

**MC68HC08QY4**  
**MC68HC08QT4**  
**MC68HC08QY2**  
**MC68HC08QT2**  
**MC68HC08QY1**  
**MC68HC08QT1**

Data Sheet


**M68HC08**  
**Microcontrollers**

MC68HC08QY4  
Rev. 2  
3/2010

[freescale.com](http://freescale.com)







**MC68HC08QY4**  
**MC68HC08QT4**  
**MC68HC08QY2**  
**MC68HC08QT2**  
**MC68HC08QY1**  
**MC68HC08QT1**

**Data Sheet**

---

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

<http://freescale.com/>

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

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

**MC68HC08QY/QT Family Data Sheet, Rev. 2**

## Revision History

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)
October, 2005	N/A	Initial release	N/A
October, 2006	1	<a href="#">1.7 Unused Pin Termination</a> — Added new section	<a href="#">20</a>
		<a href="#">Chapter 4 Auto Wakeup Module (AWU)</a> — Updated section	<a href="#">45</a>
		<a href="#">12.2 Unused Pin Termination</a> — Replaced note with new section	<a href="#">97</a>
		<a href="#">13.7.2 Stop Mode</a> — Corrected reference to BUSCLK4 to BUCLK2.	<a href="#">115</a>
		<a href="#">16.5 5-V DC Electrical Characteristics</a> — New values for DC injection current and ports Hi-Z leakage current.	<a href="#">148</a>
		<a href="#">16.8 3-V DC Electrical Characteristics</a> — New values for DC injection current and ports Hi-Z leakage current.	<a href="#">151</a>
March, 2010	2	Clarify internal oscillator trim register information.	<a href="#">27, 90, 95</a>

# List of Chapters

Chapter 1 General Description . . . . .	15
Chapter 2 Memory . . . . .	21
Chapter 3 10-Bit Analog-to-Digital Converter (ADC10) Module . . . . .	31
Chapter 4 Auto Wakeup Module (AWU) . . . . .	45
Chapter 5 Configuration Register (CONFIG) . . . . .	51
Chapter 6 Computer Operating Properly (COP). . . . .	55
Chapter 7 Central Processor Unit (CPU). . . . .	59
Chapter 8 External Interrupt (IRQ). . . . .	71
Chapter 9 Keyboard Interrupt Module (KBI). . . . .	77
Chapter 10 Low-Voltage Inhibit (LVI). . . . .	83
Chapter 11 Oscillator Module (OSC). . . . .	87
Chapter 12 Input/Output Ports (PORTS). . . . .	97
Chapter 13 System Integration Module (SIM). . . . .	103
Chapter 14 Timer Interface Module (TIM) . . . . .	119
Chapter 15 Development Support . . . . .	133
Chapter 16 Electrical Specifications . . . . .	147
Chapter 17 Ordering Information and Mechanical Specifications . . . . .	167



# Table of Contents

## Chapter 1 General Description

1.1	Introduction	15
1.2	Features	15
1.3	MCU Block Diagram	16
1.4	Pin Assignments	17
1.5	Pin Functions	17
1.6	Pin Function Priority	20
1.7	Unused Pin Termination	20

## Chapter 2 Memory

2.1	Introduction	21
2.2	Unimplemented Memory Locations	21
2.3	Reserved Memory Locations	21
2.4	Direct Page Registers	21
2.5	Unused ROM Locations	21
2.6	Random-Access Memory (RAM)	28
2.7	Read-Only Memory (ROM)	29

## Chapter 3 10-Bit Analog-to-Digital Converter (ADC10) Module

3.1	Introduction	31
3.2	Features	31
3.3	Functional Description	31
3.3.1	Clock Select and Divide Circuit	33
3.3.2	Input Select and Pin Control	34
3.3.3	Conversion Control	34
3.3.3.1	Initiating Conversions	34
3.3.3.2	Completing Conversions	34
3.3.3.3	Aborting Conversions	34
3.3.3.4	Total Conversion Time	35
3.3.4	Sources of Error	36
3.3.4.1	Sampling Error	36
3.3.4.2	Pin Leakage Error	36
3.3.4.3	Noise-Induced Errors	36
3.3.4.4	Code Width and Quantization Error	37
3.3.4.5	Linearity Errors	37
3.3.4.6	Code Jitter, Non-Monotonicity and Missing Codes	37

## Table of Contents

3.4	Interrupts	38
3.5	Low-Power Modes	38
3.5.1	Wait Mode	38
3.5.2	Stop Mode	38
3.6	ADC10 During Break Interrupts	38
3.7	I/O Signals	39
3.7.1	ADC10 Analog Power Pin ( $V_{DDA}$ )	39
3.7.2	ADC10 Analog Ground Pin ( $V_{SSA}$ )	39
3.7.3	ADC10 Voltage Reference High Pin ( $V_{REFH}$ )	39
3.7.4	ADC10 Voltage Reference Low Pin ( $V_{REFL}$ )	40
3.7.5	ADC10 Channel Pins ( $ADn$ )	40
3.8	Registers	40
3.8.1	ADC10 Status and Control Register	40
3.8.2	ADC10 Result High Register (ADRH)	42
3.8.3	ADC10 Result Low Register (ADRL)	42
3.8.4	ADC10 Clock Register (ADCLK)	43

## Chapter 4 Auto Wakeup Module (AWU)

4.1	Introduction	45
4.2	Features	45
4.3	Functional Description	46
4.4	Interrupts	46
4.5	Low-Power Modes	47
4.5.1	Wait Mode	47
4.5.2	Stop Mode	47
4.6	Registers	47
4.6.1	Port A I/O Register	47
4.6.2	Keyboard Status and Control Register	48
4.6.3	Keyboard Interrupt Enable Register	48
4.6.4	Configuration Register 2	49
4.6.5	Configuration Register 1	49

## Chapter 5 Configuration Register (CONFIG)

5.1	Introduction	51
5.2	Functional Description	51

## Chapter 6 Computer Operating Properly (COP)

6.1	Introduction	55
6.2	Functional Description	55
6.3	I/O Signals	56
6.3.1	BUSCLKX4	56
6.3.2	STOP Instruction	56
6.3.3	COPCTL Write	56



6.3.4	Power-On Reset	56
6.3.5	Internal Reset	56
6.3.6	COPD (COP Disable)	56
6.3.7	COPRS (COP Rate Select)	57
6.4	Interrupts	57
6.5	Monitor Mode	57
6.6	Low-Power Modes	57
6.6.1	Wait Mode	57
6.6.2	Stop Mode	57
6.7	COP Module During Break Mode	57
6.8	Register	57

## Chapter 7 Central Processor Unit (CPU)

7.1	Introduction	59
7.2	Features	59
7.3	CPU Registers	59
7.3.1	Accumulator	60
7.3.2	Index Register	60
7.3.3	Stack Pointer	61
7.3.4	Program Counter	61
7.3.5	Condition Code Register	62
7.4	Arithmetic/Logic Unit (ALU)	63
7.5	Low-Power Modes	63
7.5.1	Wait Mode	63
7.5.2	Stop Mode	63
7.6	CPU During Break Interrupts	63
7.7	Instruction Set Summary	64
7.8	Opcode Map	69

## Chapter 8 External Interrupt (IRQ)

8.1	Introduction	71
8.2	Features	71
8.3	Functional Description	71
8.3.1	MODE = 1	73
8.3.2	MODE = 0	73
8.4	Interrupts	74
8.5	Low-Power Modes	74
8.5.1	Wait Mode	74
8.5.2	Stop Mode	74
8.6	IRQ Module During Break Interrupts	74
8.7	I/O Signals	74
8.7.1	IRQ Input Pins ( $\overline{IRQ}$ )	74
8.8	Registers	75

## Chapter 9 Keyboard Interrupt Module (KBI)

9.1	Introduction	77
9.2	Features	77
9.3	Functional Description	77
9.3.1	Keyboard Operation	77
9.3.1.1	MODEK = 1	78
9.3.1.2	MODEK = 0	79
9.3.2	Keyboard Initialization	80
9.4	Interrupts	80
9.5	Low-Power Modes	80
9.5.1	Wait Mode	80
9.5.2	Stop Mode	80
9.6	KBI During Break Interrupts	80
9.7	I/O Signals	81
9.7.1	KBI Input Pins (KBIX:KBI0)	81
9.8	Registers	81
9.8.1	Keyboard Status and Control Register (KBSCR)	81
9.8.2	Keyboard Interrupt Enable Register (KBIER)	82
9.8.3	Keyboard Interrupt Polarity Register (KBIPR)	82

## Chapter 10 Low-Voltage Inhibit (LVI)

10.1	Introduction	83
10.2	Features	83
10.3	Functional Description	83
10.3.1	Polled LVI Operation	84
10.3.2	Forced Reset Operation	84
10.3.3	LVI Hysteresis	84
10.3.4	LVI Trip Selection	84
10.4	LVI Interrupts	85
10.5	Low-Power Modes	85
10.5.1	Wait Mode	85
10.5.2	Stop Mode	85
10.6	Registers	85

## Chapter 11 Oscillator Module (OSC)

11.1	Introduction	87
11.2	Features	87
11.3	Functional Description	87
11.3.1	Internal Signal Definitions	89
11.3.1.1	Oscillator Enable Signal (SIMOSCEN)	89
11.3.1.2	XTAL Oscillator Clock (XTALCLK)	89
11.3.1.3	RC Oscillator Clock (RCCLK)	89
11.3.1.4	Internal Oscillator Clock (INTCLK)	89

11.3.1.5	Bus Clock Times 4 (BUSCLKX4)	89
11.3.1.6	Bus Clock Times 2 (BUSCLKX2)	89
11.3.2	Internal Oscillator	89
11.3.2.1	Internal Oscillator Trimming	90
11.3.2.2	Internal to External Clock Switching	90
11.3.2.3	External to Internal Clock Switching	90
11.3.3	External Oscillator	90
11.3.4	XTAL Oscillator	91
11.3.5	RC Oscillator	92
11.4	Interrupts	92
11.5	Low-Power Modes	92
11.5.1	Wait Mode	92
11.5.2	Stop Mode	92
11.6	OSC During Break Interrupts	93
11.7	I/O Signals	93
11.7.1	Oscillator Input Pin (OSC1)	93
11.7.2	Oscillator Output Pin (OSC2)	93
11.8	Registers	94
11.8.1	Oscillator Status and Control Register	94
11.8.2	Oscillator Trim Register (OSCTRIM)	95

## Chapter 12 Input/Output Ports (PORTS)

12.1	Introduction	97
12.2	Unused Pin Termination	97
12.3	Port A	97
12.3.1	Port A Data Register	98
12.3.2	Data Direction Register A	98
12.3.3	Port A Input Pullup Enable Register	99
12.3.4	Port A Summary Table	100
12.4	Port B	100
12.4.1	Port B Data Register	100
12.4.2	Data Direction Register B	101
12.4.3	Port B Input Pullup Enable Register	102
12.4.4	Port B Summary Table	102

## Chapter 13 System Integration Module (SIM)

13.1	Introduction	103
13.2	$\overline{\text{RST}}$ and $\overline{\text{IRQ}}$ Pins Initialization	103
13.3	SIM Bus Clock Control and Generation	104
13.3.1	Bus Timing	105
13.3.2	Clock Start-Up from POR	105
13.3.3	Clocks in Stop Mode and Wait Mode	105
13.4	Reset and System Initialization	105
13.4.1	External Pin Reset	105
13.4.2	Active Resets from Internal Sources	106

## Table of Contents

13.4.2.1	Power-On Reset . . . . .	107
13.4.2.2	Computer Operating Properly (COP) Reset . . . . .	107
13.4.2.3	Illegal Opcode Reset . . . . .	107
13.4.2.4	Illegal Address Reset . . . . .	108
13.4.2.5	Low-Voltage Inhibit (LVI) Reset . . . . .	108
13.5	SIM Counter . . . . .	108
13.5.1	SIM Counter During Power-On Reset . . . . .	108
13.5.2	SIM Counter During Stop Mode Recovery . . . . .	108
13.5.3	SIM Counter and Reset States . . . . .	108
13.6	Exception Control . . . . .	109
13.6.1	Interrupts . . . . .	109
13.6.1.1	Hardware Interrupts . . . . .	109
13.6.1.2	SWI Instruction . . . . .	112
13.6.2	Interrupt Status Registers . . . . .	112
13.6.2.1	Interrupt Status Register 1 . . . . .	113
13.6.2.2	Interrupt Status Register 2 . . . . .	113
13.6.2.3	Interrupt Status Register 3 . . . . .	113
13.6.3	Reset . . . . .	114
13.6.4	Break Interrupts . . . . .	114
13.6.5	Status Flag Protection in Break Mode . . . . .	114
13.7	Low-Power Modes . . . . .	114
13.7.1	Wait Mode . . . . .	114
13.7.2	Stop Mode . . . . .	115
13.8	SIM Registers . . . . .	116
13.8.1	SIM Reset Status Register . . . . .	116
13.8.2	Break Flag Control Register . . . . .	117

## Chapter 14 Timer Interface Module (TIM)

14.1	Introduction . . . . .	119
14.2	Features . . . . .	119
14.3	Functional Description . . . . .	119
14.3.1	TIM Counter Prescaler . . . . .	119
14.3.2	Input Capture . . . . .	120
14.3.3	Output Compare . . . . .	120
14.3.3.1	Unbuffered Output Compare . . . . .	121
14.3.3.2	Buffered Output Compare . . . . .	122
14.3.4	Pulse Width Modulation (PWM) . . . . .	122
14.3.4.1	Unbuffered PWM Signal Generation . . . . .	123
14.3.4.2	Buffered PWM Signal Generation . . . . .	123
14.3.4.3	PWM Initialization . . . . .	124
14.4	Interrupts . . . . .	125
14.5	Low-Power Modes . . . . .	125
14.5.1	Wait Mode . . . . .	125
14.5.2	Stop Mode . . . . .	125
14.6	TIM During Break Interrupts . . . . .	125

14.7	I/O Signals	126
14.7.1	TIM Channel I/O Pins (TCH1:TCH0)	126
14.7.2	TIM Clock Pin (TCLK)	126
14.8	Registers	126
14.8.1	TIM Status and Control Register	126
14.8.2	TIM Counter Registers	128
14.8.3	TIM Counter Modulo Registers	128
14.8.4	TIM Channel Status and Control Registers	129
14.8.5	TIM Channel Registers	131

## Chapter 15 Development Support

15.1	Introduction	133
15.2	Break Module (BRK)	133
15.2.1	Functional Description	133
15.2.1.1	Flag Protection During Break Interrupts	135
15.2.1.2	TIM During Break Interrupts	135
15.2.1.3	COP During Break Interrupts	135
15.2.2	Break Module Registers	136
15.2.2.1	Break Status and Control Register	136
15.2.2.2	Break Address Registers	136
15.2.2.3	Break Auxiliary Register	137
15.2.2.4	Break Status Register	137
15.2.2.5	Break Flag Control Register	137
15.2.3	Low-Power Modes	138
15.3	Monitor Module (MON)	138
15.3.1	Functional Description	138
15.3.1.1	Normal Monitor Mode	141
15.3.1.2	Monitor Vectors	141
15.3.1.3	Data Format	142
15.3.1.4	Break Signal	142
15.3.1.5	Baud Rate	142
15.3.1.6	Commands	142
15.3.2	Security	146

## Chapter 16 Electrical Specifications

16.1	Introduction	147
16.2	Absolute Maximum Ratings	147
16.3	Functional Operating Range	148
16.4	Thermal Characteristics	148
16.5	5-V DC Electrical Characteristics	148
16.6	Typical 5-V Output Drive Characteristics	150
16.7	5-V Control Timing	151
16.8	3-V DC Electrical Characteristics	151
16.9	Typical 3-V Output Drive Characteristics	153
16.10	3-V Control Timing	154

## Table of Contents

16.11	1.8-V to 3.6-V DC Electrical Characteristics . . . . .	155
16.12	Typical 2-V Output Drive Characteristics . . . . .	156
16.13	1.8-V to 3.6-V Control Timing . . . . .	157
16.14	Oscillator Characteristics . . . . .	158
16.15	Supply Current Characteristics . . . . .	161
16.16	ADC10 Characteristics . . . . .	164
16.17	Timer Interface Module Characteristics . . . . .	166
16.18	Memory Characteristics . . . . .	166

## Chapter 17

### Ordering Information and Mechanical Specifications

17.1	Introduction . . . . .	167
17.2	MC Order Numbers . . . . .	167
17.3	Package Dimensions . . . . .	167

# Chapter 1

## General Description

### 1.1 Introduction

The MC68HC08QY4 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	ROM Memory Size	Analog-to-Digital Converter	Pin Count
MC68HC08QT1	1536 bytes	—	8 pins
MC68HC08QT2	1536 bytes	4 ch, 10 bit	8 pins
MC68HC08QT4	4096 bytes	4 ch, 10 bit	8 pins
MC68HC08QY1	1536 bytes	—	16 pins
MC68HC08QY2	1536 bytes	6 ch, 10 bit	16 pins
MC68HC08QY4	4096 bytes	6 ch, 10 bit	16 pins

### 1.2 Features

Features include:

- High-performance M68HC08 CPU core
- Fully upward-compatible object code with M68HC05 Family
- 5-V, 3-V, and 2-V operating voltages ( $V_{DD}$ )
- 8-MHz internal bus operation at 5 V, 4-MHz at 3 V, 2-MHz at 2 V
- Trimmable internal oscillator
  - Software selectable 1 MHz, 2 MHz, or 3.2 MHz internal bus operation
  - 8-bit trim capability
  - $\pm 25\%$  untrimmed
  - Trimmable to approximately 0.4%<sup>(1)</sup>
- 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 read-only memory (ROM)
- On-chip random-access memory (RAM)
- 2-channel, 16-bit timer interface module (TIM)

1. See [16.14 Oscillator Characteristics](#) for internal oscillator specifications

## General Description

- 6-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
  - Six shared with ADC
  - Two shared with TIM
  - 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 ( $\overline{\text{IRQ}}$ )
- 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
- MC68HC08QY4, MC68HC08QY2, and MC68HC08QY1 are available in these packages:
  - 16-pin small outline integrated circuit (SOIC) package
  - 16-pin thin shrink small outline package (TSSOP)
- MC68HC08QT4, MC68HC08QT2, and MC68HC08QT1 are available in 8-pin SOIC packages

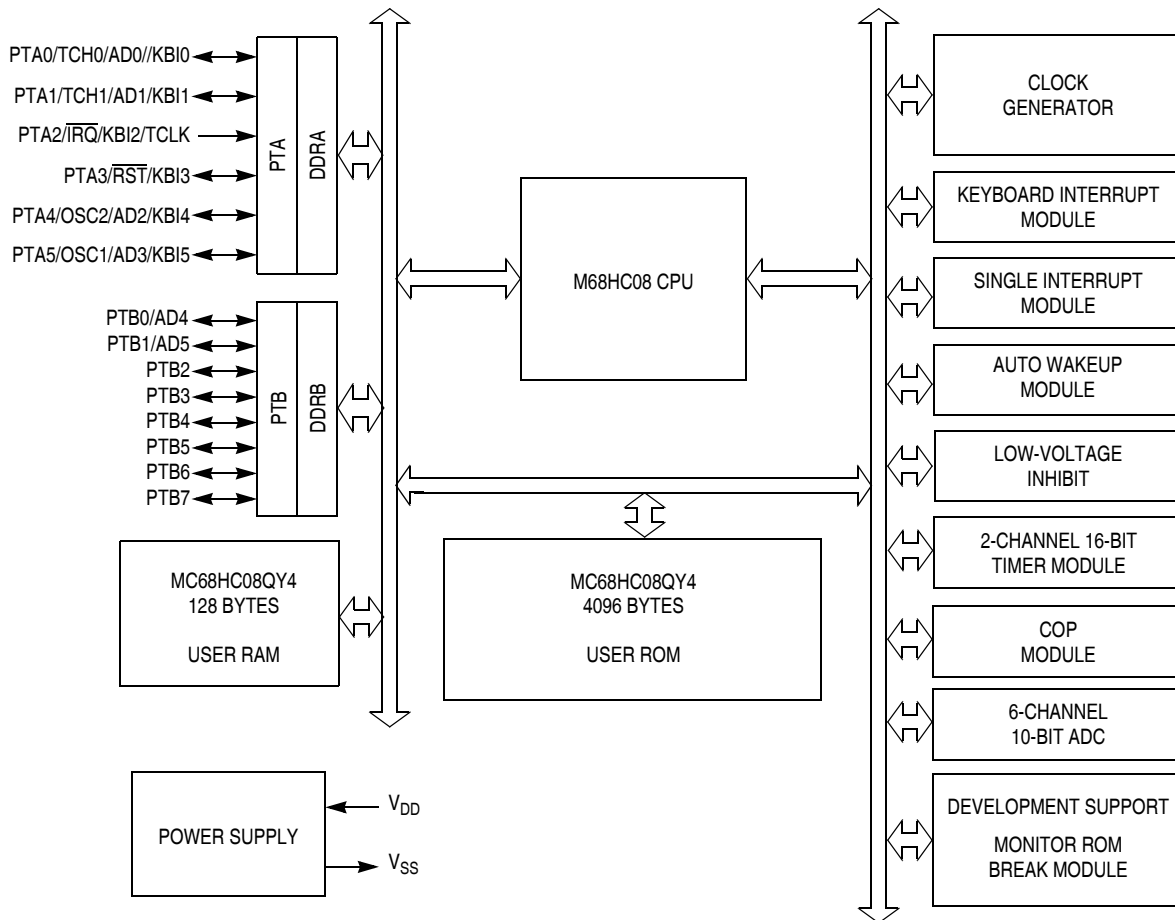
Features of the CPU08 include the following:

- 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  $8 \times 8$  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 MC68HC08QY4.





$\overline{\text{RST}}$ ,  $\overline{\text{IRQ}}$ : Pins have internal pull up device  
 All port pins have programmable pull up device  
 PTA[0:5]: Higher current sink and source capability  
 PTB[0:7]: Not available on 8-pin devices

**Figure 1-1. Block Diagram**

## 1.4 Pin Assignments

The MC68HC08QT4, MC68HC08QT2, and MC68HC08QT1 are available in 8-pin packages and the MC68HC08QY4, MC68HC08QY2, and MC68HC08QY1 in 16-pin packages. [Figure 1-2](#) shows the pin assignment for these packages.

## 1.5 Pin Functions

[Table 1-2](#) provides a description of the pin functions.

## General Description

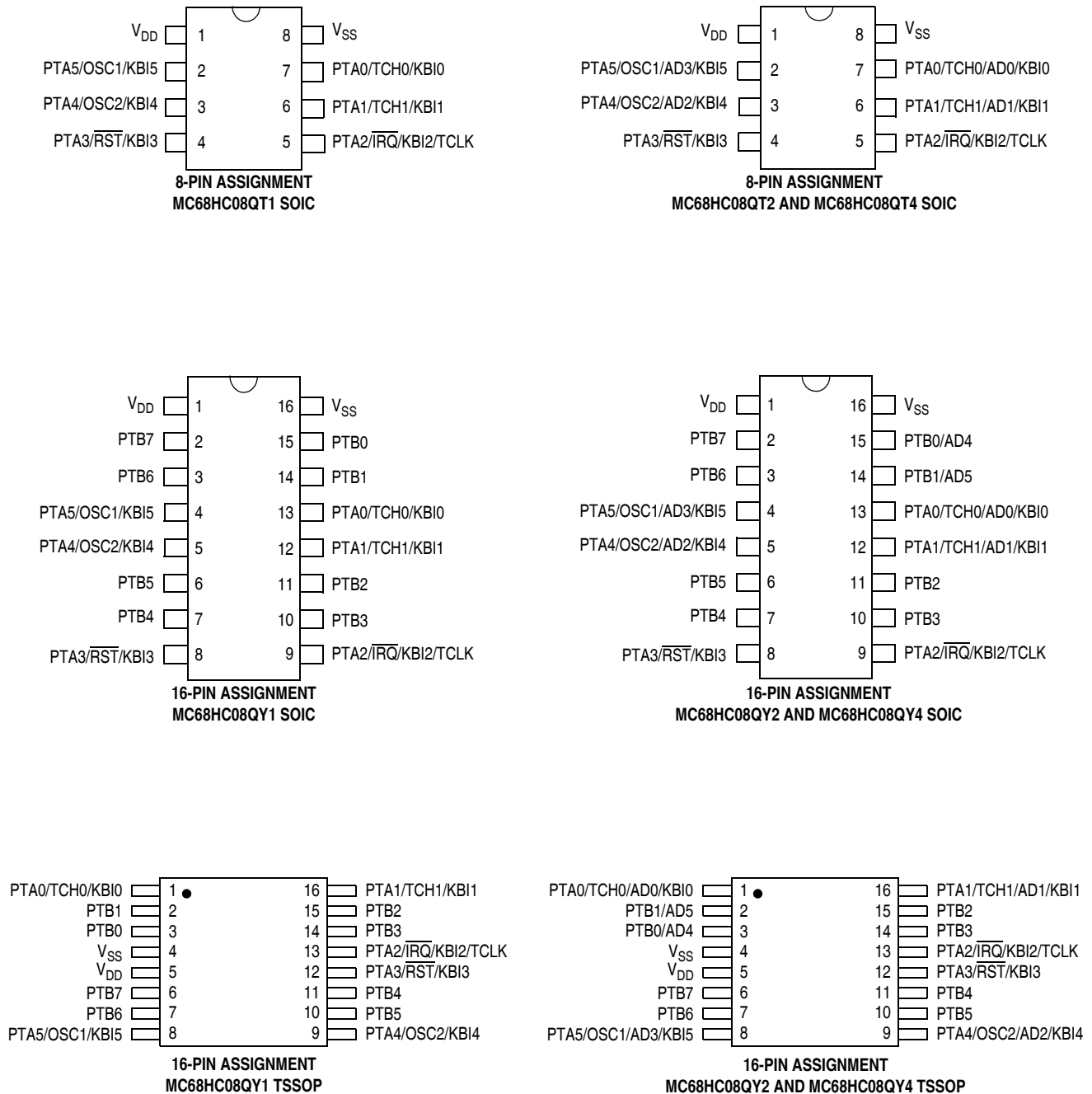


Figure 1-2. MCU Pin Assignments

Table 1-2. Pin Functions

Pin Name	Description	Input/Output
V <sub>DD</sub>	Power supply	Power
V <sub>SS</sub>	Power supply ground	Power
PTA0	PTA0 — General purpose I/O port	Input/Output
	AD0 — A/D channel 0 input	Input
	TCH0 — Timer Channel 0 I/O	Input/Output
	KBI0 — Keyboard interrupt input 0	Input
PTA1	PTA1 — General purpose I/O port	Input/Output
	AD1 — A/D channel 1 input	Input
	TCH1 — Timer Channel 1 I/O	Input/Output
	KBI1 — Keyboard interrupt input 1	Input
PTA2	PTA2 — General purpose input-only port	Input
	$\overline{\text{IRQ}}$ — 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
	$\overline{\text{RST}}$ — 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 <sup>(1)</sup>	PTB0 — General purpose I/O port	Input/Output
	AD4 — A/D channel 4 input	Input
PTB1 <sup>(1)</sup>	PTB1 — General purpose I/O port	Input/Output
	AD5 — A/D channel 5 input	Input
PTB[2:7] <sup>(1)</sup>	6 general-purpose I/O ports	Input/Output

1. The PTB pins are not available on the 8-pin packages.

## 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 <sup>(1)</sup>	AD0 → TCH0 → KBI0 → PTA0
PTA1 <sup>(1)</sup>	AD1 → TCH1 → KBI1 → PTA1
PTA2	$\overline{\text{IRQ}}$ → TCLK → KBI2 → PTA2
PTA3	$\overline{\text{RST}}$ → KBI3 → PTA3
PTA4 <sup>(1)</sup>	OSC2 → AD2 → KBI4 → PTA4
PTA5 <sup>(1)</sup>	OSC1 → AD3 → KBI5 → PTA5
PTB0 <sup>(1)</sup>	AD4 → PTB0
PTB1 <sup>(1)</sup>	AD5 → PTB1

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.

## 1.7 Unused Pin Termination

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

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

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

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

# Chapter 2

## Memory

### 2.1 Introduction

The central processor unit (CPU08) can address 64 Kbytes of memory space. The memory map is 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](#), reserved locations are marked with the word reserved or with the letter R.

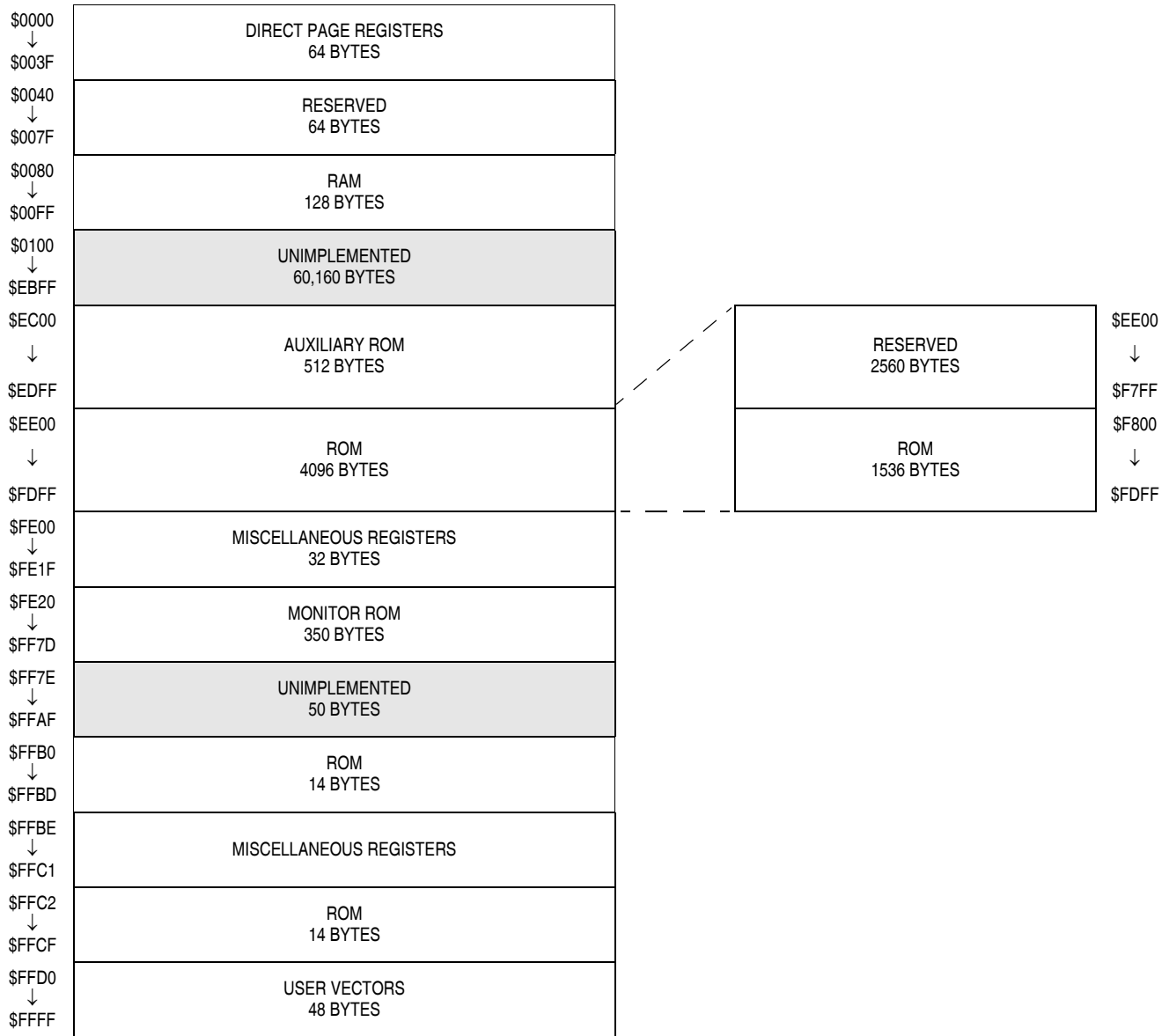
### 2.4 Direct Page Registers

[Figure 2-2](#) shows the memory mapped registers of the MC68HC08QY/QT Family. 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.

### 2.5 Unused ROM Locations

Any location in the ROM memory map that is not specified in the user supplied S-record will be factory programmed to an \$83, which is an SWI opcode. The user should provide an interrupt service routine address at the SWI interrupt vector (\$FFFC/D) that points to an appropriate error routine.

## Memory



MC68HC08QT4, MC68HC08QY4  
Memory Map

MC68HC08QT1, MC68HC08QT2,  
MC68HC08QY1, and MC68HC08QY2  
Memory Map

**Figure 2-1. Memory Map**

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
\$0000	Port A Data Register (PTA) <a href="#">See page 98.</a>	Read:	R	AWUL	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by reset							
\$0001	Port B Data Register (PTB) <a href="#">See page 100.</a>	Read:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Write:								
		Reset:	Unaffected by reset							
\$0002 ↓ \$0003	Reserved									
\$0004	Data Direction Register A (DDRA) <a href="#">See page 98.</a>	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) <a href="#">See page 101.</a>	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) <a href="#">See page 99.</a>	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) <a href="#">See page 102.</a>	Read:	PTBPUE7	PTBPUE6	PTBPUE5	PTBPUE4	PTBPUE3	PTBPUE2	PTBPUE1	PTBPUE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$0006 ↓ \$000A	Reserved									
\$001A	Keyboard Status and Control Register (KBSCR) <a href="#">See page 81.</a>	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 (KBIE) <a href="#">See page 82.</a>	Read:	0	AWUIE	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$001C	Keyboard Interrupt Polarity Register (KBIPOL) <a href="#">See page 82.</a>	Read:	0	0	KBIP5	KBIP4	KBIP3	KBIP2	KBIP1	KBIP0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

  = Unimplemented     
 R = Reserved     
 U = Unaffected

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


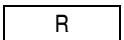
## Memory

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0	
\$001D	IRQ Status and Control Register (INTSCR) <a href="#">See page 75.</a>	Read:	0	0	0	0	IRQF	0	IMASK	MODE	
		Write:						ACK			
		Reset:	0	0	0	0	0	0	0	0	0
\$001E	Configuration Register 2 (CONFIG2) <sup>(1)</sup> <a href="#">See page 51.</a>	Read:	IRQPUD	IRQEN	R	R	R	R	OSCENIN-STOP	RSTEN	
		Write:									
		Reset:	0	0	0	0	0	0	0	0	0 <sup>(2)</sup>
		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) <sup>(1)</sup> <a href="#">See page 52.</a>	Read:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVITRIP	SSREC	STOP	COPD	
		Write:									
		Reset:	0	0	0	0	0 <sup>(2)</sup>	0	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) <a href="#">See page 126.</a>	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0	
		Write:	0			TRST					
		Reset:	0	0	1	0	0	0	0	0	0
\$0021	TIM Counter Register High (TCNTH) <a href="#">See page 128.</a>	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	0
\$0022	TIM Counter Register Low (TCNTL) <a href="#">See page 128.</a>	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	0
\$0023	TIM Counter Modulo Register High (TMODH) <a href="#">See page 128.</a>	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	
		Write:									
		Reset:	1	1	1	1	1	1	1	1	1
\$0024	TIM Counter Modulo Register Low (TMODL) <a href="#">See page 128.</a>	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	1
\$0025	TIM Channel 0 Status and Control Register (TSC0) <a href="#">See page 129.</a>	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX	
		Write:	0								
		Reset:	0	0	0	0	0	0	0	0	0
\$0026	TIM Channel 0 Register High (TCH0H) <a href="#">See page 131.</a>	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) <a href="#">See page 131.</a>	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
		Write:									
		Reset:	Indeterminate after reset								
		<div style="display: inline-block; width: 20px; height: 10px; background-color: #cccccc; border: 1px solid black;"></div> = Unimplemented		<div style="display: inline-block; width: 20px; height: 10px; background-color: #cccccc; border: 1px solid black; text-align: center; font-size: 8px;">R</div> = 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	
\$0028	TIM Channel 1 Status and Control Register (TSC1) <a href="#">See page 129.</a>	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) <a href="#">See page 131.</a>	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) <a href="#">See page 131.</a>	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
\$002B ↓ \$0035	Reserved									
\$0036	Oscillator Status and Control Register (OSCS) <a href="#">See page 94.</a>	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) <a href="#">See page 95.</a>	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) <a href="#">See page 40.</a>	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) <a href="#">See page 42.</a>	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) <a href="#">See page 42.</a>	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) <a href="#">See page 43.</a>	Read:	ADLPC	ADIV1	ADIV0	ADICLK	MODE1	MODE0	ADLSMP	ACLKEN
		Write:								
		Reset:	0	0	0	0	0	0	0	0

 = Unimplemented       = Reserved      U = Unaffected

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

## Memory

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
\$FE00	Break Status Register (BSR) <a href="#">See page 136.</a>	Read:	R	R	R	R	R	R	SBSW	R
		Write:							0	
		Reset:							0	
\$FE01	SIM Reset Status Register (SRSR) <a href="#">See page 116.</a>	Read:	POR	PIN	COP	ILOP	ILAD	MODRST	LVI	0
		Write:								
		POR:	1	0	0	0	0	0	0	0
\$FE02	Break Auxiliary Register (BRKAR) <a href="#">See page 137.</a>	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) <a href="#">See page 137.</a>	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
\$FE04	Interrupt Status Register 1 (INT1) <a href="#">See page 75.</a>	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) <a href="#">See page 75.</a>	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) <a href="#">See page 75.</a>	Read:	IF22	IF21	IF20	IF19	IF18	IF17	IF16	IF15
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
\$FE07	Reserved									
\$FE08	Non-Volatile Data Memory Status and Control (NVDMSC)	Read:	Reserved for Factory Programming of On-Chip Fuse							
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$FE09	Break Address High Register (BRKH) <a href="#">See page 136.</a>	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) <a href="#">See page 136.</a>	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) <a href="#">See page 137.</a>	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented    
 = 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	
\$FE0C	LVI Status Register (LVISR) <i>See page 85.</i>	Read:	LVIOUT	0	0	0	0	0	0	R
		Write:								
		Reset:	0	0	0	0	0	0	0	0
\$FE0D ↓ \$FE0F	Reserved									
\$FEBE ↓ \$FEBF	Reserved									
\$FFC0	Internal Oscillator Trim (Factory Programmed)	Read:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
		Write:								
		Reset:	Resets to factory programmed value							
\$FFC1	Reserved									
\$FFFF	COP Control Register (COPCTL) <i>See page 57.</i>	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  Highest	IF22– IF16	\$FFD0,1– \$FFDC,D	Not used
	IF15	\$FFDE,F	ADC conversion complete vector
	IF14	\$FFE0,1	Keyboard vector
	IF13	—	Not used
	IF12	—	Not used
	IF11	—	Not used
	IF10	—	Not used
	IF9	—	Not used
	IF8	\$FFEC,D	Reserved for test. Factory programmed with \$00 at location \$FFEC and \$83 at location \$FFED
	IF7	—	Not used
	IF6	—	Not used
	IF5	\$FFF2,3	TIM overflow vector
	IF4	\$FFF4,5	TIM channel 1 vector
	IF3	\$FFF6,7	TIM channel 0 vector
	IF2	—	Not used
	IF1	\$FFFA,B	$\overline{\text{IRQ}}$ vector
	—	\$FFFC,D	SWI vector
	—	\$FFFE,F	Reset vector

## 2.6 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)
```

---

## 2.7 Read-Only Memory (ROM)

The ROM memory is intended primarily for program storage.

Consult [Table 1-1. Summary of Device Variations](#) for ROM memory sizes available and reference [Figure 2-1](#) for user program ROM locations.

Forty-eight bytes of user vectors, \$FFD0–\$FFFF, are dedicated to user-defined reset and interrupt vectors, not all 48 bytes are used as interrupt vectors for this device. See [Table 2-1](#) for actual vectors used on this MCU.

Security has been incorporated into the MC68HC08QY/QT Family to prevent external viewing of the ROM contents. This feature ensures that customer-developed software remains proprietary. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the ROM difficult for unauthorized users.



# Chapter 3

## 10-Bit 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.

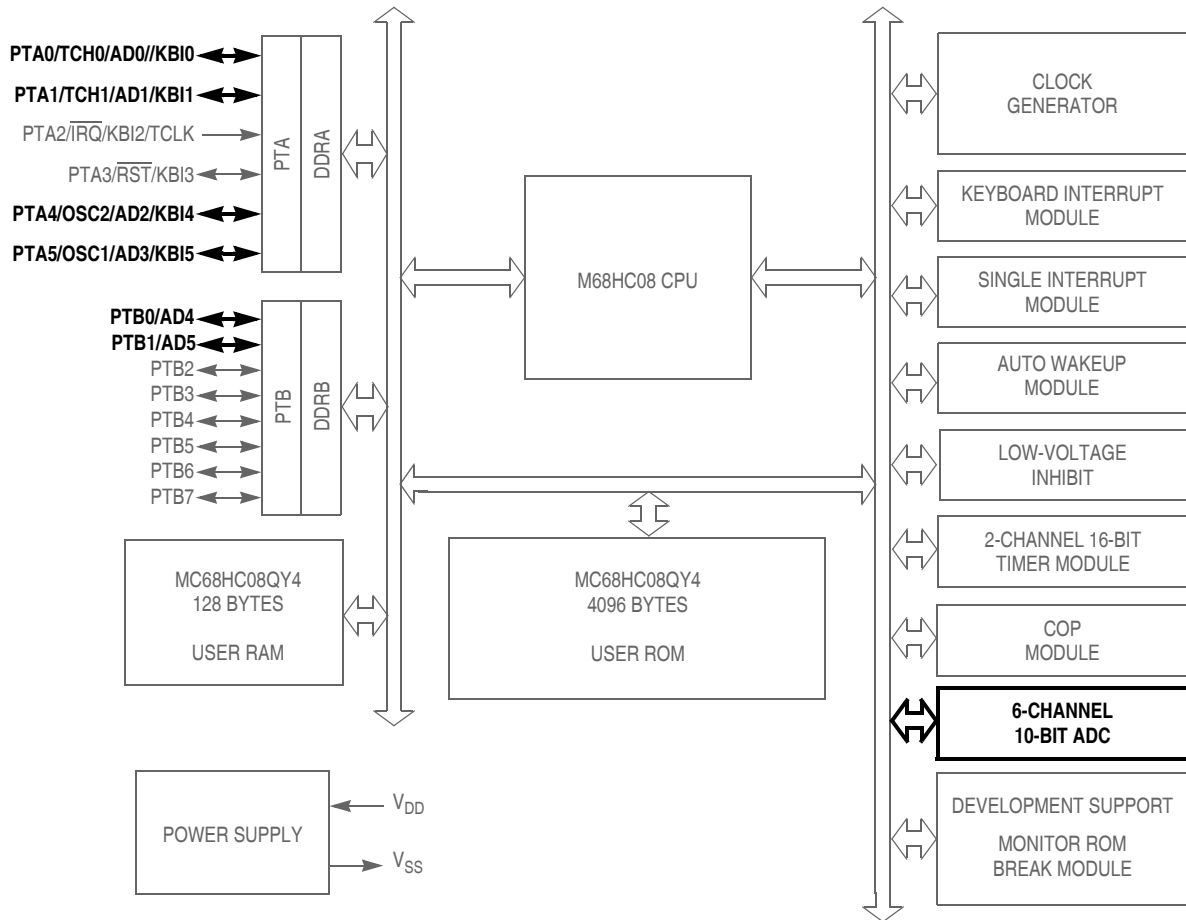
[Figure 3-2](#) shows a block diagram of the ADC10

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.*

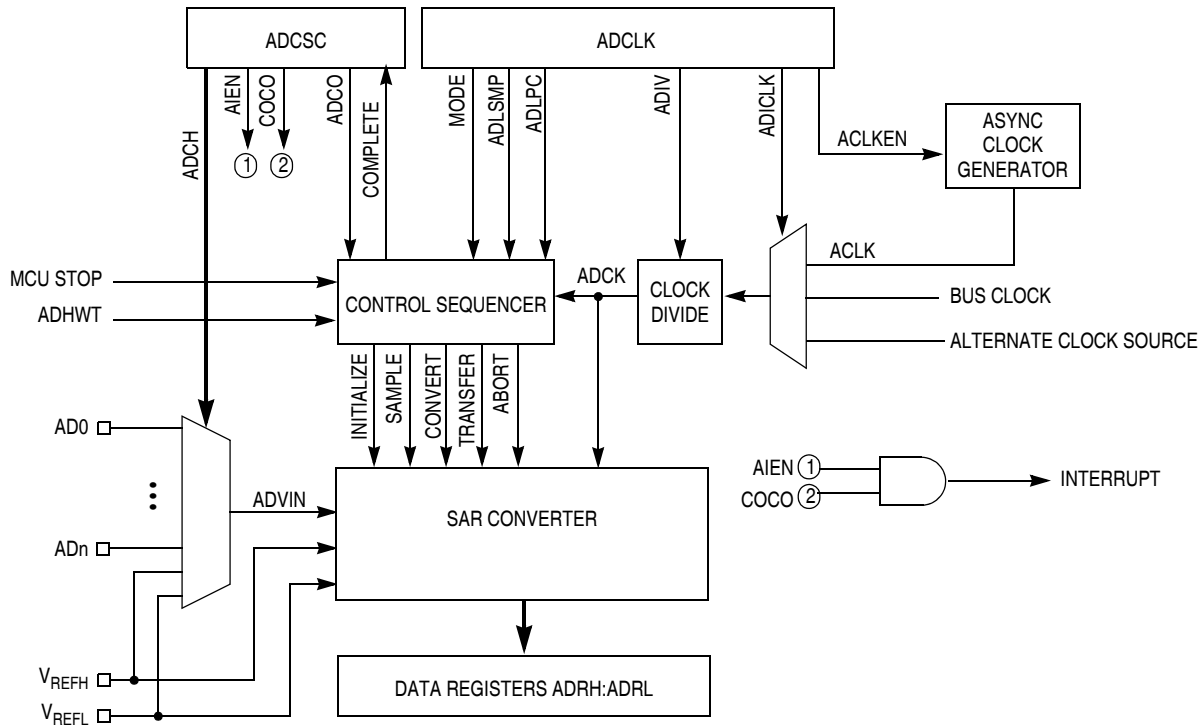
## 10-Bit Analog-to-Digital Converter (ADC10) Module



$\overline{RST}$ ,  $\overline{IRQ}$ : Pins have internal pull up device  
 All port pins have programmable pull up device  
 PTA[0:5]: Higher current sink and source capability  
 PTB[0:7]: Not available on 8-pin devices

**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 (ADV<sub>IN</sub>) is converted by a successive approximation algorithm into a 10-bit digital result. When the conversion is completed, the result is placed in the data registers (ADRH and ADRL). In 8-bit mode, the result is rounded to 8 bits and placed in ADRL. The conversion complete flag is then set and an interrupt is generated if the interrupt has been enabled.

### 3.3.1 Clock Select and Divide Circuit

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

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

Whichever clock is selected, its frequency must fall within the acceptable frequency range for ADCK. If the available clocks are too slow, the ADC10 will not perform according to specifications. If the available

clocks are too fast, then the clock must be divided to the appropriate frequency. This divider is specified by the ADIV[1:0] bits and can be divide-by 1, 2, 4, or 8.

### 3.3.2 Input Select and Pin Control

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

### 3.3.3 Conversion Control

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

#### 3.3.3.1 Initiating Conversions

A conversion is initiated:

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

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

#### 3.3.3.2 Completing Conversions

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

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

#### 3.3.3.3 Aborting Conversions

Any conversion in progress will be aborted when:

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

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

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

### 3.3.3.4 Total Conversion Time

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

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

Conversion Mode	ACLKEN	Maximum Conversion Time
8-Bit Mode (short sample — ADLSMP = 0):		
Single or 1st continuous	0	18 ADCK + 3 bus clock
Single or 1st continuous	1	18 ADCK + 3 bus clock + 5 $\mu$ s
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 $\mu$ s
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 $\mu$ s
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 $\mu$ s
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 k $\Omega$  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 k $\Omega$ . 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  $V_{ADV_{IN}} / (4096 * I_{Leak})$  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 0.1 $\mu$ F low-ESR capacitor from  $V_{REFH}$  to  $V_{REFL}$  (if available).
- There is a 0.1 $\mu$ F low-ESR capacitor from  $V_{DDA}$  to  $V_{SSA}$  (if available).
- If inductive isolation is used from the primary supply, an additional 1 $\mu$ F capacitor is placed from  $V_{DDA}$  to  $V_{SSA}$  (if available).
- $V_{SSA}$  and  $V_{REFL}$  (if available) is connected to  $V_{SS}$  at a quiet point in the ground plane.
- The MCU is placed in wait mode immediately after initiating the conversion (next instruction after write to ADCSC).
- There is no I/O switching, input or output, on the MCU during the conversion.

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

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

### 3.3.4.4 Code Width and Quantization Error

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

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

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  $\pm 1/2\text{LSB}$  in 8- or 10-bit mode. As a consequence, however, the code width of the first (\$000) conversion is only  $1/2\text{LSB}$  and the code width of the last (\$FF or \$3FF) is  $1.5\text{LSB}$ .

### 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 ( $E_{\text{ZS}}$ ) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width ( $1/2\text{LSB}$ ). Note, if the first conversion is \$001, then the difference between the actual \$001 code width and its ideal ( $1\text{LSB}$ ) is used.
- Full-Scale Error ( $E_{\text{FS}}$ ) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width ( $1.5\text{LSB}$ ). Note, if the last conversion is \$3FE, then the difference between the actual \$3FE code width and its ideal ( $1\text{LSB}$ ) 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  $\pm 1/2\text{LSB}$  but will increase with noise.
- Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage.
- Missing codes are those which are never converted for any input value.

In 8-bit or 10-bit mode, the ADC10 is guaranteed to be monotonic and to have no missing codes.

## 3.4 Interrupts

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

## 3.5 Low-Power Modes

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

### 3.5.1 Wait Mode

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

### 3.5.2 Stop Mode

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

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

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

## 3.6 ADC10 During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. BCFE in the break flag control register (BFCR) enables software to clear status bits during the break state. See BFCR in the SIM section of this data sheet.

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

To protect status bits during the break state, write a 0 to BCFE. With BCFE cleared (its default state), software can read and write registers during the break state without affecting status bits. Some status bits have a two-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is cleared. After the break, doing the second step clears the status bit.

## 3.7 I/O Signals

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

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

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

**NOTE**

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

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

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

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

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

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

**NOTE**

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

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the  $V_{REFH}$  and  $V_{REFL}$  loop. The best external component to meet this current demand is a 0.1  $\mu\text{F}$  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 ( $AD_n$ )

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  $\mu$ F capacitors with good high-frequency characteristics are sufficient. These capacitors are not necessary in all cases, but when used they must be placed as close as possible to the package pins and be referenced to  $V_{SSA}$ .

## 3.8 Registers

These registers control and monitor operation of the ADC10:

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

### 3.8.1 ADC10 Status and Control Register

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

	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

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

#### **COCO** — Conversion Complete Bit

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

- 1 = Conversion completed
- 0 = Conversion not completed

#### **AIEN** — ADC10 Interrupt Enable Bit

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

- 1 = ADC10 interrupt enabled
- 0 = ADC10 interrupt disabled

#### **ADCO** — ADC10 Continuous Conversion Bit

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



stop mode (if ACLKEN is clear), ADCLK is written, or until ADCSC is written again. If stop is entered (with ACLKEN low), continuous conversions will cease and can be restarted only with a write to ADCSC. Any write to ADCSC with ADCO set and the ADCH bits not all 1s will abort the current conversion and begin continuous conversions.

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

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

1 = Continuous conversion following a write to ADCSC

0 = One conversion following a write to ADCSC

#### ADCH[4:0] — Channel Select Bits

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

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

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	Input Select <sup>(1)</sup>
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	Unused
Continuing through					Unused
1	1	0	0	1	Unused
1	1	0	1	0	BANDGAP REF <sup>(2)</sup>
1	1	0	1	1	Reserved
1	1	1	0	0	Reserved
1	1	1	0	1	V <sub>REFH</sub>
1	1	1	1	0	V <sub>REFL</sub>
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)

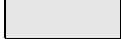
This register holds the MSBs of the result and is updated each time a conversion completes. All other bits read as 0s. Reading ADRH prevents the ADC10 from transferring subsequent conversion results into the result registers until ADRL is read. If ADRL is not read until the after next conversion is completed, 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:	ADLPC	ADIV1	ADIV0	ADICLK	MODE1	MODE0	ADLSMP	ACLKEN
Write:								
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  $\pm 1/2\text{LSB}$ .

Reset returns 8-bit mode.

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

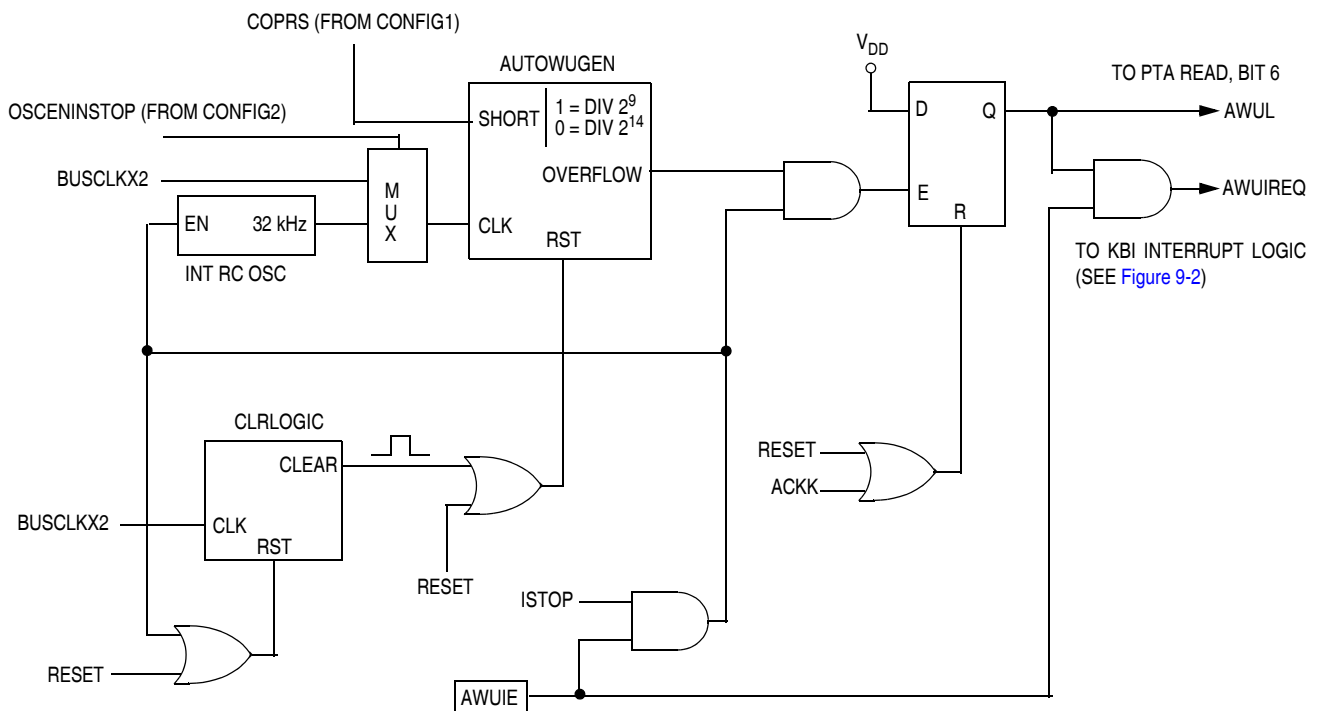


Figure 4-1. Auto Wakeup Interrupt Request Generation Logic

### 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 (INTRCOSC) 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 request:

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 is 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.

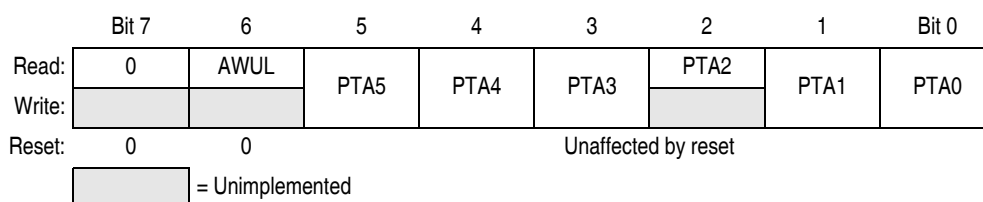
## 4.6 Registers

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.



**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 conjunction with the auto wakeup feature. To see a description of these bits, see [12.3.1 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 conjunction 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 conjunction 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:	IRQPUD	IRQEN	R	R	R	R	OSCENINSTOP	RSTEN
Write:								
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, and RSTEN bits are not used in conjunction 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:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVITRIP	SSREC	STOP	COPD
Write:								
Reset: POR:	0	0	0	0	U	0	0	0
	0	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 =  $512 \times (\text{INTRCOSC or BUSCLKX2})$

0 = Auto wakeup long cycle =  $16,384 \times (\text{INTRCOSC or BUSCLKX2})$

**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 conjunction 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 ( $32 \times \text{BUSCLKX4}$  cycles or  $4096 \times \text{BUSCLKX4}$  cycles)
- STOP instruction
- Computer operating properly module (COP)
- COP reset period (COPRS):  $8176 \times \text{BUSCLKX4}$  or  $262,128 \times \text{BUSCLKX4}$
- 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

### 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:	IRQPUD	IRQEN	R	R	R	R	OSCENINSTOP	RSTEN
Write:								
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)**

## Configuration Register (CONFIG)

### 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

### OSCEINSTOP— Oscillator Enable in Stop Mode Bit

OSCEINSTOP, 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:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVITRIP	SSREC	STOP	COPD
Write:								
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 =  $8176 \times \text{BUSCLKX4}$
- 0 = COP reset long cycle =  $262,128 \times \text{BUSCLKX4}$

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

- 1 = Auto wakeup short cycle =  $512 \times (\text{INTRCOSC or BUSCLKX4})$
- 0 = Auto wakeup long cycle =  $16,384 \times (\text{INTRCOSC or BUSCLKX4})$

### 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 2-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



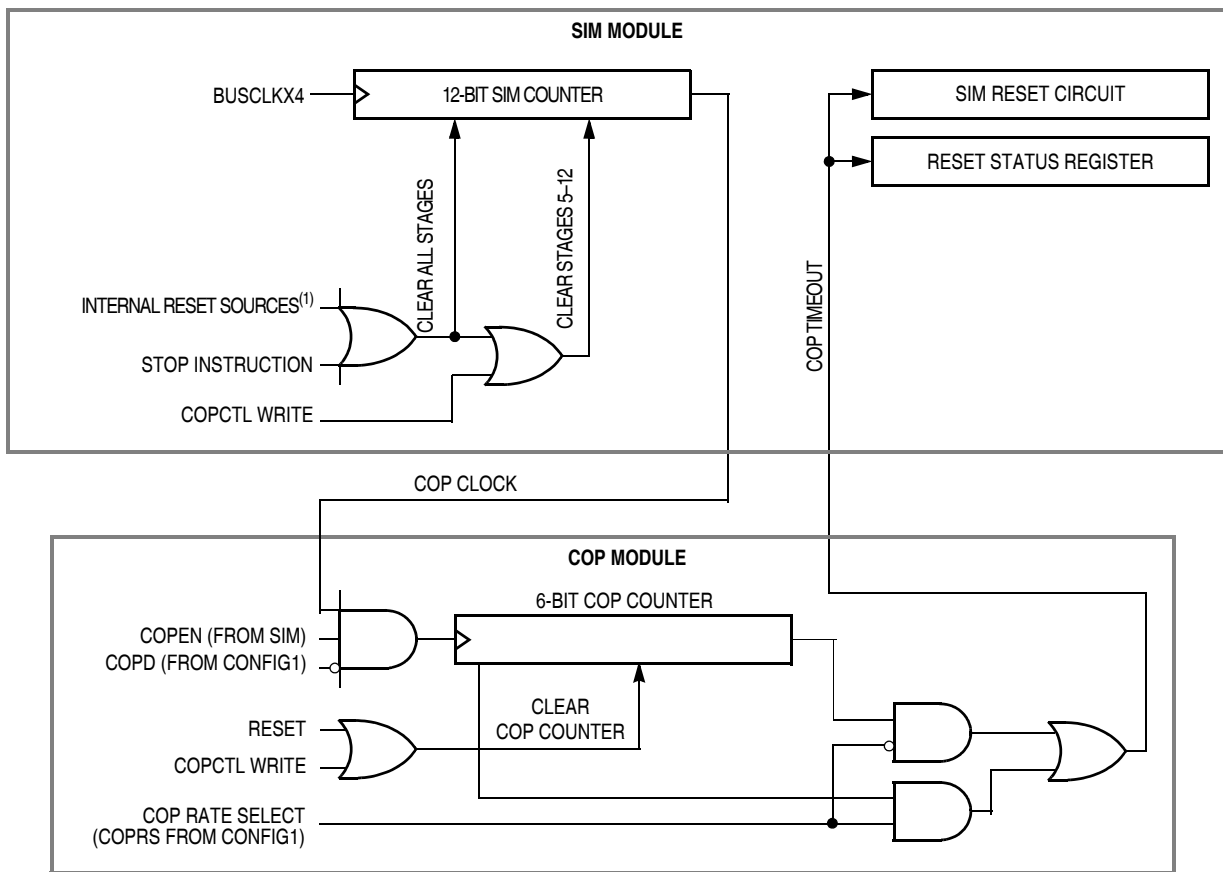
# Chapter 6

## 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 13 System Integration Module \(SIM\)](#) for more details.

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

## Computer Operating Properly (COP)

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 262,128 or 8176 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 CONFIG2 register) for  $32 \times \text{BUSCLKX4}$  cycles and sets the COP bit in the reset status register (RSR). See [13.8.1 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 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  $4096 \times \text{BUSCLKX4}$  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

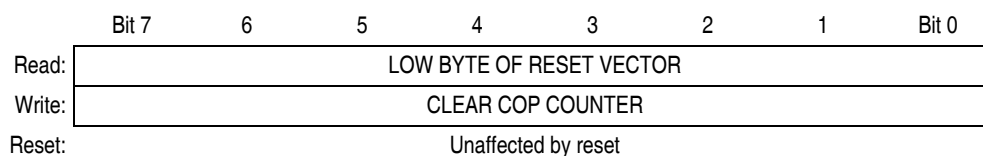
Stop mode turns off the BUSCLKX4 input to the COP and clears the SIM counter. Service the COP immediately before entering or after exiting stop mode to ensure a full COP timeout period after entering or exiting stop mode.

## 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.



**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

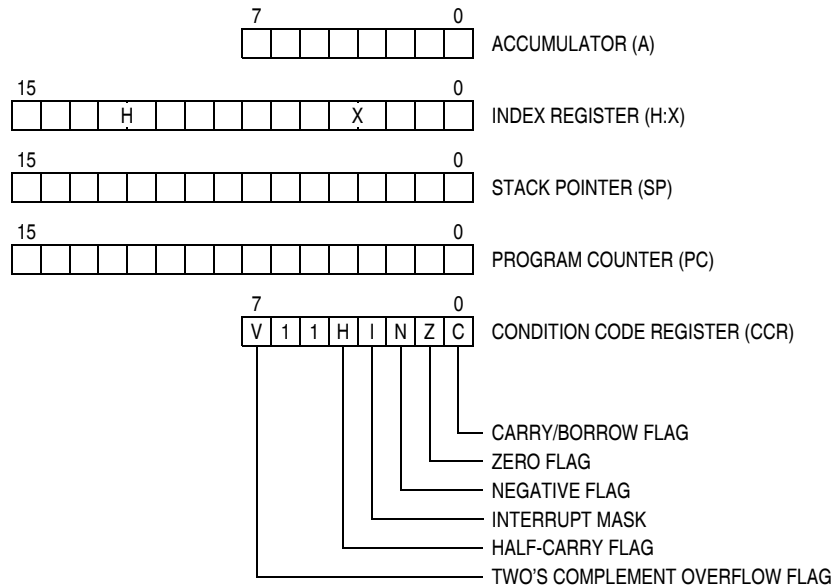
Features of the CPU include:

- 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.

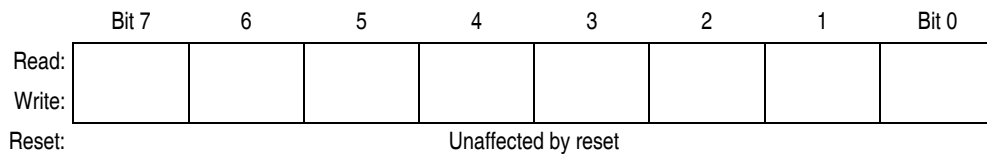
## Central Processor Unit (CPU)



**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.



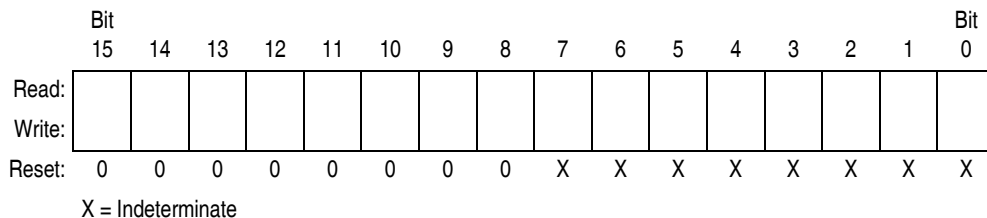
**Figure 7-2. Accumulator (A)**

### 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.

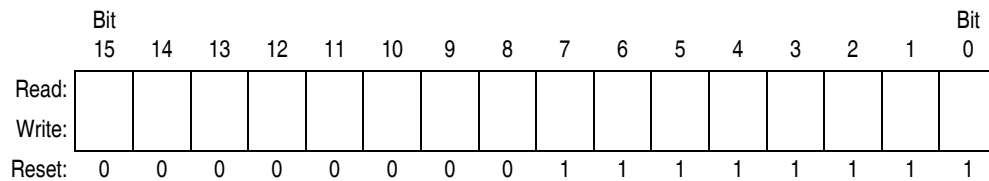


**Figure 7-3. Index Register (H:X)**

### 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

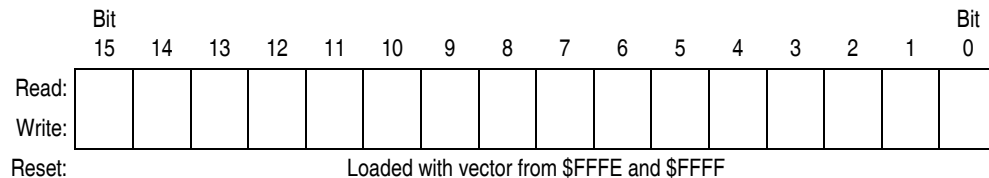
*The location of the stack is arbitrary and may be relocated anywhere in random-access memory (RAM). Moving the SP out of page 0 (\$0000 to \$00FF) frees direct address (page 0) space. For correct operation, the stack pointer must point only to RAM locations.*

### 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:	V	1	1	H	I	N	Z	C
Write:								
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)

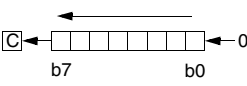
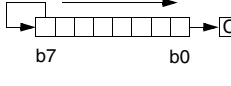
Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles	
			V	H	I	N	Z					C
ADC #opr ADC opr ADC opr ADC opr,X ADC opr,X ADC ,X ADC opr,SP ADC opr,SP	Add with Carry	$A \leftarrow (A) + (M) + (C)$	†	†	-	†	†	†	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9	ii dd ll hh ll ee ff ff ff ff ee	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	$A \leftarrow (A) + (M)$	†	†	-	†	†	†	IMM DIR EXT IX2 IX1 IX SP1 SP2	AB BB CB DB EB FB 9EEB 9EDB	ii dd hh ll ee ff ff ff ff ee	2 3 4 4 3 2 4 5
AIS #opr	Add Immediate Value (Signed) to SP	$SP \leftarrow (SP) + (16 \ll M)$	-	-	-	-	-	-	IMM	A7	ii	2
AIX #opr	Add Immediate Value (Signed) to H:X	$H:X \leftarrow (H:X) + (16 \ll M)$	-	-	-	-	-	-	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	$A \leftarrow (A) \& (M)$	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 ff ee	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	$PC \leftarrow (PC) + 2 + rel ? (C) = 0$	-	-	-	-	-	-	REL	24	rr	3
BCLR n, opr	Clear Bit n in M	$M_n \leftarrow 0$	-	-	-	-	-	-	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)	$PC \leftarrow (PC) + 2 + rel ? (C) = 1$	-	-	-	-	-	-	REL	25	rr	3
BEQ rel	Branch if Equal	$PC \leftarrow (PC) + 2 + rel ? (Z) = 1$	-	-	-	-	-	-	REL	27	rr	3
BGE opr	Branch if Greater Than or Equal To (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (N \oplus V) = 0$	-	-	-	-	-	-	REL	90	rr	3
BGT opr	Branch if Greater Than (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (Z) \mid (N \oplus V) = 0$	-	-	-	-	-	-	REL	92	rr	3
BHCC rel	Branch if Half Carry Bit Clear	$PC \leftarrow (PC) + 2 + rel ? (H) = 0$	-	-	-	-	-	-	REL	28	rr	3
BHCS rel	Branch if Half Carry Bit Set	$PC \leftarrow (PC) + 2 + rel ? (H) = 1$	-	-	-	-	-	-	REL	29	rr	3
BHI rel	Branch if Higher	$PC \leftarrow (PC) + 2 + rel ? (C) \mid (Z) = 0$	-	-	-	-	-	-	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 <i>rel</i>	Branch if Higher or Same (Same as BCC)	$PC \leftarrow (PC) + 2 + rel ? (C) = 0$	-	-	-	-	-	REL	24	rr	3	
BIH <i>rel</i>	Branch if IRQ Pin High	$PC \leftarrow (PC) + 2 + rel ? \overline{IRQ} = 1$	-	-	-	-	-	REL	2F	rr	3	
BIL <i>rel</i>	Branch if IRQ Pin Low	$PC \leftarrow (PC) + 2 + rel ? \overline{IRQ} = 0$	-	-	-	-	-	REL	2E	rr	3	
BIT # <i>opr</i> BIT <i>opr</i> BIT <i>opr</i> BIT <i>opr,X</i> BIT <i>opr,X</i> BIT <i>X</i> BIT <i>opr,SP</i> BIT <i>opr,SP</i>	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 ff ee ff	2 3 4 4 3 2 4 5
BLE <i>opr</i>	Branch if Less Than or Equal To (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (Z) \vee (N \oplus V) = 1$	-	-	-	-	-	REL	93	rr	3	
BLO <i>rel</i>	Branch if Lower (Same as BCS)	$PC \leftarrow (PC) + 2 + rel ? (C) = 1$	-	-	-	-	-	REL	25	rr	3	
BLS <i>rel</i>	Branch if Lower or Same	$PC \leftarrow (PC) + 2 + rel ? (C) \vee (Z) = 1$	-	-	-	-	-	REL	23	rr	3	
BLT <i>opr</i>	Branch if Less Than (Signed Operands)	$PC \leftarrow (PC) + 2 + rel ? (N \oplus V) = 1$	-	-	-	-	-	REL	91	rr	3	
BMC <i>rel</i>	Branch if Interrupt Mask Clear	$PC \leftarrow (PC) + 2 + rel ? (I) = 0$	-	-	-	-	-	REL	2C	rr	3	
BMI <i>rel</i>	Branch if Minus	$PC \leftarrow (PC) + 2 + rel ? (N) = 1$	-	-	-	-	-	REL	2B	rr	3	
BMS <i>rel</i>	Branch if Interrupt Mask Set	$PC \leftarrow (PC) + 2 + rel ? (I) = 1$	-	-	-	-	-	REL	2D	rr	3	
BNE <i>rel</i>	Branch if Not Equal	$PC \leftarrow (PC) + 2 + rel ? (Z) = 0$	-	-	-	-	-	REL	26	rr	3	
BPL <i>rel</i>	Branch if Plus	$PC \leftarrow (PC) + 2 + rel ? (N) = 0$	-	-	-	-	-	REL	2A	rr	3	
BRA <i>rel</i>	Branch Always	$PC \leftarrow (PC) + 2 + rel$	-	-	-	-	-	REL	20	rr	3	
BRCLR <i>n,opr,rel</i>	Branch if Bit <i>n</i> in M Clear	$PC \leftarrow (PC) + 3 + rel ? (Mn) = 0$	-	-	-	-	†	-	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 <i>rel</i>	Branch Never	$PC \leftarrow (PC) + 2$	-	-	-	-	-	REL	21	rr	3	
BRSET <i>n,opr,rel</i>	Branch if Bit <i>n</i> in M Set	$PC \leftarrow (PC) + 3 + rel ? (Mn) = 1$	-	-	-	-	†	-	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 <i>n,opr</i>	Set Bit <i>n</i> in M	$Mn \leftarrow 1$	-	-	-	-	-	-	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 <i>rel</i>	Branch to Subroutine	$PC \leftarrow (PC) + 2$ ; push (PCL) $SP \leftarrow (SP) - 1$ ; push (PCH) $SP \leftarrow (SP) - 1$ $PC \leftarrow (PC) + rel$	-	-	-	-	-	REL	AD	rr	4	
CBEQ <i>opr,rel</i> CBEQA # <i>opr,rel</i> CBEQX # <i>opr,rel</i> CBEQ <i>opr,X+,rel</i> CBEQ <i>X+,rel</i> CBEQ <i>opr,SP,rel</i>	Compare and Branch if Equal	$PC \leftarrow (PC) + 3 + rel ? (A) - (M) = \$00$ $PC \leftarrow (PC) + 3 + rel ? (A) - (M) = \$00$ $PC \leftarrow (PC) + 3 + rel ? (X) - (M) = \$00$ $PC \leftarrow (PC) + 3 + rel ? (A) - (M) = \$00$ $PC \leftarrow (PC) + 2 + rel ? (A) - (M) = \$00$ $PC \leftarrow (PC) + 4 + rel ? (A) - (M) = \$00$	-	-	-	-	-	-	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	$C \leftarrow 0$	-	-	-	-	0	INH	98		1	
CLI	Clear Interrupt Mask	$I \leftarrow 0$	-	-	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 <i>opr</i> CLRA CLR <sub>X</sub> CLR <sub>H</sub> CLR <i>opr,X</i> CLR ,X CLR <i>opr,SP</i>	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 # <i>opr</i> CMP <i>opr</i> CMP <i>opr</i> CMP <i>opr,X</i> CMP <i>opr,X</i> CMP ,X CMP <i>opr,SP</i> CMP <i>opr,SP</i>	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 ff ee ff	2 3 4 4 3 2 4 5
COM <i>opr</i> COMA COM <sub>X</sub> COM <i>opr,X</i> COM ,X COM <i>opr,SP</i>	Complement (One's Complement)	M ← (M) = \$FF - (M) A ← (A) = \$FF - (M) X ← (X) = \$FF - (M) M ← (M) = \$FF - (M) M ← (M) = \$FF - (M) M ← (M) = \$FF - (M)	0	-	-	†	†	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd   ff ff ff	4 1 1 4 3 5
CPHX # <i>opr</i> CPHX <i>opr</i>	Compare H:X with M	(H:X) - (M:M + 1)	†	-	-	†	†	†	IMM DIR	65 75	ii ii+1 dd	3 4
CPX # <i>opr</i> CPX <i>opr</i> CPX <i>opr</i> CPX ,X CPX <i>opr,X</i> CPX <i>opr,X</i> CPX <i>opr,SP</i> CPX <i>opr,SP</i>	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 ff ee ff	2 3 4 4 3 2 4 5
DAA	Decimal Adjust A	(A) <sub>10</sub>	U	-	-	†	†	†	INH	72		2
DBNZ <i>opr,rel</i> DBNZ <sub>A</sub> <i>rel</i> DBNZ <sub>X</sub> <i>rel</i> DBNZ <i>opr,X,rel</i> DBNZ ,X, <i>rel</i> DBNZ <i>opr,SP,rel</i>	Decrement and Branch if Not Zero	A ← (A) - 1 or M ← (M) - 1 or X ← (X) - 1 PC ← (PC) + 3 + <i>rel</i> ? (result) ≠ 0 PC ← (PC) + 2 + <i>rel</i> ? (result) ≠ 0 PC ← (PC) + 2 + <i>rel</i> ? (result) ≠ 0 PC ← (PC) + 3 + <i>rel</i> ? (result) ≠ 0 PC ← (PC) + 2 + <i>rel</i> ? (result) ≠ 0 PC ← (PC) + 4 + <i>rel</i> ? (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 <i>opr</i> DECA DEC <sub>X</sub> DEC <i>opr,X</i> DEC ,X DEC <i>opr,SP</i>	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 ff	4 1 1 4 3 5
DIV	Divide	A ← (H:A)/(X) H ← Remainder	-	-	-	-	†	†	INH	52		7
EOR # <i>opr</i> EOR <i>opr</i> EOR <i>opr</i> EOR <i>opr,X</i> EOR <i>opr,X</i> EOR ,X EOR <i>opr,SP</i> EOR <i>opr,SP</i>	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 ff ee ff	2 3 4 4 3 2 4 5
INC <i>opr</i> INCA INC <sub>X</sub> INC <i>opr,X</i> INC ,X INC <i>opr,SP</i>	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 ff	4 1 1 4 3 5

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

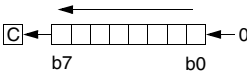
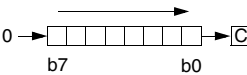
Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z				
JMP <i>opr</i> JMP <i>opr</i> JMP <i>opr,X</i> JMP <i>opr,X</i> JMP ,X	Jump	PC ← Jump Address	-	-	-	-	-	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 2
JSR <i>opr</i> JSR <i>opr</i> JSR <i>opr,X</i> JSR <i>opr,X</i> JSR ,X	Jump to Subroutine	PC ← (PC) + <i>n</i> ( <i>n</i> = 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 # <i>opr</i> LDA <i>opr</i> LDA <i>opr</i> LDA <i>opr,X</i> LDA <i>opr,X</i> LDA ,X LDA <i>opr,SP</i> LDA <i>opr,SP</i>	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 # <i>opr</i> LDHX <i>opr</i>	Load H:X from M	H:X ← (M:M + 1)	0	-	-	↑	↑	IMM DIR	45 55	ii jj dd	3 4
LDX # <i>opr</i> LDX <i>opr</i> LDX <i>opr</i> LDX <i>opr,X</i> LDX <i>opr,X</i> LDX ,X LDX <i>opr,SP</i> LDX <i>opr,SP</i>	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 ff ee ff	2 3 4 4 3 2 4 5
LSL <i>opr</i> LSLA LSLX LSL <i>opr,X</i> LSL ,X LSL <i>opr,SP</i>	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 <i>opr</i> LSRA LSRX LSR <i>opr,X</i> LSR ,X LSR <i>opr,SP</i>	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 <i>opr,opr</i> MOV <i>opr,X+</i> MOV # <i>opr,opr</i> MOV X+, <i>opr</i>	Move	(M) <sub>Destination</sub> ← (M) <sub>Source</sub> 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 <i>opr</i> NEGA NEGX NEG <i>opr,X</i> NEG ,X NEG <i>opr,SP</i>	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 # <i>opr</i> ORA <i>opr</i> ORA <i>opr</i> ORA <i>opr,X</i> ORA <i>opr,X</i> ORA ,X ORA <i>opr,SP</i> ORA <i>opr,SP</i>	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 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)

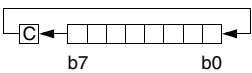
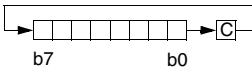
Source Form	Operation	Description	Effect on CCR					Address Mode	Opcode	Operand	Cycles	
			V	H	I	N	Z					C
PULA	Pull A from Stack	$SP \leftarrow (SP + 1); \text{Pull (A)}$	-	-	-	-	-	-	INH	86		2
PULH	Pull H from Stack	$SP \leftarrow (SP + 1); \text{Pull (H)}$	-	-	-	-	-	-	INH	8A		2
PULX	Pull X from Stack	$SP \leftarrow (SP + 1); \text{Pull (X)}$	-	-	-	-	-	-	INH	88		2
ROL <i>opr</i> ROLA ROLX ROL <i>opr,X</i> ROL ,X ROL <i>opr,SP</i>	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 <i>opr</i> RORA RORX ROR <i>opr,X</i> ROR ,X ROR <i>opr,SP</i>	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	$SP \leftarrow \$FF$	-	-	-	-	-	-	INH	9C		1
RTI	Return from Interrupt	$SP \leftarrow (SP + 1); \text{Pull (CCR)}$ $SP \leftarrow (SP + 1); \text{Pull (A)}$ $SP \leftarrow (SP + 1); \text{Pull (X)}$ $SP \leftarrow (SP + 1); \text{Pull (PCH)}$ $SP \leftarrow (SP + 1); \text{Pull (PCL)}$	↑	↑	↑	↑	↑	↑	INH	80		7
RTS	Return from Subroutine	$SP \leftarrow SP + 1; \text{Pull (PCH)}$ $SP \leftarrow SP + 1; \text{Pull (PCL)}$	-	-	-	-	-	-	INH	81		4
SBC # <i>opr</i> SBC <i>opr</i> SBC <i>opr</i> SBC <i>opr,X</i> SBC <i>opr,X</i> SBC ,X SBC <i>opr,SP</i> SBC <i>opr,SP</i>	Subtract with Carry	$A \leftarrow (A) - (M) - (C)$	↑	-	-	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP1 SP2	A2 B2 C2 D2 E2 F2 9EE2 9ED2	ii dd hh ll ee ff ff ff ff ff ee ff	2 3 4 4 3 2 4 5
SEC	Set Carry Bit	$C \leftarrow 1$	-	-	-	-	-	1	INH	99		1
SEI	Set Interrupt Mask	$I \leftarrow 1$	-	-	1	-	-	-	INH	9B		2
STA <i>opr</i> STA <i>opr</i> STA <i>opr,X</i> STA <i>opr,X</i> STA ,X STA <i>opr,SP</i> STA <i>opr,SP</i>	Store A in M	$M \leftarrow (A)$	0	-	-	↑	↑	-	DIR EXT IX2 IX1 IX SP1 SP2	B7 C7 D7 E7 F7 9EE7 9ED7	dd hh ll hh ll ee ff ff ff ff ee ff	3 4 4 3 2 4 5
STHX <i>opr</i>	Store H:X in M	$(M:M + 1) \leftarrow (H:X)$	0	-	-	↑	↑	-	DIR	35	dd	4
STOP	Enable Interrupts, Stop Processing, Refer to MCU Documentation	$I \leftarrow 0; \text{Stop Processing}$	-	-	0	-	-	-	INH	8E		1
STX <i>opr</i> STX <i>opr</i> STX <i>opr,X</i> STX <i>opr,X</i> STX ,X STX <i>opr,SP</i> STX <i>opr,SP</i>	Store X in M	$M \leftarrow (X)$	0	-	-	↑	↑	-	DIR EXT IX2 IX1 IX SP1 SP2	BF CF DF EF FF 9EEF 9EDF	dd hh ll hh ll ee ff ff ff ff ee ff	3 4 4 3 2 4 5
SUB # <i>opr</i> SUB <i>opr</i> SUB <i>opr</i> SUB <i>opr,X</i> SUB <i>opr,X</i> SUB ,X SUB <i>opr,SP</i> SUB <i>opr,SP</i>	Subtract	$A \leftarrow (A) - (M)$	↑	-	-	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP1 SP2	A0 B0 C0 D0 E0 F0 9EE0 9ED0	ii dd hh ll hh ll ee ff 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 <i>opr</i> TSTA TSTX TST <i>opr,X</i> TST ,X TST <i>opr,SP</i>	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   | <i>n</i>   | Any bit                                     |
| C     | Carry/borrow bit  | <i>opr</i> | 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  | <i>rel</i> | 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

MSB LSB	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
	0	1	2	3	4	5	6	9E6	7	8	9	A	B	C	D	9ED	E	9EE	F
0	BRSET0 3 DIR	BSET0 2 DIR	BRA 2 REL	NEG 2 DIR	NEGA 1 INH	NEGX 1 INH	NEG 2 IX1	NEG 3 SP1	NEG 1 IX	RTI 1 INH	BGE 2 REL	SUB 2 IMM	SUB 2 DIR	SUB 3 EXT	SUB 3 IX2	SUB 4 SP2	SUB 2 IX1	SUB 3 SP1	SUB 1 IX
1	BRCLR0 3 DIR	BCLR0 2 DIR	BRN 2 REL	CBEQ 3 DIR	CBEQA 3 IMM	CBEQX 3 IMM	CBEQ 3 IX1+	CBEQ 4 SP1	CBEQ 2 IX+	RTS 1 INH	BLT 2 REL	CMP 2 IMM	CMP 2 DIR	CMP 3 EXT	CMP 3 IX2	CMP 4 SP2	CMP 2 IX1	CMP 3 SP1	CMP 1 IX
2	BRSET1 3 DIR	BSET1 2 DIR	BHI 2 REL		MUL 1 INH	DIV 1 INH	NSA 1 INH		DAA 1 INH		BGT 2 REL	SBC 2 IMM	SBC 2 DIR	SBC 3 EXT	SBC 3 IX2	SBC 4 SP2	SBC 2 IX1	SBC 3 SP1	SBC 1 IX
3	BRCLR1 3 DIR	BCLR1 2 DIR	BLS 2 REL	COM 2 DIR	COMA 1 INH	COMX 1 INH	COM 2 IX1	COM 3 SP1	COM 1 IX	SWI 1 INH	BLE 2 REL	CPX 2 IMM	CPX 2 DIR	CPX 3 EXT	CPX 3 IX2	CPX 4 SP2	CPX 2 IX1	CPX 3 SP1	CPX 1 IX
4	BRSET2 3 DIR	BSET2 2 DIR	BCC 2 REL	LSR 2 DIR	LSRA 1 INH	LSRX 1 INH	LSR 2 IX1	LSR 3 SP1	LSR 1 IX	TAP 1 INH	TXS 1 INH	AND 2 IMM	AND 2 DIR	AND 3 EXT	AND 3 IX2	AND 4 SP2	AND 2 IX1	AND 3 SP1	AND 1 IX
5	BRCLR2 3 DIR	BCLR2 2 DIR	BCS 2 REL	STHX 2 DIR	LDHX 3 IMM	LDHX 2 DIR	CPHX 3 IMM		CPHX 2 DIR	TPA 1 INH	TSX 1 INH	BIT 2 IMM	BIT 2 DIR	BIT 3 EXT	BIT 3 IX2	BIT 4 SP2	BIT 2 IX1	BIT 3 SP1	BIT 1 IX
6	BRSET3 3 DIR	BSET3 2 DIR	BNE 2 REL	ROR 2 DIR	RORA 1 INH	RORX 1 INH	ROR 2 IX1	ROR 3 SP1	ROR 1 IX	PULA 1 INH		LDA 2 IMM	LDA 2 DIR	LDA 3 EXT	LDA 3 IX2	LDA 4 SP2	LDA 2 IX1	LDA 3 SP1	LDA 1 IX
7	BRCLR3 3 DIR	BCLR3 2 DIR	BEQ 2 REL	ASR 2 DIR	ASRA 1 INH	ASRX 1 INH	ASR 2 IX1	ASR 3 SP1	ASR 1 IX	PSHA 1 INH	TAX 1 INH	AIS 2 IMM	STA 2 DIR	STA 3 EXT	STA 3 IX2	STA 4 SP2	STA 2 IX1	STA 3 SP1	STA 1 IX
8	BRSET4 3 DIR	BSET4 2 DIR	BHCC 2 REL	LSL 2 DIR	LSLA 1 INH	LSLX 1 INH	LSL 2 IX1	LSL 3 SP1	LSL 1 IX	PULX 1 INH	CLC 1 INH	EOR 2 IMM	EOR 2 DIR	EOR 3 EXT	EOR 3 IX2	EOR 4 SP2	EOR 2 IX1	EOR 3 SP1	EOR 1 IX
9	BRCLR4 3 DIR	BCLR4 2 DIR	BHCS 2 REL	ROL 2 DIR	ROLA 1 INH	ROLX 1 INH	ROL 2 IX1	ROL 3 SP1	ROL 1 IX	PSHX 1 INH	SEC 1 INH	ADC 2 IMM	ADC 2 DIR	ADC 3 EXT	ADC 3 IX2	ADC 4 SP2	ADC 2 IX1	ADC 3 SP1	ADC 1 IX
A	BRSET5 3 DIR	BSET5 2 DIR	BPL 2 REL	DEC 2 DIR	DECA 1 INH	DECX 1 INH	DEC 2 IX1	DEC 3 SP1	DEC 1 IX	PULH 1 INH	CLI 1 INH	ORA 2 IMM	ORA 2 DIR	ORA 3 EXT	ORA 3 IX2	ORA 4 SP2	ORA 2 IX1	ORA 3 SP1	ORA 1 IX
B	BRCLR5 3 DIR	BCLR5 2 DIR	BMI 2 REL	DBNZ 3 DIR	DBNZA 2 INH	DBNZX 2 INH	DBNZ 3 IX1	DBNZ 4 SP1	DBNZ 2 IX	PSHH 1 INH	SEI 1 INH	ADD 2 IMM	ADD 2 DIR	ADD 3 EXT	ADD 3 IX2	ADD 4 SP2	ADD 2 IX1	ADD 3 SP1	ADD 1 IX
C	BRSET6 3 DIR	BSET6 2 DIR	BMC 2 REL	INC 2 DIR	INCA 1 INH	INCX 1 INH	INC 2 IX1	INC 3 SP1	INC 1 IX	CLRH 1 INH	RSP 1 INH		JMP 2 DIR	JMP 3 EXT	JMP 3 IX2		JMP 2 IX1		JMP 1 IX
D	BRCLR6 3 DIR	BCLR6 2 DIR	BMS 2 REL	TST 2 DIR	TSTA 1 INH	TSTX 1 INH	TST 2 IX1	TST 3 SP1	TST 1 IX		NOP 1 INH	BSR 2 REL	JSR 2 DIR	JSR 3 EXT	JSR 3 IX2		JSR 2 IX1		JSR 1 IX
E	BRSET7 3 DIR	BSET7 2 DIR	BIL 2 REL		MOV 3 DD	MOV 2 DIX+	MOV 3 IMD		MOV 2 IX+D	STOP 1 INH	*	LDX 2 IMM	LDX 2 DIR	LDX 3 EXT	LDX 3 IX2	LDX 4 SP2	LDX 2 IX1	LDX 3 SP1	LDX 1 IX
F	BRCLR7 3 DIR	BCLR7 2 DIR	BIH 2 REL	CLR 2 DIR	CLRA 1 INH	CLRX 1 INH	CLR 2 IX1	CLR 3 SP1	CLR 1 IX	WAIT 1 INH	TXA 1 INH	AIX 2 IMM	STX 2 DIR	STX 3 EXT	STX 3 IX2	STX 4 SP2	STX 2 IX1	STX 3 SP1	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  
 IMM Direct-Immediate  
 DIX+ Direct-Indexed  
 SP1 Stack Pointer, 8-Bit Offset  
 SP2 Stack Pointer, 16-Bit Offset  
 IX+ Indexed, No Offset with Post Increment  
 IX1+ Indexed, 1-Byte Offset with Post Increment

\*Pre-byte for stack pointer indexed instructions

Low Byte of Opcode in Hexadecimal

MSB	0
LSB	5 BRSET0 3 DIR

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

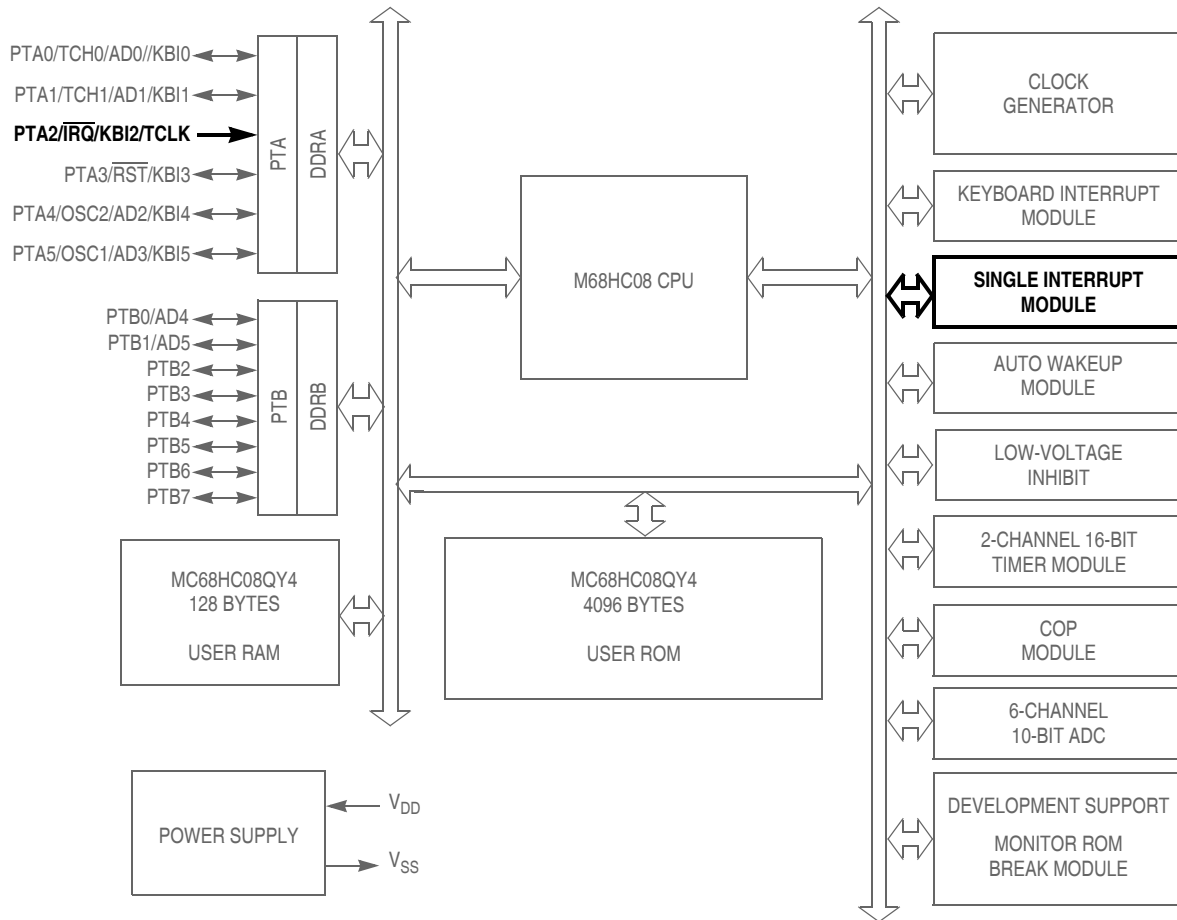
A low level applied to the external interrupt request ( $\overline{\text{IRQ}}$ ) pin can latch a CPU interrupt request. [Figure 8-2](#) shows the structure of the IRQ module.

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.

## External Interrupt (IRQ)



$\overline{RST}$ ,  $\overline{IRQ}$ : Pins have internal pull up device  
 All port pins have programmable pull up device  
 PTA[0:5]: Higher current sink and source capability  
 PTB[0:7]: Not available on 8-pin devices

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

When set, the IMASK bit in INTSCR masks the  $\overline{IRQ}$  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  $\overline{IRQ}$  interrupt request.*

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



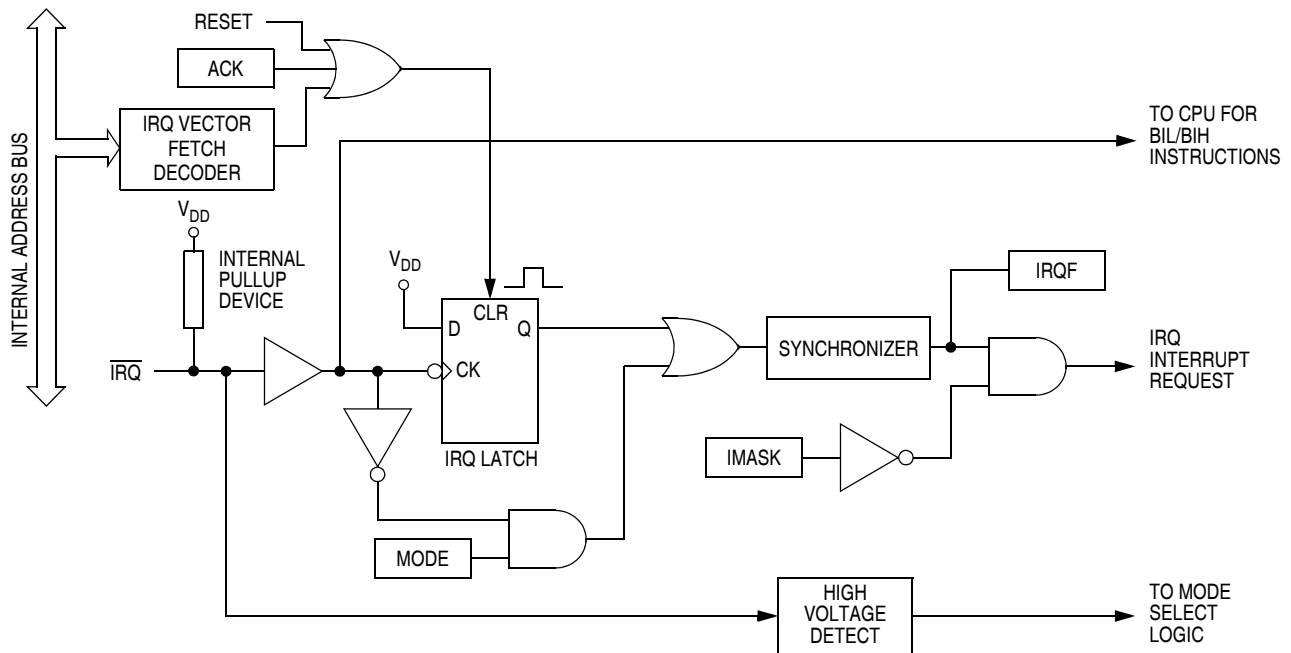


Figure 8-2. IRQ Module Block Diagram

### 8.3.1 MODE = 1

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

- Return of the  $\overline{\text{IRQ}}$  pin to a high level. As long as the  $\overline{\text{IRQ}}$  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  $\overline{\text{IRQ}}$  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  $\overline{\text{IRQ}}$  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  $\overline{\text{IRQ}}$  pin to a high level may occur in any order. The interrupt request remains pending as long as the  $\overline{\text{IRQ}}$  pin is low. A reset will clear the 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  $\overline{\text{IRQ}}$  pin.

### 8.3.2 MODE = 0

If the MODE bit is clear, the  $\overline{\text{IRQ}}$  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  $\overline{\text{IRQ}}$  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 ( $\overline{\text{IRQ}}$ )


The  $\overline{\text{IRQ}}$  pin provides a maskable external interrupt source. The  $\overline{\text{IRQ}}$  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 = IRQ interrupt pending
- 0 = 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 = IRQ interrupt request on falling edges and low levels
- 0 = 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

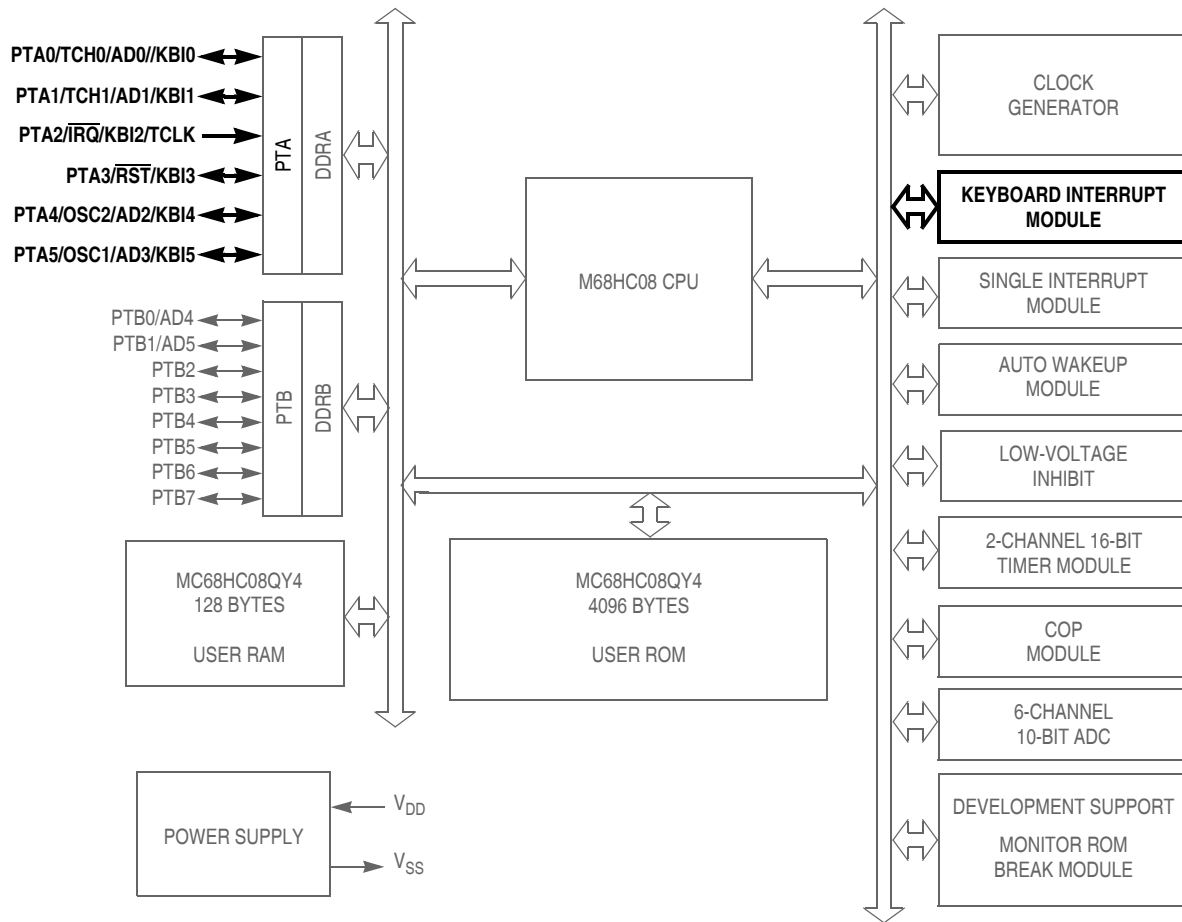
Writing to the KBIEx bits in the keyboard interrupt enable register (KBIER) independently enables or disables each KBI pin. The polarity of the keyboard interrupt is controlled using the KBIPx bits in the keyboard interrupt polarity register (KBIPR). Edge-only or edge and level sensitivity is controlled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

## Keyboard Interrupt Module (KBI)



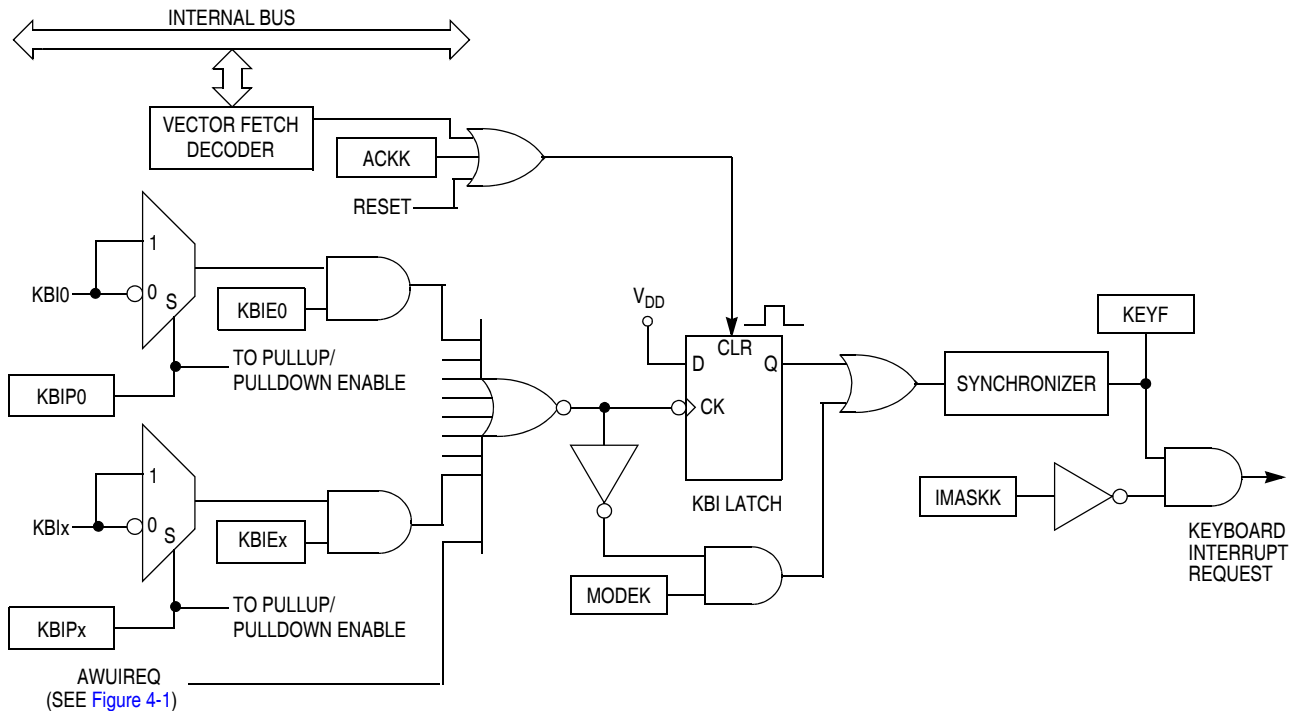
$\overline{\text{RST}}$ ,  $\overline{\text{IRQ}}$ : Pins have internal pull up device  
 All port pins have programmable pull up device  
 PTA[0:5]: Higher current sink and source capability  
 PTB[0:7]: Not available on 8-pin devices

**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.
- 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.



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

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 (KBIE<sub>x</sub>) 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:

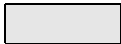
- KBSCR (keyboard interrupt status and control register)
- KBIER (keyboard interrupt enable register)
- KBIPR (keyboard interrupt polarity register)

### 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

## Keyboard Interrupt Module (KBI)

### 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.

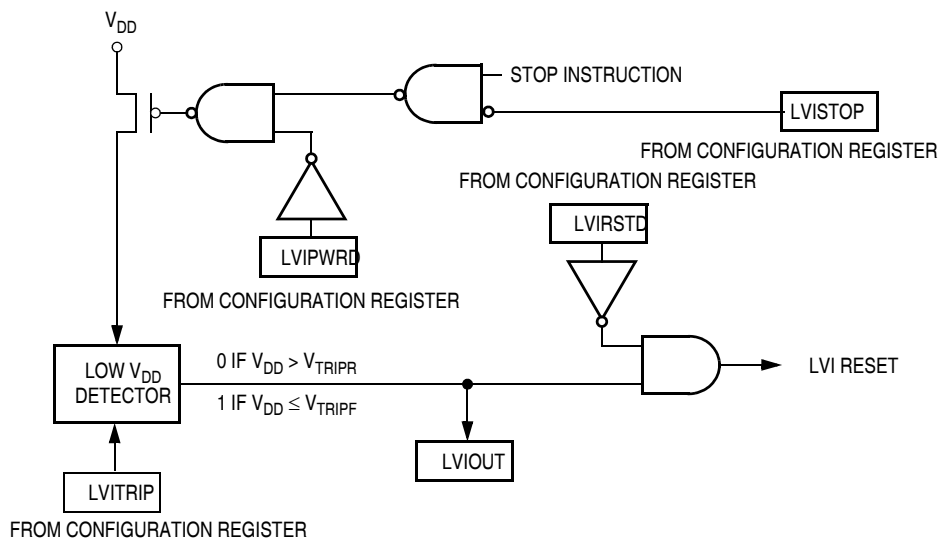
### 10.2 Features

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**

## Low-Voltage Inhibit (LVI)

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 [16.5 5-V DC Electrical Characteristics](#), [16.8 3-V DC Electrical Characteristics](#), and [16.11 1.8-V to 3.6-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 13 System Integration Module \(SIM\)](#) for the reset recovery sequence.

The output of the comparator controls the state of the LVIOOUT 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 LVIOOUT 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:

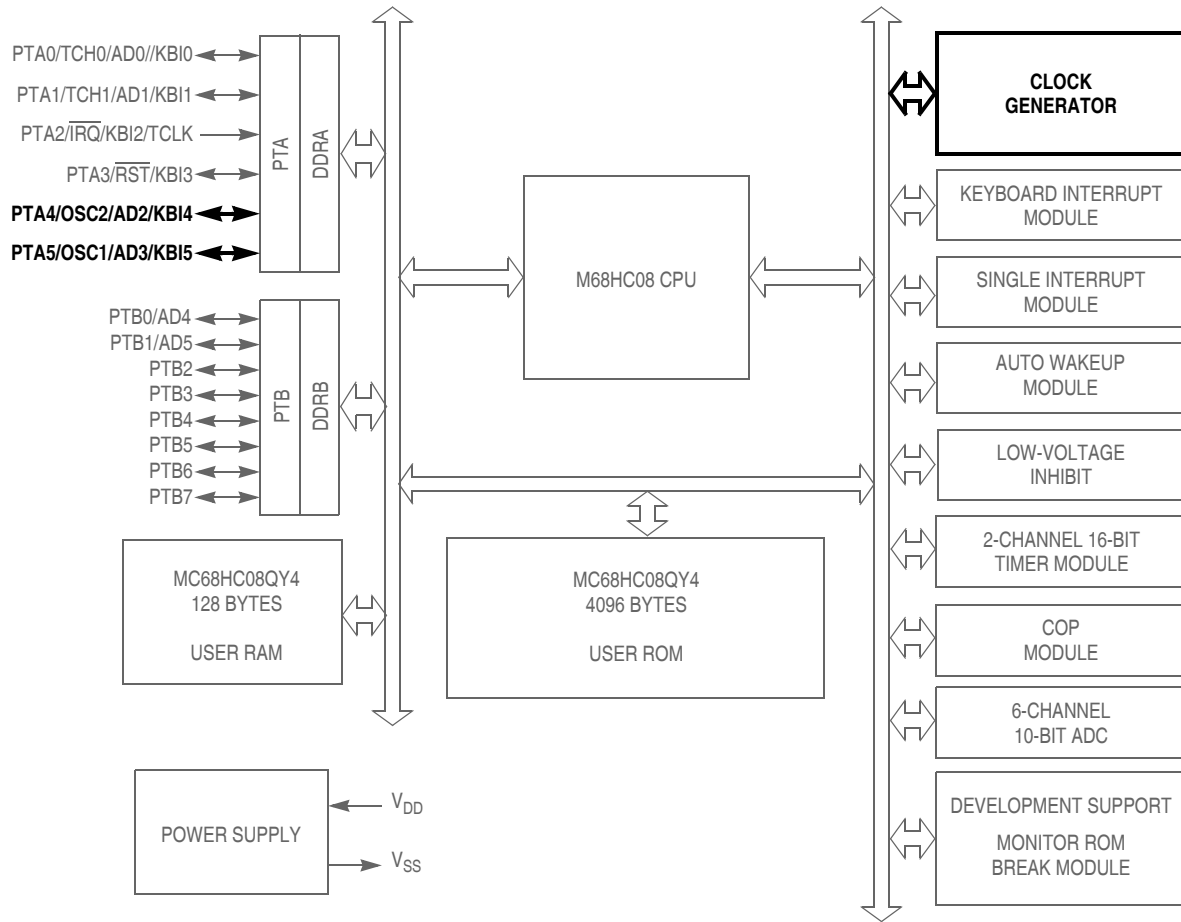
1. Internal oscillator: An internally generated, fixed frequency clock, trimmable to  $\pm 0.4\%$ . There are three choices for the internal oscillator, 12.8 MHz, 8 MHz, or 4 MHz. The 4-MHz internal oscillator is the default option out of reset.
2. External oscillator: An external clock that can be driven 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

## Oscillator Module (OSC)



$\overline{RST}$ ,  $\overline{IRQ}$ : Pins have internal pull up device  
 All port pins have programmable pull up device  
 PTA[0:5]: Higher current sink and source capability  
 PTB[0:7]: Not available on 8-pin devices

**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  $\pm 25\%$  untrimmed. An 8-bit register (OSCTRIM) allows the digital adjustment to a tolerance of  $\Delta_{TRIM\_ACC}$ . 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 as specified in the ROM order form. This trim value is stored in memory 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.

### 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.

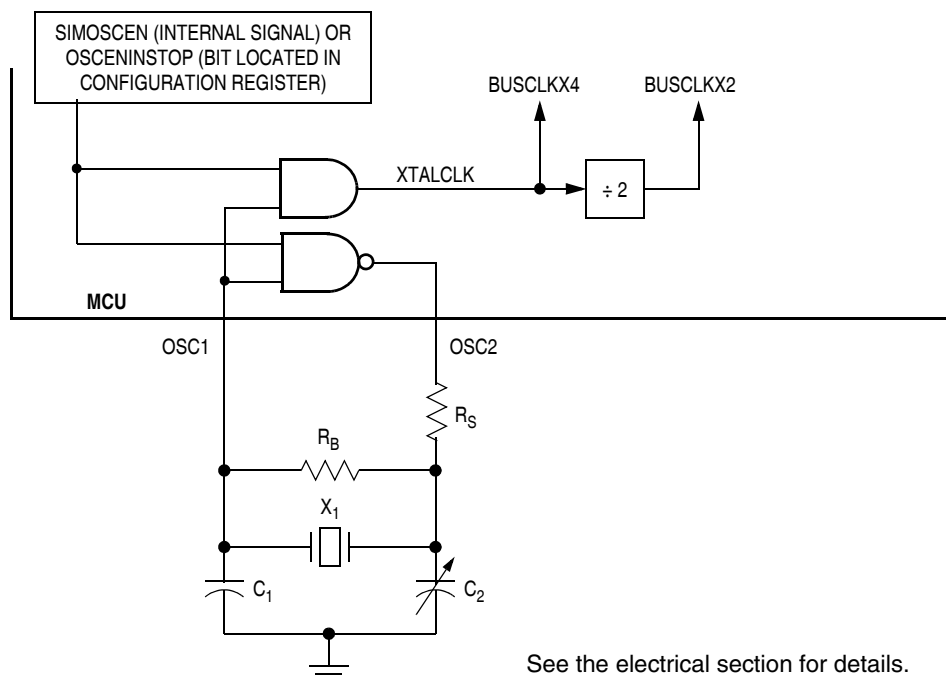
In its typical configuration, the XTAL oscillator is connected in a Pierce oscillator configuration, as shown in Figure 11-2. This figure shows only the logical representation of the internal components and may not represent actual circuitry.

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.*



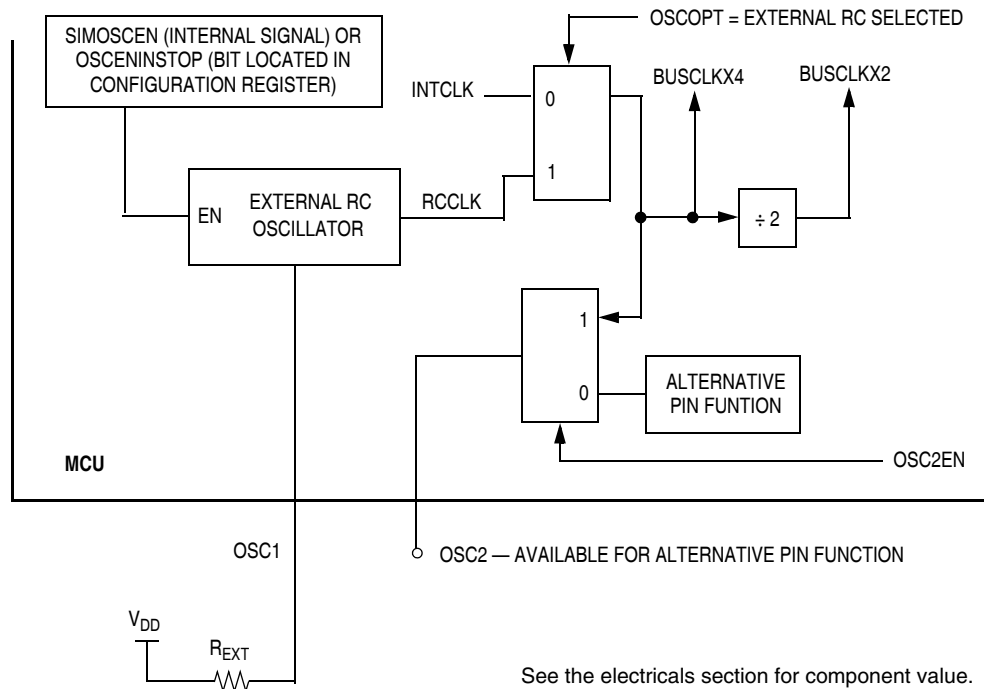
**Figure 11-2. XTAL Oscillator External Connections**

### 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

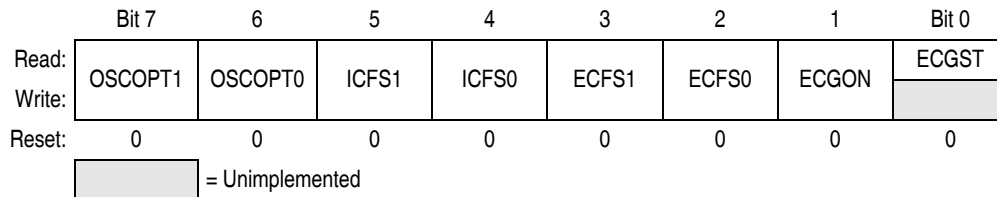
## 11.8 Registers

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.



**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:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
Write:								
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 MC68HC08QY1, MC68HC08QY2 and MC68HC08QY4 have thirteen bidirectional input-output (I/O) pins and one input only pin. The MC68HC08QT1, MC68HC08QT2 and MC68HC08QT4 has five bidirectional I/O pins and one input only pin. All I/O pins are programmable as inputs or outputs.

### 12.2 Unused Pin Termination

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

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

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

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

### 12.3 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 2-channel timer interface module (TIM) (see [Chapter 14 Timer Interface Module \(TIM\)](#)), the 10-bit ADC (see [Chapter 3 10-Bit Analog-to-Digital Converter \(ADC10\) Module](#)), the external interrupt (IRQ) pin (see [Chapter 8 External Interrupt \(IRQ\)](#)), the reset ( $\overline{RST}$ ) 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**

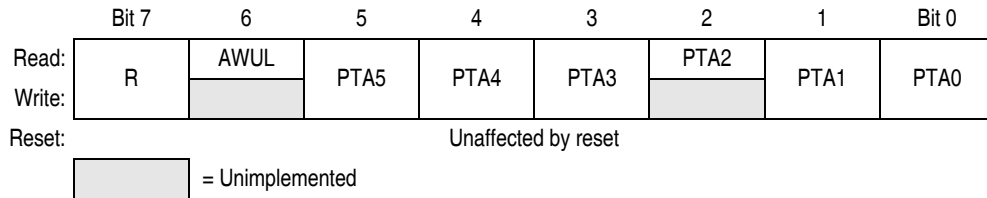
*PTA2 is input only.*

*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.*

## Input/Output Ports (PORTS)

### 12.3.1 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

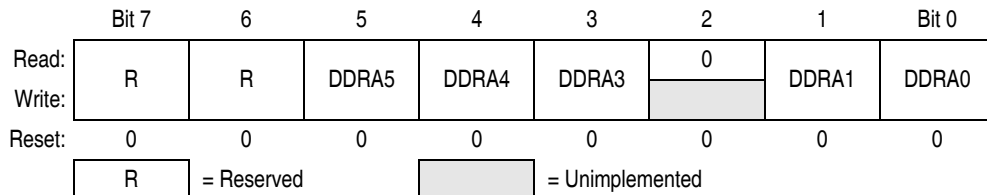
These read/write bits are software programmable. Data direction of each port A pin is under the control of the corresponding bit in data direction register A. Reset has no effect on port A data.

#### 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.

### 12.3.2 Data Direction Register A

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



**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.

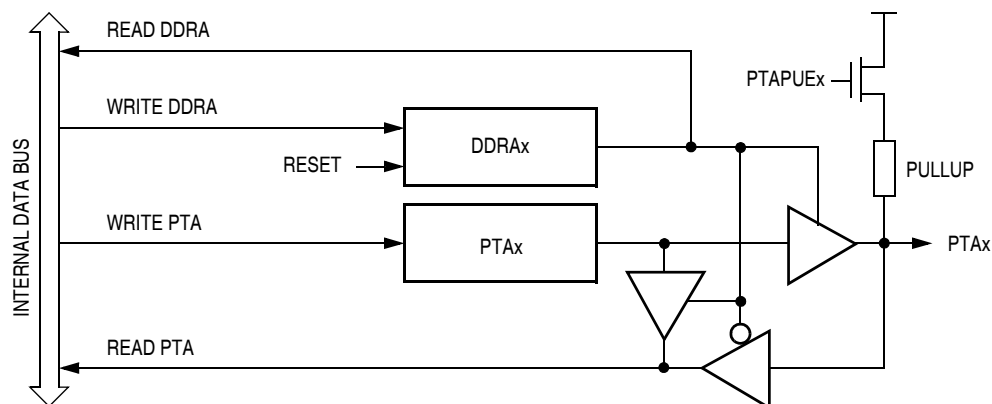
1 = Corresponding port A pin configured as output

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**

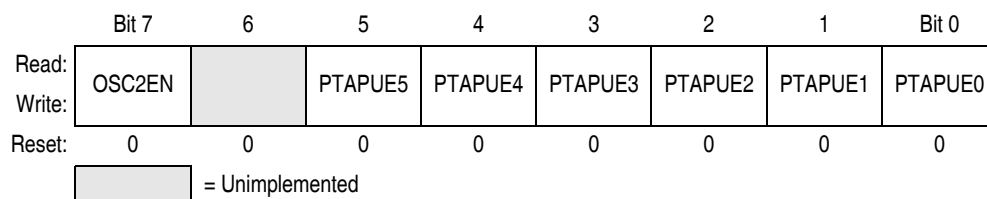
**NOTE**

*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.

### 12.3.3 Port A Input Pullup Enable Register

The port A input pullup enable register (PTAPUE) contains a software configurable pullup device for each of the 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.



**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

### 12.3.4 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 <sup>(1)</sup>	Input, V <sub>DD</sub> <sup>(2)</sup>	DDRA5–DDRA0	Pin	PTA5–PTA0 <sup>(3)</sup>
0	0	X	Input, Hi-Z <sup>(4)</sup>	DDRA5–DDRA0	Pin	PTA5–PTA0 <sup>(3)</sup>
X	1	X	Output	DDRA5–DDRA0	PTA5–PTA0	PTA5–PTA0 <sup>(5)</sup>

1. X = don't care
2. I/O pin pulled to V<sub>DD</sub> 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

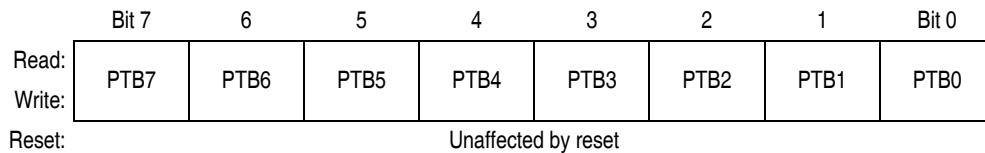
## 12.4 Port B

Port B is an 8-bit special function port that shares two of its pins with the 10-bit ADC (see [Chapter 3 10-Bit Analog-to-Digital Converter \(ADC10\) Module](#)).

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

### 12.4.1 Port B Data Register

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



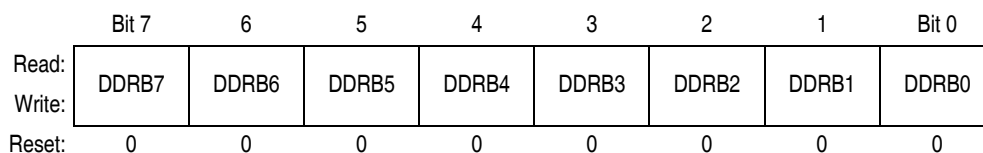
**Figure 12-5. Port B Data Register (PTB)**

#### 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.

## 12.4.2 Data Direction Register B

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



**Figure 12-6. Data Direction Register B (DDRB)**

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

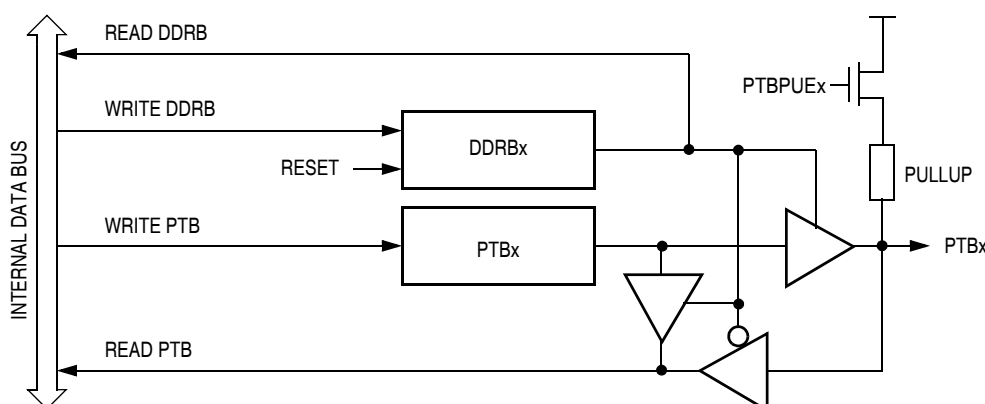
These read/write bits control port B data direction. Reset clears DDRB[7:0], configuring all port B pins as inputs.

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.*

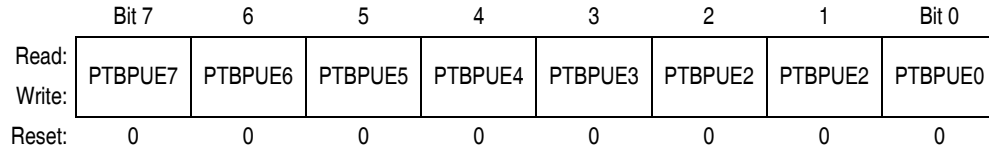


**Figure 12-7. Port B I/O Circuit**

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.

### 12.4.3 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.



**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.

### 12.4.4 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 <sup>(1)</sup>	Input, Hi-Z <sup>(2)</sup>	DDRB7–DDRB0		Pin	PTB7–PTB0 <sup>(3)</sup>
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

## System Integration Module (SIM)

### 13.1 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 13-1](#). 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 13-1. 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

### 13.2 $\overline{RST}$ 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 programming CONFIG2 accordingly. Refer to [Chapter 5 Configuration Register \(CONFIG\)](#).

## System Integration Module (SIM)

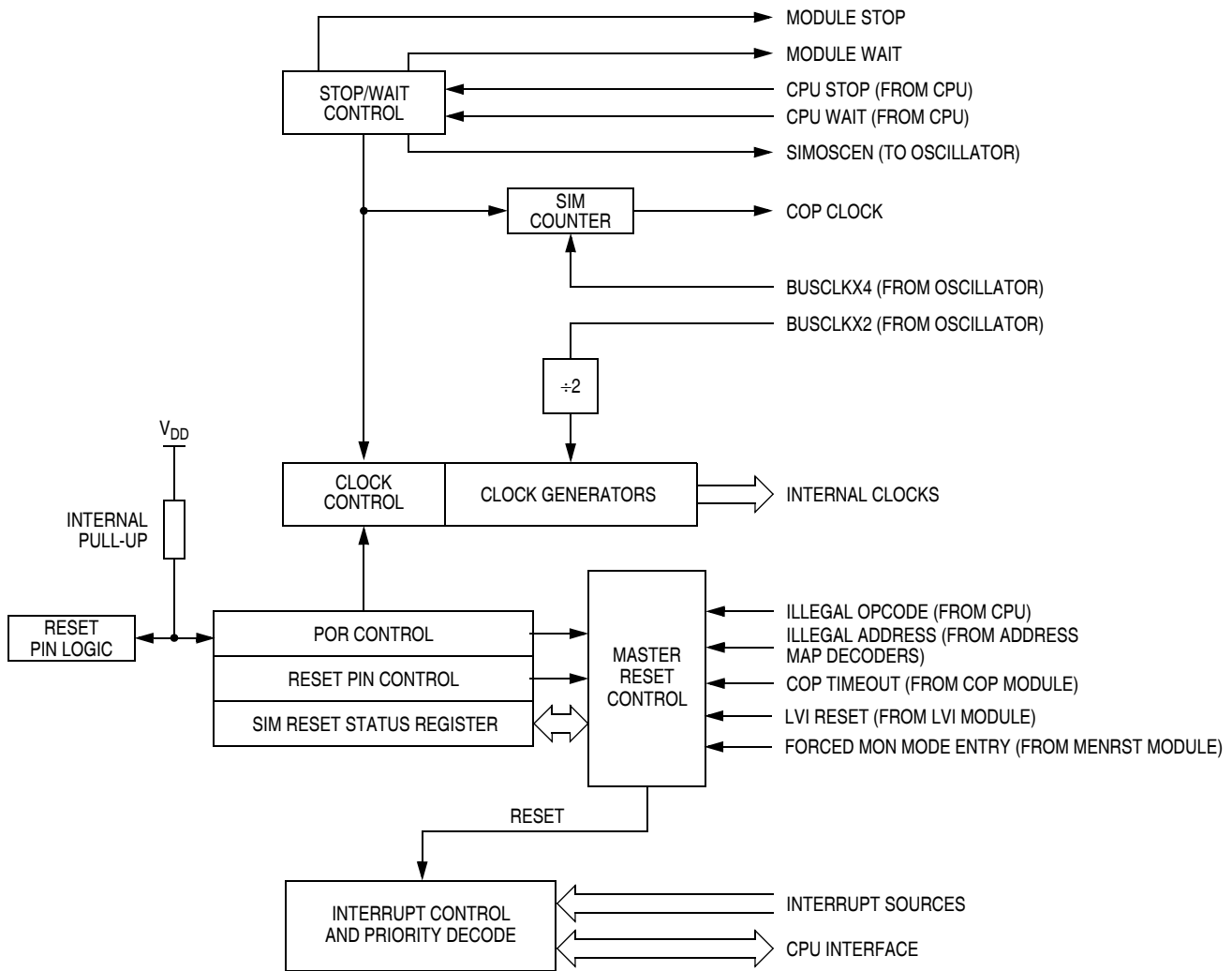


Figure 13-1. SIM Block Diagram

### 13.3 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 13-2.

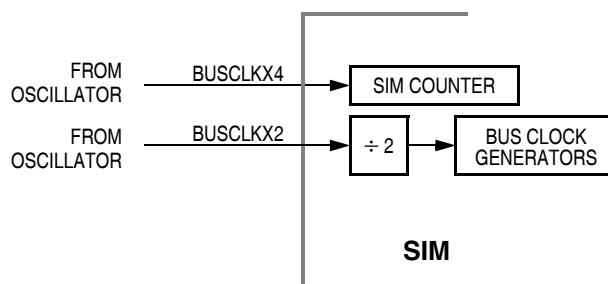


Figure 13-2. SIM Clock Signals



### 13.3.1 Bus Timing

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

### 13.3.2 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.

### 13.3.3 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 [13.7.2 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.

## 13.4 Reset and System Initialization

The MCU has these reset sources:

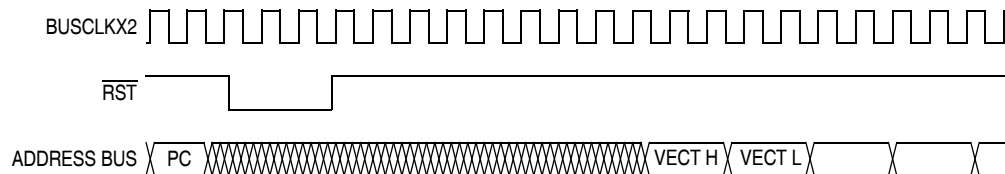
- Power-on reset module (POR)
- External reset pin ( $\overline{\text{RST}}$ )
- Computer operating properly module (COP)
- Low-voltage inhibit module (LVI)
- Illegal opcode
- Illegal address

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 [13.5 SIM Counter](#)), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). See [13.8 SIM Registers](#).

### 13.4.1 External Pin Reset

The  $\overline{\text{RST}}$  pin circuits include an internal pullup device. Pulling the asynchronous  $\overline{\text{RST}}$  pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as  $\overline{\text{RST}}$  is held low for at least the minimum  $t_{\text{RL}}$  time. [Figure 13-3](#) shows the relative timing. The  $\overline{\text{RST}}$  pin function is only available if the RSTEN bit is set in the CONFIG2 register.



**Figure 13-3. External Reset Timing**

### 13.4.2 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.

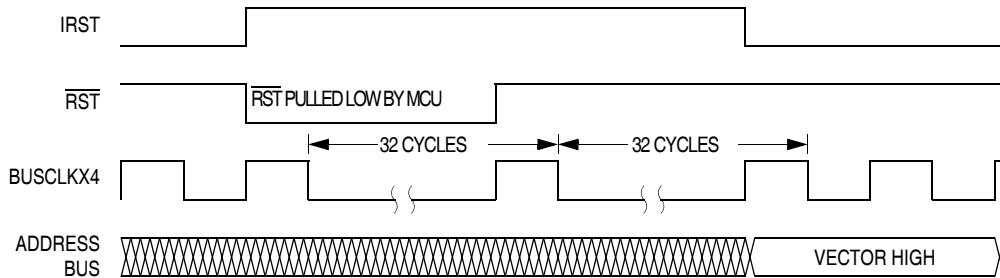
**NOTE**

*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 13-4.*

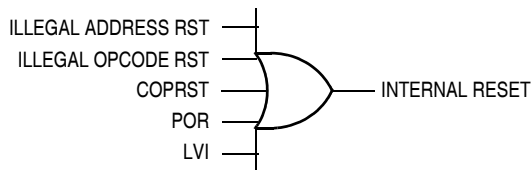
*The COP reset is asynchronous to the bus clock.*

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 13-4). An internal reset can be caused by an illegal address, illegal opcode, COP time out, LVI, or POR (see Figure 13-5).



**Figure 13-4. Internal Reset Timing**



**Figure 13-5. Sources of Internal Reset**

**Table 13-2. Reset Recovery Timing**

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

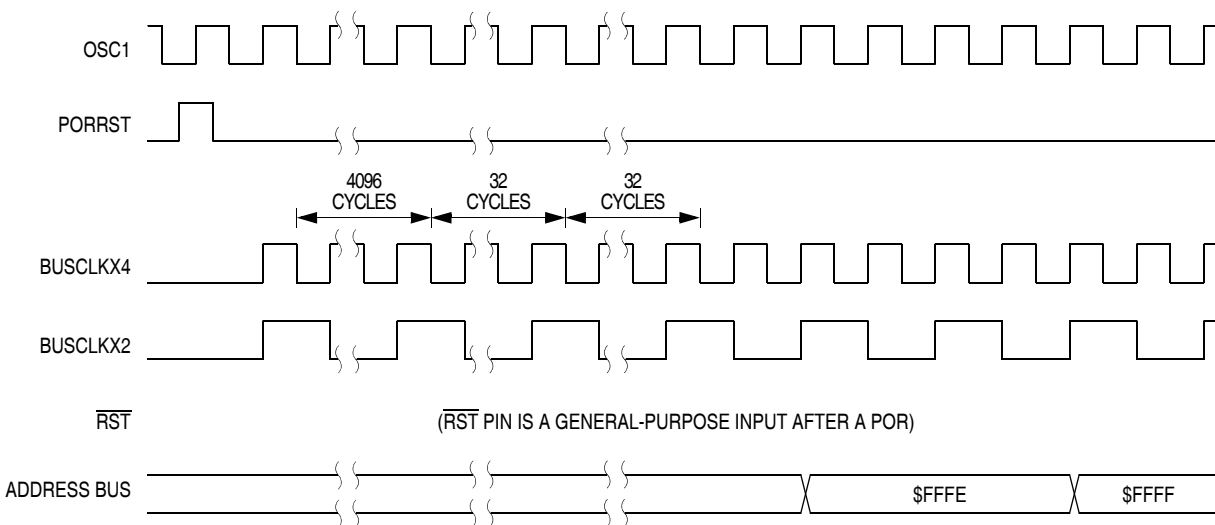
### 13.4.2.1 Power-On Reset

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 13-6](#).



**Figure 13-6. POR Recovery**

### 13.4.2.2 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 4080 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).

### 13.4.2.3 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.

### 13.4.2.4 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.

### 13.4.2.5 Low-Voltage Inhibit (LVI) Reset

The LVI asserts its output to the SIM when the  $V_{\text{DD}}$  voltage falls to the LVI trip voltage  $V_{\text{TRIPF}}$ . 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  $V_{\text{DD}}$  rises above  $V_{\text{TRIPR}}$ . 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.

## 13.5 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.

### 13.5.1 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.

### 13.5.2 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).

### 13.5.3 SIM Counter and Reset States

External reset has no effect on the SIM counter (see [13.7.2 Stop Mode](#) for details.) The SIM counter is free-running after all reset states. See [13.4.2 Active Resets from Internal Sources](#) for counter control and internal reset recovery sequences.

## 13.6 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

### 13.6.1 Interrupts

An interrupt temporarily changes the sequence of program execution to respond to a particular event. [Figure 13-7](#) 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 13-8](#) shows interrupt entry timing. [Figure 13-9](#) shows interrupt recovery timing.

#### 13.6.1.1 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 13-10](#) 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.*

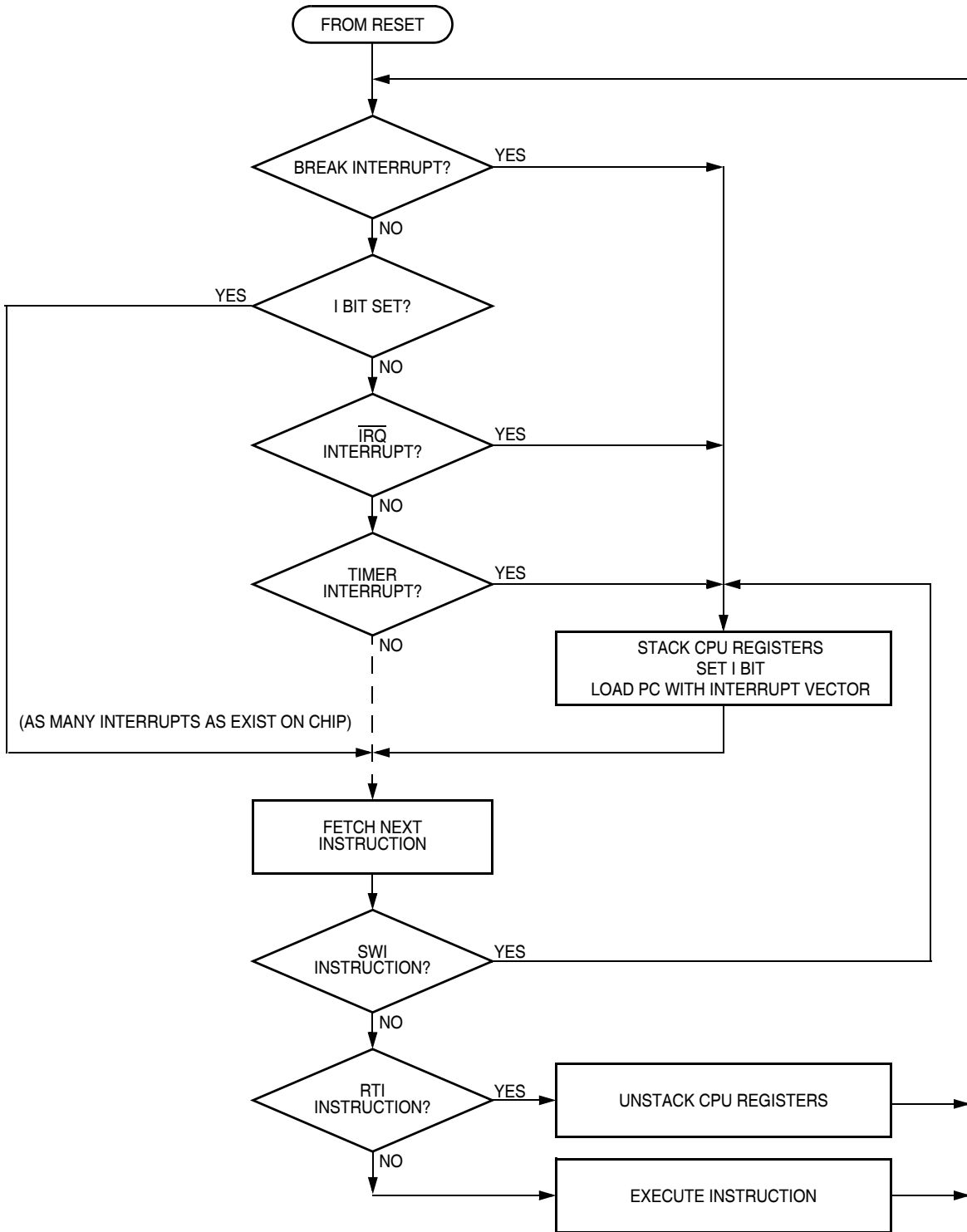
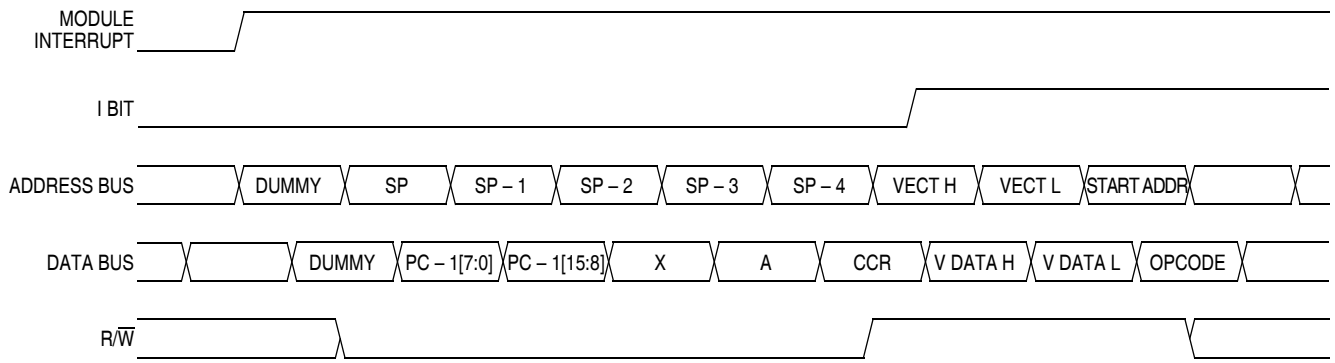
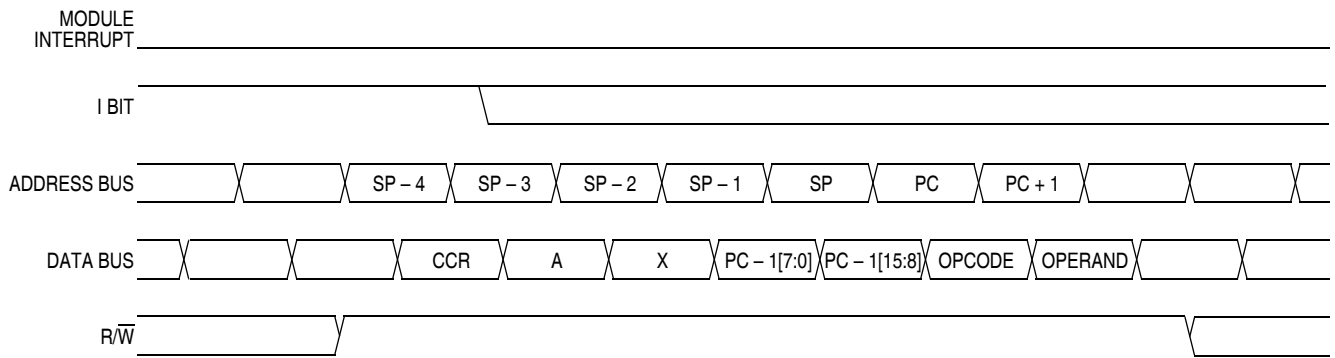


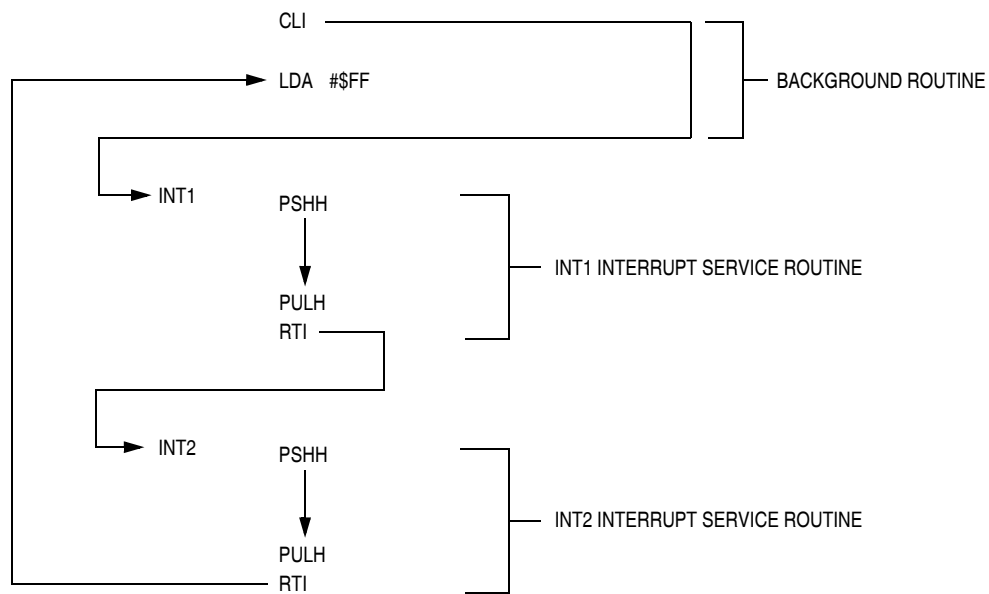
Figure 13-7. Interrupt Processing



**Figure 13-8. Interrupt Entry**



**Figure 13-9. Interrupt Recovery**



**Figure 13-10. Interrupt Recognition Example**

### 13.6.1.2 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.

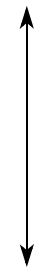
**NOTE**

*A software interrupt pushes PC onto the stack. A software interrupt does **not** push PC – 1, as a hardware interrupt does.*

### 13.6.2 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. [Table 13-3](#) summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

**Table 13-3. Interrupt Sources**

Priority	Source	Flag	Mask <sup>(1)</sup>	INT Register Flag	Vector Address
Highest  Lowest	Reset	—	—	—	\$FFFE–\$FFFF
	SWI instruction	—	—	—	\$FFFC–\$FFFD
	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
	Reserved for test <sup>(2)</sup>	—	—	IF8	\$FFEC–\$FFED
	Keyboard interrupt	KEYF	IMASKK	IF14	\$FFE0–\$FFE1
	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.

2. This vector location is reserved for test. Factory programmed with \$00 at location \$FFEC and \$83 at location \$FFED.



### 13.6.2.1 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 13-11. 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 13-3](#).

1 = Interrupt request present

0 = No interrupt request present

#### Bit 0, 1, and 3— Always read 0

### 13.6.2.2 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 13-12. Interrupt Status Register 2 (INT2)**

#### IF7–IF14 — Interrupt Flags

This flag indicates the presence of interrupt requests from the sources shown in [Table 13-3](#).

1 = Interrupt request present

0 = No interrupt request present

### 13.6.2.3 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 13-13. Interrupt Status Register 3 (INT3)**

#### IF15–IF22 — Interrupt Flags

These flags indicate the presence of interrupt requests from the sources shown in [Table 13-3](#).

1 = Interrupt request present

0 = No interrupt request present

### 13.6.3 Reset

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

### 13.6.4 Break Interrupts

The break module can stop normal program flow at a software programmable break point by asserting its break interrupt output. (See [Chapter 15 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.

### 13.6.5 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.

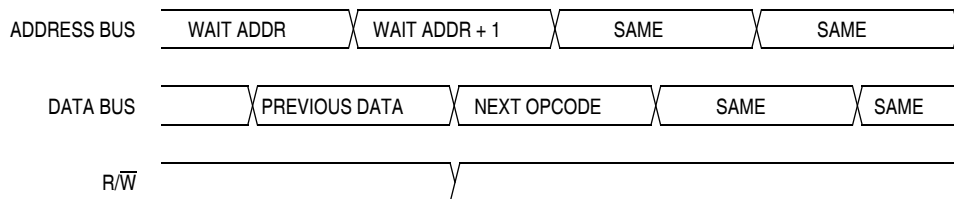
Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a two-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step is accomplished prior to entering break mode. Upon leaving break mode, execution of the second step will clear the flag as normal.

## 13.7 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.

### 13.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. [Figure 13-14](#) shows the timing for wait mode entry.



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

**Figure 13-14. 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 13-15 and Figure 13-16 show the timing for wait recovery.

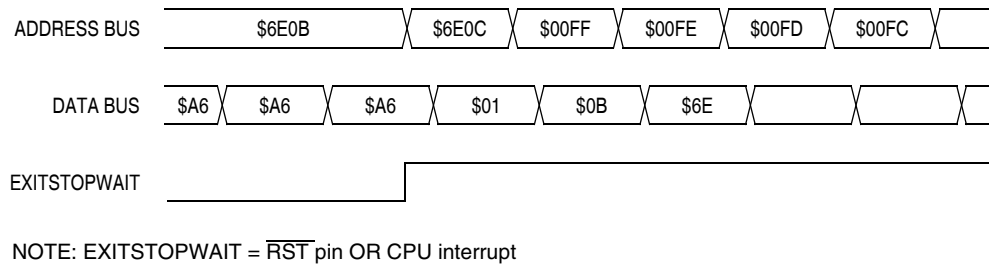
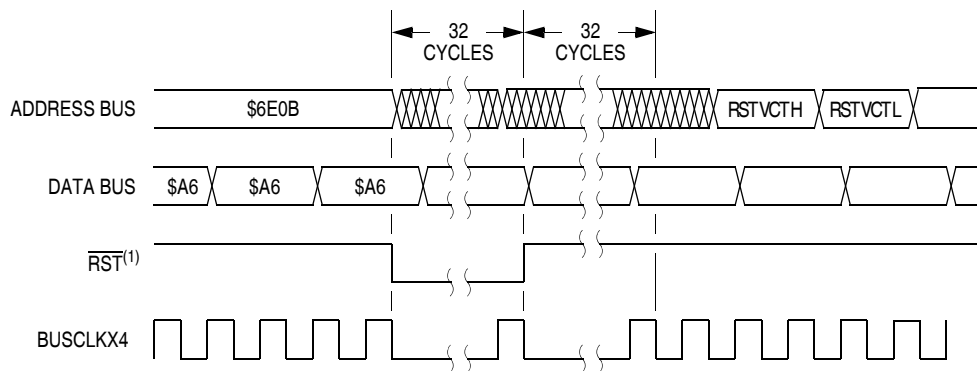


Figure 13-15. Wait Recovery from Interrupt



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

Figure 13-16. Wait Recovery from Internal Reset

### 13.7.2 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, BUSCLKX2 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**

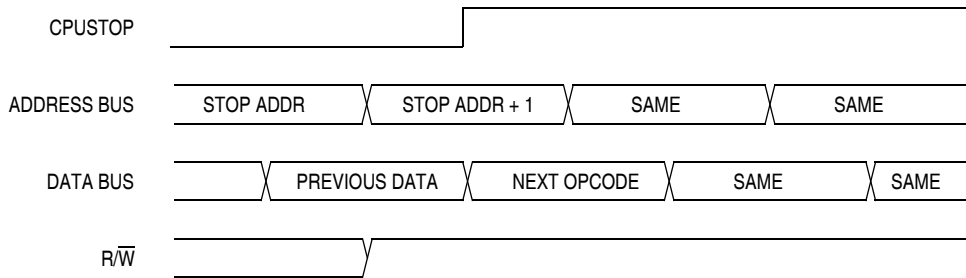
*External crystal applications should use the full stop recovery time by clearing the SSREC bit.*

## System Integration Module (SIM)

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 13-17 shows stop mode entry timing and Figure 13-18 shows the stop mode recovery time from interrupt or break

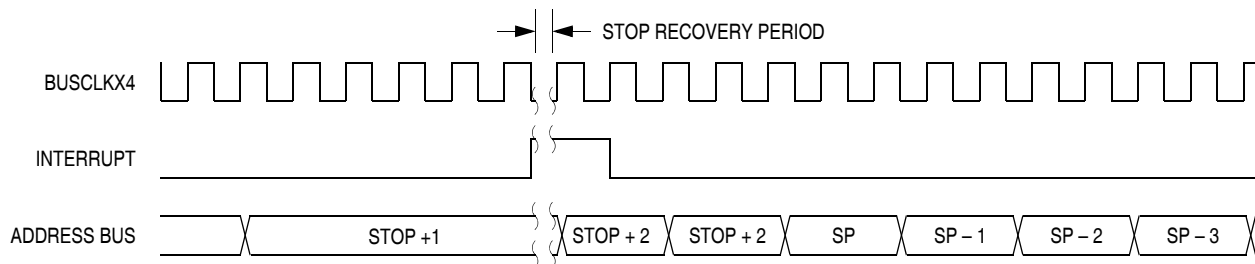
### NOTE

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 13-17. Stop Mode Entry Timing**



**Figure 13-18. Stop Mode Recovery from Interrupt**

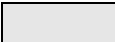
## 13.8 SIM Registers

The SIM has two memory mapped registers.

### 13.8.1 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 13-19. 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**

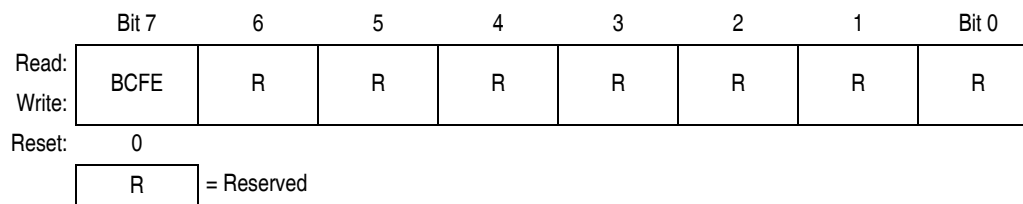
- 1 = Last reset caused by monitor mode entry when vector locations \$FFFE and \$FFFF are \$FF after POR while  $IRQ \neq V_{TST}$
- 0 = POR or read of SRSR

**LVI — Low Voltage Inhibit Reset bit**

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

**13.8.2 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.



**Figure 13-20. 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 14

## Timer Interface Module (TIM)

### 14.1 Introduction

This section describes the timer interface module (TIM). The TIM module is a 2-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 14-1](#) for port location of these shared pins.

### 14.2 Features

Features include the following:

- Two input capture/output compare channels
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
- Buffered and unbuffered 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 14-1](#)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- Counter stop and reset bits

### 14.3 Functional Description

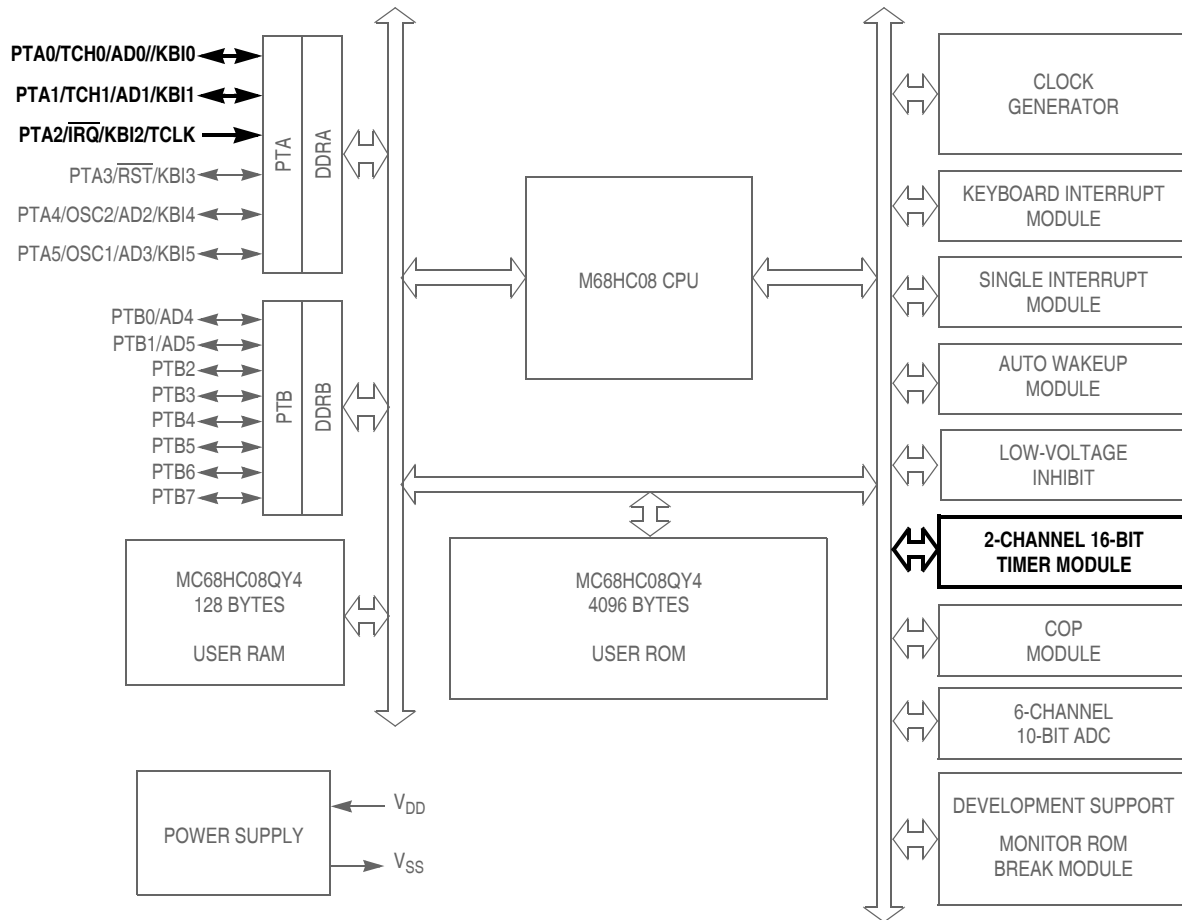
[Figure 14-2](#) shows the structure of the TIM. The central component of the TIM is the 16-bit counter that can operate as a free-running counter or a modulo up-counter. The counter provides the timing reference for the input capture and output compare functions. The counter modulo registers, TMODH:TMODL, control the modulo value of the counter. Software can read the counter value, TCNTH:TCNTL, at any time without affecting the counting sequence.

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

#### 14.3.1 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.

## Timer Interface Module (TIM)



$\overline{RST}$ ,  $\overline{IRQ}$ : Pins have internal pull up device  
 All port pins have programmable pull up device  
 PTA[0:5]: Higher current sink and source capability  
 PTB[0:7]: Not available on 8-pin devices

**Figure 14-1. Block Diagram Highlighting TIM Block and Pins**

### 14.3.2 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.

### 14.3.3 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.



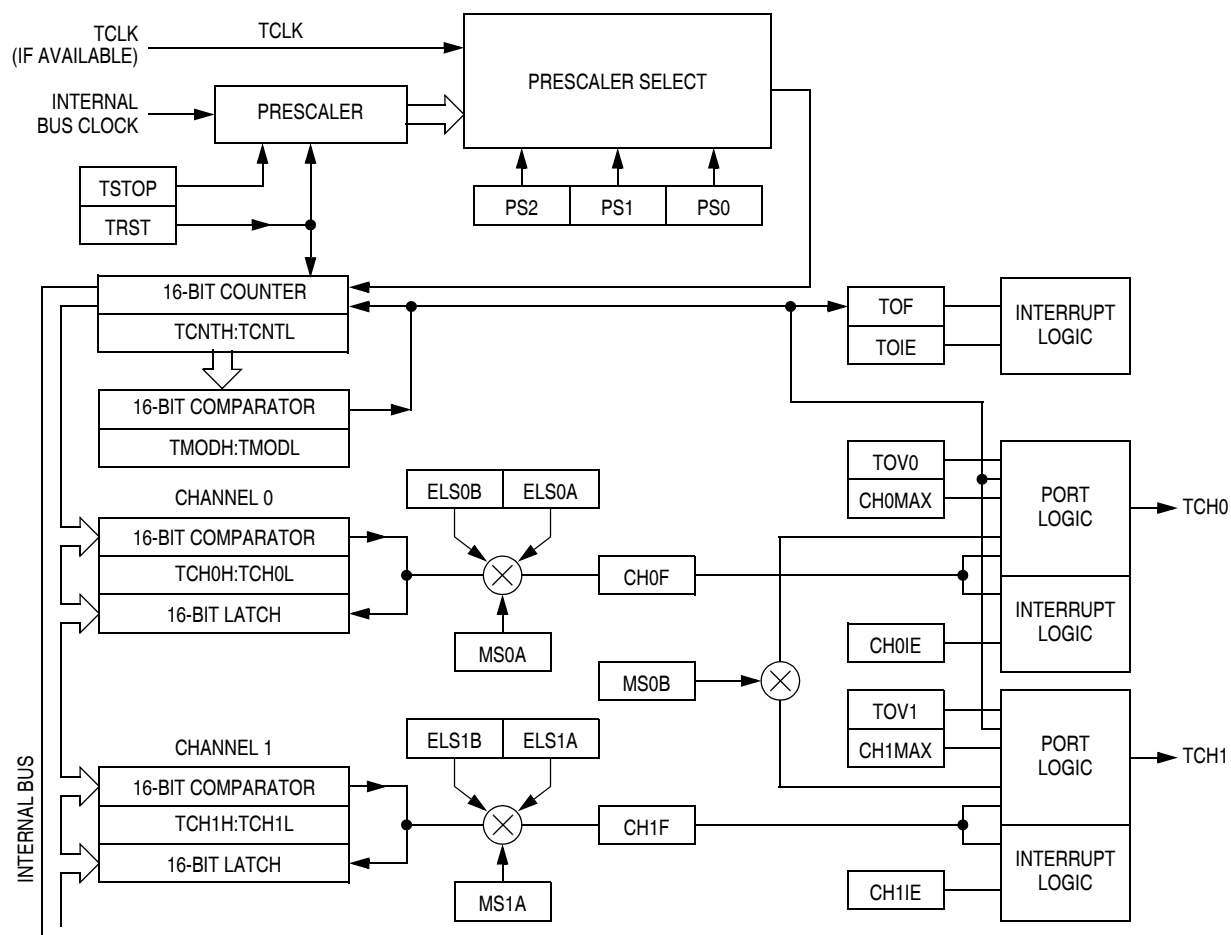


Figure 14-2. TIM Block Diagram

### 14.3.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in [14.3.3 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.

### 14.3.3.2 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.

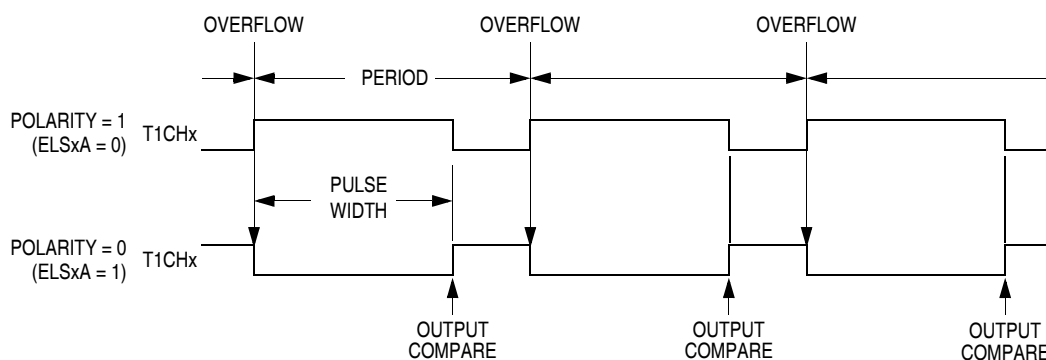
#### NOTE

*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.*

### 14.3.4 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 14-3](#) 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).



**Figure 14-3. PWM Period and Pulse Width**

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 [14.8.1 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%.

#### 14.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in [14.3.4 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.*

#### 14.3.4.2 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.

#### 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.*

### 14.3.4.3 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 14-2](#).
  - 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 14-2](#).

#### **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.

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 [14.8.1 TIM Status and Control Register](#).

## 14.4 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 (CH1F: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.

## 14.5 Low-Power Modes

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

### 14.5.1 Wait Mode

The 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.

### 14.5.2 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.

## 14.6 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.

## 14.7 I/O Signals

The TIM module can share its pins with the general-purpose I/O pins. See [Figure 14-1](#) for the port pins that are shared.

### 14.7.1 TIM Channel I/O Pins (TCH1:TCH0)

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

### 14.7.2 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 [14.8.1 TIM Status and Control Register](#). The minimum TCLK pulse width is specified in the Timer Interface Module Characteristics table in the Electricals section. The maximum TCLK frequency is the least of 4 MHz or bus frequency  $\div 2$ .

## 14.8 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 and TSC1)
- TIM channel registers (TCH0H:TCH0L and TCH1H:TCH1L)

### 14.8.1 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 14-4. 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.*

#### 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 14-1](#) shows.

**Table 14-1. Prescaler Selection**

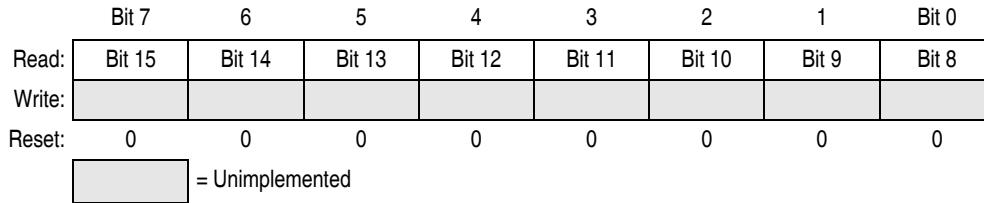
PS2	PS1	PS0	TIM Clock Source
0	0	0	Internal bus clock ÷ 1
0	0	1	Internal bus clock ÷ 2
0	1	0	Internal bus clock ÷ 4
0	1	1	Internal bus clock ÷ 8
1	0	0	Internal bus clock ÷ 16
1	0	1	Internal bus clock ÷ 32
1	1	0	Internal bus clock ÷ 64
1	1	1	TCLK (if available)

### 14.8.2 TIM Counter Registers

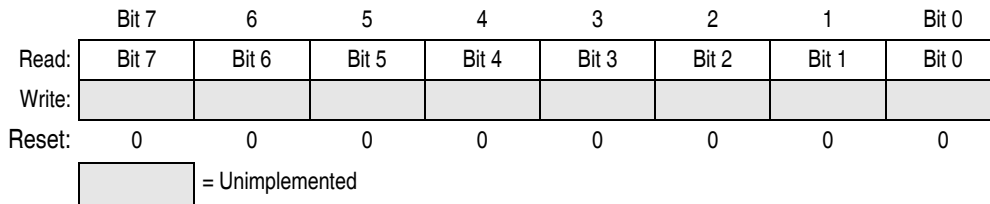
The two read-only TIM counter registers contain the high and low bytes of the value in the counter. Reading the high byte (TCNTH) latches the contents of the low byte (TCNTL) into a buffer. Subsequent reads of TCNTH do not affect the latched TCNTL value until TCNTL is read. Reset clears the TIM counter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



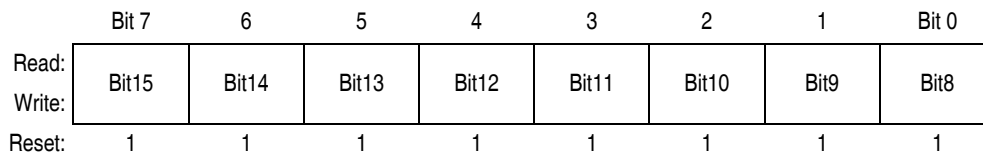
**Figure 14-5. TIM Counter High Register (TCNTH)**



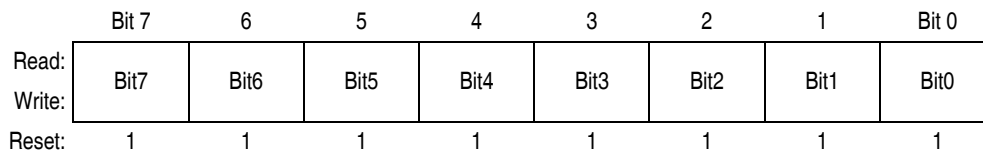
**Figure 14-6. TIM Counter Low Register (TCNTL)**

### 14.8.3 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.



**Figure 14-7. TIM Counter Modulo High Register (TMODH)**



**Figure 14-8. TIM Counter Modulo Low Register (TMODL)**

**NOTE**

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



## 14.8.4 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 14-9. 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

= Unimplemented

**Figure 14-10. TIM Channel 1 Status and Control Register (TSC1)**

### 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.

## Timer Interface Module (TIM)

Setting MS0B causes the contents of TSC1 to be ignored by the TIM and reverts TCH1 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 14-2](#).

- 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 14-2](#)).

- 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).*

### 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 14-2](#) shows how ELSxB and ELSxA work.

**Table 14-2. Mode, Edge, and Level Selection**

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

### NOTE

*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 TIM counter overflow.

0 = Channel x pin does not toggle on TIM 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 14-11 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.

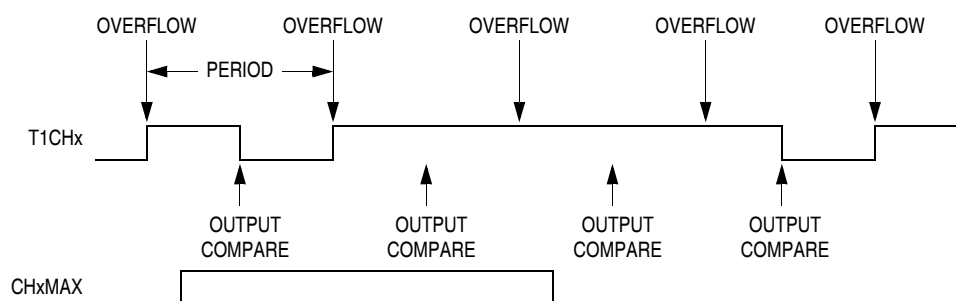


Figure 14-11. CHxMAX Latency

## 14.8.5 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 ( $MSxB:MSxA \neq 0:0$ ), writing to the high byte of the TIM channel x registers (TCHxH) inhibits output compares until the low byte (TCHxL) is written.

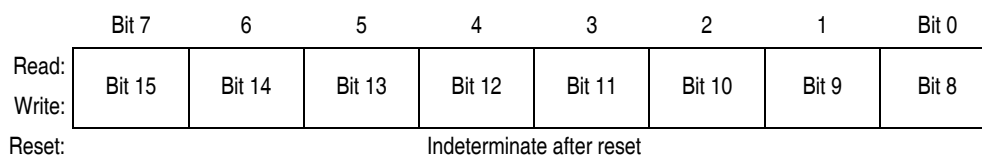


Figure 14-12. TIM Channel x Register High (TCHxH)

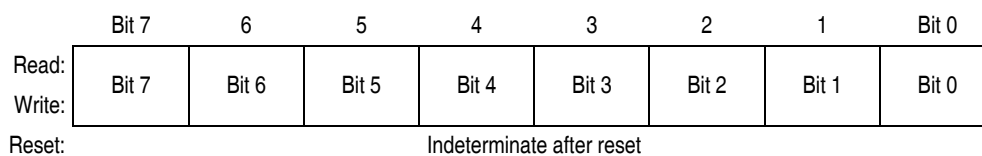


Figure 14-13. TIM Channel x Register Low (TCHxL)



# Chapter 15

## Development Support

### 15.1 Introduction

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

### 15.2 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

#### 15.2.1 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:

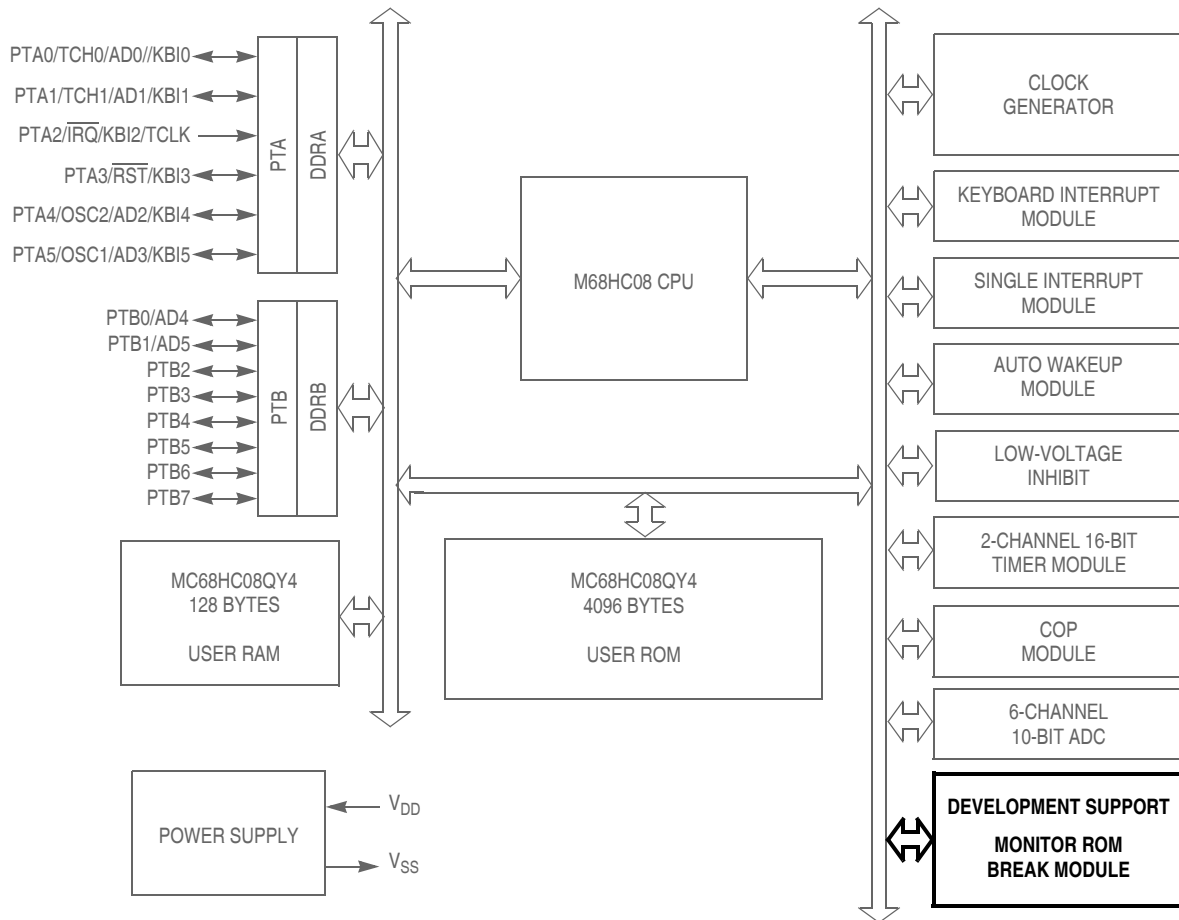
- A CPU generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a 1 to the BRKA bit in the break status and control register.

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 15-2 shows the structure of the break module.

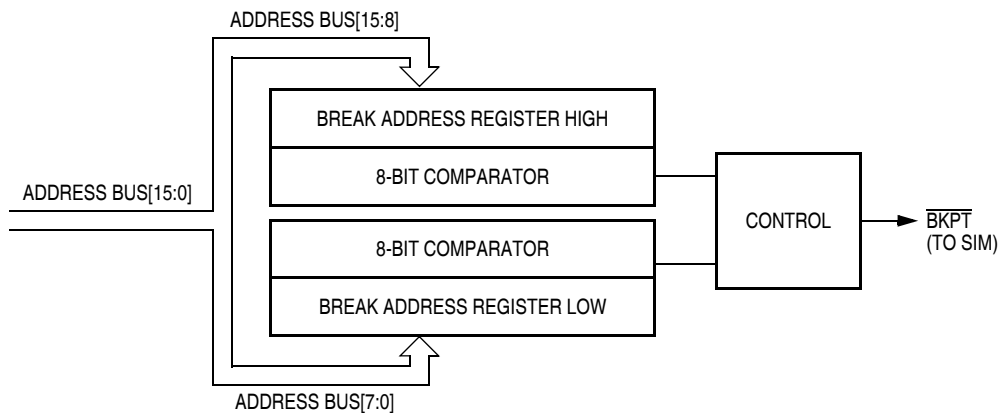
When the internal address bus matches the value written in the break address registers or when software writes a 1 to the BRKA bit in the break status and control register, the CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with \$FFFC and \$FFFD (\$FEFC and \$FEFD in monitor mode)



$\overline{RST}$ ,  $\overline{IRQ}$ : Pins have internal pull up device  
 All port pins have programmable pull up device  
 PTA[0:5]: Higher current sink and source capability  
 PTB[0:7]: Not available on 8-pin devices

**Figure 15-1. Block Diagram Highlighting BRK and MON Blocks**



**Figure 15-2. 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.*

#### **15.2.1.1 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 (BF CR) enables software to clear status bits during the break state. See [13.8.2 Break Flag Control Register](#) and the **Break Interrupts** subsection for each module.

#### **15.2.1.2 TIM During Break Interrupts**

A break interrupt stops the timer counter.

#### **15.2.1.3 COP During Break Interrupts**

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

## 15.2.2 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)

### 15.2.2.1 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 15-3. 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

### 15.2.2.2 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 15-4. 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 15-5. Break Address Register Low (BRKL)**



### 15.2.2.3 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.

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

 = Unimplemented

**Figure 15-6. 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.

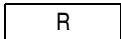
1 = COP disabled during break interrupt

0 = COP enabled during break interrupt

### 15.2.2.4 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.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	R	R	R	R	R	SBSW	R
Write:							Note <sup>(1)</sup>	
Reset:							0	

 = Reserved

1. Writing a 0 clears SBSW.

**Figure 15-7. 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.

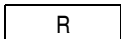
1 = Wait mode was exited by break interrupt

0 = Wait mode was not exited by break interrupt

### 15.2.2.5 Break Flag Control Register

The break control register (BF CR) 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:	BCFE	R	R	R	R	R	R	R
Write:								
Reset:	0							

 = Reserved

**Figure 15-8. Break Flag Control Register (BF CR)**

**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

**15.2.3 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.

**15.3 Monitor Module (MON)**

The monitor module allows debugging and programming of the microcontroller unit (MCU) through a single-wire interface with a host computer.

Features include:

- Normal user-mode pin functionality on most pins
- One pin dedicated to serial communication between MCU and host computer
- Standard communication baud rates, using external 9.8304 MHz oscillator
- Standard non-return-to-zero (NRZ) communication with host computer
- Execution of code in random-access memory (RAM) or read-only memory (ROM)
- ROM memory security feature<sup>(1)</sup>
- Reset into monitor mode if  $V_{TST}$  is applied to  $\overline{IRQ}$
- Use of internal oscillator once monitor mode entered and high voltage removed

**15.3.1 Functional Description**

Figure 15-9 shows a simplified diagram of monitor mode entry.

The monitor module receives and executes commands from a host computer. Figure 15-10 shows an example circuit 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 ROM difficult for unauthorized users.

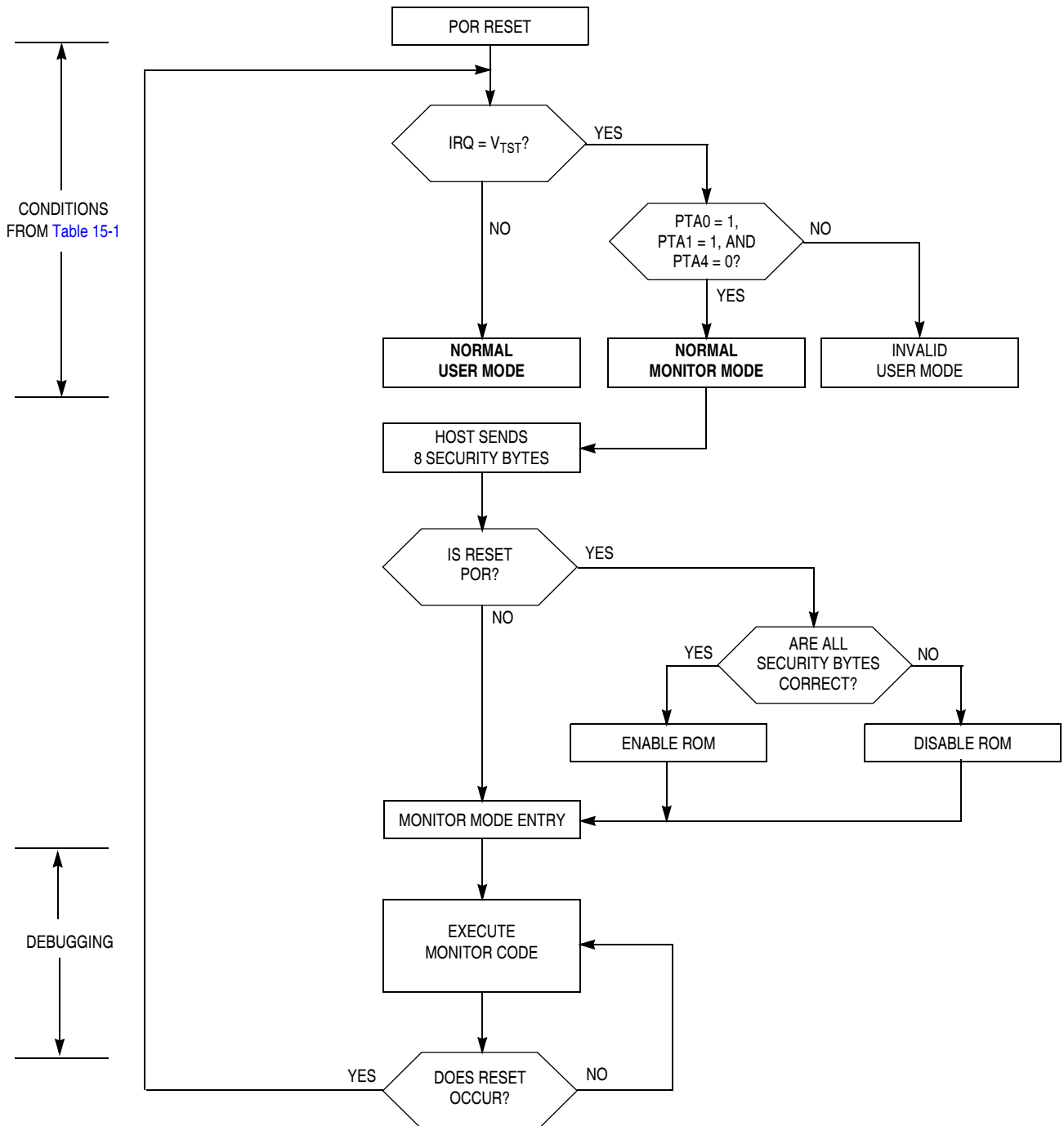
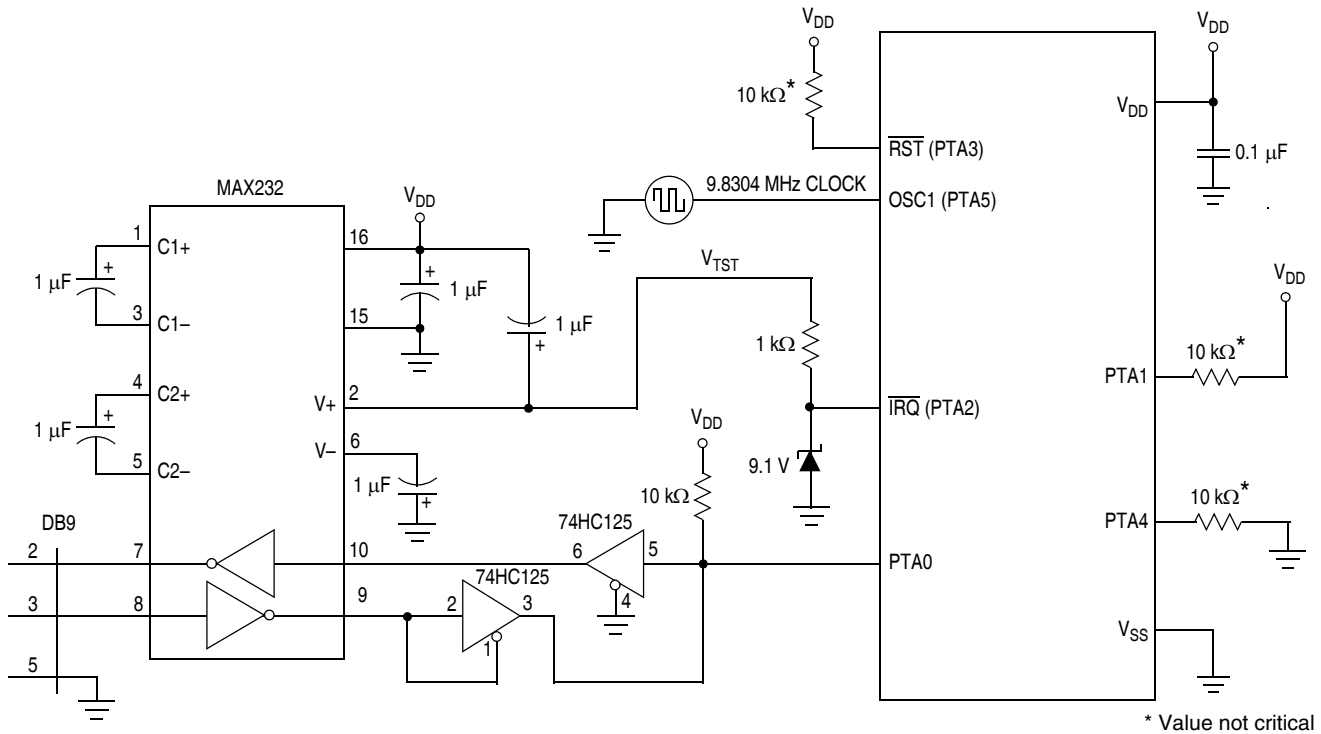


Figure 15-9. Simplified Monitor Mode Entry Flowchart



**Figure 15-10. Monitor Mode Circuit (External Clock, with 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 returned to normal logic levels after a reset into monitor mode is entered, is intended to support serial communication at 9600 baud in monitor mode by using the internal oscillator, and the internal oscillator user trim value OSCTRIM (ROM location \$FFC0) to generate the desired internal frequency (3.2 MHz). The  $\overline{IRQ}$  pin must remain at normal logic levels during the monitor session in order to maintain communication.

Table 15-1 shows the pin conditions for entering monitor mode. As specified in the table, monitor mode may be entered after a power-on reset (POR).

The rising edge of the internal  $\overline{RST}$  signal with  $V_{TST}$  applied to  $\overline{IRQ}$  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 15.3.2 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 15-1. Monitor Mode Signal Requirements and Options

Mode	$\overline{\text{IRQ}}$ (PTA2)	$\overline{\text{RST}}$ (PTA3)	Reset Vector	Serial Communi- cation	Mode Selection		COP	Communication Speed			Comments
					PTA0	PTA1		PTA4	External Clock	Bus Frequency	
Normal Monitor	$V_{\text{TST}}$	$V_{\text{DD}}$	X	1	1	0	Disabled	9.8304 MHz	2.4576 MHz	9600	Provide external clock at OSC1.
User	$V_{\text{DD}}$ to $V_{\text{SS}}$	X	X	X	X	X	Enabled	X	X	X	
MON08 Function [Pin No.]	$V_{\text{TST}}$ [6]	$\overline{\text{RST}}$ [4]	—	COM [8]	MOD0 [12]	MOD1 [10]	—	OSC1 [13]	—	—	

- PTA0 must have a pullup resistor to  $V_{\text{DD}}$  in monitor mode.
- 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.
- External clock is a 9.8304 MHz oscillator on OSC1.
- Lowering  $V_{\text{TST}}$  once monitor mode is entered allows clock source to be controlled by the OSCSC register.
- X = don't care
- 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
$V_{\text{DD}}$	15	16	NC

### 15.3.1.1 Normal Monitor Mode

$\overline{\text{RST}}$  and OSC1 functions will be active on the PTA3 and PTA5 pins respectively as long as  $V_{\text{TST}}$  is applied to the  $\overline{\text{IRQ}}$  pin. If the  $\overline{\text{IRQ}}$  pin is lowered (no longer  $V_{\text{TST}}$ ) then the MCU 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 \(CONFIG\)](#) and [11.8.1 Oscillator Status and Control Register](#)) when  $V_{\text{TST}}$  was lowered. With  $V_{\text{TST}}$  lowered, the BIH and BIL instructions will read the  $\overline{\text{IRQ}}$  pin state only if IRQEN is set in the CONFIG2 register.

### 15.3.1.2 Monitor Vectors

In monitor mode, the MCU uses different vectors for reset, SWI (software interrupt), and break interrupt than those for user mode. The alternate vectors are in the \$FE page instead of the \$FF page and allow code execution from the internal monitor firmware instead of user code.

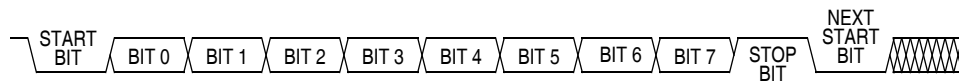
Table 15-2 summarizes the differences between user mode and monitor mode regarding vectors.

**Table 15-2. 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

### 15.3.1.3 Data Format

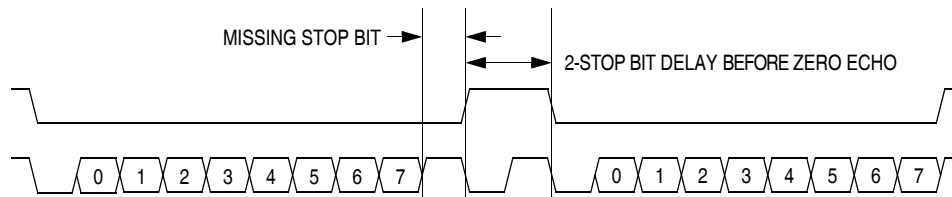
Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. Transmit and receive baud rates must be identical.



**Figure 15-11. Monitor Data Format**

### 15.3.1.4 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 15-12. Break Transaction**

### 15.3.1.5 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 15-1.

Table 15-1 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.

### 15.3.1.6 Commands

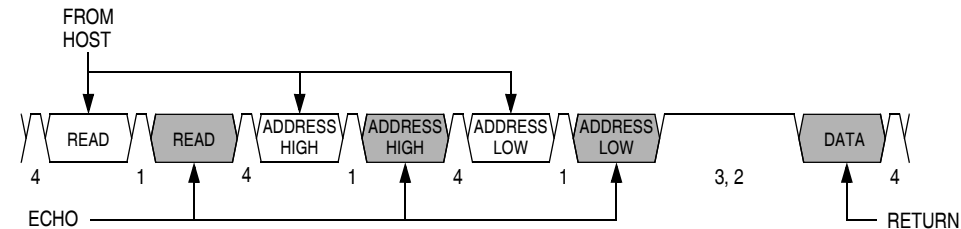
The monitor ROM firmware uses these commands:

- 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.

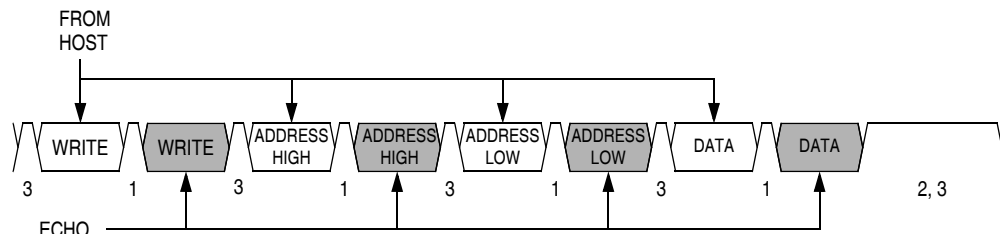
**NOTE**

*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 15-13. 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.

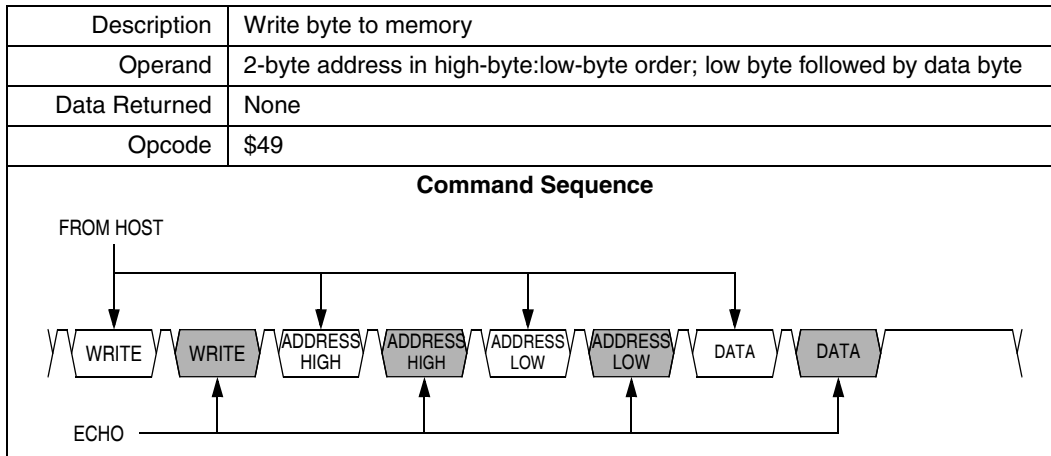
**Figure 15-14. Write Transaction**

A brief description of each monitor mode command is given in [Table 15-3](#) through [Table 15-8](#).

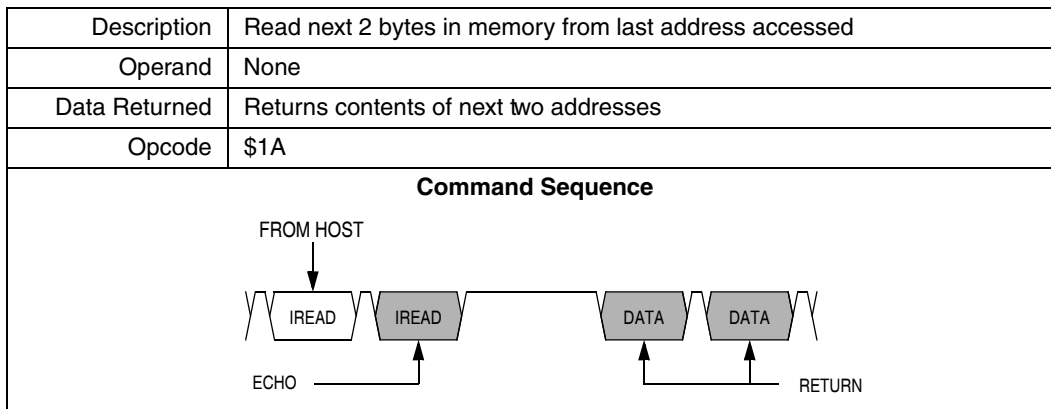
**Table 15-3. READ (Read Memory) Command**

Description	Read byte from memory
Operand	2-byte address in high-byte:low-byte order
Data Returned	Returns contents of specified address
Opcode	\$4A
<b>Command Sequence</b>	
<p>The diagram shows the command sequence: SENT TO MONITOR (READ, ADDRESS HIGH, ADDRESS HIGH, ADDRESS LOW, ADDRESS LOW), ECHO (echoes of READ, ADDRESS HIGH, ADDRESS HIGH, ADDRESS LOW), and DATA. A RETURN signal is also shown.</p>	

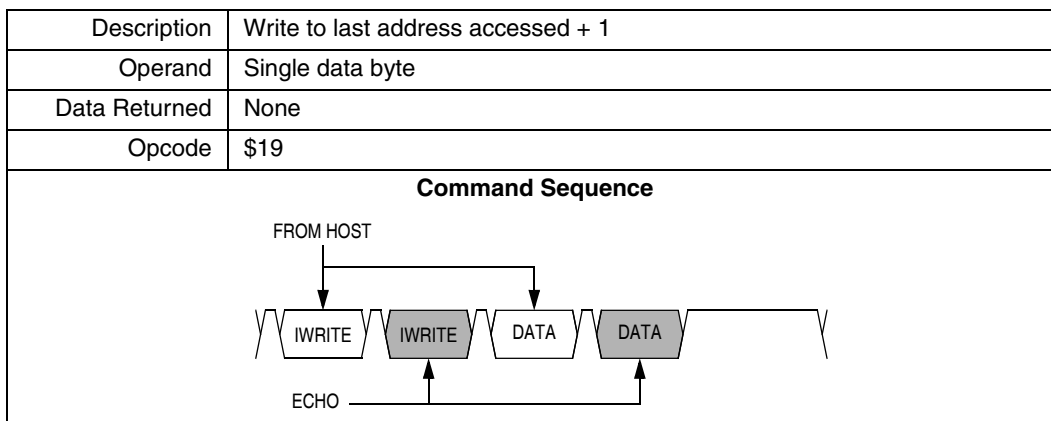
**Table 15-4. WRITE (Write Memory) Command**



**Table 15-5. IREAD (Indexed Read) Command**



**Table 15-6. IWRITE (Indexed Write) Command**



A sequence of IREAD or IWRITE commands can access a block of memory sequentially over the full 64-Kbyte memory map.



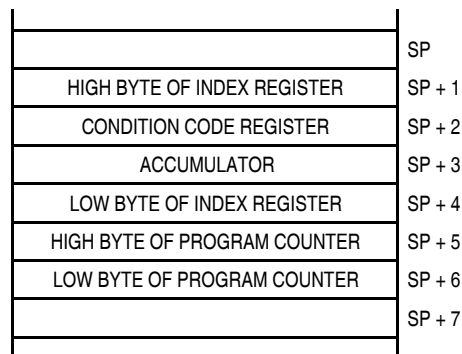
**Table 15-7. READSP (Read Stack Pointer) Command**

Description	Reads stack pointer
Operand	None
Data Returned	Returns incremented stack pointer value (SP + 1) in high-byte:low-byte order
Opcode	\$0C
<b>Command Sequence</b>	

**Table 15-8. RUN (Run User Program) Command**

Description	Executes PULH and RTI instructions
Operand	None
Data Returned	None
Opcode	\$28
<b>Command Sequence</b>	

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 15-15. Stack Pointer at Monitor Mode Entry**

### 15.3.2 Security

A security feature discourages unauthorized reading of ROM 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**

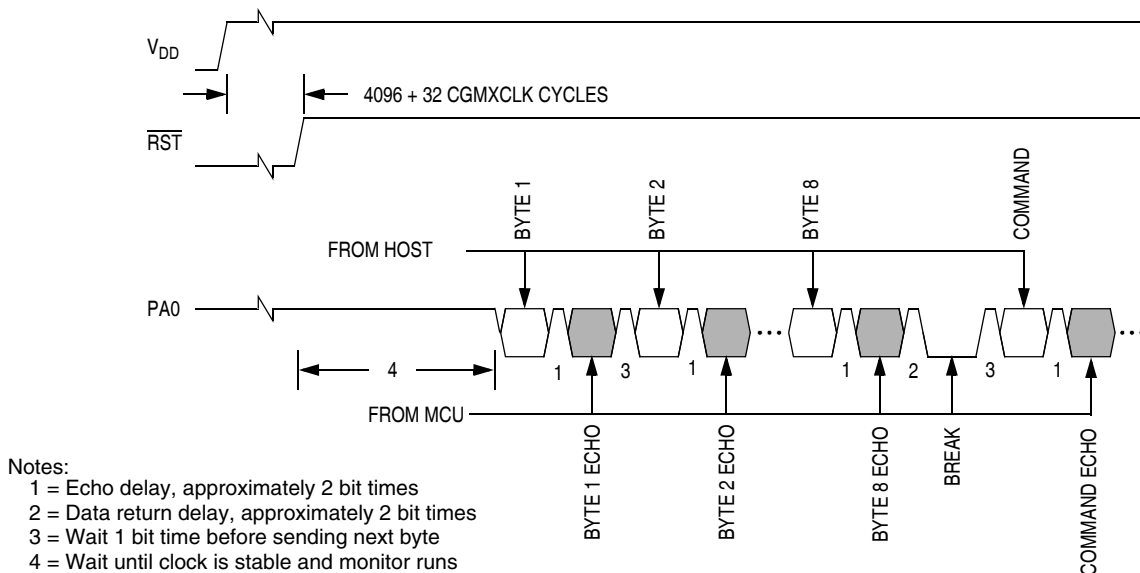
*Do not leave locations \$FFF6–\$FFFD blank. For security reasons, program locations \$FFF6–\$FFFD even if they are not used for vectors.*

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 ROM locations and execute code from ROM. 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 15-16](#).

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 ROM location returns an invalid value and trying to execute code from ROM 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 15-16. 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 ROM 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.

# Chapter 16

## Electrical Specifications

### 16.1 Introduction

This section contains electrical and timing specifications.

### 16.2 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 [16.5 5-V DC Electrical Characteristics](#), [16.8 3-V DC Electrical Characteristics](#) and [16.11 1.8-V to 3.6-V DC Electrical Characteristics](#) for guaranteed operating conditions.*

Characteristic <sup>(1)</sup>	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  $V_{SS} \leq (V_{IN} \text{ or } V_{OUT}) \leq V_{DD}$ . Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either  $V_{SS}$  or  $V_{DD}$ .)*

## 16.3 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	$V_{DD}$	1.8 to 5.5	V	—

## 16.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance 8-pin SOIC 16-pin SOIC 16-pin TSSOP	$\theta_{JA}$	142 90 133	°C/W
I/O pin power dissipation	$P_{I/O}$	User determined	W
Power dissipation <sup>(1)</sup>	$P_D$	$P_D = (I_{DD} \times V_{DD})$ $+ P_{I/O} = K/(T_J + 273^\circ\text{C})$	W
Constant <sup>(2)</sup>	K	$P_D \times (T_A + 273^\circ\text{C})$ $+ P_D^2 \times \theta_{JA}$	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  $T_A$ .

## 16.5 5-V DC Electrical Characteristics

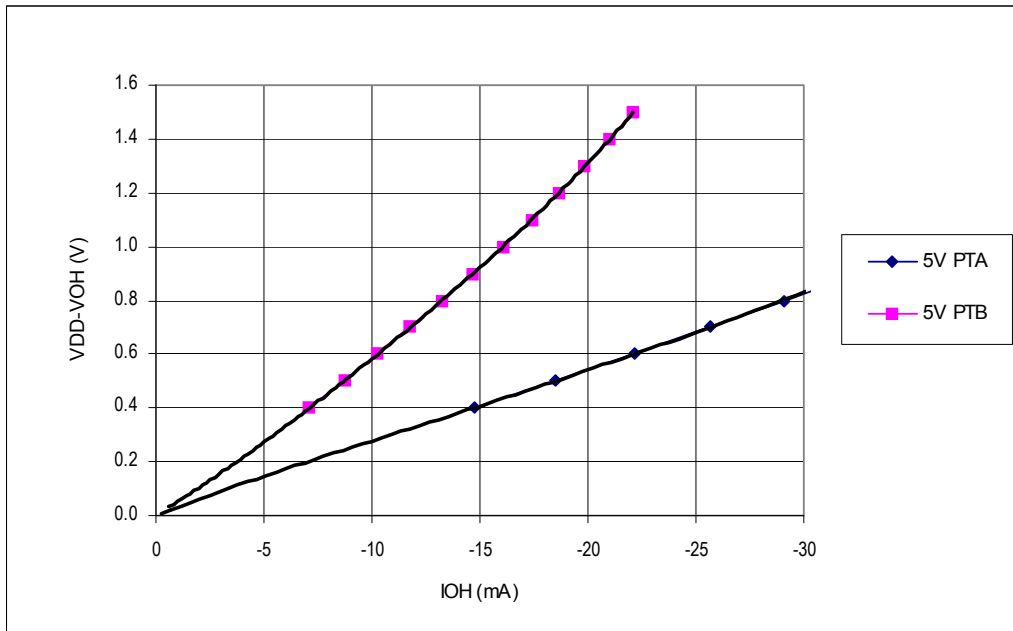
Characteristic <sup>(1)</sup>	Symbol	Min	Typ <sup>(2)</sup>	Max	Unit
Output high voltage $I_{Load} = -2.0$ mA, all I/O pins $I_{Load} = -10.0$ mA, all I/O pins $I_{Load} = -15.0$ mA, PTA0, PTA1, PTA3–PTA5 only	$V_{OH}$	$V_{DD}-0.4$ $V_{DD}-1.5$ $V_{DD}-0.8$	— — —	— — —	V
Maximum combined $I_{OH}$ (all 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

— Continued on next page

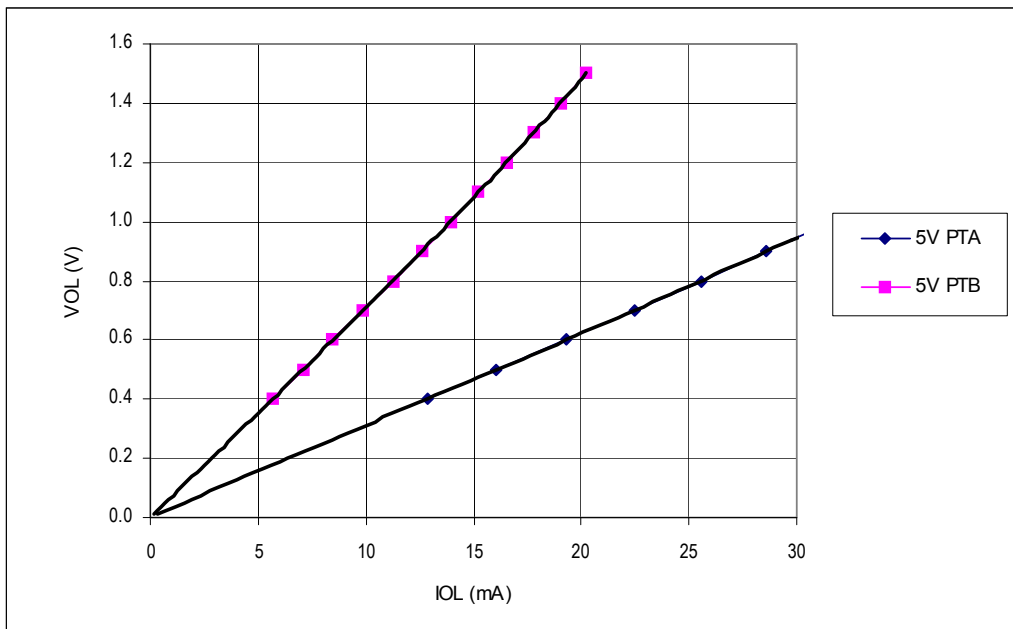
Characteristic <sup>(1)</sup>	Symbol	Min	Typ <sup>(2)</sup>	Max	Unit
Input high voltage PTA0–PTA5, PTB0–PTB7	$V_{IH}$	$0.7 \times V_{DD}$	—	$V_{DD}$	V
Input low voltage PTA0–PTA5, PTB0–PTB7	$V_{IL}$	$V_{SS}$	—	$0.3 \times V_{DD}$	V
Input hysteresis <sup>(3)</sup>	$V_{HYS}$	$0.06 \times V_{DD}$	—	—	V
DC injection current <sup>(3) (4) (5) (6)</sup> Single pin limit $V_{in} > V_{DD}$ $V_{in} < V_{SS}$ Total MCU limit, includes sum of all stressed pins $V_{in} > V_{DD}$ $V_{in} < V_{SS}$	$I_{IC}$	0 0	— —	2 –0.2	mA
Ports Hi-Z leakage current	$I_{IL}$	0	—	$\pm 1$	$\mu A$
Capacitance Ports (as input) <sup>(3)</sup>	$C_{IN}$	—	—	8	pF
POR rearm voltage	$V_{POR}$	750	—	—	mV
POR rise time ramp rate <sup>(3)(7)</sup>	$R_{POR}$	0.035	—	—	V/ms
Monitor mode entry voltage <sup>(3)</sup>	$V_{TST}$	$V_{DD} + 2.5$	—	9.1	V
Pullup resistors <sup>(8)</sup> PTA0–PTA5, PTB0–PTB7	$R_{PU}$	16	26	36	k $\Omega$
Pulldown resistors <sup>(9)</sup> PTA0–PTA5	$R_{PD}$	16	26	36	k $\Omega$
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

- $V_{DD} = 4.5$  to  $5.5$  Vdc,  $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$ , unless otherwise noted.
- Typical values reflect average measurements at midpoint of voltage range,  $25^\circ C$  only.
- Values are based on characterization results, not tested in production.
- All functional non-supply pins are internally clamped to  $V_{SS}$  and  $V_{DD}$ .
- Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.
- Power supply must maintain regulation within operating  $V_{DD}$  range during instantaneous and operating maximum current conditions. If positive injection current ( $V_{in} > V_{DD}$ ) is greater than  $I_{DD}$ , the injection current may flow out of  $V_{DD}$  and could result in external power supply going out of regulation. Ensure external  $V_{DD}$  load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if clock rate is very low (which would reduce overall power consumption).
- 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.
- $R_{PU}$  is measured at  $V_{DD} = 5.0$  V.
- $R_{PD}$  is measured at  $V_{DD} = 5.0$  V, Pulldown resistors only available when KBlx is enabled with  $KBlxPOL = 1$ .

## 16.6 Typical 5-V Output Drive Characteristics



**Figure 16-1. Typical 5-Volt Output High Voltage versus Output High Current (25°C)**



**Figure 16-2. Typical 5-Volt Output Low Voltage versus Output Low Current (25°C)**

## 16.7 5-V Control Timing

Characteristic <sup>(1)</sup>	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 <sup>(2)</sup>	$t_{RL}$	100	—	ns
$\overline{IRQ}$ interrupt pulse width low (edge-triggered) <sup>(2)</sup>	$t_{LIH}$	100	—	ns
$\overline{IRQ}$ interrupt pulse period <sup>(2)</sup>	$t_{LIL}$	Note <sup>(3)</sup>	—	$t_{cyc}$

- $V_{DD} = 4.5$  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.
- Values are based on characterization results, not tested in production.
- The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1  $t_{cyc}$ .

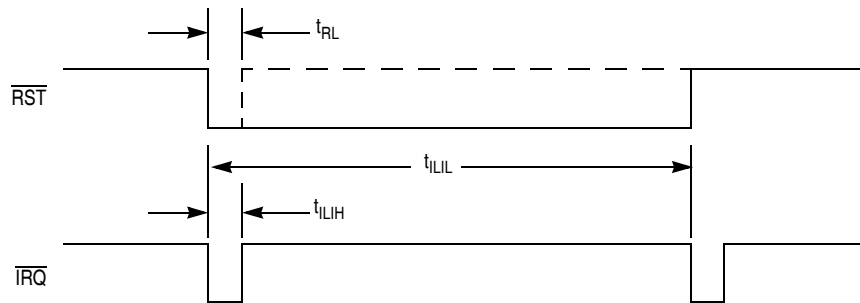


Figure 16-3.  $\overline{RST}$  and  $\overline{IRQ}$  Timing

## 16.8 3-V DC Electrical Characteristics

Characteristic <sup>(1)</sup>	Symbol	Min	Typ <sup>(2)</sup>	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	$V_{IH}$	$0.7 \times V_{DD}$	—	$V_{DD}$	V

— Continued on next page

## Electrical Specifications

Characteristic <sup>(1)</sup>	Symbol	Min	Typ <sup>(2)</sup>	Max	Unit
Input low voltage PTA0–PTA5, PTB0–PTB7	$V_{IL}$	$V_{SS}$	—	$0.3 \times V_{DD}$	V
Input hysteresis <sup>(3)</sup>	$V_{HYS}$	$0.06 \times V_{DD}$	—	—	V
DC injection current <sup>(3) (4) (5) (6)</sup> Single pin limit $V_{in} > V_{DD}$ $V_{in} < V_{SS}$ Total MCU limit, includes sum of all stressed pins $V_{in} > V_{DD}$ $V_{in} < V_{SS}$	$I_{IC}$	0 0	— —	2 –0.2	mA
Ports Hi-Z leakage current	$I_{IL}$	0	—	$\pm 1$	$\mu A$
Capacitance Ports (as input) <sup>(3)</sup>	$C_{IN}$	—	—	8	pF
POR rearm voltage	$V_{POR}$	750	—	—	mV
POR rise time ramp rate <sup>(3)(7)</sup>	$R_{POR}$	0.035	—	—	V/ms
Monitor mode entry voltage <sup>(3)</sup>	$V_{TST}$	$V_{DD} + 2.5$	—	$V_{DD} + 4.0$	V
Pullup resistors <sup>(8)</sup> PTA0–PTA5, PTB0–PTB7	$R_{PU}$	16	26	36	k $\Omega$
Pulldown resistors <sup>(9)</sup> PTA0–PTA5	$R_{PD}$	16	26	36	k $\Omega$
Low-voltage inhibit reset, trip falling voltage	$V_{TRIPF}$	1.80	1.95	2.10	V
Low-voltage inhibit reset, trip rising voltage <sup>(6)</sup>	$V_{TRIPR}$	1.90	2.05	2.20	V
Low-voltage inhibit reset/recover hysteresis	$V_{HYS}$	—	100	—	mV

- $V_{DD} = 2.7$  to  $3.3$  Vdc,  $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$ , unless otherwise noted.
- Typical values reflect average measurements at midpoint of voltage range,  $25^\circ C$  only.
- Values are based on characterization results, not tested in production.
- All functional non-supply pins are internally clamped to  $V_{SS}$  and  $V_{DD}$ .
- Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.
- Power supply must maintain regulation within operating  $V_{DD}$  range during instantaneous and operating maximum current conditions. If positive injection current ( $V_{in} > V_{DD}$ ) is greater than  $I_{DD}$ , the injection current may flow out of  $V_{DD}$  and could result in external power supply going out of regulation. Ensure external  $V_{DD}$  load will shunt current greater than maximum injection current. This will be the greatest risk when the MCU is not consuming power. Examples are: if no system clock is present, or if clock rate is very low (which would reduce overall power consumption).
- 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.
- $R_{PU}$  is measured at  $V_{DD} = 3.0$  V
- $R_{PD}$  is measured at  $V_{DD} = 3.0$  V, Pulldown resistors only available when KBIX is enabled with KBIXPOL = 1.



### 16.9 Typical 3-V Output Drive Characteristics

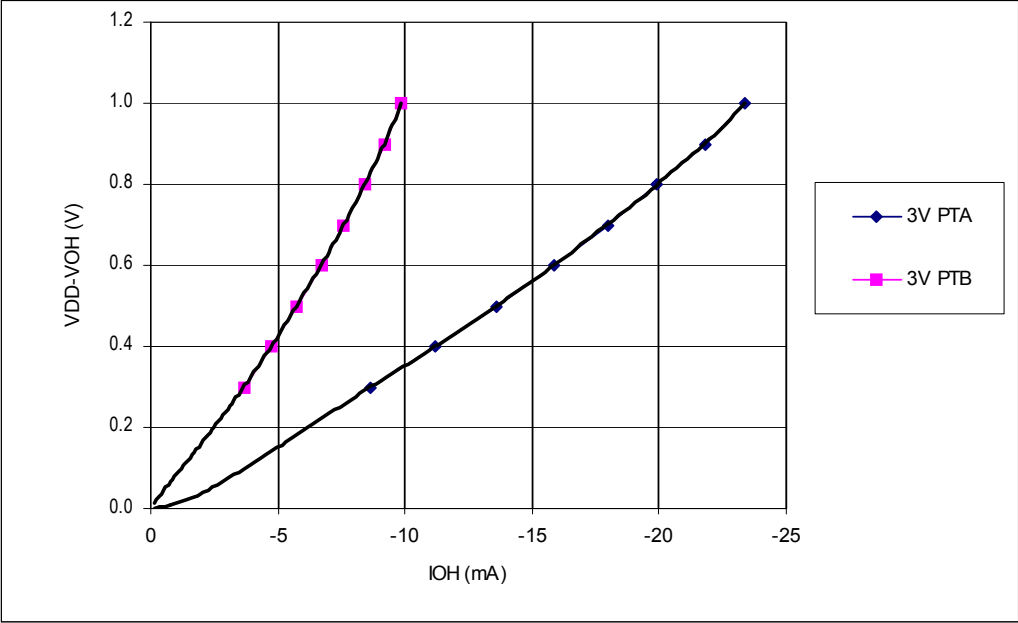


Figure 16-4. Typical 3-Volt Output High Voltage versus Output High Current (25°C)

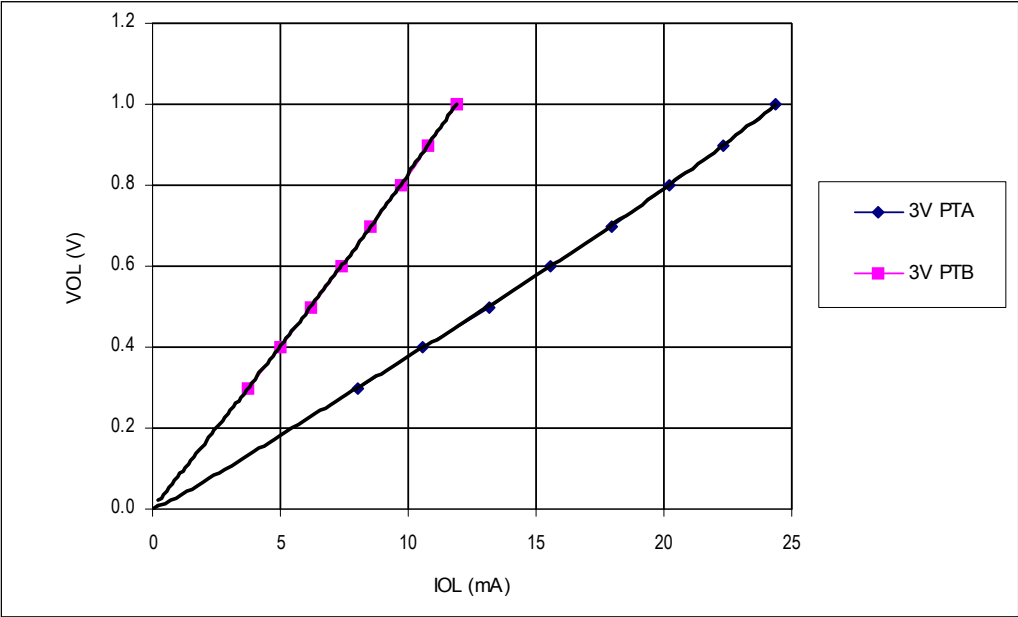


Figure 16-5. Typical 3-Volt Output Low Voltage versus Output Low Current (25°C)

### 16.10 3-V Control Timing

Characteristic <sup>(1)</sup>	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 <sup>(2)</sup>	$t_{RL}$	200	—	ns
$\overline{IRQ}$ interrupt pulse width low (edge-triggered) <sup>(2)</sup>	$t_{LIH}$	200	—	ns
$\overline{IRQ}$ interrupt pulse period <sup>(2)</sup>	$t_{LIL}$	Note <sup>(3)</sup>	—	$t_{cyc}$

- $V_{DD} = 2.7$  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.
- Values are based on characterization results, not tested in production.
- The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1  $t_{cyc}$ .

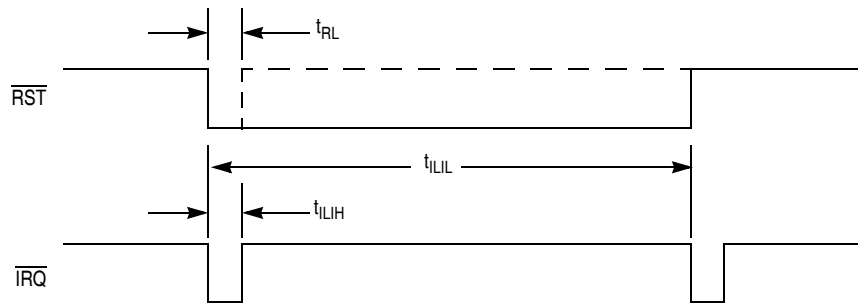


Figure 16-6.  $\overline{RST}$  and  $\overline{IRQ}$  Timing

## 16.11 1.8-V to 3.6-V DC Electrical Characteristics

Characteristic <sup>(1)</sup>	Symbol	Min	Typ <sup>(2)</sup>	Max	Unit
Output high voltage I <sub>Load</sub> = -0.6 mA, all I/O pins I <sub>Load</sub> = -2.0 mA, all I/O pins I <sub>Load</sub> = -5.0 mA, PTA0, PTA1, PTA3-PTA5 only	V <sub>OH</sub>	V <sub>DD</sub> -0.3 V <sub>DD</sub> -0.6 V <sub>DD</sub> -0.6	— — —	— — —	V
Maximum combined I <sub>OH</sub> (all I/O pins)	I <sub>OHT</sub>	—	—	50	mA
Output low voltage I <sub>Load</sub> = 0.5 mA, all I/O pins I <sub>Load</sub> = 3.0 mA, all I/O pins I <sub>Load</sub> = 6.0 mA, PTA0, PTA1, PTA3-PTA5 only	V <sub>OL</sub>	— — —	— — —	0.3 0.6 0.6	V
Maximum combined I <sub>OL</sub> (all I/O pins)	I <sub>OHL</sub>	—	—	50	mA
Input high voltage PTA0-PTA5, PTB0-PTB7	V <sub>IH</sub>	0.7 x V <sub>DD</sub>	—	V <sub>DD</sub>	V
Input low voltage PTA0-PTA5, PTB0-PTB7	V <sub>IL</sub>	V <sub>SS</sub>	—	0.3 x V <sub>DD</sub>	V
Input hysteresis <sup>(3)</sup>	V <sub>HYS</sub>	0.06 x V <sub>DD</sub>	—	—	V
DC injection current, all ports <sup>(4)</sup>	I <sub>INJ</sub>	-2	—	+2	mA
Total dc current injection (sum of all I/O) <sup>(4)</sup>	I <sub>INJTOT</sub>	-25	—	+25	mA
Ports Hi-Z leakage current	I <sub>IL</sub>	-1	±0.1	+1	µA
Capacitance Ports (as input) <sup>(3)</sup>	C <sub>IN</sub>	—	—	8	pF
POR rearm voltage	V <sub>POR</sub>	750	—	—	mV
POR rise time ramp rate <sup>(3)(5)</sup>	R <sub>POR</sub>	0.035	—	—	V/ms
Monitor mode entry voltage <sup>(3)</sup>	V <sub>TST</sub>	V <sub>DD</sub> + 2.5	—	V <sub>DD</sub> + 4.0	V
Pullup resistors <sup>(6)</sup> PTA0-PTA5, PTB0-PTB7	R <sub>PU</sub>	16	26	36	kΩ
Pulldown resistors <sup>(7)</sup> PTA0-PTA5	R <sub>PD</sub>	16	26	36	kΩ
Low-voltage inhibit reset, trip falling voltage	V <sub>TRIPF</sub>	1.80	1.95	2.10	V
Low-voltage inhibit reset, trip rising voltage <sup>(6)</sup>	V <sub>TRIPR</sub>	1.90	2.05	2.20	V
Low-voltage inhibit reset/recover hysteresis	V <sub>HYS</sub>	—	100	—	mV

1. V<sub>SS</sub> = 0 Vdc, T<sub>A</sub> = T<sub>L</sub> to T<sub>H</sub>, unless otherwise noted.

2. Typical values reflect average measurements at midpoint of voltage range, 25°C only.

3. Values are based on characterization results, not tested in production.

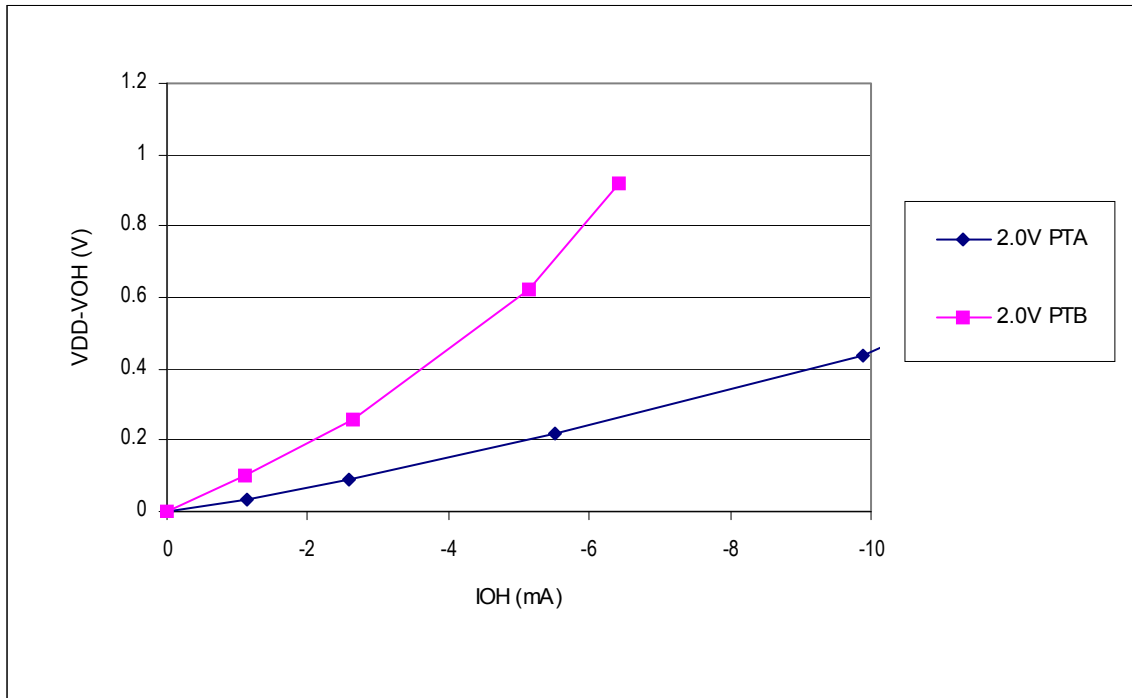
4. Guaranteed by design, not tested in production.

5. If minimum V<sub>DD</sub> is not reached before the internal POR reset is released, the LVI will hold the part in reset until minimum V<sub>DD</sub> is reached.

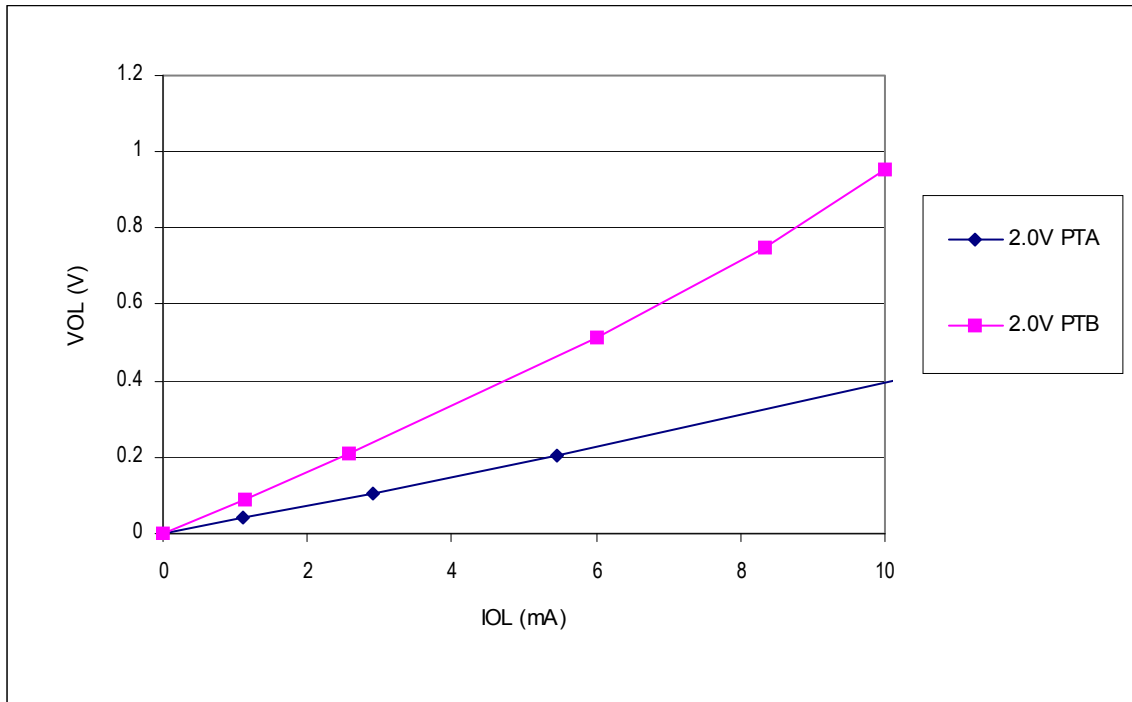
6. R<sub>PU</sub> is measured at V<sub>DD</sub> = 2.0 V

7. R<sub>PD</sub> is measured at V<sub>DD</sub> = 2.0 V, Pulldown resistors only available when KBix is enabled with KBixPOL = 1.

## 16.12 Typical 2-V Output Drive Characteristics



**Figure 16-7. Typical 2-Volt Output High Voltage versus Output High Current (25°C)**



**Figure 16-8. Typical 2-Volt Output Low Voltage versus Output Low Current (25°C)**

## 16.13 1.8-V to 3.6-V Control Timing

Characteristic <sup>(1)</sup>	Symbol	Min	Max	Unit
Internal operating frequency	$f_{OP}$ ( $f_{BUS}$ )	—	2.1	MHz
Internal clock period ( $1/f_{OP}$ )	$t_{cyc}$	475	—	ns
$\overline{RST}$ input pulse width low <sup>(2)</sup>	$t_{RL}$	400	—	ns
$\overline{IRQ}$ interrupt pulse width low (edge-triggered) <sup>(2)</sup>	$t_{LIH}$	400	—	ns
$\overline{IRQ}$ interrupt pulse period <sup>(2)</sup>	$t_{LIL}$	Note <sup>(3)</sup>	—	$t_{cyc}$

- $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$ ; timing shown with respect to 20%  $V_{DD}$  and 70%  $V_{DD}$ , unless otherwise noted.
- Values are based on characterization results, not tested in production.
- The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1  $t_{cyc}$ .

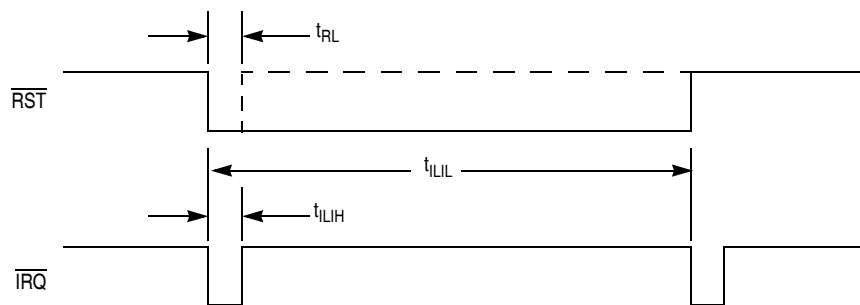


Figure 16-9.  $\overline{RST}$  and  $\overline{IRQ}$  Timing

## 16.14 Oscillator Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
Internal oscillator frequency <sup>(1)</sup> ICFS1:ICFS0 = 00 ICFS1:ICFS0 = 01 ICFS1:ICFS0 = 10 (not allowed if V <sub>DD</sub> < 2.7V)	f <sub>INTCLK</sub>	— — —	4 8 12.8	— — —	MHz
Trim accuracy - fixed voltage and temperature	Δ <sub>TRIM_ACC</sub>	—	± 0.4	—	%
Deviation from trimmed Internal oscillator <sup>(2)(3)</sup> 4, 8, 12.8 MHz, V <sub>DD</sub> = ± 10%, 0 to 70°C 4, 8, 12.8 MHz, V <sub>DD</sub> = ± 10%, -40 to 125°C 4, 8 MHz, V <sub>DD</sub> = 1.8 V to 3.6 V, 0 to 70°C	Δ <sub>INT_TRIM</sub>	— — —	± 2 — —	— ± 5 ± 5	%
External RC oscillator frequency, RCCLK <sup>(1)(2)</sup> V <sub>DD</sub> ≥ 2.7 V V <sub>DD</sub> < 2.7 V	f <sub>RCCLK</sub>	2 2	—	10 8.4	MHz
External clock reference frequency <sup>(1)(4)(5)</sup> V <sub>DD</sub> ≥ 4.5 V V <sub>DD</sub> ≥ 2.7 V V <sub>DD</sub> < 2.7 V	f <sub>OSCCLK</sub>	dc dc dc	—	32 16 8.4	MHz
RC oscillator external resistor V <sub>DD</sub> = 5 V V <sub>DD</sub> = 3 V V <sub>DD</sub> = 2 V	R <sub>EXT</sub>	See <a href="#">Figure 16-10</a> See <a href="#">Figure 16-11</a> See <a href="#">Figure 16-12</a>			—
Crystal frequency, XTALCLK <sup>(1)(6)(7)</sup> ECFS1:ECFS0 = 00 (V <sub>DD</sub> ≥ 4.5 V) ECFS1:ECFS0 = 00 (not allowed if V <sub>DD</sub> < 2.7V) ECFS1:ECFS0 = 01 ECFS1:ECFS0 = 10	f <sub>OSCCLK</sub>	8 8 1 30	—	32 16 8 100	MHz MHz MHz kHz
ECFS1:ECFS0 = 00 <sup>(8)</sup> Feedback bias resistor Crystal load capacitance <sup>(9)</sup> Crystal capacitors <sup>(9)</sup>	R <sub>B</sub> C <sub>L</sub> C <sub>1</sub> , C <sub>2</sub>	— — —	1 20 (2 x C <sub>L</sub> ) - 5pF	— — —	MΩ pF pF
ECFS1:ECFS0 = 01 <sup>(8)</sup> Crystal series damping resistor f <sub>OSCCLK</sub> = 1 MHz f <sub>OSCCLK</sub> = 4 MHz f <sub>OSCCLK</sub> = 8 MHz Feedback bias resistor Crystal load capacitance <sup>(9)</sup> Crystal capacitors <sup>(9)</sup>	R <sub>S</sub> R <sub>B</sub> C <sub>L</sub> C <sub>1</sub> , C <sub>2</sub>	— — — — — —	20 10 0 5 18 (2 x C <sub>L</sub> ) - 10 pF	— — — — — —	kΩ kΩ kΩ MΩ pF pF

Continued on next page

Characteristic	Symbol	Min	Typ	Max	Unit
AWU module, internal RC oscillator frequency	$f_{\text{INTRC}}$	—	32	—	kHz

1. Bus frequency,  $f_{\text{OP}}$ , is oscillator frequency divided by 4.
2. Value is deviation from 25°C and midpoint of voltage range. Factory trimming is done at @25°C and at voltage as specified on customer ROM order form, for example 5.0V for  $5V \pm 10\%$ , 3.0V for 1.8V to 3.6V device.
3. Values are based on characterization results, not tested in production.
4. No more than 10% duty cycle deviation from 50%.
5. When external oscillator clock is greater than 1MHz, ECFS1:ECFS0 must be 00 or 01
6. Use fundamental mode only, do **not** use overtone crystals or overtone ceramic resonators
7. 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.
8. Do not use damping resistor when ECFS1:ECFS0 = 01, 10 or 11
9. Consult crystal vendor data sheet.

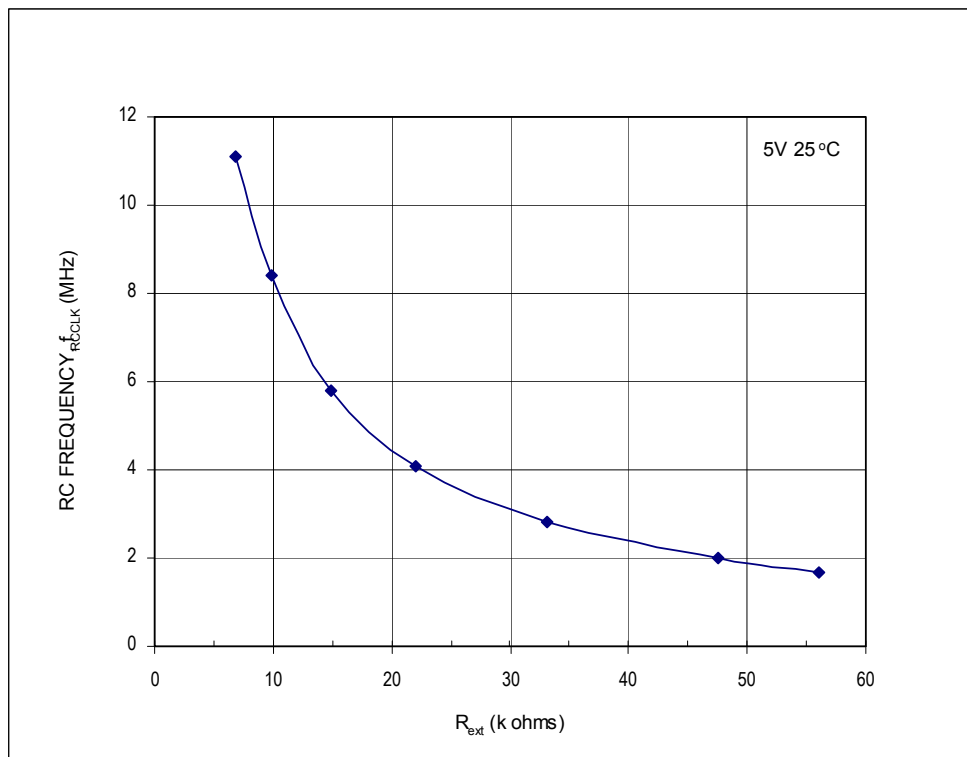


Figure 16-10. RC versus Frequency (5 Volts @ 25°C)

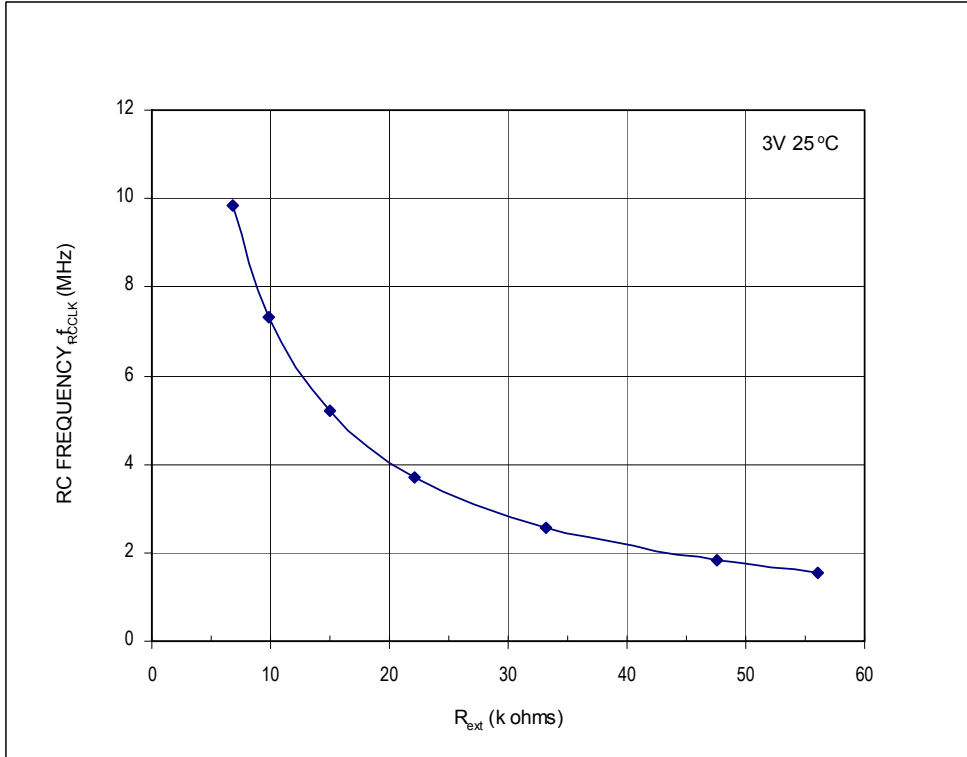


Figure 16-11. RC versus Frequency (3 Volts @ 25°C)

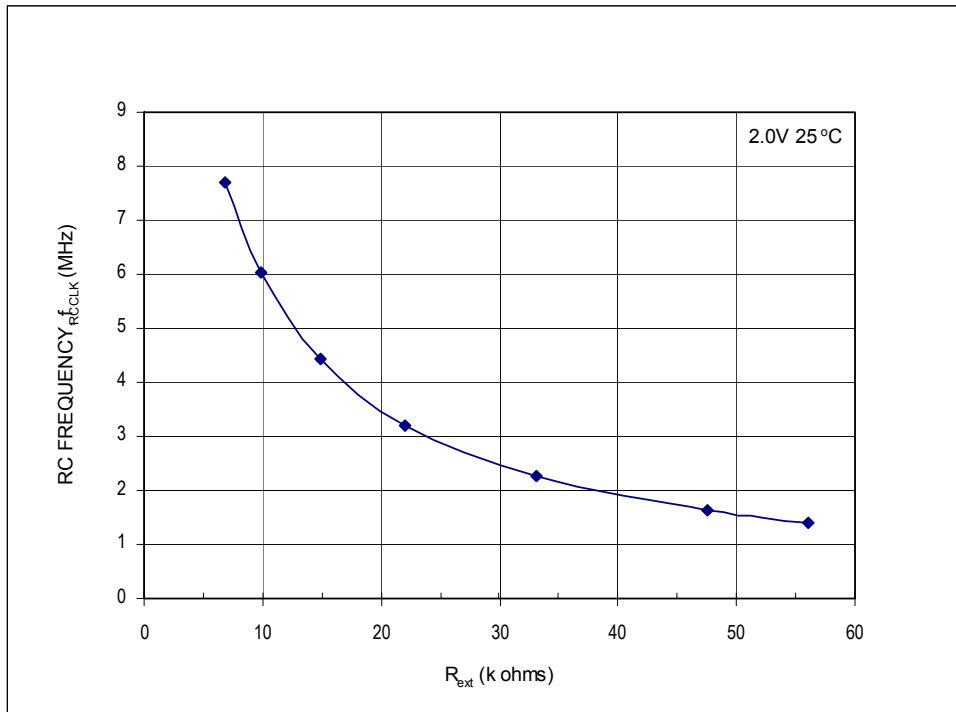


Figure 16-12. RC versus Frequency (2 Volts @ 25°C)



## 16.15 Supply Current Characteristics

Characteristic <sup>(1)</sup>	Voltage	Bus Frequency (MHz)	Symbol	Typ <sup>(2)</sup>	Max	Unit
Run mode $V_{DD}$ supply current <sup>(3)</sup>	5.0	3.2	$RI_{DD}$	3.2	4.2	mA
	3.0	3.2		1.8	2.5	
	2.0	1.0		0.5	1.0	
Wait mode $V_{DD}$ supply current <sup>(4)</sup>	5.0	3.2	$WI_{DD}$	1.0	1.5	mA
	3.0	3.2		0.67	1.0	
	2.0	1.0		0.40	0.8	
Stop mode $V_{DD}$ supply current <sup>(5)</sup> –40 to 85°C –40 to 125°C 25°C with auto wake-up enabled Incremental current with LVI enabled at 25°C	5.0		$SI_{DD}$	0.26	1.0	$\mu$ A
				—	5.0	
				12	—	
Stop mode $V_{DD}$ supply current <sup>(4)</sup> –40 to 85°C –40 to 125°C 25°C with auto wake-up enabled Incremental current with LVI enabled at 25°C	3.0		$SI_{DD}$	0.23	0.5	$\mu$ A
				—	4.0	
				2	—	
Stop mode $V_{DD}$ supply current <sup>(4)</sup> –40 to 85°C –40 to 125°C 25°C with auto wake-up enabled Incremental current with LVI enabled at 25°C	2.0		$SI_{DD}$	0.21	0.4	$\mu$ A
				—	2.0	
				1	—	
Stop mode $V_{DD}$ supply current <sup>(4)</sup> –40 to 85°C –40 to 125°C 25°C with auto wake-up enabled Incremental current with LVI enabled at 25°C	2.0		$SI_{DD}$	0.21	0.4	$\mu$ A
				—	2.0	
				1	—	

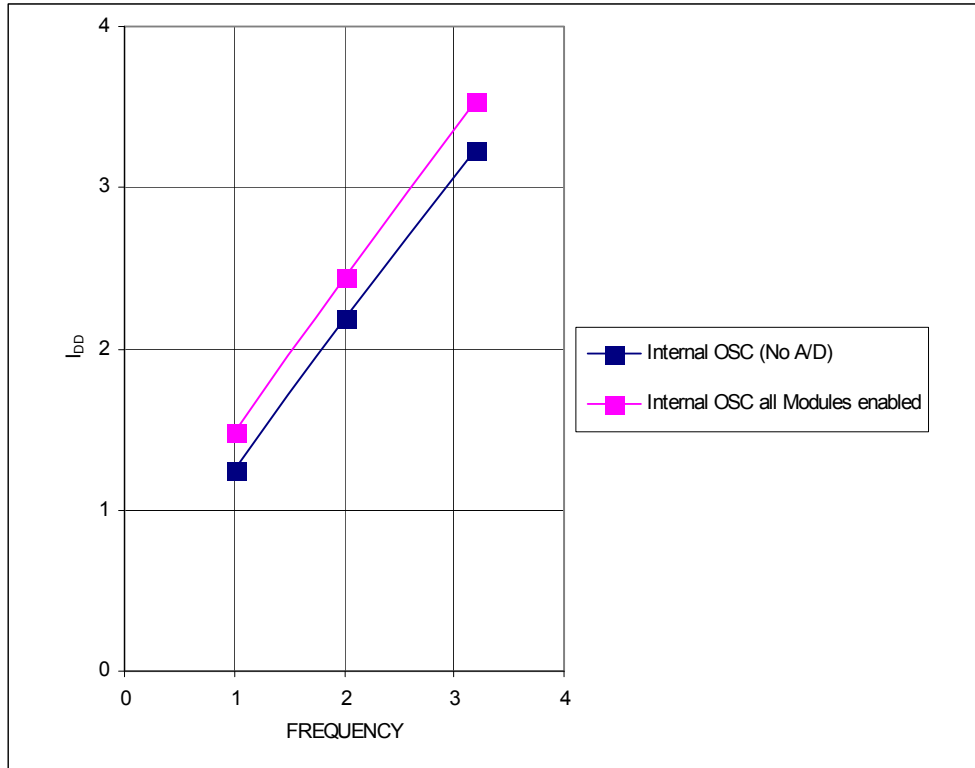
1.  $V_{SS} = 0$  Vdc,  $T_A = T_L$  to  $T_H$ , unless otherwise noted.

2. Typical values reflect average measurement at 25°C only.

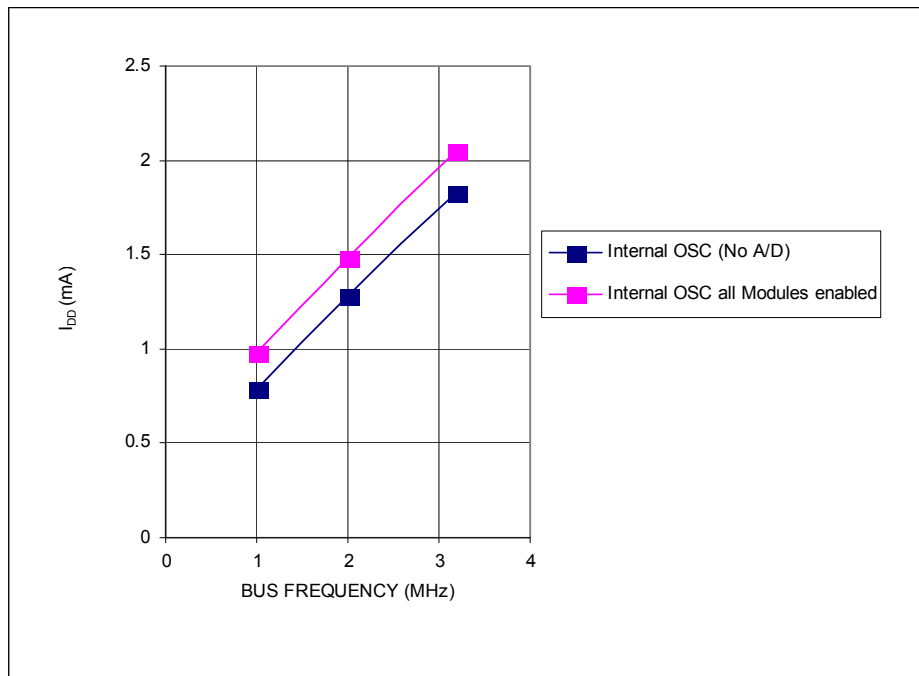
3. Run (operating)  $I_{DD}$  measured using trimmed internal oscillator, ADC off, all modules enabled. All pins configured as inputs and tied to 0.2 V from rail.

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.

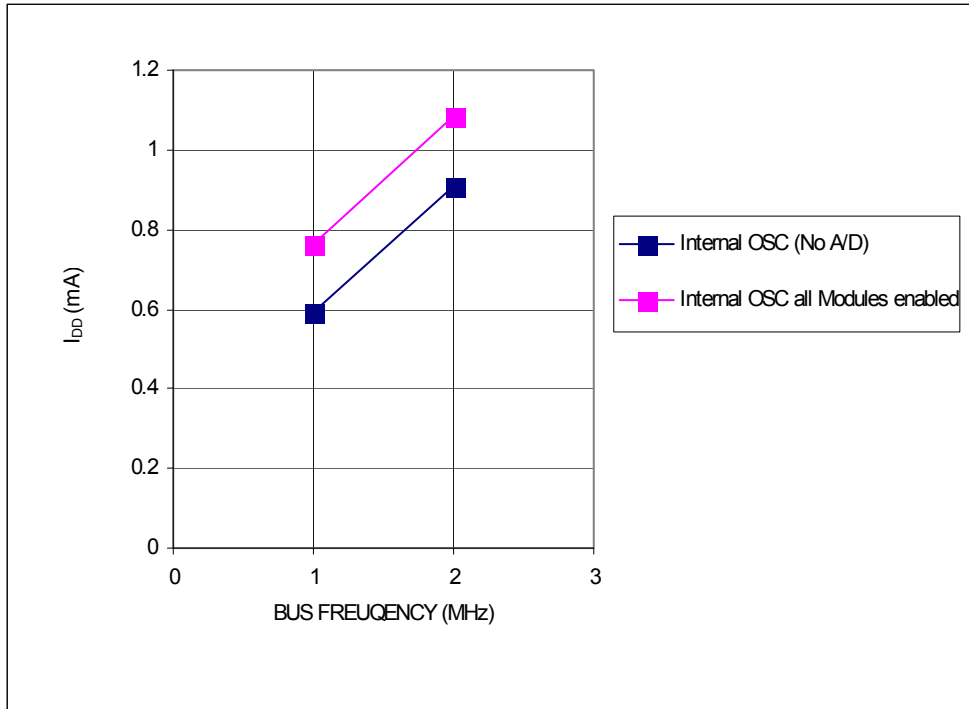
5. Stop  $I_{DD}$  measured with all pins configured as inputs and tied to 0.2 V from rail. On the 8-pin versions, port B is configured as inputs with pullups enabled.



**Figure 16-13. Typical 5-Volt Run Current versus Bus Frequency (25°C)**



**Figure 16-14. Typical 3-Volt Run Current versus Bus Frequency (25°C)**



**Figure 16-15. Typical 2-Volt Run Current versus Bus Frequency (25°C)**

## 16.16 ADC10 Characteristics

Characteristic	Conditions	Symbol	Min	Typ <sup>(1)</sup>	Max	Unit	Comment
Supply voltage	Absolute	$V_{DD}$	1.8	—	5.5	V	
Supply Current ADLPC = 1 ADLSMP = 1 ADCO = 1	$V_{DD} = 2.2$ V	$I_{DD}^{(2)}$	—	50	75	$\mu$ A	
	$V_{DD} \leq 3.6$ V (3.3 V Typ)		—	55	—		
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	75	—		
Supply current ADLPC = 1 ADLSMP = 0 ADCO = 1	$V_{DD} = 2.2$ V	$I_{DD}^{(2)}$	—	110	—	$\mu$ A	
	$V_{DD} \leq 3.6$ V (3.3 V Typ)		—	120	—		
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	175	—		
Supply current ADLPC = 0 ADLSMP = 1 ADCO = 1	$V_{DD} = 2.2$ V	$I_{DD}^{(2)}$	—	135	—	$\mu$ A	
	$V_{DD} \leq 3.6$ V (3.3 V Typ)		—	140	—		
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	180	—		
Supply current ADLPC = 0 ADLSMP = 0 ADCO = 1	$V_{DD} = 2.2$ V	$I_{DD}^{(2)}$	—	320	—	$\mu$ A	
	$V_{DD} \leq 3.6$ V (3.3 V Typ)		—	340	—		
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	440	615		
ADC internal clock	High speed (ADLPC = 0)	$f_{ADCK}$	0.40 <sup>(3)</sup>	—	2.00	MHz	$t_{ADCK} = 1/f_{ADCK}$
	Low power (ADLPC = 1)		0.40 <sup>(3)</sup>	—	1.00		
Conversion time <sup>(4)</sup> 10-bit Mode	Short sample (ADLSMP = 0)	$t_{ADC}$	19	19	20	$t_{ADCK}$ cycles	
	Long sample (ADLSMP = 1)		39	39	40		
Conversion time <sup>(4)</sup> 8-bit Mode	Short sample (ADLSMP = 0)	$t_{ADC}$	16	16	17	$t_{ADCK}$ cycles	
	Long sample (ADLSMP = 1)		36	36	37		
Sample time	Short sample (ADLSMP = 0)	$t_{ADS}$	4	4	4	$t_{ADCK}$ cycles	
	Long sample (ADLSMP = 1)		24	24	24		
Input voltage		$V_{ADIN}$	$V_{SS}$	—	$V_{DD}$	V	
Input capacitance		$C_{ADIN}$	—	7	10	pF	Not tested
Input impedance		$R_{ADIN}$	—	5	15	k $\Omega$	Not tested
Analog source impedance		$R_{AS}$	—	—	10	k $\Omega$	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	$\pm 1.5$	$\pm 2.5$	LSB	Includes quantization
	8-bit mode		0	$\pm 0.7$	$\pm 1.0$		

Continued on next page

Characteristic	Conditions	Symbol	Min	Typ <sup>(1)</sup>	Max	Unit	Comment
Differential non-linearity	10-bit mode	DNL	0	± 0.5	—	LSB	
	8-bit mode		0	± 0.3	—		
Monotonicity and no-missing-codes guaranteed							
Integral non-linearity	10-bit mode	INL	0	± 0.5	—	LSB	
	8-bit mode		0	± 0.3	—		
Zero-scale error	10-bit mode	E <sub>ZS</sub>	0	± 0.5	—	LSB	V <sub>ADIN</sub> = V <sub>SS</sub>
	8-bit mode		0	± 0.3	—		
Full-scale error	10-bit mode	E <sub>FS</sub>	0	± 0.5	—	LSB	V <sub>ADIN</sub> = V <sub>DD</sub>
	8-bit mode		0	± 0.3	—		
Quantization error	10-bit mode	E <sub>Q</sub>	—	—	± 0.5	LSB	8-bit mode is not truncated
	8-bit mode		—	—	± 0.5		
Input leakage error	10-bit mode	E <sub>IL</sub>	0	± 0.2	± 5	LSB	Pad leakage <sup>(5)</sup> * R <sub>AS</sub>
	8-bit mode		0	± 0.1	± 1.2		
Bandgap voltage input <sup>(6)</sup>		V <sub>BG</sub>	1.17	1.245	1.32	V	

1. Typical values assume V<sub>DD</sub> = 5.0 V, temperature = 25°C, f<sub>ADCK</sub> = 1.0 MHz unless otherwise stated. Typical values are for reference only and are not tested in production.
2. Incremental I<sub>DD</sub> 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.

### 16.17 Timer Interface Module Characteristics

Characteristic	Symbol	Min	Max	Unit
Timer input capture pulse width <sup>(1)</sup>	$t_{TH}, t_{TL}$	2	—	$t_{cyc}$
Timer input capture period	$t_{TLTL}$	Note <sup>(2)</sup>	—	$t_{cyc}$
Timer input clock pulse width <sup>(1)</sup>	$t_{TCL}, t_{TCH}$	$t_{cyc} + 5$	—	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  $t_{cyc}$ .

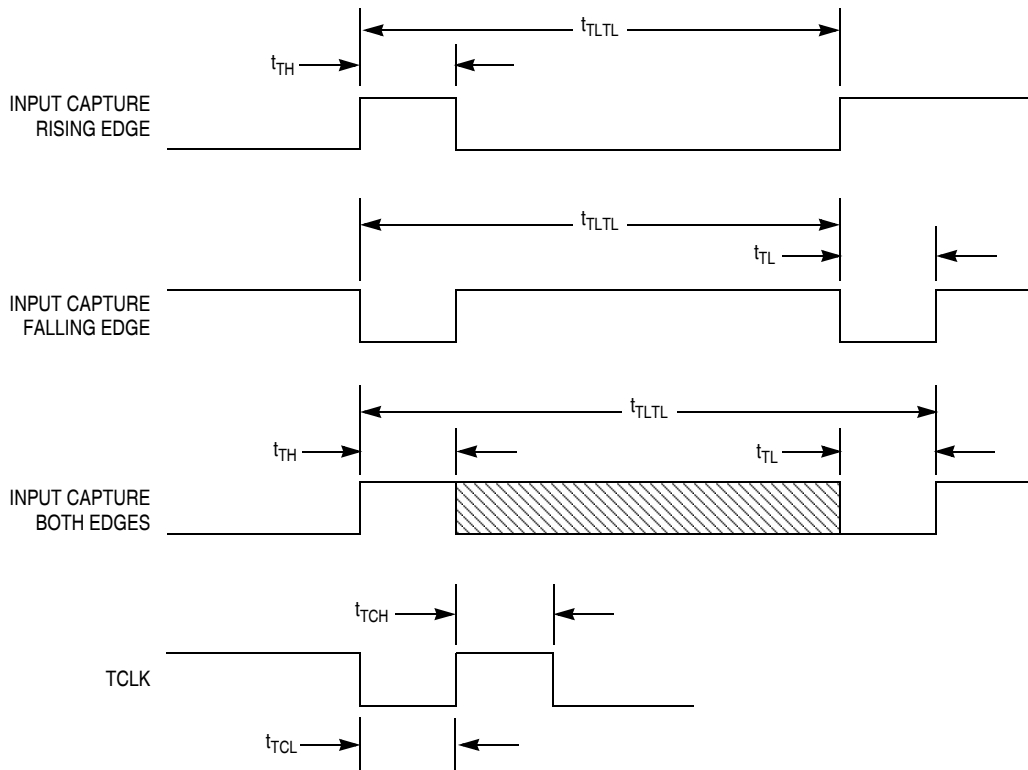


Figure 16-16. Timer Input Timing

### 16.18 Memory Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
RAM data retention voltage <sup>(1)</sup>	$V_{RDR}$	1.3	—	—	V

1. Values are based on characterization results, not tested in production.

# Chapter 17

## Ordering Information and Mechanical Specifications

### 17.1 Introduction

This section contains ordering numbers for MC68HC08QY1, MC68HC08QY2, MC68HC08QY4, MC68HC08QT1, MC68HC08QT2, and MC69HC08QT4. In addition to package dimensions for:

- 8-pin small outline integrated circuit (SOIC) package
- 16-pin SOIC
- 16-pin thin shrink small outline package (TSSOP)

The following figures show the latest package drawings at the time of this publication. To make sure that you have the latest package specifications, contact your local Freescale Semiconductor sales office

### 17.2 MC Order Numbers

These part numbers are generic numbers only. To place an order, ROM code must be submitted to the ROM Processing Center (RPC). Refer to the Customer Interface Tool (CIT) link at:

<http://freescale.com>

**Table 17-1. MC Order Numbers**

MC Order Number	ADC	Memory	Package
MC68HC08QY1	—	1536 bytes	16-pin SOIC and TSSOP
MC68HC08QY2	Yes	1536 bytes	
MC68HC08QY4	Yes	4096 bytes	
MC68HC08QT1	—	1536 bytes	8-pin SOIC
MC68HC08QT2	Yes	1536 bytes	
MC68HC08QT4	Yes	4096 bytes	

Temperature and package designators:

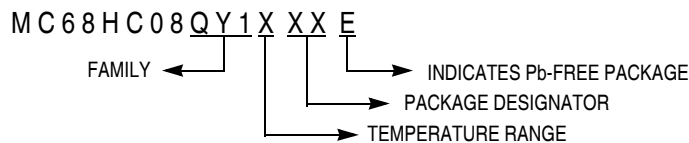
C = -40°C to +85°C

V = -40°C to +105°C

M = -40°C to +125°C

DW = Small outline integrated circuit package (SOIC)

DT = Thin shrink small outline package (TSSOP)



**Figure 17-1. Device Numbering System**

### 17.3 Package Dimensions

Refer to the following pages for detailed package dimensions.







## **How to Reach Us:**

### **Home Page:**

www.freescale.com

### **E-mail:**

support@freescale.com

### **USA/Europe or Locations Not Listed:**

Freescale Semiconductor  
Technical Information Center, CH370  
1300 N. Alma School Road  
Chandler, Arizona 85224  
+1-800-521-6274 or +1-480-768-2130  
support@freescale.com

### **Europe, Middle East, and Africa:**

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
support@freescale.com

### **Japan:**

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064  
Japan  
0120 191014 or +81 3 5437 9125  
support.japan@freescale.com

### **Asia/Pacific:**

Freescale Semiconductor Hong Kong Ltd.  
Technical Information Center  
2 Dai King Street  
Tai Po Industrial Estate  
Tai Po, N.T., Hong Kong  
+800 2666 8080  
support.asia@freescale.com

### **For Literature Requests Only:**

Freescale Semiconductor Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-441-2447 or 303-675-2140  
Fax: 303-675-2150  
LDCForFreescaleSemiconductor@hibbertgroup.com

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

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