MC68HC908QL4

Data Sheet

M68HC08 Microcontrollers

MC68HC908QL4
Rev. 8
04/2010

freescale.com

freescale™
semiconductor

MC68HC908QL4

Data Sheet

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

http://www.freescale.com

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

Revision History (Sheet 1 of 2)

Date	Revision Level	Description	Page Number(s)
September, 2003	N/A	Initial release	N/A
November, 2003	1.0	17.3 Functional Operating Range — Corrected operating voltage range	226
		17.6 Control Timing — Corrected values for internal operating frequency and internal clock period	228
		17.14 5.0-Volt ADC Characteristics — Replaced ADC characteristic table found in initial release	236
		17.15 3.3-Volt ADC Characteristics — Added	237
March, 2004 Continued on next page	2.0	Figure 2-2. Control, Status, and Data Registers Corrected reset state for the FLASH Block Protect Register. Corrected reset value for the Internal Oscillator Time Value	 33 34
		Table 7-1. Instruction Set Summary — Added WAIT instruction	83
		11.8.1 Oscillator Status and Control Register — Revised description of ECGON bit for clarity	117
March, 2004 Continued	2.0	Table 13-3. Interrupt Sources — Corrected address locations for SLIC, KBI, and ADC	140
		14.3.5 SLIC Wait (Core Specific) — Revised description for clarity	144
		14.3.7 SLIC Stop (Core Specific) — Revised description for clarity	144
		14.6.6.2 Byte Transfer Mode Operation — Revised definition of Receiver Buffer Overrun Error	156
		14.7.1 LIN Message Frame Header — Revised third paragraph of description	161
		14.14 Sleep and Wakeup Operation — Revised second paragraph of description	174
		15.8 Input/Output Signals — Corrected reference from PTA0/TCH) to PTB0/TCH0	196
		15.8.2 TIM Channel I/O Pins (PTB0/TCH0 and PTA1/TCH1) — Corrected reference to from PTA0/TCH) to PTB0/TCH0	196
		Figure 16-1. Block Diagram Highlighting BRK and MON Blocks — Added	206
		17.5 5-V DC Electrical Characteristics — Updated table notes	227
		17.8 5-V Oscillator Characteristics — Updated table notes	230
		17.9 3.3-V DC Electrical Characteristics — Updated table notes	231
June, 2004	3.0	Modular sections reworked for clarity.	Throughout

Revision History (Sheet 2 of 2)

Date	Revision Level	Description	Page Number(s)
December, 2004	4.0	Updated to final Freescale format. Corrections per email review. Replaced ADC chapter with latest.	Throughout
		Table 1-3. Function Priority in Shared Pins — Updated entry for PTA2	23
		Figure 1-2. MCU Pin Assignments — Corrected pin assignments for the TSSOP packages.	21
		17.11 Oscillator Characteristics — Updated deviation from trimmed internal oscillator specifications.	210
		17.8 3.3-V DC Electrical Characteristics — Corrected capacitance values	208
		17.11 Oscillator Characteristics — Corrected Note 8	211
August, 2005	5.0	13.4.1 External Pin Reset — Corrected location of RSTEN bit from CONFIG1 to CONFIG2.	120
		13.4.2 Active Resets from Internal Sources — Corrected location of RSTEN bit from CONFIG1 to CONFIG2.	120
		Chapter 18 Ordering Information and Mechanical Specifications — Replaced case outlines with appropriate 98A drawing.	219
September, 2005	6.0	Figure 2-1. Memory Map — Corrected address labels.	26
October, 2006	7.0	Removed all references to the MC68HC908QL3 and MC68HC908QL2.	Throughout
		1.6 Unused Pin Termination — Added new section.	24
		Chapter 4 Auto Wakeup Module (AWU) — Updated entire section for clarity.	57
		12.2 Unused Pin Termination — Added new section.	111
		The following sections were updated to show that a break interrupt inhibits input captures: 15.6 TIM During Break Interrupts 16.2.1.2 TIM During Break Interrupts	 179 189
		Updated DC injection current specification in the following subsections: 17.5 5-V DC Electrical Characteristics 17.8 3.3-V DC Electrical Characteristics	 204 207
April, 2010	8.0	Clarify internal oscillator trim register information.	33, 41, 104, 109

List of Chapters

Table of Contents

Chapter 1
General Description

Chapter 2
Memory

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

Chapter 4
Auto Wakeup Module (AWU)

Chapter 5
Configuration Register (CONFIG)

Chapter 6
Computer Operating Properly (COP)

Chapter 7
Central Processor Unit (CPU)

Chapter 8
External Interrupt (IRQ)

Chapter 9
Keyboard Interrupt Module (KBI)

Chapter 10
Low-Voltage Inhibit (LVI)

Chapter 11
Oscillator Module (OSC)

Chapter 12
Input/Output Ports (PORTS)

Chapter 13
System Integration Module (SIM)

Chapter 14
Slave LIN Interface Controller (SLIC) Module

Chapter 15
Timer Interface Module (TIM)

Chapter 16
Development Support

Chapter 17
Electrical Specifications

Chapter 18
Ordering Information and Mechanical Specifications

Chapter 1
General Description

1.1 Introduction

The MC68HC908QL4 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.

1.2 Features

Features include:

- High-performance M68HC08 CPU core
- Fully upward-compatible object code with M68HC05 Family
- 5-V and 3.3-V operating voltages (V_{DD})
- 8-MHz internal bus operation at 5 V, 4-MHz at 3.3 V
- Software configurable input clock from either internal or external source
- Trimmable internal oscillator
  - Selectable 1 MHz, 2 MHz, or 3.2MHz or 6.4 MHz internal bus operation
  - 8-bit trim capability
  - Trimmable to approximately 0.4%[1]
  - ± 25% untrimmed
- Software selectable crystal oscillator range, 32–100 kHz, 1–8 MHz, and 8–32 MHz
- Auto wakeup from STOP capability using dedicated internal 32-kHz RC or bus clock source
- On-chip in-application programmable FLASH memory
  - Internal program/erase voltage generation
  - Monitor ROM containing user callable program/erase routines
  - FLASH security[2]
- On-chip random-access memory (RAM)

[1] See 17.11 Oscillator Characteristics for internal oscillator specifications

[2] No security feature is absolutely secure. However, Freescale’s strategy is to make reading or copying the FLASH difficult for unauthorized users.

- Slave LIN interface controller (SLIC) module
  - Full LIN messaging buffering of Identifier and 8 data bytes
  - Automatic baud rate and LIN message frame synchronization:

    No prior programming of bit rate required, 1–20 kbps LIN bus speed operation
    All LIN messages will be received (no message loss due to synchronization process)
    Input clock tolerance as high as ±50%, allowing internal oscillator to remain untrimmed
    Incoming break symbols allowed to be 10 to 20 bit times without message loss
    Supports automatic software trimming of internal oscillator using LIN synchronization data
  - Automatic processing and verification of LIN SYNCH BREAK and SYNCH BYTE
  - Automatic checksum calculation and verification with error reporting
  - Maximum of 2 interrupts per LIN message frame
  - Full LIN error checking and reporting
  - High-speed LIN capability up to 83.33 kbps to 120.00 kbps
  - Switchable UART-like byte transfer mode for processing bytes one at a time without LIN message framing constraints
  - Configurable digital receive filter
- 2-channel, 16-bit timer interface module (TIM) with external clock source input
- 6-channel, 10-bit analog-to-digital converter (ADC) with internal bandgap reference channel (ADC10)
- 6-bit keyboard interrupt with wakeup feature (KBI)
  - Programmable for rising/falling or high/low level detect
  - Software selectable to use internal or external pullup/pulldown device
- External asynchronous interrupt pin with internal pullup ($\overline{\text{IRQ}}$)
- Master asynchronous reset pin with internal pullup ($\overline{\text{RST}}$)
- 13 bidirectional input/output (I/O) lines and one input only:
  - Six shared with keyboard interrupt function
  - Six shared with ADC10
  - Two shared with TIM
  - Two shared with SLIC
  - One shared with reset
  - One input only shared with external interrupt (IRQ)
  - High current sink/source capability
  - Selectable pullups on all ports (pullup/down on port A), selectable on an individual bit basis
  - Three-state ability on all port pins
- Low-voltage inhibit (LVI) module features:
  - Software selectable trip point in CONFIG register

- System protection features:
  - Computer operating properly (COP) watchdog
  - Low-voltage detection with reset
  - Illegal opcode detection with reset
  - Illegal address detection with reset
- Power-on reset
- Memory mapped I/O registers
- Power saving stop and wait modes
- Available packages:
  - 16-pin small outline integrated circuit (SOIC) package
  - 16-pin thin shrink small outline package (TSSOP)

Features of the CPU08 include the following:

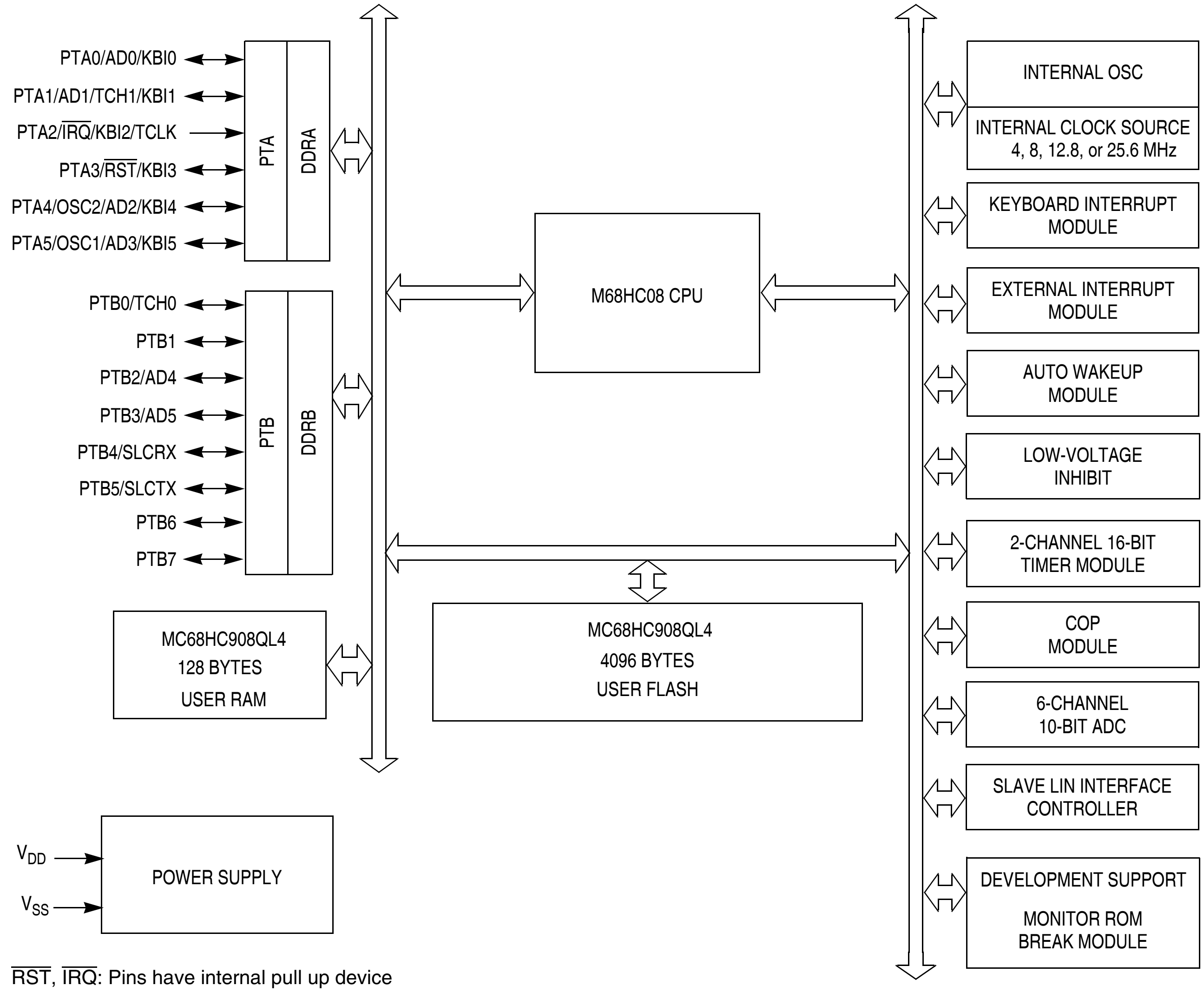
- Enhanced HC05 programming model
- Extensive loop control functions
- 16 addressing modes (eight more than the HC05)
- 16-bit index register and stack pointer
- Memory-to-memory data transfers
- Fast 8 × 8 multiply instruction
- Fast 16/8 divide instruction
- Binary-coded decimal (BCD) instructions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

1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908QL4.

1.4 Pin Functions

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



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device (pullup/down on port A)
PTA[0:5]: Higher current sink and source capability

Figure 1-1. Block Diagram

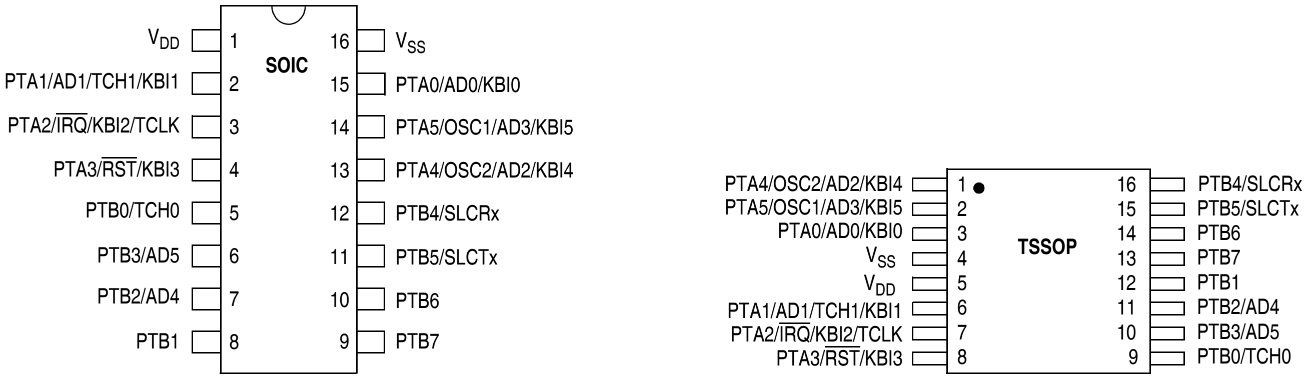


Figure 1-2. MCU Pin Assignments

Table 1-1. Pin Functions

Pin Name	Description	Input/Output
V_{DD}	Power supply	Power
V_{SS}	Power supply ground	Power
PTA0	PTA0 — General purpose I/O port	Input/Output
	AD0 — A/D channel 0 input	Input
	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 — External clock source input for the TIM module	Input
PTA3	PTA3 — General purpose I/O port	Input/Output
	$\overline{\text{RST}}$ — Reset input, active low with internal pullup and Schmitt trigger input	Input
	KBI3 — Keyboard interrupt input 3	Input
PTA4	PTA4 — General purpose I/O port	Input/Output
	OSC2 — XTAL oscillator output (XTAL option only) RC or internal oscillator output (OSC2EN = 1 in PTAPUE register)	Output Output
	AD2 — A/D channel 2 input	Input
	KBI4 — Keyboard interrupt input 4	Input
PTA5	PTA5 — General purpose I/O port	Input/Output
	OSC1 — XTAL, RC, or external oscillator input	Input
	AD3 — A/D channel 3 input	Input
	KBI5 — Keyboard interrupt input 5	Input
PTB0	PTB0 — General purpose I/O port	Input/Output
	TCH0 — Timer Channel 0 I/O	Input/Output
PTB1	PTB1 — General purpose I/O port	Input/Output
PTB2	PTB2 — General purpose I/O port	Input/Output
	AD4 — A/D channel 4 input	Input
PTB3	PTB3 — General purpose I/O port	Input/Output
	AD5 — A/D channel 5 input	Input
PTB4	PTB4 — General purpose I/O port	Input/Output
	SLCRx — SLC receive input	Input
PTB5	PTB5 — General purpose I/O port	Input/Output
	SLCTx — SLC transmit output	Output
PTB6, PTB7	General-purpose I/O port	Input/Output

1.5 Pin Function Priority

Table 1-2 defines the priority of a shared pin if multiple functions are enabled. Only the shared pins are shown in the table.

Table 1-2. Function Priority in Shared Pins

Pin Name	Highest-to-Lowest Priority Sequence
PTA0[1]	AD0 → TCH1 → KBI0 → PTA0
PTA1[1]	AD1 → KBI1 → PTA1
PTA2	TCLK → $\overline{\text{IRQ}}$ → KBI2 → PTA2[2]
PTA3	$\overline{\text{RST}}$ → KBI3 → PTA3
PTA4[1]	OSC2 → AD2 → KBI4 → PTA4
PTA5[1]	OSC1 → AD3 → KBI5 → PTA5
PTB0	TCH0 → PTB0
PTB1	PTB1
PTB2[1]	AD4 → PTB2
PTB3[1]	AD5 → PTB3
PTB4	SLCRx → PTB4
PTB5	SLCTx → PTB5

[1] When a pin is to be used as an ADC pin, the I/O port function should be left as an input. The ADC does not override the port data direction register.

[2] TCLK is not included in the priority scheme. When TCLK is enabled the other shared functions in the pin should be disabled.

1.6 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 MC68HC908QL4. 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.

$0000 ↓ $0051	DIRECT PAGE REGISTERS 81 BYTES
$0052 ↓ $007F	UNIMPLEMENTED 47 BYTES
$0080 ↓ $00FF	RAM 128 BYTES
$0100 ↓ $2B7D	UNIMPLEMENTED 10,878 BYTES
$2B7E ↓ $2E1F	AUXILIARY ROM 674 BYTES
$2E20 ↓ $EDFF	UNIMPLEMENTED 49120 BYTES
$EE00 ↓ $FDFF	FLASH MEMORY 4096 BYTES
$FE00 ↓ $FE0F	MISCELLANEOUS REGISTERS 16 BYTES
$FE10 ↓ $FE1F	UNIMPLEMENTED 16 BYTES
$FE20 ↓ $FF7D	MONITOR ROM 350 BYTES
$FF7E ↓ $FFBD	UNIMPLEMENTED 64 BYTES
$FFBE ↓ $FFC1	MISCELLANEOUS REGISTERS 4 BYTES
$FFC2 ↓ $FFCF	UNIMPLEMENTED 14 BYTES
$FFD0 ↓ $FFFF	USER VECTORS 48 BYTES

Figure 2-1. Memory Map

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read:	0	AWUL	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:	▨	▨				▨		
	See page 112.	Reset:	U	0	U	U	U	U	U	U
$0001	Port B Data Register (PTB)	Read:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Write:								
	See page 114.	Reset:	Unaffected by reset							
$0002 ↓ $0003	Reserved									
$0004	Data Direction Register A (DDRA)	Read:	0	0	DDRA5	DDRA4	DDRA3	0	DDRA1	DDRA0
		Write:	▨	▨				▨		
	See page 112.	Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB)	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Write:								
	See page 115.	Reset:	0	0	0	0	0	0	0	0
$0006 ↓ $000A	Reserved									
$000B	Port A Input Pullup/Down Enable Register (PTAPUE)	Read:	OSC2EN	0	PTAPUE5	PTAPUE4	PTAPUE3	PTAPUE2	PTAPUE1	PTAPUE0
		Write:		▨						
	See page 113.	Reset:	0	0	0	0	0	0	0	0
$000C	Port B Input Pullup Enable Register (PTBPUE)	Read:	PTBPUE7	PTBPUE6	PTBPUE5	PTBPUE4	PTBPUE3	PTBPUE2	PTBPUE1	PTBPUE0
		Write:								
	See page 116.	Reset:	0	0	0	0	0	0	0	0
$000D ↓ $0019	Reserved									
$001A	Keyboard Status and Control Register (KBSCR)	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
		Write:	▨	▨	▨	▨	▨	ACKK		
	See page 94.	Reset:	0	0	0	0	0	0	0	0
$001B	Keyboard Interrupt Enable Register (KBIER)	Read:	0	AWUIE	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:	▨							
	See page 95.	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented   R = Reserved   U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 1 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001C	Keyboard Interrupt Polarity Register (KBIPR) See page 95.	Read:	0	0	KBIP5	KBIP4	KBIP3	KBIP2	KBIP1	KBIP0
		Write:	▨	▨						
		Reset:	0	0	0	0	0	0	0	0
$001D	IRQ Status and Control Register (INTSCR) See page 87.	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:	▨	▨	▨	▨	▨	ACK		
		Reset:	0	0	0	0	0	0	0	0
$001E	Configuration Register 2 (CONFIG2)[1] See page 63.	Read:	IRQPUD	IRQEN	R	R	R	R	OSCENIN-STOP	RSTEN
		Write:								
		Reset:	0	0	0	0	0	0	0	0[2]

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001F	Configuration Register 1 (CONFIG1)[1] See page 64.	Read:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVITRIP	SSREC	STOP	COPD
		Write:								
		Reset:	0	0	0	0	0[2]	0	0	0

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0020	TIM Status and Control Register (TSC) See page 180.	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	▨			
		Reset:	0	0	1	0	0	0	0	0
$0021	TIM Counter Register High (TCNTH) See page 182.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$0022	TIM Counter Register Low (TCNTL) See page 182.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$0023	TIM Counter Modulo Register High (TMODH) See page 182.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0024	TIM Counter Modulo Register Low (TMODL) See page 182.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	TIM Channel 0 Status and Control Register (TSC0) See page 183.	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0026	TIM Channel 0 Register High (TCH0H) See page 185.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0027	TIM Channel 0 Register Low (TCH0L) See page 185.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0028	TIM Channel 1 Status and Control Register (TSC1) See page 183.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		░					
		Reset:	0	0	0	0	0	0	0	0
$0029	TIM Channel 1 Register High (TCH1H) See page 185.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002A	TIM Channel 1 Register Low (TCH1L) See page 185.	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 (OSCSC) See page 108.	Read:	OSCOPT1	OSCOPT0	ICFS1	ICFS0	ECFS1	ECFS0	ECGON	ECGST
		Write:								░
		Reset:	0	0	1	0	0	0	0	0
$0037	Reserved									
$0038	Oscillator Trim Register (OSCTRIM) See page 109.	Read:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
		Write:								
		Reset:	1	0	0	0	0	0	0	0
$0039 ↓ $003B	Reserved									
$003C	ADC10 Status and Control Register (ADSCR) See page 52.	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:	░							
		Reset:	0	0	0	1	1	1	1	1
$003D	ADC10 Data Register High (ADRH) See page 54.	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

░ = Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$003E	ADC10 Data Register Low (ADRL) See page 54.	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$003F	ADC10 Clock Register (ADCLK) See page 55.	Read:	ADLPC	ADIV1	ADIV0	ADICLK	MODE1	MODE0	ADLSMP	ADACKEN
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0040	SLIC Control Register 1 (SLCC1) See page 139.	Read:	0	0	INITREQ	0	WAKETX	TXABRT	IMSG	SLCIE
		Write:	▨	▨		▨				
		Reset:	0	0	1	0	0	0	0	0
$0041	SLIC Control Register 2 (SLCC2) See page 140.	Read:	0	0	0	0	SLCWCM	BTM	0	SLCE
		Write:	▨	▨	▨	▨			▨	
		Reset:	0	0	0	0	0	0	0	0
$0042	SLIC Status Register (SLCS) See page 141.	Read:	SLCACT	0	INITACK	0	0	0	0	SLCF
		Write:	▨	▨	▨	▨	▨	▨	▨	
		Reset:	0	0	1	0	0	0	0	0
$0043	SLIC Prescale Register (SLCP) See page 142.	Read:	RXFP1	RXFP0	0	0	0	0	0	0
		Write:			▨	▨	▨	▨	▨	▨
		Reset:	1	0	0	0	0	0	0	0
$0044	SLIC Bit Time Register High (SLCBTH) See page 143.	Read:	0	0	0	BT12	BT11	BT10	BT9	BT8
		Write:	▨	▨	▨					
		Reset:	0	0	0	0	0	0	0	0
$0045	SLIC Bit Time Register Low (SLCBTL) See page 143.	Read:	BT7	BT6	BT5	BT4	BT3	BT2	BT1	0
		Write:								▨
		Reset:	0	0	0	0	0	0	0	0
$0046	SLIC State Vector Register (SLCSV) See page 144.	Read:	0	0	I3	I2	I1	I0	0	0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$0047	SLIC Data Length Code Register (SLCDLC) See page 148.	Read:	TXGO	CHKMOD	DLC5	DLC4	DLC3	DLC2	DLC1	DLC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0048	SLIC Identifier Register (SLCID) See page 149.	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	0	0	0	0	0	0	0	0
$0049	SLIC Data Register 7 (SLCD7) See page 149.	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented   R = Reserved   U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
	SLIC Data Register 6	Read:	R7	R6	R5	R4	R3	R2	R1	R0
$004A	(SLCD6)	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	See page 149.	Reset:	0	0	0	0	0	0	0	0
	SLIC Data Register 5	Read:	R7	R6	R5	R4	R3	R2	R1	R0
$004B	(SLCD5)	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	See page 149.	Reset:	0	0	0	0	0	0	0	0
	SLIC Data Register 4	Read:	R7	R6	R5	R4	R3	R2	R1	R0
$004C	(SLCD4)	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	See page 149.	Reset:	0	0	0	0	0	0	0	0
	SLIC Data Register 3	Read:	R7	R6	R5	R4	R3	R2	R1	R0
$004D	(SLCD3)	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	See page 149.	Reset:	0	0	0	0	0	0	0	0
	SLIC Data Register 2	Read:	R7	R6	R5	R4	R3	R2	R1	R0
$004E	(SLCD2)	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	See page 149.	Reset:	0	0	0	0	0	0	0	0
	SLIC Data Register 1	Read:	R7	R6	R5	R4	R3	R2	R1	R0
$004F	(SLCD1)	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	See page 149.	Reset:	0	0	0	0	0	0	0	0
	SLIC Data Register 0	Read:	R7	R6	R5	R4	R3	R2	R1	R0
$0050	(SLCD0)	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	See page 149.	Reset:	0	0	0	0	0	0	0	0
$0051 ↓ $005F	Reserved									
	Break Status Register	Read:	R	R	R	R	R	R	SBSW	R
$FE00	(BSR)	Write:							See note 1	
	See page 191.	Reset:							0	
	SIM Reset Status Register	Read:	POR	PIN	COP	ILOP	ILAD	MODRST	LVI	0
$FE01	(SRSR)	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	See page 131.	POR:	1	0	0	0	0	0	0	0
	Break Auxiliary	Read:	0	0	0	0	0	0	0	BDCOP
$FE02	Register (BRKAR)	Write:	▨	▨	▨	▨	▨	▨	▨	
	See page 191.	Reset:	0	0	0	0	0	0	0	0

1. Writing a 0 clears SBSW.

▨ = Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE03	Break Flag Control Register (BFCR) See page 191.	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE04	Interrupt Status Register 1 (INT1) See page 127.	Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE05	Interrupt Status Register 2 (INT2) See page 127.	Read:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE06	Interrupt Status Register 3 (INT3) See page 128.	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	FLASH Control Register (FLCR) See page 35.	Read:	0	0	0	0	HVEN	MASS	ERASE	PGM
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	0	0	0	0
$FE09	Break Address High Register (BRKH) See page 190.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0A	Break Address Low Register (BRKL) See page 190.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0B	Break Status and Control Register (BRKSCR) See page 191.	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:			▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$FE0C	LVI Status Register (LVISR) See page 99.	Read:	LVIOUT	0	0	0	0	0	0	R
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$FE0D ↓ $FE0F	Reserved									
$FFBE	FLASH Block Protect Register (FLBPR) See page 40.	Read:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	0
		Write:								▨
		Reset:	Unaffected by reset							

▨ = Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FFBF	Reserved									
$FFC0	Internal Oscillator Trim (Factory Programmed, V_{DD} = 5.0 V)	Read:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
		Write:								
		Reset:	Resets to factory programmed value							
$FFC1	Reserved									
$FFFF	COP Control Register (COPCTL) See page 69.	Read:	LOW BYTE OF RESET VECTOR							
		Write:	WRITING CLEARS COP COUNTER (ANY VALUE)							
		Reset:	Unaffected by reset							

▨ = Unimplemented R = Reserved U = Unaffected

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

Table 2-1. Vector Addresses

Vector Priority	Vector	Address	Vector
Lowest	IF22–IF16	$FFD0,1–$FFDC,D	Unused vectors
↑	IF15	$FFDE,F	ADC conversion complete vector
	IF14	$FFE0,1	Keyboard vector
	IF13	$FFE2,3	Unused vector
	IF12	$FFE4,5	Unused vector
	IF11	$FFE6,7	Unused vector
	IF10	$FFE8,9	Unused vector
	IF9	$FFEA,B	SLIC vector
	IF8	$FFFC,D	Unused vector
	IF7	$FFEE,F	Unused vector
	IF6	$FFF0,1	Unused vector
	IF5	$FFF2,3	TIM overflow vector
	IF4	$FFF4,5	TIM channel 1 vector
	IF3	$FFF6,7	TIM channel 0 vector
	IF2	$FFF8,9	Unused vector
	IF1	$FFFA,B	$\overline{\text{IRQ}}$ vector
↓	—	$FFFC,D	SWI vector
Highest	—	$FFFE,F	Reset vector

2.5 Random-Access Memory (RAM)

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

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

For compatibility with older M68HC05 MCUs, the HC08 resets the stack pointer to $00FF. In the devices that have RAM above $00FF, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables.

Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM).

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

2.6 FLASH Memory (FLASH)

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

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

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

NOTE

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

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

2.6.1 FLASH Control Register

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

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

▨ = Unimplemented

Figure 2-3. FLASH Control Register (FLCR)

HVEN — High Voltage Enable Bit

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

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

MASS — Mass Erase Control Bit

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

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

ERASE — Erase Control Bit

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

1 = Erase operation selected
0 = Erase operation unselected

PGM — Program Control Bit

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

1 = Program operation selected
0 = Program operation unselected

2.6.2 FLASH Page Erase Operation

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

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

NOTE

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

NOTE

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

CAUTION

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

2.6.3 FLASH Mass Erase Operation

Use the following procedure to erase the entire FLASH memory to read as a 1:

1. Set both the ERASE bit and the MASS bit in the FLASH control register.
2. Read the FLASH block protect register.
3. Write any data to any FLASH address[1] within the FLASH memory address range.
4. Wait for a time, t_{NVS}.
5. Set the HVEN bit.
6. Wait for a time, t_{MErase}.
7. Clear the ERASE and MASS bits.

NOTE

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

8. Wait for a time, t_{NVHL}.
9. Clear the HVEN bit.
10. After time, t_{RCV}, the memory can be accessed in read mode again.

NOTE

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

CAUTION

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

[1] When in monitor mode, with security sequence failed (see 16.3.2 Security), write to the FLASH block protect register instead of any FLASH address.

2.6.4 FLASH Program Operation

Programming of the FLASH memory is done on a row basis. A row consists of 32 consecutive bytes starting from addresses $XX00, $XX20, $XX40, $XX60, $XX80, $XXA0, $XXC0, or $XXE0. Use the following step-by-step procedure to program a row of FLASH memory

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

NOTE

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

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

NOTE

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

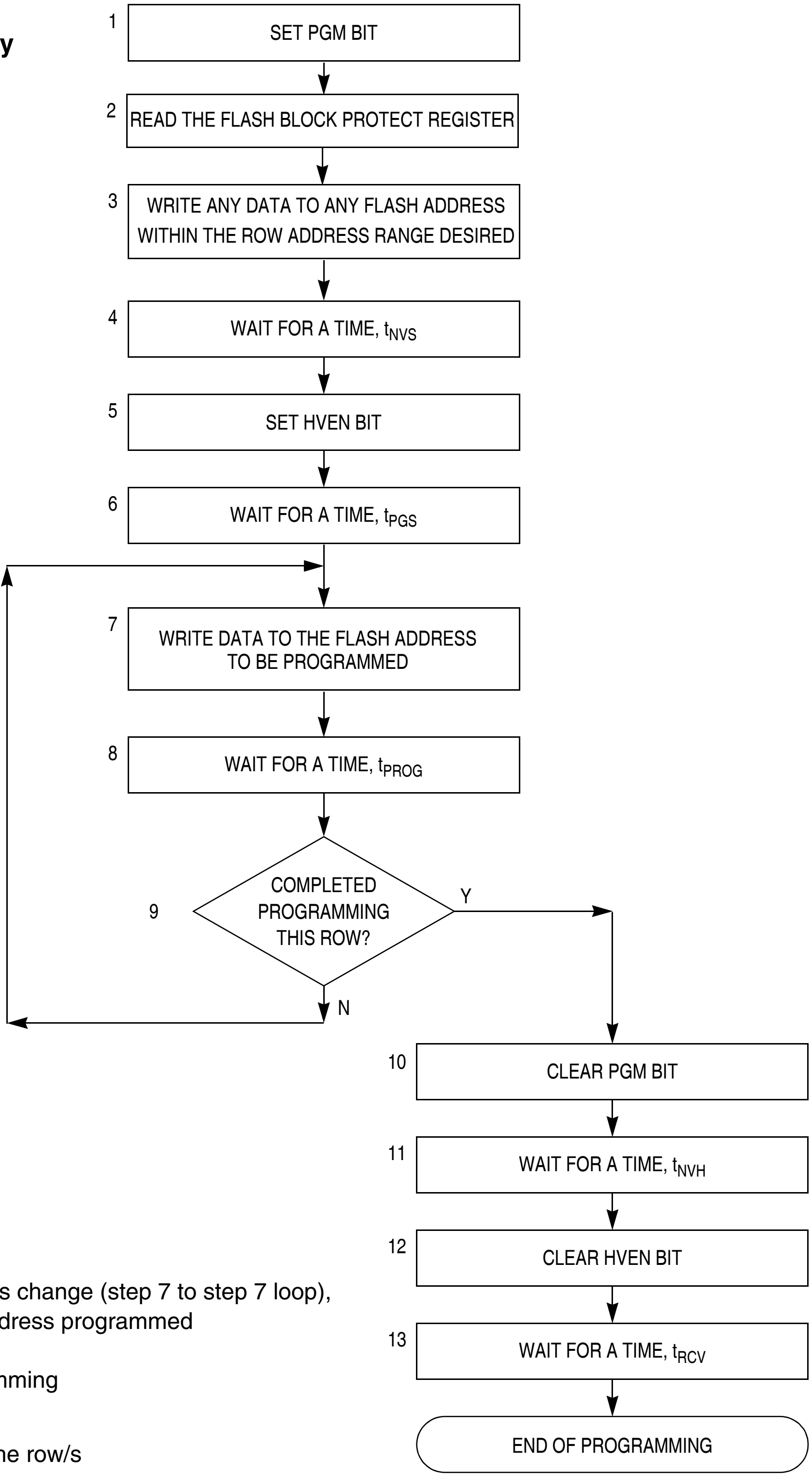
This program sequence is repeated throughout the memory until all data is programmed.

NOTE

Programming and erasing of FLASH locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Do not exceed t_{PROG} maximum, see 17.15 Memory Characteristics.

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

Algorithm for Programming a Row (32 Bytes) of FLASH Memory


1
SET PGM BIT
2
READ THE FLASH BLOCK PROTECT REGISTER
3
WRITE ANY DATA TO ANY FLASH ADDRESS WITHIN THE ROW ADDRESS RANGE DESIRED
4
WAIT FOR A TIME, t_{NVS}
5
SET HVEN BIT
6
WAIT FOR A TIME, t_{PGS}
7
WRITE DATA TO THE FLASH ADDRESS TO BE PROGRAMMED
8
WAIT FOR A TIME, t_{PROG}
9
COMPLETED PROGRAMMING THIS ROW?
Y
N
10
CLEAR PGM BIT
11
WAIT FOR A TIME, t_{NVH}
12
CLEAR HVEN BIT
13
WAIT FOR A TIME, t_{RCV}
END OF PROGRAMMING

NOTES:

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

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

Figure 2-4. FLASH Programming Flowchart

2.6.5 FLASH Protection

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

NOTE

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

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

When bits within the FLBPR are programmed, they lock a block of memory. The address ranges are shown in 2.6.6 FLASH Block Protect Register. Once the FLBPR is programmed with a value other than $FF, any erase or program of the FLBPR or the protected block of FLASH memory is prohibited. Mass erase is disabled whenever any block is protected (FLBPR does not equal $FF). The FLBPR itself can be erased or programmed only with an external voltage, V_{TST}, present on the $\overline{IRQ}$ pin. This voltage also allows entry from reset into monitor mode.

2.6.6 FLASH Block Protect Register

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

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

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

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

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

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

The resultant 16-bit address is used for specifying the start address of the FLASH memory for block protection. The FLASH is protected from this start address to the end of FLASH memory, at $FFFF. With this mechanism, the protect start address can be $XX00, $XX40, $XX80, or $XXC0 within the FLASH memory. See Figure 2-6 and Table 2-2.

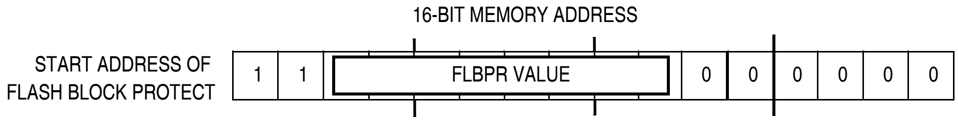


Figure 2-6. FLASH Block Protect Start Address

Table 2-2. Examples of Protect Start Address

BPR[7:0]	**Start of Address of Protect Range[1]**
$00–$B8	The entire FLASH memory is protected.
$B9 (**1011 1001**)	$EE40 (11**10 1110 01**00 0000)
$BA (**1011 1010**)	$EE80 (11**10 1110 10**00 0000)
$BB (**1011 1011**)	$EEC0 (11**10 1110 11**00 0000)
$BC (**1011 1100**)	$EF00 (11**10 1111 00**00 0000)
and so on...	
$DE (**1101 1110**)	$F780 (11**11 0111 10**00 0000)
$DF (**1101 1111**)	$F7C0 (11**11 0111 11**00 0000)
$FE (**1111 1110**)	$FF80 (11**11 1111 10**00 0000) FLBPR, internal oscillator trim value, and vectors are protected
$FF	The entire FLASH memory is not protected.

[1] The end address of the protected range is always $FFFF.

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

3.1 Introduction

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

The ADC10 module shares its pins with general-purpose input/output (I/O) port pins. See Figure 3-1 for port location of these shared pins. The ADC10 on this MCU uses V_{DD} and V_{SS} as its supply and reference pins. This MCU uses BUSCLKX4 as its alternate clock source for the ADC. This MCU does not have a hardware conversion trigger.

3.2 Features

Features of the ADC10 module include:

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

3.3 Functional Description

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

Figure 3-2 shows a block diagram of the ADC10

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

NOTE

Input voltage must not exceed the analog supply voltages.

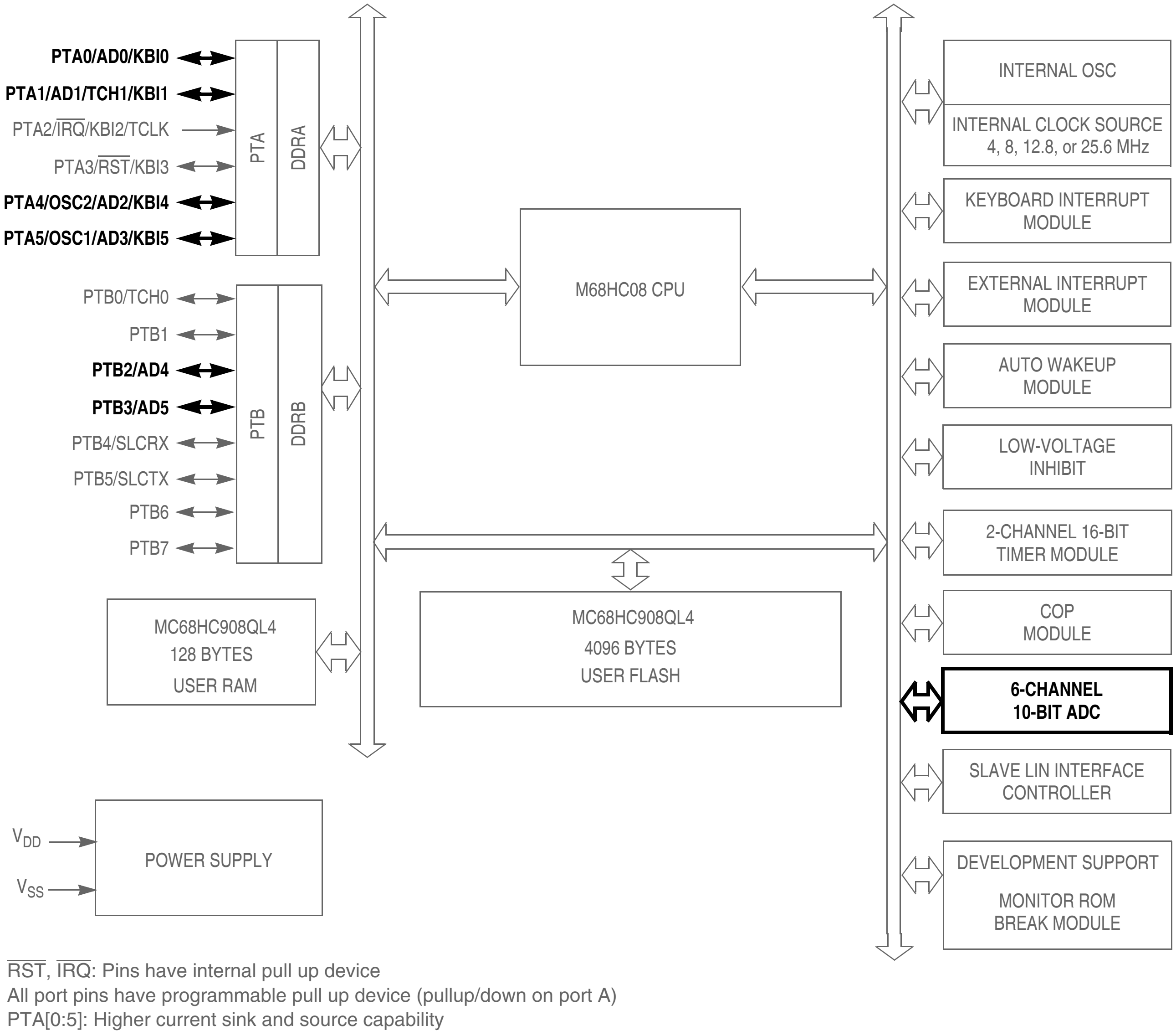


Figure 3-1. Block Diagram Highlighting ADC10 Block and Pins

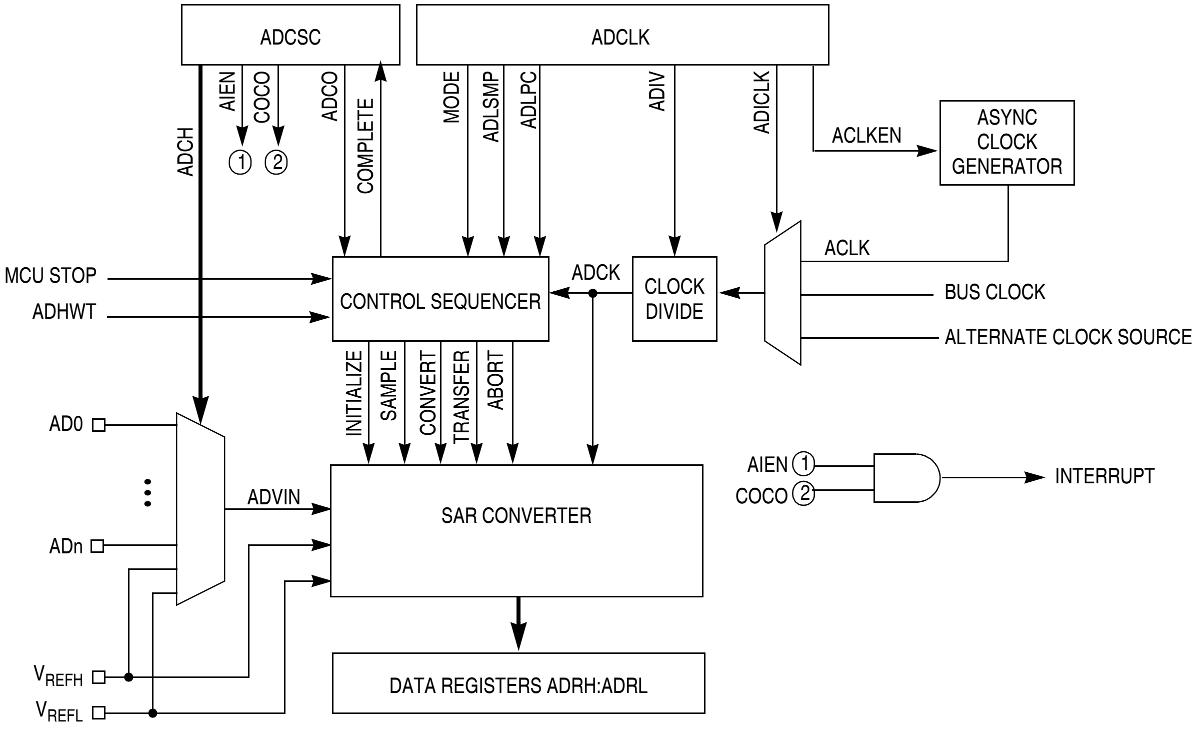


Figure 3-2. ADC10 Block Diagram

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

3.3.1 Clock Select and Divide Circuit

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

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

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

3.3.2 Input Select and Pin Control

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

3.3.3 Conversion Control

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

3.3.3.1 Initiating Conversions

A conversion is initiated:

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

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

3.3.3.2 Completing Conversions

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

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

3.3.3.3 Aborting Conversions

Any conversion in progress will be aborted when:

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

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

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

3.3.3.4 Total Conversion Time

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

Table 3-1. Total Conversion Time versus Control Conditions

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

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

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

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

NOTE

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

3.3.4 Sources of Error

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

3.3.4.1 Sampling Error

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

3.3.4.2 Pin Leakage Error

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

3.3.4.3 Noise-Induced Errors

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

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

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

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

3.3.4.4 Code Width and Quantization Error

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

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

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

3.3.4.5 Linearity Errors

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

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

3.3.4.6 Code Jitter, Non-Monotonicity and Missing Codes

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

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

3.4 Interrupts

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

3.5 Low-Power Modes

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

3.5.1 Wait Mode

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

3.5.2 Stop Mode

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

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

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

3.6 ADC10 During Break Interrupts

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

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

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

3.7 I/O Signals

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

3.7.1 ADC10 Analog Power Pin (V_{DDA})

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

NOTE

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

3.7.2 ADC10 Analog Ground Pin (V_{SSA})

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

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

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

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

NOTE

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

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

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

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

3.7.5 ADC10 Channel Pins (ADn)

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

3.8 Registers

These registers control and monitor operation of the ADC10:

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

3.8.1 ADC10 Status and Control Register

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

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

▨ = Unimplemented

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

COCO — Conversion Complete Bit

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

1 = Conversion completed
0 = Conversion not completed

AIEN — ADC10 Interrupt Enable Bit

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

1 = ADC10 interrupt enabled
0 = ADC10 interrupt disabled

ADCO — ADC10 Continuous Conversion Bit

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

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

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

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

ADCH[4:0] — Channel Select Bits

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

Table 3-2. Input Channel Select

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	Input Select[1]
0	0	0	0	0	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[2]
1	1	0	1	1	Reserved
1	1	1	0	0	Reserved
1	1	1	0	1	V_{REFH}
1	1	1	1	0	V_{REFL}
1	1	1	1	1	Low-power state

[1] If any unused or reserved channels are selected, the resulting conversion will be unknown.
[2] Requires LVI to be powered (LVIPWRD = 0, in CONFIG1)

3.8.2 ADC10 Result High Register (ADRH)

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

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

▨ = Unimplemented

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

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

▨ = Unimplemented

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

3.8.3 ADC10 Result Low Register (ADRL)

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

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

▨ = Unimplemented

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

3.8.4 ADC10 Clock Register (ADCLK)

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

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

Figure 3-7. ADC10 Clock Register (ADCLK)

ADLPC — ADC10 Low-Power Configuration Bit

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

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

ADIV[1:0] — ADC10 Clock Divider Bits

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

Table 3-3. ADC10 Clock Divide Ratio

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

ADICLK — Input Clock Select Bit

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

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

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

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

Reset returns 8-bit mode.

00 = 8-bit, right-justified, 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.

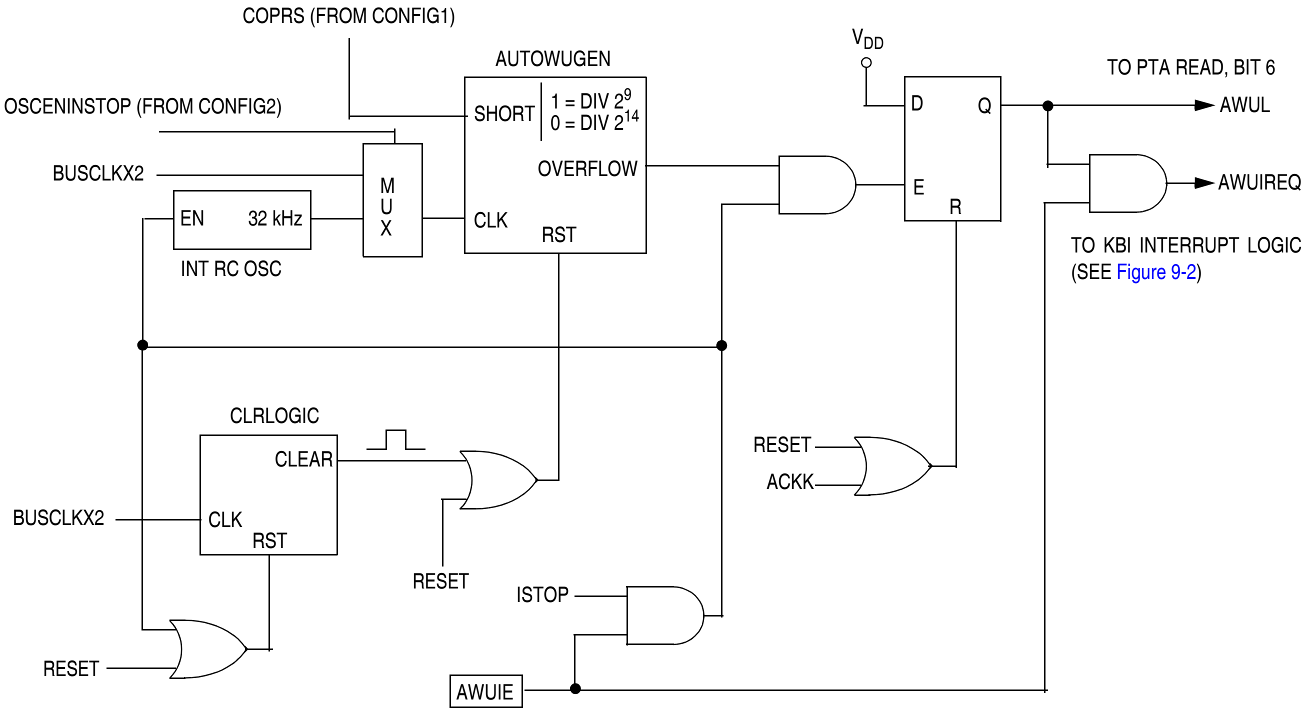


Figure 4-1. Auto Wakeup Interrupt Request Generation Logic

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.

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

▨ = Unimplemented

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

AWUL — Auto Wakeup Latch

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

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

NOTE

PTA5–PTA0 bits are not used in conjuction with the auto wakeup feature. To see a description of these bits, see 12.3.1 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 conjuction with the auto wakeup feature. To see a description of this bit, see 9.8.1 Keyboard Status and Control Register (KBSCR).

4.6.3 Keyboard Interrupt Enable Register

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

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

▨ = Unimplemented

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

AWUIE — Auto Wakeup Interrupt Enable Bit

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

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

NOTE

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

4.6.4 Configuration Register 2

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

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

Figure 4-5. Configuration Register 2 (CONFIG2)

OSCENINSTOP — Oscillator Enable in Stop Mode Bit

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

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

NOTE

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

4.6.5 Configuration Register 1

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

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVITRIP	SSREC	STOP	COPD
Reset: POR:	0	0	0	0	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 × (INTRCOSC or BUSCLKX2)
0 = Auto wakeup long cycle = 16,384 × (INTRCOSC or BUSCLKX2)

SSREC — Short Stop Recovery Bit

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

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

NOTE

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

Chapter 5
Configuration Register (CONFIG)

5.1 Introduction

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

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

5.2 Functional Description

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

NOTE

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

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

R = Reserved U = Unaffected

Figure 5-1. Configuration Register 2 (CONFIG2)

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

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

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

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

OSCENINSTOP— Oscillator Enable in Stop Mode Bit

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

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

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

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

NOTE

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

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

U = Unaffected

Figure 5-2. Configuration Register 1 (CONFIG1)

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

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

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

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

LVISTOP — LVI Enable in Stop Mode Bit

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

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

LVIRSTD — LVI Reset Disable Bit

LVIRSTD disables the reset signal from the LVI module.

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

LVIPWRD — LVI Power Disable Bit

LVIPWRD disables the LVI module.

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

LVITRIP — LVI Trip Point Selection Bit

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

1 = LVI operates for a 5-V protection
0 = LVI operates for a 3.3-V protection

NOTE

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

SSREC — Short Stop Recovery Bit

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

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

NOTE

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

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.

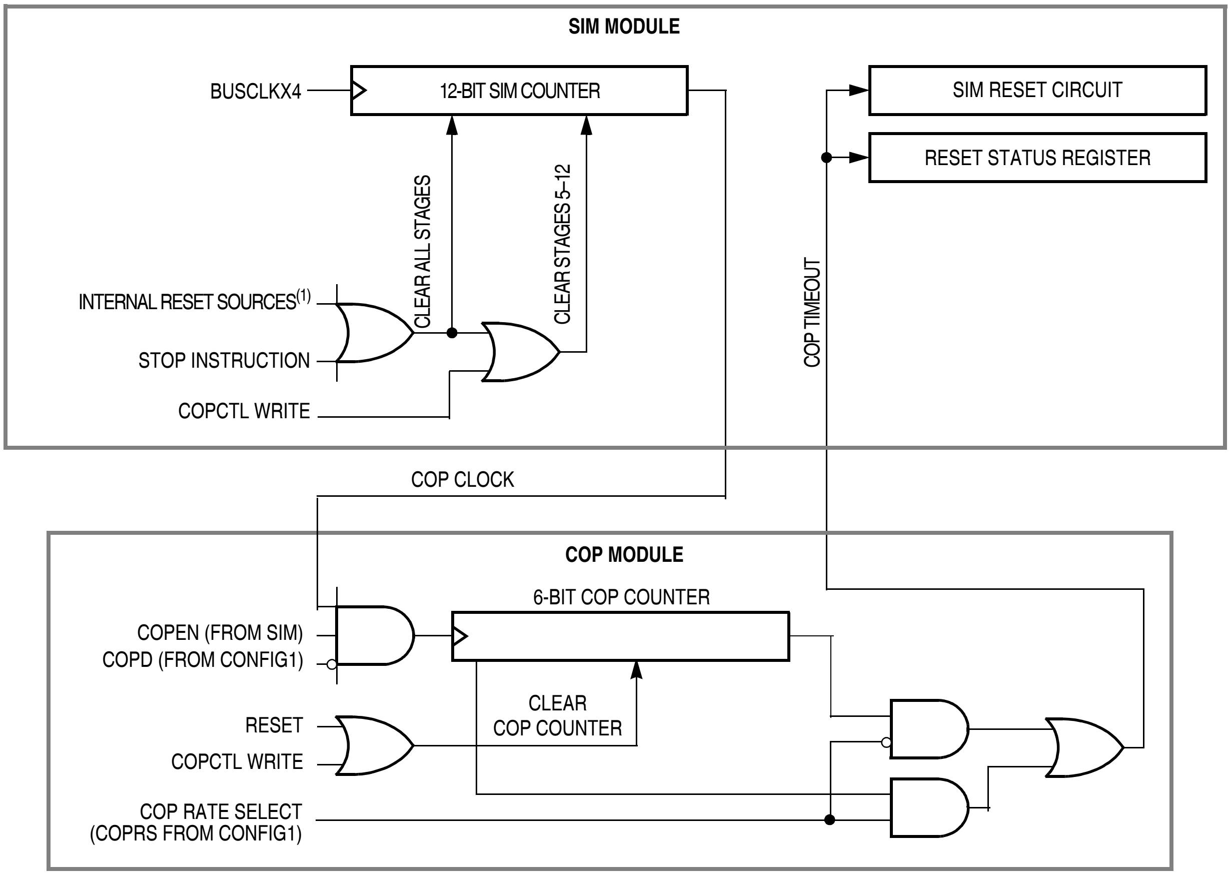
1 = COP module disabled
0 = COP module enabled

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

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 CONFIG1 register) for 32 × 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 × 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 $\overline{\text{IRQ}}$ 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.

Address: $FFFF

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

Figure 6-2. COP Control Register (COPCTL)

Chapter 7
Central Processor Unit (CPU)

7.1 Introduction

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

7.2 Features

Features of the CPU include:

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

7.3 CPU Registers

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

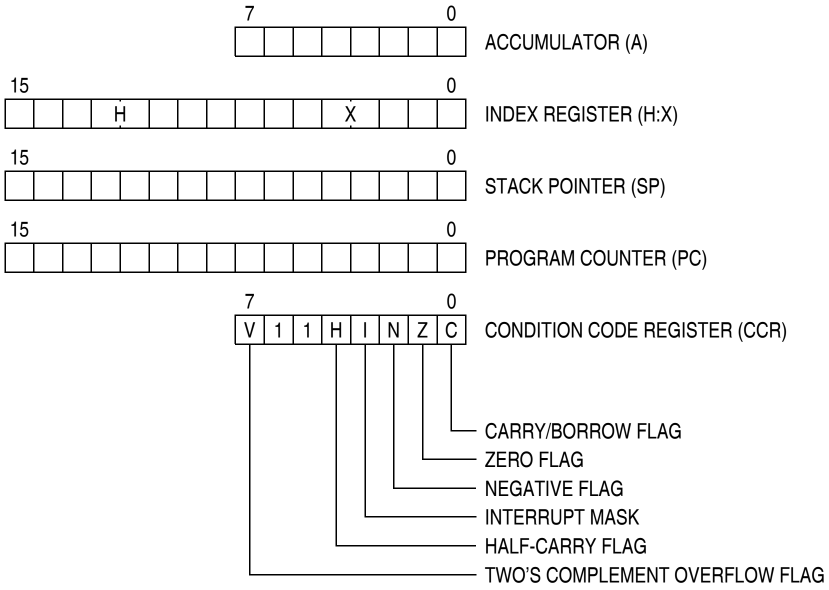


Figure 7-1. CPU Registers

7.3.1 Accumulator

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

	Bit 7	6	5	4	3	2	1	Bit 0
Read:								
Write:								
Reset:	Unaffected by reset							

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.

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

X = Indeterminate

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.

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

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.

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:																
Reset:	Loaded with vector from $FFFE and $FFFF															

Figure 7-5. Program Counter (PC)

7.3.5 Condition Code Register

The 8-bit condition code register contains the interrupt mask and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to 1. The following paragraphs describe the functions of the condition code register.

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

X = Indeterminate

Figure 7-6. Condition Code Register (CCR)

V — Overflow Flag

The CPU sets the overflow flag when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow flag.

1 = Overflow
0 = No overflow

H — Half-Carry Flag

The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an add-without-carry (ADD) or add-with-carry (ADC) operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C flags to determine the appropriate correction factor.

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

I — Interrupt Mask

When the interrupt mask is set, all maskable CPU interrupts are disabled. CPU interrupts are enabled when the interrupt mask is cleared. When a CPU interrupt occurs, the interrupt mask is set automatically after the CPU registers are saved on the stack, but before the interrupt vector is fetched.

1 = Interrupts disabled
0 = Interrupts enabled

NOTE

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

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

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

N — Negative Flag

The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result.

1 = Negative result
0 = Non-negative result

Z — Zero Flag

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

1 = Zero result
0 = Non-zero result

C — Carry/Borrow Flag

The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions — such as bit test and branch, shift, and rotate — also clear or set the carry/borrow flag.

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

7.4 Arithmetic/Logic Unit (ALU)

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

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

7.5 Low-Power Modes

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

7.5.1 Wait Mode

The WAIT instruction:

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

7.5.2 Stop Mode

The STOP instruction:

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

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

7.6 CPU During Break Interrupts

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

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

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

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

7.7 Instruction Set Summary

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

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

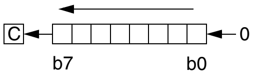
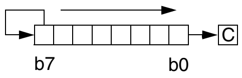
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X ADC *opr*,SP ADC *opr*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9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

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

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

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

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

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

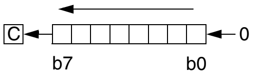
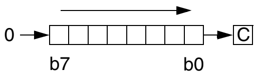
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff 	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR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

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

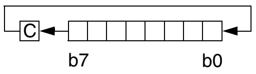
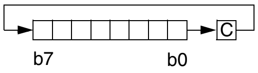
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–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

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

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

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

7.8 Opcode Map

See Table 7-2.

Table 7-2. Opcode Map

	Bit Manipulation		Branch	Read-Modify-Write						Control		Register/Memory							
	DIR	DIR	REL	DIR	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX
MSB LSB	0	1	2	3	4	5	6	9E6	7	8	9	A	B	C	D	9ED	E	9EE	F
0	5 BRSET0 3 DIR	4 BSET0 2 DIR	3 BRA 2 REL	4 NEG 2 DIR	1 NEGA 1 INH	1 NEGX 1 INH	4 NEG 2 IX1	5 NEG 3 SP1	3 NEG 1 IX	7 RTI 1 INH	3 BGE 2 REL	2 SUB 2 IMM	3 SUB 2 DIR	4 SUB 3 EXT	4 SUB 3 IX2	5 SUB 4 SP2	3 SUB 2 IX1	4 SUB 3 SP1	2 SUB 1 IX
1	5 BRCLR0 3 DIR	4 BCLR0 2 DIR	3 BRN 2 REL	5 CBEQ 3 DIR	4 CBEQA 3 IMM	4 CBEQX 3 IMM	5 CBEQ 3 IX1+	6 CBEQ 4 SP1	4 CBEQ 2 IX+	4 RTS 1 INH	3 BLT 2 REL	2 CMP 2 IMM	3 CMP 2 DIR	4 CMP 3 EXT	4 CMP 3 IX2	5 CMP 4 SP2	3 CMP 2 IX1	4 CMP 3 SP1	2 CMP 1 IX
2	5 BRSET1 3 DIR	4 BSET1 2 DIR	3 BHI 2 REL		5 MUL 1 INH	7 DIV 1 INH	3 NSA 1 INH		2 DAA 1 INH		3 BGT 2 REL	2 SBC 2 IMM	3 SBC 2 DIR	4 SBC 3 EXT	4 SBC 3 IX2	5 SBC 4 SP2	3 SBC 2 IX1	4 SBC 3 SP1	2 SBC 1 IX
3	5 BRCLR1 3 DIR	4 BCLR1 2 DIR	3 BLS 2 REL	4 COM 2 DIR	1 COMA 1 INH	1 COMX 1 INH	4 COM 2 IX1	5 COM 3 SP1	3 COM 1 IX	9 SWI 1 INH	3 BLE 2 REL	2 CPX 2 IMM	3 CPX 2 DIR	4 CPX 3 EXT	4 CPX 3 IX2	5 CPX 4 SP2	3 CPX 2 IX1	4 CPX 3 SP1	2 CPX 1 IX
4	5 BRSET2 3 DIR	4 BSET2 2 DIR	3 BCC 2 REL	4 LSR 2 DIR	1 LSRA 1 INH	1 LSRX 1 INH	4 LSR 2 IX1	5 LSR 3 SP1	3 LSR 1 IX	2 TAP 1 INH	2 TXS 1 INH	2 AND 2 IMM	3 AND 2 DIR	4 AND 3 EXT	4 AND 3 IX2	5 AND 4 SP2	3 AND 2 IX1	4 AND 3 SP1	2 AND 1 IX
5	5 BRCLR2 3 DIR	4 BCLR2 2 DIR	3 BCS 2 REL	4 STHX 2 DIR	3 LDHX 3 IMM	4 LDHX 2 DIR	3 CPHX 3 IMM		4 CPHX 2 DIR	1 TPA 1 INH	2 TSX 1 INH	2 BIT 2 IMM	3 BIT 2 DIR	4 BIT 3 EXT	4 BIT 3 IX2	5 BIT 4 SP2	3 BIT 2 IX1	4 BIT 3 SP1	2 BIT 1 IX
6	5 BRSET3 3 DIR	4 BSET3 2 DIR	3 BNE 2 REL	4 ROR 2 DIR	1 RORA 1 INH	1 RORX 1 INH	4 ROR 2 IX1	5 ROR 3 SP1	3 ROR 1 IX	2 PULA 1 INH		2 LDA 2 IMM	3 LDA 2 DIR	4 LDA 3 EXT	4 LDA 3 IX2	5 LDA 4 SP2	3 LDA 2 IX1	4 LDA 3 SP1	2 LDA 1 IX
7	5 BRCLR3 3 DIR	4 BCLR3 2 DIR	3 BEQ 2 REL	4 ASR 2 DIR	1 ASRA 1 INH	1 ASRX 1 INH	4 ASR 2 IX1	5 ASR 3 SP1	3 ASR 1 IX	2 PSHA 1 INH	1 TAX 1 INH	2 AIS 2 IMM	3 STA 2 DIR	4 STA 3 EXT	4 STA 3 IX2	5 STA 4 SP2	3 STA 2 IX1	4 STA 3 SP1	2 STA 1 IX
8	5 BRSET4 3 DIR	4 BSET4 2 DIR	3 BHCC 2 REL	4 LSL 2 DIR	1 LSLA 1 INH	1 LSLX 1 INH	4 LSL 2 IX1	5 LSL 3 SP1	3 LSL 1 IX	2 PULX 1 INH	1 CLC 1 INH	2 EOR 2 IMM	3 EOR 2 DIR	4 EOR 3 EXT	4 EOR 3 IX2	5 EOR 4 SP2	3 EOR 2 IX1	4 EOR 3 SP1	2 EOR 1 IX
9	5 BRCLR4 3 DIR	4 BCLR4 2 DIR	3 BHCS 2 REL	4 ROL 2 DIR	1 ROLA 1 INH	1 ROLX 1 INH	4 ROL 2 IX1	5 ROL 3 SP1	3 ROL 1 IX	2 PSHX 1 INH	1 SEC 1 INH	2 ADC 2 IMM	3 ADC 2 DIR	4 ADC 3 EXT	4 ADC 3 IX2	5 ADC 4 SP2	3 ADC 2 IX1	4 ADC 3 SP1	2 ADC 1 IX
A	5 BRSET5 3 DIR	4 BSET5 2 DIR	3 BPL 2 REL	4 DEC 2 DIR	1 DECA 1 INH	1 DECX 1 INH	4 DEC 2 IX1	5 DEC 3 SP1	3 DEC 1 IX	2 PULH 1 INH	2 CLI 1 INH	2 ORA 2 IMM	3 ORA 2 DIR	4 ORA 3 EXT	4 ORA 3 IX2	5 ORA 4 SP2	3 ORA 2 IX1	4 ORA 3 SP1	2 ORA 1 IX
B	5 BRCLR5 3 DIR	4 BCLR5 2 DIR	3 BMI 2 REL	5 DBNZ 3 DIR	3 DBNZA 2 INH	3 DBNZX 2 INH	5 DBNZ 3 IX1	6 DBNZ 4 SP1	4 DBNZ 2 IX	2 PSHH 1 INH	2 SEI 1 INH	2 ADD 2 IMM	3 ADD 2 DIR	4 ADD 3 EXT	4 ADD 3 IX2	5 ADD 4 SP2	3 ADD 2 IX1	4 ADD 3 SP1	2 ADD 1 IX
C	5 BRSET6 3 DIR	4 BSET6 2 DIR	3 BMC 2 REL	4 INC 2 DIR	1 INCA 1 INH	1 INCX 1 INH	4 INC 2 IX1	5 INC 3 SP1	3 INC 1 IX	1 CLRH 1 INH	1 RSP 1 INH		2 JMP 2 DIR	3 JMP 3 EXT	4 JMP 3 IX2		3 JMP 2 IX1		2 JMP 1 IX
D	5 BRCLR6 3 DIR	4 BCLR6 2 DIR	3 BMS 2 REL	3 TST 2 DIR	1 TSTA 1 INH	1 TSTX 1 INH	3 TST 2 IX1	4 TST 3 SP1	2 TST 1 IX		1 NOP 1 INH	4 BSR 2 REL	4 JSR 2 DIR	5 JSR 3 EXT	6 JSR 3 IX2		5 JSR 2 IX1		4 JSR 1 IX
E	5 BRSET7 3 DIR	4 BSET7 2 DIR	3 BIL 2 REL		5 MOV 3 DD	4 MOV 2 DIX+	4 MOV 3 IMD		4 MOV 2 IX+D	1 STOP 1 INH	*	2 LDX 2 IMM	3 LDX 2 DIR	4 LDX 3 EXT	4 LDX 3 IX2	5 LDX 4 SP2	3 LDX 2 IX1	4 LDX 3 SP1	2 LDX 1 IX
F	5 BRCLR7 3 DIR	4 BCLR7 2 DIR	3 BIH 2 REL	3 CLR 2 DIR	1 CLRA 1 INH	1 CLRX 1 INH	3 CLR 2 IX1	4 CLR 3 SP1	2 CLR 1 IX	1 WAIT 1 INH	1 TXA 1 INH	2 AIX 2 IMM	3 STX 2 DIR	4 STX 3 EXT	4 STX 3 IX2	5 STX 4 SP2	3 STX 2 IX1	4 STX 3 SP1	2 STX 1 IX

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

*Pre-byte for stack pointer indexed instructions

	MSB LSB	0	High Byte of Opcode in Hexadecimal
Low Byte of Opcode in Hexadecimal	0	5 BRSET0 3 DIR	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.

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 $\overline{\text{IRQ}}$ pin.

The $\overline{\text{IRQ}}$ 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

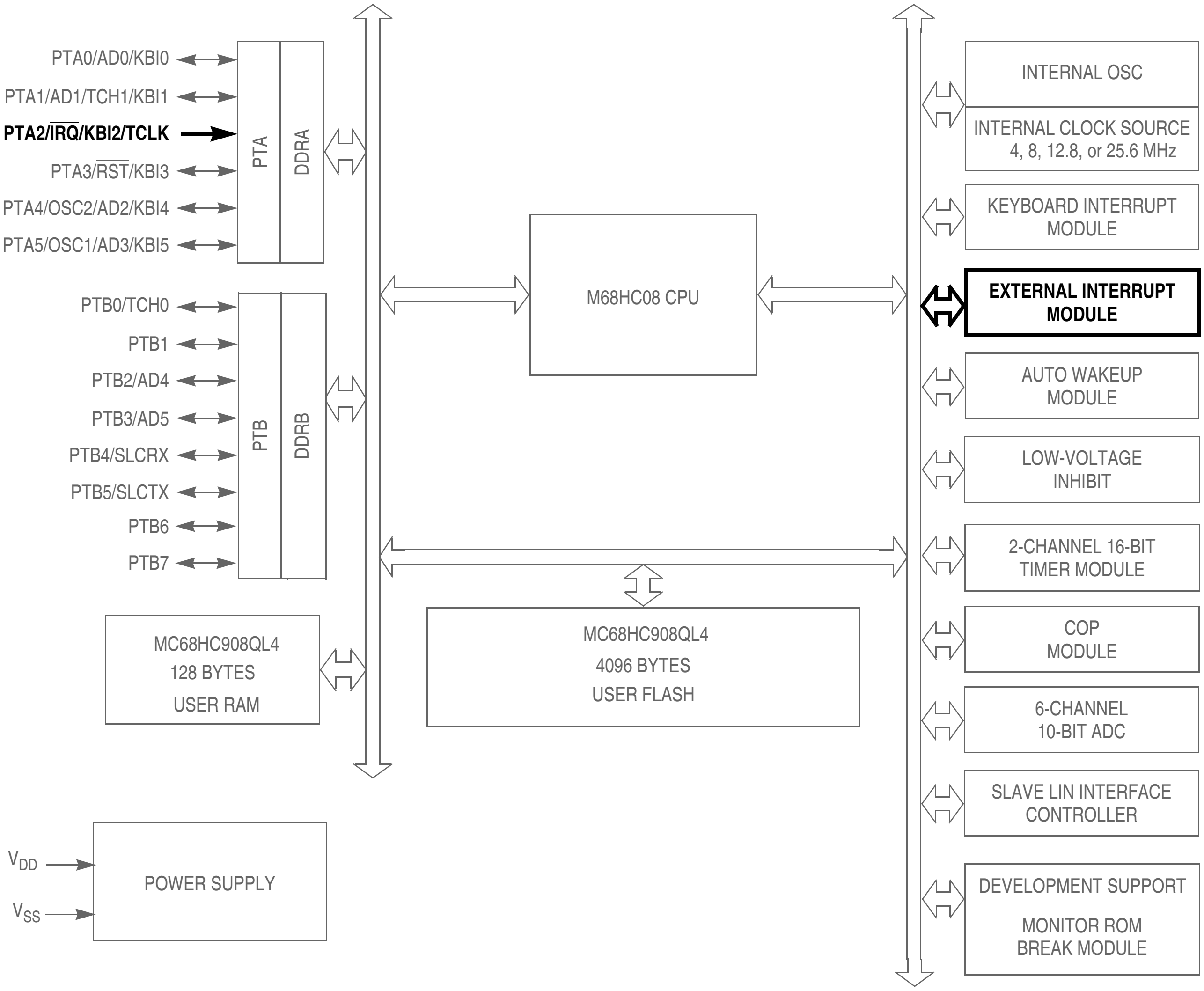
8.3 Functional Description

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.



$\overline{\text{RST}}$, $\overline{\text{IRQ}}$: Pins have internal pull up device
All port pins have programmable pull up device (pullup/down on port A)
PTA[0:5]: Higher current sink and source capability

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

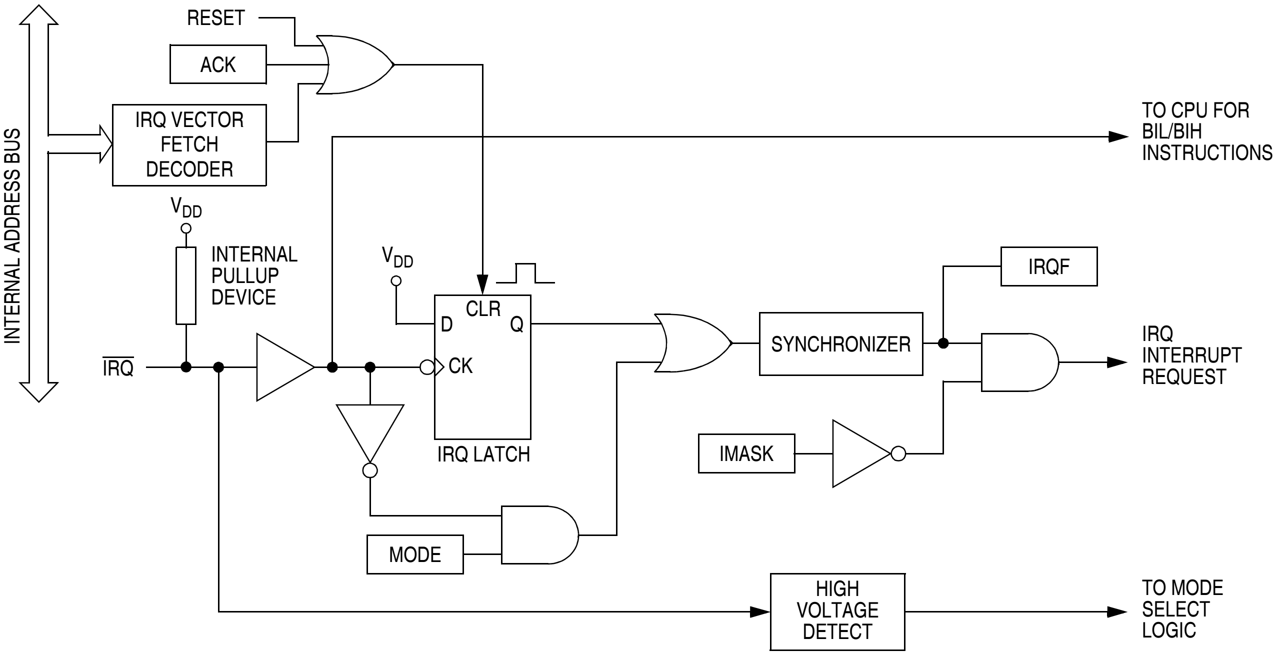


Figure 8-2. IRQ Module Block Diagram

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

ACK — IRQ Interrupt Request Acknowledge Bit

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

IMASK — IRQ Interrupt Mask Bit

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

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

MODE — IRQ Edge/Level Select Bit

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

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

Chapter 9
Keyboard Interrupt Module (KBI)

9.1 Introduction

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

The KBI shares its pins with general-purpose input/output (I/O) port pins. See Figure 9-1 for port location of these shared pins.

9.2 Features

Features of the keyboard interrupt module include:

- Keyboard interrupt pins with separate keyboard interrupt enable bits and one keyboard interrupt mask
- Programmable edge-only or edge and level interrupt sensitivity
- Edge sensitivity programmable for rising or falling edge
- Level sensitivity programmable for high or low level
- KBI can be configured to use either internal or external pullup/pulldown devices using PTAPUE, see 12.3.3 Port A Input Pullup/Down Enable Register
  – When in internal device is enabled, pullup or pulldown device automatically configured based on the polarity of edge or level detect
- Exit from low-power modes

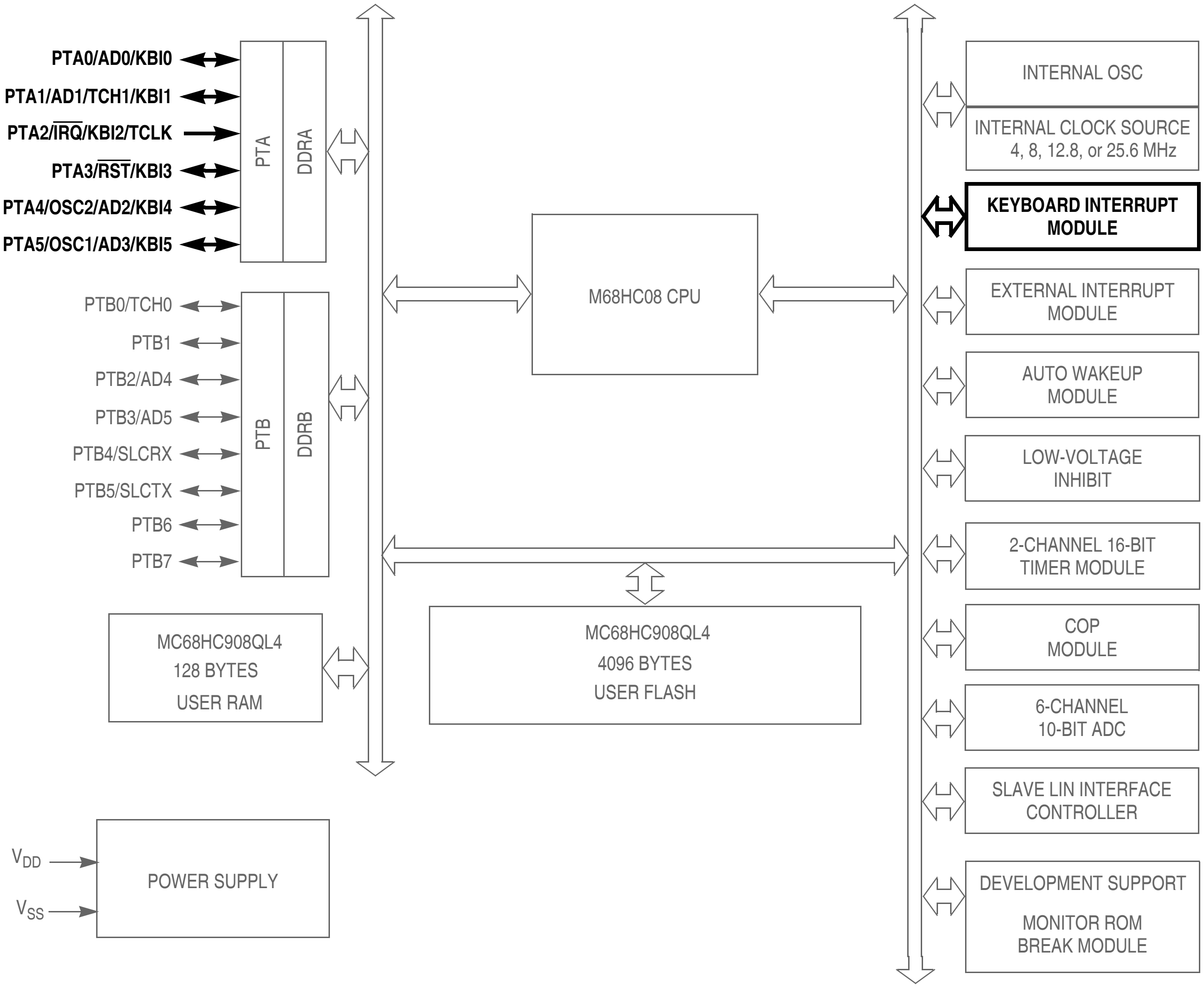
9.3 Functional Description

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 and the corresponding port pullup/down enable bit, PTAPUEx see 12.3.3 Port A Input Pullup/Down Enable Register. On falling edge or low level detection, a pullup device is configured. On rising edge or high level detection, a pulldown device is configured.

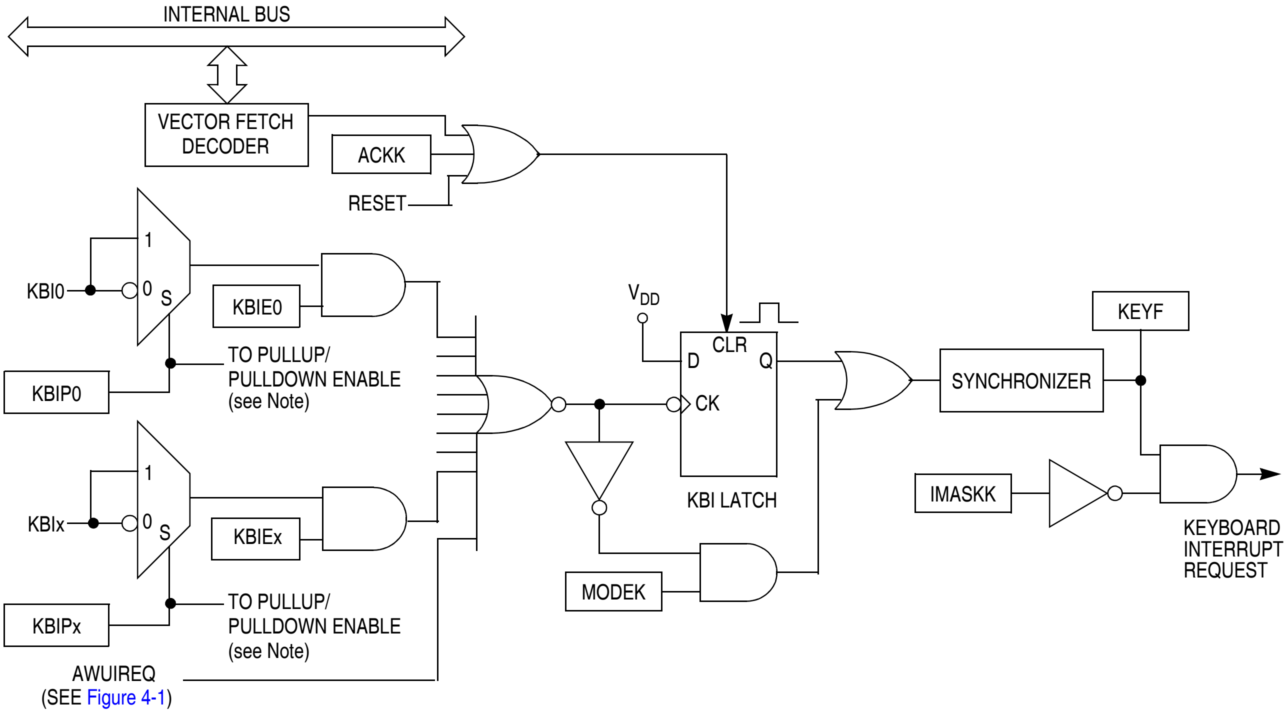


$\overline{\text{RST}}$, $\overline{\text{IRQ}}$: Pins have internal pull up device
All port pins have programmable pull up device (pullup/down on port A)
PTA[0:5]: Higher current sink and source capability

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

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

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



NOTE:
To enable internal pullup/pulldown, requires both the KBIPx and the corresponding PTAPUEN to both be set.

Figure 9-2. Keyboard Interrupt Block Diagram

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.

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

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

9.3.1.2 MODEK = 0

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

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

NOTE

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

9.3.2 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup or pulldown device if selected 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 (KBI5:KBI0)

Each KBI pin is independently programmable as an external interrupt source. KBI pin polarity can be controlled independently. Each KBI pin when enabled can be configured to use an internal pullup/pulldown device using the corresponding PTAPUEx bit see 12.3.3 Port A Input Pullup/Down Enable Register. The selection of pullup or pulldown is automatically configured to match the polarity selected in KBIPR.

9.8 Registers

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

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

9.8.1 Keyboard Status and Control Register (KBSCR)

Features of the KBSCR:

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

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

▨ = Unimplemented

Figure 9-3. Keyboard Status and Control Register (KBSCR)

Bits 7–4 — Not used

KEYF — Keyboard Flag Bit

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

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

ACKK — Keyboard Acknowledge Bit

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

IMASKK— Keyboard Interrupt Mask Bit

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

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

MODEK — Keyboard Triggering Sensitivity Bit

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

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

9.8.2 Keyboard Interrupt Enable Register (KBIER)

KBIER enables or disables each keyboard interrupt pin.

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

▨ = Unimplemented

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

KBIE5–KBIE0 — Keyboard Interrupt Enable Bits

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

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

NOTE

KBIEx bit does not automatically enable the internal pullup/pulldown device. Internal pullup/pulldown device is selected using the corresponding PTAPUEx bit. Refer to PTAPUE bit description, see 12.3.3 Port A Input Pullup/Down Enable Register.

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. Port pulldown is enabled if the corresponding PTAPUE bit is set.
0 = Keyboard polarity is low level and/or falling edge. Port pullup is enabled if the corresponding PTAPUE bit is set.

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.

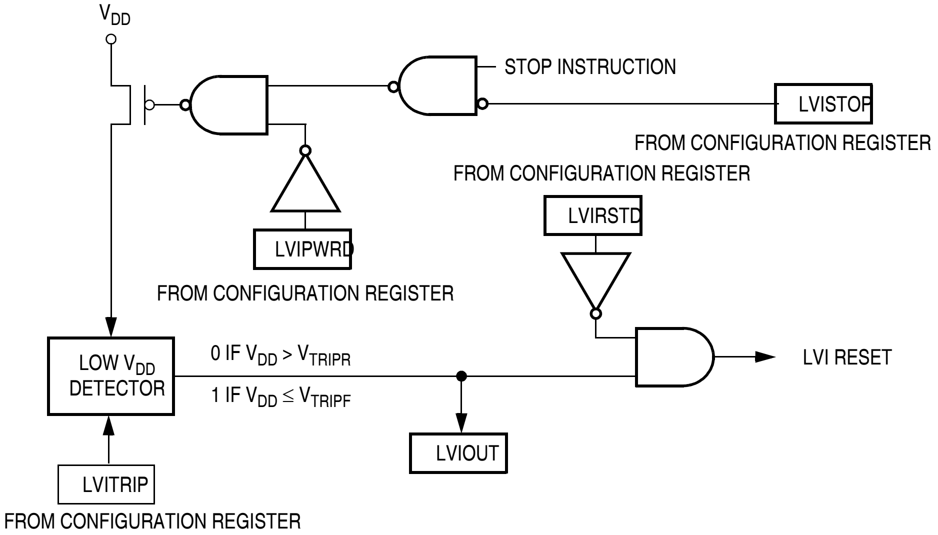
10.2 Features

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.



V_{DD}
STOP INSTRUCTION
LVISTOP
FROM CONFIGURATION REGISTER
FROM CONFIGURATION REGISTER
LVIRSTD
LVIPWRD
FROM CONFIGURATION REGISTER
LOW V_{DD} DETECTOR
0 IF $V_{DD} > V_{TRIPR}$
1 IF $V_{DD} \leq V_{TRIPF}$
LVI RESET
LVIOUT
LVITRIP
FROM CONFIGURATION REGISTER

Figure 10-1. LVI Module Block Diagram

The LVI module contains a bandgap reference circuit and comparator. When the LVITRIP bit is cleared, the default state at power-on reset, V_{TRIPF} is configured for the lower V_{DD} operating range. The actual trip points are specified in 17.5 5-V DC Electrical Characteristics and 17.8 3.3-V DC Electrical Characteristics.

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

After an LVI reset occurs, the MCU remains in reset until V_{DD} rises above V_{TRIPR}. See Chapter 13 System Integration Module (SIM) for the reset recovery sequence.

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

The LVI is enabled out of reset. The following bits located in the configuration register can alter the default conditions.

- Setting the LVI power disable bit, LVIPWRD, disables the LVI.
- Setting the LVI reset disable bit, LVIRSTD, prevents the LVI module from generating a reset.
- Setting the LVI enable in stop mode bit, LVISTOP, enables the LVI to operate in stop mode.
- Setting the LVI trip point bit, LVITRIP, configures the trip point voltage (V_{TRIPF}) for the higher V_{DD} operating range.

10.3.1 Polled LVI Operation

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

10.3.2 Forced Reset Operation

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

10.3.3 LVI Hysteresis

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

10.3.4 LVI Trip Selection

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

NOTE

The MCU is guaranteed to operate at a minimum supply voltage. The trip point (V_{TRIPF}) may be lower than this. See Chapter 17 Electrical Specifications for the actual trip point voltages.

10.4 LVI Interrupts

The LVI module does not generate interrupt requests.

10.5 Low-Power Modes

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

10.5.1 Wait Mode

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

10.5.2 Stop Mode

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

10.6 Registers

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

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

▨ = Unimplemented R = Reserved

Figure 10-2. LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

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

Table 10-1. LVIOUT Bit Indication

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

Chapter 11
Oscillator Module (OSC)

11.1 Introduction

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

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

11.2 Features

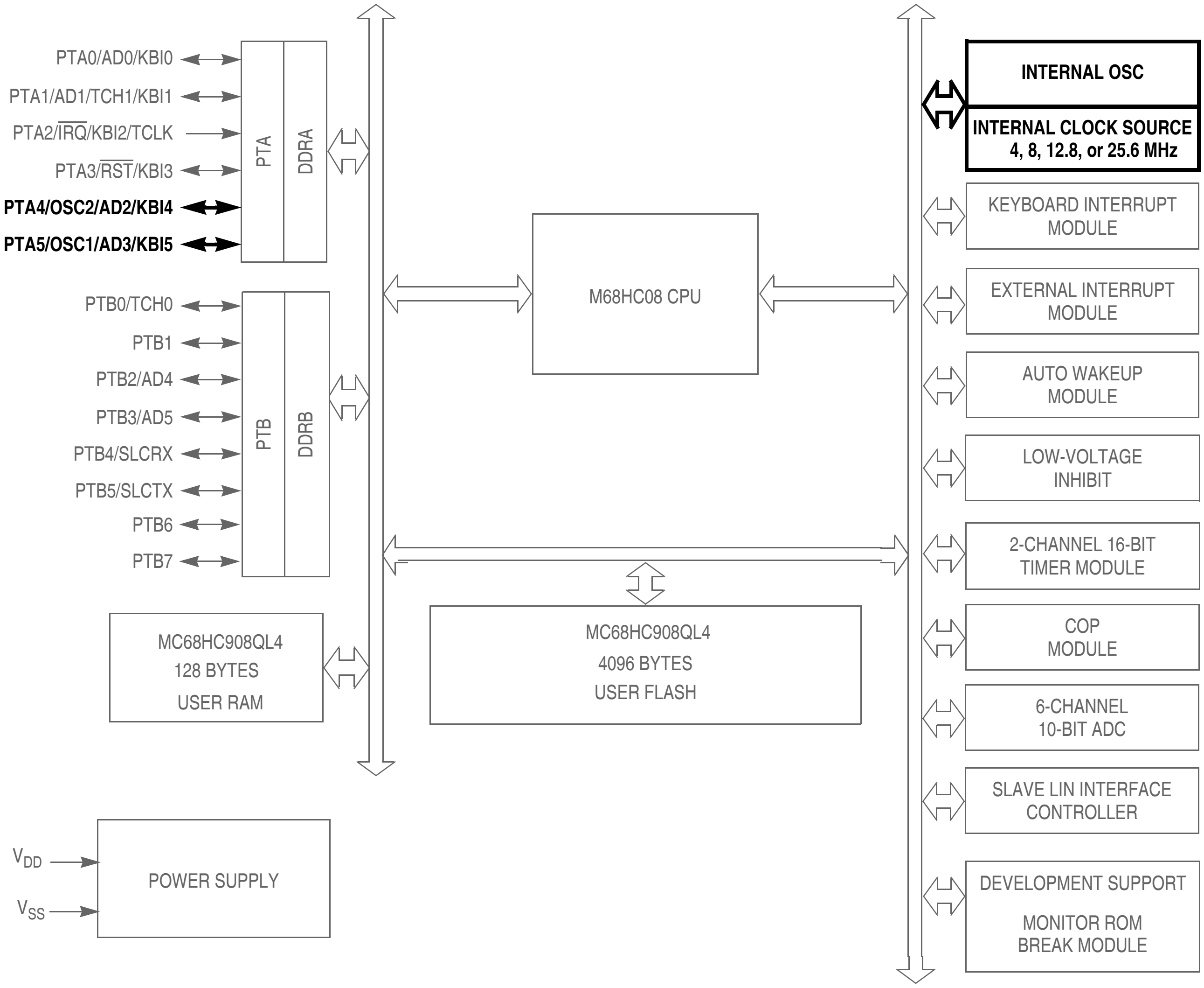
The bus clock frequency is one fourth of any of these clock source options:

1. Internal oscillator: An internally generated, fixed frequency clock, trimmable to ± 0.4%. There are four choices for the internal oscillator, 25.6 MHz, 12.8 MHz, 8 MHz or 4 MHz. The 12.8-MHz internal oscillator is the default option out of reset.
2. External oscillator: An external clock that can be driven directly into OSC1.
3. External RC: A built-in oscillator module (RC oscillator) that requires an external R connection only. The capacitor is internal to the chip.
4. External crystal: A built-in XTAL oscillator that requires an external crystal or ceramic-resonator. There are three crystal frequency ranges supported, 8–32 MHz, 1–8 MHz, and 32–100 kHz.

11.3 Functional Description

The oscillator contains these major subsystems:

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



RST, IRQ: Pins have internal pull up device
All port pins have programmable pull up device (pullup/down on port A)
PTA[0:5]: Higher current sink and source capability

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

11.3.1 Internal Signal Definitions

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

11.3.1.1 Oscillator Enable Signal (SIMOSCEN)

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

11.3.1.2 XTAL Oscillator Clock (XTALCLK)

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

11.3.1.3 RC Oscillator Clock (RCCLK)

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

11.3.1.4 Internal Oscillator Clock (INTCLK)

INTCLK is the internal oscillator output signal. INTCLK is software selectable to be nominally 25.6 MHz, 12.8 MHz, 8.0 MHz, or 4.0 MHz. INTCLK can be digitally adjusted using the oscillator trimming feature of the OSCTRIM register (see 11.3.2.1 Internal Oscillator Trimming).

11.3.1.5 Bus Clock Times 4 (BUSCLKX4)

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

11.3.1.6 Bus Clock Times 2 (BUSCLKX2)

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

11.3.2 Internal Oscillator

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

The internal oscillator is capable of generating clocks of 25.6 MHz, 12.8 MHz, 8.0 MHz, or 4.0 MHz (INTCLK) resulting in a bus frequency (INTCLK divided by 4) of 6.4 MHz, 3.2 MHz, 2.0 MHz, or 1.0 MHz respectively. The bus clock is software selectable and defaults to the 3.2-MHz bus out of reset. Users can increase the bus frequency based on the voltage range of their application.

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

11.3.2.1 Internal Oscillator Trimming

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

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

11.3.2.2 Internal to External Clock Switching

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

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

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

11.3.2.3 External to Internal Clock Switching

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

11.3.3 External Oscillator

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

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

11.3.4 XTAL Oscillator

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

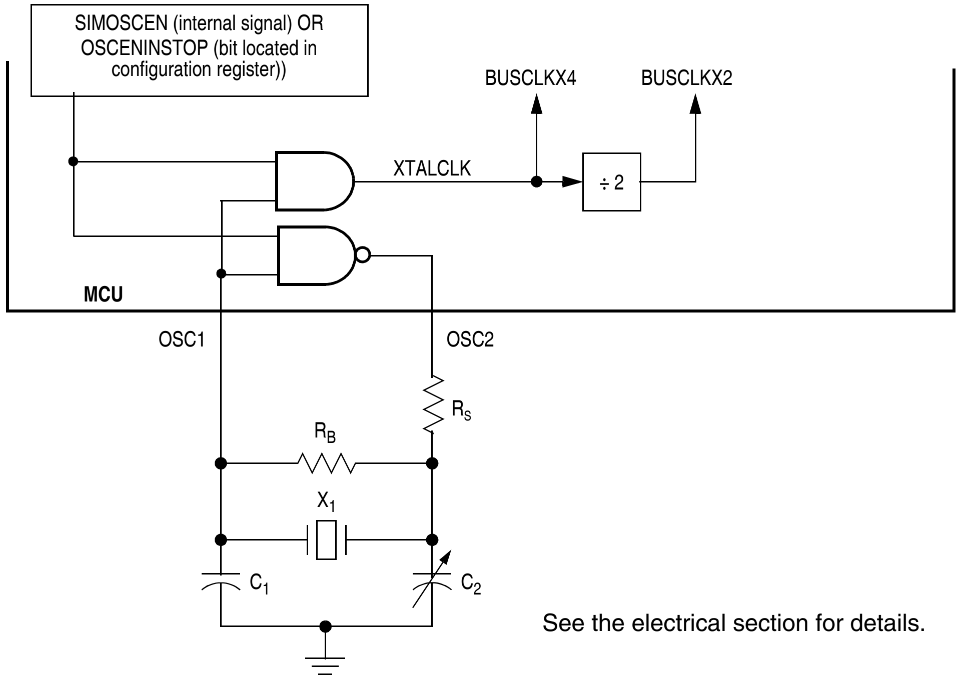
In its typical configuration, the XTAL oscillator is connected in a Pierce oscillator configuration, as shown in Figure 11-2. This figure shows only the logical representation of the internal components and may not represent actual circuitry.

The oscillator configuration uses five components:

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

NOTE

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



See the electrical section for details.

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}.

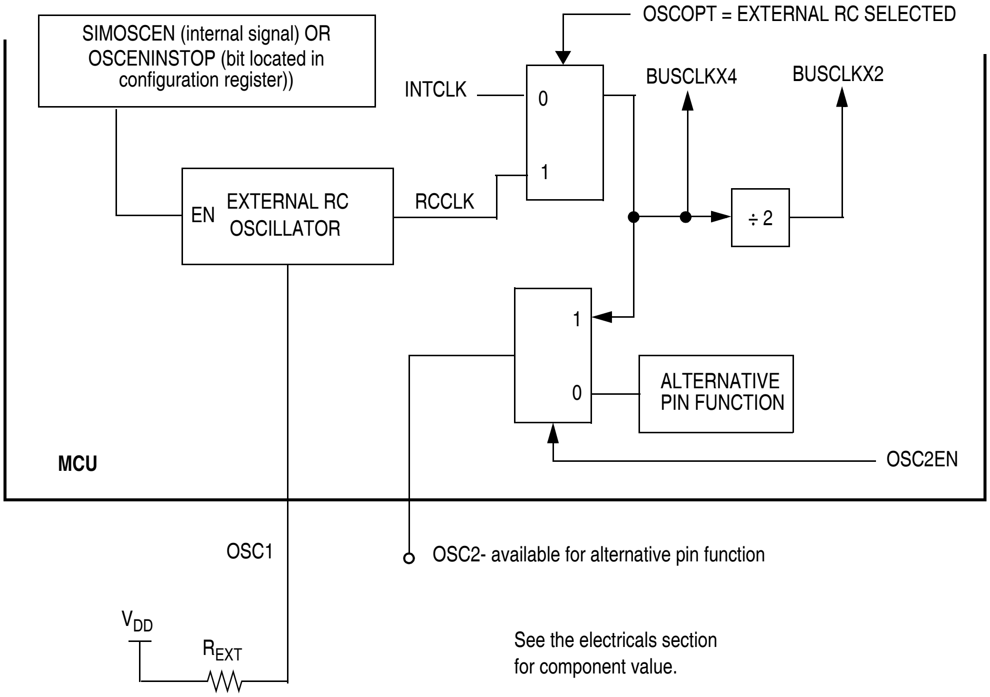

OSCOPT = EXTERNAL RC SELECTED
SIMOSCEN (internal signal) OR OSCENINSTOP (bit located in configuration register))
INTCLK
0
1
BUSCLKX4
BUSCLKX2
EN
EXTERNAL RC OSCILLATOR
RCCLK
÷ 2
1
0
ALTERNATIVE PIN FUNCTION
OSC2EN
MCU
OSC1
OSC2- available for alternative pin function
V_{DD}
R_{EXT}
See the electricals section for component value.

Figure 11-3. RC Oscillator External Connections

11.4 Interrupts

There are no interrupts associated with the OSC module.

11.5 Low-Power Modes

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

11.5.1 Wait Mode

The OSC module remains active in wait mode.

11.5.2 Stop Mode

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

11.6 OSC During Break Interrupts

There are no status flags associated with the OSC module.

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

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

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

11.7 I/O Signals

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

11.7.1 Oscillator Input Pin (OSC1)

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

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

11.7.2 Oscillator Output Pin (OSC2)

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

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

Table 11-1. OSC2 Pin Function

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

11.8 Registers

The oscillator module contains two registers:

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

11.8.1 Oscillator Status and Control Register

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

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

▨ = Unimplemented

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

OSCOPT1:OSCOPT0 — OSC Option Bits

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

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

ICFS1:ICFS0 — Internal Clock Frequency Select Bits

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

ICFS1	ICFS0	Internal Clock Frequency
0	0	4.0 MHz
0	1	8.0 MHz
1	0	12.8 MHz — default reset condition
1	1	25.6 MHz

ECFS1:ECFS0 — External Crystal Frequency Select Bits

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

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

ECGON — External Clock Generator On Bit

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

1 = External clock enabled
0 = External clock disabled

ECGST — External Clock Status Bit

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

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

11.8.2 Oscillator Trim Register (OSCTRIM)

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

Figure 11-5. Oscillator Trim Register (OSCTRIM)

TRIM7–TRIM0 — Internal Oscillator Trim Factor Bits

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

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

Chapter 12
Input/Output Ports (PORTS)

12.1 Introduction

The MC68HC908QL4 has thirteen bidirectional input-output (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 15 Timer Interface Module (TIM)), the 10-bit ADC (see Chapter 3 Analog-to-Digital Converter (ADC10) Module), the external interrupt (IRQ) pin (see Chapter 8 External Interrupt (IRQ)), the reset (RST) pin enabled using a configuration register (see Chapter 5 Configuration Register (CONFIG)) and the oscillator pins (see Chapter 11 Oscillator Module (OSC)).

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

NOTE

PTA2 is input only.

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

12.3.1 Port A Data Register

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

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

R = Reserved ▨ = Unimplemented

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/down 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.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	R	DDRA5	DDRA4	DDRA3	0	DDRA1	DDRA0
Write:						▨		
Reset:	0	0	0	0	0	0	0	0

R = Reserved ▨ = Unimplemented

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.

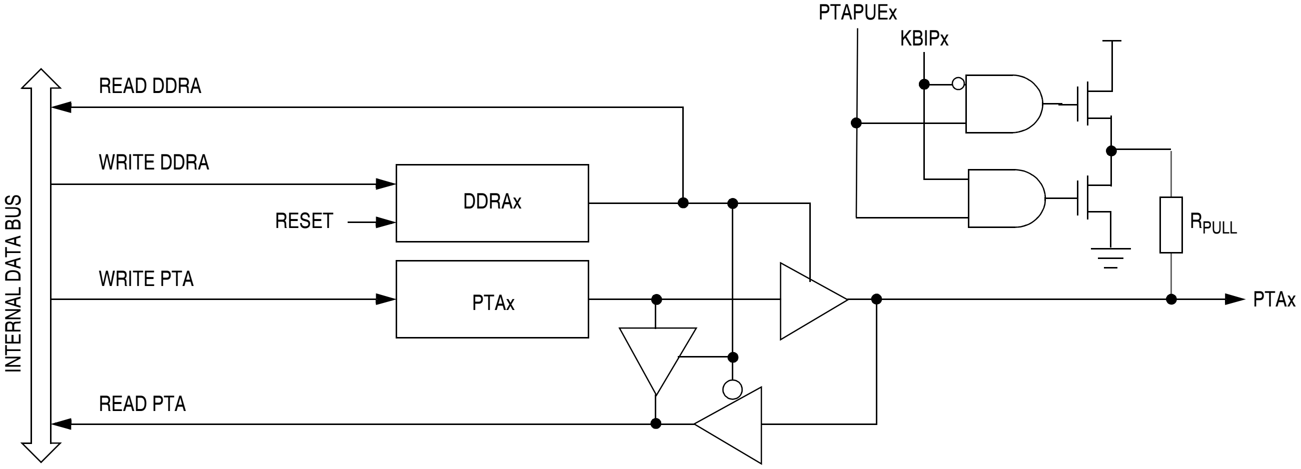


Figure 12-3. Port A I/O Circuit

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/Down Enable Register

The port A input pullup/down enable register (PTAPUE) contains a software configurable pullup/down 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/down device is automatically and dynamically disabled when its corresponding DDRAx bit is configured as output. The pull device polarity is defined by the KBIPR register, see 9.8.3 Keyboard Interrupt Polarity Register (KBIPR).

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

▨ = Unimplemented

Figure 12-4. Port A Input Pullup/Down 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/down functions

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

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

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

0 = Pullup/down 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[1]	Input, V_{Pull}[2]	DDRA5–DDRA0	Pin	PTA5–PTA0[3]
0	0	X	Input, Hi-Z[4]	DDRA5–DDRA0	Pin	PTA5–PTA0[3]
X	1	X	Output	DDRA5–DDRA0	PTA5–PTA0	PTA5–PTA0[5]

1. X = don't care
2. I/O pin pulled to V_{Pull} (V_{DD} or V_{SS}) by internal pullup or pulldown.
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 its pins with the 2-channel timer interface module (TIM) (see Chapter 15 Timer Interface Module (TIM)), the 10-bit ADC (see Chapter 3 Analog-to-Digital Converter (ADC10) Module), and the slave LIN interface controller (SLIC) module (see Chapter 14 Slave LIN Interface Controller (SLIC) 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.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
Reset:	Unaffected by reset							

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.

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

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

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

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

1 = Corresponding port B pin configured as output
0 = Corresponding port B pin configured as input

NOTE

Avoid glitches on port B pins by writing to the port B data register before changing data direction register B bits from 0 to 1. Figure 12-7 shows the port B I/O logic.

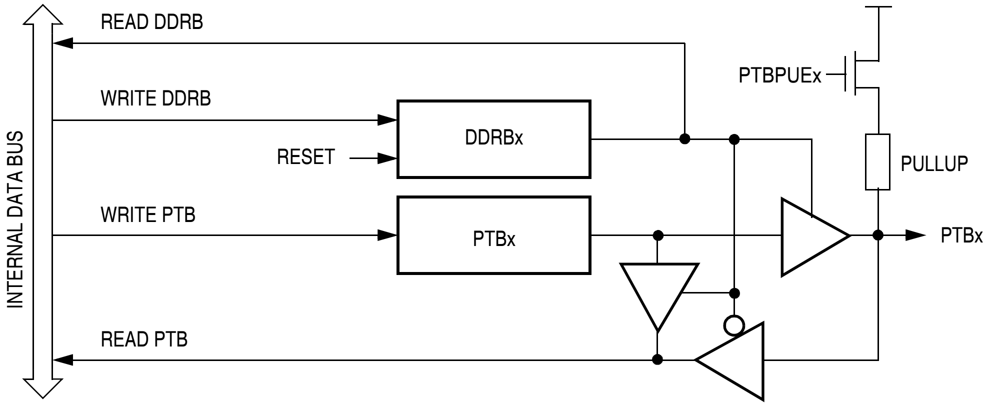


Figure 12-7. Port B I/O Circuit

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

12.4.3 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.

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

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

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

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

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

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

12.4.4 Port B Summary Table

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

Table 12-2. Port B Pin Functions

DDRB Bit	PTB Bit	I/O Pin Mode	Accesses to DDRB	Accesses to PTB	
			Read/Write	Read	Write
0	X[1]	Input, Hi-Z[2]	DDRB7–DDRB0	Pin	PTB7–PTB0[3]
1	X	Output	DDRB7–DDRB0	Pin	PTB7–PTB0

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

Chapter 13
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

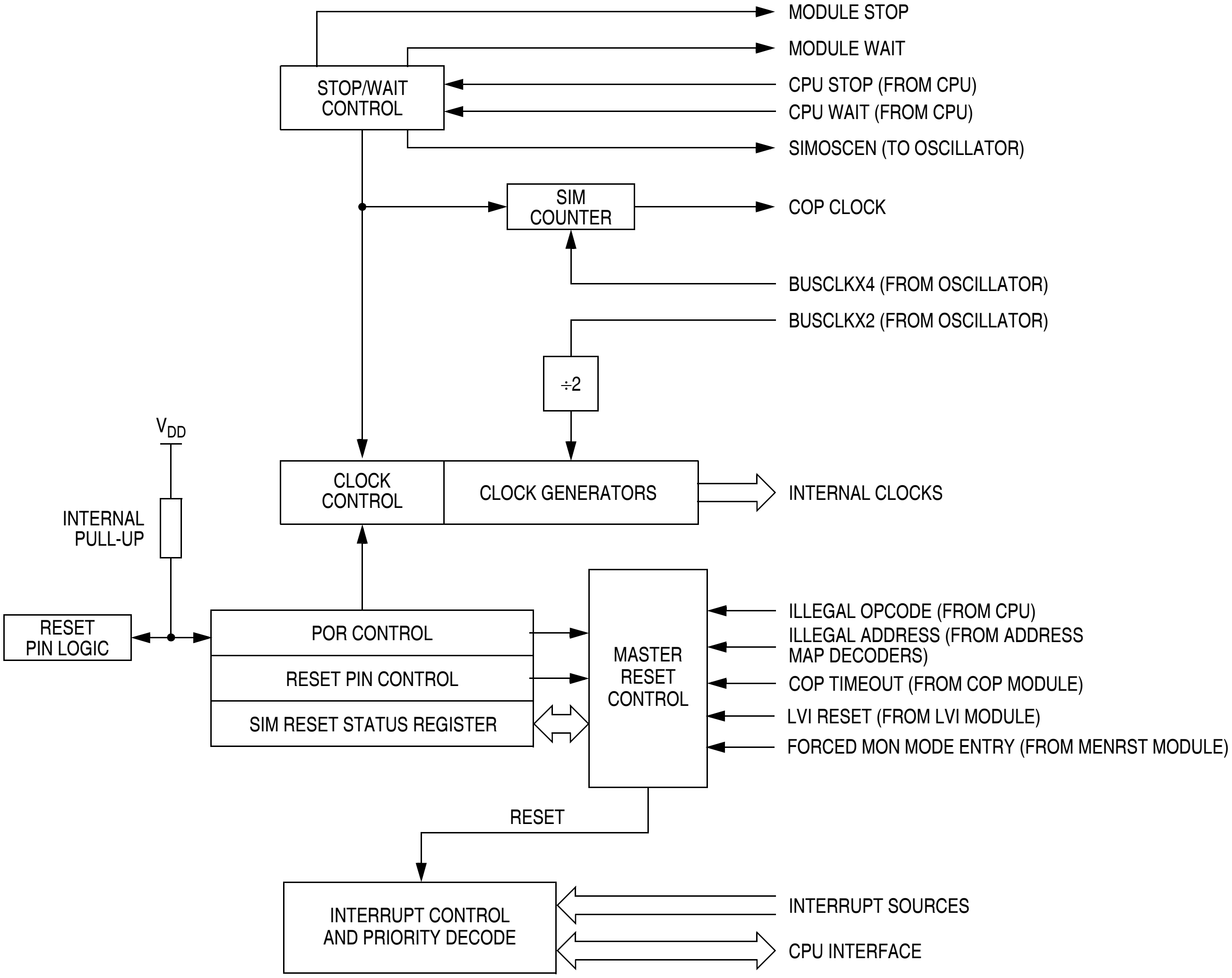


Figure 13-1. SIM Block Diagram

13.2 $\overline{\text{RST}}$ and $\overline{\text{IRQ}}$ Pins Initialization

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

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.

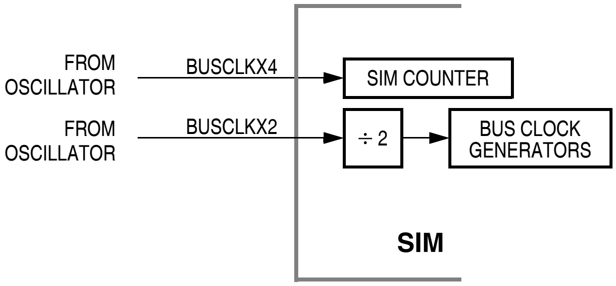


Figure 13-2. SIM Clock Signals

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_{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.

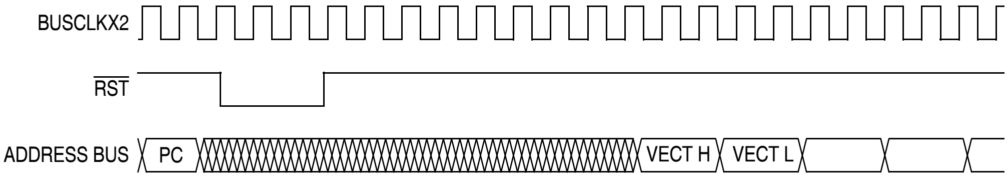


Figure 13-3. External Reset Timing

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).

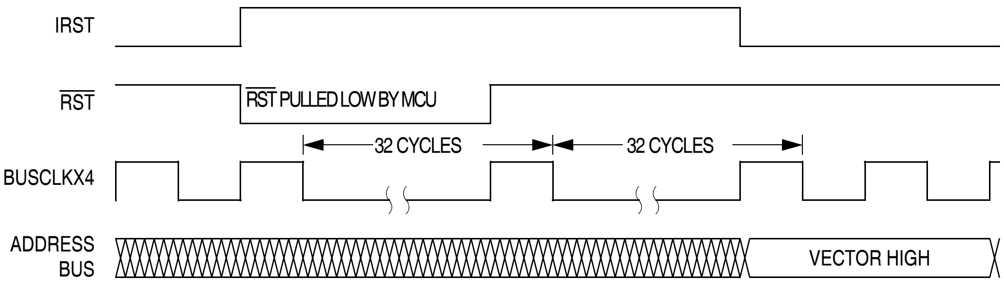


Figure 13-4. Internal Reset Timing

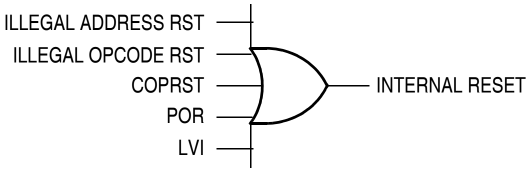


Figure 13-5. Sources of Internal Reset

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.

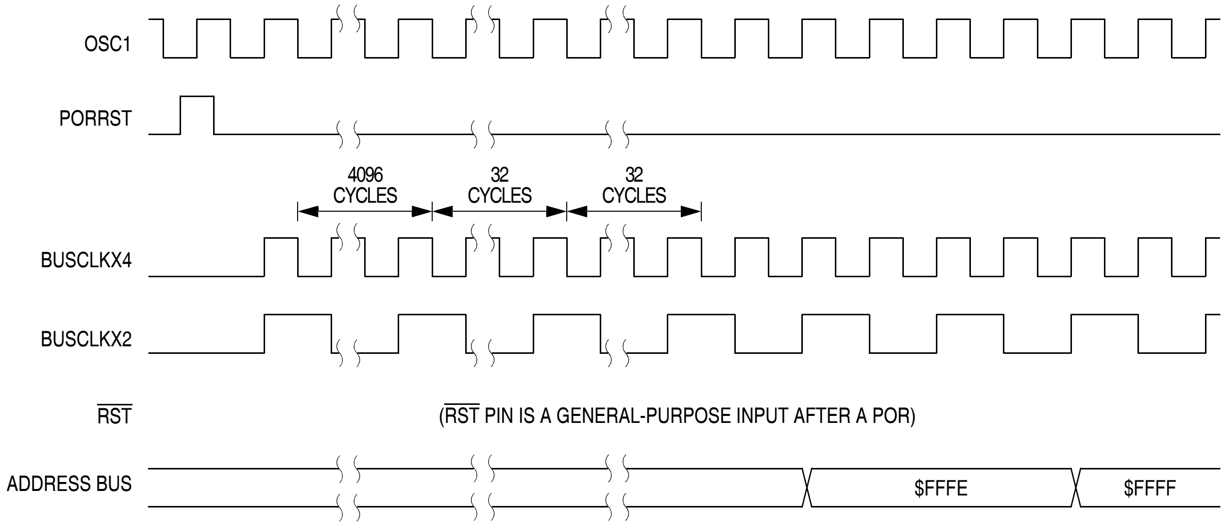


Figure 13-6. POR Recovery

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_{DD} voltage falls to the LVI trip voltage V_{TRIPF}. The LVI bit in the SIM reset status register (SRSR) is set, and the external reset pin ($\overline{\text{RST}}$) is held low while the

SIM counter counts out 4096 BUSCLKX4 cycles after V_{DD} rises above V_{TRIPR}. Sixty-four BUSCLKX4 cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur. The SIM actively pulls down the ($\overline{RST}$) 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.

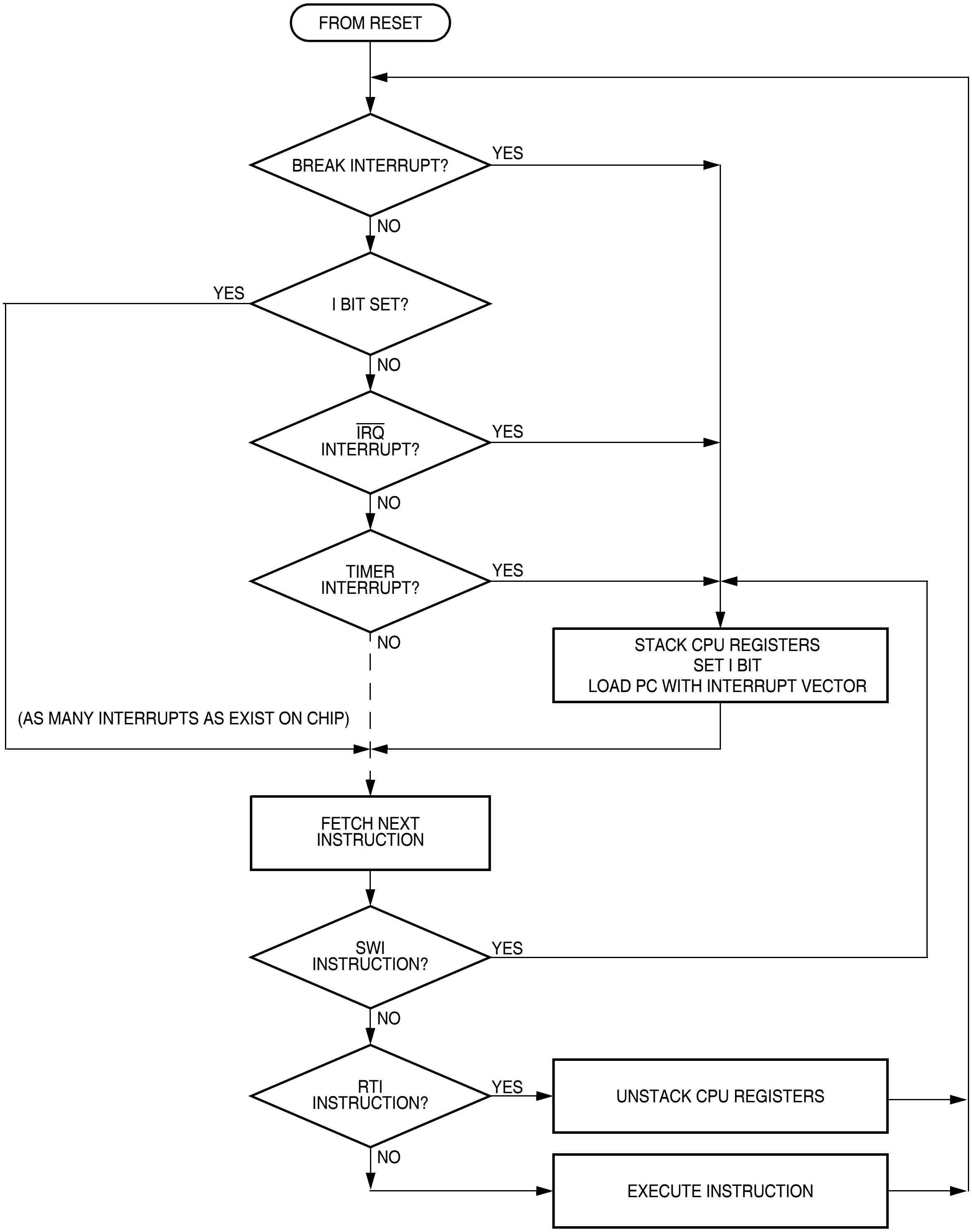


Figure 13-7. Interrupt Processing

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.

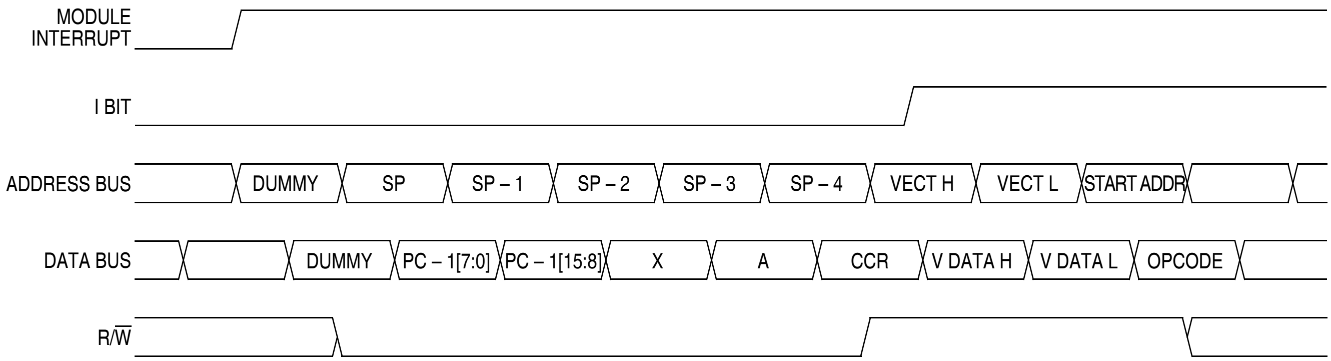


Figure 13-8. Interrupt Entry

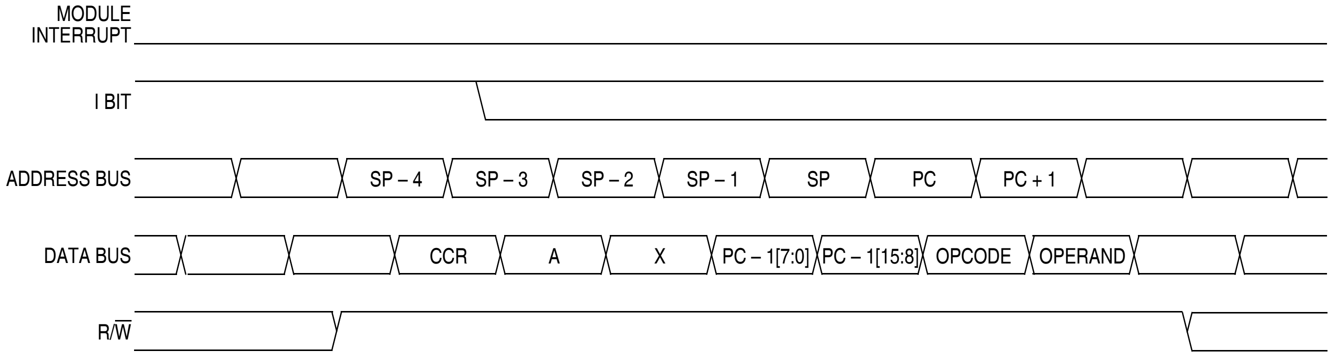


Figure 13-9. Interrupt Recovery

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.

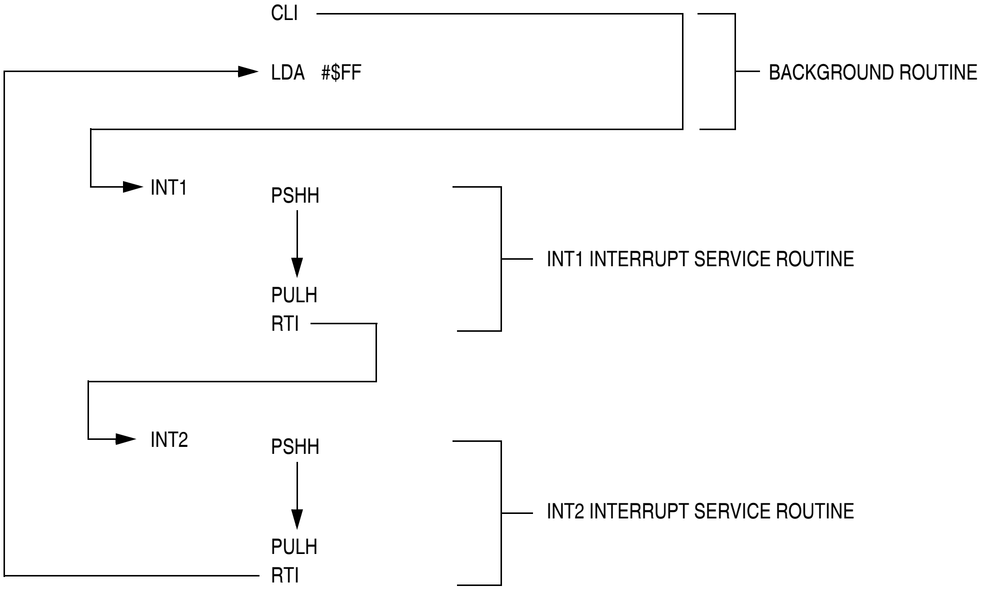

CLI
LDA #$FF
BACKGROUND ROUTINE
INT1
PSHH
PULH
RTI
INT1 INTERRUPT SERVICE ROUTINE
INT2
PSHH
PULH
RTI
INT2 INTERRUPT SERVICE ROUTINE

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[1]	INT Register Flag	Vector Address
Highest	Reset	—	—	—	$FFFE–$FFFF
	SWI instruction	—	—	—	$FFFC–$FFFD
	$\overline{\text{IRQ}}$ pin	IRQF1	IMASK1	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
	SLIC interrupt	SLCF	SLCIE	IF9	$FFEA–$FFEB
	Keyboard interrupt	KEYF	IMASKK	IF14	$FFE0–$FFE1
Lowest	ADC conversion complete interrupt	COCO	AIEN	IF15	$FFDE–$FFDF

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

	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

13.6.2.1 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

13.6.2.2 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 16 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.

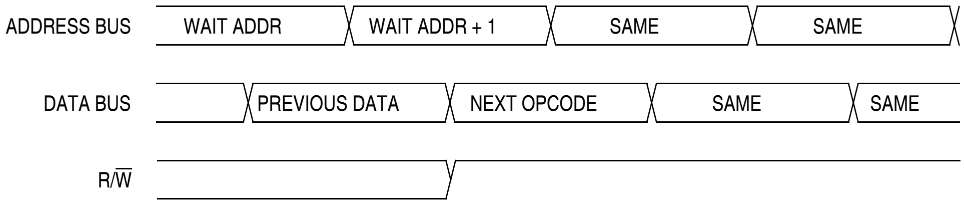
Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a two-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first 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.



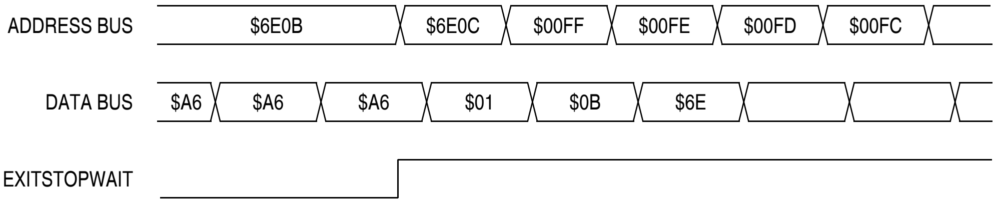
NOTE: Previous data can be operand data or the WAIT opcode, depending on the last instruction.

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.

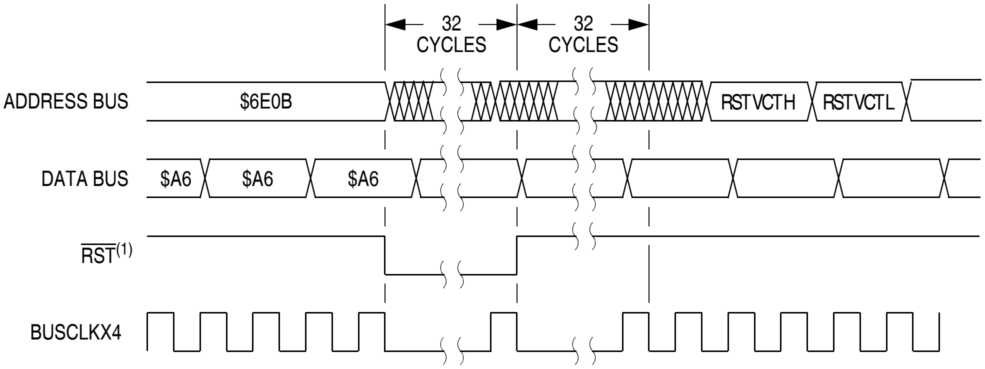
Wait mode can also be exited by a reset (or break in emulation mode). A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the break status register (BSR). If the COP disable bit, COPD, in the configuration register is 0, then the computer operating properly module (COP) is enabled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

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



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

Figure 13-15. Wait Recovery from Interrupt



1. $\overline{\text{RST}}$ is only available if the RSTEN bit in the CONFIG1 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, BUSCLKX4 will remain running in STOP and can be used to run the AWU. Stop recovery time is selectable using the SSREC bit in the configuration register 1 (CONFIG1). If SSREC is set, stop recovery is reduced from the normal delay of 4096 BUSCLKX4 cycles down to 32. This is ideal for the internal oscillator, RC oscillator, and external oscillator options which do not require long start-up times from stop mode.

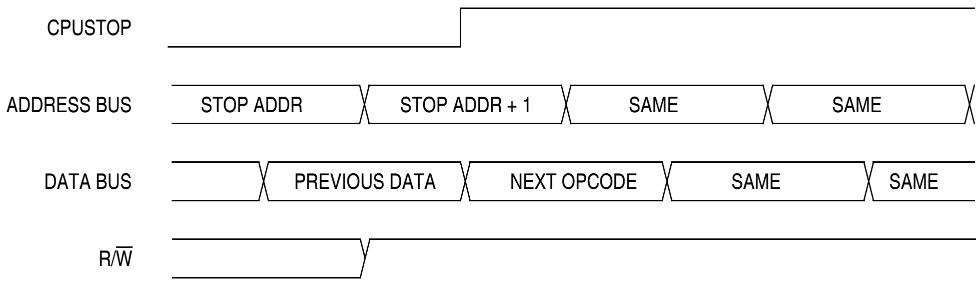
NOTE

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

The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. Figure 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

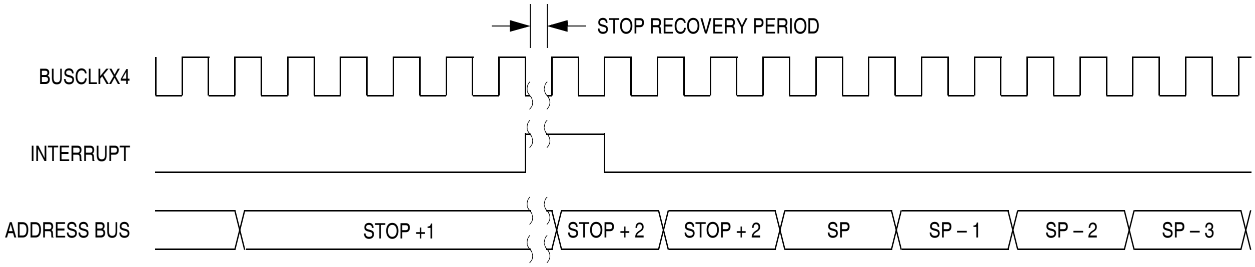


Figure 13-18. Stop Mode Recovery from Interrupt

13.8 SIM Registers

The SIM has two memory mapped registers.

13.8.1 SIM Reset Status Register

This register contains seven flags that show the source of the last reset. Clear the SIM reset status register by reading it. A power-on reset sets the POR bit and clears all other bits in the register.

	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 ($\overline{\text{RST}}$)
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 IRQB = V_{DD}
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.

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

R	= Reserved

Figure 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
Slave LIN Interface Controller (SLIC) Module

14.1 Introduction

The slave LIN interface controller (SLIC) is designed to provide slave node connectivity on a local interconnect network (LIN) sub-bus. LIN is an open-standard serial protocol developed for the automotive industry to connect sensors, motors, and actuators.

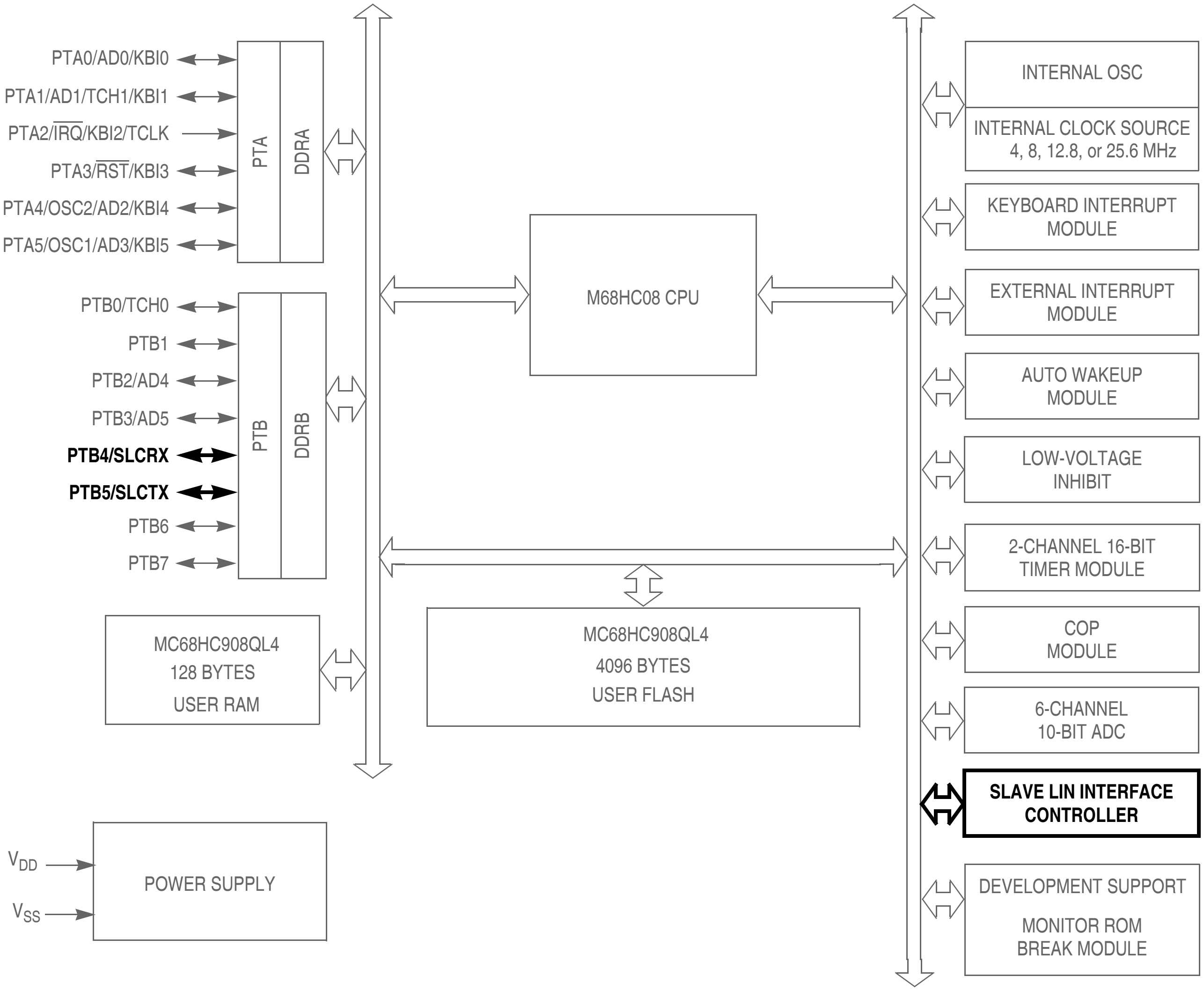
The SLIC shares its pins with general-purpose input/output (I/O) port pins. See Figure 14-1 for port location of these shared pins.

14.2 Features

The SLIC includes these distinctive features:

- Full LIN message buffering of identifier and 8 data bytes
- Automatic bit rate and LIN message frame synchronization:
  - No prior programming of bit rate required, 1–20 kbps LIN bus speed operation
  - All LIN messages will be received (no message loss due to synchronization process)
  - Input clock tolerance as high as ±50%, allowing internal oscillator to remain untrimmed
  - Incoming break symbols always allowed to be 10 or more bit times without message loss
  - Supports automatic software trimming of internal oscillator using LIN synchronization data
- Automatic processing and verification of LIN SYNCH BREAK and SYNCH BYTE
- Automatic checksum calculation and verification with error reporting
- Maximum of two interrupts per standard LIN message frame with no errors
- Full LIN error checking and reporting
- High-speed LIN capability up to 83.33 kbps to 120.00 kbps[1]
- Configurable digital receive filter
- Streamlined interrupt servicing through use of a state vector register
- Switchable UART-like byte transfer mode for processing bytes one at a time without LIN message framing constraints
- Enhanced checksum (includes ID) generation and verification

[1] Maximum bit rate of SLIC module dependent upon frequency of SLIC input clock.



$\overline{\text{RST}}$, $\overline{\text{IRQ}}$: Pins have internal pull up device
All port pins have programmable pull up device (pullup/down on port A)
PTA[0:5]: Higher current sink and source capability

Figure 14-1. Block Diagram Highlighting SLIC Block and Pins

14.3 Functional Description

The SLIC provides full standard LIN message buffering for a slave node, minimizing the need for CPU intervention. Routine protocol functions (such as synchronization to the communication channel, reception, and verification of header data) and generation of the checksum are handled automatically by the SLIC. This allows application software to be greatly simplified relative to standard UART implementations, as well as reducing the impact of interrupts needed in those applications to handle each byte of a message independently.

Additionally, the SLIC has the ability to automatically synchronize to any LIN message, regardless of the LIN bus bit rate (1–20 kbps), properly receiving that message without prior programming of the target LIN bit rate. Furthermore, this can even be accomplished using an untrimmed internal oscillator, provided its accuracy is at least ±50% of nominal.

The SLIC also has a simple UART-like byte transfer mode, which allows the user to send and receive single bytes of data in half-duplex 8-N-1 format (8-bit data, no parity, 1 stop bit) without the need for LIN message framing.

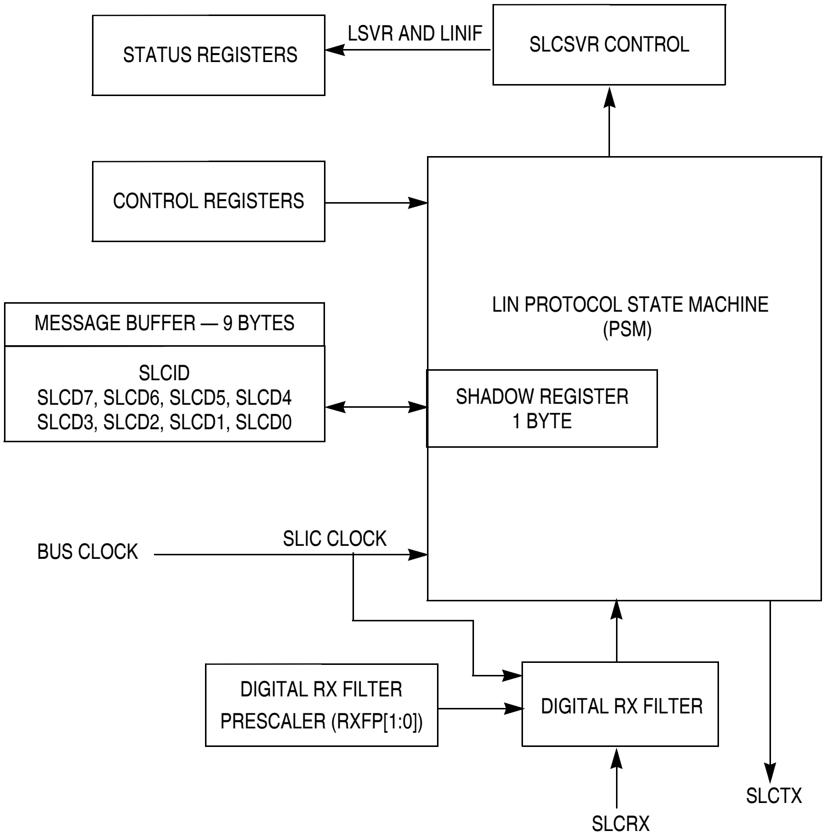


Figure 14-2. SLIC Module Block Diagram

14.4 Interrupts

The SLIC module contains one interrupt vector, which can be triggered by sources encoded in the SLIC state vector register. See 14.8.6 SLIC State Vector Register.

14.5 Modes of Operation

Figure 14-3 shows the modes in which the SLIC will operate.

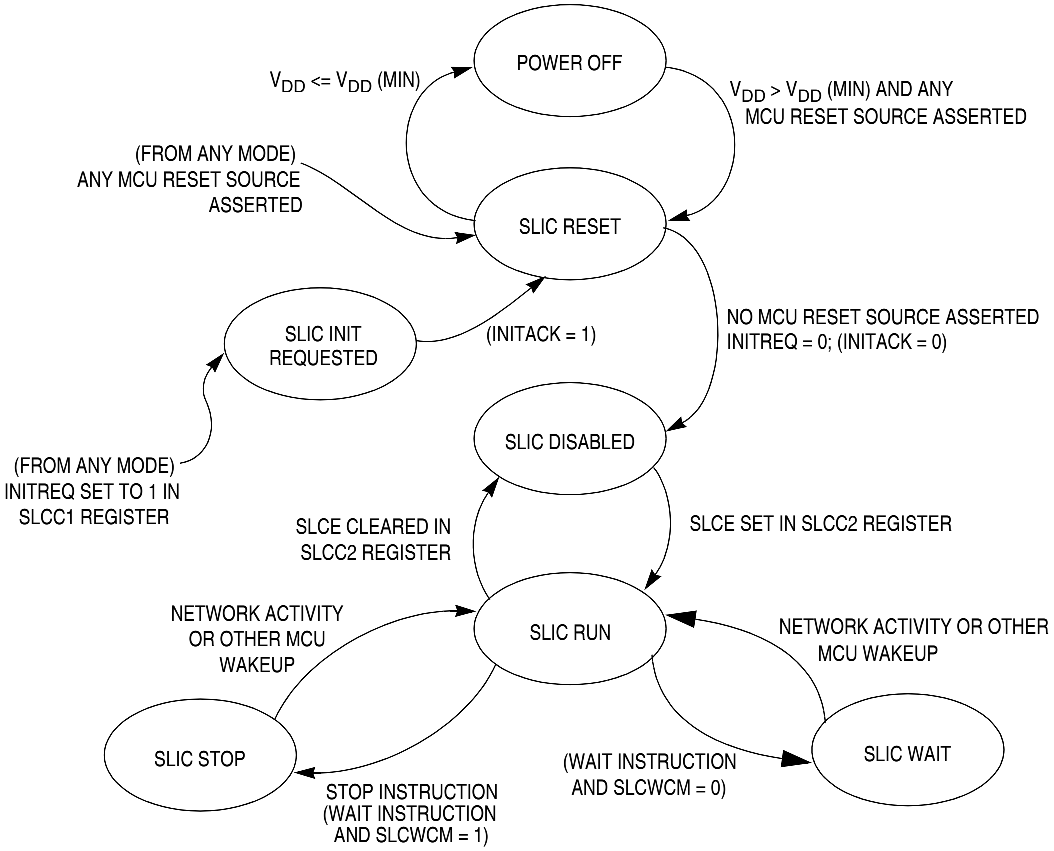

POWER OFF
V_{DD} <= V_{DD} (MIN)
V_{DD} > V_{DD} (MIN) AND ANY MCU RESET SOURCE ASSERTED
(FROM ANY MODE) ANY MCU RESET SOURCE ASSERTED
SLIC RESET
SLIC INIT REQUESTED
(INITACK = 1)
NO MCU RESET SOURCE ASSERTED INITREQ = 0; (INITACK = 0)
SLIC DISABLED
(FROM ANY MODE) INITREQ SET TO 1 IN SLCC1 REGISTER
SLCE CLEARED IN SLCC2 REGISTER
SLCE SET IN SLCC2 REGISTER
NETWORK ACTIVITY OR OTHER MCU WAKEUP
SLIC RUN
NETWORK ACTIVITY OR OTHER MCU WAKEUP
SLIC STOP
SLIC WAIT
(WAIT INSTRUCTION AND SLCWCM = 0)
STOP INSTRUCTION (WAIT INSTRUCTION AND SLCWCM = 1)

Figure 14-3. SLIC Operating Modes

14.5.1 Power Off

This mode is entered from the reset mode whenever the SLIC module supply voltage V_{DD} drops below its minimum specified value for the SLIC module to guarantee operation. The SLIC module will be placed in the reset mode by a system low-voltage reset (LVR) before being powered down. In this mode, the pin input and output specifications are not guaranteed.

14.5.2 Reset

This mode is entered from the power off mode whenever the SLIC module supply voltage V_{DD} rises above its minimum specified value ($V_{DD(MIN)}$) and some MCU reset source is asserted. To prevent the SLIC from entering an unknown state, the internal MCU reset is asserted while powering up the SLIC module. SLIC reset mode is also entered from any other mode as soon as one of the MCU's possible reset sources (e.g., LVR, POR, COP watchdog, $\overline{\text{RST}}$ pin, etc.) is asserted. SLIC reset mode may also be entered by the user software by asserting the INITREQ bit. INITACK indicates whether the SLIC module is in the reset mode as a result of writing INITREQ in SLCC1. While in the reset state the SLIC module clocks are stopped. Clearing the INITREQ allows the SLIC to proceed and enter SLIC run mode (if SLCE is set). The module will clear INITACK after the module has left reset mode and the SLIC will seek the next LIN header. It is the responsibility of the user to verify that this operation is compatible with the application before implementing this feature.

In this mode, the internal SLIC module voltage references are operative, V_{DD} is supplied to the internal circuits, which are held in their reset state and the internal SLIC module system clock is running. Registers will assume their reset condition. Outputs are held in their programmed reset state, inputs and network activity are ignored.

14.5.3 SLIC Disabled

This mode is entered from the reset mode after all MCU reset sources are no longer asserted or INITREQ is cleared by the user and the SLIC module clears INITACK. It is entered from the run mode whenever SLCE in SLCC2 is cleared. In this mode the SLIC clock is stopped to conserve power and allow the SLIC module to be configured for proper operation on the LIN bus.

14.5.4 SLIC Run

This mode is entered from the SLIC disabled mode when SLCE in SLCC2 is set. It is entered from the SLIC wait mode whenever activity is sensed on the LIN bus or some other MCU source wakes the CPU out of wait mode.

It is entered from the SLIC stop mode whenever network activity is sensed or some other MCU source wakes the CPU out of stop mode. Messages will not be received properly until the clocks have stabilized and the CPU is also in the run mode.

14.5.5 SLIC Wait

This power conserving mode is automatically entered from the run mode whenever the CPU executes a WAIT instruction and SLCWCM in SLCC1 is previously cleared. In this mode, the SLIC module internal clocks continue to run. Any activity on the LIN network will cause the SLIC module to exit SLIC wait mode and return to SLIC run.

14.5.6 Wakeup from SLIC Wait with CPU in WAIT

If the CPU executes the WAIT instruction and the SLIC module enters the wait mode (SLCWCM = 0), the clocks to the SLIC module as well as the clocks in the MCU continue to run. Therefore, the message that wakes up the SLIC module from WAIT and the CPU from wait mode will also be received correctly by the SLIC module. This is because all of the required clocks continue to run in the SLIC module in wait mode.

14.5.7 SLIC Stop

This power conserving mode is automatically entered from the run mode whenever the CPU executes a STOP instruction, or if the CPU executes a WAIT instruction and SLCWCM in SLCC1 is previously set. In this mode, the SLIC internal clocks are stopped. Any activity on the network will cause the SLIC module to exit SLIC stop mode and generate an unmaskable interrupt of the CPU. This wakeup interrupt state is reflected in the SLCSV, encoded as the highest priority interrupt. This interrupt can be cleared by the CPU with a read of the SLCSV and clearing of the SLCF interrupt flag. Depending upon which low-power mode instruction the CPU executes to cause the SLIC module to enter SLIC stop, the message which wakes up the SLIC module (and the CPU) may or may not be received.

There are two different possibilities:

1. Wakeup from SLIC Stop with CPU in STOP

   When the CPU executes the STOP instruction, all clocks in the MCU, including clocks to the SLIC module, are turned off. Therefore, the message which wakes up the SLIC module and the CPU from stop mode will not be received. This is due primarily to the amount of time required for the MCU's oscillator to stabilize before the clocks can be applied internally to the other MCU modules, including the SLIC module.

2. Wakeup from SLIC Stop with CPU in WAIT

   If the CPU executes the WAIT instruction and the SLIC module enters the stop mode (SLCWCM = 1), the clocks to the SLIC module are turned off, but the clocks in the MCU continue to run. Therefore, the message which wakes up the SLIC module from stop and the CPU from wait mode will be received correctly by the SLIC module. This is because very little time is required for the CPU to turn the clocks to the SLIC module back on after the wakeup interrupt occurs.

NOTE

While the SLIC module will correctly receive a message which arrives when the SLIC module is in stop or wait mode and the MCU is in wait mode, if the user enters this mode while a message is being received, the data in the message will become corrupted. This is due to the steps required for the SLIC module to resume operation upon exiting stop or wait mode, and its subsequent resynchronization with the LIN bus.

14.5.8 Normal and Emulation Mode Operation

The SLIC module operates in the same manner in all normal and emulation modes. All SLIC module registers can be read and written except those that are reserved, unimplemented, or write once. The user must be careful not to unintentionally write a register when using 16-bit writes to avoid unexpected SLIC module behavior.

14.5.9 Special Mode Operation

Some aspects of SLIC module operation can be modified in special test mode. This mode is reserved for internal use only.

14.5.10 Low-Power Options

The SLIC module can save power in disabled, wait, and stop modes. A complete description of what the SLIC module does while in a low-power mode can be found in 14.5 Modes of Operation.

14.6 SLIC During Break Interrupts

The BCFE bit in the BSCR register has no affect on the SLIC module. Therefore the SLIC modules status bits cannot be protected during break.

14.7 I/O Signals

The SLIC module can share its two pins with the general-purpose I/O pins. See Figure 14-1 for the port pins that are shared.

14.7.1 SLCTX — SLIC Transmit Pin

The SLCTX pin serves as the serial output of the SLIC module.

14.7.2 SLCRX — SLIC Receive Pin

The SLCRX pin serves as the serial input of the SLIC module. This input feeds directly into the digital receive filter block which filters out noise glitches from the incoming data stream.

14.8 Registers

14.8.1 SLIC Control Register 1

SLIC control register 1 (SLCC1) contains bits used to control various basic features of the SLIC module, including features used for initialization and at runtime.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	INITREQ	0	WAKETX	TXABRT	IMSG	SLCIE
Write:	▨	▨		▨				
Reset:	0	0	1	0	0	0	0	0

▨ = Unimplemented

Figure 14-4. SLIC Control Register 1 (SLCC1)

INITREQ —Initialization Request

Requesting initialization mode by setting this bit will place the SLIC module into its initialized state immediately. If transmission or reception of data is in progress, the transaction will be terminated immediately upon entry into initialization mode (signified by INITACK being set to 1). To return to normal SLIC operation after the SLIC has been initialized (the INITACK is high), the INITREQ must be cleared by software.

1 = Request for SLIC to be put into reset state immediately
0 = Normal operation

WAKETX — Transmit Wakeup Symbol

This bit allows the user to transmit a wakeup symbol on the LIN bus. When set, this sends a wakeup symbol, as defined in the LIN specification a single time, then resets to 0. This bit will read 1 while the wakeup symbol is being transmitted on the bus. This bit will be automatically cleared when the wakeup symbol is complete.

1 = Send wakeup symbol on LIN bus
0 = Normal operation

TXABRT — Transmit Abort Message

1 = Transmitter aborts current transmission at next byte boundary; TXABRT resets to 0 after the transmission is successfully aborted
0 = Normal operation

IMSG — SLIC Ignore Message Bit

IMSG cannot be cleared by a write of 0, but is cleared automatically by the SLIC module after the next BREAK/SYNC symbol pair is validated. After it is set, IMSG will not keep data from being written to the receive data buffer, which means that the buffers cannot be assumed to contain known valid message data until the next receive buffer full interrupt. IMSG must not be used in BTM mode.

1 = SLIC to ignore data field of message, SLIC interrupts are suppressed until the next message header arrives
0 = Normal operation

SLCIE — SLIC Interrupt Enable

1 = SLIC interrupt sources are enabled
0 = SLIC interrupt sources are disabled

14.8.2 SLIC Control Register 2

SLIC control register 2 (SLCC2) contains bits used to control various features of the SLIC module.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	SLCWCM	BTM	0	SLCE
Write:	▨	▨	▨	▨			▨	
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-5. SLIC Control Register 2 (SLCC2)

SLCWCM — SLIC Wait Clock Mode

This bit can only be written once out of reset state.

1 = SLIC clocks stop when the CPU is placed into wait mode
0 = SLIC clocks continue to run when the CPU is placed into wait mode so that the SLIC can receive messages and wakeup the CPU.

BTM — UART Byte Transfer Mode

Byte transmit mode bypasses the normal LIN message framing and checksum monitoring and allows the user to send and receive single bytes in a method similar to a half-duplex UART. When enabled, this mode reads the bit time register (SLCBT) value and assumes this is the value corresponding to the number of SLIC clock counts for one bit time to establish the desired UART bit rate. The user software must initialize this register prior to sending or receiving data, based on the input clock selection, prescaler stage choice, and desired bit rate.

BTM forces the data length in SLCDLC to one byte (DLC = 0x00) and disables the checksum circuitry so that CHKMOD has no effect. Refer to 14.9.15 Byte Transfer Mode Operation for more detailed information about how to use this mode. BTM sets up the SLIC module to send and receive one byte at a time, with 8-bit data, no parity, and one stop bit (8-N-1). This is the most commonly used setup for UART communications and should work for most applications. This is fixed in the SLIC and is not configurable.

1 = UART byte transfer mode enabled
0 = UART byte transfer mode disabled

SLCE — SLIC Module Enable

1 = SLIC module enabled
0 = SLIC module disabled

NOTE

To guarantee timing, the user must ensure that the SLIC clock used allows the proper communications timing tolerances and therefore internal oscillator circuits might not be appropriate for use with BTM mode.

14.8.3 SLIC Status Register

SLIC status register (SLCS) contains bits used to monitor the status of the SLIC module.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SLCACT	0	INITACK	0	0	0	0	SLCF
Write:	▨	▨	▨	▨	▨	▨	▨	
Reset:	0	0	1	0	0	0	0	0

▨ = Unimplemented

Figure 14-6. SLIC Status Register (SLCS)

SLCACT — SLIC Active (Oscillator Trim Blocking Semaphore)

SLCACT is used to indicate if it is safe to trim the oscillator based upon current SLIC activity in LIN mode. This bit indicates that the SLIC module might be currently receiving a message header, synchronization byte, ID byte, or sending or receiving data bytes. This bit is read-only.

1 = SLIC module activity (not safe to trim oscillator)

SLCACT is automatically set to 1 if a falling edge is seen on the SLCRX pin and has successfully been passed through the digital RX filter. This edge is the potential beginning of a LIN message frame.

0 = SLIC module not active (safe to trim oscillator)

SLCACT is cleared by the SLIC module only upon assertion of the RX Message Buffer Full Checksum OK (SLCSV = $10) or the TX Message Buffer Empty Checksum Transmitted (SLCSV = $08) interrupt sources.

NOTE

SLCACT may not be clear during all idle times of the bus. For example, if IMSG was used to ignore the data interrupts of an extended message frame, SLCACT will remain set until another LIN message is received and either the RX Message Buffer Full Checksum OK (SLCSV = $10) or the TX Message Buffer Empty Checksum Transmitted (SLCSV = $08) interrupt sources are asserted and cleared. When clear, SLCACT always indicates times when the SLIC module is not active, but it is possible for the SLIC module to be not active with SLCACT set. SLCACT has no meaning in BTM mode.

INITACK — Initialization Mode Acknowledge

INITACK indicates whether the SLIC module is in the reset mode as a result of writing INITREQ in SLCC1. While in the reset state the SLIC module clocks are stopped. Clearing the INITREQ allows the SLIC to proceed and enter SLIC run mode (if SLCE is set). The module will clear INITACK after the module has left reset mode and the SLIC will seek the next LIN header. This bit is read-only.

1 = SLIC module is in reset state

0 = Normal operation

SLCF — SLIC Interrupt Flag

The SLCF interrupt flag indicates if a SLIC module interrupt is pending. If set, the SLCSV is then used to determine what interrupt is pending. This flag is cleared by writing a 1 to the bit. If additional interrupt sources are pending, the bit will be automatically set to 1 again by the SLIC.

1 = SLIC interrupt pending
0 = No SLIC interrupt pending

14.8.4 SLIC Prescaler Register

SLIC prescaler register (SLCP) is used to configure the delay of the digital receive filter circuit, and hence the width of noise pulse which is blocked by the filter. The SLIC clock is used to drive the digital receive circuit. Variations on the input clock will affect filter performance proportionally, so if internal oscillators are used, worst case oscillator frequencies must be accounted for when determining prescaler settings to ensure that the frequency of the SLIC clock always remains between 2 MHz and 8 MHz.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	RXFP1	RXFP0	0	0	0	0	0	0
Write:			▨	▨	▨	▨	▨	▨
Reset:	1	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-7. SLIC Prescale Register (SLCP)

Table 14-1. Digital Receive Filter Clock Prescaler

RXFP[1:0]	Digital RX Filter Clock Prescaler (Divide by)	Maximum Filter Delay (in µs)					
		SLIC Clock (in MHz)					
		2	3.2	4	6	6.4	8
$00	1	8	5	4	2.67	2.5	2
$01	2	16	10	8	5.33	5	4
$10	3 (default)	24	15	12	8	7.5	6
$11	4	32	20	16	10.67	10	8

RXFP[1:0] — Receive Filter Prescaler

These bits configure the effective filter width for the digital receive filter circuit. The RXFP bits control the maximum number of SLIC clock counts required for the filter to change state, which determines the total maximum filter delay. Any pulse which is smaller than the maximum filter delay value will be rejected by the filter and ignored as noise. For this reason, the user must choose the prescaler value appropriately to ensure that all valid message traffic is able to pass the filter for the desired bit rate. For more details about setting up the digital receive filter, please refer to 14.9.17 Digital Receive Filter.

The frequency of the SLIC clock must be between 2 MHz and 8 MHz, factoring in worst case possible numbers due to untrimmed process variations, as well as temperature and voltage variations in oscillator frequency. This will guarantee greater than 1.5% accuracy for all LIN messages from 1–20 kbps. The faster this input clock is, the greater the resulting accuracy and the higher the possible bit rates at which the SLIC can send and receive. In LIN systems, the bit rates will not exceed 20 kbps; however, the SLIC module is capable of much higher speeds without any configuration changes, for cases such as high-speed downloads for reprogramming of FLASH memory or diagnostics in a test environment where radiated emissions requirements are not as stringent. In these situations, the user may choose to run faster than the 20 kbps limit which is imposed by the LIN specification for EMC reasons. Details of how to calculate maximum bit rates and operate the SLIC above 20 kbps are detailed in 14.9.14 High-Speed LIN Operation. Refer to 14.9.6 SLIC Module Initialization Procedure for more information on when to set up this register.

14.8.5 SLIC Bit Time Registers

NOTE

In this subsection, the SLIC bit time registers are collectively referred to as SLCBT.

In LIN operating mode (BTM = 0), the SLCBT is updated by the SLIC upon reception of a LIN break-synch combination and provides the number of SLIC clock counts that equal one LIN bit time to the user software. This value can be used by the software to calculate the clock drift in the oscillator as an offset to a known count value (based on nominal oscillator frequency and LIN bus speed). The user software can then trim the oscillator to compensate for clock drift. Refer to 14.9.16 Oscillator Trimming with SLIC for more information on this procedure. The user cannot read the bit time value from SLCBTH and SLCBTL any time after the identifier byte is received until the beginning of the next LIN message frame on the bus. The beginning of this message frame activity will be indicated by SLCACT.

When in byte transfer mode (BTM = 1), the SLCBT must be written by the user to set the length of one bit at the desired bit rate in SLIC clock counts. The user software must initialize this number prior to sending or receiving data, based on the input clock selection, prescaler stage choice, and desired bit rate. This setting is similar to choosing an input capture or output compare value for a timer. The closest even value must be chosen for this value, as BT0 will be forced to 0 by the SLIC module and any odd value will always be reduced to the next lowest even integer value. For example, a write of value 51 (0x33) will be forced to value of 50 and read back as 0x32. A write to both registers is required to update the bit time value.

NOTE

The SLIC bit time will not be updated until a write of the SLCBTL has occurred.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	BT12	BT11	BT10	BT9	BT8
Write:	▨	▨	▨					
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-8. SLIC Bit Time Register High (SLCBTH)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	BT7	BT6	BT5	BT4	BT3	BT2	BT1	0
Write:								▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-9. SLIC Bit Time Register Low (SLCBTL)

BT — Bit Time Value

BT displays the number of SLIC clocks that equals one bit time in LIN mode (BTM = 0). For details of the use of the SLCBT registers in LIN mode for trimming of the internal oscillator, refer to 14.9.16 Oscillator Trimming with SLIC.

BT sets the number of SLIC clocks that equals one bit time in byte transfer mode (BTM = 1). For details of the use of the SLCBT registers in BTM mode, refer to 14.9.15 Byte Transfer Mode Operation.

NOTE

Do not write to unimplemented bits as unexpected operation may occur.

14.8.6 SLIC State Vector Register

SLIC state vector register (SLCSV) is provided to substantially decrease the CPU overhead associated with servicing interrupts while under operation of a LIN protocol. It provides an index offset that is directly related to the LIN module's current state, which can be used with a user supplied jump table to rapidly enter an interrupt service routine. This eliminates the need for the user to maintain a duplicate state machine in software.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	I3	I2	I1	I0	0	0
Write:	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-10. SLIC State Vector Register (SLCSV)

READ: any time
WRITE: ignored

I[3:0] — Interrupt State Vector (Bits 5–2)

These bits indicate the source of the interrupt request that is currently pending.

14.8.6.1 LIN Mode Operation

Table 14-2 shows the possible values for the possible sources for a SLIC interrupt while in LIN mode operation (BTM = 0).

Table 14-2. Interrupt Sources Summary (BTM = 0)

SLCSV	I3	I2	I1	I0	Interrupt Source	Priority
$00	0	0	0	0	No Interrupts Pending	0 (Lowest)
$04	0	0	0	1	No-Bus-Activity	1
$08	0	0	1	0	TX Message Buffer Empty Checksum Transmitted	2
$0C	0	0	1	1	TX Message Buffer Empty	3
$10	0	1	0	0	RX Message Buffer Full Checksum OK	4
$14	0	1	0	1	RX Data Buffer Full No Errors	5
$18	0	1	1	0	Bit-Error	6

Table 14-2. Interrupt Sources Summary (BTM = 0) (Continued)

SLCSV	I3	I2	I1	I0	Interrupt Source	Priority
$1C	0	1	1	1	Receiver Buffer Overrun	7
$20	1	0	0	0	Reserved	8
$24	1	0	0	1	Checksum Error	9
$28	1	0	1	0	Byte Framing Error	10
$2C	1	0	1	1	Identifier Received Successfully	11
$30	1	1	0	0	Identifier Parity Error	12
$34	1	1	0	1	Inconsistent-Synch-Field-Error	13
$38	1	1	1	0	Reserved	14
$3C	1	1	1	1	Wakeup	15 (Highest)

- No Interrupts Pending
  This value indicates that all pending interrupt sources have been serviced. In polling mode, the SLCSV is read and interrupts serviced until this value reads back 0. This source will not generate an interrupt of the CPU, regardless of state of SLCIE.
- No Bus Activity (LIN specified error)
  The No-Bus-Activity condition occurs if no valid SYNCH BREAK FIELD or BYTE FIELD was received for more than 2^{23} SLIC clock counts since the reception of the last valid message. For example, with 6.4 MHz SLIC clock frequency, a No-Bus-Activity interrupt will occur about 1.31 seconds after the bus begins to idle.
- TX Message Buffer Empty — Checksum Transmitted
  When the entire LIN message frame has been transmitted successfully, complete with the appropriately selected checksum byte, this interrupt source is asserted. This source is used for all standard LIN message frames and the final set of bytes with extended LIN message frames.
- TX Message Buffer Empty
  This interrupt source indicates that all 8 bytes in the LIN message buffer have been transmitted with no checksum appended. This source is used for intermediate sets of 8 bytes in extended LIN message frames.
- RX Message Buffer Full — Checksum OK
  When the entire LIN message frame has been received successfully, complete with the appropriately selected checksum byte, and the checksum calculates correctly, this interrupt source is asserted. This source is used for all standard LIN message frames and the final set of bytes with extended LIN message frames. To clear this source, SLCD0 must be read first.
- RX Data Buffer Full — No Errors
  This interrupt source indicates that 8 bytes have been received with no checksum byte and are waiting in the LIN message buffer. This source is used for intermediate sets of 8 bytes in extended LIN message frames. To clear this source, SLCD0 must be read first.
- Bit Error
  A unit that is sending a bit on the bus also monitors the bus. A BIT_ERROR must be detected at that bit time, when the bit value that is monitored is different from the bit value that is sent. The SLIC will terminate the data transmission upon detection of a bit error, according to the LIN

specification. Bit errors are not checked when the LIN bus is running at high speed due to the effects of physical layer round trip delay. Bit errors are fully checked at all LIN 2.0 compliant speeds of 20 kbps and below.

- Receiver Buffer Overrun Error
  This error is an indication that the receive buffer has not been emptied and additional bytes have been received, resulting in lost data. Because this interrupt is higher priority than the receive buffer full interrupts, it will appear first when an overflow condition occurs. There will, however, be a pending receive interrupt which must also be cleared after the buffer overrun flag is cleared.
- Checksum Error (LIN specified error)
  The checksum error occurs when the calculated checksum value does not match the expected value. If this error is encountered, it is important to verify that the correct checksum calculation method was employed for this message frame. Refer to the LIN specification for more details on the calculations.
- Byte Framing Error
  This error comes from the standard UART definition for byte encoding and occurs when the STOP bit is sampled and reads back as a 0. STOP should always read as 1. In LIN mode (BTM=0), if a byte framing error occurs in an identifier byte of a LIN header the user must set and then clear CHKMOD to ensure that the checksum calculation is reset. Failure to do so can result in an improperly calculated enhanced checksum for the subsequent LIN frame. Because any byte framing error indicates a corrupted byte, the best practice is to always toggle CHKMOD in the case of a byte framing error.

NOTE

A byte framing error can also be an indication that the number of data bytes received in a LIN message frame does not match the value written to the SLCDLC register. See 14.9.7 Handling LIN Message Headers for more details.

- Identifier Received Successfully
  This interrupt source indicates that a LIN identifier byte has been received with correct parity and is waiting in the LIN identifier buffer (SLCID). Upon reading this interrupt source from SLCSV, the user can then decode the identifier in software to determine the nature of the LIN message frame. To clear this source, SLCID must be read.
- Identifier-Parity-Error
  A parity error in the identifier (i.e., corrupted identifier) will be flagged. Typical LIN slave applications do not distinguish between an unknown but valid identifier, and a corrupted identifier. However, it is mandatory for all slave nodes to evaluate in case of a known identifier all eight bits of the ID-Field and distinguish between a known and a corrupted identifier. The received identifier value is reported in SLCID so that the user software can choose to acknowledge or ignore the parity error message.
- Inconsistent-Synch-Field-Error
  An Inconsistent-Synch-Field-Error must be detected if a slave detects the edges of the SYNCH FIELD outside the given tolerance.
- Wakeup
  The wakeup interrupt source indicates that the SLIC module has entered SLIC run mode from SLIC stop mode.

14.8.6.2 Byte Transfer Mode Operation

When byte transfer mode is enabled (BTM = 1), many of the interrupt sources for the SLCSV no longer apply, as they are specific to LIN operations. Table 14-3 shows those interrupt sources which are applicable to BTM operations. The value of the SLCSV for each interrupt source remains the same, as well as the priority of the interrupt source.

Table 14-3. Interrupt Sources Summary (BTM = 1)

SLCSV	I3	I2	I1	I0	Interrupt Source	Priority
$00	0	0	0	0	No Interrupts Pending	0 (Lowest)
$0C	0	0	1	1	TX Message Buffer Empty	3
$14	0	1	0	1	RX Data Buffer Full No Errors	5
$18	0	1	1	0	Bit-Error	6
$1C	0	1	1	1	Receiver Buffer Overrun	7
$28	1	0	1	0	Byte Framing Error	10
$38	1	1	1	0	Reserved	14
$3C	1	1	1	1	Wakeup	15 (Highest)

- No Interrupts Pending
  This value indicates that all pending interrupt sources have been serviced. In polling mode, the SLCSV is read and interrupts serviced until this value reads back 0. This source will not generate an interrupt of the CPU, regardless of state of SLCIE.
- TX Message Buffer Empty
  In byte transfer mode, this interrupt source indicates that the byte in the SLCID has been transmitted.
- RX Data Buffer Full — No Errors
  This interrupt source indicates that a byte has been received and is waiting in SLCID. To clear this source, SLCID must be read first.
- Bit-Error
  A unit that is sending a bit on the bus also monitors the bus. A BIT_ERROR must be detected at that bit time, when the bit value that is monitored is different from the bit value that is sent.
- Receiver Buffer Overrun Error
  This error is an indication that the receive buffer has not been emptied and additional byte(s) have been received, resulting in lost data. Because this interrupt is higher priority than the receive buffer full interrupts, it will appear first when an overflow condition occurs. There will, however, be a pending receive interrupt which must also be cleared after the buffer overrun flag is cleared.
- Byte Framing Error
  This error comes from the standard UART definition for byte encoding and occurs when STOP is sampled and reads back as a 0. STOP should always read as 1.
- Wakeup
  The wakeup interrupt source indicates that the SLIC module has entered SLIC run mode from SLIC wait mode.

14.8.7 SLIC Data Length Code Register

The SLIC data length code register (SLCDLC) is the primary functional control register for the SLIC module during normal LIN operations. It contains the data length code of the message buffer, indicating how many bytes of data are to be sent or received, as well as the checksum mode control and transmit enabling bit.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TXGO	CHKMOD	DLC5	DLC4	DLC3	DLC2	DLC1	DLC0
Reset:	0	0	0	0	0	0	0	0

Figure 14-11. SLIC Data Length Code Register (SLCDLC)

TXGO — SLIC Transmit Go

This bit controls whether the SLIC module is sending or receiving data bytes. This bit is automatically reset to 0 after a transmit operation is complete or an error is encountered and transmission has been aborted.

1 = Initiate SLIC transmit

The SLIC assumes the user has loaded the proper data into the message buffer and will begin transmitting the number of bytes indicated in the SLCDLC bits. If the number of bytes is greater than 8, the first 8 bytes will be transmitted and an interrupt will be triggered (if unmasked) for the user to enter the next bytes of the message. If the number of bytes is 8 or fewer, the SLIC will transmit the appropriate number of bytes and automatically append the checksum to the transmission.

0 = SLIC receive data

CHKMOD — LIN Checksum Mode

CHKMOD is used to decide what checksum method to use for this message frame. Resets after error code or message frame complete. CHKMOD must be written (if desired) only after the reception of an identifier and before the reception or transmission of data bytes. Writing this bit to a one clears the current checksum calculation. The one exception is when a byte framing error is detected and the checksum calculation should be reset. (See Byte Framing Error description in section 14.8.6.1 LIN Mode Operation.)

1 = Checksum calculated without the identifier byte (LIN spec <= 1.3)

0 = Checksum calculated with the identifier byte included
(SAE J2602/LIN 2.0)

DLC — Data Length Control Bits

The value of the bits indicate the number of data bytes in message. Values $00–$07 are for "normal" LIN messaging. Values $08–$3F are for "extended" LIN messaging.

Table 14-4. Data Length Control

DLC[5:0]	Message Data Length (Number of Bytes)
$00	1
$01	2
$02	3
...	...
$3D	62
$3E	63
$3F	64

14.8.8 SLIC Identifier and Data Registers

The SLIC identifier (SLCID) and eight data registers (SLCD7–SLCD0) comprise the transmit and receive buffer and are used to read/write the identifier and message buffer 8 data bytes. In BTM mode (BTM = 1), only SLCID is used to send and receive bytes, as only one byte is handled at any one time. The number of bytes to be read from or written to these registers is determined by the user software and written to SLCDLC. To obtain proper data, reads and writes to these registers must be made based on the proper length corresponding to a particular message. It is the responsibility of the user software to keep track of this value to prevent data corruption. For example, it is possible to read data from locations in the message buffer which contain erroneous or old data if the user software reads more data registers than were updated by the incoming message, as indicated in SLCDLC.

NOTE

An incorrect length value written to SLCDLC can result in the user software misreading or miswriting data in the message buffer. An incorrect length value might also result in SLIC error messages. For example, if a 4-byte message is to be received, but the user software incorrectly reports a 3-byte length to the DLC, the SLIC will assume the 4th data byte is actually a checksum value and attempt to validate it as such. If this value doesn't match the calculated value, an incorrect checksum error will occur. If it does happen to match the expected value, then the message would be received as a 3-byte message with valid checksum. Either case is incorrect behavior for the application and can be avoided by ensuring that the correct length code is used for each identifier.

The first data byte received after the LIN identifier in a LIN message frame will be loaded into SLCD0. The next byte (if applicable) will be loaded into SLCD1, and so forth.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R7	R6	R5	R4	R3	R2	R1	R0
Write:	T7	T6	T5	T4	T3	T2	T1	T0
Reset:	0	0	0	0	0	0	0	0

Figure 14-12. SLIC Identifier Register (SLCID)

The SLIC identifier register is used to capture the incoming LIN identifier and when the SLCSV value indicates that the identifier has been received successfully, this register contains the received identifier value. If the incoming identifier contained a parity error, this register value will not contain valid data.

In byte transfer mode (BTM = 1), this register is used for sending and receiving each byte of data. When transmitting bytes, the data is loaded into this register, then TXGO in SLCDLC is set to initiate the transmission. When receiving bytes, they are read from this register only.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R7	R6	R5	R4	R3	R2	R1	R0
Write:	T7	T6	T5	T4	T3	T2	T1	T0
Reset:	0	0	0	0	0	0	0	0

Figure 14-13. SLIC Data Register x (SLCD7–SLCD0)

R — Read SLC Receive Data

T — Write SLC Transmit Data

14.9 Initialization/Application Information

The LIN specification defines a standard LIN “MESSAGE FRAME” as the basic format for transferring data across a LIN network. A standard MESSAGE FRAME is composed as shown in Figure 14-14 (shown with 8 data bytes).

LIN transmits all data, identifier, and checksum characters as standard UART characters with eight data bits, no parity, and one stop bit. Therefore, each byte has a length of 10 bits, including the start and stop bits. The data bits are transmitted least significant bit (LSB) first.

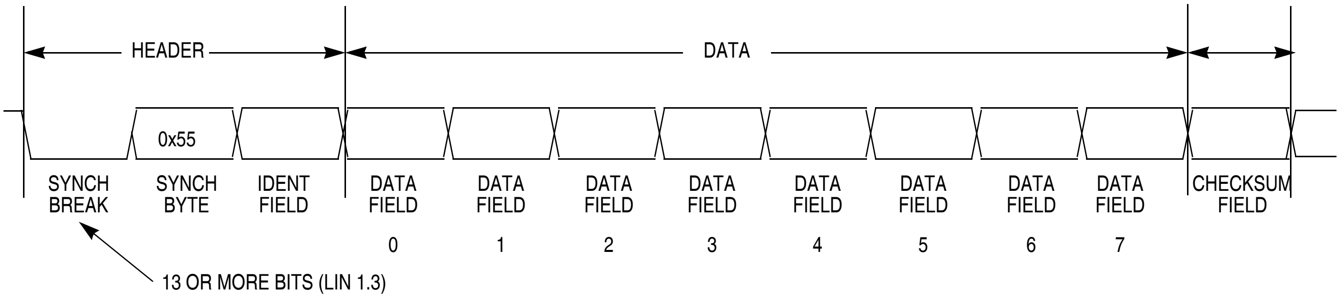


Figure 14-14. Typical LIN MESSAGE FRAME

14.9.1 LIN Message Frame Header

The HEADER section of all LIN messages is transmitted by the master node in the network and contains synchronization data, as well as the identifier to define what information is to be contained in the message frame. Formally, the header is comprised of three parts:

1. SYNCH BREAK
2. SYNCH BYTE (0x55)
3. IDENTIFIER FIELD

The first two components are present to allow the LIN slave nodes to recognize the beginning of the message frame and derive the bit rate of the master module.

The SYNCH BREAK allows the slave to see the beginning of a message frame on the bus. The SLIC module can receive a standard 10-bit break character for the SYNCH BREAK, or any break symbol 10 or more bit times in length. This encompasses the LIN requirement of 13 or more bits of length for the SYNCH BREAK character.

The SYNCH BYTE is always a 0x55 data byte, providing five falling edges for the slave to derive the bit rate of the master node.

The identifier byte indicates to the slave what is the nature of the data in the message frame. This data might be supplied from either the master node or the slave node, as determined at system design time. The slave node must read this identifier, check for parity errors, and determine whether it is to send or receive data in the data field.

More information on the HEADER is contained in 14.9.7.1 LIN Message Headers.

14.9.2 LIN Data Field

The data field is comprised of standard bytes (eight data bits, no parity, one stop bit) of data, from 0–8 bytes for normal LIN frames and greater than eight bytes for extended LIN frames. The SLIC module will either transmit or receive these bytes, depending upon the user code interpretation of the identifier byte. Data is always transmitted into the data field least significant byte (LSB) first.

The SLIC module can automatically handle up to 64 bytes in extended LIN message frames without significantly changing program execution.

14.9.3 LIN Checksum Field

The checksum field is a data integrity measure for LIN message frames, used to signal errors in data consistency. The LIN 1.3 checksum calculation only covers the data field, but the SLIC module also supports an enhanced checksum calculation which also includes the identifier field. For more information on the checksum calculation, refer to 14.9.13 LIN Data Integrity Checking Methods.

14.9.4 SLIC Module Constraints

In designing a practical module, certain reasonable constraints must be placed on the LIN message traffic which are not necessarily explicitly specified in the LIN specification. The SLIC module presumes that:

- Timeout for no-bus-activity = 1 second.

14.9.5 SLCSV Interrupt Handling

Each change of state of the SLIC module is encoded in the SLIC state vector register (SLCSV). This is an efficient method of handling state changes, indicating to the user not only the current status of the SLIC module, but each state change will also generate an interrupt (if SLIC interrupts are enabled). For more detailed information on the SLCSV, please refer to 14.8.6 SLIC State Vector Register.

In the software diagrams in the following subsections, when an interrupt is shown, the first step must always be reading SLCSV to determine what is the current status of the SLIC module. Likewise, when the diagrams indicate to “EXIT ISR”, the final step to exiting the interrupt service routine is to clear the SLCF interrupt flag. This can only be done if the SLCSV has first been read, and in the case that data has been received (such as an ID byte or command message data) the SLCD has been read at least one time.

After SLCSV is read, it will switch to the next pending state, so the user must be sure it is copied only once into a software variable at the beginning of the interrupt service routine to avoid inadvertently clearing a pending interrupt source. Additional decisions based on this value must be made from the software variable, rather than from the SLCSV itself.

After exiting the ISR, normal application code may resume. If the diagram indicates to “RETURN TO IDLE,” it indicates that all processing for the current message frame has been completed. If an error was detected and the corresponding error code loaded into the SLCSV, any pending data in the data buffer will be flushed out and the SLIC returned to its idle state, seeking out the next message frame header.

14.9.6 SLIC Module Initialization Procedure

14.9.6.1 LIN Mode Initialization

The SLIC module does not require very much initialization, due to its self-synchronizing design. Because no prior knowledge of the bit rate is required to synchronize to the LIN bus, no programming of bit rate is required.

At initialization time, the user must configure:

- SLIC prescale register (SLIC digital receive filter adjustment).
- Wait clock mode operation.

The SLIC clock is the same as the CPU bus clock. The module is designed to provide better than 1% bit rate accuracy at the lowest value of the SLIC clock frequency and the accuracy improves as the SLIC clock frequency is increased. For this reason, it is advantageous to choose the fastest SLIC clock which is still within the acceptable operating range of the SLIC. Because the SLIC may be used with MCUs with internal oscillators, the tolerance of the oscillator must be taken into account to ensure that SLIC clock frequency does not exceed the bounds of the SLIC clock operating range. This is especially important if the user wishes to use the oscillator untrimmed, where process variations might result in MCU frequency offsets of ±25%.

The acceptable range of SLIC clock frequencies is 2–8 MHz to guarantee LIN operations with greater than 1.5% accuracy across the 1–20 kbps range of LIN bit rates. The user must ensure that the fastest possible SLIC clock frequency never exceeds 8 MHz or that the slowest possible SLIC clock never falls below 2 MHz under worst case conditions. This would include, for example, oscillator frequency variations due to untrimmed oscillator tolerance, temperature variation, or supply voltage variation.

To initialize the SLIC module into LIN operating mode, the user must perform the following steps prior to needing to receive any LIN message traffic. These steps assume the MCU has been reset either by a power-on reset (POR) or any other MCU reset mechanism.

The steps for SLIC Initialization for LIN operation are:

1. Write SLCC1 to clear INITREQ.
2. When INITACK = 0, write SLCC1 & SLCC2 with desired values for:
   a. SLCWCM — Wait clock mode (default = leaving SLIC clock running when in CPU wait).
3. Write SLCP to set up prescalers for:
   a. RXFP — Digital receive filter clock prescaler (default = SLIC divided by 3).
4. Enable the SLIC module by writing SLCC2:
   a. SLCE = 1 to place SLIC module into run mode.
   b. BTM = 0 to disable byte transfer mode (default).
5. Write SLCC1 to enable SLIC interrupts (if desired).

14.9.6.2 Byte Transfer Mode Initialization

Bit rate synchronization is handled automatically in LIN mode, using the synchronization data contained in each LIN message to derive the desired bit rate. In byte transfer mode (BTM = 1); however, the user must set up the bit rate for communications using the clock selection, prescaler, and SLCBT.

More information on byte transfer mode is described in 14.9.15 Byte Transfer Mode Operation, including the performance parameters on recommended maximum speeds, bit time resolution, and oscillator tolerance requirements.

After the desired settings of prescalers and bit time are determined, the SLIC Initialization for BTM operation is virtually identical to that of LIN operation.

The steps are:

1. Write SLCC1 to clear INITREQ.
2. When INITACK = 0, write SLCC2 with desired values for:
   a. SLCWCM — Wait clock mode (default = leaving SLIC clock running when in CPU wait).
3. Write SLCICP to set up prescalers for:
   a. RXFP — Digital receive filter clock prescaler (default = SLIC divided by 3).

4. Enable the SLIC module by writing SLCC2:
   a. SLCE = 1 to place SLIC module into run mode.
   b. BTM = 1 to enable byte transfer mode (default = disabled).
5. Write SLCC1 to enable SLIC interrupts (if desired).

14.9.7 Handling LIN Message Headers

Figure 14-15 shows how the SLIC module deals with incoming LIN message headers.

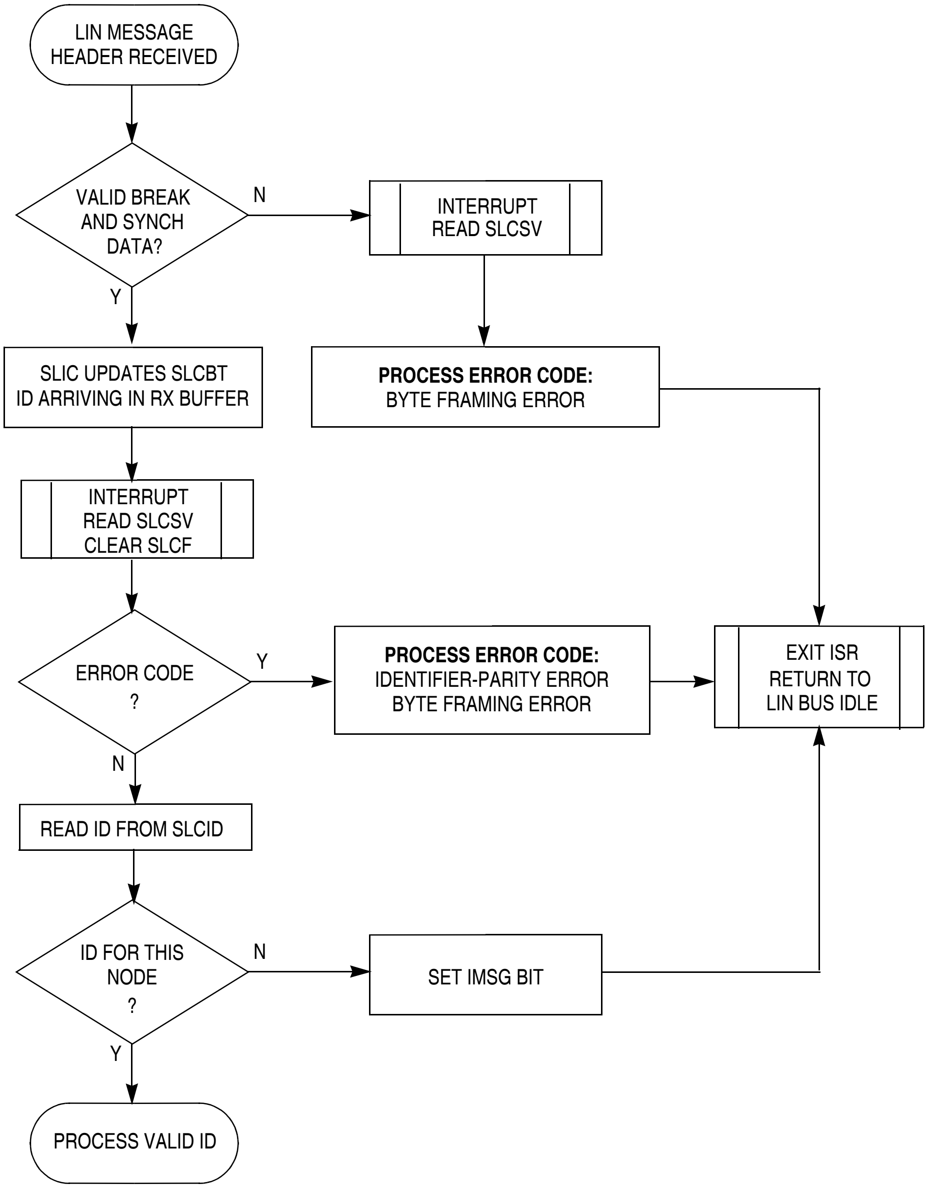

LIN MESSAGE HEADER RECEIVED
VALID BREAK AND SYNCH DATA?
N
Y
INTERRUPT READ SLCSV
SLIC UPDATES SLCBT ID ARRIVING IN RX BUFFER
PROCESS ERROR CODE: BYTE FRAMING ERROR
INTERRUPT READ SLCSV CLEAR SLCF
ERROR CODE ?
Y
N
PROCESS ERROR CODE: IDENTIFIER-PARITY ERROR BYTE FRAMING ERROR
EXIT ISR RETURN TO LIN BUS IDLE
READ ID FROM SLCID
ID FOR THIS NODE ?
N
Y
SET IMSG BIT
PROCESS VALID ID

Figure 14-15. Handling LIN Message Headers

14.9.7.1 LIN Message Headers

All LIN message frame headers are comprised of three components:

- The first is the SYNCHRONIZATION BREAK (SYNCH BREAK) symbol, which is a dominant (low) pulse at least 13 or more bit times long, followed by a recessive (high) synchronization delimiter of at least one bit time. In LIN 2.0, this is allowed to be 10 or more bit times in length.
- The second part is called the SYNCHRONIZATION FIELD (SYNCH FIELD) and is a single byte with value 0x55. This value was chosen as it is the only one which provides a series of five falling (recessive to dominant) transitions on the bus.
- The third section of the message frame header is the IDENTIFIER FIELD (ID). The identifier is covered more in 14.9.8 Handling Command Message Frames and 14.9.9 Handling Request LIN Message Frames.

The SLIC automatically reads the incoming pattern of the SYNCHRONIZATION BREAK and FIELD and determines the bit rate of the LIN data frame, as well as checking for errors in form and discerning between a genuine BREAK/FIELD combination and a similar byte pattern somewhere in the data stream. After the header has been verified to be valid and has been processed, the SLIC module updates the SLIC bit time register (SLCBT) with the value obtained from the SYNCH FIELD and begins to receive the ID.

If there are errors in the SYNCH BREAK/FIELD pattern, then an interrupt is generated. If unmasked, it will trigger an MCU interrupt request and the resulting code in the SLIC state vector register (SLCSV) will be an “Inconsistent-Synch-Field-Error,” based on the LIN protocol specification.

After the ID for the message frame has been received, an interrupt is generated by the SLIC and will trigger an MCU interrupt request if unmasked. At this point, it might be possible that the ID was received with errors such as a parity error (based on the LIN specification) or a byte framing error. If the ID did not have any errors, it will be copied into the SLCD for the software to read. The SLCSV will indicate the type error or that the ID was received correctly.

In a LIN system, the meaning and function of all messages, and therefore all message identifiers, is pre-defined by the system designer. This information can be collected and stored in a standardized format file, called a Configuration Language Description (CLD) file. In using the SLIC module, it is the responsibility of the user software to determine the nature of the incoming message, and therefore how to further handle that message.

The simplest case is when the SLIC receives a message which the user software determines is of no interest to the application. In other words, the slave node does not need to receive or transmit any data for this message frame. This might also apply to messages with zero data bytes (which is allowed by the LIN specification). At this point, the user can set the IMSG control bit, and exit the interrupt service routine by clearing the SLCIF flag. Because there is no data to be sent or received, the SLIC will not generate another interrupt until the next message frame header or bus goes idle long enough to trigger a “No-Bus-Activity” error according to the LIN specification.

NOTE

IMSG will prevent another interrupt from occurring for the current message frame; however, if data bytes are appearing on the bus they may be received and copied into the message buffer. This will delete any previous data which might have been present in the buffer, even though no interrupt is triggered to indicate the arrival of this data.

At the time the ID is read, the user might also choose to read SLCBT and copy this value out to an application variable. This data can then be used at a time appropriate to both the application software and the LIN communications to adjust the trim of the internal oscillator. This operation must be handled very carefully to avoid adjusting the base timing of the MCU at the wrong time, adversely affecting the operation of the SLIC module or of the application itself. More information about this is contained in 14.9.16 Oscillator Trimming with SLIC.

If the user software determines that the ID read out of the SLCD corresponds to a command or request message for which this node needs to receive or transmit data (respectively), it will then move on to procedures described in 14.9.8 Handling Command Message Frames and 14.9.9 Handling Request LIN Message Frames.

For clarification, in this document, “command” messages will refer to any message frame where the SLIC module is receiving data bytes and “request” messages refer to message frames where the SLIC module will be expected to transmit data bytes. This is a generic description and should not be confused with the terminology in the LIN specification. The LIN use of the terms “command” and “request” have the same basic meaning, but are limited in scope to specific identifier values of 0x3C and 0x3D. In the SLIC module documentation, these terms have been used to describe these functional types of messages, regardless of the specific identifier value used.

14.9.7.2 Possible Errors on Message Headers

Possible errors on message headers are:

- Identifier-Parity-Error
- Byte Framing Error

14.9.8 Handling Command Message Frames

Figure 14-16 shows how to handle command message frames, where the SLIC module is receiving data from the master node.

Command message frames refer to LIN messages frames where the master node is “commanding” the slave node to do something. The implication is that the slave will then be receiving data from the master for this message frame. This can be a standard LIN message frame of 1–8 data bytes, a reserved LIN system message (using 0x3C identifier), or an extended command message frame utilizing the reserved 0x3E user defined identifier or perhaps the 0x3F LIN reserved extended identifier. The SLIC module is capable of handling message frames containing up to 64 bytes of data, while still automatically calculating and/or verifying the checksum.

14.9.8.1 Standard Command Message Frames

After the application software has read the incoming identifier and determined that it is a valid identifier which cannot be ignored using IMSG, it must determine if this message frame is a command message frame or a request message frame. (i.e., should the application receive data from the master or send data back to the master?)

The first case, shown in Figure 14-16 deals with command messages, where the SLIC will be receiving data from the master node. If the received identifier corresponds to a standard LIN command frame (i.e., 1–8 data bytes), the user must then write the number of bytes (determined by the system designer and directly linked with this particular identifier) corresponding to the length of the message frame into SLCDLC. The two most significant bits of this register are used for special control bits describing the nature of this message frame.

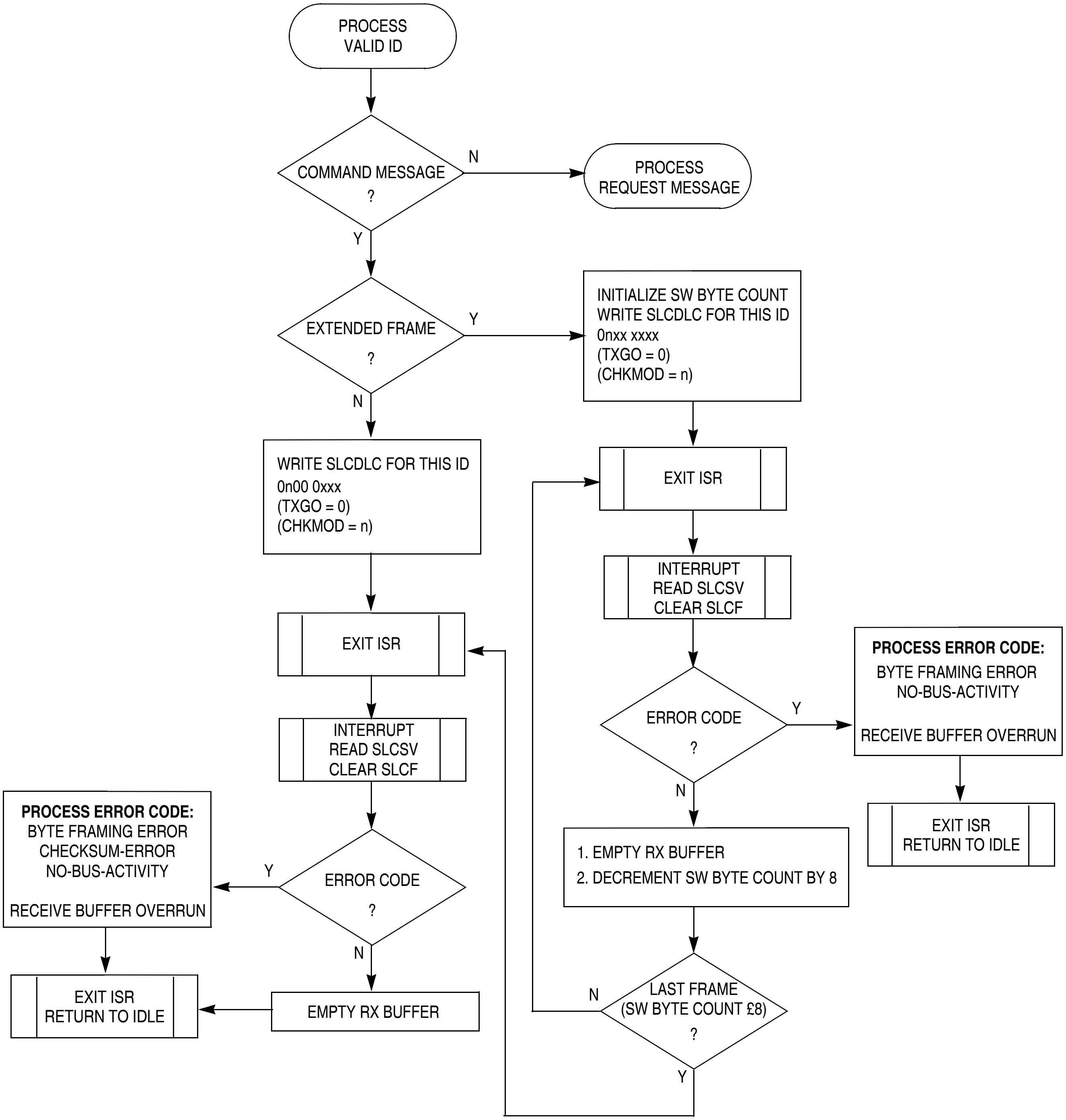


Figure 14-16. Handling Command Messages (Data Receive)

The SLIC transmit go (TXGO) bit should be 0 for command frames, indicating to the SLIC that data is coming from the master. The checksum mode control (CHKMOD) bit allows the user to select which method of checksum calculation is desired for this message frame. The LIN 1.3 checksum does not include the identifier byte in the calculation, while the SAE version does include this byte. Because the identifier is already received by the SLIC by this time, the default is to include it in the calculation. If a LIN 1.3 checksum is desired, a 1 in CHKMOD will reset the checksum circuitry to begin calculating the checksum on the first data byte. Using CHKMOD in this way allows the SLIC to receive messages with either method of data consistency check and change on a frame-by-frame basis. If a system uses both types of data consistency checking methods, the software must simply change the setting of this bit based on the identifier of each message. If the network only uses one type of check, CHKMOD can be set as a constant value in the user's code. If CHKMOD is not written on each frame, care must be taken not to accidentally modify the bit when writing the data length and TXGO bits. This is especially true if using C code without carefully inspecting the output of the compiler and assembler.

The control bits and data length code are contained in one register, allowing the user to maximize the efficiency of the identifier processing by writing a single byte value to indicate the nature of the message frame. This allows very efficient identifier processing code, which is important in a command frame, as the master node can be sending data immediately following the identifier byte which might be as little as one byte in length. The SLIC module uses a separate internal storage area for the incoming data bytes, so there is no danger of losing incoming data, but the user should spend as little time as possible within the ISR to ensure that the application or other ISRs are able to use the majority of the CPU bandwidth.

The identifier must be processed in a maximum of 2 byte times on the LIN bus to ensure that the ISR completes before the checksum would be received for the shortest possible message. This should be easily achievable, as the only operations required are to read SLCID and look up the checksum method, data length, and command/request state of that identifier, then write that value to the SLCDLC. This can be easily streamlined in code with a lookup table of identifiers and corresponding SLCDLC bytes.

NOTE

Once the ID is decoded for a message header and a length code written to SLCDLC, the SLIC is expecting that number of bytes to be received. If the SLIC module doesn't receive the number of bytes indicated in the SLCDLC register, it will continue to look for data bytes. If another message header begins, a byte framing error will trigger on the break symbol of that second message. The second message will still properly generate an ID received interrupt, but the byte framing error prior to this is an indication to the application that the previous message was not properly handled and should be discarded.

14.9.8.2 Extended Command Message Frames

Handling of extended frames is very similar to handling of standard frames, providing that the length is less than or equal to 64 bytes. Because the SLIC module can only receive 8 bytes at a time, the receive buffer must be emptied periodically for long message frames. This is not standard LIN operation, and is likely only to be used for downloading calibration data or reprogramming FLASH devices in a factory or service facility, so the added steps required for processing are not as critical to performance. During these types of operations, the application code is likely very limited in scope and special adjustments can be made to compensate for added message processing time.

For extended command frames, the data length is still written one time at the time the identifier is decoded, along with the TXGO and CHKMOD bits. When this is done, a software counter must also be initialized to keep track of how many bytes are expected to be received in the message frame. The ISR completes, clearing the SLCF flag, and resumes application execution. The SLIC will generate an interrupt, if unmasked, after 8 bytes are received or an error is detected. At this interrupt, the SLCSV will indicate an error condition (in case of byte framing error, idle bus) or that the receive buffer is full. If the data is successfully received, the user must then empty the buffer by reading SLCD7-SLCD0 and then subtract 8 from the software byte count. When this software counter reaches 8 or fewer, the remaining data bytes will fit in the buffer and only one interrupt should occur. At this time, the final interrupt may be handled normally, continuing to use the software counter to read the proper number of bytes from the appropriate SLCD registers.

NOTE

Do not write SLCDLC more than one time per LIN message frame. The SLIC tracks the number of sent or received bytes based on the value written to this register at the beginning of the data field and rewriting this register will corrupt the checksum calculation and cause unpredictable behavior in the SLIC module. The application software must track the number of sent or received bytes to know what the current byte count in the SLIC is. If programming in C, make sure to use the VOLATILE modifier on this variable (or make it a global variable) to ensure that it keeps its value between interrupts.

14.9.8.3 Possible Errors on Command Message Data

Possible errors on command message data are:

- Byte Framing Error
- Checksum-Error (LIN specified error)
- No-Bus-Activity (LIN specified error)
- Receiver Buffer Overrun Error

14.9.9 Handling Request LIN Message Frames

Figure 14-17 shows how to handle request message frames, where the SLIC module is sending data to the master node.

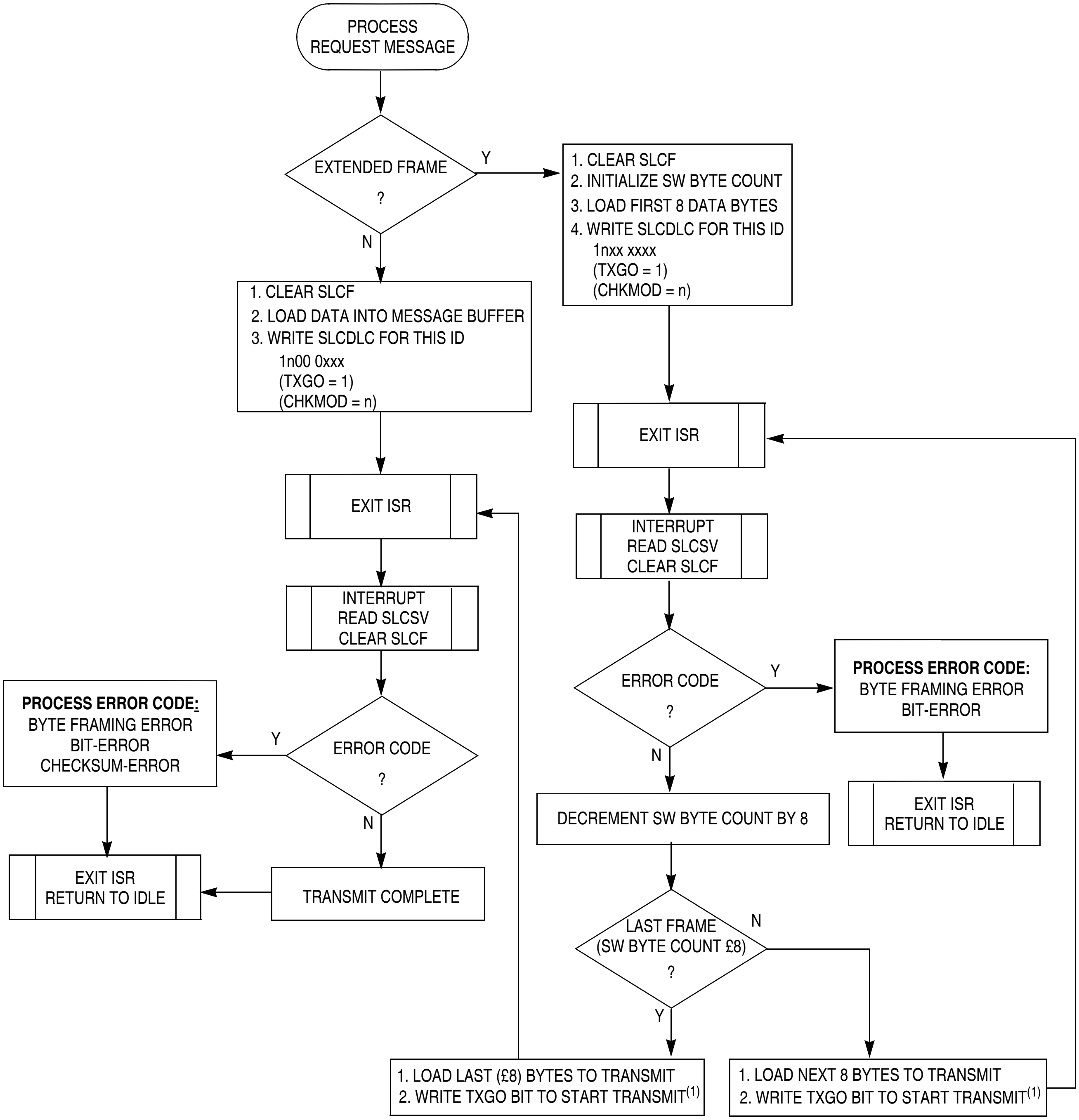
Request message frames refer to LIN messages frames where the master node is “requesting” the slave node to supply information. The implication is that the slave will then be transmitting data to the master for this message frame. This can be a standard LIN message frame of 1–8 data bytes, a reserved LIN system message (using 0x3D identifier), or an extended request message frame utilizing the reserved 0x3E identifier or perhaps the 0x3F LIN reserved extended identifier. The SLIC module is capable of handling request message frames containing up to 64 bytes of data, while still automatically calculating and/or verifying the checksum.

14.9.9.1 Standard Request Message Frames

Dealing with request messages with the SLIC is very similar to dealing with command messages, with one important difference. Because the SLIC is now to be transmitting data in the LIN message frame, the user software must load the data to be transmitted into the message buffer prior to initiating the transmission. This means an extra step is taken inside the interrupt service routine after the identifier has been decoded and is determined to be an ID for a request message frame.

Figure 14-17 deals with request messages, where the SLIC will be transmitting data to the master node. If the received identifier corresponds to a standard LIN command frame (i.e., 1-8 data bytes), the message processing is very simple. The user must load the data to be transmitted into the transmit buffer by writing it to the SLCD registers. The first byte to be transmitted on the LIN bus must be loaded into SLCD0, then SLCD1 for the second byte, etc. After all of the bytes to be transmitted are loaded in this way, a single write to SLCDLC will allow the user to encode the number of data bytes to be transmitted (1–8 bytes for standard request frames), set the proper checksum calculation method for the data

(CHKMOD), as well as signal the SLIC that the buffer is ready by writing a 1 to TXGO. TXGO will remain set to 1 until the buffer is sent successfully or an error is encountered, signaling to the application code that the buffer is in process of transmitting. In cases of 1–8 data bytes only being sent (standard LIN request frames), the SLIC automatically calculates and transmits the checksum byte at the end of the message frame. The user can exit the ISR after SLCDLC has been written and the SLCF flag has been cleared.



Note 1. When writing TXGO bit only, ensure that CHKMOD and data length values are not accidentally modified.

Figure 14-17. Handling Request LIN Message Frames

The next SLIC interrupt which occurs, if unmasked, will indicate the end of the request message frame and will either indicate that the frame was properly transmitted or that an error was encountered during transmission. Refer to 14.9.9.4 Possible Errors on Request Message Data for more detailed explanation of these possible errors. This interrupt also signals to the application that the message frame is complete and all data bytes and the checksum value have been properly transmitted onto the bus.

The SLIC module cannot begin to transmit the data until the user writes a 1 to TXGO, indicating that data is ready. If the user writes TXGO without loading data into the transmit buffer, whatever data is in storage will be transmitted, where the number of bytes transmitted is based on the data length value in the data length register. Similarly, if the user writes the wrong value for the number of data bytes to transmit, the SLIC will transmit that number of bytes, potentially transmitting garbage data onto the bus. The checksum calculation is performed based on the data transmitted, and will therefore still be calculated.

The identifier must be processed, data must be loaded into the transmit buffer, and the SLCDLC value written to initiate data transmission in a certain amount of time, based on the LIN specification. If the user waits too long to start transmission, the master node will observe an idle bus and trigger a Slave Not Responding error condition. The same error can be triggered if the transmission begins too late and does not complete before the message frame times out. Refer to the LIN specification for more details on timing constraints and requirements for LIN slave devices. This is especially important when dealing with extended request frames, when the data must be loaded in 8 byte sections (maximum) to be transmitted at each interrupt.

14.9.9.2 Extended Request Message Frames

Handling of extended frames is very similar to handling of standard frames, providing that the length is less than or equal to 64 bytes. Because the SLIC module can only transmit 8 bytes at a time, the transmit buffer must be loaded periodically for extended message frames. This is not standard LIN operation, and is likely only to be used for special cases, so the added steps required for processing should not be as critical to performance. During these types of operations, the application code is likely very limited in scope and special adjustments can be made to compensate for added message processing time.

When handling extended request frames, it is important to clear the SLCF flag first, before loading any data or writing TXGO. The data length is still written only one time, at the time the identifier is decoded, along with the TXGO and CHKMOD bits, after the first 8 data bytes are loaded into the transmit buffer. When this is done, a software counter must also be initialized to keep track of how many bytes are to be transmitted in the message frame. The SLIC will generate an interrupt, if unmasked, after 8 bytes are transmitted or an error is detected. At this interrupt, the SLCSV will indicate an error condition (in case of byte framing error or bit error) or that the transmit buffer is empty. If the data is transmitted successfully, the user must then clear the SLCF flag, subtract 8 from the software byte count, load the next 8 bytes into the SLCD registers, and write a 1 to TXGO to tell the SLIC that the buffers are loaded and transmission can commence. When this software counter reaches 8 or fewer, the remaining data bytes will fit in the transmit buffer and the SLIC will automatically append the checksum value to the frame after the last byte is sent.

NOTE

Do not write the CHKMOD or data length values in SLCDLC more than one time per message frame. The SLIC tracks the number of sent or received bytes based on the value written to this register at the beginning of the data field and rewriting this register will corrupt the checksum calculation and cause unpredictable behavior in the SLIC module. The application software must track the number of sent or received bytes to know what the current *byte count in the SLIC is. If programming in C, make sure to use the STATIC modifier on this variable (or make it a global variable) to ensure that it keeps its value between interrupts.*

14.9.9.3 Transmit Abort

The transmit abort bit (TXABRT) in SLCC1 allows the user to cease transmission of data on the next byte boundary. When this bit is set to 1, it will finish transmitting the byte currently being transmitted, then cease transmission. After the transmission is successfully aborted, TXABRT will automatically be reset by the SLIC to 0. If the SLIC is not in process of transmitting at the time TXABRT is written to 1, there is no effect and TXABRT will read back as 0.

14.9.9.4 Possible Errors on Request Message Data

Possible errors on request message data are:

- Byte Framing Error
- Checksum-Error (LIN specified error)
- Bit-Error

14.9.10 Handling IMSG to Minimize Interrupts

The IMSG feature is designed to minimize the number of interrupts required to maintain LIN communications. On a network with many slave nodes, it is very likely that a particular slave will observe messages which are not intended for that node. When the SLIC module detects any message header, it synchronizes to that message frame and bit rate, then interrupts the CPU after the identifier byte has been successfully received and parity checked. At this time, if the software determines that the message may be ignored, IMSG may be set to indicate to the module that the data field of the message frame is to be ignored and no additional interrupts should be generated until the next valid message header is received. The bit is automatically reset to 0 after the current message frame is complete and the LIN bus returns to idle state. This reduces the load on the CPU and allows the application software to immediately begin performing any operations which might otherwise not be allowed while receiving messaging.

NOTE

IMSG will prevent another interrupt from occurring for the current message frame, however if data bytes are appearing on the bus they may be received and copied into the message buffer. This will delete any previous data which might have been present in the buffer, even though no interrupt is triggered to indicate the arrival of this data.

14.9.11 Sleep and Wakeup Operation

The SLIC module itself has no special sleep mode, but does support low-power modes and wake-up on network activity. For low-power operations, the user must select whether or not to allow the SLIC clock to continue operating when the MCU issues a wait instruction through the SLC wait clock mode (SLCWCM) bit in SLCC1. If SLCWCM = 1, the SLIC will enter SLIC STOP mode when the MCU executes a WAIT instruction. If SLCWCM = 0, the SLIC will enter SLIC WAIT mode when the MCU executes a WAIT instruction. For more information on these modes, as well as wakeup options from these modes, please refer to 14.5 Modes of Operation.

When network activity occurs, the SLIC module will wake the MCU out of stop or wait mode, and return the SLIC module to SLIC run mode. If the SLIC was in SLIC wait mode, normal SLIC interrupt processing will resume. If the SLIC was in SLIC stop mode, SLCSV will indicate wakeup as the interrupt source so that the user knows that the SLIC module brought the MCU out of stop or wait.

In a LIN system, a system message is generally sent to all nodes to indicate that they are to enter low-power network sleep mode. After a node enters sleep mode, it waits for outside events, such as switch or sensor inputs or network traffic to bring it out of network sleep mode. If the node using the SLIC module is awakened by a source other than network traffic, such as a switch input, the LIN specification requires this node to issue a wake-up signal to the rest of the network. The SLIC module supports this feature using WAKETX in SLCC2. The user software may set this bit and one LIN wake-up signal is immediately transmitted on the bus, then the bit is automatically cleared by the SLIC module. If another wake-up signal is required to be sent, the user must set WAKETX again.

In a LIN system, the LIN physical interface can often also provide an output to the $\overline{\text{IRQ}}$ pin to provide a wake-up mechanism on network activity. The physical layer might also control voltage regulation supply to the MCU, cutting power to the MCU when the physical layer is placed in its low-power mode. The user must take care to ensure that the interaction between the physical layer, $\overline{\text{IRQ}}$ pin, SLIC transmit and receive pins, and power supply regulator is fully understood and designed to ensure proper operation.

14.9.12 Polling Operation

It is possible to operate the SLIC module in polling mode, if desired. The primary difference is that the SLIC interrupt request should not be enabled (SLCIE = 0). The SLCSV will update and operate properly and interrupt requests will be indicated with the SLCF flag, which can be polled to determine status changes in the SLIC module. It is required that the polling rate be fast enough to ensure that SLIC status changes be recognized and processed in time to ensure that all application timings can be met.

14.9.13 LIN Data Integrity Checking Methods

The SLIC module supports two different LIN-based data integrity options:

- The first option supports LIN 1.3 and older methods of checksum calculations.
- The second option supports an optional additional enhanced checksum calculation which has greater data integrity coverage.

The LIN 1.3 and earlier specifications transmit a checksum byte in the “CHECKSUM FIELD” of the LIN message frame. This CHECKSUM FIELD contains the inverted modulo-256 sum over all data bytes. The sum is calculated by an “ADD with Carry” where the carry bit of each addition is added to the least significant bit (LSB) of its resulting sum. This guarantees security also for the MSBs of the data bytes. The sum of modulo-256 sum over all data bytes and the checksum byte must be ’0xFF’.

An optional checksum calculation can also be performed on a LIN data frame which is very similar to the LIN 1.3 calculation, but with one important distinction. This enhanced calculation simply includes the identifier field as the first value in the calculation, whereas the LIN 1.3 calculation begins with the least significant byte of the data field (which is the first byte to be transmitted on the bus). This enhanced calculation further ensures that the identifier field is correct and ties the identifier and data together under a common calculation, ensuring greater reliability.

In the SLIC module, either checksum calculation can be performed on any given message frame by simply writing or clearing CHKMOD in SLCDLC, as desired, when the identifier for the message frame is decoded. The appropriate calculation for each message frame should be decided at system design time and documented in the LIN description file, indicating to the user which calculation to use for a particular identifier.

14.9.14 High-Speed LIN Operation

High-speed LIN operation does not necessarily require any reconfiguration of the SLIC module, depending upon what maximum LIN bit rate is desired. Several factors affect the performance of the SLIC module at LIN speeds higher than 20 kbps, all of which are functions of the speed of the SLIC clock and the prescaler of the digital filter. The tightest constraint comes from the need to maintain ±1.5% accuracy with the master node timing. This requires that the SLIC module be able to sample the incoming data stream accurately enough to guarantee that accuracy. Table 14-5 shows the maximum LIN bit rates allowable to maintain this accuracy.

Table 14-5. Maximum LIN Bit Rates for High-Speed Operation

SLIC Clock (MHz)	Maximum LIN Bit Rate for ±1% SLIC Accuracy (Bits / Second)	Maximum LIN Bit Rate for ±1.5% SLIC Accuracy (Bits / Second)
8	80,000	120,000
6.4	64,000	96,000
4.8	48,000	72,000
4	40,000	60,000
3.2	32,000	48,000
2.4	24,000	36,000
2	20,000	30,000

The above numbers assume a perfect input waveforms into the SLCRX pin, where 1 and 0 bits are of equal length and are exactly the correct length for the appropriate speed. Factors such as physical layer wave shaping and ground shift can affect the symmetry of these waveforms, causing bits to appear shortened or lengthened as seen by the SLIC module. The user must take these factors into account and base the maximum speed upon the shortest possible bit time that the SLIC module may observe, factoring in all physical layer effects. On some LIN physical layer devices it is possible to turn off wave shaping circuitry for high-speed operation, removing this portion of the physical layer error.

The digital receive filter can also affect high speed operation if it is set too low and begins to filter out valid message traffic. Under ideal conditions, this will not happen, as the digital filter maximum speeds allowable are higher than the speeds allowed for ±1.5% accuracy. If the digital receive filter prescaler is set to divide- by-4; however, the filter delay is very close to the ±1.5% accuracy maximum bit time.

For example, with a SLIC clock of 4 MHz, the SLIC module is capable of maintaining ±1.5% accuracy up to 60,000 bps. If the digital receive filter prescaler is set to divide-by-4, this means that the filter will only pass message traffic which is 62,500 bps or slower under ideal circumstances. This is only a difference of 2,500 bps (4.17% of the nominal valid message traffic speed). In this case, the user must ensure that with all errors accounted for, no bit will appear shorter than 16 μs
(1 bit at 62,500 bps) or the filter will block that bit. This is far too narrow a margin for safe design practices. The better solution would be to reduce the filter prescaler, increasing the gap between the filter cut-off point and the nominal speed of valid message traffic. Changing the prescaler to divide by 2 in this example gives a filter cut-off of 125,000 bps, which is 60,000 bps faster than the nominal speed of the LIN bus and much less likely to interfere with valid message traffic.

To ensure that all valid messages pass the filter stage in high-speed operation, it is best to ensure that the filter cut-off point is at least 2 times the nominal speed of the fastest message traffic to appear on the bus. Refer to Table 14-6 for a more complete list of the digital receive filter delays as they relate to the maximum LIN bus frequency. Table 14-7 repeats much of the data found in Table 14-6; however, the filter delay values (cutoff values) are shown in the frequency and time domains. Note that Table 14-7 shows the filter performance under ideal conditions.

When switching between a low-speed (< 4800 bps) to a high-speed (> 40000 bps) LIN message, the master node must allow a minimum idle time of eight bit times (of the slowest bit rate) between the messages. This prevents a valid message at another frequency from being detected as an invalid message.

Table 14-6. Maximum LIN Bit Rates for High-Speed Operation Due to Digital Receive Filter

SLIC Clock (MHz)	Maximum LIN Bit Rate for ±1.5% SLIC Accuracy (for Master-Slave Communication (Bits / Second) DIGITAL RX FILTER NOT CONSIDERED	Maximum LIN Bit Rate with Digital RX Filter Set to ÷4 (Bits / Second)	Maximum LIN Bit Rate with Digital RX Filter Set to ÷3 (Bits / Second)	Maximum LIN Bit Rate with Digital RX Filter Set to ÷2 (Bits / Second)	Maximum LIN Bit Rate with Digital RX Filter Set to ÷1 (Bits / Second)
		THESE PRESCALERS NOT RECOMMENDED FOR HIGH-SPEED LIN OPERATION			
8	120,000	120,000[1]	120,000[1]	120,000[1]	120,000[1]
6.4	96,000	100,000	120,000[1]	120,000[1]	120,000[1]
4.8	72,000	75,000	100,000	120,000[1]	120,000[1]
4	60,000	62,500	83,333	120,000[1]	120,000[1]
3.2	48,000	50,000	66,667	100,000	120,000[1]
2.4	36,000	37,500	50,000	75,000	120,000[1]
2	30,000	31,250	41,667	62,500	120,000[1]

[1] Bit rates over 120,000 bits per second are not recommended for LIN communications, as physical layer delay between the TX and RX pins can cause the stop bit of a byte to be mis-sampled as the last data bit. This could result in a byte framing error.

Table 14-7. Digital Receive Filter Absolute Cutoff (Ideal Conditions)

SLIC Clock (MHz)	Digital RX Filter Set to ÷4		Digital RX Filter Set to ÷3		Digital RX Filter Set to ÷2		Digital RX Filter Set to ÷1	
	Max. Bit Rate (Bits / Sec)	Min Pulse Width Allowed (μs)	Max. Bit Rate (Bits / Sec)	Min Pulse Width Allowed (μs)	Max. Bit Rate (Bits / Sec)	Min Pulse Width Allowed (μs)	Max. Bit Rate (Bits / Sec)	Min Pulse Width Allowed (μs)
8	125,000	8.0	166,667	6.0	250,000	4.0	500,000	2.0
6.4	100,000	10.0	133,333	7.5	200,000	5.0	400,000	2.5
4.8	75,000	13.3	100,000	10.0	150,000	6.7	300,000	3.3
4	62,500	16.0	83,333	12.0	125,000	8.0	250,000	4.0
3.2	50,000	20.0	66,667	15.0	100,000	10.0	200,000	5.0
2.4	37,500	26.7	50,000	20.0	75,000	13.3	150,000	6.7
2	31,250	32.0	41,667	24.0	62,500	16.0	125,000	8.0

14.9.15 Byte Transfer Mode Operation

This subsection describes the operation and limitations of the optional UART-like byte transfer mode (BTM). This mode allows sending and receiving individual bytes, but changes the behavior of the SLCBT registers (now read/write registers) and locks the SLCDLC to 1 byte data length. The SLCBT value now becomes the bit time reference for the SLIC, where the software sets the length of one bit time rather than the SLIC module itself. This is similar to an input capture/output compare (IC/OC) count in a timer module, where the count value represents the number of SLIC clock counts in one bit time.

Byte transfer mode assumes that the user has a very stable, precise oscillator, resonator, or clock reference input into the MCU and is therefore not appropriate for use with internal oscillators. There is no synchronization method available to the user in this mode and the user must tell the SLIC how many clock counts comprise a bit time. Figure 14-18, Figure 14-19, Figure 14-20, and Figure 14-21 show calculations to determine the SLCBT value for different settings.

NOTE

It is possible to use the LIN autobauding circuitry in a non-LIN system to derive the correct bit timing values if system constraints allow. To do this the SLIC module must be activated in LIN mode (BTM=0) and receive a break symbol, 0x55 data byte and one additional data byte (at the desired BTM speed). Upon receiving this sequence of symbols which appears to be a LIN header, the SLIC module will assert an ID received successfully interrupt (SLCSV=0x2C). The value in the SLCBT registers will reflect the bit rate which the 0x55 data character was received and can be saved to RAM. The user then switches the SLIC into BTM mode and reloads this value from RAM and the SLIC will be configured to communicate in BTM mode at the baud rate which the 0x55 data character was sent.

In the example in Figure 14-18, the user should write 0x16, as a write of 0x15 (decimal value of 21) would automatically revert to 0x14, resulting in transmitted bit times that are 1.33 SLIC clock periods too short rather than 0.667 SLIC clock periods too long. The optimal choice, which gives the smallest resolution error, is the closest even number of SLIC clocks to the exact calculated SLCBT value.

There is a trade-off between maximum bit rate and resolution with the SLIC in BTM mode. Faster SLIC clock speeds improve resolution, but require higher numbers to be written to the SLCBT registers for a given desired bit rate. It is up to the user to determine what level of resolution is acceptable for the given application.

Desired Bit Rate:	57,600 bps
External Crystal Frequency:	4.9152 MHz

$$\frac{\text{1 Second}}{\text{57,600 Bits}} = \frac{\text{17.36111 ms}}{\text{1 Bit}}$$

$$\frac{\text{1 Second}}{\text{4,915,200 CGMXCLK Periods}} \text{ X } \frac{\text{4 CGMXCLK Period}}{\text{1 SLIC Clock Period}} = \frac{\text{813.802 ns}}{\text{1 SLIC Clock Period}}$$

$$\frac{\text{17.36111 ms}}{\text{1 Bit}} \text{ X } \frac{\text{1 SLIC Clock Period}}{\text{813.802 ns}} = \frac{\text{21.33 SLIC Clock Periods}}{\text{1 Bit}}$$

Therefore, the closest SLCBT value would be 21 SLIC clocks (SLCBT = 0x0015).
Because you can only use even values in SLCBT, the closest acceptable value is 22 (0x0016).

Figure 14-18. SLCBT Value Calculation Example 1

Desired Bit Rate: 57,600 bps
External Crystal Frequency: 9.8304 MHz

$$\frac{\text{1 Second}}{\text{57,600 Bits}} = \frac{\text{17.36111 ms}}{\text{1 Bit}}$$

$$\frac{\text{1 Second}}{\text{9,830,400 CGMXCLK Periods}} \text{ X } \frac{\text{4 CGMXCLK Period}}{\text{1 SLIC Clock Period}} = \frac{\text{406.90 ns}}{\text{1 SLIC Clock Period}}$$

$$\frac{\text{17.36111 ms}}{\text{1 Bit}} \text{ X } \frac{\text{1 SLIC Clock Period}}{\text{406.90 ns}} = \frac{\text{42.67 SLIC Clock Periods}}{\text{1 Bit}}$$

Therefore, the closest SLCBT value would be 42 SLIC clocks (SLCBT = 0x002A).

Figure 14-19. SLCBT Value Calculation Example 2

Desired Bit Rate: 15,625 bps
External Crystal Frequency: 8.000 MHz

$$\frac{\text{1 Second}}{\text{15,625 Bits}} = \frac{\text{64 ms}}{\text{1 Bit}}$$

$$\frac{\text{1 Second}}{\text{8,000,000 CGMXCLK Periods}} \text{ X } \frac{\text{4 CGMXCLK Period}}{\text{1 SLIC Clock Period}} = \frac{\text{500 ns}}{\text{1 SLIC Clock Period}}$$

$$\frac{\text{64 ms}}{\text{1 Bit}} \text{ X } \frac{\text{1 SLIC Clock Period}}{\text{500 ns}} = \frac{\text{128 SLIC Clock Periods}}{\text{1 Bit}}$$

Therefore, the closest SLCBT value would be 128 SLIC clocks (SLCBT = 0x0080).

Figure 14-20. SLCBT Value Calculation Example 3

Desired Bit Rate: 9.615 bps
External Crystal Frequency: 8.000 MHz

$$\frac{\text{1 Second}}{\text{9,615 Bits}} = \frac{\text{104.004 ms}}{\text{1 Bit}}$$

$$\frac{\text{1 Second}}{\text{8,000,000 CGMXCLK Periods}} \text{ X } \frac{\text{4 CGMXCLK Period}}{\text{1 SLIC Clock Period}} = \frac{\text{500 ns}}{\text{1 SLIC Clock Period}}$$

$$\frac{\text{104.004 ms}}{\text{1 Bit}} \text{ X } \frac{\text{1 SLIC Clock Period}}{\text{500 ns}} = \frac{\text{208.008 SLIC Clock Periods}}{\text{1 Bit}}$$

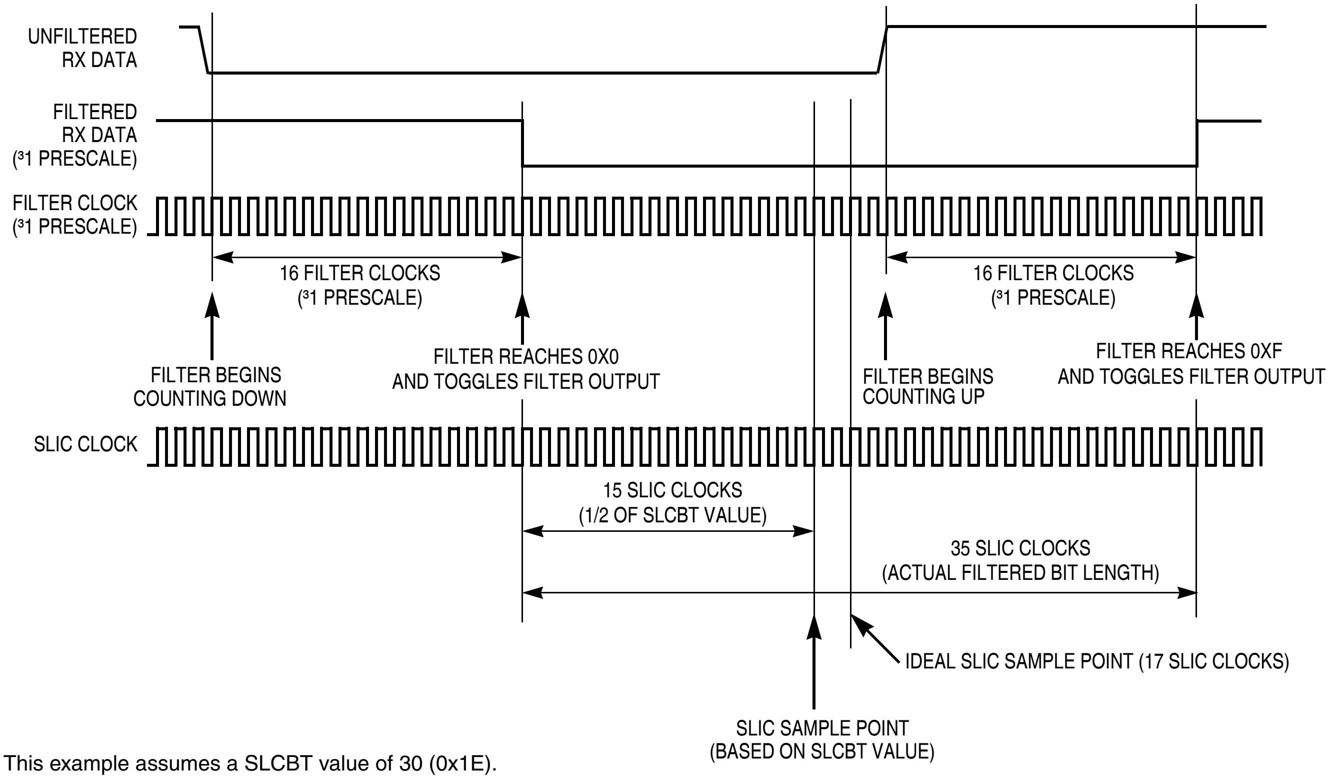
Therefore, the closest SLCBT value would be 42 SLIC clocks (SLCBT = 0x00D0).

Figure 14-21. SLCBT Value Calculation Example 4

This resolution affects the sampling accuracy of the SLIC module on receiving bytes, but only as far as locating the sample point of each bit within a given byte. The best sample point of the bit may be off by as much as one SLIC clock period from the exact center of the bit, if the proper SLCBT value for the desired bit rate is an odd number of SLIC clock periods.

Figure 14-22 shows an example of this error. In this example, the user has additionally chosen an incorrect value of 30 SLIC clocks for the length of one bit time, and a filter prescaler of 1. This makes little difference in the receive sampling of this particular bit, as the sample point is still within the bit and the digital filter will catch any noise pulses shorter than 16 filter clocks long.The ideal value of SLCBT would be 35 SLIC clocks, but the closest available value is 34, placing the sample point at 17 SLIC clocks into the bit.

The error in the bit time value chosen by the user in the above example will grow throughout the byte, as the sample point for the next bit will be only 30 SLIC clock cycles later (1 full bit time at this bit rate setting). The SLIC resynchronizes upon every falling edge received. In a 0x00 data byte, however, there are no falling edges after the beginning of the start bit. This means that the accumulated error of the sampling point over the data byte with these settings could be as high as 30 SLIC clock cycles (10 bits x {2 SLIC clocks due to user error + 1 SLIC clock resolution error}) placing it at the boundary between the last bit and the stop bit. This could result in missampling and missing a byte framing error on the last bit on high speed communications when the SLCBT count is relatively low. A properly chosen SLCBT value would result in a maximum error of 10 SLIC clock counts over a given byte. This is less than one filter delay time, and will not cause missampling of any of the bits in that byte. At the falling edge of the next start bit, the SLIC will resynchronize and any accumulated sampling error returns to 0. The sampling error becomes even less significant at lower speeds, when higher values of SLCBT are used to define a bit time, as the worst case bit time resolution error is still only one SLIC clock per bit (or maximum of 10 SLIC clocks per byte).


UNFILTERED RX DATA
FILTERED RX DATA (^{3}1 PRESCALE)
FILTER CLOCK (^{3}1 PRESCALE)
16 FILTER CLOCKS (^{3}1 PRESCALE)
16 FILTER CLOCKS (^{3}1 PRESCALE)
FILTER BEGINS COUNTING DOWN
FILTER REACHES 0X0 AND TOGGLES FILTER OUTPUT
FILTER BEGINS COUNTING UP
FILTER REACHES 0XF AND TOGGLES FILTER OUTPUT
SLIC CLOCK
15 SLIC CLOCKS (1/2 OF SLCBT VALUE)
35 SLIC CLOCKS (ACTUAL FILTERED BIT LENGTH)
IDEAL SLIC SAMPLE POINT (17 SLIC CLOCKS)
SLIC SAMPLE POINT (BASED ON SLCBT VALUE)

This example assumes a SLCBT value of 30 (0x1E).
Transmitted bits will be sent out as 30 SLIC clock cycles long.

The proper closest SLCBT setting would be 34 (0x22), which gives the ideal sample point of 17 SLIC clocks and transmitted bits are 34 SLIC clocks long.

Figure 14-22. BTM Mode Receive Byte Sampling Example

The error also comes into effect with transmitted bit times. Using the previous example with a SLCBT value of 34, transmitted bits will appear as 34 SLIC clock periods long. This is one SLIC clock short of the proper length. Depending on the frequency of the SLIC clock, one period of the SLIC clock might be a large or a small fraction of one ideal bit time. Raising the frequency of the SLIC clock will reduce this error relative to the ideal bit time, improving the resolution of the SLIC clock relative to the bit rate of the bus. In any case, the error is still one SLIC clock cycle. Raising the SLIC clock frequency, however, requires programming a higher value for SLCBT to maintain the same target bit rate.

Smaller values of SLCBT combined with higher values of the SLIC clock frequency (smaller clock period) will give faster bit rates, but the SLIC clock period becomes an increasingly significant portion of one bit time.

Because BTM mode does not perform any synchronization and relies on the accuracy of the data provided by the user software to set its sample point and generate transmitted bits, the constraint on maximum speeds is only limited to the limits imposed by the digital filter delay and the SLIC input clock. Because the digital filter delay cannot be less than 16 SLIC clock cycles, the fastest possible pulse which would pass the filter is 16 clock periods at 8 MHz, or 500,000 bits/second. The values shown in Table 14-7 are the same values shown in Table 14-8 and indicate the absolute fastest bit rates which could just pass the minimum digital filter settings (prescaler = divide by 1) under perfect conditions.

Because perfect conditions are almost impossible to attain, more robust values must be chosen for bit rates. For reliable communication, it is best to ensure that a bit time is no smaller 2x–3x longer than the filter delay on the digital receive filter. This is true in LIN or BTM mode and ensures that valid data bits which have been shortened due to ground shift, asymmetrical rise and fall times, etc., are accepted by the filter without exception. This would translate to 2x to 3x reduction in the maximum speeds shown in Table 14-7. Recommended maximum bit rates are shown in Table 14-8, and ensure that a single bit time is at least twice the length of one filter delay value. If system noise is not adequately filtered out it might be necessary to change the prescaler of the filter and lower the bit rate of the communication. If valid communications are being absorbed by the filter, corrective action must be taken to ensure that either the bit rate is reduced or whatever physical fault is causing bit times to shorten is corrected (ground offset, asymmetrical rise/fall times, insufficient physical layer supply voltage, etc.).

Table 14-8. Recommended Maximum Bit Rates for BTM Operation Due to Digital Filter

SLIC Clock (MHz)	Maximum BTM Bit Rate with Digital RX Filter Set to ÷4 (Bits / Second)	Maximum BTM Bit Rate with Digital RX Filter Set to ÷3 (Bits / Second)	Maximum BTM Bit Rate with Digital RX Filter Set to ÷2 (Bits / Second)	Maximum BTM Bit Rate with Digital RX Filter Set to ÷1 (Bits / Second)
8	62,500	83,333	120,000[1]	120,000[1]
6.4	50,000	66,667	100,000	120,000[1]
4.8	37,500	50,000	75,000	120,000[1]
4	31,250	41,667	62,500	120,000[1]
3.2	25,000	33,333	50,000	100,000
2.4	18,750	25,000	37,500	75,000
2	15,625	20,833	31,250	62,500

[1] Bit rates over 120,000 bits per second are not recommended for BTM communications, as physical layer delay between the TX and RX pins can cause the stop bit of a byte to be missampled as the last data bit. This could result in a byte framing error.

14.9.16 Oscillator Trimming with SLIC

SLCACT can be used as an indicator of LIN bus activity. SLCACT tells the user that the SLIC is currently processing a message header (therefore synchronizing to the bus) or processing a message frame (including checksum). Therefore, at idle times between message frames or during a message frame which has been marked as a “don’t care” by writing IMSG, it is possible to trim the oscillator circuit of the MCU with no impact to the LIN communications.

It is important to note the exact mechanisms with which the SLIC sets and clears SLCACT. Any falling edge which successfully passes through the digital receive filter will cause SLCACT to become set. This might even include noise pulses, if they are of sufficient length to pass through the digital RX filter. Although in these cases SLCACT is becoming set on a noise spike, it is very probable that noise of this nature will cause other system issues as well such as corruption of the message frame. The software can then further qualify if it is appropriate to trim the oscillator.

SLCACT will only be cleared by the SLIC upon successful completion of a normal LIN message frame (see 14.8.3 SLIC Status Register description for more detail). This means that in some cases, if a message frame terminates with an error condition or some source other than those cited in the SLCACT bit description, SLCACT might remain set during an otherwise idle bus time. SLCACT will then clear upon the successful completion of the next LIN message frame.

These mechanisms might result in SLCACT being set when it is safe (from the SLIC module perspective) to trim the oscillator. However, SLCACT will only be clear when the SLIC considers it safe to trim the oscillator.

In a particular system, it might also be possible to improve the opportunities for trimming by using system knowledge and use of IMSG. If a message ID is known to be considered a “don’t care” by this particular node, it should be safe to trim the oscillator during that message frame (provided that it is safe for the application software as well). After the software has done an identifier lookup and determined that the ID corresponds to a “don’t care” message, the software might choose to set IMSG. From that time, the application software should have at least one byte time of message traffic in which to trim the oscillator before that ignored message frame expires, regardless of the state of SLCACT. If the length of that ignored message frame is known, that knowledge might also be used to extend the time of this oscillator trimming opportunity.

Now that the mechanisms for recognizing when the SLIC module indicates safe oscillator trimming opportunities are understood, it is important to understand how to derive the information needed to perform the trimming.

The value in SLCBT will indicate how many SLIC clock cycles comprise one bit time and for any given LIN bus speed, this will be a fixed value if the oscillator is running at its ideal frequency. It is possible to use this ideal value combined with the measured value in SLCBT to determine how to adjust the oscillator of the microcontroller.

The actual oscillator trimming algorithm is very specific to each particular implementation, and applications might or might not require the oscillator even to be trimmed. The SLIC can maintain communications even with input oscillator variation of ±50% (with 4 MHz nominal, that means that any input clock into the SLIC from 2 MHz to 6 MHz will still guarantee communications). Because Freescale internal oscillators are at least within ±25% of their nominal value, even when untrimmed, this means that trimming of the oscillator is not even required for LIN communications. If the application can tolerate the range of frequencies which might appear within this manufacturing range, then it is not necessary ever to trim the oscillator. This can be a tremendous advantage to the customer, enabling migration to very low-cost ROM devices which have no non-volatile memory in which to store the trim value.

NOTE

Even though most internal oscillators are within ±25% before trimming, they are stable at some frequency in that range, within at least ±5% over the entire operating voltage and temperature range. The trimming operation simply eliminates the offset due to factory manufacturing variations to re-center the base oscillator frequency to the nominal value. Please refer to the electrical specifications for the oscillator for more specific information, as exact specifications might differ from module to module.

14.9.17 Digital Receive Filter

The receiver section of the SLIC module includes a digital low-pass filter to remove narrow noise pulses from the incoming message. block diagram of the digital filter is shown in Figure 14-23.

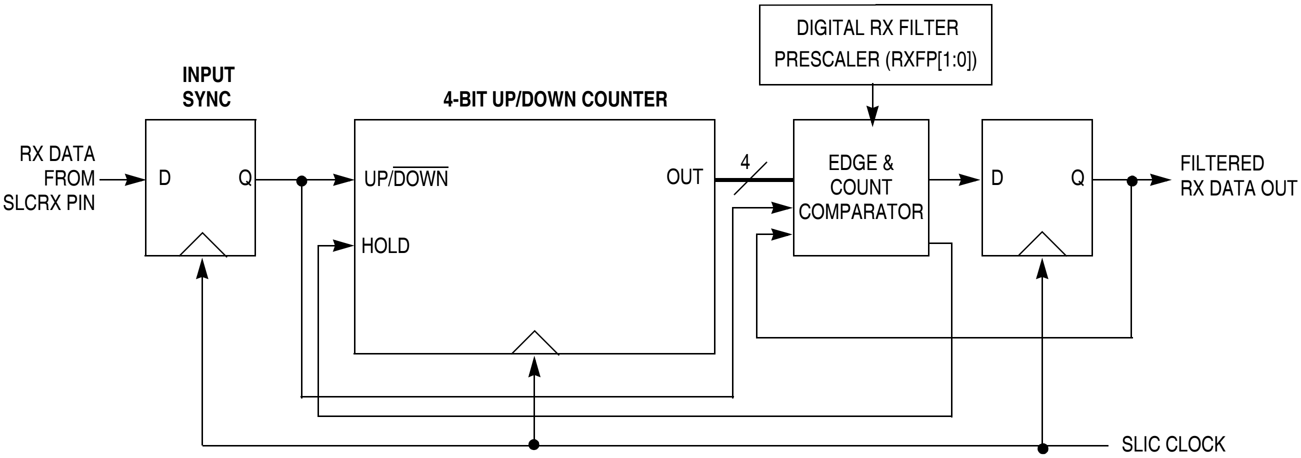


Figure 14-23. SLIC Module Rx Digital Filter Block Diagram

14.9.17.1 Digital Filter Operation

The clock for the digital filter is provided by the SLIC Interface clock. At each positive edge of the clock signal, the current state of the receiver input signal from the SLCRX pad is sampled. The SLCRX signal state is used to determine whether the counter should increment or decrement at the next positive edge of the clock signal.

The counter will increment if the input data sample is high but decrement if the input sample is low. The counter will thus progress up towards the highest count value (determined by RXFP bit settings), on average, the SLCRX signal remains high or progress down towards ‘0’ if, on average, the SLCRX signal remains low. The final counter value which determines when the filter will change state is generated by shifting the RXFP value right two positions and bitwise OR-ing the result with the value 0x0F. For example, a prescale setting of divide by 3 (RXFP = 0x80) would give a count value of 0x2F.

When the counter eventually reaches this value, the digital filter decides that the condition of the SLCRX signal is at a stable logic level 1 and the data latch is set, causing the filtered Rx data signal to become a logic level 1. Furthermore, the counter is prevented from overflowing and can only be decremented from this state.

Alternatively, when the counter eventually reaches the value ‘0’, the digital filter decides that the condition of the SLCRX signal is at a stable logic level 0 and the data latch is reset, causing the filtered Rx data signal to become a logic level 0. Furthermore, the counter is prevented from underflowing and can only be incremented from this state.

The data latch will retain its value until the counter next reaches the opposite end point, signifying a definite transition of the SLCRX signal.

14.9.17.2 Digital Filter Performance

The performance of the digital filter is best described in the time domain rather than the frequency domain.

If the signal on the SLCRX signal transitions, then there will be a delay before that transition appears at the filtered Rx data output signal. This delay will be between 15 and 16 clock periods, depending on where the transition occurs with respect to the sampling points. This ‘filter delay’ is not an issue for SLIC operation, as there is no need for message arbitration.

The effect of random noise on the SLCRX signal depends on the characteristics of the noise itself. Narrow noise pulses on the SLCRX signal will be completely ignored if they are shorter than the filter delay. This provides a degree of low-pass filtering. Figure 14-23 shows the configuration of the digital receive filter and the consequential effect on the filter delay. This filter delay value indicates that for a particular setup, only pulses of which are greater than the filter delay will pass the filter.

For example, if the frequency of the SLIC clock (f_{SLIC}) is 3.2 MHz, then the period (t_{SLIC}) is of the SLIC clock is 313 ns. With the default receive filter prescaler setting of division by 3, the resulting maximum filter delay in the absence of noise will be 15.00 μs. By simply changing the prescaler of the receive filter, the user can then select alternatively 5 μs, 10 μs, or 20 μs as a minimum filter delay according to the systems requirements.

If noise occurs during a symbol transition, the detection of that transition may be delayed by an amount equal to the length of the noise burst. This is just a reflection of the uncertainty of where the transition is truly occurring within the noise.

NOTE

The user must always account for the worst case bit timing of their LIN bus when configuring the digital receive filter, especially if running at faster speeds. Ground offset and other physical layer conditions can cause shortening of bits as seen at the digital receive pin, for example. If these shortened bit lengths are less than the filter delay, the bits will be interpreted by the filter as noise and will be blocked, even though the nominal bit timing might be greater than the filter delay.

Chapter 15
Timer Interface Module (TIM)

15.1 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 15-1 for port location of these shared pins.

15.2 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 15-1
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- Counter stop and reset bits

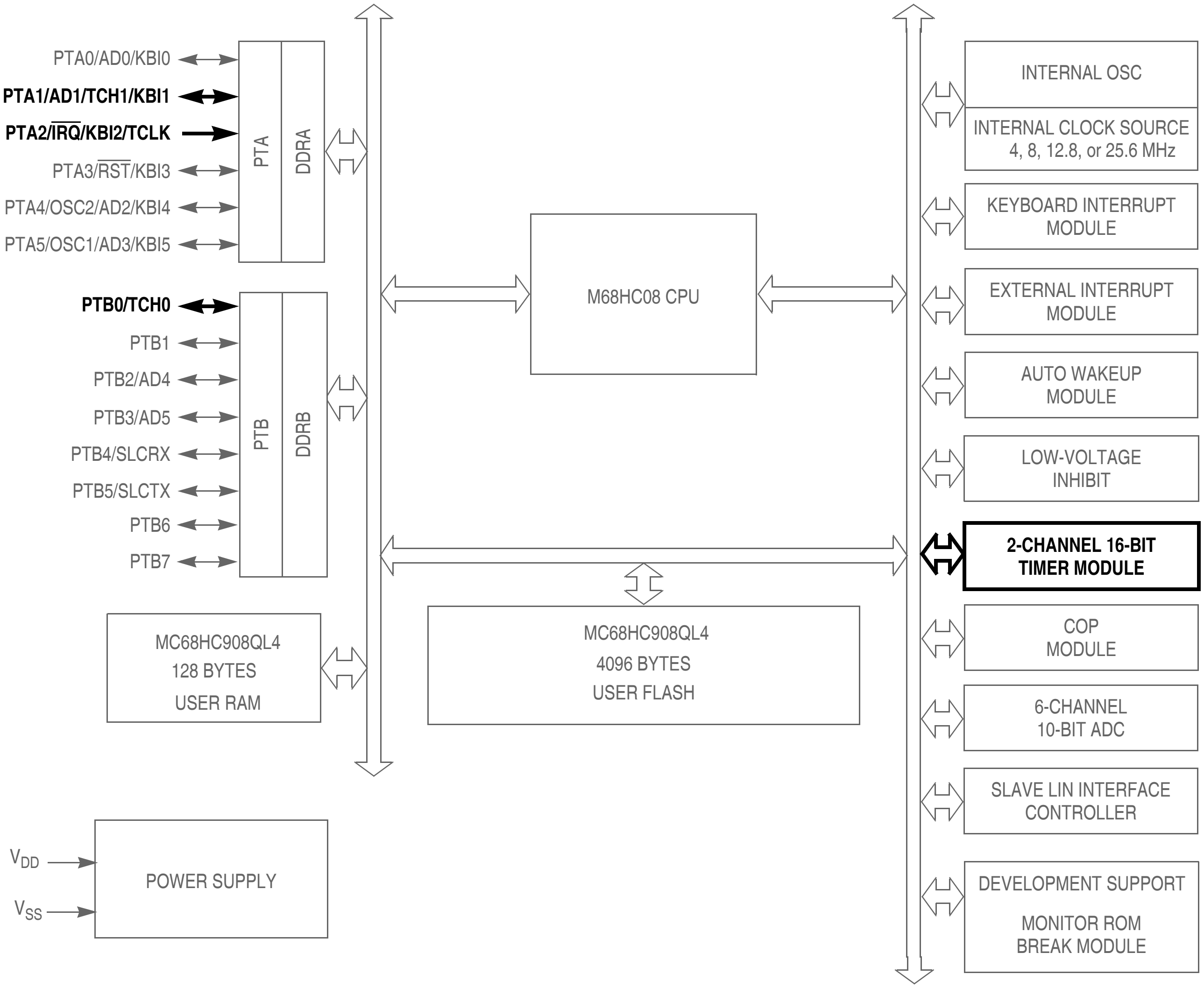
15.3 Functional Description

Figure 15-2 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.

15.3.1 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.



$\overline{\text{RST}}$, $\overline{\text{IRQ}}$: Pins have internal pull up device
All port pins have programmable pull up device (pullup/down on port A)
PTA[0:5]: Higher current sink and source capability

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

15.3.2 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.

15.3.3 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.

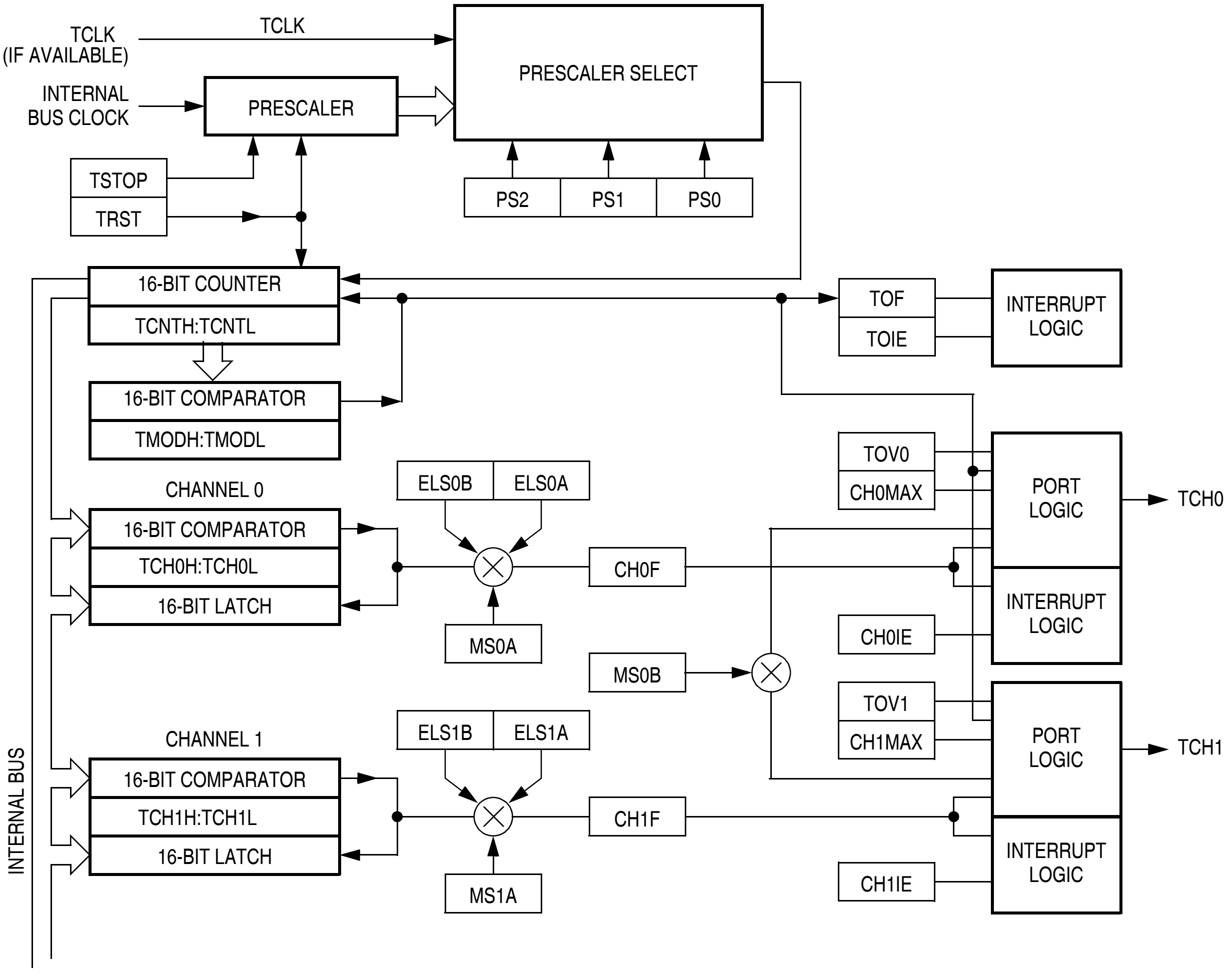


Figure 15-2. TIM Block Diagram

15.3.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in 15.3.3 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.

15.3.3.2 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.

15.3.4 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 15-3 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).

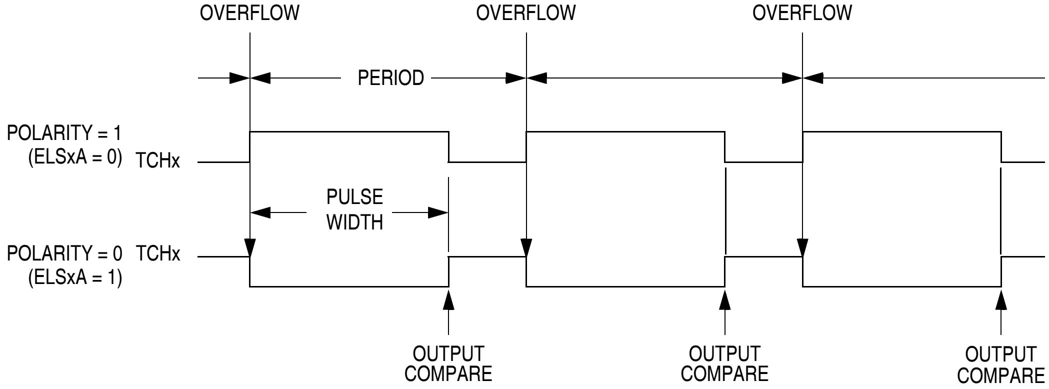


Figure 15-3. PWM Period and Pulse Width

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 15.8.1 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%.

15.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in 15.3.4 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.

15.3.4.2 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.

15.3.4.3 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 15-2.
   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 15-2.

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 (TSCR0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on 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 15.8.1 TIM Status and Control Register.

15.4 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.

15.5 Low-Power Modes

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

15.5.1 Wait Mode

The 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.

15.5.2 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.

15.6 TIM During Break Interrupts

A break interrupt stops the counter and inhibits input captures.

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

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

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

15.7 I/O Signals

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

15.7.1 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.

15.7.2 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]. 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 ÷ 2.

15.8 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)

15.8.1 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 15-4. 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 15-1 shows.

Table 15-1. 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)

15.8.2 TIM Counter Registers

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

NOTE

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

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

▨ = Unimplemented

Figure 15-5. TIM Counter High Register (TCNTH)

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

▨ = Unimplemented

Figure 15-6. TIM Counter Low Register (TCNTL)

15.8.3 TIM Counter Modulo Registers

The read/write TIM modulo registers contain the modulo value for the counter. When the counter reaches the modulo value, the overflow flag (TOF) becomes set, and the counter resumes counting from $0000 at the next timer clock. Writing to the high byte (TMODH) inhibits the TOF bit and overflow interrupts until the low byte (TMODL) is written. Reset sets the TIM counter modulo registers.

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

Figure 15-7. TIM Counter Modulo High Register (TMODH)

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

Figure 15-8. TIM Counter Modulo Low Register (TMODL)

NOTE

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

15.8.4 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 15-9. 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 15-10. 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.

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 15-2.

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 15-2).

1 = Initial output level low
0 = Initial output level high

NOTE

Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIM status and control register (TSC).

Table 15-2. Mode, Edge, and Level Selection

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

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to an I/O port, and pin TCHx is available as a general-purpose I/O pin. Table 15-2 shows how ELSxB and ELSxA work.

NOTE

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

TOVx — Toggle-On-Overflow Bit

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

1 = Channel x pin toggles on 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 15-11 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.

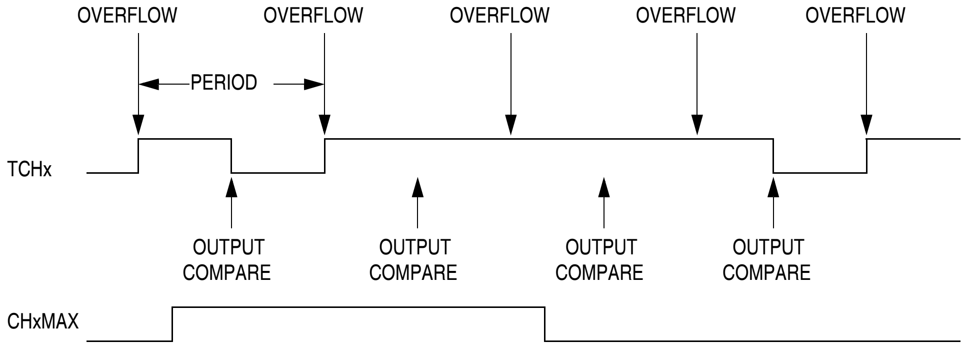


Figure 15-11. CHxMAX Latency

15.8.5 TIM Channel Registers

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

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

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

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

Figure 15-12. TIM Channel x Register High (TCHxH)

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

Figure 15-13. TIM Channel Register Low (TCHxL)

Chapter 16
Development Support

16.1 Introduction

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

16.2 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

16.2.1 Functional Description

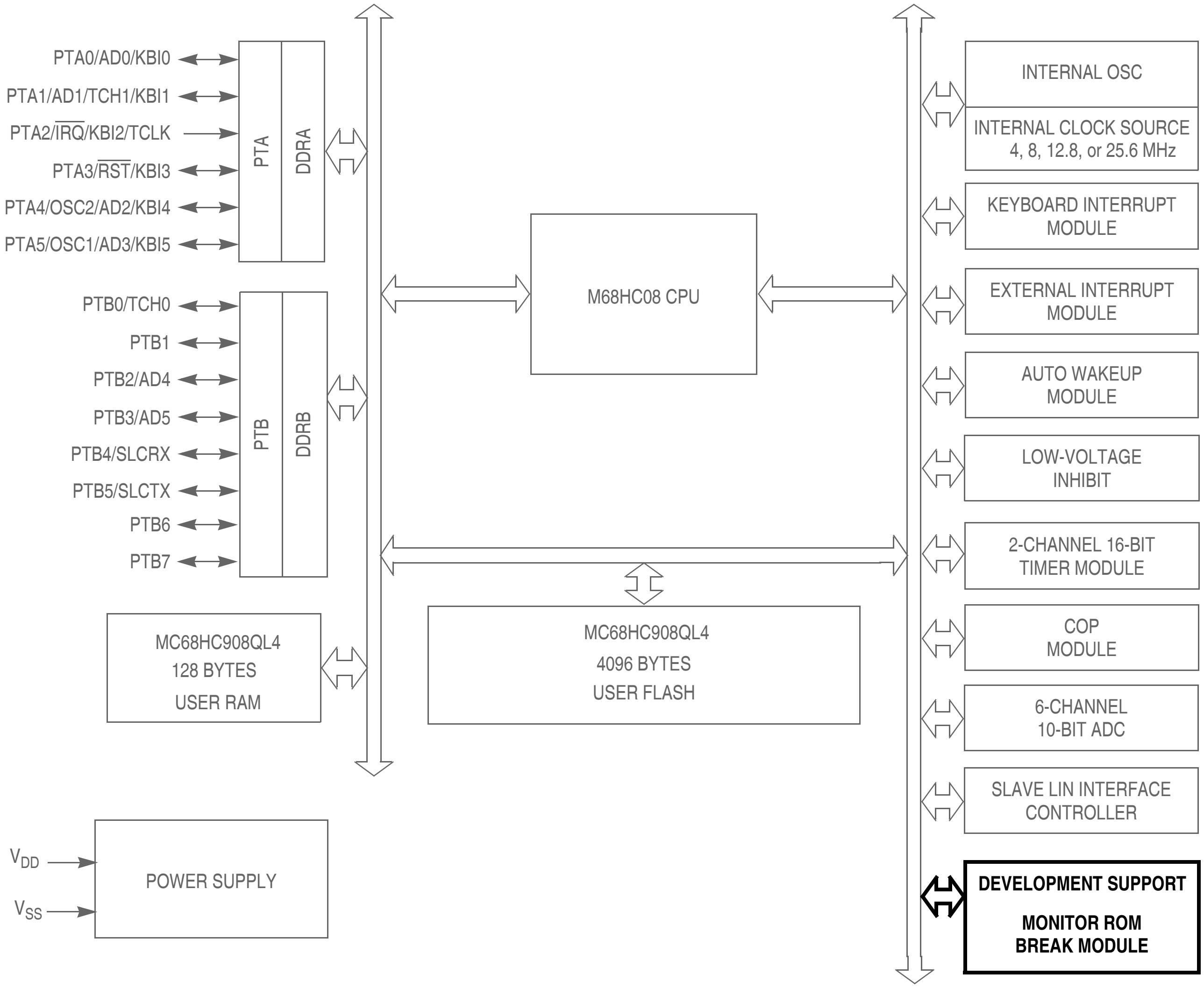
When the internal address bus matches the value written in the break address registers, the break module issues a breakpoint signal ($\overline{\text{BKPT}}$) to the system integration module (SIM). The SIM then causes the CPU to load the instruction register with a software interrupt instruction (SWI). 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 16-2 shows the structure of the break module.



$\overline{\text{RST}}$, $\overline{\text{IRQ}}$: Pins have internal pull up device
All port pins have programmable pull up device (pullup/down on port A)
PTA[0:5]: Higher current sink and source capability

Figure 16-1. Block Diagram Highlighting BRK and MON Blocks

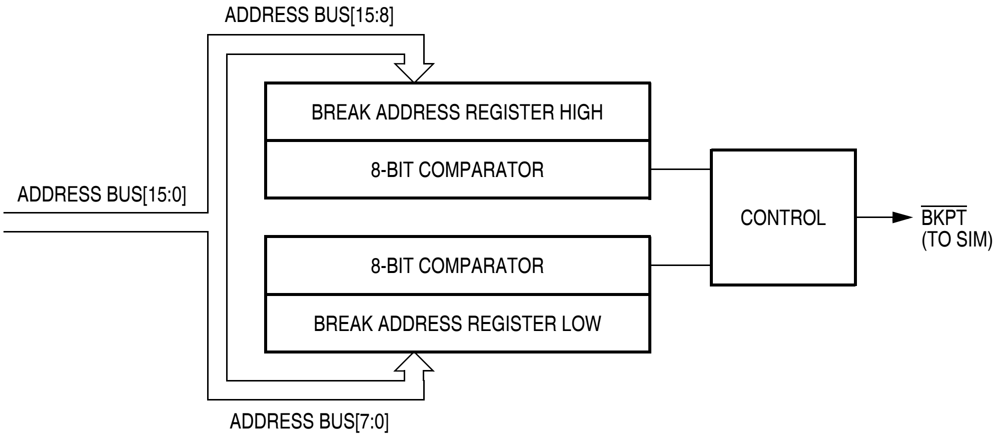


Figure 16-2. Break Module Block Diagram

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

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

The break interrupt timing is:

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

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

CAUTION

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

16.2.1.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether or not module status bits can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See 13.8.2 Break Flag Control Register and the **Break Interrupts** subsection for each module.

16.2.1.2 TIM During Break Interrupts

A break interrupt stops the timer counter and inhibits input captures.

16.2.1.3 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).

16.2.2 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)

16.2.2.1 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 16-3. 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

16.2.2.2 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: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Figure 16-4. Break Address Register High (BRKH)

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

Figure 16-5. Break Address Register Low (BRKL)

16.2.2.3 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 16-6. 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

16.2.2.4 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[1]	
Reset:							0	

R = Reserved

[1] Writing a 0 clears SBSW.

Figure 16-7. 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

16.2.2.5 Break Flag Control Register

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

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

R = Reserved

Figure 16-8. Break Flag Control Register (BFCR)

BCFE — Break Clear Flag Enable Bit

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

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

16.2.3 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.

16.3 Monitor Module (MON)

The monitor module allows debugging and programming of the microcontroller unit (MCU) through a single-wire interface with a host computer. Monitor mode entry can be achieved without use of the higher test voltage, V_{TST}, as long as vector addresses $FFFE and $FFFF are blank, thus reducing the hardware requirements for in-circuit programming.

Features include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between MCU and host computer
- Standard non-return-to-zero (NRZ) communication with host computer
- Standard communication baud rate (7200 @ 2-MHz bus frequency)
- Execution of code in random-access memory (RAM) or FLASH
- FLASH memory security feature[1]
- FLASH memory programming interface
- Use of external 9.8304 MHz crystal or clock to generate internal frequency of 2.4576 MHz
- Simple internal oscillator mode of operation (no external clock or high voltage)
- Monitor mode entry without high voltage, V_{TST}, if reset vector is blank ($FFFE and $FFFF contain $FF)
- Normal monitor mode entry if V_{TST} is applied to $\overline{\text{IRQ}}$

16.3.1 Functional Description

Figure 16-9 shows a simplified diagram of monitor mode entry.

The monitor module receives and executes commands from a host computer. Figure 16-10, Figure 16-11, and Figure 16-12 show example circuits used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

Simple monitor commands can access any memory address. In monitor mode, the MCU can execute code downloaded into RAM by a host computer while most MCU pins retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.

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

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

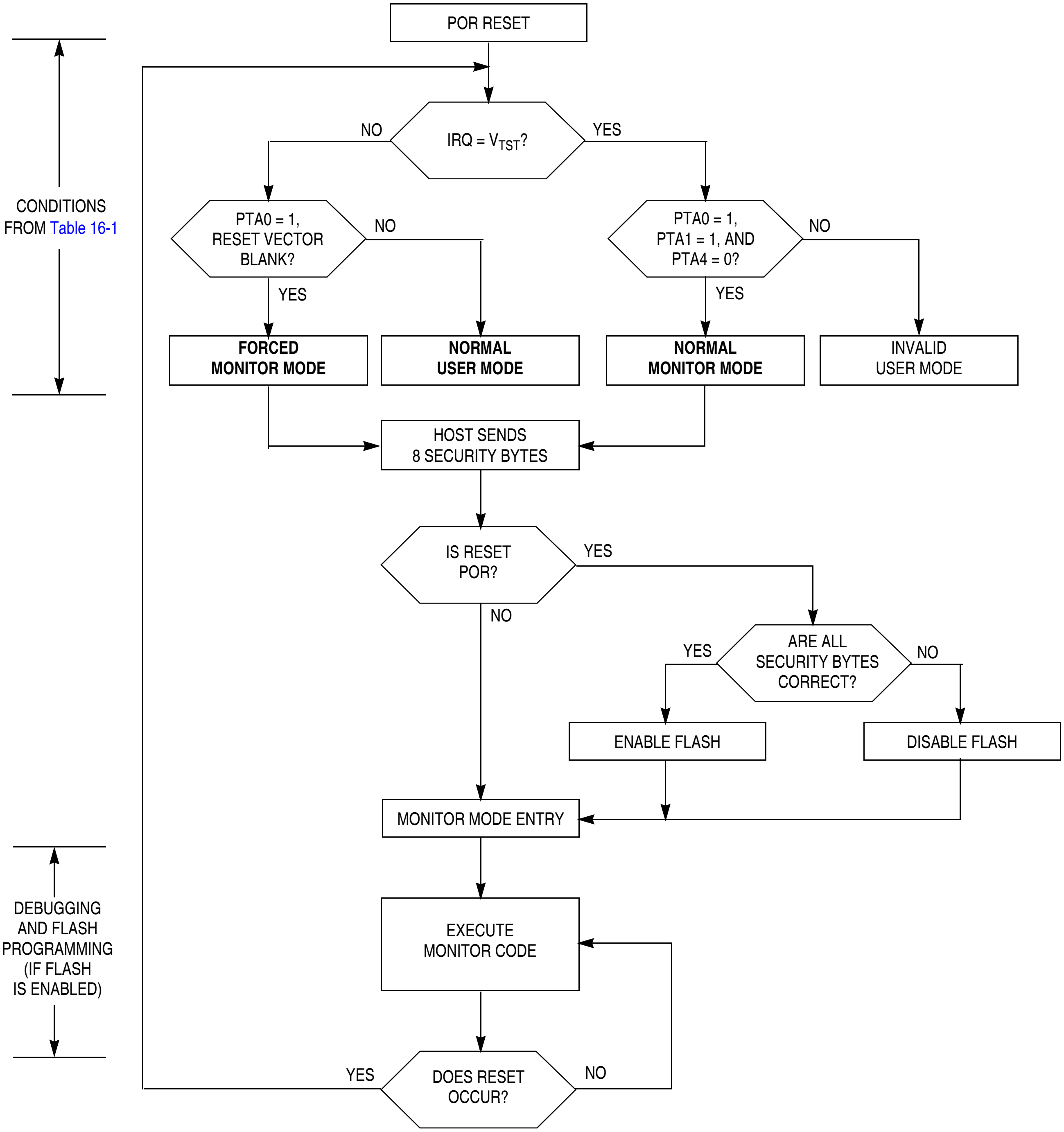


Figure 16-9. Simplified Monitor Mode Entry Flowchart

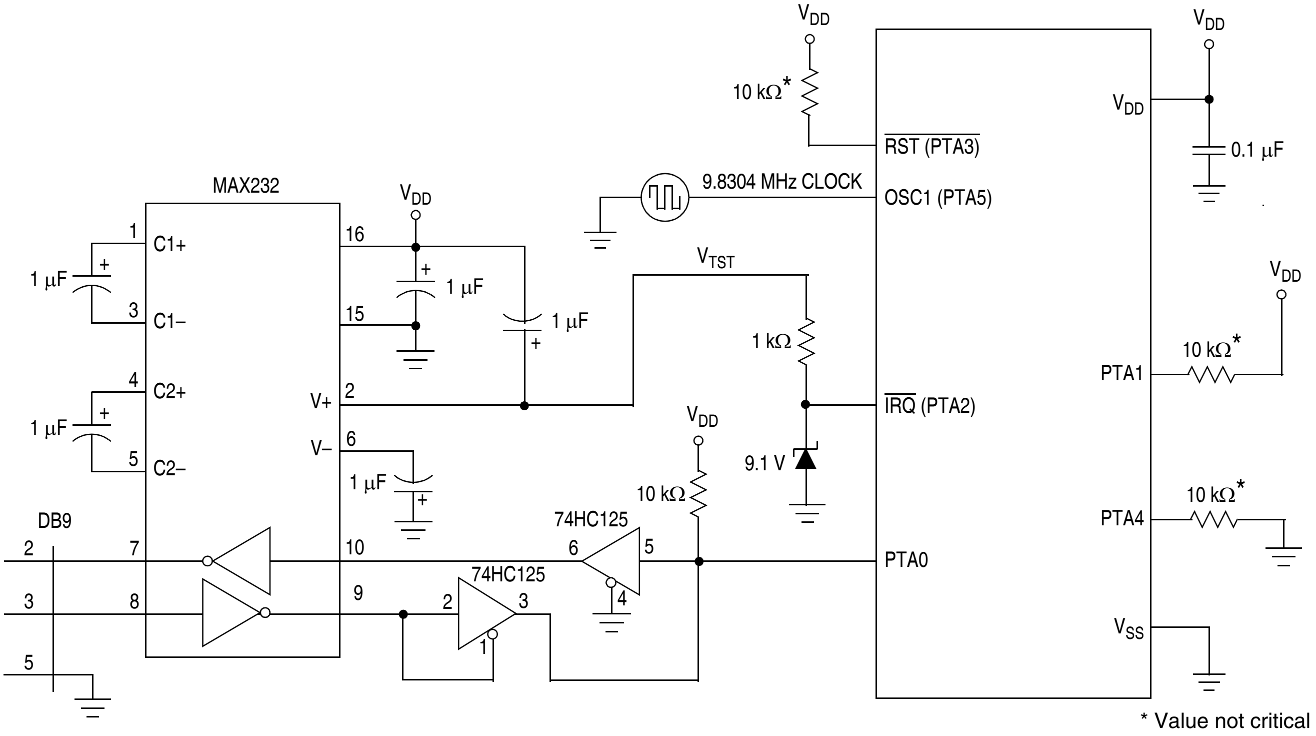


Figure 16-10. Monitor Mode Circuit (External Clock, with High Voltage)

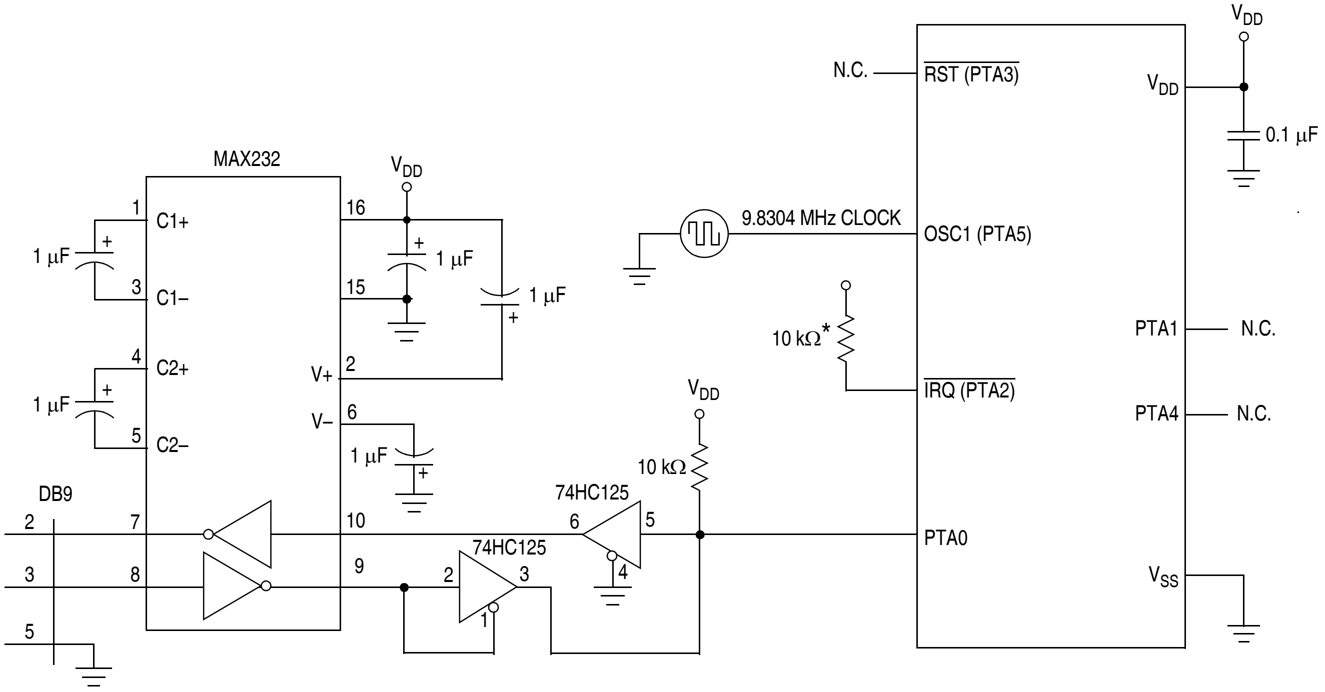


Figure 16-11. Monitor Mode Circuit (External Clock, No High Voltage)

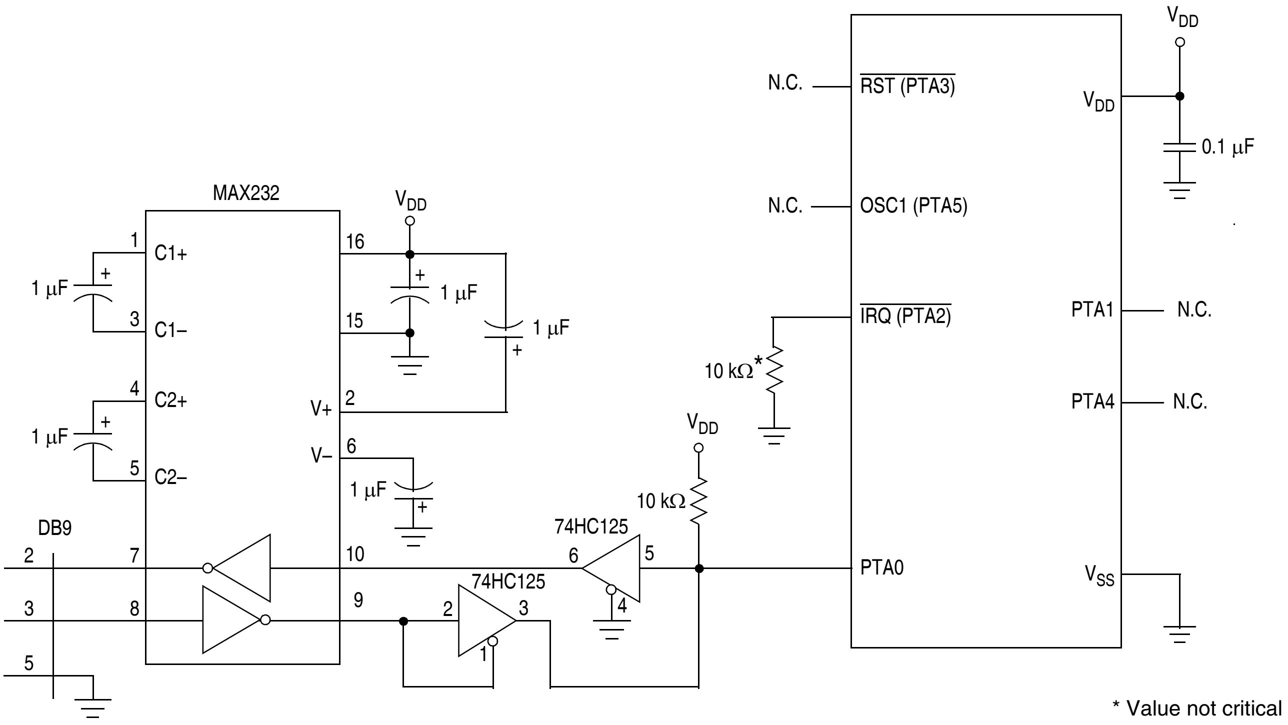


Figure 16-12. Monitor Mode Circuit (Internal Clock, No High Voltage)

Table 16-1 shows the pin conditions for entering monitor mode. As specified in the table, monitor mode may be entered after a power-on reset (POR) and will allow communication at 9600 baud provided one of the following sets of conditions is met:

- If \$FFFE and \$FFFF do not contain \$FF (programmed state):
  - The external clock is 9.8304 MHz
  - $\overline{IRQ}$ = V_{TST}
- If \$FFFE and \$FFFF contain \$FF (erased state):
  - The external clock is 9.8304 MHz
  - $\overline{IRQ}$ = V_{DD} (this can be implemented through the internal $\overline{IRQ}$ pullup)
- If \$FFFE and \$FFFF contain \$FF (erased state):

$\overline{IRQ}$ = V_{SS} (internal oscillator is selected, no external clock required)

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

Once out of reset, the MCU waits for the host to send eight security bytes (see 16.3.2 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 16-1. 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	Baud Rate	
Normal Monitor	V_{TST}	V_{DD}	X	1	1	0	Disabled	9.8304 MHz	2.4576 MHz	9600	Provide external clock at OSC1.
Forced Monitor	V_{DD}	X	$FFFF (blank)	1	X	X	Disabled	9.8304 MHz	2.4576 MHz	9600	Provide external clock at OSC1.
	V_{SS}	X	$FFFF (blank)	1	X	X	Disabled	X	3.2 MHz (Trimmed)	9600	Internal clock is active.
User	X	X	Not $FFFF	X	X	X	Enabled	X	X	X	
MON08 Function [Pin No.]	V_{TST} [6]	$\overline{\text{RST}}$ [4]	—	COM [8]	MOD0 [12]	MOD1 [10]	—	OSC1 [13]	—	—	

1. PTA0 must have a pullup resistor to V_{DD} in monitor mode.
2. Communication speed in the table is an example to obtain a baud rate of 9600. Baud rate using external oscillator is bus frequency / 256 and baud rate using internal oscillator is bus frequency / 333.
3. External clock is a 9.8304 MHz oscillator on OSC1.
4. X = don't care
5. MON08 pin refers to P&E Microcomputer Systems' MON08-Cyclone 2 by 8-pin connector.

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

.

16.3.1.1 Normal Monitor Mode

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

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

16.3.1.2 Forced Monitor Mode

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

NOTE

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

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

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

If IRQ is held high, all pins will default to regular input port functions except for PTA0 and PTA5 which will operate as a serial communication port and OSC1 input respectively (refer to Figure 16-11). That will allow the clock to be driven from an external source through OSC1 pin.

If IRQ is held low, all pins will default to regular input port function except for PTA0 which will operate as serial communication port. Refer to Figure 16-12.

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

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

16.3.1.3 Monitor Vectors

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

NOTE

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

Table 16-2 summarizes the differences between user mode and monitor mode regarding vectors.

Table 16-2. Mode Differences

Modes	Functions					
	Reset Vector High	Reset Vector Low	Break Vector High	Break Vector Low	SWI Vector High	SWI Vector Low
User	$FFFE	$FFFF	$FFFC	$FFFD	$FFFC	$FFFD
Monitor	$FEFE	$FEFF	$FEFC	$FEFD	$FEFC	$FEFD

16.3.1.4 Data Format

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

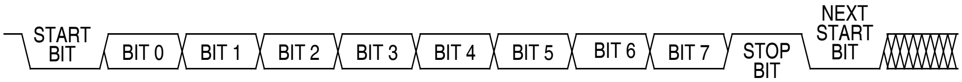


Figure 16-13. Monitor Data Format

16.3.1.5 Break Signal

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

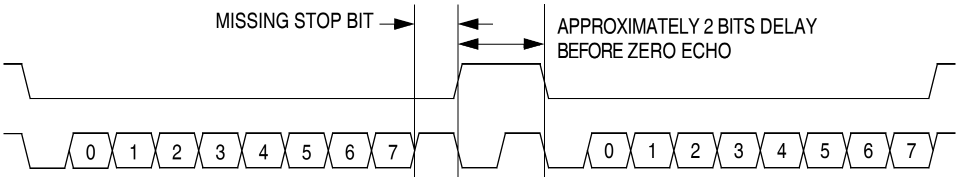


Figure 16-14. Break Transaction

16.3.1.6 Baud Rate

The monitor communication baud rate is controlled by the frequency of the external or internal oscillator and the state of the appropriate pins as shown in Table 16-1.

Table 16-1 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.

16.3.1.7 Commands

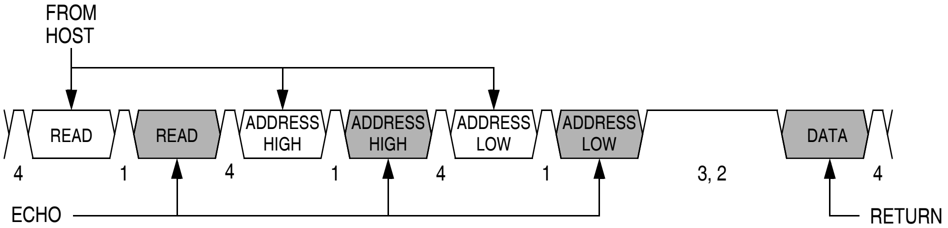
The monitor ROM firmware uses these commands:

- READ (read memory)
- WRITE (write memory)
- IREAD (indexed read)
- IWRITE (indexed write)
- READSP (read stack pointer)
- RUN (run user program)

The monitor ROM firmware echoes each received byte back to the PTA0 pin for error checking. An 11-bit delay at the end of each command allows the host to send a break character to cancel the command. A delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. The data returned by a read command appears after the echo of the last byte of the command.

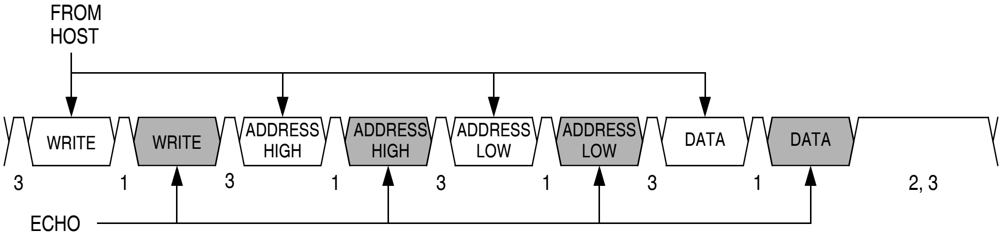
NOTE

Wait one bit time after each echo before sending the next byte.



Notes:
1 = Echo delay, approximately 2 bit times
2 = Data return delay, approximately 2 bit times
3 = Cancel command delay, 11 bit times
4 = Wait 1 bit time before sending next byte.

Figure 16-15. Read Transaction



Notes:
1 = Echo delay, approximately 2 bit times
2 = Cancel command delay, 11 bit times
3 = Wait 1 bit time before sending next byte.

Figure 16-16. Write Transaction

A brief description of each monitor mode command is given in Table 16-3 through Table 16-8.

Table 16-3. READ (Read Memory) Command

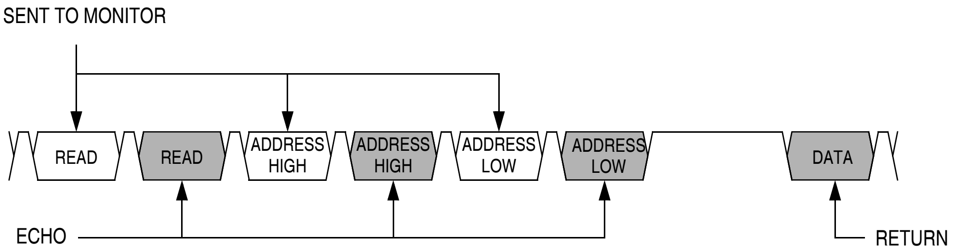
Description	Read byte from memory
Operand	2-byte address in high-byte:low-byte order
Data Returned	Returns contents of specified address
Opcode	$4A
Command Sequence 	

Table 16-4. WRITE (Write Memory) Command

Description	Write byte to memory
Operand	2-byte address in high-byte:low-byte order; low byte followed by data byte
Data Returned	None
Opcode	$49

Command Sequence

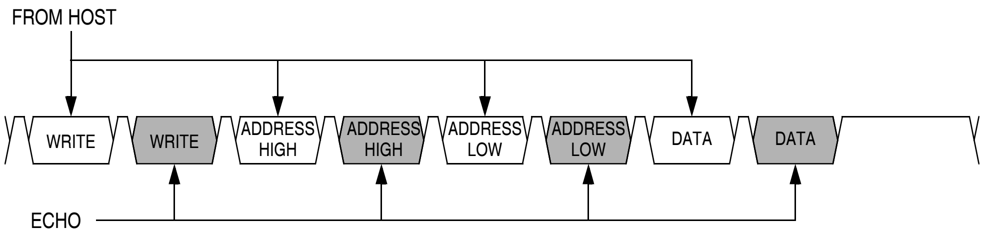


Table 16-5. IREAD (Indexed Read) Command

Description	Read next 2 bytes in memory from last address accessed
Operand	None
Data Returned	Returns contents of next two addresses
Opcode	$1A

Command Sequence

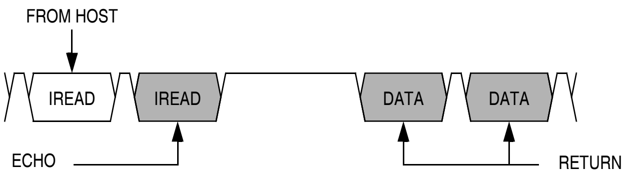
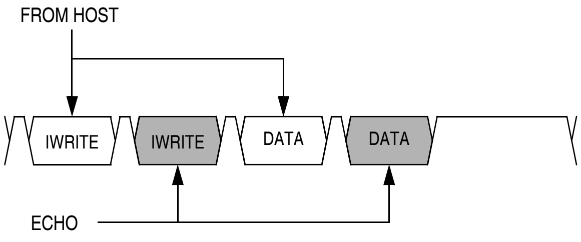


Table 16-6. IWRITE (Indexed Write) Command

Description	Write to last address accessed + 1
Operand	Single data byte
Data Returned	None
Opcode	$19

Command Sequence



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

Table 16-7. READSP (Read Stack Pointer) Command

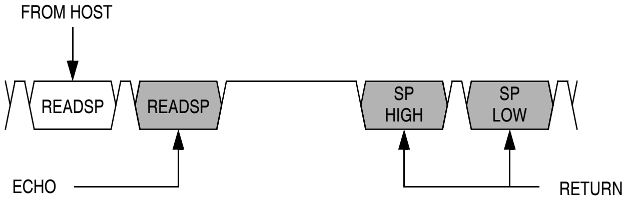
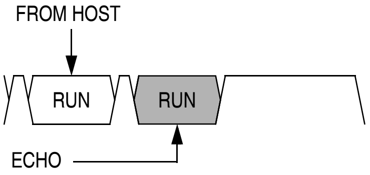
<table>
<tr><td>Description</td><td>Reads stack pointer</td></tr>
<tr><td>Operand</td><td>None</td></tr>
<tr><td>Data Returned</td><td>Returns incremented stack pointer value (SP + 1) in high-byte:low-byte order</td></tr>
<tr><td>Opcode</td><td>$0C</td></tr>
<tr><td colspan="2"><b>Command Sequence</b>

 </td></tr>
</table>

Table 16-8. RUN (Run User Program) Command

<table>
<tr><td>Description</td><td>Executes PULH and RTI instructions</td></tr>
<tr><td>Operand</td><td>None</td></tr>
<tr><td>Data Returned</td><td>None</td></tr>
<tr><td>Opcode</td><td>$28</td></tr>
<tr><td colspan="2"><b>Command Sequence</b>

 </td></tr>
</table>

The MCU executes the SWI and PSHH instructions when it enters monitor mode. The RUN command tells the MCU to execute the PULH and RTI instructions. Before sending the RUN command, the host can modify the stacked CPU registers to prepare to run the host program. The READSP command returns the incremented stack pointer value, SP + 1. The high and low bytes of the program counter are at addresses SP + 5 and SP + 6.

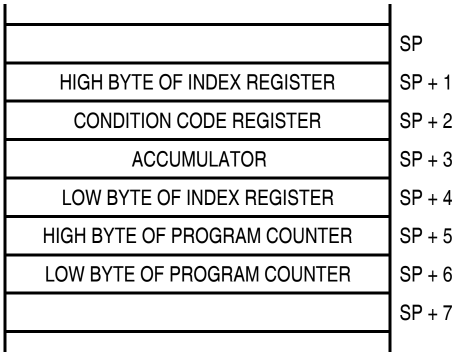

SP
HIGH BYTE OF INDEX REGISTER
SP + 1
CONDITION CODE REGISTER
SP + 2
ACCUMULATOR
SP + 3
LOW BYTE OF INDEX REGISTER
SP + 4
HIGH BYTE OF PROGRAM COUNTER
SP + 5
LOW BYTE OF PROGRAM COUNTER
SP + 6
SP + 7

Figure 16-17. Stack Pointer at Monitor Mode Entry

16.3.2 Security

A security feature discourages unauthorized reading of FLASH locations while in monitor mode. The host can bypass the security feature at monitor mode entry by sending eight security bytes that match the bytes at locations $FFF6–$FFFD. Locations $FFF6–$FFFD contain user-defined data.

NOTE

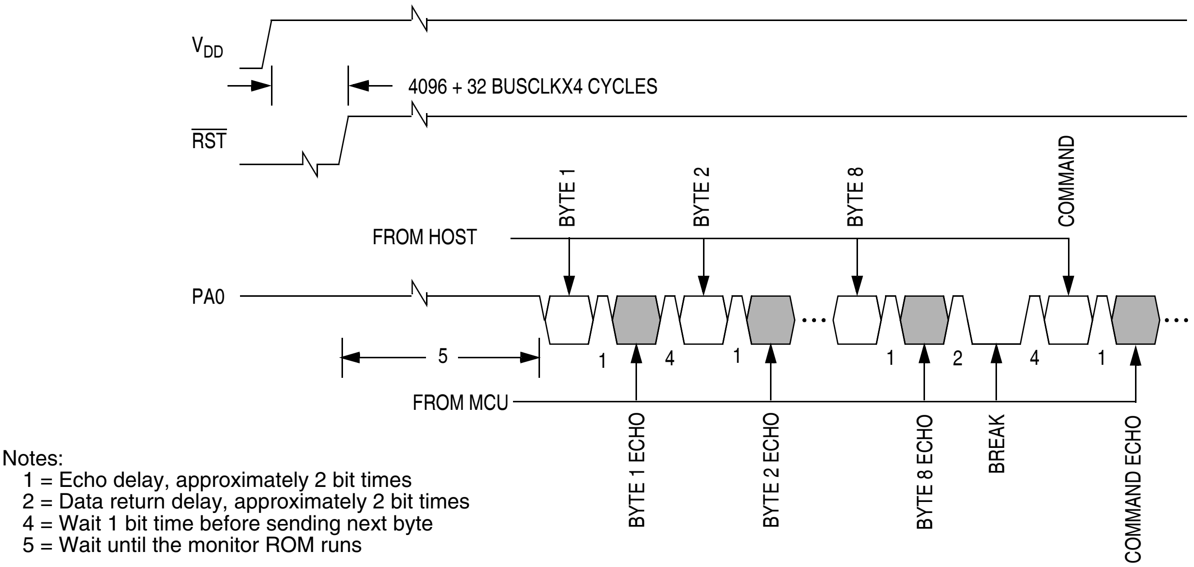
Do not leave locations $FFF6–$FFFD blank. For security reasons, program locations $FFF6–$FFFD even if they are not used for vectors.

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PTA0. If the received bytes match those at locations $FFF6–$FFFD, the host bypasses the security feature and can read all FLASH locations and execute code from FLASH. Security remains bypassed until a power-on reset occurs. If the reset was not a power-on reset, security remains bypassed and security code entry is not required. See Figure 16-18.

Upon power-on reset, if the received bytes of the security code do not match the data at locations $FFF6–$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. After receiving the eight security bytes from the host, the MCU transmits a break character, signifying that it is ready to receive a command.

NOTE

The MCU does not transmit a break character until after the host sends the eight security bytes.



Notes:
1 = Echo delay, approximately 2 bit times
2 = Data return delay, approximately 2 bit times
4 = Wait 1 bit time before sending next byte
5 = Wait until the monitor ROM runs

Figure 16-18. Monitor Mode Entry Timing

To determine whether the security code entered is correct, check to see if bit 6 of RAM address $80 is set. If it is, then the correct security code has been entered and FLASH can be accessed.

If the security sequence fails, the device should be reset by a power-on reset and brought up in monitor mode to attempt another entry. After failing the security sequence, the FLASH module can also be mass erased by executing an erase routine that was downloaded into internal RAM. The mass erase operation clears the security code locations so that all eight security bytes become $FF (blank).

Chapter 17
Electrical Specifications

17.1 Introduction

This section contains electrical and timing specifications.

17.2 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 17.5 5-V DC Electrical Characteristics and 17.8 3.3-V DC Electrical Characteristics for guaranteed operating conditions.

Characteristic[1]	Symbol	Value	Unit
Supply voltage	V_{DD}	−0.3 to +6.0	V
Input voltage	V_{IN}	V_{SS} −0.3 to V_{DD} +0.3	V
Mode entry voltage, $\overline{\text{IRQ}}$ 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 $V_{SS} \leq (V_{IN}$ 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}.)

17.3 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}	3.0 to 5.5	V	—

17.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance 16-pin SOIC 16-pin TSSOP	θ_{JA}	90 133	°C/W
I/O pin power dissipation	$P_{I/O}$	User determined	W
Power dissipation(1)	P_D	$P_D = (I_{DD} \times V_{DD}) + P_{I/O} = K/(T_J + 273°C)$	W
Constant(2)	K	$P_D \times (T_A + 273°C) + P_D^2 \times \theta_{JA}$	W/°C
Average junction temperature	T_J	$T_A + (P_D \times \theta_{JA})$	°C
Maximum junction temperature	T_{JM}	150	°C

1. Power dissipation is a function of temperature.
2. K constant unique to the device. K can be determined for a known T_A and measured $P_{D.}$ With this value of K, P_D and T_J can be determined for any value of T_A.

17.5 5-V DC Electrical Characteristics

Characteristic(1)	Symbol	Min	Typ(2)	Max	Unit
Output high voltage I_{Load} = –2.0 mA, all I/O pins I_{Load} = –10.0 mA, all I/O pins I_{Load} = –15.0 mA, PTA0, PTA1, PTA3–PTA5 only	V_{OH}	V_{DD}–0.4 V_{DD}–1.5 V_{DD}–0.8	— — —	— — —	V
Maximum combined I_{OH} (all I/O pins)	I_{OHT}	—	—	50	mA
Output low voltage I_{Load} = 1.6 mA, all I/O pins I_{Load} = 10.0 mA, all I/O pins I_{Load} = 15.0 mA, PTA0, PTA1, PTA3–PTA5 only	V_{OL}	— — —	— — —	0.4 1.5 0.8	V
Maximum combined I_{OL} (all I/O pins)	I_{OHL}	—	—	50	mA
Input high voltage PTA0–PTA5, PTB0–PTB7	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V

Table continued on next page.

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Input low voltage PTA0–PTA5, PTB0–PTB7	V_{IL}	V_{SS}	—	0.3 x V_{DD}	V
Input hysteresis[3]	V_{HYS}	0.06 x V_{DD}	—	—	V
DC injection current[3] [4] [5] [6] 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 0 0	 — — — —	 2 –0.2 25 –5	mA
Ports Hi-Z leakage current	I_{IL}	0	—	±1	µA
Capacitance Ports (as input)[3]	C_{IN}	—	—	8	pF
POR rearm voltage	V_{POR}	750	—	—	mV
POR rise time ramp rate[3][7]	R_{POR}	0.035	—	—	V/ms
Monitor mode entry voltage [3]	V_{TST}	V_{DD} + 2.5	—	9.1	V
Pullup resistors[8] PTA0–PTA5, PTB0–PTB7	R_{PU}	16	26	36	kΩ
Pulldown resistors[9] PTA0–PTA5	R_{PD}	16	26	36	kΩ
Low-voltage inhibit reset, trip falling voltage	V_{TRIPF}	3.90	4.20	4.50	V
Low-voltage inhibit reset, trip rising voltage	V_{TRIPR}	4.00	4.30	4.60	V
Low-voltage inhibit reset/recover hysteresis	V_{HYS}	—	100	—	mV

1. V_{DD} = 4.5 to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted.
2. Typical values reflect average measurements at midpoint of voltage range, 25•C only.
3. This parameter is characterized and not tested on each device.
4. All functional non-supply pins are internally clamped to V_{SS} and V_{DD}.
5. 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.
6. 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).
7. 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.
8. R_{PU} is measured at V_{DD} = 5.0 V. Pullup resistors only available when PTAPUEx is enabled with KBIPx = 0.
9. R_{PD} is measured at V_{DD} = 5.0 V, Pulldown resistors only available when PTAPUEx is enabled with KBIPx =1.

17.6 Typical 5-V Output Drive Characteristics

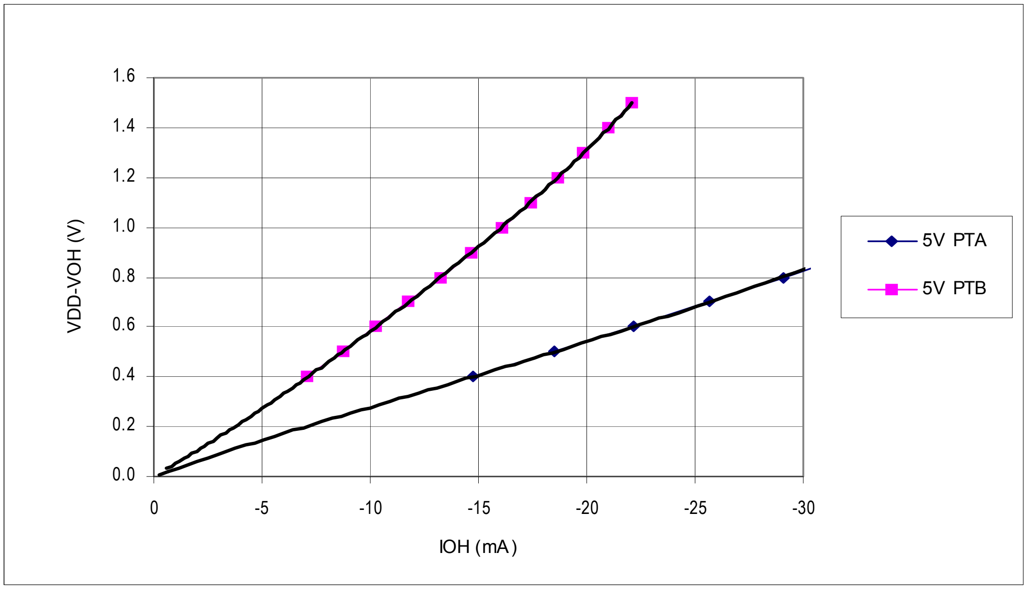


Figure 17-1. Typical 5-Volt Output High Voltage versus Output High Current (25•C)

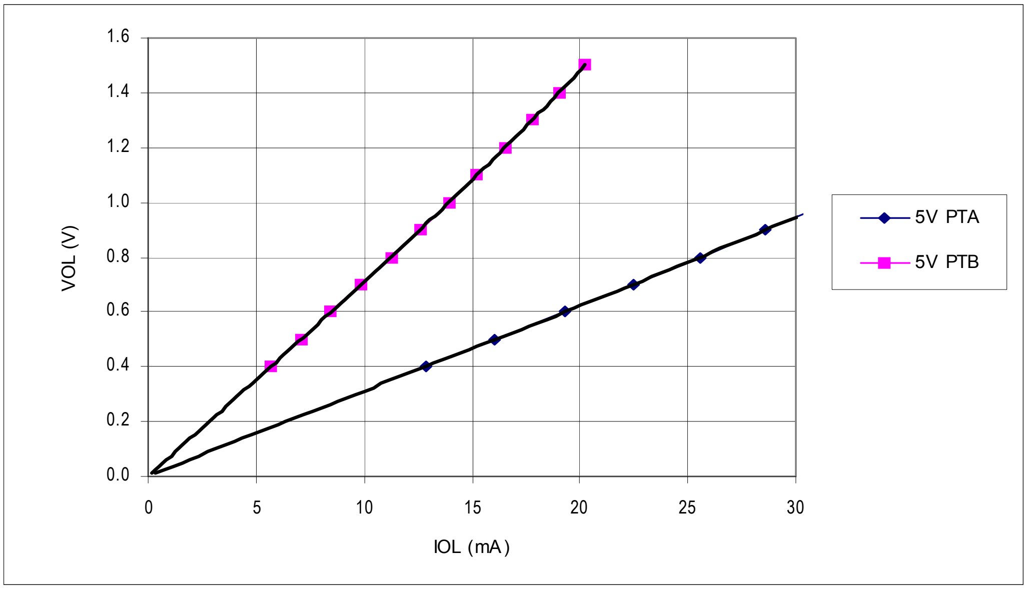


Figure 17-2. Typical 5-Volt Output Low Voltage versus Output Low Current (25•C)

17.7 5-V Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Internal operating frequency	f_{OP} (f_{BUS})	—	8	MHz
Internal clock period (1/f_{OP})	t_{cyc}	125	—	ns
$\overline{RST}$ input pulse width low[2]	t_{RL}	100	—	ns
$\overline{IRQ}$ interrupt pulse width low (edge-triggered)[2]	t_{ILIH}	100	—	ns
$\overline{IRQ}$ interrupt pulse period[2]	t_{ILIL}	Note[3]	—	t_{cyc}

[1] V_{DD} = 4.5 to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H; timing shown with respect to 20% V_{DD} and 70% V_{SS}, unless otherwise noted.

[2] Values are based on characterization results, not tested in production.

[3] The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 t_{cyc}.

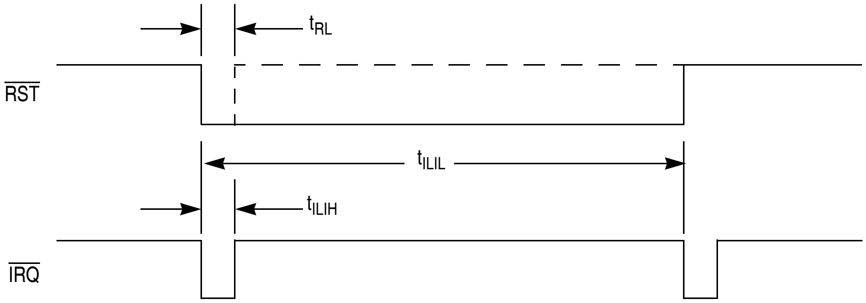


Figure 17-3. $\overline{RST}$ and $\overline{IRQ}$ Timing

17.8 3.3-V DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Output high voltage I_{Load} = –0.6 mA, all I/O pins I_{Load} = –4.0 mA, all I/O pins I_{Load} = –10.0 mA, PTA0, PTA1, PTA3–PTA5 only	V_{OH}	 V_{DD}−0.3 V_{DD}−1.0 V_{DD}−0.8	 — — —	 — — —	V
Maximum combined I_{OH} (all I/O pins)	I_{OHT}	—	—	50	mA
Output low voltage I_{Load} = 0.5 mA, all I/O pins I_{Load} = 6.0 mA, all I/O pins I_{Load} = 10.0 mA, PTA0, PTA1, PTA3–PTA5 only	V_{OL}	 — — —	 — — —	 0.3 1.0 0.8	V
Maximum combined I_{OL} (all I/O pins)	I_{OHL}	—	—	50	mA
Input high voltage PTA0–PTA5, PTB0–PTB7	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input low voltage PTA0–PTA5, PTB0–PTB7	V_{IL}	V_{SS}	—	0.3 x V_{DD}	V

— Continued on next page

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Input hysteresis[3]	V_{HYS}	0.06 x V_{DD}	—	—	V
DC injection current[3] [4] [5] [6] 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 0 0	— — — —	2 –0.2 25 –5	mA
Ports Hi-Z leakage current	I_{IL}	0	—	±1	μA
Capacitance Ports (as input)[3]	C_{IN}	—	—	8	pF
POR rearm voltage	V_{POR}	0.75	—	—	V
POR rise time ramp rate[3][7]	R_{POR}	0.035	—	—	V/ms
Monitor mode entry voltage [3]	V_{TST}	V_{DD} + 2.5	—	V_{DD} + 4.0	V
Pullup resistors[8] PTA0–PTA5, PTB0–PTB7	R_{PU}	16	26	36	kΩ
Pulldown resistors[9] PTA0–PTA5	R_{PD}	16	26	36	kΩ
Low-voltage inhibit reset, trip falling voltage	V_{TRIPF}	2.65	2.8	3.0	V
Low-voltage inhibit reset, trip rising voltage	V_{TRIPR}	2.75	2.9	3.10	V
Low-voltage inhibit reset/recover hysteresis	V_{HYS}	—	100	—	mV

1. V_{DD} = 3.0 to 3.6 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted.
2. Typical values reflect average measurements at midpoint of voltage range, 25•C only.
3. This parameter is characterized and not tested on each device.
4. All functional non-supply pins are internally clamped to V_{SS} and V_{DD}.
5. 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.
6. 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).
7. 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.
8. R_{PU} is measured at V_{DD} = 3.3 V. Pullup resistors only available when PTAPUEx is enabled with KBIPx = 0.
9. R_{PD} is measured at V_{DD} = 3.3 V, Pulldown resistors only available when PTAPUEx is enabled with KBIPx =1.

17.9 Typical 3.3-V Output Drive Characteristics

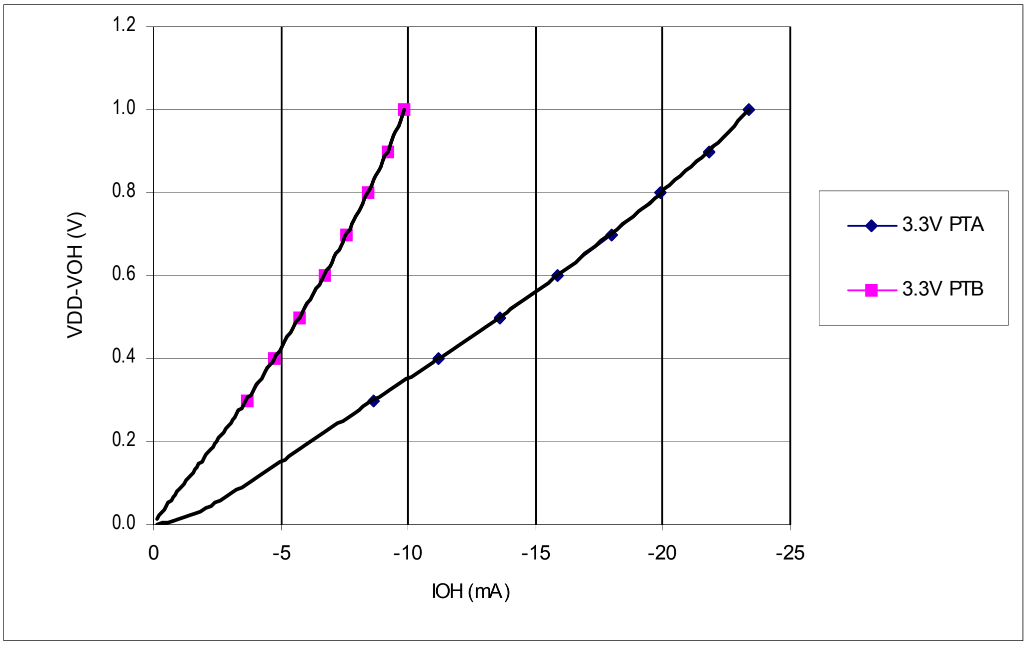


Figure 17-4. Typical 3.3-Volt Output High Voltage versus Output High Current (25•C)

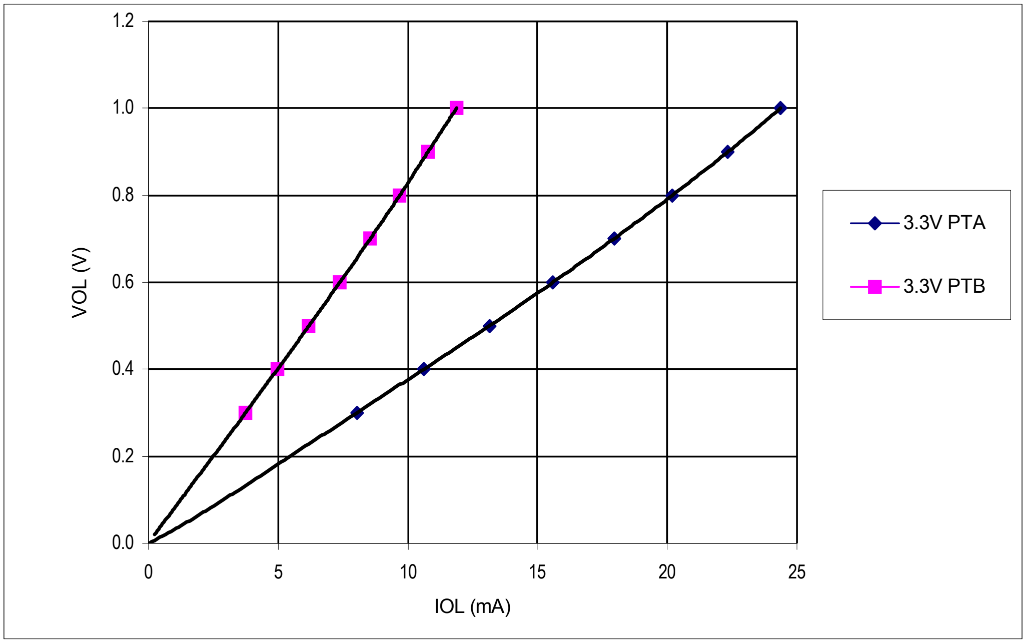


Figure 17-5. Typical 3.3-Volt Output Low Voltage versus Output Low Current (25•C)

17.10 3.3-V Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Internal operating frequency	f_{OP} (f_{Bus})	—	4	MHz
Internal clock period (1/f_{OP})	t_{cyc}	250	—	ns
$\overline{\text{RST}}$ input pulse width low[2]	t_{RL}	200	—	ns
$\overline{\text{IRQ}}$ interrupt pulse width low (edge-triggered)[2]	t_{ILIH}	200	—	ns
$\overline{\text{IRQ}}$ interrupt pulse period[2]	t_{ILIL}	Note[3]	—	t_{cyc}

1. V_{DD} = 3.0 to 3.6 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H; timing shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted.
2. Values are based on characterization results, not tested in production.
3. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 t_{cyc}.

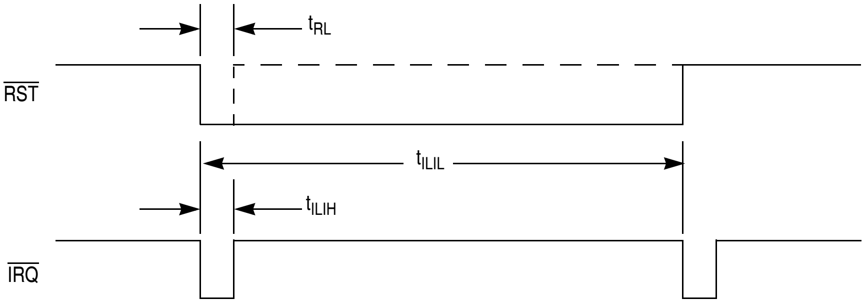


Figure 17-6. $\overline{\text{RST}}$ and $\overline{\text{IRQ}}$ Timing

17.11 Oscillator Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
Internal oscillator frequency[1]	f_{INTCLK}				MHz
ICFS1:ICFS0 = 00		—	4	—	
ICFS1:ICFS0 = 01		—	8	—	
ICFS1:ICFS0 = 10 (not allowed if $V_{DD} < 3.0V$)		—	12.8	—	
ICFS1:ICFS0 = 11 (not allowed if $V_{DD} < 4.5V$)		—	25.6	—	
Deviation from trimmed Internal oscillator [2][3]	ACC_{INT}				%
4, 8, 12.8, 25.6MHz, fixed voltage, fixed temp		—	± 0.4	—	
4, 8, 12.8MHz, V_{DD} ± 10%, 0 to 70°C		—	± 2	—	
4, 8, 12.8MHz, V_{DD} ± 10%, –40 to 85°C		—	—	± 5	
4, 8, 12.8MHz, V_{DD} ± 10%, –40 to 105°C		—	—	± 5	
4, 8, 12.8MHz, V_{DD} ± 10%, –40 to 125°C		—	—	± 5	
25.6MHz, V_{DD} ± 10%, 0 to 70°C		—	± 5	—	
25.6MHz, V_{DD} ± 10%, –40 to 85°C		—	—	± 10	
25.6MHz, V_{DD} ± 10%, –40 to 105°C		—	—	± 10	
25.6MHz, V_{DD} ± 10%, –40 to 125°C		—	—	± 10	

— Continued on next page

Characteristic	Symbol	Min	Typ	Max	Unit
External RC oscillator frequency, RCCLK [1][2] $V_{DD} \geq 3.0$ V $V_{DD} < 3.0$ V	f_{RCCLK}	 2 2	 — —	 10 8.4	 MHz
External clock reference frequencyy[1][4][5] $V_{DD} \geq 4.5$V $V_{DD} \geq 3.0$V $V_{DD} < 3.0$V	$f_{OSCXCLK}$	 dc dc dc	— — —	 32 16 8.4	 MHz
RC oscillator external resistor $V_{DD} = 5$ V $V_{DD} = 3$ V	R_{EXT}	 See Figure 17-7 See Figure 17-8			—
Crystal frequency, XTALCLK[1][6][7] ECFS1:ECFS0 = 00 ($V_{DD} \geq 4.5$ V) ECFS1:ECFS0 = 00 (not allowed if $V_{DD} < 3.0$V) ECFS1:ECFS0 = 01 ECFS1:ECFS0 = 10	$f_{OSCXCLK}$	 8 8 1 30	 — — — —	 32 16 8 100	 MHz MHz MHz kHz
ECFS1:ECFS0 = 00 [8] Feedback bias resistor Crystal load capacitance[9] Crystal capacitors[9]	 R_B C_L C_1,C_2	 — — —	 1 20 (2 x C_L) – 5pF	 — — —	 MΩ pF pF
ECFS1:ECFS0 = 01[8] Crystal series damping resistor $f_{OSCXCLK}$ = 1 MHz $f_{OSCXCLK}$ = 4 MHz $f_{OSCXCLK}$ = 8 MHz Feedback bias resistor Crystal load capacitance[9] Crystal capacitors[9]	 R_S R_B C_L C_1,C_2	 — — — — — —	 20 10 0 5 18 (2 x C_L) –10 pF	 — — — — — —	 kΩ kΩ kΩ MΩ pF pF
AWU Module: Internal RC oscillator frequency	f_{INTRC}	—	32	—	kHz

[1] Bus frequency, f_{OP}, is oscillator frequency divided by 4.

[2] Deviation values assumes trimming @25•C and midpoint of voltage range, for example 5.0 V for 5 V ± 10%, 3.3 V for 3.3 V ± 10%.

[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 = 00, 10 or 11

[9] Consult crystal vendor data sheet.

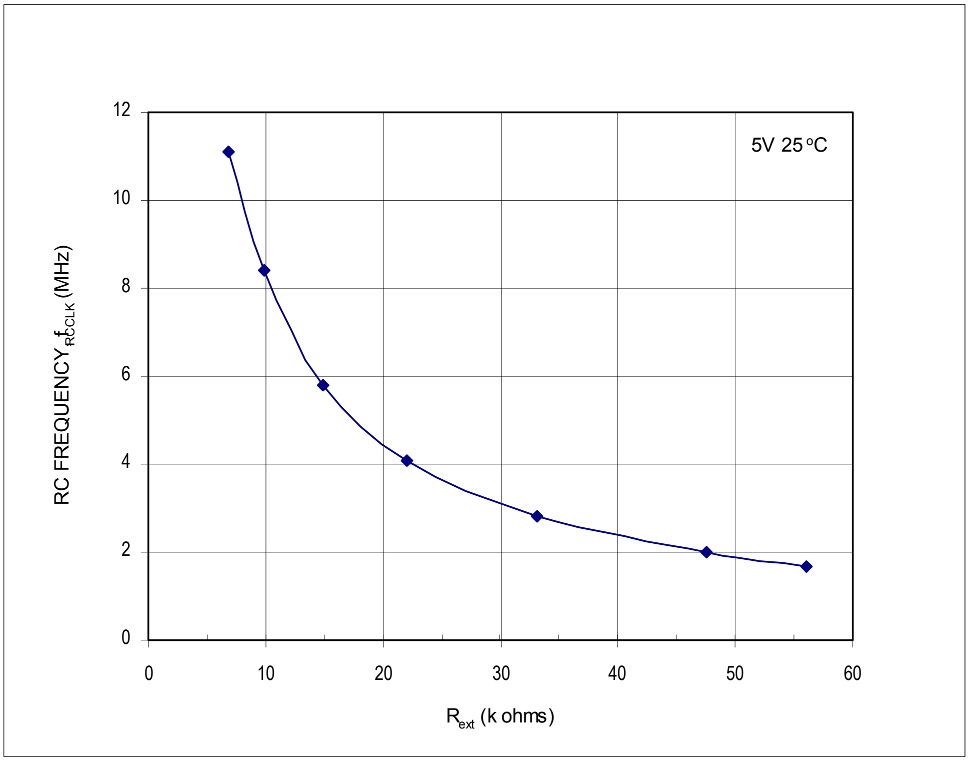


Figure 17-7. RC versus Frequency (5 Volts @ 25•C)

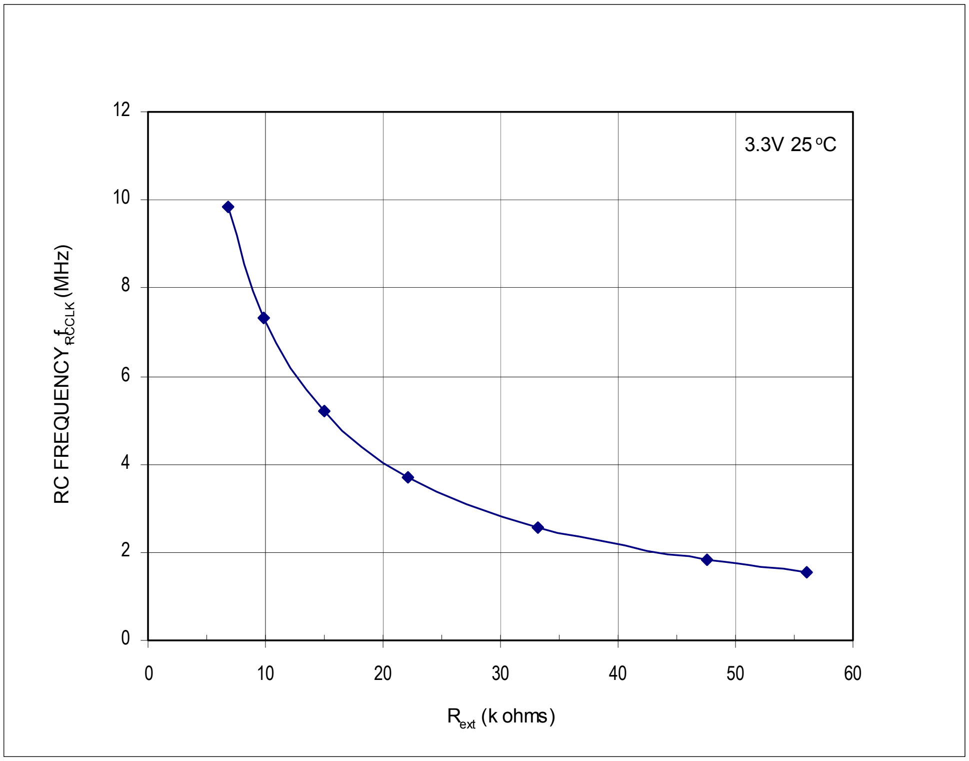


Figure 17-8. RC versus Frequency (3.3 Volts @ 25•C)

17.12 Supply Current Characteristics

Characteristic[1]	Voltage	Bus Frequency (MHz)	Symbol	Typ[2]	Max	Unit
Run mode V_{DD} supply current[3]	5.0 3.3	3.2 3.2	RI_{DD}	7.25 3.2	8.5 4.5	mA
Wait mode V_{DD} supply current[4]	5.0 3.3	3.2 3.2	WI_{DD}	1.25 0.80	1.8 1.2	mA
Stop mode V_{DD} supply current[5]	5.0		SI_{DD}			
–40 to 85°C				0.26	1	µA
–40 to 105°C				—	2	µA
–40 to 125°C				—	5	µA
25°C with auto wake-up enabled				12	—	µA
Incremental current with LVI enabled at 25°C				125	—	µA
Stop mode V_{DD} supply current[4]	3.3					
–40 to 85°C				0.23	0.5	µA
–40 to 105°C				—	1	µA
–40 to 125°C				—	4	µA
25°C with auto wake-up enabled				2	—	µA
Incremental current with LVI enabled at 25°C				100	—	µA

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.

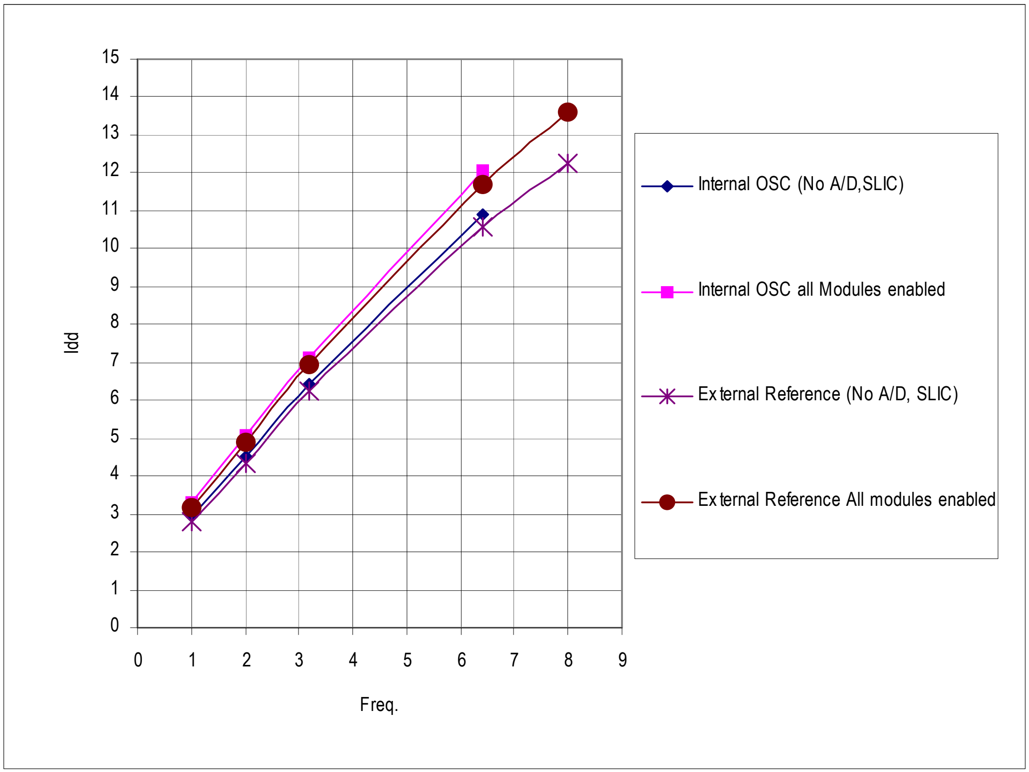


Figure 17-9. Typical 5-Volt Run Current versus Bus Frequency (25•C)

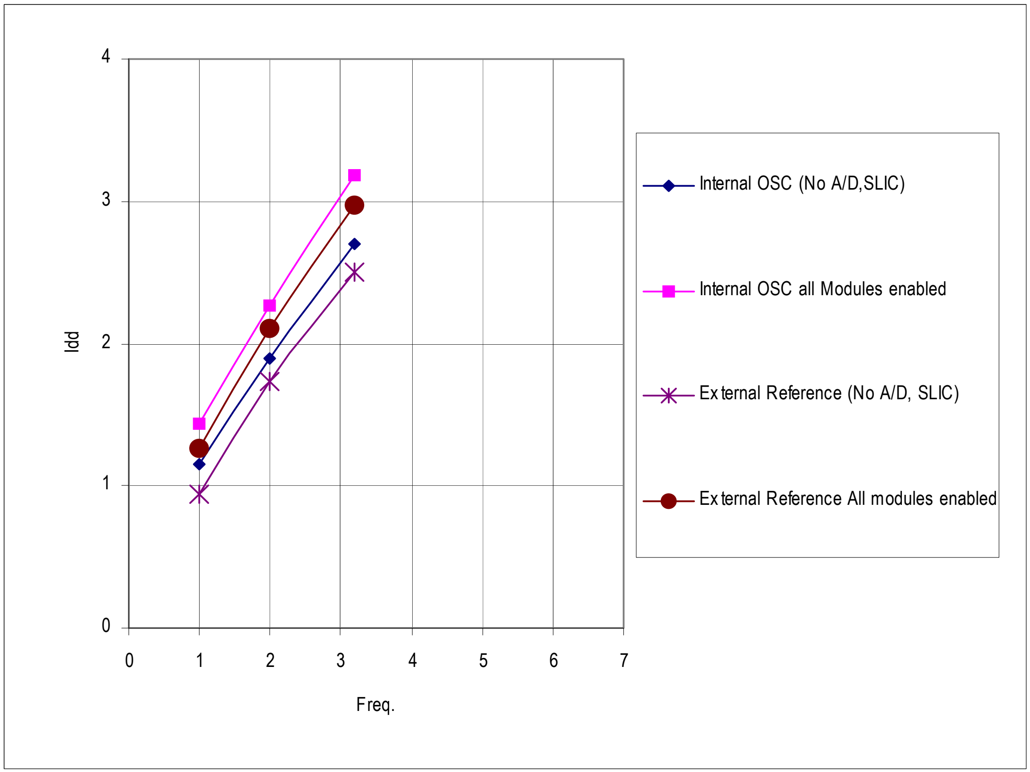


Figure 17-10. Typical 3.3-Volt Run Current versus Bus Frequency (25•C)

17.13 ADC10 Characteristics

Characteristic	Conditions	Symbol	Min	Typ(1)	Max	Unit	Comment
Supply voltage	Absolute	V_{DD}	3.0	—	5.5	V	
Supply Current ADLPC = 1 ADLSMP = 1 ADCO = 1	$V_{DD} \leq 3.6$ V (3.3 V Typ)	I_{DD}(2)	—	55	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	75	—		
Supply current ADLPC = 1 ADLSMP = 0 ADCO = 1	$V_{DD} \leq 3.6$ V (3.3 V Typ)	I_{DD}(2)	—	120	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	175	—		
Supply current ADLPC = 0 ADLSMP = 1 ADCO = 1	$V_{DD} \leq 3.6$ V (3.3 V Typ)	I_{DD}(2)	—	140	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	180	—		
Supply current ADLPC = 0 ADLSMP = 0 ADCO = 1	$V_{DD} \leq 3.6$ V (3.3 V Typ)	I_{DD}(2)	—	340	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	440	615		
ADC internal clock	High speed (ADLPC = 0)	f_{ADCK}	0.40(3)	—	2.00	MHz	$t_{ADCK} = 1/f_{ADCK}$
	Low power (ADLPC = 1)		0.40(3)	—	1.00		
Conversion time (4) 10-bit Mode	Short sample (ADLSMP = 0)	t_{ADC}	19	19	21	t_{ADCK} cycles	
	Long sample (ADLSMP = 1)		39	39	41		
Conversion time (4) 8-bit Mode	Short sample (ADLSMP = 0)	t_{ADC}	16	16	18	t_{ADCK} cycles	
	Long sample (ADLSMP = 1)		36	36	38		
Sample time	Short sample (ADLSMP = 0)	t_{ADS}	4	4	4	t_{ADCK} cycles	
	Long sample (ADLSMP = 1)		24	24	24		
Input voltage		V_{ADIN}	V_{SS}	—	V_{DD}	V	
Input capacitance		C_{ADIN}	—	7	10	pF	Not tested
Input impedance		R_{ADIN}	—	5	15	kΩ	Not tested
Analog source impedance		R_{AS}	—	—	10	kΩ	External to MCU
Ideal resolution (1 LSB)	10-bit mode	RES	1.758	5	5.371	mV	$V_{REFH}/2^N$
	8-bit mode		7.031	20	21.48		
Total unadjusted error	10-bit mode	E_{TUE}	0	±1.5	±2.5	LSB	Includes quantization
	8-bit mode		0	±0.7	±1.0		
Differential non-linearity	10-bit mode	DNL	0	±0.5	—	LSB	
	8-bit mode		0	±0.3	—		
	Monotonicity and no-missing-codes guaranteed						

— Continued on next page

Characteristic	Conditions	Symbol	Min	Typ[1]	Max	Unit	Comment
Integral non-linearity	10-bit mode	INL	0	±0.5	—	LSB	
	8-bit mode		0	±0.3	—		
Zero-scale error	10-bit mode	E_{ZS}	0	±0.5	—	LSB	$V_{ADIN} = V_{SS}$
	8-bit mode		0	±0.3	—		
Full-scale error	10-bit mode	E_{FS}	0	±0.5	—	LSB	$V_{ADIN} = V_{DD}$
	8-bit mode		0	±0.3	—		
Quantization error	10-bit mode	E_Q	—	—	±0.5	LSB	8-bit mode is not truncated
	8-bit mode		—	—	±0.5		
Input leakage error	10-bit mode	E_{IL}	0	±0.2	±5	LSB	Pad leakage[5] * R_{AS}
	8-bit mode		0	±0.1	±1.2		
Bandgap voltage input[6]		V_{BG}	1.17	1.245	1.32	V	

1. Typical values assume V_{DD} = 5.0 V, temperature = 25•C, f_{ADCK} = 1.0 MHz unless otherwise stated. Typical values are for reference only and are not tested in production.
2. Incremental I_{DD} added to MCU mode current.
3. Values are based on characterization results, not tested in production.
4. Reference the ADC module specification for more information on calculating conversion times.
5. Based on typical input pad leakage current.
6. LVI must be enabled, (LVIPWRD = 0, in CONFIG1). Voltage input to ADCH4:0 = $1A, an ADC conversion on this channel allows user to determine supply voltage.

17.14 Timer Interface Module Characteristics

Characteristic	Symbol	Min	Max	Unit
Timer input capture pulse width[1]	t_{TH}, t_{TL}	2	—	t_{cyc}
Timer input capture period	t_{TLTL}	Note[2]	—	t_{cyc}
Timer input clock pulse width[1]	t_{TCL}, t_{TCH}	t_{cyc} + 5	—	ns

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

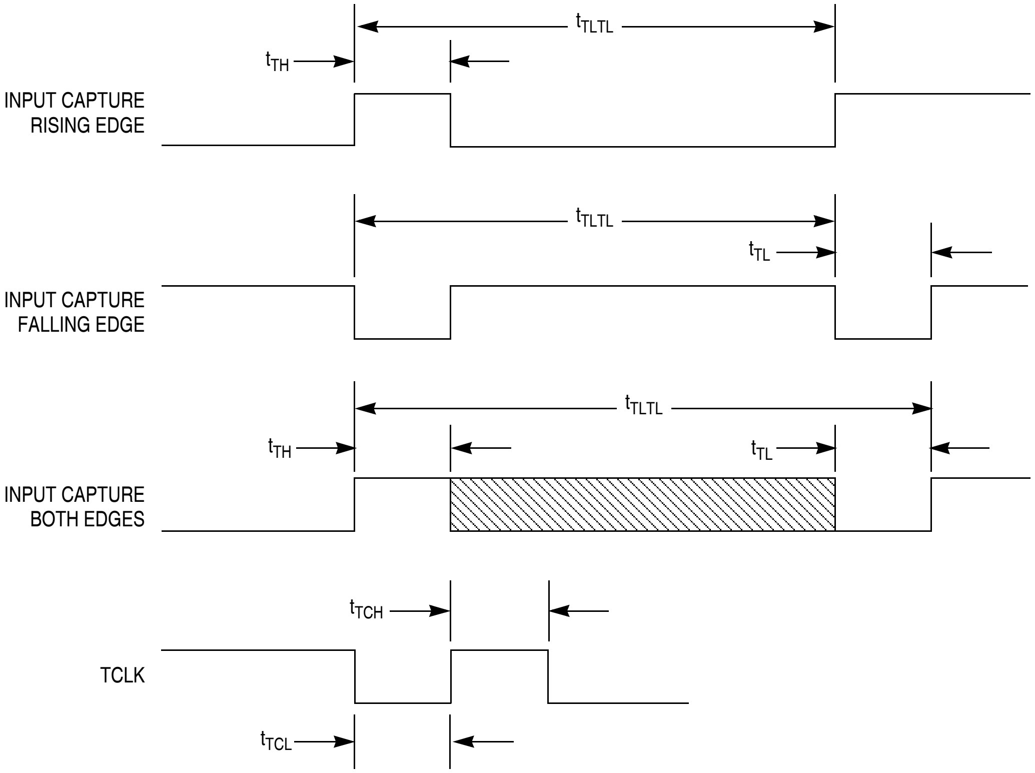


Figure 17-11. Input Capture Timing

17.15 Memory Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
RAM data retention voltage [1]	V_{RDR}	1.3	—	—	V
FLASH program bus clock frequency	—	1	—	—	MHz
FLASH PGM/ERASE supply voltage (V_{DD})	$V_{PGM/ERASE}$	2.7	—	5.5	V
FLASH read bus clock frequency	f_{Read}[2]	0	—	8 M	Hz
FLASH page erase time <1 K cycles >1 K cycles	t_{Erase}	 0.9 3.6	 1 4	 1.1 5.5	ms
FLASH mass erase time	t_{MErase}	4	—	—	ms
FLASH PGM/ERASE to HVEN setup time	t_{NVS}	10	—	—	μs
FLASH high-voltage hold time	t_{NVH}	5	—	—	μs
FLASH high-voltage hold time (mass erase)	t_{NVHL}	100	—	—	μs
FLASH program hold time	t_{PGS}	5	—	—	μs
FLASH program time	t_{PROG}	30	—	40	μs
FLASH return to read time	t_{RCV}[3]	1	—	—	μs
FLASH cumulative program hv period	t_{HV}[4]	—	—	4	ms
FLASH endurance[5]	—	10 k	100 k	—	Cycles
FLASH data retention time[6]	—	15	100	—	Years

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

[2] f_{Read} is defined as the frequency range for which the FLASH memory can be read.

[3] t_{RCV} is defined as the time it needs before the FLASH can be read after turning off the high voltage charge pump, by clearing HVEN to 0.

[4] t_{HV} is defined as the cumulative high voltage programming time to the same row before next erase. t_{HV} must satisfy this condition: $t_{NVS} + t_{NVH} + t_{PGS} + (t_{PROG} \text{ x } 32) \leq t_{HV}$ maximum.

[5] Typical endurance was evaluated for this product family. For additional information on how Freescale Semiconductor defines *Typical Endurance*, please refer to Engineering Bulletin EB619.

[6] Typical data retention values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25•C using the Arrhenius equation. For additional information on how Freescale Semiconductor defines *Typical Data Retention*, please refer to Engineering Bulletin EB618.

Chapter 18
Ordering Information and Mechanical Specifications

18.1 Introduction

This section provides ordering information for the MC68HC908QL4 along with the dimensions for:

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

18.2 MC Order Numbers

Table 18-1. MC Order Numbers

MC Order Number	ADC	FLASH Memory	Package
MC908QL4	Yes	4096 bytes	16-pins SOIC, and TSSOP

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)

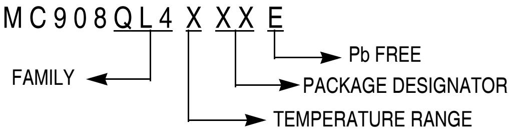
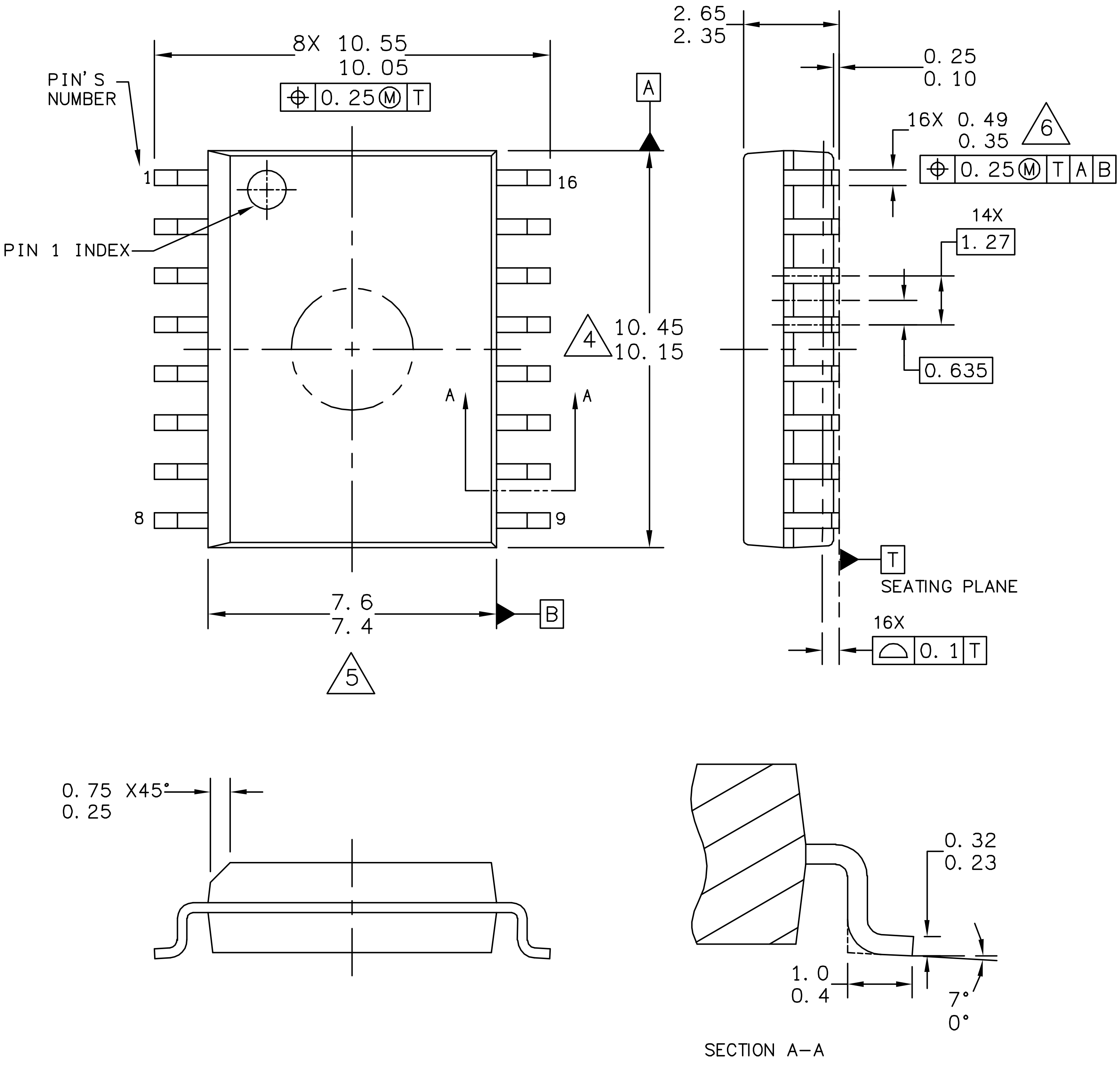


Figure 18-1. Device Numbering System

18.3 Package Dimensions

Refer to the following pages for detailed package dimensions.



© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	**MECHANICAL OUTLINE**	PRINT VERSION NOT TO SCALE
TITLE: 16LD SOIC W/B, 1.27 PITCH CASE–OUTLINE	DOCUMENT NO: 98ASB42567B	REV: F
	CASE NUMBER: 751G–04	02 JUN 2005
	STANDARD: JEDEC MS–013AA	

NOTES:

1. DIMENSIONS ARE IN MILLIMETERS.

2. DIMENSIONING AND TOLERANCING PER ASME Y14.5M–1994.

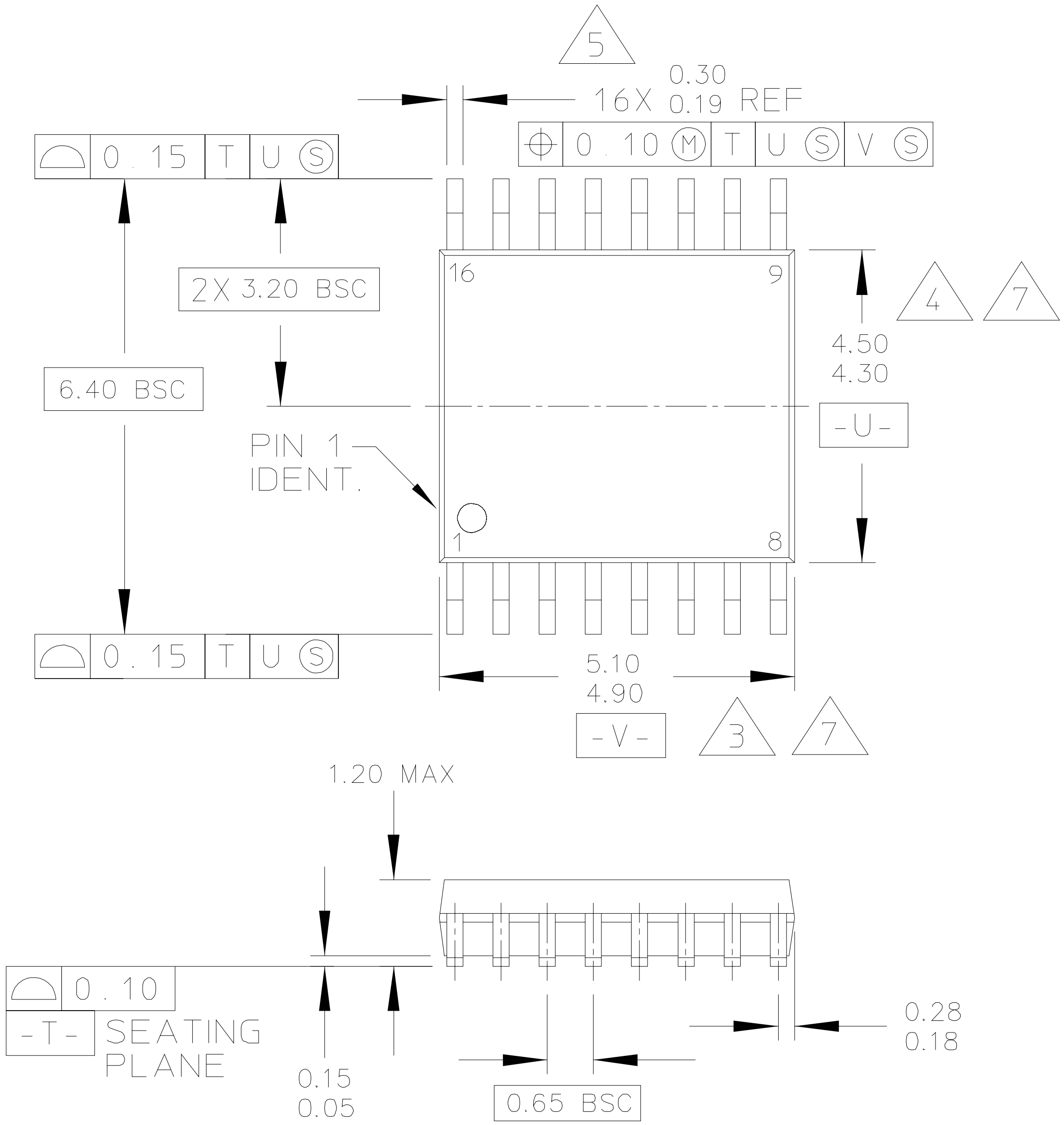
3. DATUMS A AND B TO BE DETERMINED AT THE PLANE WHERE THE BOTTOM OF THE LEADS EXIT THE PLASTIC BODY.

△4. THIS DIMENSION DOES NOT INCLUDE MOLD FLASH, PROTRUSION OR GATE BURRS. MOLD FLASH, PROTRUSION OR GATE BURRS SHALL NOT EXCEED 0.15 MM PER SIDE. THIS DIMENSION IS DETERMINED AT THE PLANE WHERE THE BOTTOM OF THE LEADS EXIT THE PLASTIC BODY.

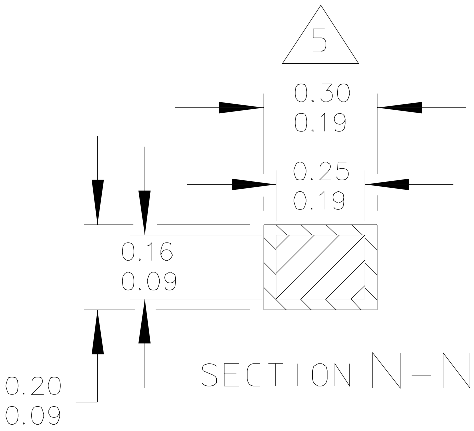
△5. THIS DIMENSION DOES NOT INCLUDE INTER–LEAD FLASH OR PROTRUSIONS. INTER–LEAD FLASH AND PROTRUSIONS SHALL NOT EXCEED 0.25 MM PER SIDE. THIS DIMENSION IS DETERMINED AT THE PLANE WHERE THE BOTTOM OF THE LEADS EXIT THE PLASTIC BODY.

△6. THIS DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED 0.62 mm.

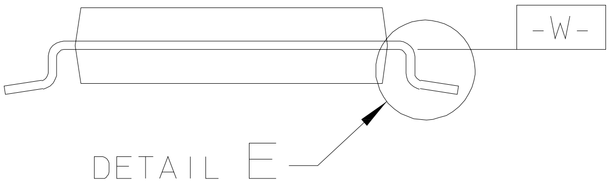
MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: 16LD SOIC W/B, 1.27 PITCH, CASE OUTLINE	DOCUMENT NO: 98ASB42567B	REV: F
	CASE NUMBER: 751G–04	02 JUN 2005
	STANDARD: JEDEC MS–013AA	

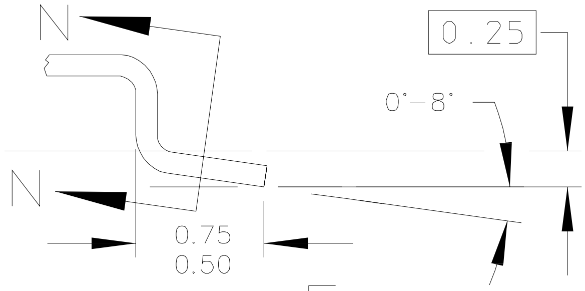


© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: 16 LD TSSOP, PITCH 0.65MM	DOCUMENT NO: 98ASH70247A	REV: B
	CASE NUMBER: 948F-01	19 MAY 2005
	STANDARD: JEDEC	



SECTION N-N





DETAIL E

	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: 16 LD TSSOP, PITCH 0.65MM	DOCUMENT NO: 98ASH70247A	REV: B
	CASE NUMBER: 948F-01	19 MAY 2005
	STANDARD: JEDEC	

NOTES:

1. CONTROLLING DIMENSION: MILLIMETER

2. DIMENSIONS AND TOLERANCES PER ANSI Y14.5M-1982.

△3 DIMENSION DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS. MOLD FLASH OR GATE BURRS SHALL NOT EXCEED 0.15 PER SIDE.

△4 DIMENSION DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION. INTERLEAD FLASH OR PROTRUSION SHALL NOT EXCEED 0.25 PER SIDE.

△5 DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 TOTAL IN EXCESS OF THE DIMENSION AT MAXIMUM MATERIAL CONDITION.

6. TERMINAL NUMBERS ARE SHOWN FOR REFERENCE ONLY.

△7 DIMENSIONS ARE TO BE DETERMINED AT DATUM PLANE -W-.

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: 16 LD TSSOP, PITCH 0.65MM	DOCUMENT NO: 98ASH70247A	REV: B
	CASE NUMBER: 948F-01	19 MAY 2005
	STANDARD: JEDEC	

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

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

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

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

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

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

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

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

freescale™
semiconductor