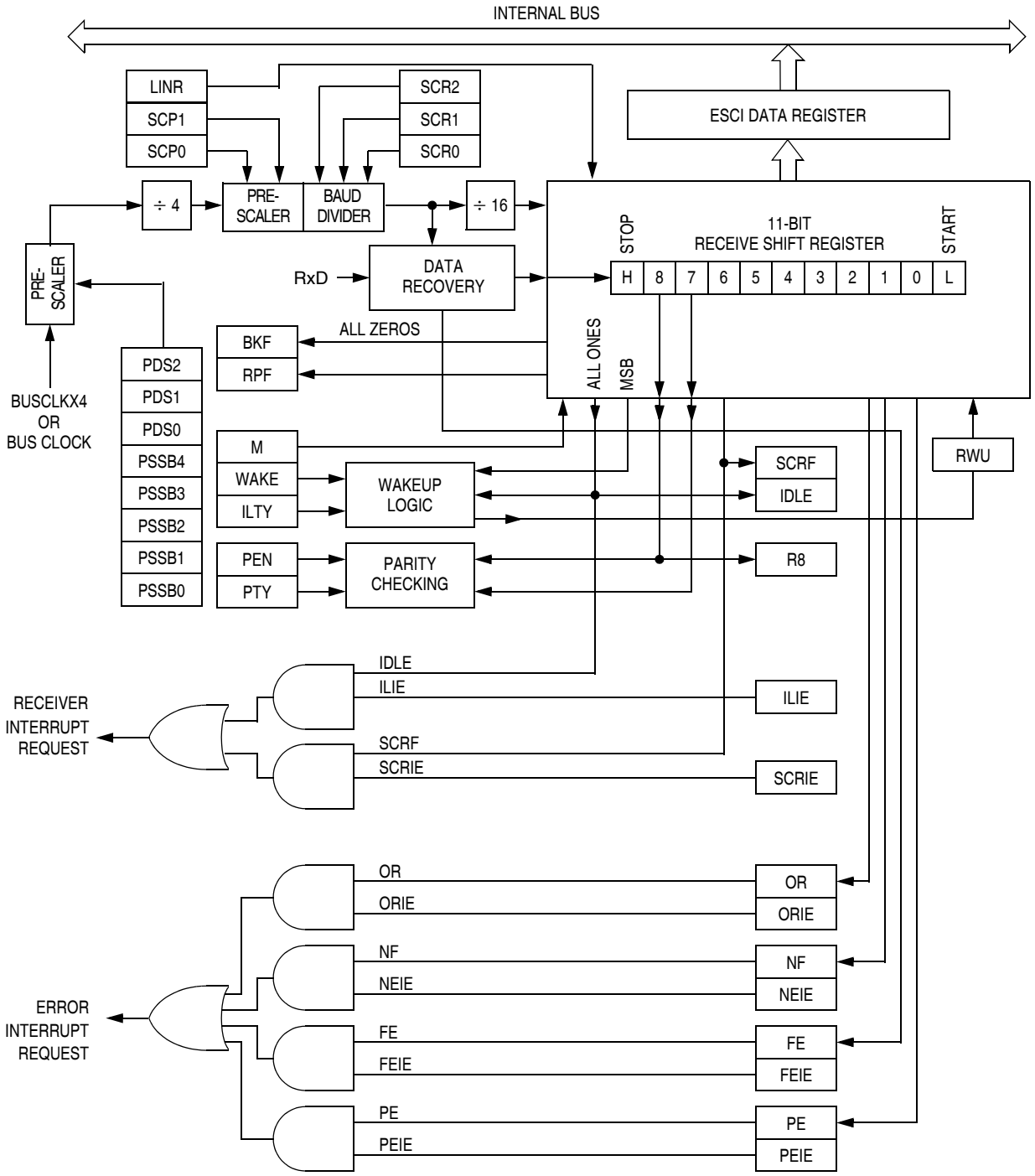


Figure 13-5. ESCI Receiver Block Diagram

13.3.3.3 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.

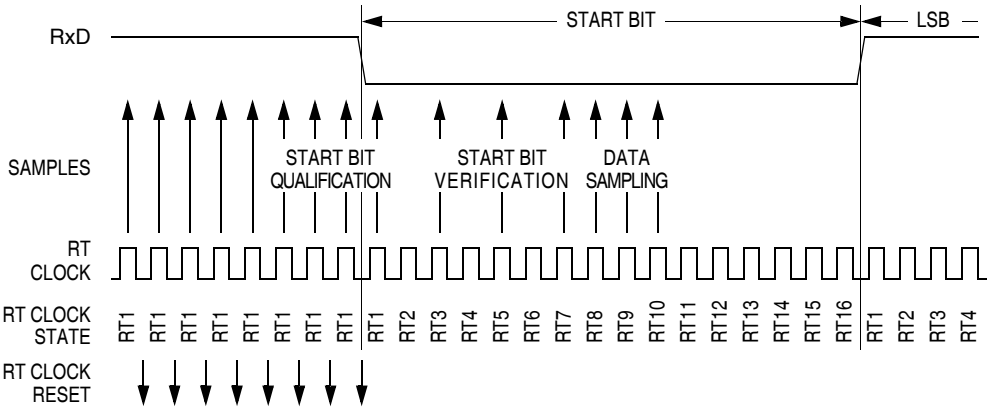


Figure 13-6. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 13-1 summarizes the results of the start bit verification samples.

Table 13-1. 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 13-2 summarizes the results of the data bit samples.

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

Table 13-3. 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

13.3.3.4 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.

13.3.3.5 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.

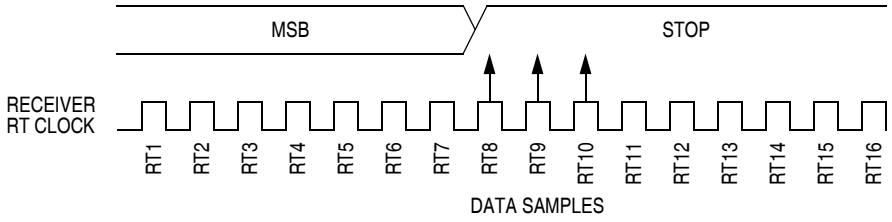


Figure 13-7. Slow Data

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.

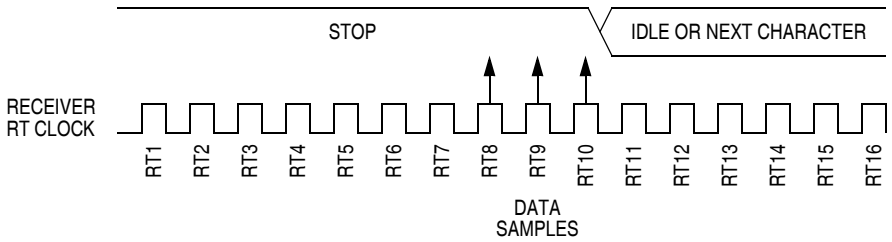


Figure 13-8. Fast Data

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:

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

13.3.3.6 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 Interrupts

The following sources can generate ESCI interrupt requests.

13.4.1 Transmitter Interrupts

These conditions can generate 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 interrupt request. Setting the ESCI transmit interrupt enable bit, SCTIE, in SCC2 enables the SCTE bit to generate transmitter 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 interrupt requests.

13.4.2 Receiver Interrupts

These sources can generate 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 interrupt request. Setting the ESCI receive interrupt enable bit, SCRIE, in SCC2 enables the SCRF bit to generate receiver 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 interrupt requests.

13.4.3 Error Interrupts

These receiver error flags in SCS1 can generate 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 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 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 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 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 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 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.

13.7 I/O Signals

The ESCI module can share its pins with the general-purpose I/O pins. See Figure 13-1 for the port pins that are shared.

13.7.1 ESCI Transmit Data (TxD)

The TxD pin is the serial data output from the ESCI transmitter. When the ESCI is enabled, the TxD pin becomes an output.

13.7.2 ESCI Receive Data (RxD)

The RxD pin is the serial data input to the ESCI receiver. When the ESCI is enabled, the RxD pin becomes an input.

13.8 Registers

The following registers control and monitor operation of the ESCI:

- 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

	Bit 7	6	5	4	3	2	1	Bit 0
Read: 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.

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.

1 = ESCI enabled
0 = ESCI disabled

TXINV — Transmit Inversion Bit

This read/write bit reverses the polarity of transmitted data.

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 13-4).The ninth bit can serve as a receiver wakeup signal or as a parity bit.

1 = 9-bit ESCI characters
0 = 8-bit ESCI characters

Table 13-4. 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.

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.

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 13-4). When enabled, the parity function inserts a parity bit in the MSB position (see Table 13-2).

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 13-4).

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 interrupt requests:
  - SCTE bit to generate transmitter interrupt requests
  - TC bit to generate transmitter interrupt requests
  - SCRF bit to generate receiver interrupt requests
  - IDLE bit to generate receiver interrupt requests
- Enables the transmitter

- Enables the receiver
- Enables ESCI wakeup
- Transmits ESCI break characters

	Bit 7	6	5	4	3	2	1	Bit 0
Read: 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 interrupt requests. Setting the SCTIE bit in SCC2 enables the SCTE bit to generate interrupt requests.

1 = SCTE enabled to generate interrupt
0 = SCTE not enabled to generate interrupt

TCIE — Transmission Complete Interrupt Enable Bit

This read/write bit enables the TC bit to generate ESCI transmitter interrupt requests.

1 = TC enabled to generate interrupt requests
0 = TC not enabled to generate interrupt requests

SCRIE — ESCI Receive Interrupt Enable Bit

This read/write bit enables the SCRF bit to generate ESCI receiver interrupt requests. Setting the SCRIE bit in SCC2 enables the SCRF bit to generate interrupt requests.

1 = SCRF enabled to generate interrupt
0 = SCRF not enabled to generate interrupt

ILIE — Idle Line Interrupt Enable Bit

This read/write bit enables the IDLE bit to generate ESCI receiver interrupt requests.

1 = IDLE enabled to generate interrupt requests
0 = IDLE not enabled to generate interrupt requests

TE — Transmitter Enable Bit

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 1s from the transmit shift register to the TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the TxD returns to the idle condition (high). Clearing and then setting TE during a transmission queues an idle character to be sent after the character currently being transmitted.

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.

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.

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.

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

	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).

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.

ORIE — Receiver Overrun Interrupt Enable Bit

This read/write bit enables ESCI error interrupt requests generated by the receiver overrun bit, OR.

1 = ESCI error interrupt requests from OR bit enabled
0 = ESCI error interrupt requests from OR bit disabled

NEIE — Receiver Noise Error Interrupt Enable Bit

This read/write bit enables ESCI error interrupt requests generated by the noise error bit, NE.

1 = ESCI error interrupt requests from NE bit enabled
0 = ESCI error interrupt requests from NE bit disabled

FEIE — Receiver Framing Error Interrupt Enable Bit

This read/write bit enables ESCI error interrupt requests generated by the framing error bit, FE.

1 = ESCI error interrupt requests from FE bit enabled
0 = ESCI error interrupt requests from FE bit disabled

PEIE — Receiver Parity Error Interrupt Enable Bit

This read/write bit enables ESCI receiver interrupt requests generated by the parity error bit, PE.

1 = ESCI error interrupt requests from PE bit enabled
0 = ESCI error 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

	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 interrupt request. When the SCTIE bit in SCC2 is set, SCTE generates an ESCI transmitter interrupt request. In normal operation, clear the SCTE bit by reading SCS1 with SCTE set and then writing to SCDR

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 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.

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 interrupt request. When the SCRIE bit in SCC2 is set the SCRF generates a interrupt request. In normal operation, clear the SCRF bit by reading SCS1 with SCRF set and then reading the SCDR.

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 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.

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 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.

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.

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 interrupt request if the NEIE bit in SCC3 is also set. Clear the NF bit by reading SCS1 and then reading the SCDR.

1 = Noise detected
0 = No noise detected

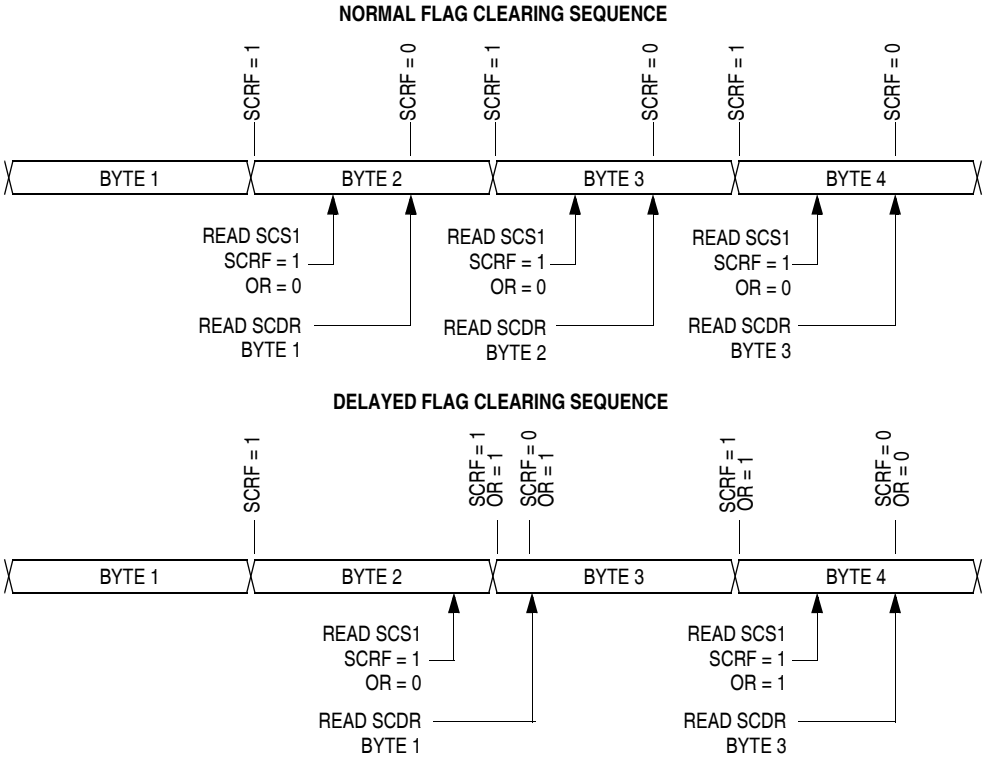


Figure 13-13. Flag Clearing Sequence

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 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.

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 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.

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
- Reception in progress

	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 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.

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.

	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 SCDR accesses the read-only received data bits, R7:R0.
Writing to SCDR writes the data to be transmitted, T7:T0.

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.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: 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 Transmit Enable

This read/write bit selects the enhanced ESCI features for the local interconnect network (LIN) protocol as shown in Table 13-5.

LINR — LIN Receiver Bits

This read/write bit selects the enhanced ESCI features for the local interconnect network (LIN) protocol as shown in Table 13-5.

Table 13-5. ESCI LIN Control Bits

LINT	LINR	M	Functionality
0	0	X	Normal ESCI functionality
0	1	0	11-bit break detect enabled for LIN receiver
0	1	1	12-bit break detect enabled for LIN receiver
1	0	0	13-bit generation enabled for LIN transmitter
1	0	1	14-bit generation enabled for LIN transmitter
1	1	0	11-bit break detect/13-bit generation enabled for LIN
1	1	1	12-bit break detect/14-bit generation enabled for LIN

In LIN (version 1.2 and later) 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. Because 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 13-6.

Table 13-6. 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 13-7. Reset clears SCR2–SCR0.

Table 13-7. 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.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: 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 13-8.

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 13-8. 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 13-9.

Table 13-9. 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

Continued on next page

Table 13-9. ESCI Prescaler Divisor Fine Adjust (Continued)

PSSB[4:3:2:1:0]	Prescaler Divisor Fine Adjust (PDFA)
0 1 1 1 1	15/32 = 0.46875
1 0 0 0 0	16/32 = 0.5
1 0 0 0 1	17/32 = 0.53125
1 0 0 1 0	18/32 = 0.5625
1 0 0 1 1	19/32 = 0.59375
1 0 1 0 0	20/32 = 0.625
1 0 1 0 1	21/32 = 0.65625
1 0 1 1 0	22/32 = 0.6875
1 0 1 1 1	23/32 = 0.71875
1 1 0 0 0	24/32 = 0.75
1 1 0 0 1	25/32 = 0.78125
1 1 0 1 0	26/32 = 0.8125
1 1 0 1 1	27/32 = 0.84375
1 1 1 0 0	28/32 = 0.875
1 1 1 0 1	29/32 = 0.90625
1 1 1 1 0	30/32 = 0.9375
1 1 1 1 1	31/32 = 0.96875

Use the following formula to calculate the ESCI baud rate:

$$\text{Baud rate} = \frac{\text{Frequency of the SCI clock source}}{64 \times \text{BPD} \times \text{BD} \times (\text{PD} + \text{PDFA})}$$

where:

SCI clock source = bus clock or BUSCLKX4 (selected by ESCIBDSRC in the configuration register)
BPD = Baud rate register prescaler divisor
BD = Baud rate divisor
PD = Prescaler divisor
PDFA = Prescaler divisor fine adjust

Table 13-10 shows the ESCI baud rates that can be generated with a 4.9152-MHz bus frequency.

Table 13-10. ESCI Baud Rate Selection Examples

PDS[2:1:0]	PSSB[4:3:2:1:0]	SCP[1:0]	Prescaler Divisor (BPD)	SCR[2:1:0]	Baud Rate Divisor (BD)	Baud Rate (f_{Bus}= 4.9152 MHz)
0 0 0	X X X X X	0 0	1	0 0 0	1	76,800
1 1 1	0 0 0 0 0	0 0	1	0 0 0	1	9600
1 1 1	0 0 0 0 1	0 0	1	0 0 0	1	9562.65
1 1 1	0 0 0 1 0	0 0	1	0 0 0	1	9525.58
1 1 1	1 1 1 1 1	0 0	1	0 0 0	1	8563.07
0 0 0	X X X X X	0 0	1	0 0 1	2	38,400
0 0 0	X X X X X	0 0	1	0 1 0	4	19,200
0 0 0	X X X X X	0 0	1	0 1 1	8	9600
0 0 0	X X X X X	0 0	1	1 0 0	16	4800
0 0 0	X X X X X	0 0	1	1 0 1	32	2400
0 0 0	X X X X X	0 0	1	1 1 0	64	1200
0 0 0	X X X X X	0 0	1	1 1 1	128	600
0 0 0	X X X X X	0 1	3	0 0 0	1	25,600
0 0 0	X X X X X	0 1	3	0 0 1	2	12,800
0 0 0	X X X X X	0 1	3	0 1 0	4	6400
0 0 0	X X X X X	0 1	3	0 1 1	8	3200
0 0 0	X X X X X	0 1	3	1 0 0	16	1600
0 0 0	X X X X X	0 1	3	1 0 1	32	800
0 0 0	X X X X X	0 1	3	1 1 0	64	400
0 0 0	X X X X X	0 1	3	1 1 1	128	200
0 0 0	X X X X X	1 0	4	0 0 0	1	19,200
0 0 0	X X X X X	1 0	4	0 0 1	2	9600
0 0 0	X X X X X	1 0	4	0 1 0	4	4800
0 0 0	X X X X X	1 0	4	0 1 1	8	2400
0 0 0	X X X X X	1 0	4	1 0 0	16	1200
0 0 0	X X X X X	1 0	4	1 0 1	32	600
0 0 0	X X X X X	1 0	4	1 1 0	64	300
0 0 0	X X X X X	1 0	4	1 1 1	128	150
0 0 0	X X X X X	1 1	13	0 0 0	1	5908
0 0 0	X X X X X	1 1	13	0 0 1	2	2954
0 0 0	X X X X X	1 1	13	0 1 0	4	1477
0 0 0	X X X X X	1 1	13	0 1 1	8	739
0 0 0	X X X X X	1 1	13	1 0 0	16	369
0 0 0	X X X X X	1 1	13	1 0 1	32	185
0 0 0	X X X X X	1 1	13	1 1 0	64	92
0 0 0	X X X X X	1 1	13	1 1 1	128	46

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 can control operation mode via the ESCI arbiter control register (SCIACTL).

13.9.1 ESCI Arbiter Control Register

	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 13-18. ESCI Arbiter Control Register (SCIACTL)

AM1 and AM0 — Arbiter Mode Select Bits

These read/write bits select the mode of the arbiter module as shown in Table 13-11.

Table 13-11. 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.

ACLK — Arbiter Counter Clock Select Bit

This read/write bit selects the arbiter counter clock source.

1 = Arbiter counter is clocked with one half of the ESCI input clock generated by the ESCI prescaler
0 = Arbiter counter is clocked with the bus clock divided by four

NOTE

For ACLK = 1, the arbiter input clock is driven from the ESCI prescaler. The prescaler can be clocked by either the bus clock or BUSCLKX4 depending on the state of the ESCIBDSRC bit in configuration register.

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.

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.

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.

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.

13.9.2 ESCI Arbiter Data Register

	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 13-19. 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.

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 bus clock. 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 13-20.
2. ACLK = 1 — The counter is clocked with one half of the ESCI input clock generated by the ESCI prescaler. The counter is started when a 0 is detected on RxD (see Figure 13-21). A 0 on RxD on enabling the bit time measurement with ACLK = 1 leads to immediate start of the counter (see Figure 13-22). The counter will be stopped on the next rising edge of RxD. This mode is used to measure the length of a received break.

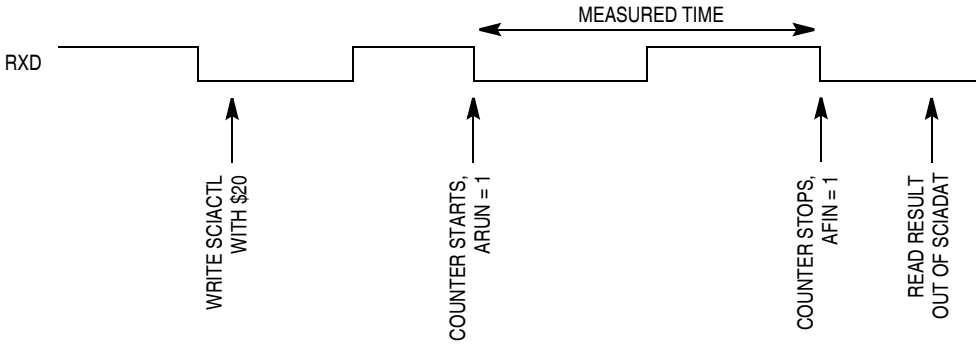


Figure 13-20. Bit Time Measurement with ACLK = 0

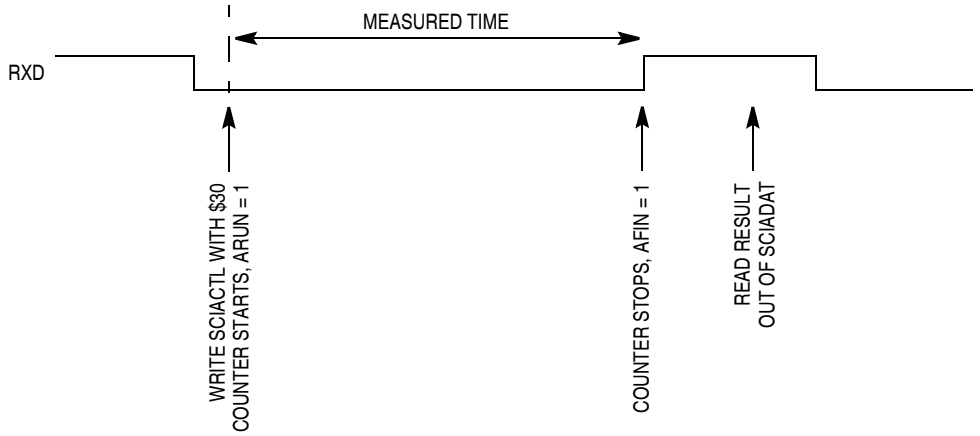


Figure 13-21. Bit Time Measurement with ACLK = 1, Scenario A

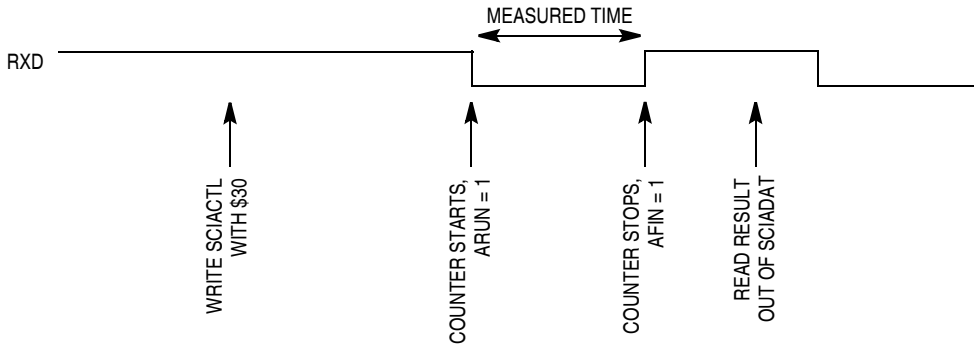


Figure 13-22. Bit Time Measurement with ACLK = 1, Scenario B

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 transmit shift register, see Figure 13-2), the counter is started. When the counter reaches $38 (ACLK = 0) or $08 (ACLK = 1), RxD is statically sensed. If in this case, RxD is sensed low (for example, another bus is driving the bus dominant) ALOST is set. As long as ALOST is set, the TxD pin is forced to 1, resulting in a seized transmission.

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

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. Together with the central processor unit (CPU), the SIM controls all microcontroller unit (MCU) activities. A block diagram of the SIM is shown in Figure 14-1. The SIM is a system state controller that coordinates CPU and exception timing.

The SIM is responsible for:

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

Table 14-1. Signal Name Conventions

Signal Name	Description
BUSCLKX4	Buffered clock from the internal, RC or XTAL oscillator circuit.
BUSCLKX2	The BUSCLKX4 frequency divided by two. This signal is again divided by two in the SIM to generate the internal bus clocks (bus clock = BUSCLKX4 ÷ 4).
Address bus	Internal address bus
Data bus	Internal data bus
PORRST	Signal from the power-on reset module to the SIM
IRST	Internal reset signal
R/$\overline{W}$	Read/write signal

14.2 $\overline{RST}$ and $\overline{IRQ}$ Pins Initialization

$\overline{RST}$ and $\overline{IRQ}$ pins come out of reset as PTA3 and PTA2 respectively. $\overline{RST}$ and $\overline{IRQ}$ functions can be activated by programing CONFIG2 accordingly. Refer to Chapter 5 Configuration Register (CONFIG).

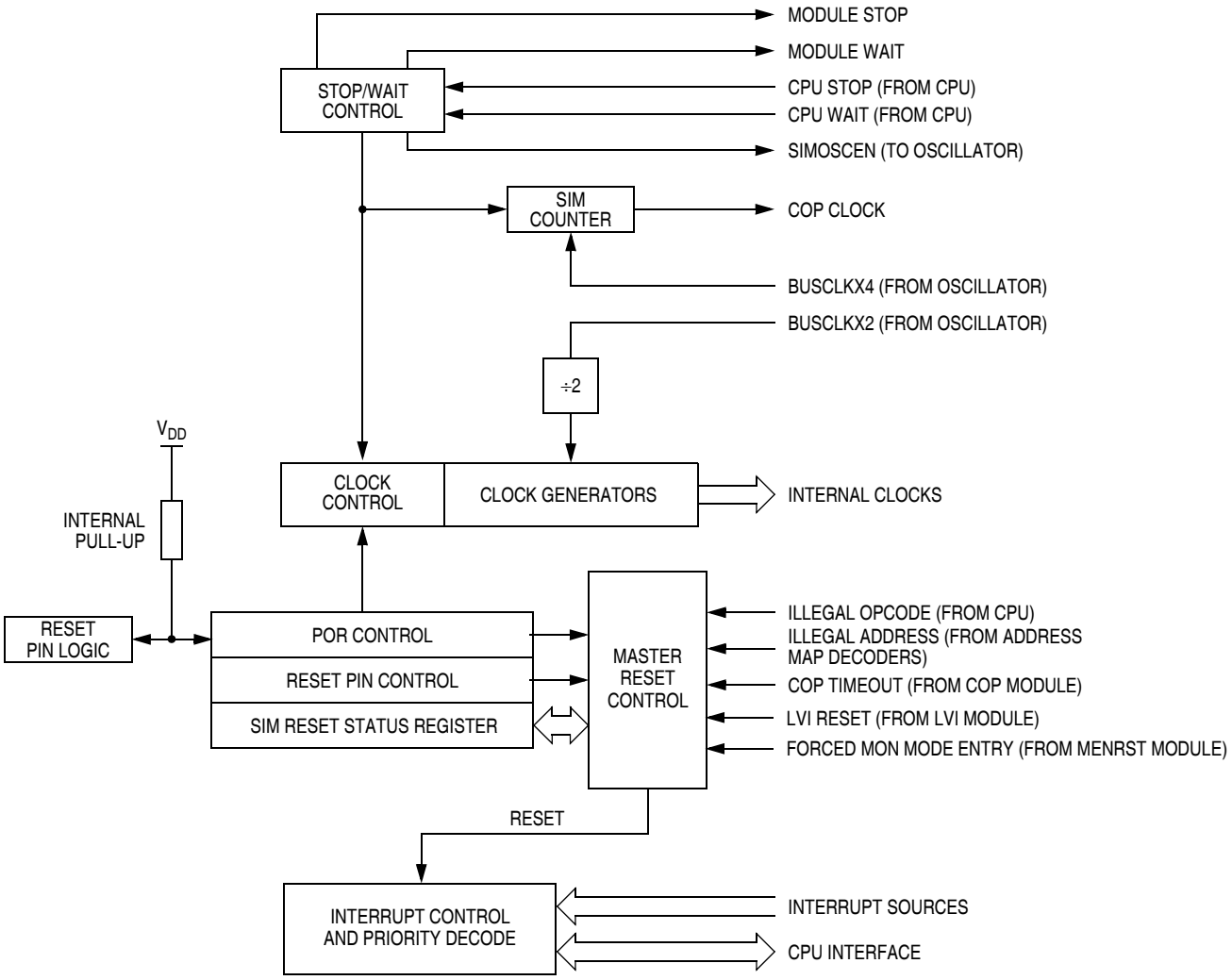


Figure 14-1. SIM Block Diagram

14.3 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, BUSCLKX2, as shown in Figure 14-2.

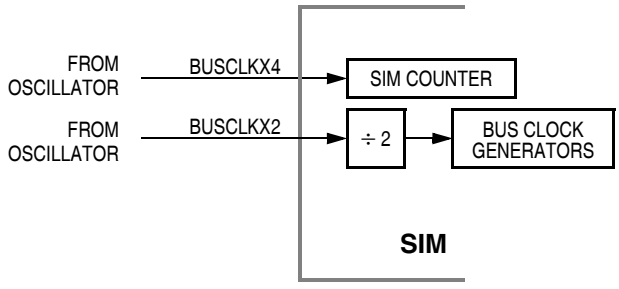


Figure 14-2. SIM Clock Signals

14.3.1 Bus Timing

In user mode, the internal bus frequency is the oscillator frequency (BUSCLKX4) divided by four.

14.3.2 Clock Start-Up from POR

When the power-on reset module generates a reset, the clocks to the CPU and peripherals are inactive and held in an inactive phase until after the 4096 BUSCLKX4 cycle POR time out has completed. The IBUS clocks start upon completion of the time out.

14.3.3 Clocks in Stop Mode and Wait Mode

Upon exit from stop mode by an interrupt or reset, the SIM allows BUSCLKX4 to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay time out. This time out is selectable as 4096 or 32 BUSCLKX4 cycles. See 14.7.2 Stop Mode.

In wait mode, the CPU clocks are inactive. The SIM also produces two sets of clocks for other modules. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

14.4 Reset and System Initialization

The MCU has these reset sources:

- Power-on reset module (POR)
- External reset pin ($\overline{RST}$)
- Computer operating properly module (COP)
- Low-voltage inhibit module (LVI)
- Illegal opcode
- Illegal address

All of these resets produce the vector $FFFE–FFFF ($FEFE–FEFF in monitor mode) and assert the internal reset signal (IRST). IRST causes all registers to be returned to their default values and all modules to be returned to their reset states.

An internal reset clears the SIM counter (see 14.5 SIM Counter), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). See 14.8 SIM Registers.

14.4.1 External Pin Reset

The $\overline{RST}$ pin circuits include an internal pullup device. Pulling the asynchronous $\overline{RST}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{RST}$ is held low for at least the minimum t_{RL} time. Figure 14-3 shows the relative timing. The $\overline{RST}$ pin function is only available if the RSTEN bit is set in the CONFIG2 register.

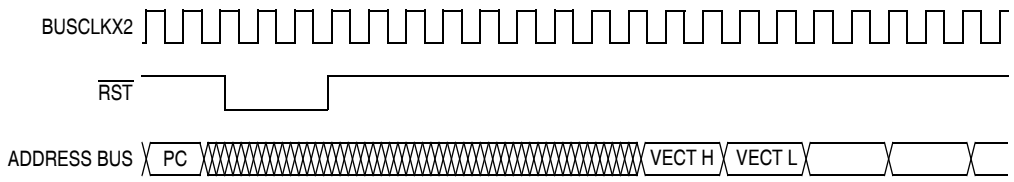


Figure 14-3. External Reset Timing

14.4.2 Active Resets from Internal Sources

The $\overline{\text{RST}}$ pin is initially setup as a general-purpose input after a POR. Setting the RSTEN bit in the CONFIG2 register enables the pin for the reset function. This section assumes the RSTEN bit is set when describing activity on the $\overline{\text{RST}}$ pin.

NOTE

For POR and LVI resets, the SIM cycles through 4096 BUSCLKX4 cycles. The internal reset signal then follows the sequence from the falling edge of $\overline{\text{RST}}$ shown in Figure 14-4.

The COP reset is asynchronous to the bus clock.

The active reset feature allows the part to issue a reset to peripherals and other chips within a system built around the MCU.

All internal reset sources actively pull the $\overline{\text{RST}}$ pin low for 32 BUSCLKX4 cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles (see Figure 14-4). An internal reset can be caused by an illegal address, illegal opcode, COP time out, LVI, or POR (see Figure 14-5).

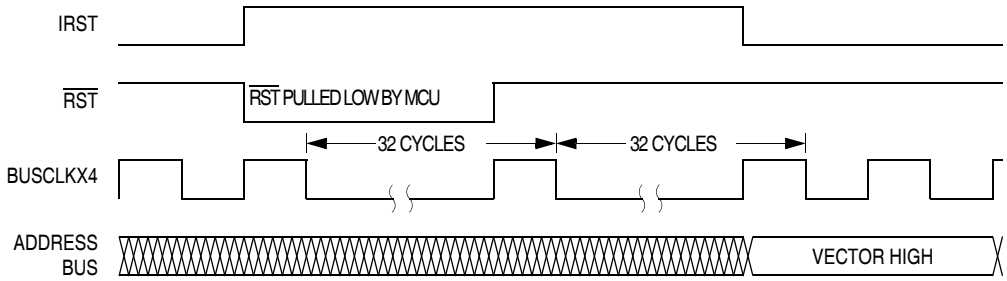


Figure 14-4. Internal Reset Timing

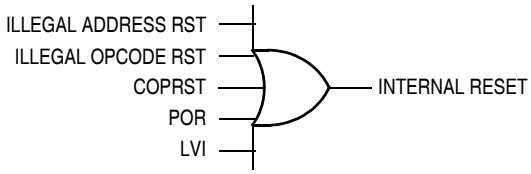


Figure 14-5. Sources of Internal Reset

Table 14-2. Reset Recovery Timing

Reset Recovery Type	Actual Number of Cycles
POR/LVI	4163 (4096 + 64 + 3)
All others	67 (64 + 3)

14.4.2.1 Power-On Reset

When power is first applied to the MCU, the power-on reset module (POR) generates a pulse to indicate that power on has occurred. The SIM counter counts out 4096 BUSCLKX4 cycles. Sixty-four BUSCLKX4 cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur.

At power on, the following events occur:

- A POR pulse is generated.
- The internal reset signal is asserted.
- The SIM enables the oscillator to drive BUSCLKX4.
- Internal clocks to the CPU and modules are held inactive for 4096 BUSCLKX4 cycles to allow stabilization of the oscillator.
- The POR bit of the SIM reset status register (SRSR) is set.

See Figure 14-6.

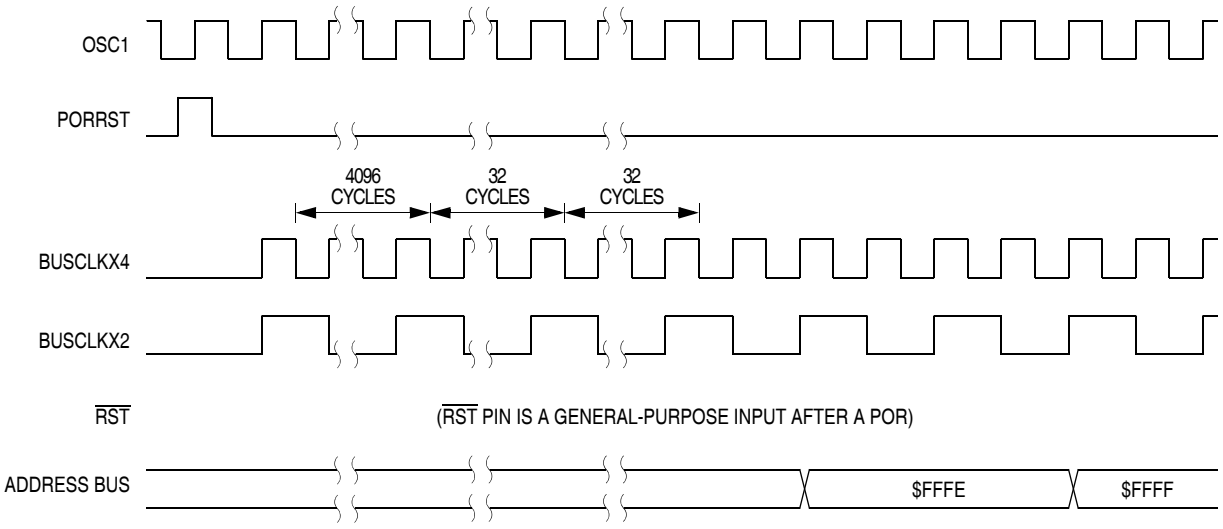


Figure 14-6. POR Recovery

14.4.2.2 Computer Operating Properly (COP) Reset

An input to the SIM is reserved for the COP reset signal. The overflow of the COP counter causes an internal reset and sets the COP bit in the SIM reset status register (SRSR). The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

To prevent a COP module time out, 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 ($2^{12} - 2^4$) BUSCLKX4 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 time out.

The COP module is disabled during a break interrupt with monitor mode when BDCOP bit is set in break auxiliary register (BRKAR).

14.4.2.3 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 mask option register is 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

14.4.2.4 Illegal Address Reset

An opcode fetch from an unmapped address generates an illegal address reset. The SIM verifies that the CPU is fetching an opcode prior to asserting the ILAD bit in the SIM reset status register (SRSR) and resetting the MCU. A data fetch from an unmapped address does not generate a reset. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources. See Figure 2-1. Memory Map for memory ranges.

14.4.2.5 Low-Voltage Inhibit (LVI) Reset

The LVI asserts its output to the SIM when the V_{DD} voltage falls to the LVI trip voltage V_{TRIPF}. The LVI bit in the SIM reset status register (SRSR) is set, and the external reset pin ($\overline{\text{RST}}$) is held low while the SIM counter counts out 4096 BUSCLKX4 cycles after V_{DD} rises above V_{TRIPR}. Sixty-four BUSCLKX4 cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur. The SIM actively pulls down the ($\overline{\text{RST}}$) pin for all internal reset sources.

14.5 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 uses 12 stages for counting, followed by a 13th stage that triggers a reset of SIM counters and supplies the clock for the COP module. The SIM counter is clocked by the falling edge of BUSCLKX4.

14.5.1 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 oscillator to drive the bus clock state machine.

14.5.2 SIM Counter During Stop Mode Recovery

The SIM counter also is used for stop mode recovery. The STOP instruction clears the SIM counter. After an interrupt, break, or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the configuration register 1 (CONFIG1). If the SSREC bit is a 1, then the stop recovery is reduced from the normal delay of 4096 BUSCLKX4 cycles down to 32 BUSCLKX4 cycles. This is ideal for applications using canned oscillators that do not require long start-up times from stop mode. External crystal applications should use the full stop recovery time, that is, with SSREC cleared in the configuration register 1 (CONFIG1).

14.5.3 SIM Counter and Reset States

External reset has no effect on the SIM counter (see 14.7.2 Stop Mode for details.) The SIM counter is free-running after all reset states. See 14.4.2 Active Resets from Internal Sources for counter control and internal reset recovery sequences.

14.6 Exception Control

Normal sequential program execution can be changed in three different ways:

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

14.6.1 Interrupts

An interrupt temporarily changes the sequence of program execution to respond to a particular event. Figure 14-7 flow charts the handling of system interrupts.

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared).

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. Figure 14-8 shows interrupt entry timing. Figure 14-9 shows interrupt recovery timing.

14.6.1.1 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 LDA instruction is executed.

The LDA opcode is prefetched by both the INT1 and INT2 return-from-interrupt (RTI) instructions. However, in the case of the INT1 RTI prefetch, this is a redundant operation.

NOTE

To maintain compatibility with the M6805 Family, the H register is not pushed on the stack during interrupt entry. If the interrupt service routine modifies the H register or uses the indexed addressing mode, software should save the H register and then restore it prior to exiting the routine.

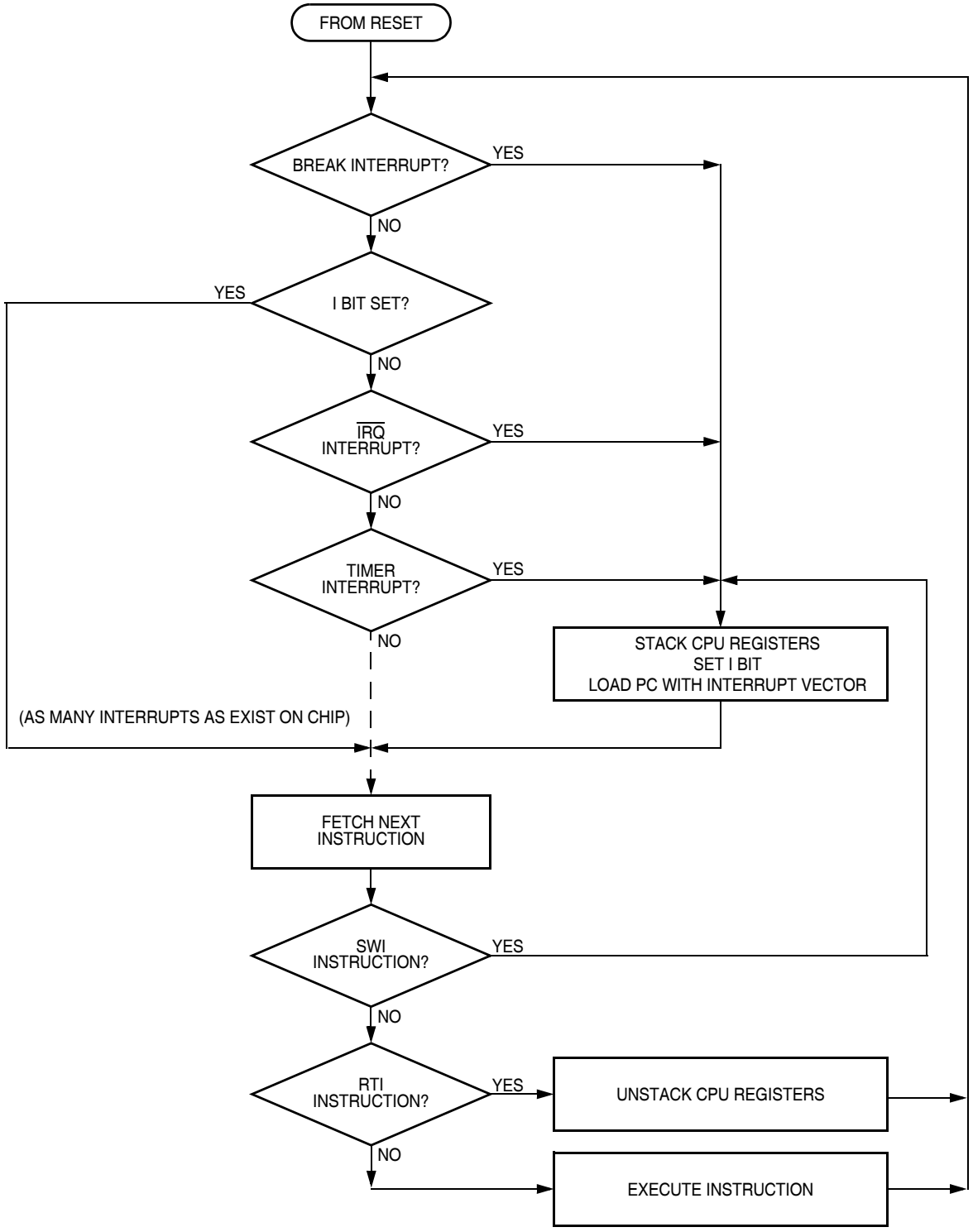


Figure 14-7. Interrupt Processing

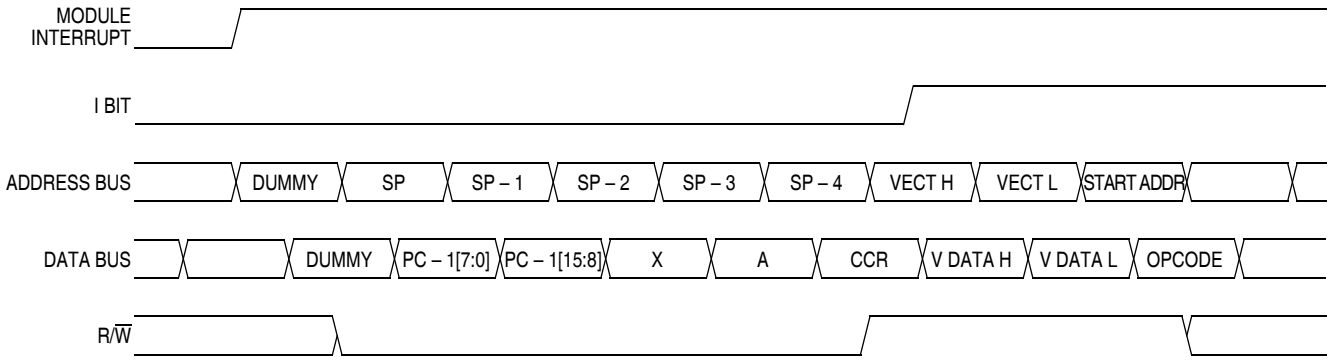


Figure 14-8. Interrupt Entry

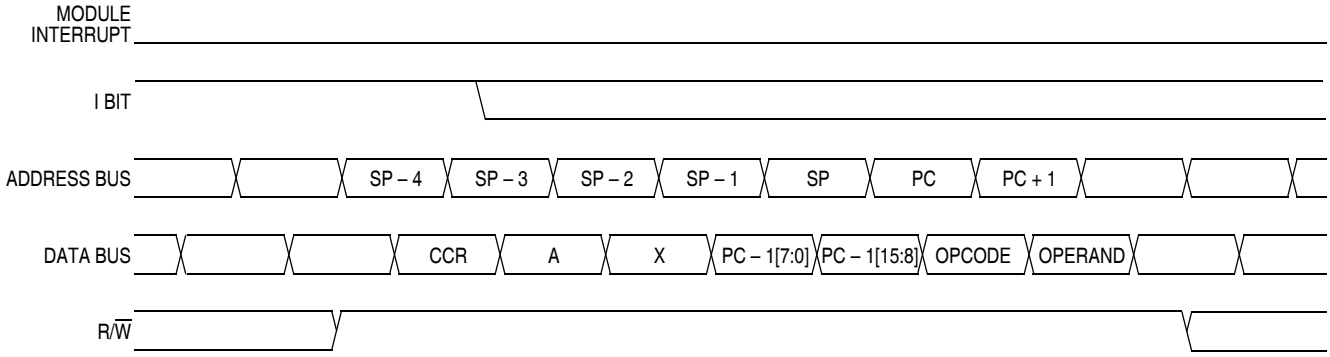


Figure 14-9. Interrupt Recovery

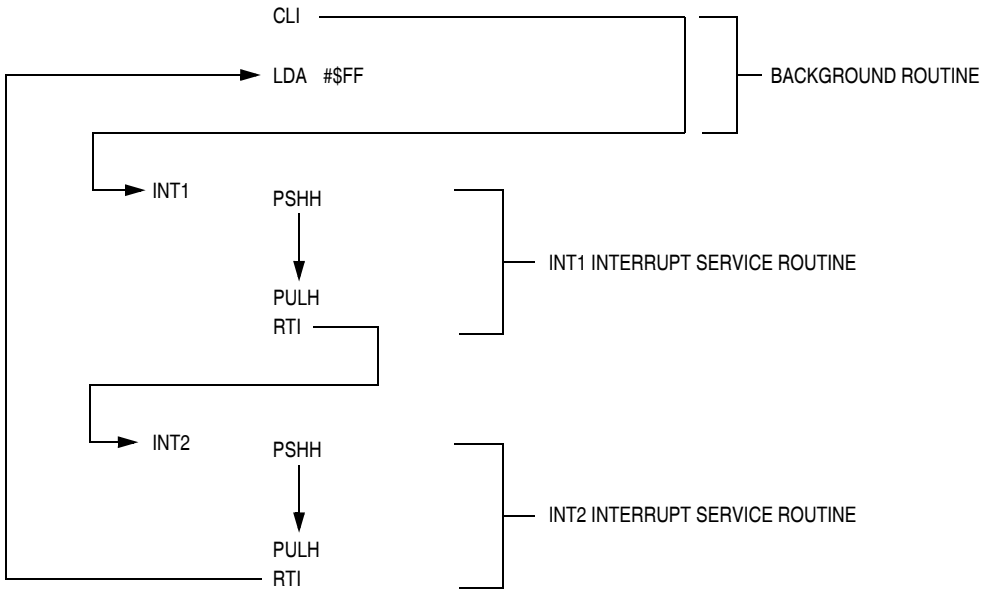


Figure 14-10. Interrupt Recognition Example

14.6.1.2 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.*

14.6.2 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	Source	Flag	Mask[1]	INT Register Flag	Vector Address
Highest	Reset	—	—	—	$FFFE–$FFFF
	SWI instruction	—	—	—	$FFFC–$FFFD
	$\overline{\text{IRQ}}$ pin	IRQF	IMASK	IF1	$FFFA–$FFFB
	Timer channel 0 interrupt	CH0F	CH0IE	IF3	$FFF6–$FFF7
	Timer channel 1 interrupt	CH1F	CH1IE	IF4	$FFF4–$FFF5
	Timer overflow interrupt	TOF	TOIE	IF5	$FFF2–$FFF3
	TIM channel 2 vector	CH2F	CH2IE	IF6	$FFF0–$FFF1
	TIM channel 3 vector	CH3F	CH3IE	IF7	$FFEE–$FFEF
	ESCI error vector	OR, HF, FE, PE	ORIE, NEIE, FEIE, PEIE	IF9	$FFEA–$FFEB
	ESCI receive vector	SCRF	SCRIE	IF10	$FFE8–$FFE9
	ESCI transmit vector	SCTE, TC	SCTIE, TCIE	IF11	$FFE6–$FFE7
	SPI receive	SPRF, OVRF, MODF	SPRIE, ERRIE	IF12	$FFE4–$FFE5
	SPI transmit	SPTE	SPTIE	IF13	$FFE2–$FFE3
	Keyboard interrupt	KEYF	IMASKK	IF14	$FFE0–$FFE1
Lowest	ADC conversion complete interrupt	COCO	AIEN	IF15	$FFDE–$FFDF

1. The I bit in the condition code register is a global mask for all interrupt sources except the SWI instruction.

14.6.2.1 Interrupt Status Register 1

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF6	IF5	IF4	IF3	0	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)

IF1 and IF3–IF6 — Interrupt Flags

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 0, 1, and 3— Always read 0

14.6.2.2 Interrupt Status Register 2

	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)

IF7–IF14 — Interrupt Flags

This flag indicates the presence of interrupt requests from the sources shown in Table 14-3.

1 = Interrupt request present
0 = No interrupt request present

14.6.2.3 Interrupt Status Register 3

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF22	IF21	IF20	IF19	IF18	IF17	IF16	IF15
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

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

IF15–IF22 — Interrupt Flags

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 Reset

All reset sources always have equal and highest priority and cannot be arbitrated.

14.6.4 Break Interrupts

The break module can stop normal program flow at a software programmable break point by asserting its break interrupt output. (See Chapter 17 Development Support.) 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.

14.6.5 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 break flag control register (BFCR).

Protecting flags in break mode ensures that set flags will not be cleared while in break mode. This protection allows registers to be freely read and written during break mode without losing status flag information.

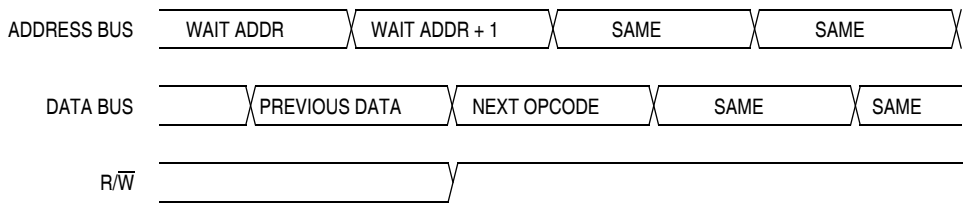
Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a two-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first 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. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

14.7.1 Wait Mode

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



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

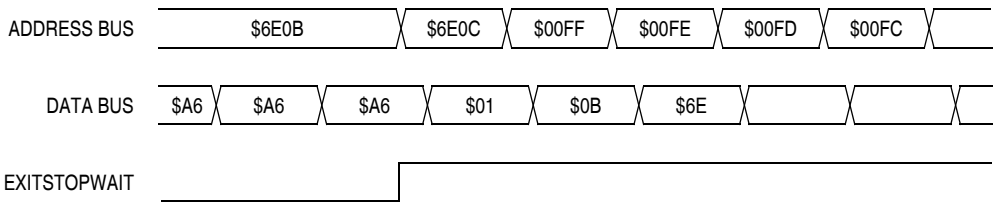
Figure 14-14. Wait Mode Entry Timing

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.

In wait mode, the CPU clocks are inactive. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

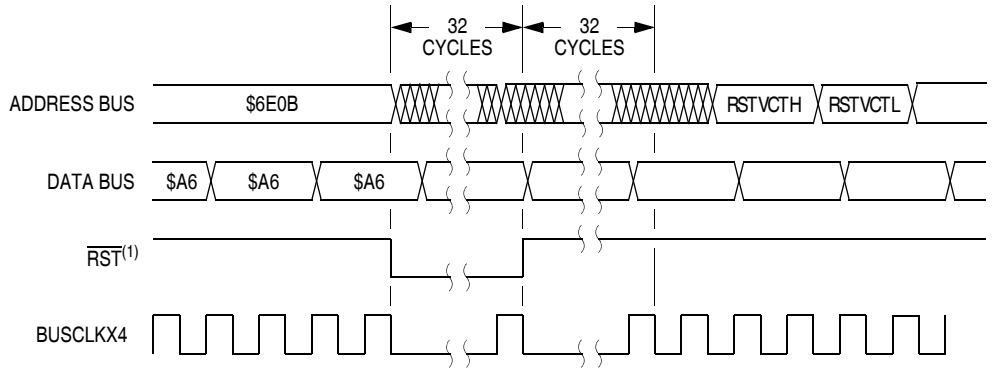
Wait mode can also be exited by a reset (or break in emulation mode). A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the break status register (BSR). If the COP disable bit, COPD, in the configuration register is 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.

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



NOTE: EXITSTOPWAIT = $\overline{\text{RST}}$ pin OR CPU interrupt

Figure 14-15. Wait Recovery from Interrupt



1. $\overline{\text{RST}}$ is only available if the RSTEN bit in the CONFIG2 register is set.

Figure 14-16. Wait Recovery from Internal Reset

14.7.2 Stop Mode

In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset or break also causes an exit from stop mode.

The SIM disables the oscillator signals (BUSCLKX2 and BUSCLKX4) in stop mode, stopping the CPU and peripherals. If OSCENINSTOP is set, BUSCLKX4 will remain running in STOP and can be used to run the AWU. Stop recovery time is selectable using the SSREC bit in the configuration register 1 (CONFIG1). If SSREC is set, stop recovery is reduced from the normal delay of 4096 BUSCLKX4 cycles down to 32. This is ideal for the internal oscillator, RC oscillator, and external oscillator options which do not require long start-up times from stop mode.

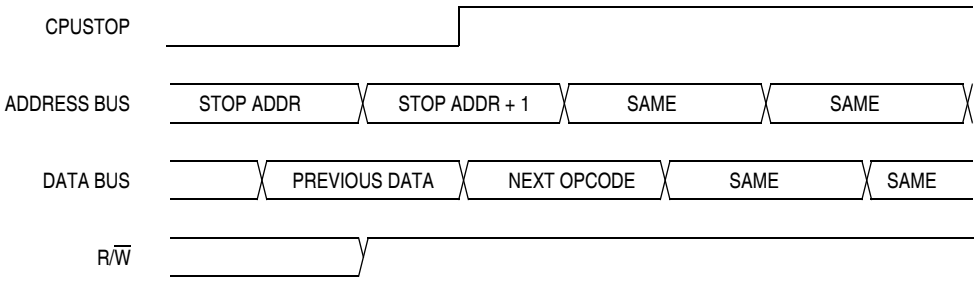
NOTE

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 and Figure 14-18 shows the stop mode recovery time from interrupt or break

NOTE

To minimize stop current, all pins configured as inputs should be driven to a logic 1 or logic 0.



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

Figure 14-17. Stop Mode Entry Timing

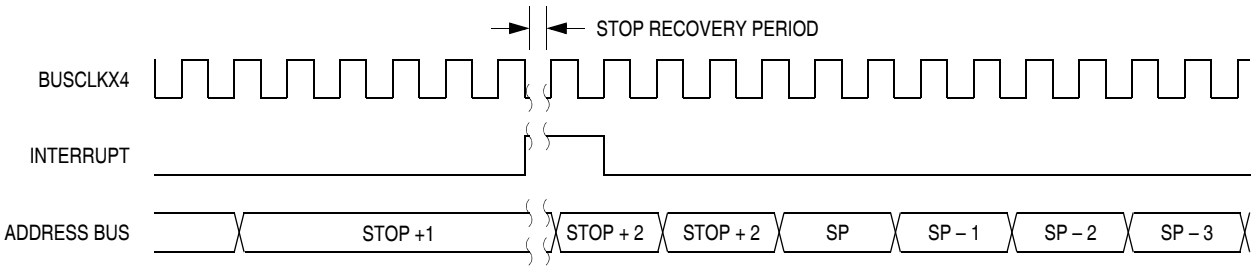


Figure 14-18. Stop Mode Recovery from Interrupt

14.8 SIM Registers

The SIM has two memory mapped registers.

14.8.1 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 SRSR. 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 bit can both be set if the power supply has a slow rise time.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	POR	PIN	COP	ILOP	ILAD	MODRST	LVI	0
Write:								
POR:	1	0	0	0	0	0	0	0

= Unimplemented

Figure 14-19. SIM Reset Status Register (SRSR)

POR — Power-On Reset Bit

1 = Last reset caused by POR circuit
0 = Read of SRSR

PIN — External Reset Bit

1 = Last reset caused by external reset pin ($\overline{RST}$)
0 = POR or read of SRSR

COP — Computer Operating Properly Reset Bit

1 = Last reset caused by COP counter
0 = POR or read of SRSR

ILOP — Illegal Opcode Reset Bit

1 = Last reset caused by an illegal opcode
0 = POR or read of SRSR

ILAD — Illegal Address Reset Bit (illegal attempt to fetch an opcode from an unimplemented address)

1 = Last reset caused by an opcode fetch from an illegal address
0 = POR or read of SRSR

MODRST — Monitor Mode Entry Module Reset Bit

1 = Last reset caused by monitor mode entry when vector locations $FFFE and $FFFF are $FF after POR while $\overline{IRQ} \neq V_{TST}$
0 = POR or read of SRSR

LVI — Low Voltage Inhibit Reset bit

1 = Last reset caused by LVI circuit
0 = POR or read of SRSR

14.8.2 Break Flag Control Register

The break control register (BFCR) contains a bit that enables software to clear status bits while the MCU is in a break state.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BCFE	R	R	R	R	R	R	R
Reset:	0							

R = Reserved

Figure 14-20. Break Flag Control Register (BFCR)

BCFE — Break Clear Flag Enable Bit

This read/write bit enables software to clear status bits by accessing status registers while the MCU is in a break state. To clear status bits during the break state, the BCFE bit must be set.

1 = Status bits clearable during break
0 = Status bits not clearable during break

Chapter 15
Serial Peripheral Interface (SPI) Module

15.1 Introduction

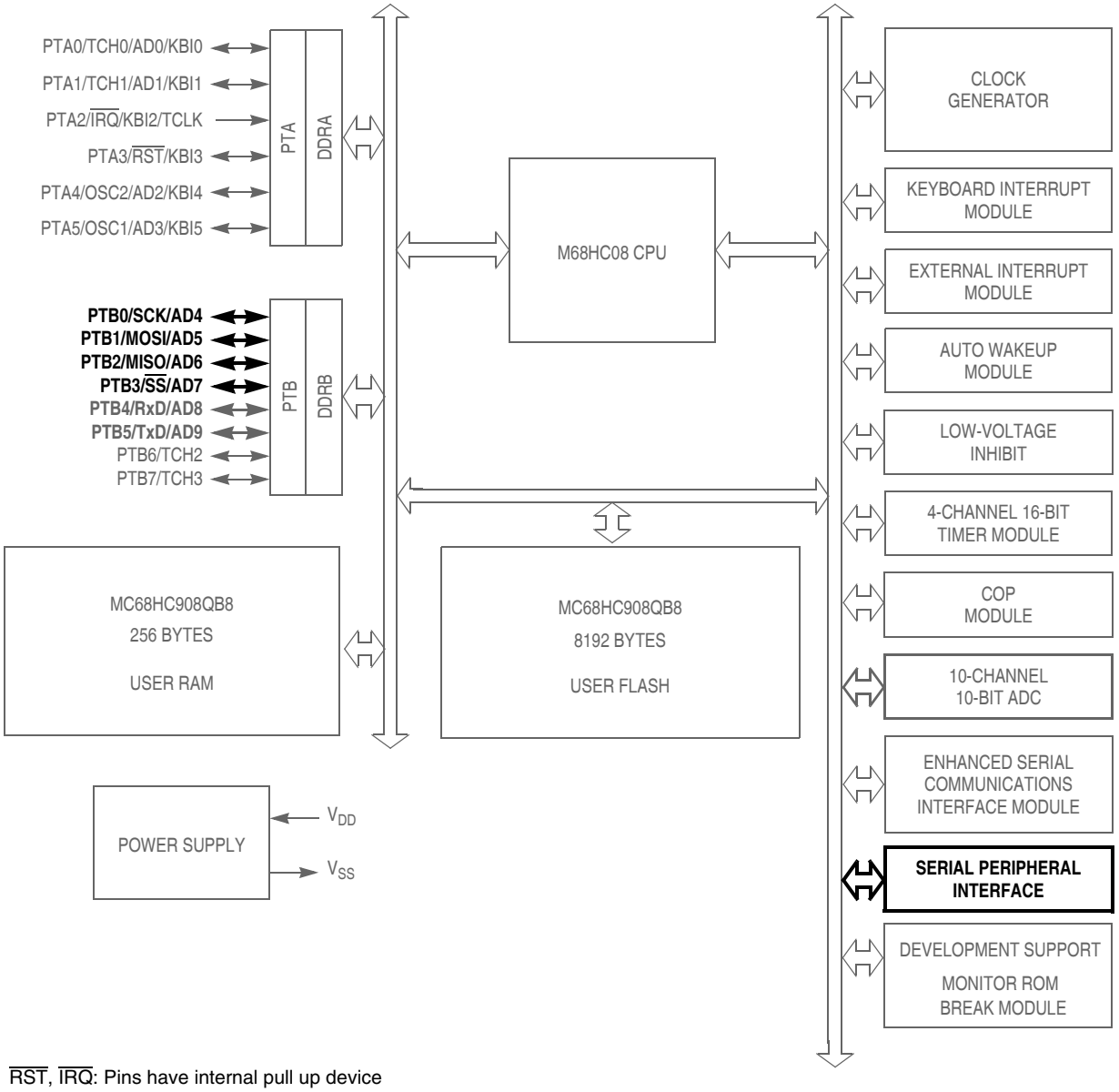
This section describes the serial peripheral interface (SPI) module, which allows full-duplex, synchronous, serial communications with peripheral devices.

The SPI shares its pins with general-purpose input/output (I/O) port pins. See Figure 15-1 for port location of these shared pins.

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:
  - SPRF (SPI receiver full)
  - SPTE (SPI transmitter empty)
- Mode fault error flag with interrupt capability
- Overflow error flag with interrupt capability
- Programmable wired-OR mode



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device
PTA[0:5]: Higher current sink and source capability

Figure 15-1. Block Diagram Highlighting SPI Block and Pins

15.3 Functional Description

The SPI module allows full-duplex, synchronous, serial communication between the MCU and peripheral devices, including other MCUs. Software can poll the SPI status flags or SPI operation can be interrupt driven.

The following paragraphs describe the operation of the SPI module.

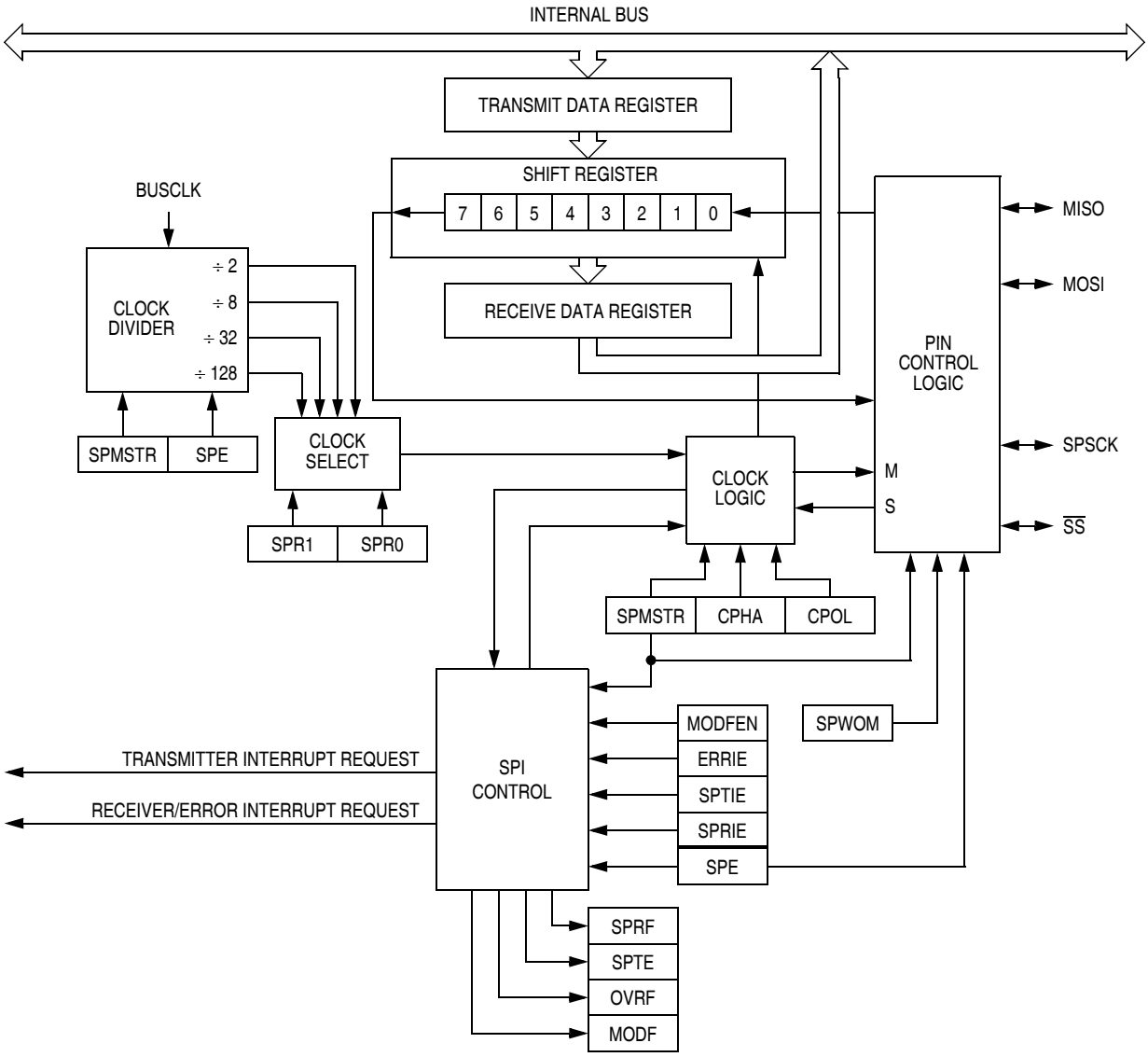


Figure 15-2. SPI Module Block Diagram

15.3.1 Master Mode

The SPI operates in master mode when the SPI master bit, SPMSTR, is set.

NOTE

In a multi-SPI system, configure the SPI modules as master or slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. See 15.8.1 SPI Control Register.

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the transmit data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The byte begins shifting out on the MOSI pin under the control of the serial clock. See Figure 15-3.

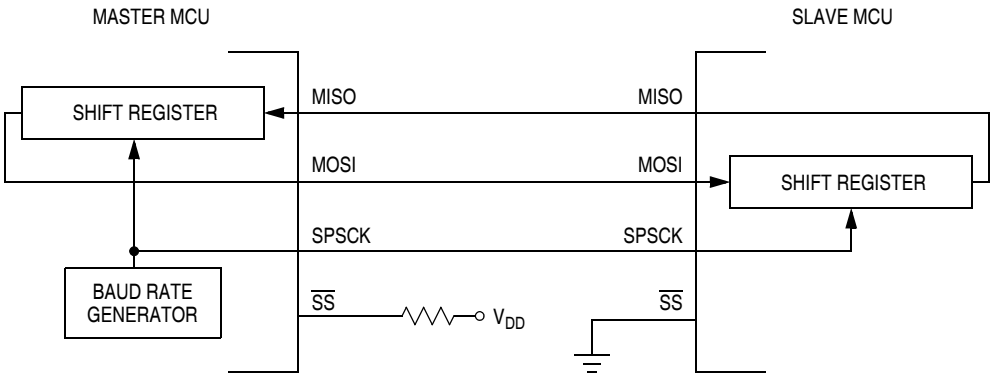


Figure 15-3. Full-Duplex Master-Slave Connections

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See 15.8.2 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.

While the byte shifts out on the MOSI pin of the master, another byte shifts in from the slave on the master's MISO pin. The transmission ends when the receiver full bit, SPRF, becomes set. At the same time that SPRF becomes set, the byte from the slave transfers to the receive data register. In normal operation, SPRF signals the end of a transmission. Software clears SPRF by reading the SPI status and control register (SPSCR) with SPRF set and then reading the SPI data register (SPDR). Writing to SPDR clears SPTE.

15.3.2 Slave Mode

The SPI operates in slave mode when SPMSTR is clear. In slave mode, the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be low. $\overline{SS}$ must remain low until the transmission is complete. See 15.3.6.2 Mode Fault Error.

In a slave SPI module, data enters the shift register under the control of the serial clock from the master SPI module. After a byte enters the shift register of a slave SPI, it transfers to the receive data register, and the SPRF bit is set. To prevent an overflow condition, slave software then must read the receive data register before another full byte enters the shift register.

The maximum frequency of the SPSCK for an SPI configured as a slave is the bus clock speed (which is twice as fast as the fastest master SPSCK clock that can be generated). The frequency of the SPSCK for an SPI configured as a slave does not have to correspond to any SPI baud rate. The baud rate only

controls the speed of the SPSCK generated by an SPI configured as a master. Therefore, the frequency of the SPSCK for an SPI configured as a slave can be any frequency less than or equal to the bus speed.

When the master SPI starts a transmission, the data in the slave shift register begins shifting out on the MISO pin. The slave can load its shift register with a new byte for the next transmission by writing to its transmit data register. The slave must write to its transmit data register at least one bus cycle before the master starts the next transmission. Otherwise, the byte already in the slave shift register shifts out on the MISO pin. Data written to the slave shift register during a transmission remains in a buffer until the end of the transmission.

When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transmission. See 15.3.3 Transmission Formats.

NOTE

SPSCK must be in the proper idle state before the slave is enabled to prevent SPSCK from appearing as a clock edge.

15.3.3 Transmission Formats

During an SPI transmission, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock synchronizes shifting and sampling on the two serial data lines. A slave select line allows selection of an individual slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. On a master SPI device, the slave select line can optionally be used to indicate multiple-master bus contention.

15.3.3.1 Clock Phase and Polarity Controls

Software can select any of four combinations of serial clock (SPSCK) phase and polarity using two bits in the SPI control register (SPCR). The clock polarity is specified by the CPOL control bit, which selects an active high or low clock and has no significant effect on the transmission format.

The clock phase (CPHA) control bit selects one of two fundamentally different transmission formats. The clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

NOTE

Before writing to the CPOL bit or the CPHA bit, disable the SPI by clearing the SPI enable bit (SPE).

15.3.3.2 Transmission Format When CPHA = 0

Figure 15-4 shows an SPI transmission in which CPHA = 0. The figure should not be used as a replacement for data sheet parametric information.

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. After the transmission begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the transmit data register and transferred to the shift register after the current transmission.

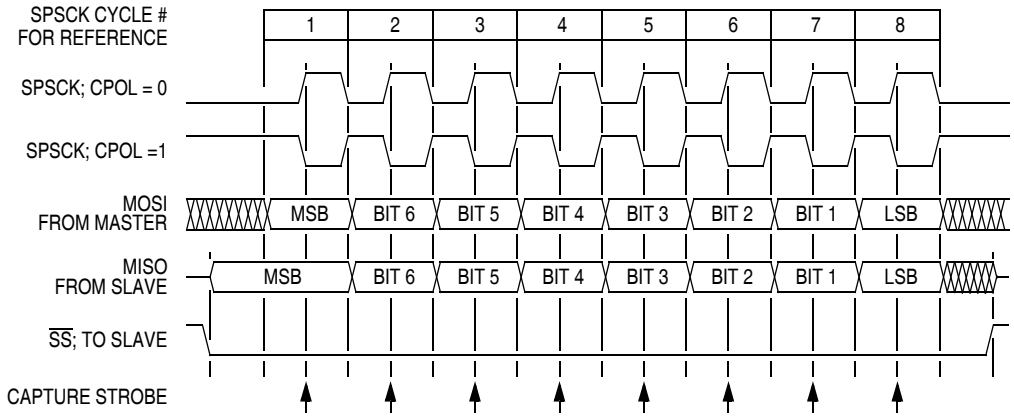


Figure 15-4. Transmission Format (CPHA = 0)

Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram because the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is low, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See 15.3.6.2 Mode Fault Error.) When CPHA = 0, the first SPSCK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SPSCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the slave data transmission. The slave's $\overline{SS}$ pin must be toggled back to high and then low again between each byte transmitted as shown in Figure 15-5.

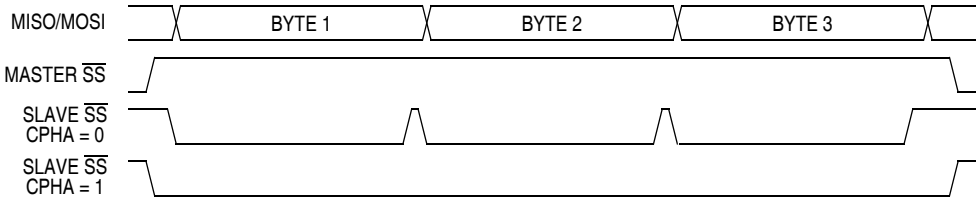


Figure 15-5. CPHA/$\overline{SS}$ Timing

15.3.3.3 Transmission Format When CPHA = 1

Figure 15-6 shows an SPI transmission in which CPHA = 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram because the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is low, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See 15.3.6.2 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.

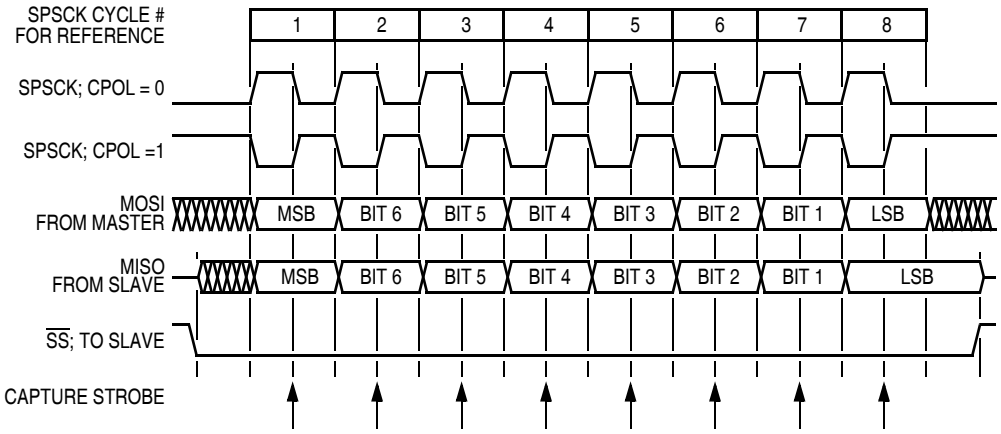


Figure 15-6. Transmission Format (CPHA = 1)

When CPHA = 1 for a slave, the first edge of the SPSCK indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. After the transmission begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the first edge of SPSCK. Any data written after the first edge is stored in the transmit data register and transferred to the shift register after the current transmission.

15.3.3.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), writing to the SPDR starts a transmission. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SPSCK signal. When CPHA = 0, the SPSCK signal remains inactive for the first half of the first SPSCK cycle. When CPHA = 1, the first SPSCK cycle begins with an edge on the SPSCK line from its inactive to its active level. The SPI clock rate (selected by SPR1:SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See Figure 15-7.) The internal SPI clock in the master is a free-running derivative of the internal MCU clock. To conserve power, it is enabled only when both the SPE and SPMSTR bits are set. Because the SPI clock is free-running, it is uncertain where the write to the SPDR occurs relative to the slower SPSCK. This uncertainty causes the variation in the initiation delay shown in Figure 15-7. This delay is no longer than a single SPI bit time. That is, the maximum delay is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

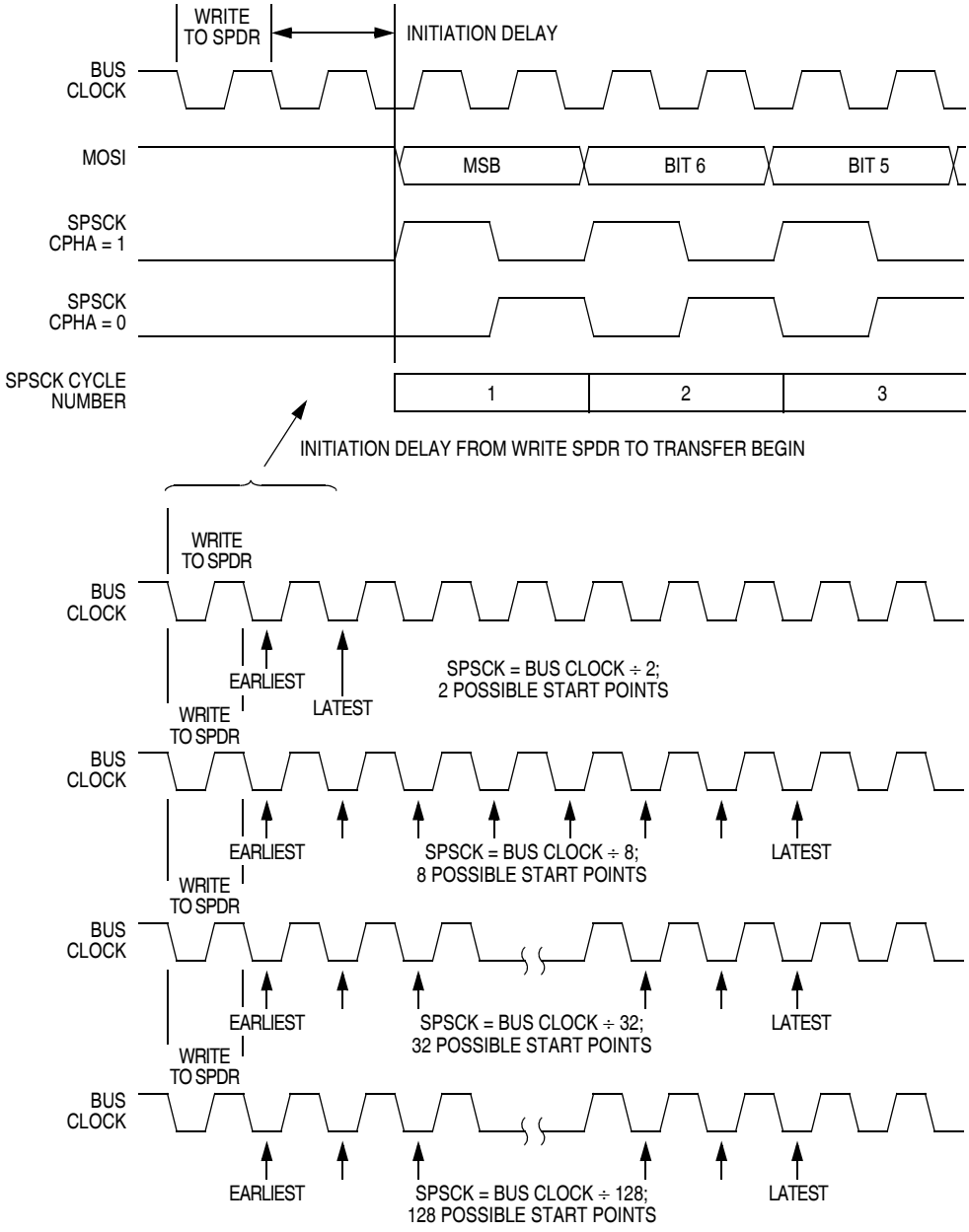


Figure 15-7. Transmission Start Delay (Master)

15.3.4 Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the transmit data register only when the SPTE bit is high. Figure 15-8 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA: CPOL = 1:0).

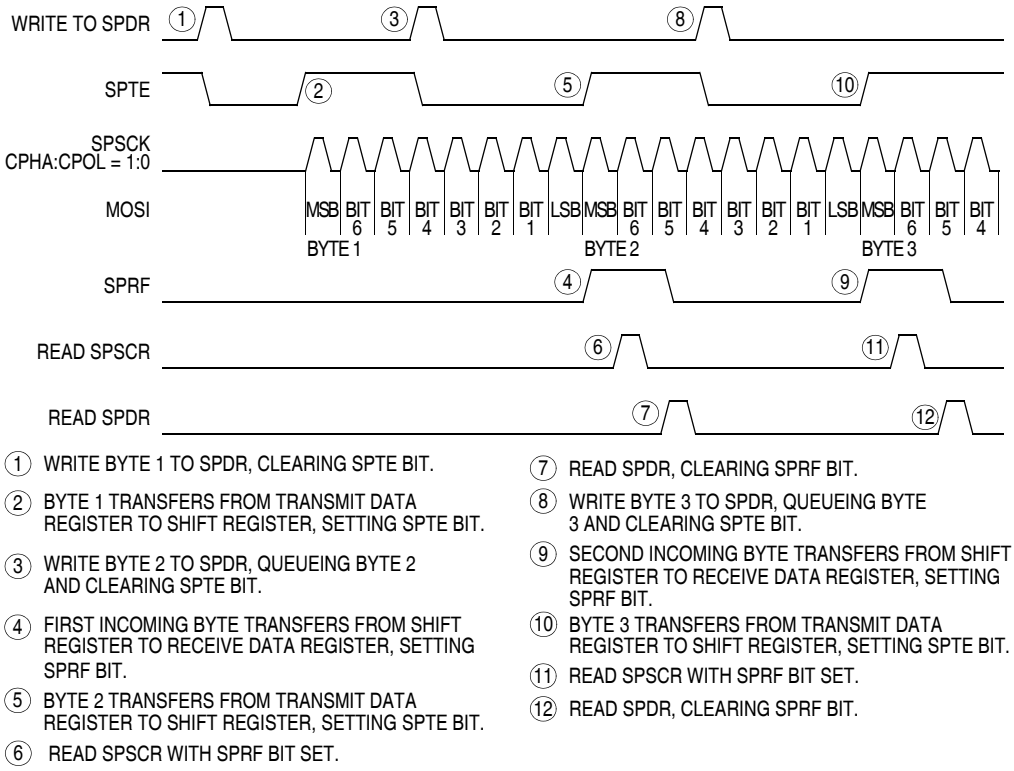


Figure 15-8. SPRF/SPTE interrupt Timing

The transmit data buffer allows back-to-back transmissions without the slave precisely timing its writes between transmissions as in a system with a single data buffer. Also, if no new data is written to the data buffer, the last value contained in the shift register is the next data word to be transmitted.

For an idle master or idle slave that has no data loaded into its transmit buffer, the SPTE is set again no more than two bus cycles after the transmit buffer empties into the shift register. This allows the user to queue up a 16-bit value to send. For an already active slave, the load of the shift register cannot occur until the transmission is completed. This implies that a back-to-back write to the transmit data register is not possible. SPTE indicates when the next write can occur.

15.3.5 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is 0. Whenever SPE is 0, the following occurs:

- The SPTE flag is set.
- Any transmission currently in progress is aborted.
- The shift register is cleared.
- The SPI state counter is cleared, making it ready for a new complete transmission.
- All the SPI pins revert back to being general-purpose I/O.

These items are reset only by a system reset:

- All control bits in the SPCR register
- All control bits in the SPSCR register (MODFEN, ERRIE, SPR1, and SPR0)
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, the user can clear SPE between transmissions without having to set all control bits again when SPE is set high for the next transmission.

By not resetting the SPRF, OVRF, and MODF flags, the user can still service these interrupts after the SPI has been disabled. The user can disable the SPI by writing 0 to the SPE bit. The SPI can also be disabled by a mode fault occurring in an SPI that was configured as a master with the MODFEN bit set.

15.3.6 Error Conditions

The following flags signal SPI error conditions:

- Overflow (OVRF) — Failing to read the SPI data register before the next full byte enters the shift register sets the OVRF bit. The new byte does not transfer to the receive data register, and the unread byte still can be read. OVRF is in the SPI status and control register.
- Mode fault error (MODF) — The MODF bit indicates that the voltage on the slave select pin ($\overline{SS}$) is inconsistent with the mode of the SPI. MODF is in the SPI status and control register.

15.3.6.1 Overflow Error

The overflow flag (OVRF) becomes set if the receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. The bit 1 capture strobe occurs in the middle of SPSCK cycle 7 (see Figure 15-4 and Figure 15-6.) If an overflow occurs, all data received after the overflow and before the OVRF bit is cleared does not transfer to the receive data register and does not set the SPI receiver full bit (SPRF). The unread data that transferred to the receive data register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPI status and control register and then reading the SPI data register.

OVRF generates a receiver/error interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF interrupts share the same interrupt vector (see Figure 15-11.) It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

If the SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. Figure 15-9 shows how it is possible to miss an overflow. The first part of Figure 15-9 shows how it is possible to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF bit can be set in between the time that SPSCR and SPDR are read.

In this case, an overflow can be missed easily. Because no more SPRF interrupts can be generated until this OVRF is serviced, it is not obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR following the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions can set the SPRF bit. Figure 15-10 illustrates this process. Generally, to avoid this second SPSCR read, enable OVRF by setting the ERRIE bit.

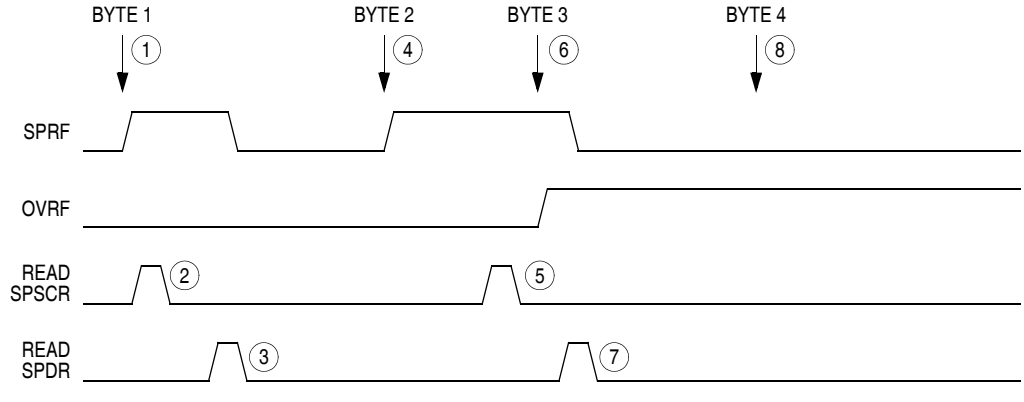


Figure 15-9. Missed Read of Overflow Condition

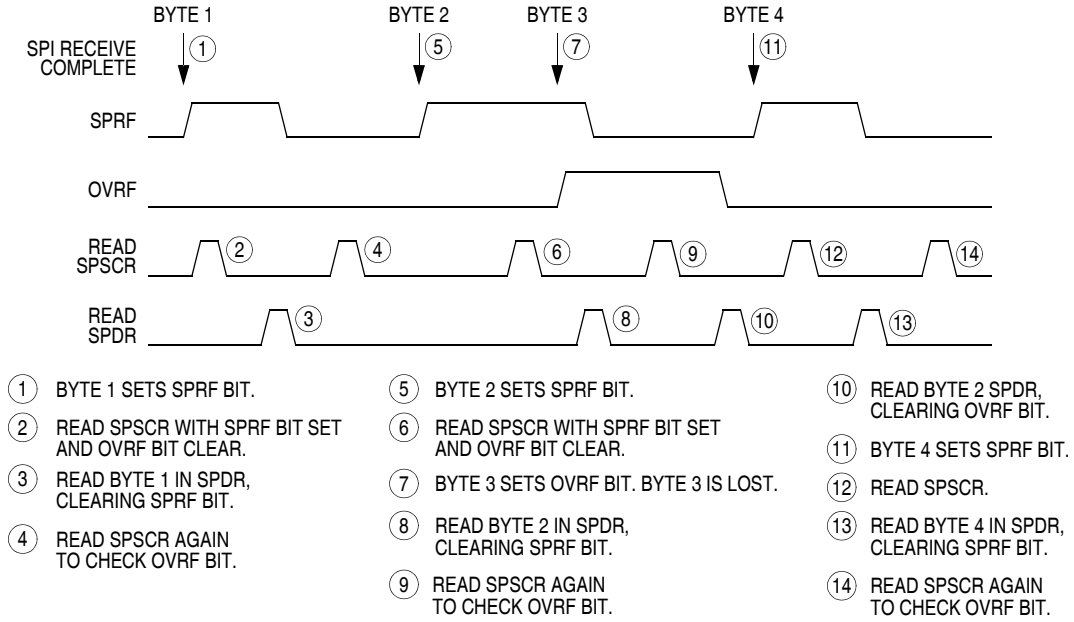


Figure 15-10. Clearing SPRF When OVRF Interrupt Is Not Enabled

15.3.6.2 Mode Fault Error

Setting SPMSTR selects master mode and configures the SPSCK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SPSCK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, becomes set any time the state of the slave select pin, $\overline{SS}$, is inconsistent with the mode selected by SPMSTR.

To prevent SPI pin contention and damage to the MCU, a mode fault error occurs if:

- The $\overline{SS}$ pin of a slave SPI goes high during a transmission
- The $\overline{SS}$ pin of a master SPI goes low at any time

For the MODF flag to be set, the mode fault error enable bit (MODFEN) must be set. Clearing the MODFEN bit does not clear the MODF flag but does prevent MODF from being set again after MODF is cleared.

MODF generates a receiver/error interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF interrupts share the same interrupt vector. (See Figure 15-11.) It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes low. A mode fault in a master SPI causes the following events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error interrupt request.
- The SPE bit is cleared.
- The SPTE bit is set.
- The SPI state counter is cleared.
- The data direction register of the shared I/O port regains control of port drivers.

NOTE

To prevent bus contention with another master SPI after a mode fault error, clear all SPI bits of the data direction register of the shared I/O port before enabling the SPI.

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{SS}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{SS}$ goes low and ends after the incoming SPSCK goes to its idle level following the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SPSCK returns to its idle level following the shift of the last data bit. See 15.3.3 Transmission Formats.

NOTE

Setting the MODF flag does not clear the SPMSTR bit. SPMSTR has no function when SPE = 0. Reading SPMSTR when MODF = 1 shows the difference between a MODF occurring when the SPI is a master and when it is a slave.

When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is low) and later unselected ($\overline{SS}$ is high) even if no SPSCK is sent to that slave. This happens because $\overline{SS}$ low indicates the start of the transmission (MISO driven out with the value of MSB) for CPHA = 0. When CPHA = 1, a slave can be selected and then later unselected with no transmission occurring. Therefore, MODF does not occur because a transmission was never begun.

In a slave SPI (MSTR = 0), MODF generates an SPI receiver/error interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transmission by clearing the SPE bit of the slave.

NOTE

A high on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.

To clear the MODF flag, read the SPSCR with the MODF bit set and then write to the SPCR register. This entire clearing mechanism must occur with no MODF condition existing or else the flag is not cleared.

15.4 Interrupts

Four SPI status flags can be enabled to generate interrupt requests. See Table 15-1.

Table 15-1. SPI Interrupts

Flag	Request
SPTE Transmitter empty	SPI transmitter interrupt request (SPTIE = 1, SPE = 1)
SPRF Receiver full	SPI receiver interrupt request (SPRIE = 1)
OVRF Overflow	SPI receiver/error interrupt request (ERRIE = 1)
MODF Mode fault	SPI receiver/error interrupt request (ERRIE = 1)

Reading the SPI status and control register with SPRF set and then reading the receive data register clears SPRF. The clearing mechanism for the SPTE flag is requires only a write to the transmit data register.

The SPI transmitter interrupt enable bit (SPTIE) enables the SPTE flag to generate transmitter interrupt requests, provided that the SPI is enabled (SPE = 1).

The SPI receiver interrupt enable bit (SPRIE) enables SPRF to generate receiver interrupt requests, regardless of the state of SPE. See Figure 15-11.

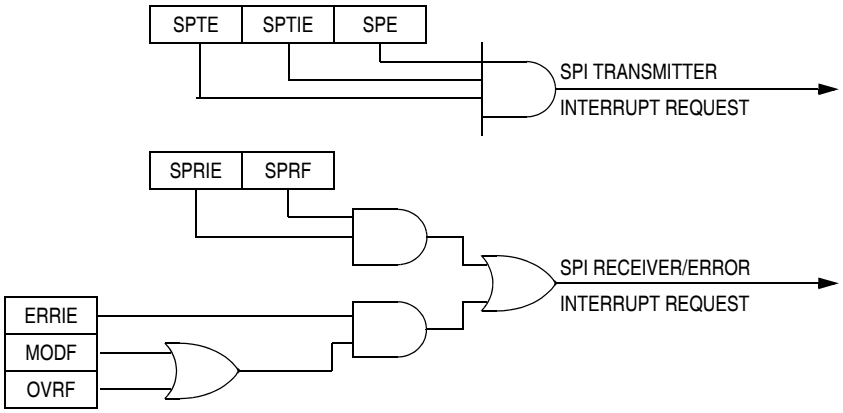


Figure 15-11. SPI Interrupt Request Generation

The error interrupt enable bit (ERRIE) enables both the MODF and OVRF bits to generate a receiver/error interrupt request.

The mode fault enable bit (MODFEN) can prevent the MODF flag from being set so that only the OVRF bit is enabled by the ERRIE bit to generate receiver/error interrupt requests.

The following sources in the SPI status and control register can generate interrupt requests:

- SPI receiver full bit (SPRF) — SPRF becomes set every time a byte transfers from the shift register to the receive data register. If the SPI receiver interrupt enable bit, SPRIE, is also set, SPRF generates an SPI receiver/error interrupt request.
- SPI transmitter empty bit (SPTE) — SPTE becomes set every time a byte transfers from the transmit data register to the shift register. If the SPI transmit interrupt enable bit, SPTIE, is also set, SPTE generates an SPTE interrupt request.

15.5 Low-Power Modes

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

15.5.1 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 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 interrupt requests by setting the error interrupt enable bit (ERRIE). See 15.4 Interrupts.

15.5.2 Stop Mode

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

15.6 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 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.

15.7 I/O Signals

The SPI module can share its pins with the general-purpose I/O pins. See Figure 15-1 for the port pins that are shared.

The SPI module has four I/O pins:

- MISO — Master input/slave output
- MOSI — Master output/slave input
- SPSCK — Serial clock
- $\overline{SS}$ — Slave select

15.7.1 MISO (Master In/Slave Out)

MISO is one of the two SPI module pins that transmits serial data. In full duplex operation, the MISO pin of the master SPI module is connected to the MISO pin of the slave SPI module. The master SPI simultaneously receives data on its MISO pin and transmits data from its MOSI pin.

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is 0 and its $\overline{SS}$ pin is low. To support a multiple-slave system, a high on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

When enabled, the SPI controls data direction of the MISO pin regardless of the state of the data direction register of the shared I/O port.

15.7.2 MOSI (Master Out/Slave In)

MOSI is one of the two SPI module pins that transmits serial data. In full-duplex operation, the MOSI pin of the master SPI module is connected to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin.

When enabled, the SPI controls data direction of the MOSI pin regardless of the state of the data direction register of the shared I/O port.

15.7.3 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.

15.7.4 $\overline{SS}$ (Slave Select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, $\overline{SS}$ is used to select a slave. For CPHA = 0, the $\overline{SS}$ is used to define the start of a transmission. (See 15.3.3 Transmission Formats.) Because it is used to indicate the start of a transmission, $\overline{SS}$ must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low between transmissions for the CPHA = 1 format. See Figure 15-12.

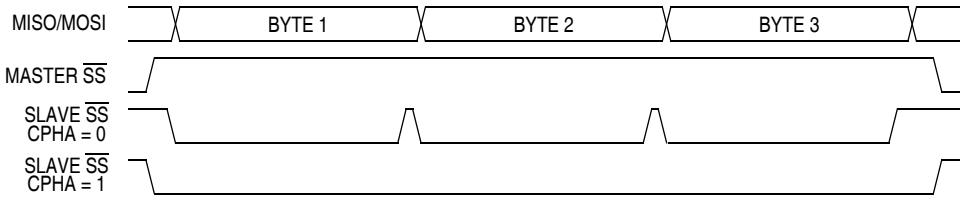


Figure 15-12. CPHA/SS Timing

When an SPI is configured as a slave, the 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 SS from creating a MODF error. See 15.8.2 SPI Status and Control Register.

NOTE

A high on the SS pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.

When an SPI is configured as a master, the SS input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. (See 15.3.6.2 Mode Fault Error.) For the state of the SS pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is 0 for an SPI master, the SS pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. When MODFEN is 1, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

User software can read the state of the SS pin by configuring the appropriate pin as an input and reading the port data register. See Table 15-2.

Table 15-2. SPI Configuration

SPE	SPMSTR	MODFEN	SPI Configuration	Function of SS Pin
0	X[1]	X	Not enabled	General-purpose I/O; SS ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	General-purpose I/O; SS ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

1. X = Don't care

15.8 Registers

The following registers allow the user to control and monitor SPI operation:

- SPI control register (SPCR)
- SPI status and control register (SPSCR)
- SPI data register (SPDR)

15.8.1 SPI Control Register

The SPI control register:

- Enables SPI module interrupt requests
- Configures the SPI module as master or slave
- Selects serial clock polarity and phase
- Configures the SPSCK, MOSI, and MISO pins as open-drain outputs
- Enables the SPI module

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
Reset:	0	0	1	0	1	0	0	0

R = Reserved

Figure 15-13. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

This read/write bit enables 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.

1 = SPRF interrupt requests enabled
0 = SPRF interrupt requests disabled

SPMSTR — SPI Master Bit

This read/write bit selects master mode operation or slave mode operation.

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 15-6.) To transmit data between SPI modules, the SPI modules must have identical CPOL values.

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 15-6.) To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the $\overline{SS}$ pin of the slave SPI module must be high between bytes. (See Figure 15-12.)

SPWOM — SPI Wired-OR Mode Bit

This read/write bit configures pins SPSCK, MOSI, and MISO so that these pins become open-drain outputs.

1 = Wired-OR SPSCK, MOSI, and MISO pins
0 = Normal push-pull SPSCK, MOSI, and MISO pins

SPE — SPI Enable

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. (See 15.3.5 Resetting the SPI.)

1 = SPI module enabled
0 = SPI module disabled

SPTIE— SPI Transmit Interrupt Enable

This read/write bit enables interrupt requests generated by the SPTE bit. SPTE is set when a byte transfers from the transmit data register to the shift register.

1 = SPTE interrupt requests enabled
0 = SPTE interrupt requests disabled

15.8.2 SPI Status and Control Register

The SPI status and control register contains flags to signal these conditions:

- Receive data register full
- Failure to clear SPRF bit before next byte is received (overflow error)
- Inconsistent logic level on $\overline{SS}$ pin (mode fault error)
- Transmit data register empty

The SPI status and control register also contains bits that perform these functions:

- Enable error interrupts
- Enable mode fault error detection
- Select master SPI baud rate

	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 15-14. 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 interrupt request if the SPRIE bit in the SPI control register is set also. During an SPRF interrupt, user software can clear SPRF by reading the SPI status and control register with SPRF set followed by a read of the SPI data register.

1 = Receive data register full
0 = Receive data register not full

ERRIE — Error Interrupt Enable Bit

This read/write bit enables the MODF and OVRF bits to generate interrupt requests.

1 = MODF and OVRF can generate interrupt requests
0 = MODF and OVRF cannot generate interrupt requests

OVRF — Overflow Bit

This clearable, read-only flag is set if software does not read the byte in the receive data register before the next full byte enters the shift register. In an overflow condition, the byte already in the receive data register is unaffected, and the byte that shifted in last is lost. Clear the OVRF bit by reading the SPI status and control register with OVRF set and then reading the receive data register.

1 = Overflow
0 = No overflow

MODF — Mode Fault Bit

This clearable, read-only flag is set in a slave SPI if the $\overline{SS}$ pin goes high during a transmission with MODFEN set. In a master SPI, the MODF flag is set if the $\overline{SS}$ pin goes low at any time with the MODFEN bit set. Clear MODF by reading the SPI status and control register (SPSCR) with MODF set and then writing to the SPI control register (SPCR).

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 interrupt request if the SPTIE bit in the SPI control register is also set.

NOTE

Do not write to the SPI data register unless SPTE is high.

During an SPTE interrupt, user software can clear SPTE by writing to the transmit data register.

1 = Transmit data register empty
0 = Transmit data register not empty

MODFEN — Mode Fault Enable Bit

This read/write bit, when set, allows the MODF flag to be set. If the MODF flag is set, clearing MODFEN does not clear the MODF flag. If the SPI is enabled as a master and the MODFEN bit is 0, then the $\overline{SS}$ pin is available as a general-purpose I/O.

If the MODFEN bit is 1, then 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 15.7.4 SS (Slave Select).

If the MODFEN bit is 0, the level of the $\overline{SS}$ pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation. See 15.3.6.2 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.

Table 15-3. SPI Master Baud Rate Selection

SPR1 and SPR0	Baud Rate Divisor (BD)
00	2
01	8
10	32
11	128

Use this formula to calculate the SPI baud rate:

$$\text{Baud rate} = \frac{\text{BUSCLK}}{\text{BD}}$$

15.8.3 SPI Data Register

The SPI data register consists of the read-only receive data register and the write-only transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate registers that can contain different values. See Figure 15-2.

	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 15-15. 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 because the register read is not the same as the register written.

Chapter 16
Timer Interface Module (TIM)

16.1 Introduction

This section describes the timer interface module (TIM). The TIM module is a 4-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions.

The TIM module shares its pins with general-purpose input/output (I/O) port pins. See Figure 16-1 for port location of these shared pins.

16.2 Features

Features include the following:

- Four 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 output compare pulse-width modulation (PWM) signal generation
- Programmable clock input
  - 7-frequency internal bus clock prescaler selection
  - External clock input pin if available, See Figure 16-1
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- Counter stop and reset bits

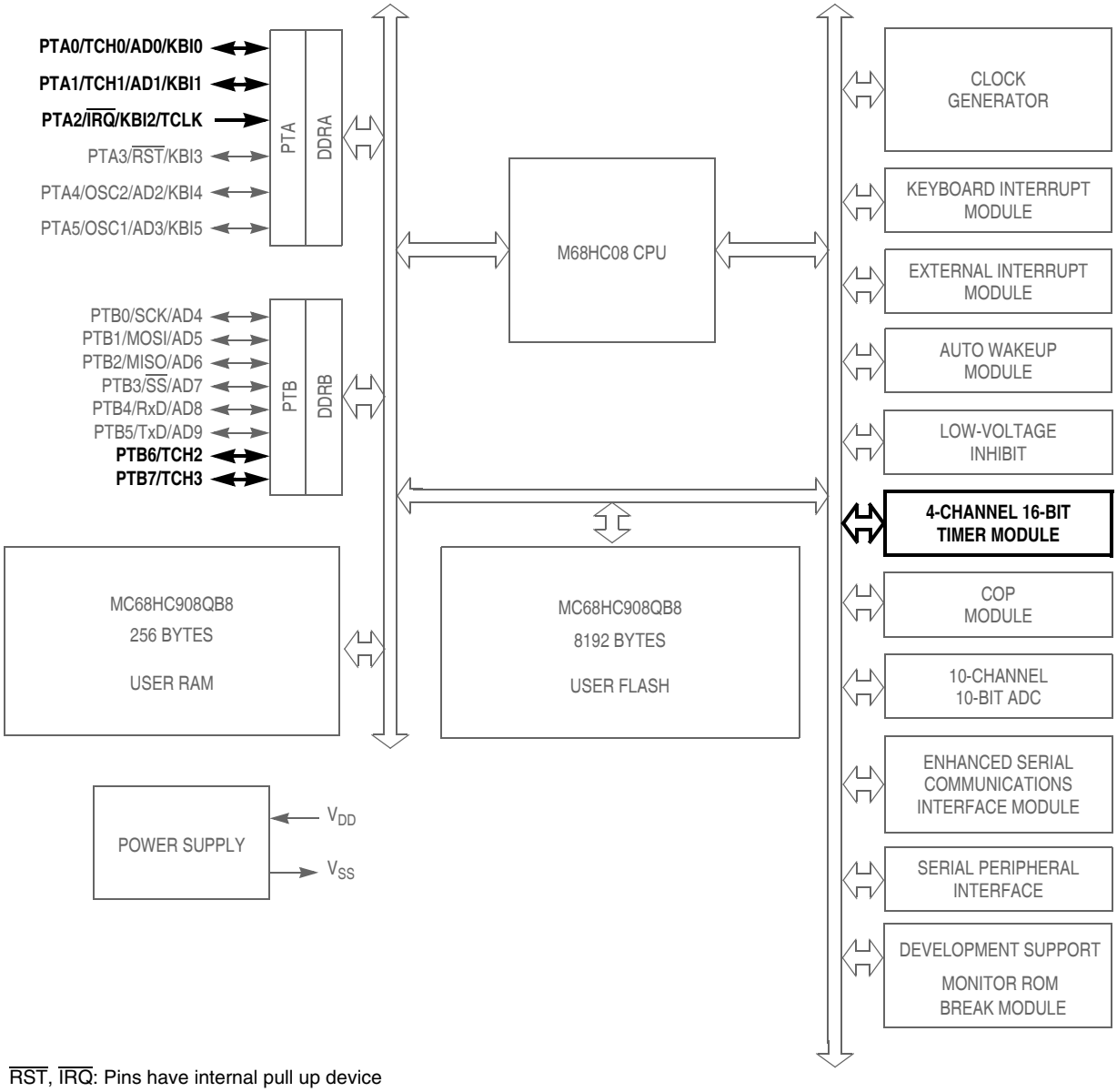
16.3 Functional Description

Figure 16-2 shows the structure of the TIM. The central component of the TIM is the 16-bit counter that can operate as a free-running counter or a modulo up-counter. The counter provides the timing reference for the input capture and output compare functions. The TIM counter modulo registers, TMODH:TMODL, control the modulo value of the counter. Software can read the counter value, TCNTH:TCNTL, at any time without affecting the counting sequence.

The four TIM channels are programmable independently as input capture or output compare channels.

16.3.1 TIM Counter Prescaler

The TIM clock source is one of the seven prescaler outputs or the external clock input pin, TCLK if available. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIM status and control register (TSC) select the clock source.



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device
PTA[0:5]: Higher current sink and source capability

Figure 16-1. Block Diagram Highlighting TIM Block and Pins

16.3.2 Input Capture

With the input capture function, the TIM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TIM latches the contents of the counter into the TIM channel registers, TCHxH:TCHxL. The polarity of the active edge is programmable. Input captures can be enabled to generate interrupt requests.

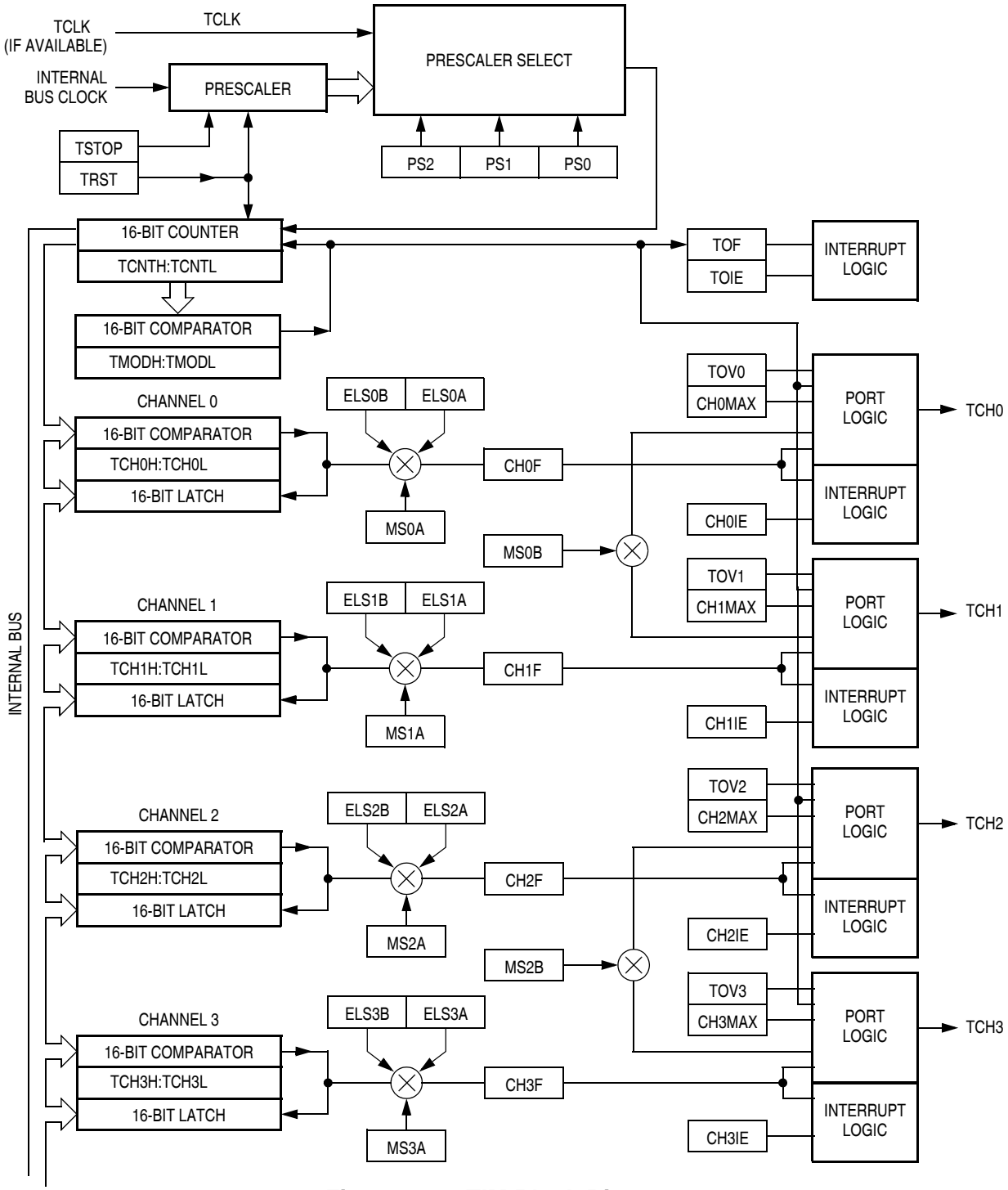


Figure 16-2. TIM Block Diagram

16.3.3 Output Compare

With the output compare function, the TIM 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 TIM can set, clear, or toggle the channel pin. Output compares can be enabled to generate interrupt requests.

16.3.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in 16.3.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM channel registers.

An unsynchronized write to the TIM 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 TIM overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIM may pass the new value before it is written.

Use the following methods to synchronize unbuffered changes in the output compare value on channel x:

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable TIM overflow interrupts and write the new value in the TIM overflow interrupt routine. The TIM 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.

16.3.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the TCH0 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The output compare value in the TIM channel 0 registers initially controls the output on the TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM 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 TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, TCH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered output compare channel whose output appears on the TCH2 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIM channel 2 status and control register (TSC2) links channel 2 and channel 3. The output compare value in the TIM channel 2 registers initially controls the output on the TCH2 pin. Writing to the TIM channel 3 registers enables the TIM channel 3 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM channel registers (2 or 3) that control the output are the ones written to last. TSC2 controls and monitors the buffered output compare function, and TIM channel 3 status and control register (TSC3) is unused. While the MS2B bit is set, the channel 3 pin, TCH3, 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.

16.3.4 Pulse Width Modulation (PWM)

By using the toggle-on-overflow feature with an output compare channel, the TIM can generate a PWM signal. The value in the TIM counter modulo registers determines the period of the PWM signal. The channel pin toggles when the counter reaches the value in the TIM counter modulo registers. The time between overflows is the period of the PWM signal.

As Figure 16-3 shows, the output compare value in the TIM channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIM to clear the channel pin on output compare if the polarity of the PWM pulse is 1 (ELSxA = 0). Program the TIM to set the pin if the polarity of the PWM pulse is 0 (ELSxA = 1).

The value in the TIM 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 TIM counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is 000. See 16.8.1 TIM Status and Control Register.

The value in the TIM 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 TIM channel registers produces a duty cycle of 128/256 or 50%.

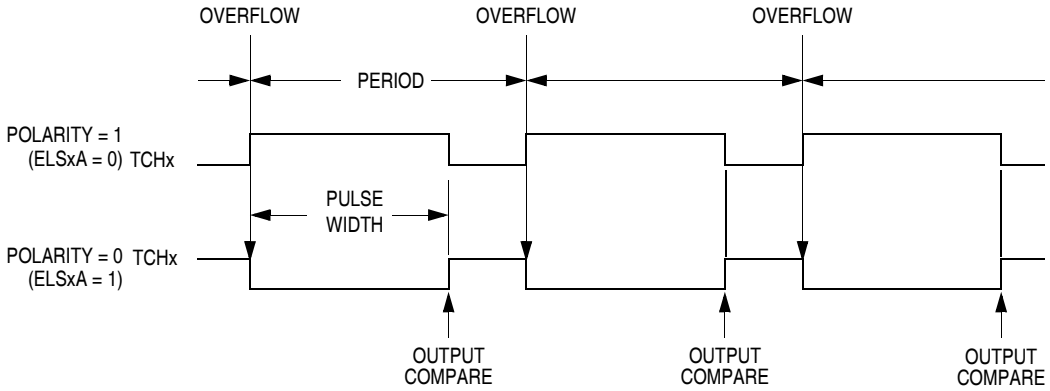


Figure 16-3. PWM Period and Pulse Width

16.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in 16.3.4 Pulse Width Modulation (PWM). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

An unsynchronized write to the TIM 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 TIM overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIM may pass the new value before it is written to the timer channel (TCHxH:TCHxL).

Use the following methods to synchronize unbuffered changes in the PWM pulse width on channel x:

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable TIM overflow interrupts and write the new value in the TIM overflow interrupt routine. The TIM 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.

16.3.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the TCH0 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The TIM channel 0 registers initially control the pulse width on the TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (0 or 1) that control the pulse width are the ones written to last. TSC0 controls and monitors the buffered PWM function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, TCH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered PWM channel whose output appears on the TCH2 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIM channel 2 status and control register (TSC2) links channel 2 and channel 3. The TIM channel 2 registers initially control the pulse width on the TCH2 pin. Writing to the TIM channel 3 registers enables the TIM channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (2 or 3) that control the pulse width are the ones written to last. TSC2 controls and monitors the buffered PWM function, and TIM channel 3 status and control register (TSC3) is unused. While the MS2B bit is set, the channel 3 pin, TCH3, 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.

16.3.4.3 PWM Initialization

To ensure correct operation when generating unbuffered or buffered PWM signals, use the following initialization procedure:

1. In the TIM status and control register (TSC):
   a. Stop the counter by setting the TIM stop bit, TSTOP.
   b. Reset the counter and prescaler by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH:TCHxL), write the value for the required pulse width.
4. In TIM channel x status and control register (TSCx):
   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 16-2.
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (polarity 1 — to clear output on compare) or 1:1 (polarity 0 — to set output on compare) to the edge/level select bits, ELSxB:ELSxA. The output action on compare must force the output to the complement of the pulse width level. See Table 16-2.

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 TIM status control register (TSC), clear the TIM stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIM channel 0 registers (TCH0H:TCH0L) initially control the buffered PWM output. TIM status control register 0 (TSC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Setting MS2B links channels 2 and 3 and configures them for buffered PWM operation. The TIM channel 2 registers (TCH2H:TCH2L) initially control the buffered PWM output. TIM status control register 2 (TSC2) controls and monitors the PWM signal from the linked channels. MS2B takes priority over MS2A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIM 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 16.8.4 TIM Channel Status and Control Registers.

16.4 Interrupts

The following TIM sources can generate interrupt requests:

- TIM overflow flag (TOF) — The TOF bit is set when the counter reaches the modulo value programmed in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow interrupt requests. TOF and TOIE are in the TSC register.
- TIM channel flags (CH3F:CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM interrupt requests are controlled by the channel x interrupt enable bit, CHxIE. Channel x TIM interrupt requests are enabled when CHxIE =1. CHxF and CHxIE are in the TSCx register.

16.5 Low-Power Modes

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

16.5.1 Wait Mode

The TIM remains active after the execution of a WAIT instruction. In wait mode the TIM registers are not accessible by the CPU. Any enabled interrupt request from the TIM can bring the MCU out of wait mode.

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

16.5.2 Stop Mode

The TIM module is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions. TIM operation resumes after an external interrupt. If stop mode is exited by reset, the TIM is reset.

16.6 TIM During Break Interrupts

A break interrupt stops the counter.

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.

16.7 I/O Signals

The TIM module can share its pins with the general-purpose I/O pins. See Figure 16-1 for the port pins that are shared.

16.7.1 TIM Channel I/O Pins (TCH3:TCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. TCH0 and TCH2 can be configured as buffered output compare or buffered PWM pins.

16.7.2 TIM Clock Pin (TCLK)

TCLK is an external clock input that can be the clock source for the counter instead of the prescaled internal bus clock. Select the TCLK input by writing 1s to the three prescaler select bits, PS[2:0]. (See 16.8.1 TIM Status and Control Register.) The minimum TCLK pulse width is specified in 18.16 Timer Interface Module Characteristics. The maximum TCLK frequency is the least of 4 MHz or bus frequency ÷ 2.

16.8 Registers

The following registers control and monitor operation of the TIM:

- TIM status and control register (TSC)
- TIM control registers (TCNTH:TCNTL)
- TIM counter modulo registers (TMODH:TMODL)
- TIM channel status and control registers (TSC0 through TSC3)
- TIM channel registers (TCH0H:TCH0L through TCH3H:TCH3L)

16.8.1 TIM Status and Control Register

The TIM status and control register (TSC) does the following:

- Enables TIM overflow interrupts
- Flags TIM overflows
- Stops the counter
- Resets the counter
- Prescales the counter clock

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
Write:	0			TRST				
Reset:	0	0	1	0	0	0	0	0

= Unimplemented

Figure 16-4. TIM Status and Control Register (TSC)

TOF — TIM Overflow Flag Bit

This read/write flag is set when the counter reaches the modulo value programmed in the TIM counter modulo registers. Clear TOF by reading the TSC register when TOF is set and then writing a 0 to TOF. If another TIM 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. Writing a 1 to TOF has no effect.

1 = Counter has reached modulo value
0 = Counter has not reached modulo value

TOIE — TIM Overflow Interrupt Enable Bit

This read/write bit enables TIM overflow interrupts when the TOF bit becomes set.

1 = TIM overflow interrupts enabled
0 = TIM overflow interrupts disabled

TSTOP — TIM Stop Bit

This read/write bit stops the counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the counter until software clears the TSTOP bit.

1 = Counter stopped
0 = Counter active

NOTE

Do not set the TSTOP bit before entering wait mode if the TIM is required to exit wait mode. Also, when the TSTOP bit is set and the timer is configured for input capture operation, input captures are inhibited until the TSTOP bit is cleared.

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 — TIM Reset Bit

Setting this write-only bit resets the counter and the TIM prescaler. Setting TRST has no effect on any other timer registers. Counting resumes from $0000. TRST is cleared automatically after the counter is reset and always reads as 0.

1 = Prescaler and counter cleared
0 = No effect

NOTE

Setting the TSTOP and TRST bits simultaneously stops the 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 counter as Table 16-1 shows.

Table 16-1. Prescaler Selection

PS2	PS1	PS0	TIM 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	TCLK (if available)

16.8.2 TIM Counter Registers

The two read-only TIM counter registers contain the high and low bytes of the value in the counter. Reading the high byte (TCNTH) latches the contents of the low byte (TCNTL) into a buffer. Subsequent reads of TCNTH do not affect the latched TCNTL value until TCNTL is read. Reset clears the TIM counter registers. Setting the TIM reset bit (TRST) also clears the TIM counter registers.

NOTE

If you read TCNTH during a break interrupt, be sure to unlatch TCNTL by reading TCNTL before exiting the break interrupt. Otherwise, TCNTL retains the value latched during the break.

	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

= Unimplemented

Figure 16-5. TIM Counter High Register (TCNTH)

	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 16-6. TIM Counter Low Register (TCNTL)

16.8.3 TIM Counter Modulo Registers

The read/write TIM modulo registers contain the modulo value for the counter. When the counter reaches the modulo value, the overflow flag (TOF) becomes set, and the 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 TIM counter modulo registers.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Reset:	1	1	1	1	1	1	1	1

Figure 16-7. TIM Counter Modulo High Register (TMODH)

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Reset:	1	1	1	1	1	1	1	1

Figure 16-8. TIM Counter Modulo Low Register (TMODL)

NOTE

Reset the counter before writing to the TIM counter modulo registers.

16.8.4 TIM Channel Status and Control Registers

Each of the TIM channel status and control registers does the following:

- Flags input captures and output compares
- Enables input capture and output compare interrupts
- Selects input capture, output compare, or PWM operation
- Selects high, low, or toggling output on output compare
- Selects rising edge, falling edge, or any edge as the active input capture trigger
- Selects output toggling on TIM overflow
- Selects 0% and 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

	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

Figure 16-9. TIM Channel 0 Status and Control Register (TSC0)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 16-10. TIM Channel 1 Status and Control Register (TSC1)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 16-11. TIM Channel 2 Status and Control Register (TSC2)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-12. TIM Channel 3 Status and Control Register (TSC3)

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 counter registers matches the value in the TIM channel x registers.

Clear CHxF by reading the TSCx 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.

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 TIM interrupt service requests on channel x.

1 = Channel x interrupt requests enabled
0 = Channel x interrupt requests disabled

MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TSC0 and TSC2 registers.

Setting MS0B causes the contents of TSC1 to be ignored by the TIM and reverts TCH1 to general-purpose I/O.

Setting MS2B causes the contents of TSC3 to be ignored by the TIM and reverts TCH3 to general-purpose I/O.

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 16-2.

1 = Unbuffered output compare/PWM operation
0 = Input capture operation

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin (see Table 16-2).

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 TIM status and control register (TSC).

Table 16-2. Mode, Edge, and Level Selection

MSxB	MSxA	ELSxB	ELSxA	Mode	Configuration
X	0	0	0	Output preset	Pin under port control; initial output level high
X	1	0	0		Pin under port control; initial output level low
0	0	0	1	Input capture	Capture on rising edge only
0	0	1	0		Capture on falling edge only
0	0	1	1		Capture on rising or falling edge
0	1	0	0	Output compare or PWM	Software compare only
0	1	0	1		Toggle output on compare
0	1	1	0		Clear output on compare
0	1	1	1		Set output on compare
1	X	0	1	Buffered output compare or buffered PWM	Toggle output on compare
1	X	1	0		Clear output on compare
1	X	1	1		Set output on compare

ELSxB and ELSxA — Edge/Level Select Bits

When channel x is an input capture channel, these read/write bits control the active edge-sensing logic on channel x.

When channel x is an output compare channel, ELSxB and ELSxA control the channel x output behavior when an output compare occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to an I/O port, and pin TCHx is available as a general-purpose I/O pin. Table 16-2 shows how ELSxB and ELSxA work.

NOTE

After initially enabling a TIM channel register for input capture operation and selecting the edge sensitivity, clear CHxF to ignore any erroneous edge detection flags.

TOVx — Toggle-On-Overflow Bit

When channel x is an output compare channel, this read/write bit controls the behavior of the channel x output when the counter overflows. When channel x is an input capture channel, TOVx has no effect.

1 = Channel x pin toggles on counter overflow.
0 = Channel x pin does not toggle on counter overflow.

NOTE

When TOVx is set, a counter overflow takes precedence over a channel x output compare if both occur at the same time.

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at 1, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 16-13 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.

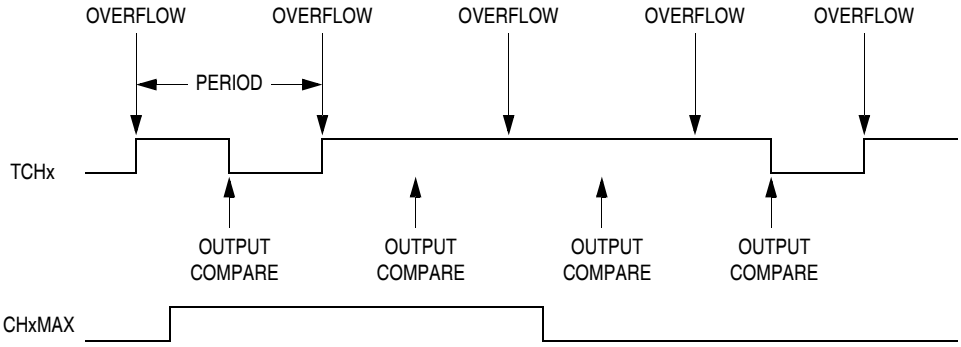


Figure 16-13. CHxMAX Latency

16.8.5 TIM Channel Registers

These read/write registers contain the captured counter value of the input capture function or the output compare value of the output compare function. The state of the TIM channel registers after reset is unknown.

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the TIM channel x registers (TCHxH) inhibits input captures until the low byte (TCHxL) is read.

In output compare mode (MSxB:MSxA ≠ 0:0), writing to the high byte of the TIM channel x registers (TCHxH) inhibits output compares until the low byte (TCHxL) is written.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	Indeterminate after reset							

Figure 16-14. TIM Channel x Register High (TCHxH)

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	Indeterminate after reset							

Figure 16-15. TIM Channel x Register Low (TCHxL)

Chapter 17
Development Support

17.1 Introduction

This section describes the break module, the monitor module (MON), and the monitor mode entry methods.

17.2 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 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

17.2.1 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:

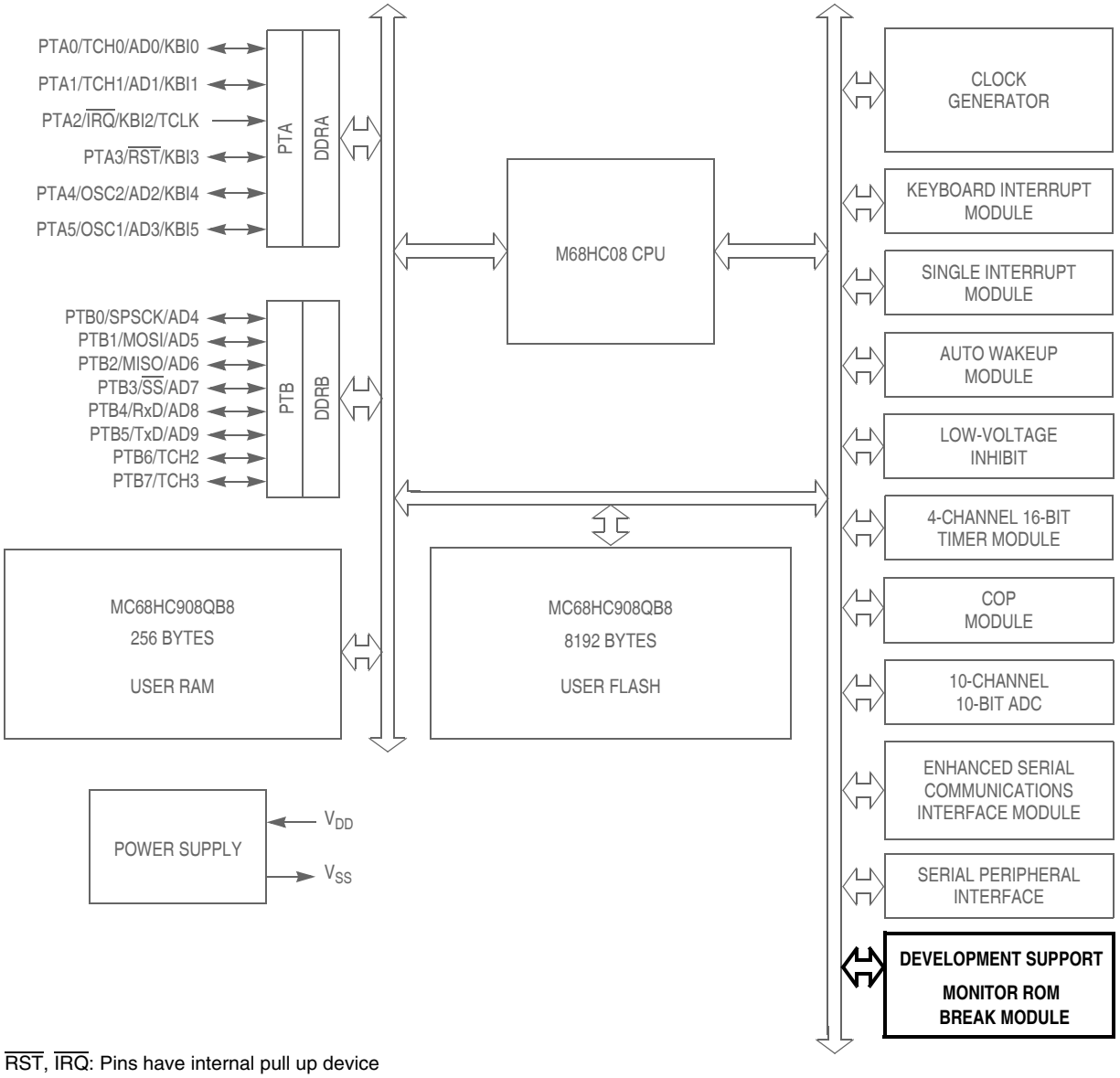
- A CPU generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a 1 to the BRKA bit in the break status and control register.

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 17-2 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)



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device
PTA[0:5]: Higher current sink and source capability

Figure 17-1. Block Diagram Highlighting BRK and MON Blocks

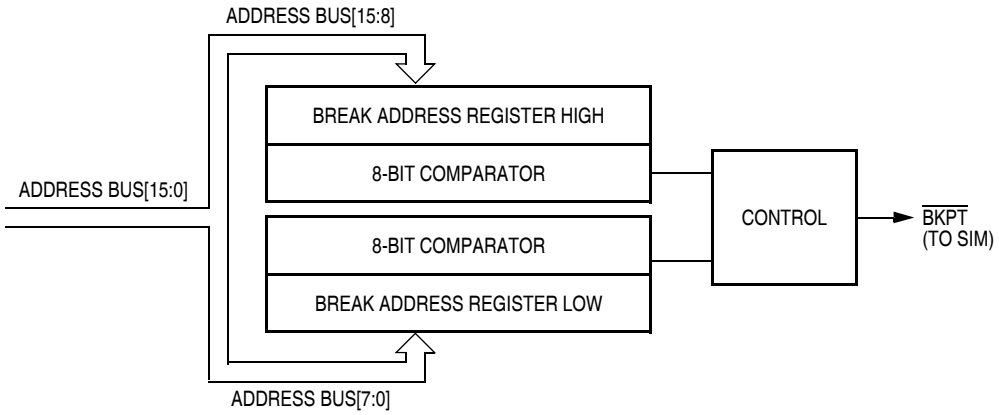


Figure 17-2. Break Module Block Diagram

The break interrupt timing is:

- When a break address is placed at the address of the instruction opcode, the instruction is not executed until after completion of the break interrupt routine.
- When a break address is placed at an address of an instruction operand, the instruction is executed before the break interrupt.
- When software writes a 1 to the BRKA bit, the break interrupt occurs just before the next instruction is executed.

By updating a break address and clearing the BRKA bit in a break interrupt routine, a break interrupt can be generated continuously.

CAUTION

A break address should be placed at the address of the instruction opcode. When software does not change the break address and clears the BRKA bit in the first break interrupt routine, the next break interrupt will not be generated after exiting the interrupt routine even when the internal address bus matches the value written in the break address registers.

17.2.1.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether or not module status bits can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See 14.8.2 Break Flag Control Register and the **Break Interrupts** subsection for each module.

17.2.1.2 TIM During Break Interrupts

A break interrupt stops the timer counter.

17.2.1.3 COP During Break Interrupts

The COP is disabled during a break interrupt in monitor mode when the BDCOP bit is set in the break auxiliary register (BRKAR).

17.2.2 Break Module Registers

These registers control and monitor operation of the break module:

- Break status and control register (BRKSCR)
- Break address register high (BRKH)
- Break address register low (BRKL)
- Break status register (BSR)
- Break flag control register (BFCR)

17.2.2.1 Break Status and Control Register

The break status and control register (BRKSCR) contains break module enable and status bits.

	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 17-3. Break Status and Control Register (BRKSCR)

BRKE — Break Enable Bit

This read/write bit enables breaks on break address register matches. Clear BRKE by writing a 0 to bit 7. Reset clears the BRKE bit.

1 = Breaks enabled on 16-bit address match
0 = Breaks disabled

BRKA — Break Active Bit

This read/write status and control bit is set when a break address match occurs. Writing a 1 to BRKA generates a break interrupt. Clear BRKA by writing a 0 to it before exiting the break routine. Reset clears the BRKA bit.

1 = Break address match
0 = No break address match

17.2.2.2 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.

	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 17-4. Break Address Register High (BRKH)

	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 17-5. Break Address Register Low (BRKL)

17.2.2.3 Break Auxiliary Register

The break auxiliary register (BRKAR) contains a bit that enables software to disable the COP while the MCU is in a state of break interrupt with monitor mode.

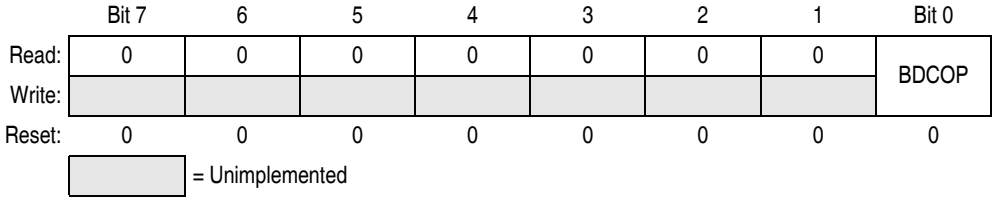


Figure 17-6. Break Auxiliary Register (BRKAR)

BDCOP — Break Disable COP Bit

This read/write bit disables the COP during a break interrupt. Reset clears the BDCOP bit.

1 = COP disabled during break interrupt
0 = COP enabled during break interrupt

17.2.2.4 Break Status Register

The break status register (BSR) contains a flag to indicate that a break caused an exit from wait mode. This register is only used in emulation mode.

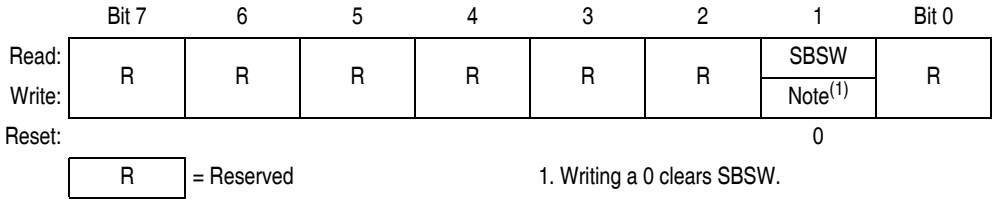


Figure 17-7. Break Status Register (BSR)

SBSW — SIM Break Stop/Wait

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it.

1 = Wait mode was exited by break interrupt
0 = Wait mode was not exited by break interrupt

17.2.2.5 Break Flag Control Register

The break control register (BFCR) contains a bit that enables software to clear status bits while the MCU is in a break state.

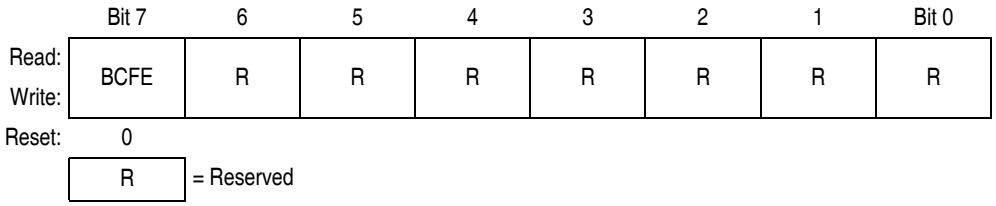


Figure 17-8. Break Flag Control Register (BFCR)

BCFE — Break Clear Flag Enable Bit

This read/write bit enables software to clear status bits by accessing status registers while the MCU is in a break state. To clear status bits during the break state, the BCFE bit must be set.

1 = Status bits clearable during break
0 = Status bits not clearable during break

17.2.3 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.

17.3 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 include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between MCU and host computer
- Standard non-return-to-zero (NRZ) communication with host computer
- Standard communication baud rate (7200 @ 2-MHz bus frequency)
- Execution of code in random-access memory (RAM) or FLASH
- FLASH memory security feature(1)
- FLASH memory programming interface
- Use of external 9.8304 MHz oscillator to generate internal frequency of 2.4576 MHz
- Simple internal oscillator mode of operation (no external clock or high voltage)
- 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 $\overline{IRQ}$

17.3.1 Functional Description

Figure 17-9 shows a simplified diagram of monitor mode entry.

The monitor module receives and executes commands from a host computer. Figure 17-10, Figure 17-11, and Figure 17-12 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.

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

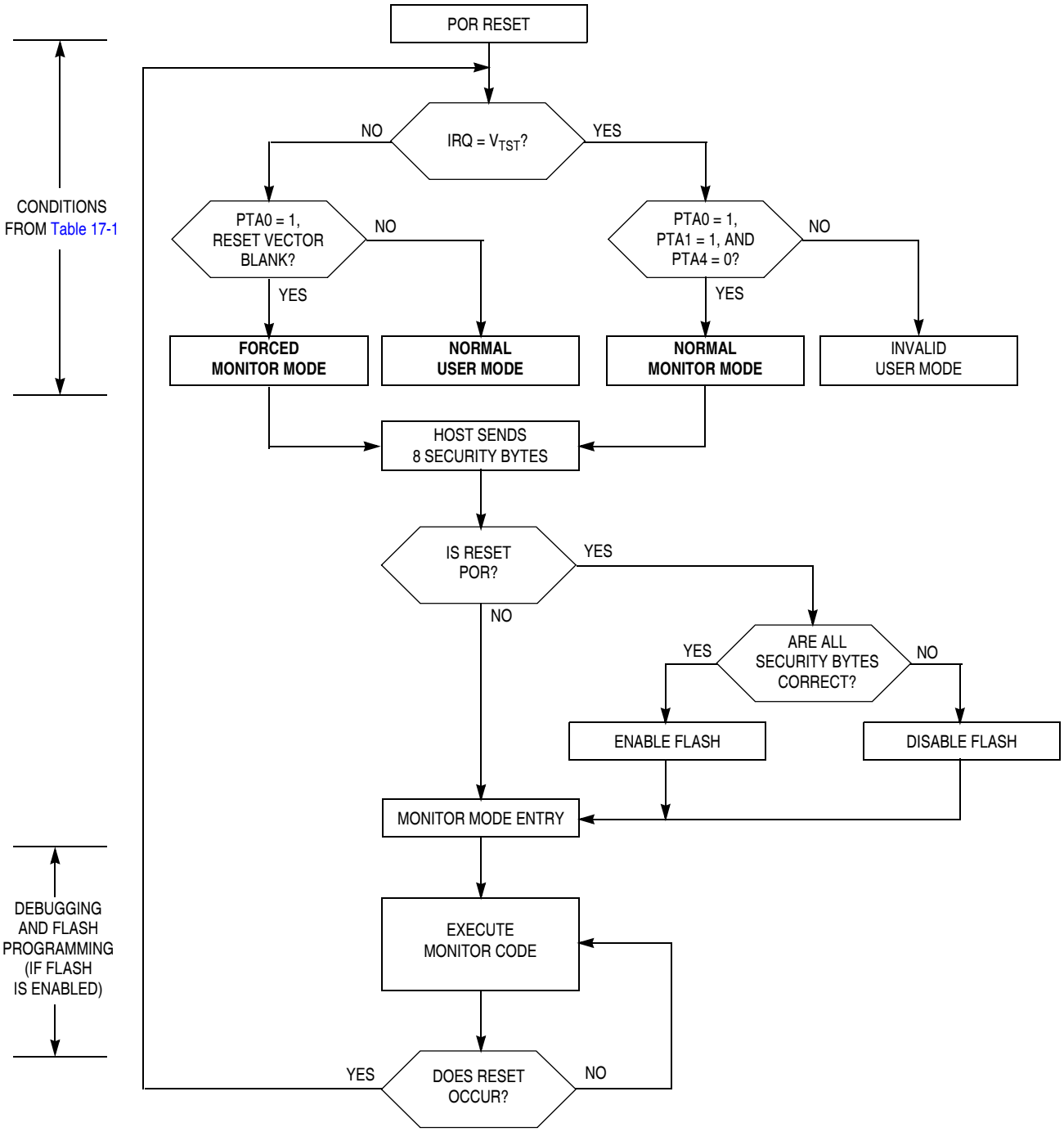


Figure 17-9. Simplified Monitor Mode Entry Flowchart

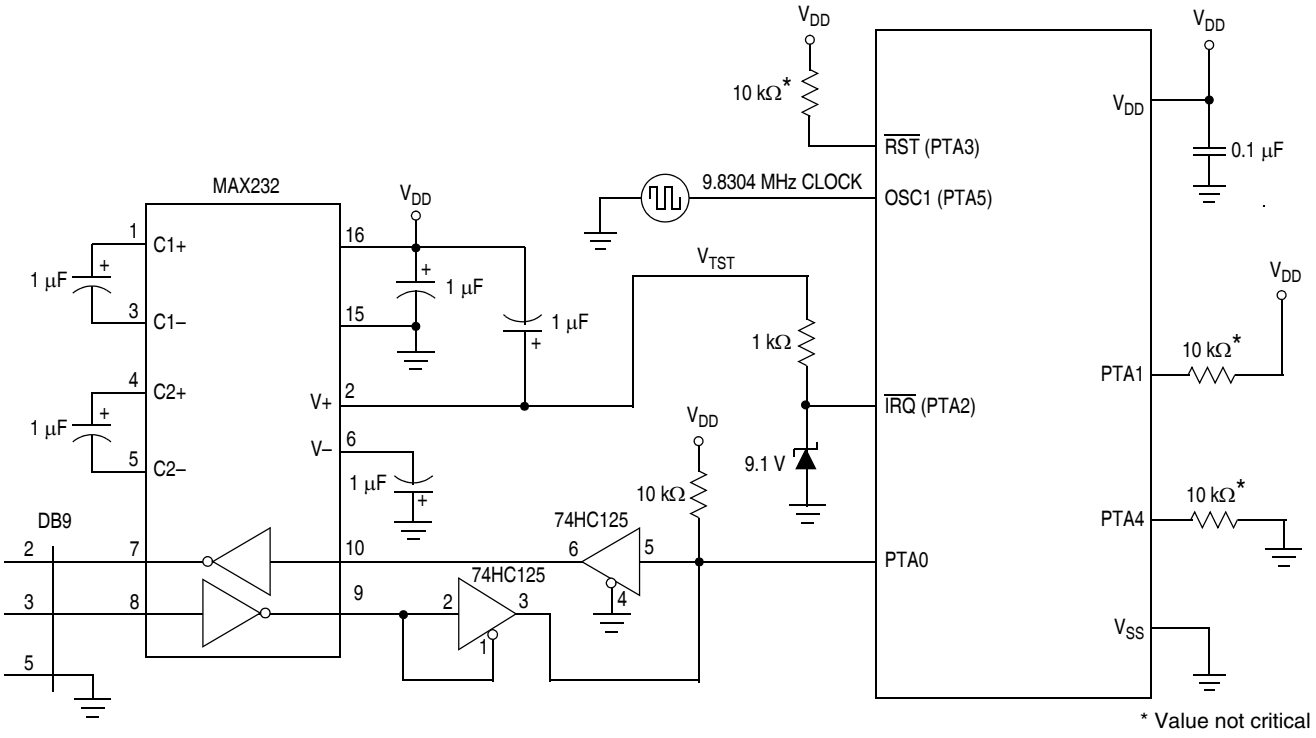


Figure 17-10. Monitor Mode Circuit (External Clock, with High Voltage)

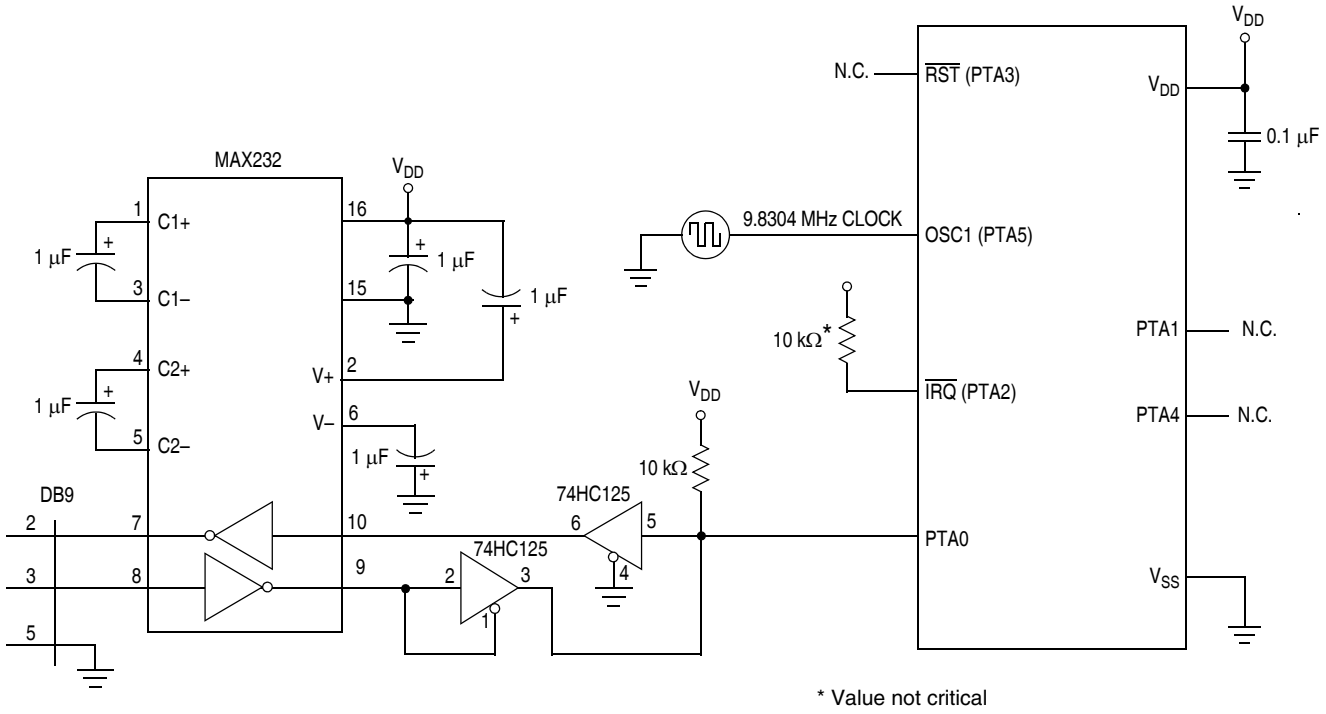


Figure 17-11. Monitor Mode Circuit (External Clock, No High Voltage)

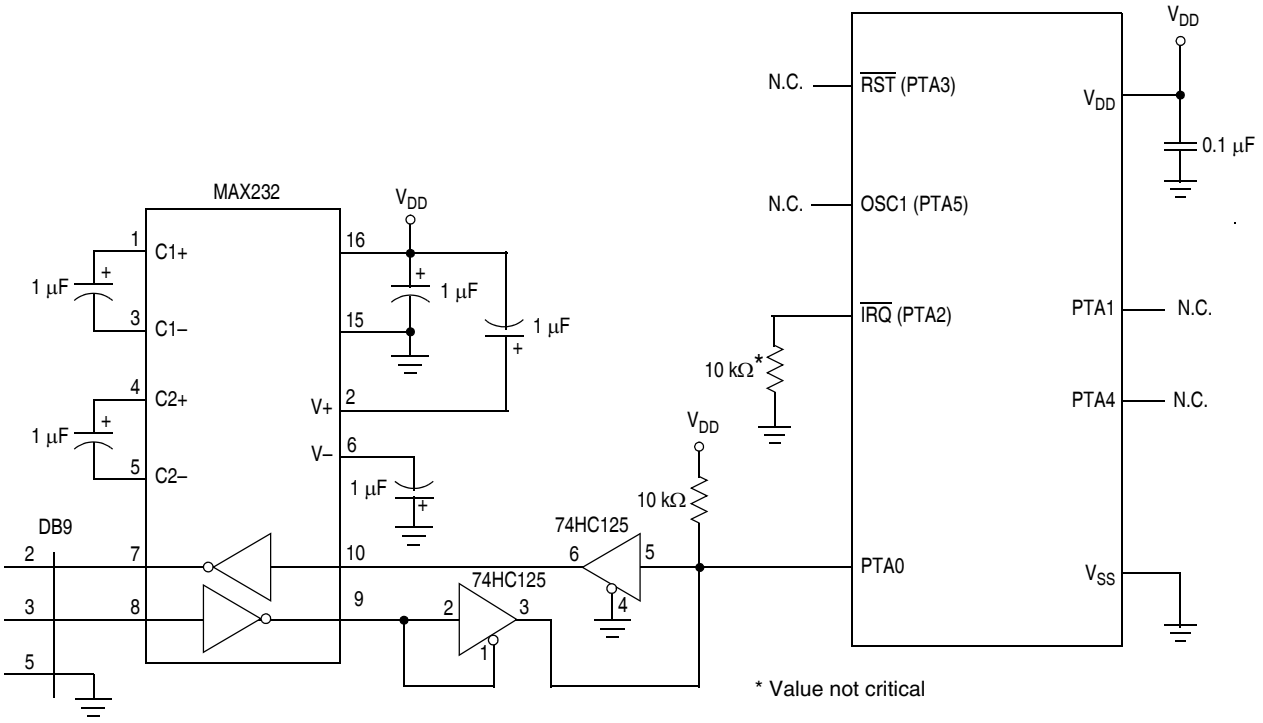


Figure 17-12. Monitor Mode Circuit (Internal Clock, No High Voltage)

The monitor code has been updated from previous versions of the monitor code to allow enabling the internal oscillator to generate the internal clock. This addition, which is enabled when $\overline{IRQ}$ is held low out of reset, is intended to support serial communication/programming at 9600 baud in monitor mode by using the internal oscillator, and the internal oscillator user trim value OSCTRIM (FLASH location $FFC0, if programmed) to generate the desired internal frequency (3.2 MHz). Since this feature is enabled only when $\overline{IRQ}$ is held low out of reset, it cannot be used when the reset vector is programmed (i.e., the value is not $FFFF) because entry into monitor mode in this case requires V_{TST} on $\overline{IRQ}$. The $\overline{IRQ}$ pin must remain low during this monitor session in order to maintain communication.

Table 17-1 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 do not contain $FF (programmed state):
  - The external clock is 9.8304 MHz
  - $\overline{IRQ}$ = V_{TST}
- If $FFFE and $FFFF contain $FF (erased state):
  - The external clock is 9.8304 MHz
  - $\overline{IRQ}$ = V_{DD} (this can be implemented through the internal $\overline{IRQ}$ pullup)
- If $FFFE and $FFFF contain $FF (erased state):
  - $\overline{IRQ}$ = V_{SS} (internal oscillator is selected, no external clock required)

The rising edge of the internal $\overline{RST}$ signal latches the monitor mode. Once monitor mode is latched, the values on PTA1 and PTA4 pins can be changed.

Once out of reset, the MCU waits for the host to send eight security bytes (see 17.3.2 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.

Table 17-1. Monitor Mode Signal Requirements and Options

Mode	$\overline{IRQ}$ (PTA2)	$\overline{RST}$ (PTA3)	Reset Vector	Serial Communication	Mode Selection		COP	Communication Speed			Comments
				PTA0	PTA1	PTA4		External Clock	Bus Frequency	Baud Rate	
Normal Monitor	V_{TST}	V_{DD}	X	1	1	0	Disabled	9.8304 MHz	2.4576 MHz	9600	Provide external clock at OSC1.
Forced Monitor	V_{DD}	X	$FFFF (blank)	1	X	X	Disabled	9.8304 MHz	2.4576 MHz	9600	Provide external clock at OSC1.
	V_{SS}	X	$FFFF (blank)	1	X	X	Disabled	X	3.2 MHz (Trimmed)	9600	Internal clock is active.
User	X	X	Not $FFFF	X	X	X	Enabled	X	X	X	
MON08 Function [Pin No.]	V_{TST} [6]	$\overline{RST}$ [4]	—	COM [8]	MOD0 [12]	MOD1 [10]	—	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. Baud rate using external oscillator is bus frequency / 256 and baud rate using internal oscillator is bus frequency / 335.
3. External clock is a 9.8304 MHz oscillator on OSC1.
4. Lowering V_{TST} once monitor mode is entered allows the clock source to be controlled by the OSCSC register.
5. X = don't care
6. MON08 pin refers to P&E Microcomputer Systems' MON08-Cyclone 2 by 8-pin connector.

NC	1	2	GND
NC	3	4	RST
NC	5	6	IRQ
NC	7	8	PTA0
NC	9	10	PTA4
NC	11	12	PTA1
OSC1	13	14	NC
V_{DD}	15	16	NC

17.3.1.1 Normal Monitor Mode

$\overline{RST}$ and OSC1 functions will be active on the PTA3 and PTA5 pins respectively as long as V_{TST} is applied to the $\overline{IRQ}$ pin. If the $\overline{IRQ}$ pin is lowered (no longer V_{TST}) then the chip will still be operating in monitor mode, but the pin functions will be determined by the settings in the configuration registers (see Chapter 5 Configuration Register (CONFIG)) when V_{TST} was lowered. With V_{TST} lowered, the BIH and BIL instructions will read the $\overline{IRQ}$ pin state only if IRQEN is set in the CONFIG2 register.

If monitor mode was entered with V_{TST} on $\overline{IRQ}$, then the COP is disabled as long as V_{TST} is applied to $\overline{IRQ}$.

17.3.1.2 Forced Monitor Mode

If entering monitor mode without high voltage on $\overline{IRQ}$, then startup port pin requirements and conditions, (PTA1/PTA4) are not in effect. This is to reduce circuit requirements when performing in-circuit programming.

NOTE

If the reset vector is blank and monitor mode is entered, the chip will see an additional reset cycle after the initial power-on reset (POR). Once the reset vector has been programmed, the traditional method of applying a voltage, V_{TST}, to $\overline{IRQ}$ must be used to enter monitor mode.

If monitor mode was entered as a result of the reset vector being blank, the COP is always disabled regardless of the state of $\overline{IRQ}$.

If the voltage applied to the $\overline{IRQ}$ is less than V_{TST}, the MCU will come out of reset in user mode. Internal circuitry monitors the reset vector fetches and will assert an internal reset if it detects that the reset vectors are erased ($FF). When the MCU comes out of reset, it is forced into monitor mode without requiring high voltage on the $\overline{IRQ}$ pin. Once out of reset, the monitor code is initially executing with the internal clock at its default frequency.

If $\overline{IRQ}$ is held high, all pins will default to regular input port functions except for PTA0 and PTA5 which will operate as a serial communication port and OSC1 input respectively (refer to Figure 17-11). That will allow the clock to be driven from an external source through OSC1 pin.

If $\overline{IRQ}$ is held low, all pins will default to regular input port function except for PTA0 which will operate as serial communication port. Refer to Figure 17-12.

Regardless of the state of the $\overline{IRQ}$ pin, it will not function as a port input pin in monitor mode. Bit 2 of the Port A data register will always read 0. The BIH and BIL instructions will behave as if the $\overline{IRQ}$ pin is enabled, regardless of the settings in the configuration register. See Chapter 5 Configuration Register (CONFIG).

The COP module is disabled in forced monitor mode. Any reset other than a power-on reset (POR) will automatically force the MCU to come back to the forced monitor mode.

17.3.1.3 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.

NOTE

Exiting monitor mode after it has been initiated by having a blank reset vector requires a power-on reset (POR). Pulling $\overline{RST}$ (when $\overline{RST}$ pin available) low will not exit monitor mode in this situation.

Table 17-2 summarizes the differences between user mode and monitor mode regarding vectors.

Table 17-2. Mode Difference

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

17.3.1.4 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.

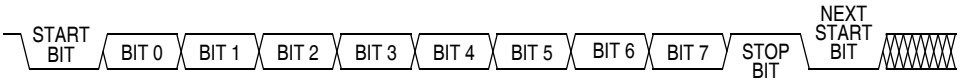


Figure 17-13. Monitor Data Format

17.3.1.5 Break Signal

A start bit (logic 0) followed by nine logic 0 bits is a break signal. When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits and then echoes back the break signal.

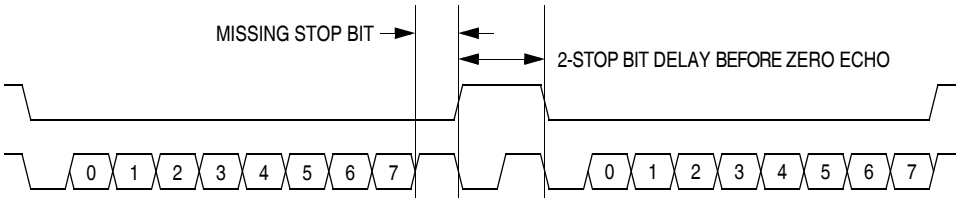


Figure 17-14. Break Transaction

17.3.1.6 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 17-1.

Table 17-1 also lists the bus frequencies to achieve standard baud rates. The effective baud rate is the bus frequency divided by 256 when using an external oscillator. When using the internal oscillator in forced monitor mode, the effective baud rate is the bus frequency divided by 335.

17.3.1.7 Commands

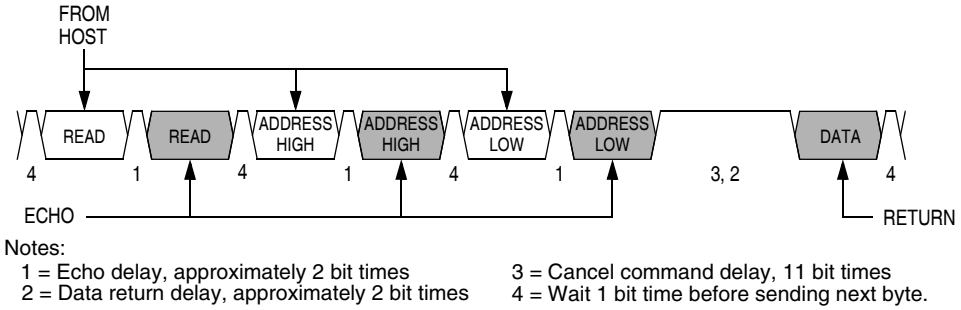
The monitor ROM firmware uses these commands:

- READ (read memory)
- WRITE (write memory)
- IREAD (indexed read)
- IWRITE (indexed write)
- READSP (read stack pointer)
- RUN (run user program)

The monitor ROM firmware echoes each received byte back to the PTA0 pin for error checking. An 11-bit delay at the end of each command allows the host to send a break character to cancel the command. A delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. The data returned by a read command appears after the echo of the last byte of the command.

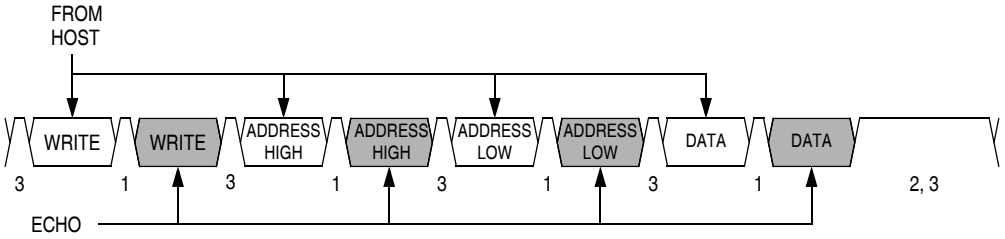
NOTE

Wait one bit time after each echo before sending the next byte.



Notes:
1 = Echo delay, approximately 2 bit times
2 = Data return delay, approximately 2 bit times
3 = Cancel command delay, 11 bit times
4 = Wait 1 bit time before sending next byte.

Figure 17-15. Read Transaction



Notes:
1 = Echo delay, approximately 2 bit times
2 = Cancel command delay, 11 bit times
3 = Wait 1 bit time before sending next byte.

Figure 17-16. Write Transaction

A brief description of each monitor mode command is given in Table 17-3 through Table 17-8.

Table 17-3. 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	

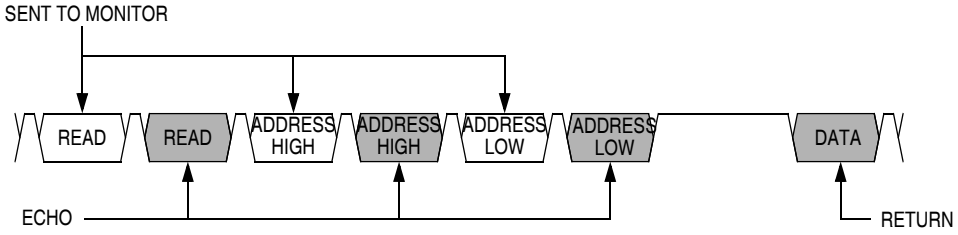


Table 17-4. WRITE (Write Memory) Command

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

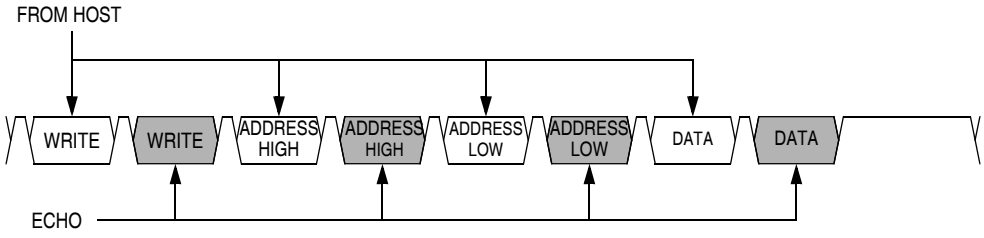


Table 17-5. IREAD (Indexed Read) Command

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

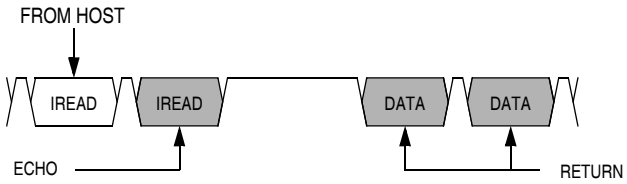
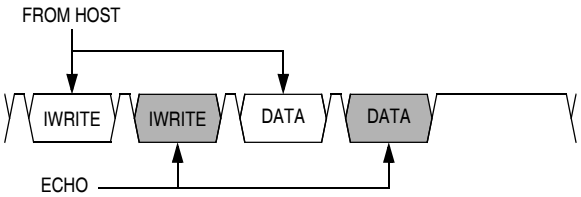


Table 17-6. IWRITE (Indexed Write) Command

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 17-7. 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	

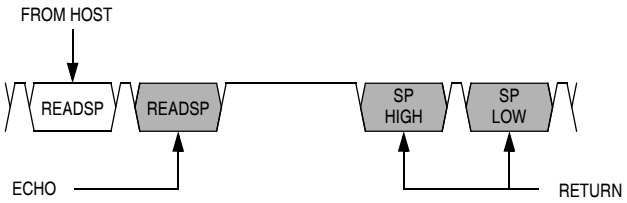
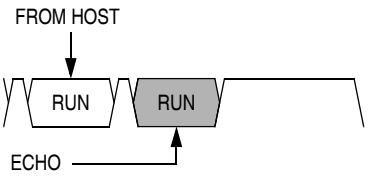


Table 17-8. RUN (Run User Program) Command

Description	Executes PULH and RTI instructions
Operand	None
Data Returned	None
Opcode	$28
Command Sequence	



The MCU executes the SWI and PSHH instructions when it enters monitor mode. The RUN command tells the MCU to execute the PULH and RTI instructions. Before sending the RUN command, the host can modify the stacked CPU registers to prepare to run the host program. The READSP command returns the incremented stack pointer value, SP + 1. The high and low bytes of the program counter are at addresses SP + 5 and SP + 6.

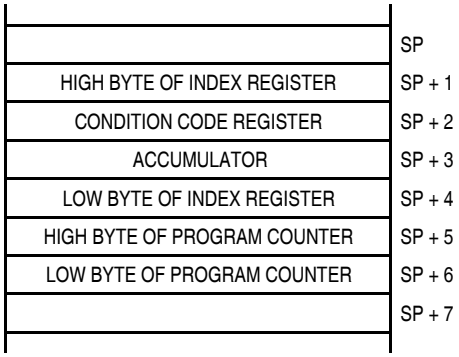


Figure 17-17. Stack Pointer at Monitor Mode Entry

17.3.2 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 17-18.

Upon power-on reset, if the received bytes of the security code do not match the data at locations $FFF6–$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. After receiving the eight security bytes from the host, the MCU transmits a break character, signifying that it is ready to receive a command.

NOTE

The MCU does not transmit a break character until after the host sends the eight security bytes.

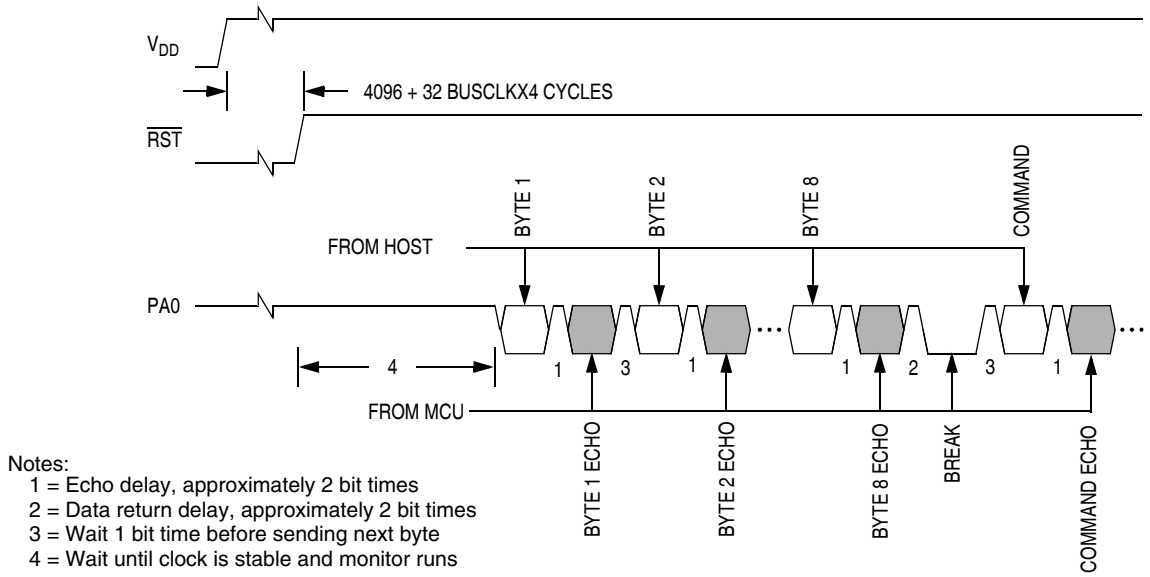


Figure 17-18. Monitor Mode Entry Timing

To determine whether the security code entered is correct, check to see if bit 6 of RAM address $80 is set. If it is, then the correct security code has been entered and FLASH can be accessed.

If the security sequence fails, the device should be reset by a power-on reset and brought up in monitor mode to attempt another entry. After failing the security sequence, the FLASH module can also be mass erased by executing an erase routine that was downloaded into internal RAM. The mass erase operation clears the security code locations so that all eight security bytes become $FF (blank).

Chapter 18
Electrical Specifications

18.1 Introduction

This section contains electrical and timing specifications.

18.2 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 18.5 5-V DC Electrical Characteristics and 18.8 3-V 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
Mode entry voltage, $\overline{IRQ}$ pin	V_{TST}	V_{SS} −0.3 to +9.1	V
Maximum current per pin excluding PTA0–PTA5, V_{DD}, and V_{SS}	I	±15	mA
Maximum current for pins PTA0–PTA5	I_{PTA0}—I_{PTA5}	±25	mA
Storage temperature	T_{STG}	−55 to +150	°C
Maximum current out of V_{SS}	I_{MVSS}	100	mA
Maximum current into V_{DD}	I_{MVDD}	100	mA

1. Voltages references 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}.)

18.3 Functional Operating Range

Characteristic	Symbol	Value	Unit	Temperature Code
Operating temperature range	T_A (T_L to T_H)	−40 to +125 −40 to +105 −40 to +85	°C	M V C
Operating voltage range	V_{DD}	2.7 to 5.5	V	—

18.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance 16-pin PDIP 16-pin SOIC 16-pin TSSOP	θ_{JA}	 76 90 133	°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
Maximum junction temperature	T_{JM}	150	°C

1. Power dissipation is a function of temperature.
2. K 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.

18.5 5-V DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Output high voltage I_{Load} = –2.0 mA, all I/O pins I_{Load} = –10.0 mA, all I/O pins I_{Load} = –15.0 mA, PTA0, PTA1, PTA3–PTA5 only	V_{OH}	 V_{DD}–0.4 V_{DD}–1.5 V_{DD}–0.8	 — — —	 — — —	V
Maximum combined I_{OH} (all I/O pins)	I_{OHT}	—	—	50	mA
Output low voltage I_{Load} = 1.6 mA, all I/O pins I_{Load} = 10.0 mA, all I/O pins I_{Load} = 15.0 mA, PTA0, PTA1, PTA3–PTA5 only	V_{OL}	 — — —	 — — —	 0.4 1.5 0.8	V
Maximum combined I_{OL} (all I/O pins)	I_{OHL}	—	—	50	mA
Input high voltage PTA0–PTA5, PTB0–PTB7	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input low voltage PTA0–PTA5, PTB0–PTB7	V_{IL}	V_{SS}	—	0.3 x V_{DD}	V
Input hysteresis[3]	V_{HYS}	0.06 x V_{DD}	—	—	V
DC injection current, all ports[4]	I_{INJ}	–2	—	+2	mA
Total dc current injection (sum of all I/O)[4]	I_{INJTOT}	–25	—	+25	mA
Ports Hi-Z leakage current	I_{IL}	–1	±0.1	+1	µA
Capacitance Ports (as input)[3]	C_{IN}	—	—	8	pF
POR rearm voltage	V_{POR}	750	—	—	mV
POR rise time ramp rate[3][5]	R_{POR}	0.035	—	—	V/ms
Monitor mode entry voltage [3]	V_{TST}	V_{DD} + 2.5	—	9.1	V
Pullup resistors[6] PTA0–PTA5, PTB0–PTB7	R_{PU}	16	26	36	kΩ
Pulldown resistors[7] PTA0–PTA5	R_{PD}	16	26	36	kΩ
Low-voltage inhibit reset, trip falling voltage	V_{TRIPF}	3.90	4.20	4.50	V
Low-voltage inhibit reset, trip rising voltage	V_{TRIPR}	4.00	4.30	4.60	V
Low-voltage inhibit reset/recover hysteresis	V_{HYS}	—	100	—	mV

1. V_{DD} = 4.5 to 5.5 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. Typical values are for reference only and are not tested in production.
3. Values are based on characterization results, not tested in production.
4. Guaranteed by design, not tested in production.
5. If minimum V_{DD} is not reached before the internal POR reset is released, the LVI will hold the part in reset until minimum V_{DD} is reached.
6. R_{PU} is measured at V_{DD} = 5.0 V.
7. R_{PD} is measured at V_{DD} = 5.0 V, Pulldown resistors only available when KBIx is enabled with KBIxPOL =1.

18.6 Typical 5-V Output Drive Characteristics

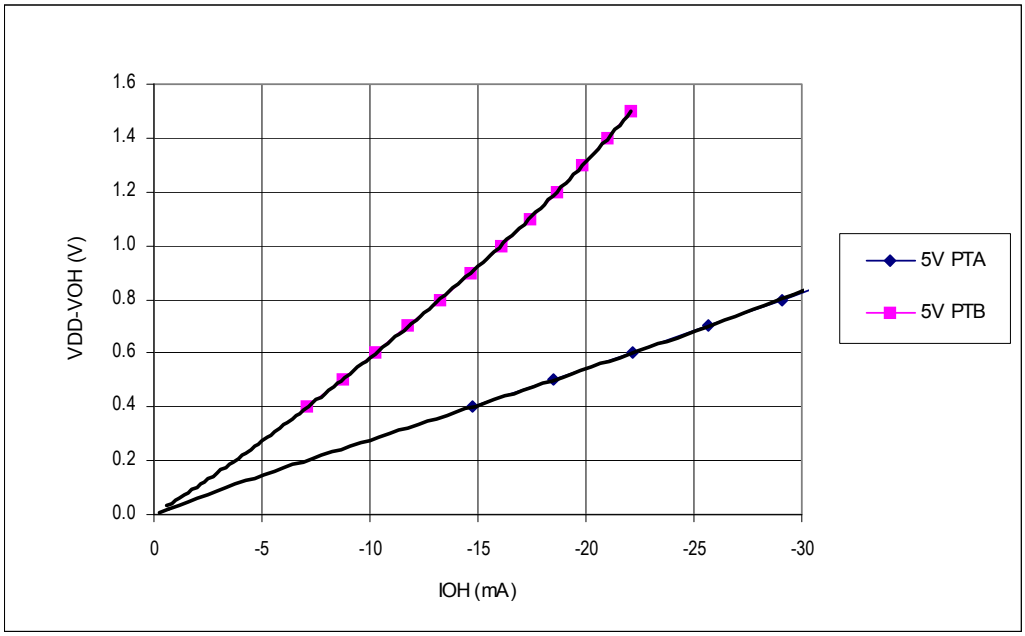


Figure 18-1. Typical 5-Volt Output High Voltage versus Output High Current (25°C)

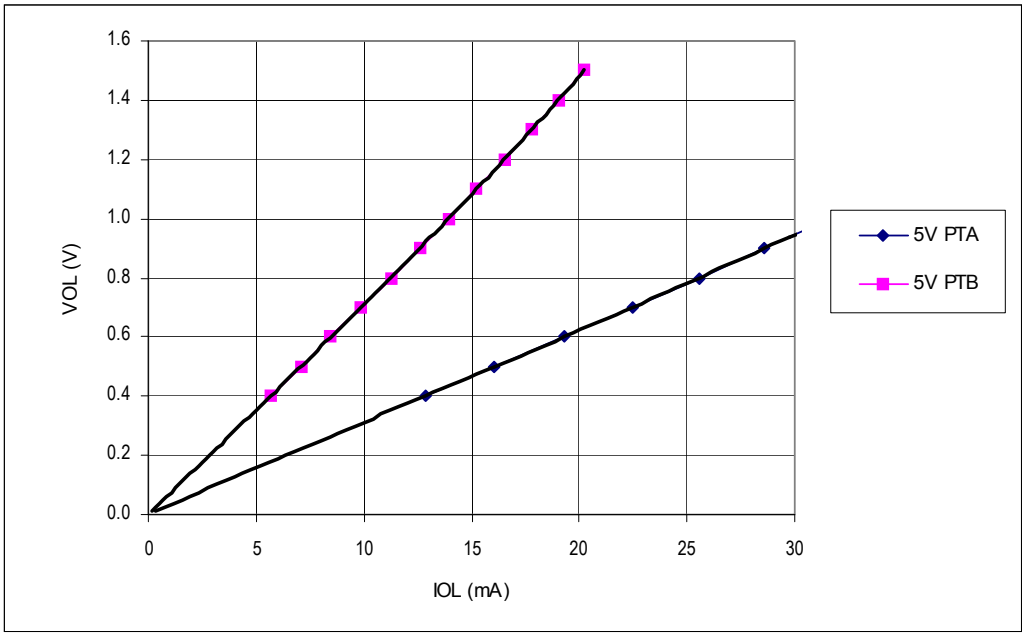


Figure 18-2. Typical 5-Volt Output Low Voltage versus Output Low Current (25°C)

18.7 5-V Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Internal operating frequency	f_{OP} (f_{BUS})	—	8	MHz
Internal clock period ($1/f_{OP}$)	t_{cyc}	125	—	ns
$\overline{RST}$ input pulse width low[2]	t_{RL}	100	—	ns
$\overline{IRQ}$ interrupt pulse width low (edge-triggered)[2]	t_{ILIH}	100	—	ns
$\overline{IRQ}$ interrupt pulse period[2]	t_{ILIL}	Note[3]	—	t_{cyc}

1. V_{DD} = 4.5 to 5.5 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. Values are based on characterization results, not tested in production.
3. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 t_{cyc}.

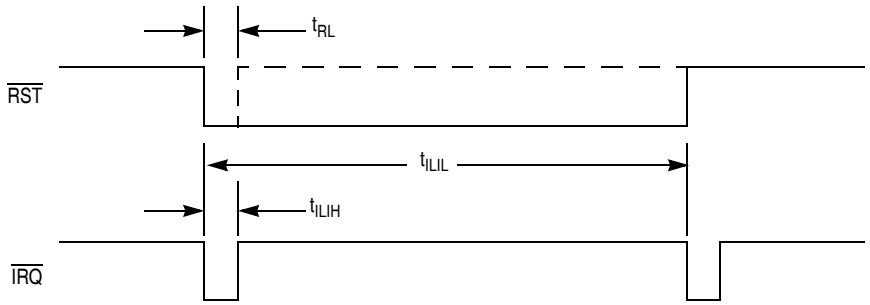


Figure 18-3. $\overline{RST}$ and $\overline{IRQ}$ Timing

18.8 3-V DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Output high voltage I_{Load} = –0.6 mA, all I/O pins I_{Load} = –4.0 mA, all I/O pins I_{Load} = –10.0 mA, PTA0, PTA1, PTA3–PTA5 only	V_{OH}	 V_{DD}–0.3 V_{DD}–1.0 V_{DD}–0.8	 — — —	 — — —	V
Maximum combined I_{OH} (all I/O pins)	I_{OHT}	—	—	50	mA
Output low voltage I_{Load} = 0.5 mA, all I/O pins I_{Load} = 6.0 mA, all I/O pins I_{Load} = 10.0 mA, PTA0, PTA1, PTA3–PTA5 only	V_{OL}	 — — —	 — — —	 0.3 1.0 0.8	V
Maximum combined I_{OL} (all I/O pins)	I_{OHL}	—	—	50	mA
Input high voltage PTA0–PTA5, PTB0–PTB7	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input low voltage PTA0–PTA5, PTB0–PTB7	V_{IL}	V_{SS}	—	0.3 x V_{DD}	V
Input hysteresis[3]	V_{HYS}	0.06 x V_{DD}	—	—	V
DC injection current, all ports[4]	I_{INJ}	–2	—	+2	mA
Total dc current injection (sum of all I/O)[4]	I_{INJTOT}	–25	—	+25	mA
Ports Hi-Z leakage current	I_{IL}	–1	±0.1	+1	µA
Capacitance Ports (as input)[3]	C_{IN}	—	—	8	pF
POR rearm voltage	V_{POR}	750	—	—	mV
POR rise time ramp rate[3][5]	R_{POR}	0.035	—	—	V/ms
Monitor mode entry voltage [3]	V_{TST}	V_{DD} + 2.5	—	V_{DD} + 4.0	V
Pullup resistors[6] PTA0–PTA5, PTB0–PTB7	R_{PU}	16	26	36	kΩ
Pulldown resistors[7] PTA0–PTA5	R_{PD}	16	26	36	kΩ
Low-voltage inhibit reset, trip falling voltage	V_{TRIPF}	2.40	2.55	2.70	V
Low-voltage inhibit reset, trip rising voltage[6]	V_{TRIPR}	2.475	2.625	2.775	V
Low-voltage inhibit reset/recover hysteresis	V_{HYS}	—	75	—	mV

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. Typical values are for reference only and are not tested in production.
3. Values are based on characterization results, not tested in production.
4. Guaranteed by design, not tested in production.
5. If minimum V_{DD} is not reached before the internal POR reset is released, the LVI will hold the part in reset until minimum V_{DD} is reached.
6. R_{PU} is measured at V_{DD} = 3.0 V
7. R_{PD} is measured at V_{DD} = 3.0 V, Pulldown resistors only available when KBIx is enabled with KBIxPOL =1.

18.9 Typical 3-V Output Drive Characteristics

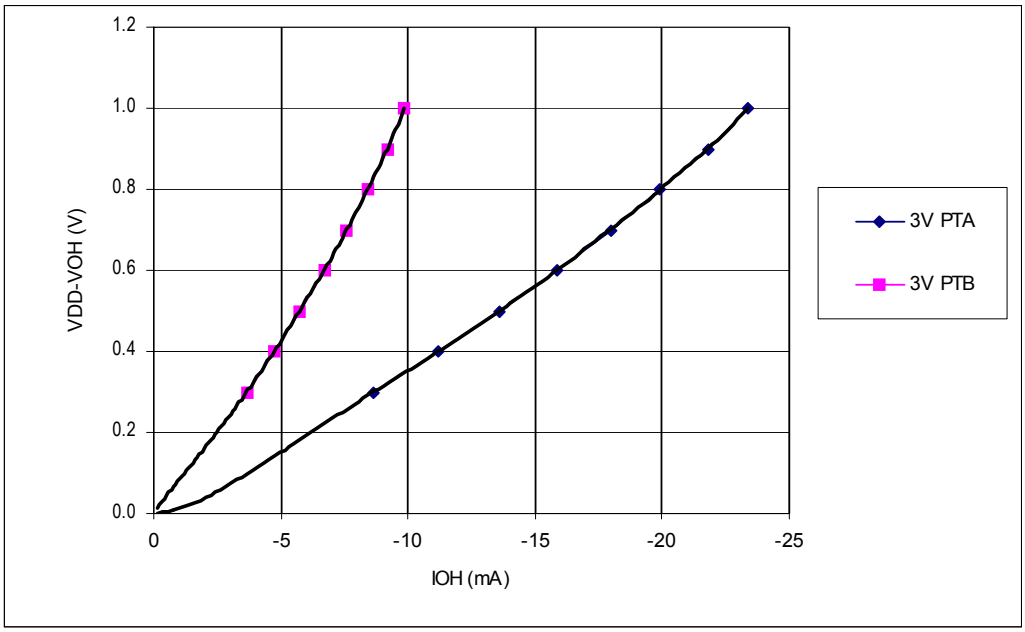


Figure 18-4. Typical 3-Volt Output High Voltage versus Output High Current (25°C)

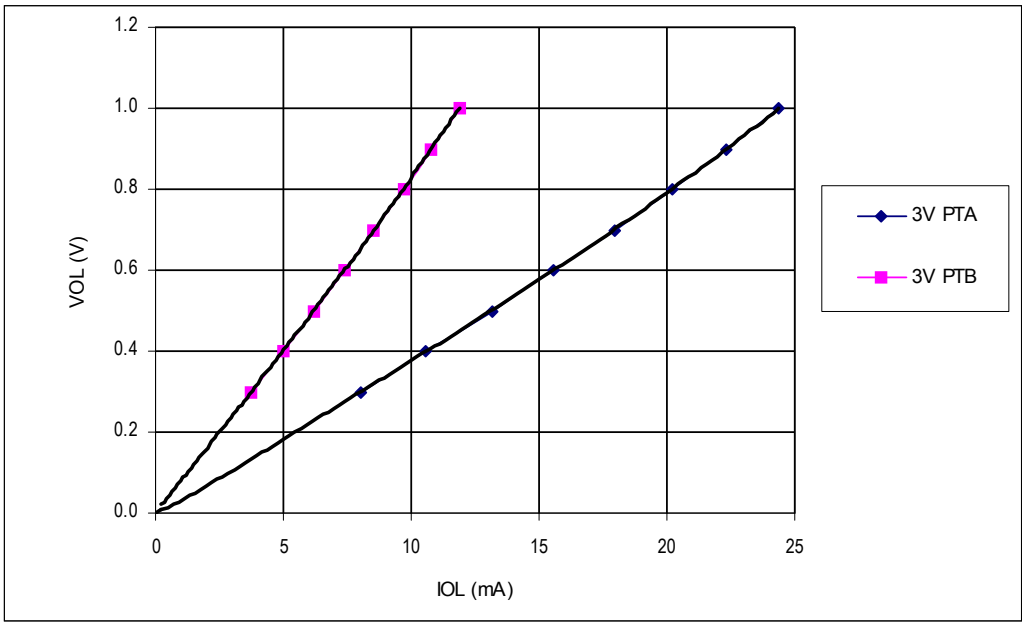


Figure 18-5. Typical 3-Volt Output Low Voltage versus Output Low Current (25°C)

18.10 3-V Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Internal operating frequency	f_{OP} (f_{Bus})	—	4	MHz
Internal clock period ($1/f_{OP}$)	t_{cyc}	250	—	ns
$\overline{RST}$ input pulse width low[2]	t_{RL}	200	—	ns
$\overline{IRQ}$ interrupt pulse width low (edge-triggered)[2]	t_{ILIH}	200	—	ns
$\overline{IRQ}$ interrupt pulse period[2]	t_{ILIL}	Note[3]	—	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_{DD}, unless otherwise noted.
2. Values are based on characterization results, not tested in production.
3. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 t_{cyc}.

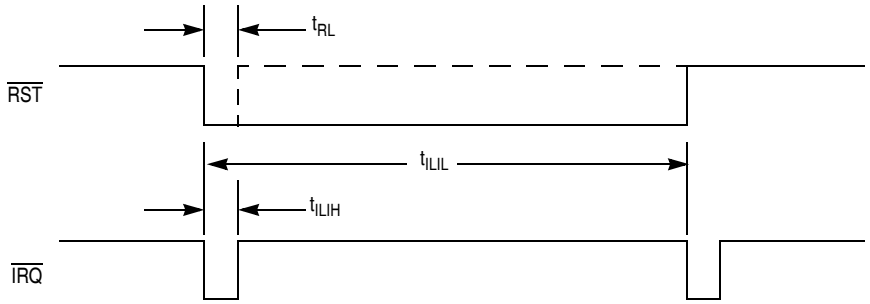


Figure 18-6. $\overline{RST}$ and $\overline{IRQ}$ Timing

18.11 Oscillator Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
Internal oscillator frequency[1] ICFS1:ICFS0 = 00 ICFS1:ICFS0 = 01 ICFS1:ICFS0 = 10 (not allowed if V_{DD} <2.7V)	f_{INTCLK}	— — —	4 8 12.8	— — —	MHz
Trim accuracy[2][3]	Δ_{TRIM_ACC}	—	± 0.4	—	%
Deviation from trimmed Internal oscillator[3][4] 4, 8, 12.8MHz, V_{DD} ± 10%, 0 to 70°C 4, 8, 12.8MHz, V_{DD} ± 10%, –40 to 125°C	Δ_{INT_TRIM}	— —	± 2 —	— ± 5	%
External RC oscillator frequency, RCCLK [1][3]	f_{RCCLK}	2	—	10	MHz
External clock reference frequencyy[1][5][6] $V_{DD} \geq 4.5V$ $V_{DD} < 4.5V$	$f_{OSCXCLK}$	dc dc	—	32 16	MHz
RC oscillator external resistor[3] V_{DD} = 5 V V_{DD} = 3 V	R_{EXT}	See Figure 18-7 See Figure 18-8			—
Crystal frequency, XTALCLK[1][7][8] ECFS1:ECFS0 = 00 ($V_{DD} \geq 4.5$ V) ECFS1:ECFS0 = 00 ECFS1:ECFS0 = 01 ECFS1:ECFS0 = 10	$f_{OSCXCLK}$	8 8 1 30	—	32 16 8 100	MHz MHz MHz kHz
ECFS1:ECFS0 = 00 [9] Feedback bias resistor Crystal load capacitance[10] Crystal capacitors[10]	 R_B C_L C_1,C_2	 — — —	 1 20 (2 x C_L) – 5pF	 — — —	 MΩ pF pF
ECFS1:ECFS0 = 01[9] Crystal series damping resistor $f_{OSCXCLK}$ = 1 MHz $f_{OSCXCLK}$ = 4 MHz $f_{OSCXCLK}$ = 8 MHz Feedback bias resistor Crystal load capacitance[10] Crystal capacitors[10]	 R_S R_B C_L C_1,C_2	 — — — — — —	 20 10 0 5 18 (2 x C_L) –10 pF	 — — — — — —	 kΩ kΩ kΩ MΩ pF pF
AWU module internal RC oscillator frequency	f_{INTRC}	—	32	—	kHz

1. Bus frequency, f_{OP}, is oscillator frequency divided by 4.
2. Factory trimmed to provided 12.8MHz accuracy requirement (± 5%, @25•C) for forced monitor mode communication. User should trim in-circuit to obtain the most accurate internal oscillator frequency for his application.
3. Values are based on characterization results, not tested in production.
4. Deviation values assumes trimming in target application @25•C and midpoint of voltage range, for example 5.0 V for 5 V ± 10% operation.
5. No more than 10% duty cycle deviation from 50%.
6. When external oscillator clock is greater than 1MHz, ECFS1:ECFS0 must be 00 or 01
7. Use fundamental mode only, do **not** use overtone crystals or overtone ceramic resonators
8. Due to variations in electrical properties of external components such as, ESR and Load Capacitance, operation above 16 MHz is not guaranteed for all crystals or ceramic resonators. Operation above 16 MHz requires that a Negative Resistance Margin (NRM) characterization and component optimization be performed by the crystal or ceramic resonator vendor for every different type of crystal or ceramic resonator which will be used. This characterization and optimization must be performed at the extremes of voltage and temperature which will be applied to the microcontroller in the application. The NRM must meet or exceed 10x the maximum ESR of the crystal or ceramic resonator for acceptable performance.
9. Do not use damping resistor when ECFS1:ECFS0 = 00 or 10
10. Consult crystal vendor data sheet.

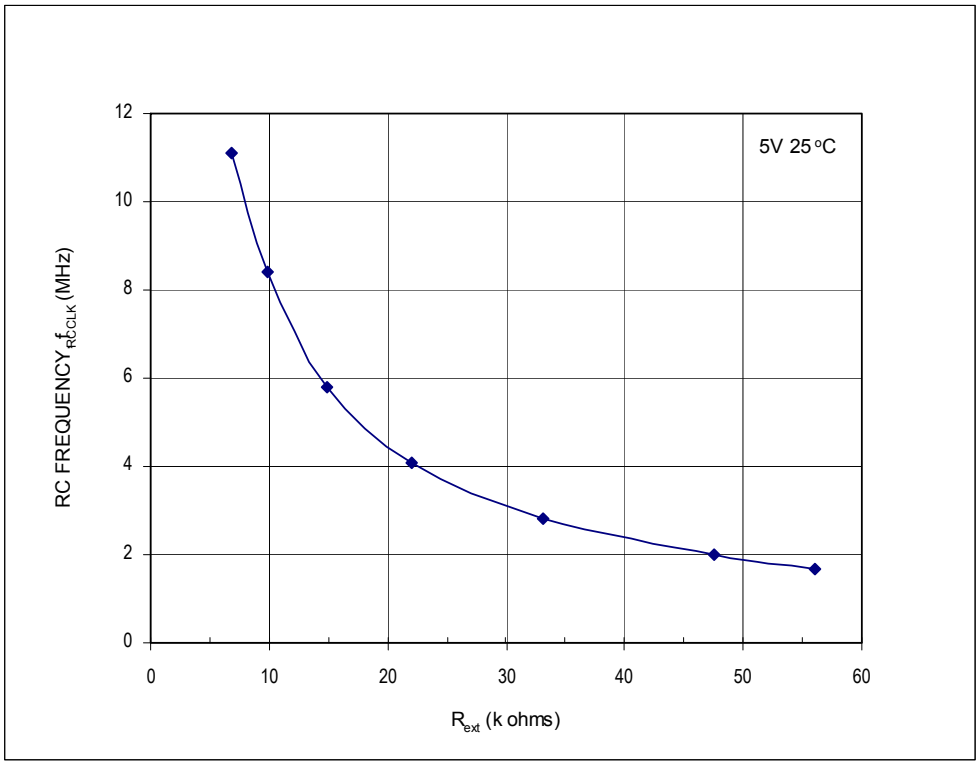


Figure 18-7. RC versus Frequency (5 Volts @ 25°C)

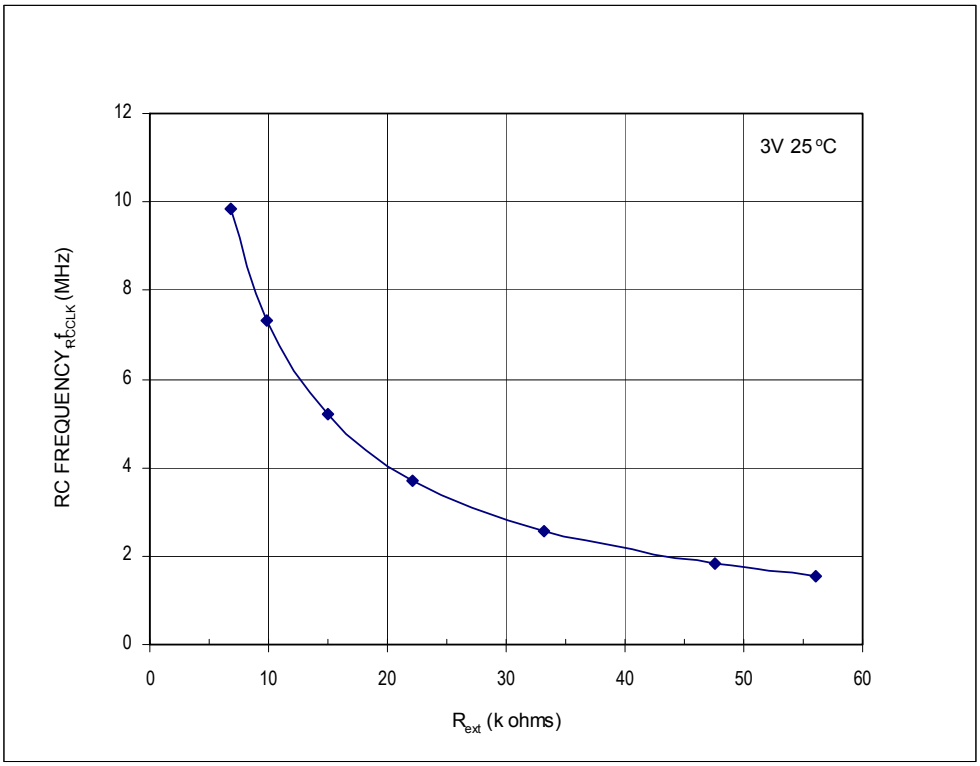


Figure 18-8. RC versus Frequency (3 Volts @ 25°C)

18.12 Supply Current Characteristics

Characteristic[1]	Voltage	Bus Frequency (MHz)	Symbol	Typ[2]	Max	Unit
Run mode V_{DD} supply current[3]	5.0 3.0	3.2 3.2	RI_{DD}	7.25 3.1	8.5 3.8	mA
Wait mode V_{DD} supply current[4]	5.0 3.0	3.2 3.2	WI_{DD}	1.0 0.67	2.0 1.2	mA
Stop mode V_{DD} supply current[5] –40 to 85°C –40 to 105°C –40 to 125°C 25°C with auto wake-up enabled[6] Incremental current with LVI enabled	5.0		SI_{DD}	 0.26 — — 12 125	 1.0 2.0 5.0 — —	µA
Stop mode V_{DD} supply current[4] –40 to 85°C –40 to 105°C –40 to 125°C 25°C with auto wake-up enabled[6] Incremental current with LVI enabled	3.0			 0.23 — — 2 100	 0.8 1.0 4.0 — —	µA

1. V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted.
2. Typical values reflect average measurements, 25•C only. Typical values are for reference only and are not tested in production.
3. Run (operating) I_{DD} measured using trimmed internal oscillator, ADC off, all modules enabled. All pins configured as inputs and tied to 0.2 V from rail.
4. Wait I_{DD} measured using trimmed internal oscillator, ADC off, all modules enabled. All pins configured as inputs and tied to 0.2 V from rail.
5. Stop I_{DD} measured with all pins configured as inputs and tied to 0.2 V from rail.
6. Values are based on characterization results, not tested in production.

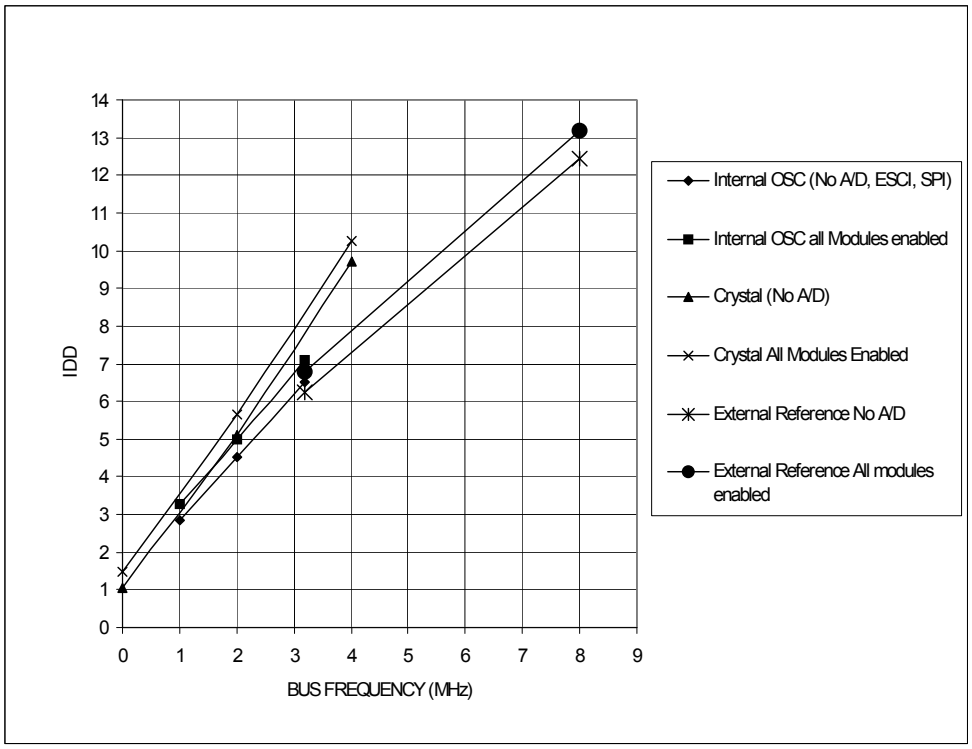


Figure 18-9. Typical 5-Volt Run Current versus Bus Frequency (25•C)

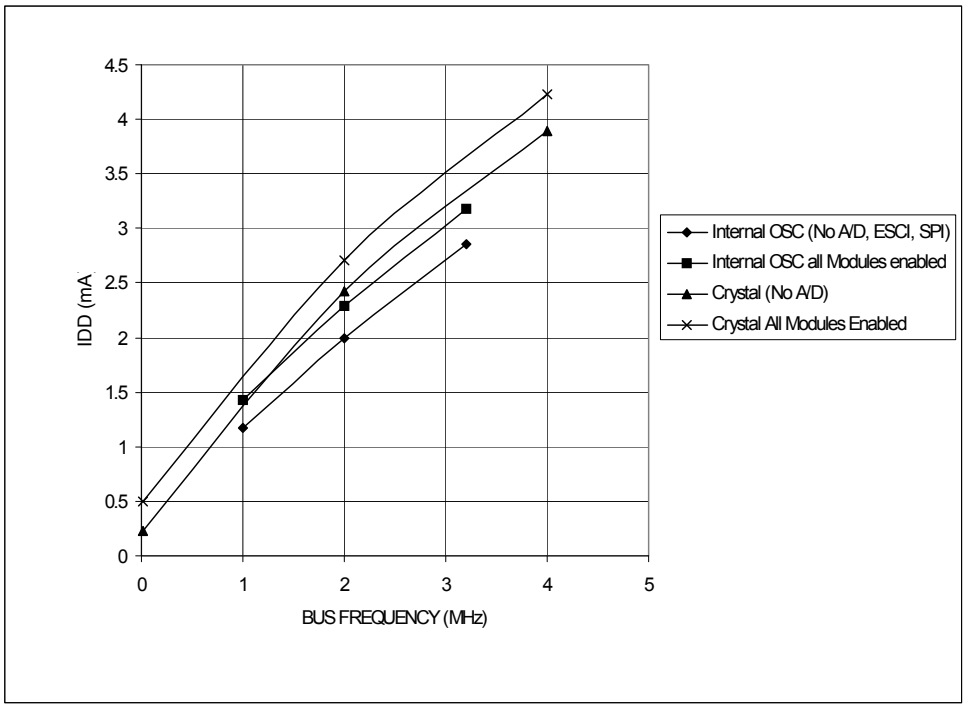


Figure 18-10. Typical 3-Volt Run Current versus Bus Frequency (25•C)

18.13 ADC10 Characteristics

Characteristic	Conditions	Symbol	Min	Typ(1)	Max	Unit	Comment
Supply voltage	Absolute	V_{DD}	2.7	—	5.5	V	
Supply Current ADLPC = 1 ADLSMP = 1 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 ADLPC = 1 ADLSMP = 0 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 ADLPC = 0 ADLSMP = 1 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 ADLPC = 0 ADLSMP = 0 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		
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	21	t_{ADCK} cycles	
	Long sample (ADLSMP = 1)		39	39	41		
Conversion time (4) 8-bit Mode	Short sample (ADLSMP = 0)	t_{ADC}	16	16	18	t_{ADCK} cycles	
	Long sample (ADLSMP = 1)		36	36	38		
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						

— Continued on next page

Characteristic	Conditions	Symbol	Min	Typ(1)	Max	Unit	Comment
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		
Bandgap voltage (3)(6)		V_{BG}	1.17	1.245	1.32	V	

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.
6. LVI must be enabled, (LVIPWRD = 0, in CONFIG1). Voltage input to ADCH4:0 = $1A, an ADC conversion on this channel allows user to determine supply voltage.

18.14 5.0-Volt SPI Characteristics

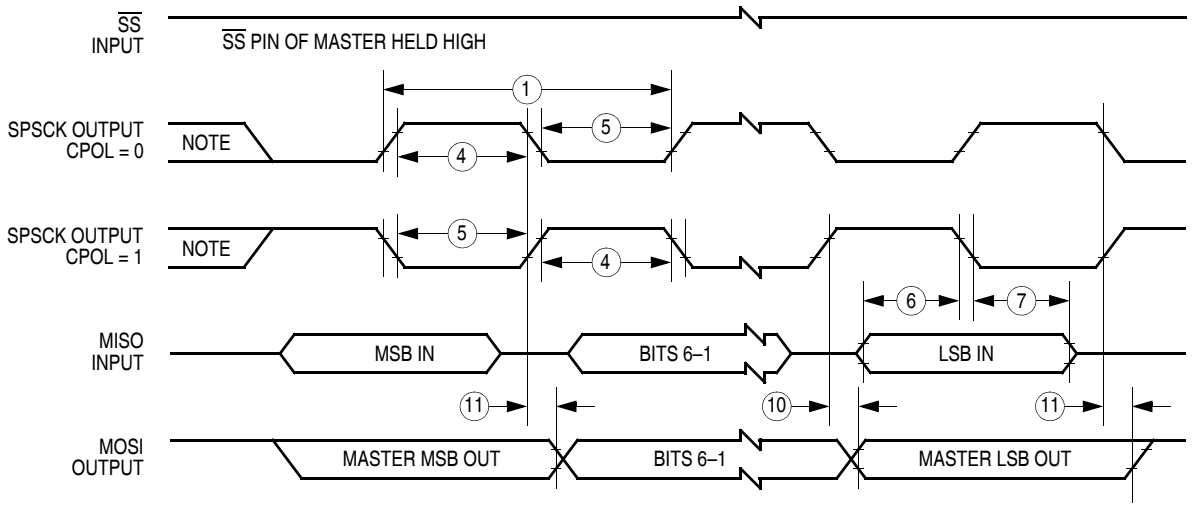
Diagram Number[1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	 $f_{OP(M)}$ $f_{OP(S)}$	 $f_{OP}/128$ dc	 $f_{OP}/2$ f_{OP}	 MHz MHz
1	Cycle time Master Slave	 $t_{cyc(M)}$ $t_{cyc(S)}$	 2 1	 128 —	 t_{cyc} t_{cyc}
2	Enable lead time	$t_{Lead(S)}$	1	—	t_{cyc}
3	Enable lag time	$t_{Lag(S)}$	1	—	t_{cyc}
4	Clock (SPSCK) high time Master Slave	 $t_{SCKH(M)}$ $t_{SCKH(S)}$	 t_{cyc} –25 1/2 t_{cyc} –25	 64 t_{cyc} —	 ns ns
5	Clock (SPSCK) low time Master Slave	 $t_{SCKL(M)}$ $t_{SCKL(S)}$	 t_{cyc} –25 1/2 t_{cyc} –25	 64 t_{cyc} —	 ns ns
6	Data setup time (inputs) Master Slave	 $t_{SU(M)}$ $t_{SU(S)}$	 30 30	 — —	 ns ns
7	Data hold time (inputs) Master Slave	 $t_{H(M)}$ $t_{H(S)}$	 30 30	 — —	 ns ns
8	Access time, slave[3] CPHA = 0 CPHA = 1	 $t_{A(CP0)}$ $t_{A(CP1)}$	 0 0	 40 40	 ns ns
9	Disable time, slave[4]	$t_{DIS(S)}$	—	40	ns
10	Data valid time, after enable edge Master Slave[5]	 $t_{V(M)}$ $t_{V(S)}$	 — —	 50 50	 ns ns
11	Data hold time, outputs, after enable edge Master Slave	 $t_{HO(M)}$ $t_{HO(S)}$	 0 0	 — —	 ns ns

1. Numbers refer to dimensions in Figure 18-11 and Figure 18-12.
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

18.15 3.0-Volt SPI Characteristics

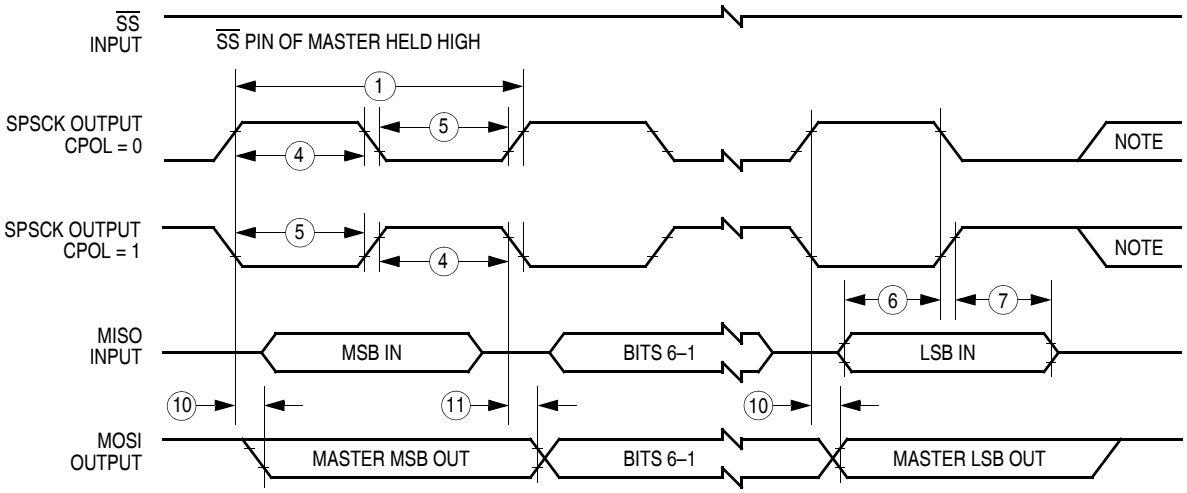
Diagram Number[1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	$f_{OP(M)}$ $f_{OP(S)}$	$f_{OP}/128$ DC	$f_{OP}/2$ f_{OP}	MHz MHz
1	Cycle time Master Slave	$t_{cyc(M)}$ $t_{cyc(S)}$	2 1	128 —	t_{cyc} t_{cyc}
2	Enable lead time	$t_{Lead(S)}$	1	—	t_{cyc}
3	Enable lag time	$t_{Lag(S)}$	1	—	t_{cyc}
4	Clock (SPSCK) high time Master Slave	$t_{SCKH(M)}$ $t_{SCKH(S)}$	t_{cyc} –35 1/2 t_{cyc} –35	64 t_{cyc} —	ns ns
5	Clock (SPSCK) low time Master Slave	$t_{SCKL(M)}$ $t_{SCKL(S)}$	t_{cyc} –35 1/2 t_{cyc} –35	64 t_{cyc} —	ns ns
6	Data setup time (inputs) Master Slave	$t_{SU(M)}$ $t_{SU(S)}$	40 40	— —	ns ns
7	Data hold time (inputs) Master Slave	$t_{H(M)}$ $t_{H(S)}$	40 40	— —	ns ns
8	Access time, slave[3] CPHA = 0 CPHA = 1	$t_{A(CP0)}$ $t_{A(CP1)}$	0 0	50 50	ns ns
9	Disable time, slave[4]	$t_{DIS(S)}$	—	50	ns
10	Data valid time, after enable edge Master Slave[5]	$t_{V(M)}$ $t_{V(S)}$	— —	60 60	ns ns
11	Data hold time, outputs, after enable edge Master Slave	$t_{HO(M)}$ $t_{HO(S)}$	0 0	— —	ns ns

1. Numbers refer to dimensions in Figure 18-11 and Figure 18-12.
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.

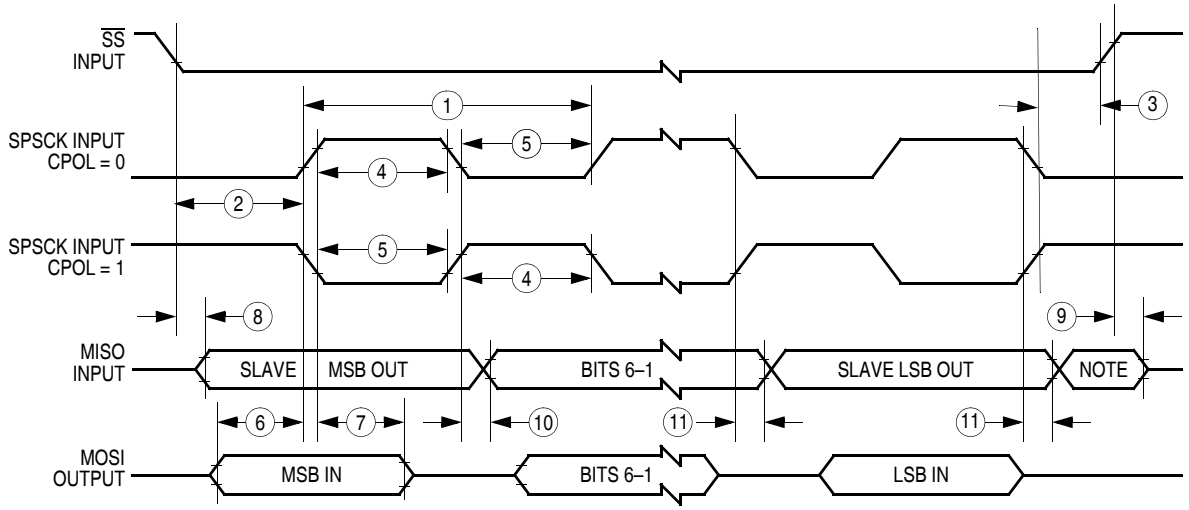
a) SPI Master Timing (CPHA = 0)



Note: This last clock edge is generated internally, but is not seen at the SPSCK pin.

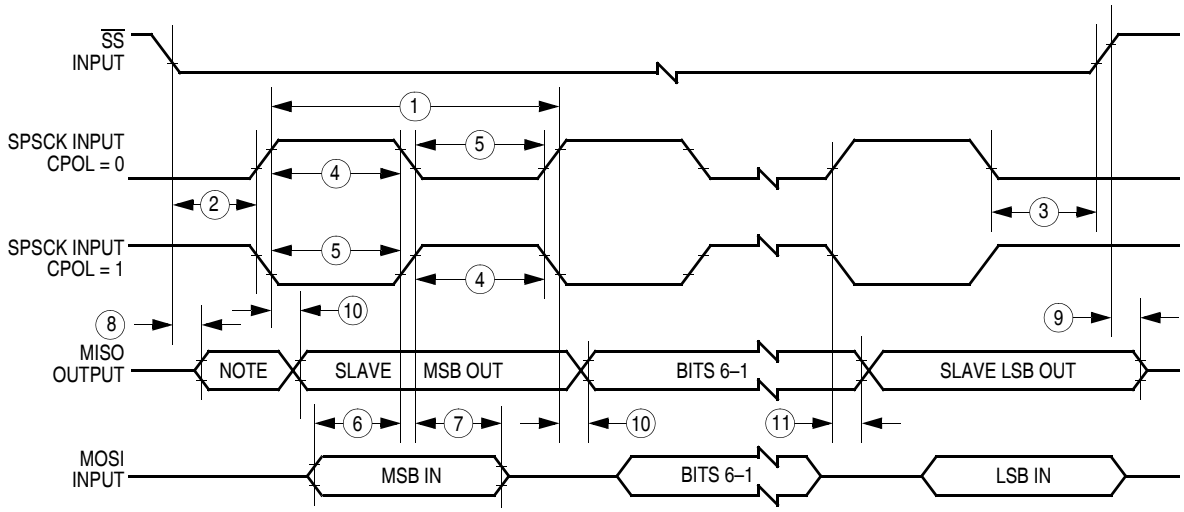
b) SPI Master Timing (CPHA = 1)

Figure 18-11. 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 18-12. SPI Slave Timing

18.16 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}.

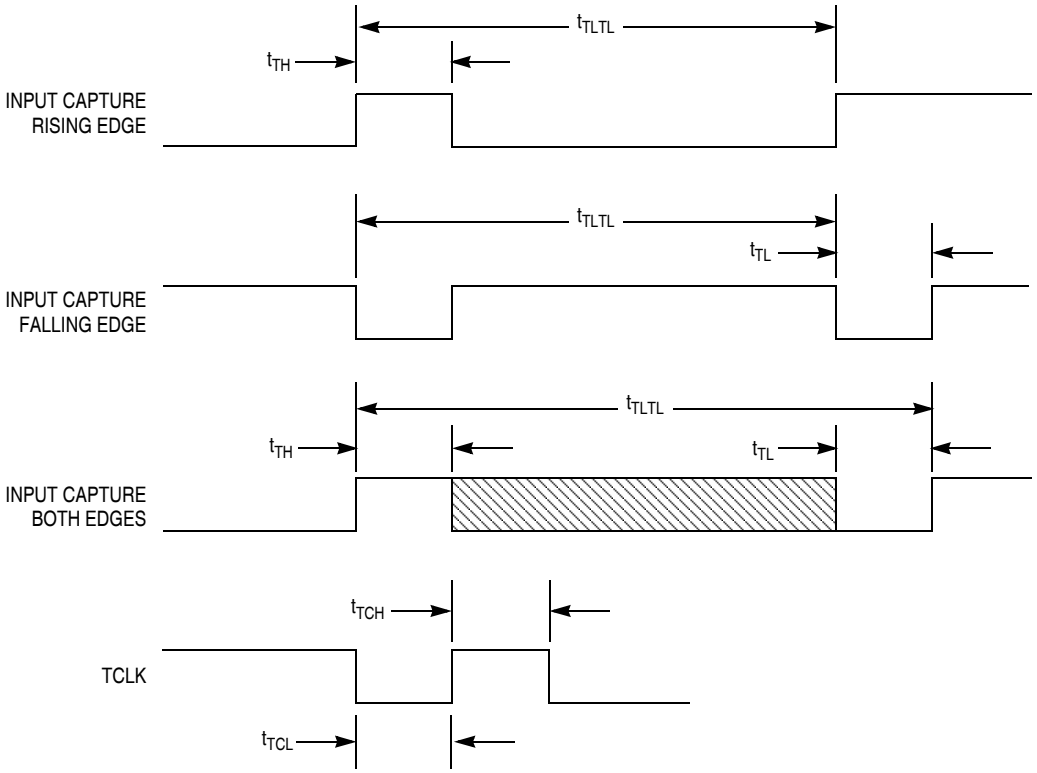


Figure 18-13. Timer Input Timing

18.17 Memory Characteristics

Characteristic	Symbol	Min	Typ(1)	Max	Unit
RAM data retention voltage (2)	V_{RDR}	1.3	—	—	V
FLASH program bus clock frequency	—	1	—	—	MHz
FLASH PGM/ERASE supply voltage (V_{DD})	$V_{PGM/ERASE}$	2.7	—	5.5	V
FLASH read bus clock frequency	f_{Read}(3)	0	—	8 M	Hz
FLASH page erase time <1 K cycles >1 K cycles	t_{Erase}	 0.9 3.6	 1 4	 1.1 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}(4)	1	—	—	µs
FLASH cumulative program HV period	t_{HV}(5)	—	—	4	ms
FLASH endurance(6)	—	10 k	100 k	—	Cycles
FLASH data retention time(7)	—	15	100	—	Years

1. Typical values are for reference only and are not tested in production.
2. Values are based on characterization results, not tested in production.
3. f_{Read} is defined as the frequency range for which the FLASH memory can be read.
4. 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.
5. t_{HV} is defined as the cumulative high voltage programming time to the same row before next erase. t_{HV} must satisfy this condition: $t_{NVS} + t_{NVH} + t_{PGS} + (t_{PROG} \times 32) \leq t_{HV}$ maximum.
6. Typical endurance was evaluated for this product family. For additional information on how Freescale Semiconductor defines *Typical Endurance*, please refer to Engineering Bulletin EB619.
7. Typical data retention values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25•C using the Arrhenius equation. For additional information on how Freescale Semiconductor defines *Typical Data Retention*, please refer to Engineering Bulletin EB618.

Chapter 19
Ordering Information and Mechanical Specifications

19.1 Introduction

This section contains order numbers for the MC68HC908QB8, MC68HC908QB4, and MC68HC908QY8. Dimensions are given for:

- 16-pin PDIP
- 16-pin SOIC
- 16-pin thin shrink small outline packages (TSSOP)

19.2 MC Order Numbers

Table 19-1. MC Order Numbers

MC Order Number	Package
MC908QB8	16-pins PDIP, SOIC, and TSSOP
MC908QB4	
MC908QY8	

Temperature and package designators:
C = –40•C to +85•C
V = –40•C to +105•C
M = –40•C to +125•C
P = Plastic dual in-line package (PDIP)
DW = Small outline integrated circuit package (SOIC)
DT = Thin shrink small outline package (TSSOP)
E = Lead free

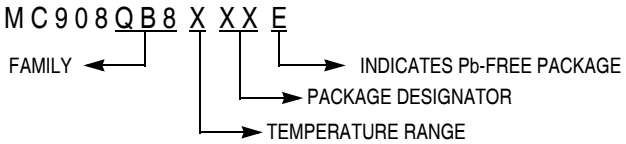


Figure 19-1. Device Numbering System

19.3 Package Dimensions

Refer to the following pages for detailed package dimensions.