MOTOROLA
intelligence everywhere™

digital dna™

MC68HC908EY16

Advance Information

M68HC08
Microcontrollers

MC68HC908EY16/D
Rev. 4.0, 2/2003

MOTOROLA.COM/SEMICONDUCTORS

MC68HC908EY16

Advance Information — Rev 4.0

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola 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 which may be provided in Motorola 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. Motorola does not convey any license under its patent rights nor the rights of others. Motorola 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 Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola 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 Motorola was negligent regarding the design or manufacture of the part. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

List of Paragraphs

Table of Contents

List of Paragraphs

Table of Contents

List of Figures

List of Tables

Section 1. General Description

Section 2. Memory Map

Section 3. Random Access Memory (RAM)

Section 4. FLASH Memory

Section 5. Central Processor Unit (CPU)

Section 6. System Integration Module (SIM)

Section 7. Internal Clock Generator (ICG) Module

Section 8. Configuration Registers (CONFIG1 & CONFIG2)

Section 9. Break Module (BRK)

Section 10. Monitor ROM (MON)

Section 11. Computer Operating Properly (COP) Module

Section 12. Low-Voltage Inhibit (LVI) Module

Section 13. External Interrupt (IRQ)

Section 14. Enhanced Serial Communications Interface (ESCI) Module

Section 15. Serial Peripheral Interface (SPI) Module

Section 16. Timer Interface A (TIMA) Module

Section 17. Timer Interface B (TIMB) Module

Section 18. BEMF Module

Section 19. Keyboard Interrupt (KBD) Module

Section 20. Timebase Module (TBM)

Section 21. Analog-to-Digital Converter (ADC) Module

Section 22. Input/Output (I/O) Ports

Section 23. Electrical Specifications

Section 24. Mechanical Specifications

Section 25. Ordering Information

Revision History

Glossary

Table of Contents

List of Figures

List of Figures

List of Figures

List of Figures

List of Tables

List of Tables

Section 1. General Description

1.1 Contents

1.2 Introduction

The MC68HC908EY16 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.

1.3 Features

For convenience, features have been organized to reflect:

- Standard features of the MC68HC908EY16
- Features of the CPU08

Standard features of the MC68HC908EY16 include:

- High-performance M68HC08 architecture optimized for C-compilers
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- 8-MHz internal bus frequency at 5V
- Internal oscillator requiring no external components:
  - Software selectable bus frequencies
  - 25 percent accuracy with trim capability to 2 percent
  - Clock monitor
  - Option to allow use of external clock source or external crystal/ceramic resonator
- 15,872 bytes of on-chip FLASH memory with in-circuit programming
- FLASH program memory security[1]
- 512 bytes of on-chip random-access memory (RAM)

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

- Low voltage inhibit (LVI) module
- Internal clock generator module (ICG)
- Two 16-bit, 2-channel timer (TIMA and TIMB) interface modules with selectable input capture, output compare, and pulse-width modulation (PWM) capability on each channel
- 8-channel, 10-bit successive approximation analog-to-digital converter (ADC)
- Enhanced serial communications interface module (ESCI) for local interconnect network (LIN) connectivity
- Serial peripheral interface (SPI)
- Timebase Module (TBM)
- 5-bit keyboard interrupt (KBI) with wakeup feature
- 24 general-purpose input/output (I/O) pins
- External asynchronous interrupt pin with internal pullup ($\overline{\text{IRQ}}$)
- System protection features:
  - Optional computer operating properly (COP) reset
  - Illegal opcode detection with reset
  - Illegal address detection with reset
- 32-pin quad flat pack (QFP) package
- Low-power design; fully static with stop and wait modes
- Internal pullups on $\overline{\text{IRQ}}$ and $\overline{\text{RST}}$ to reduce customer system cost
- Standard low-power modes of operation:
  - Wait mode
  - Stop mode
- Master reset pin ($\overline{\text{RST}}$) and power-on reset (POR)
- BREAK module (BRK) to allow single breakpoint setting during in-circuit debugging

- Higher current source capability on nine port lines for LED drive (PTA6/$\overline{SS}$, PTA5/SPSCK, PTA4/$\overline{KBD4}$, PTA3/$\overline{KBD3}$, PTA2/$\overline{KBD2}$, PTA1/$\overline{KBD1}$, PTA0/$\overline{KBD0}$, PTC1/MOSI, and PTC0/MISO)

Features of the CPU08 include:

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

1.4 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908EY16.

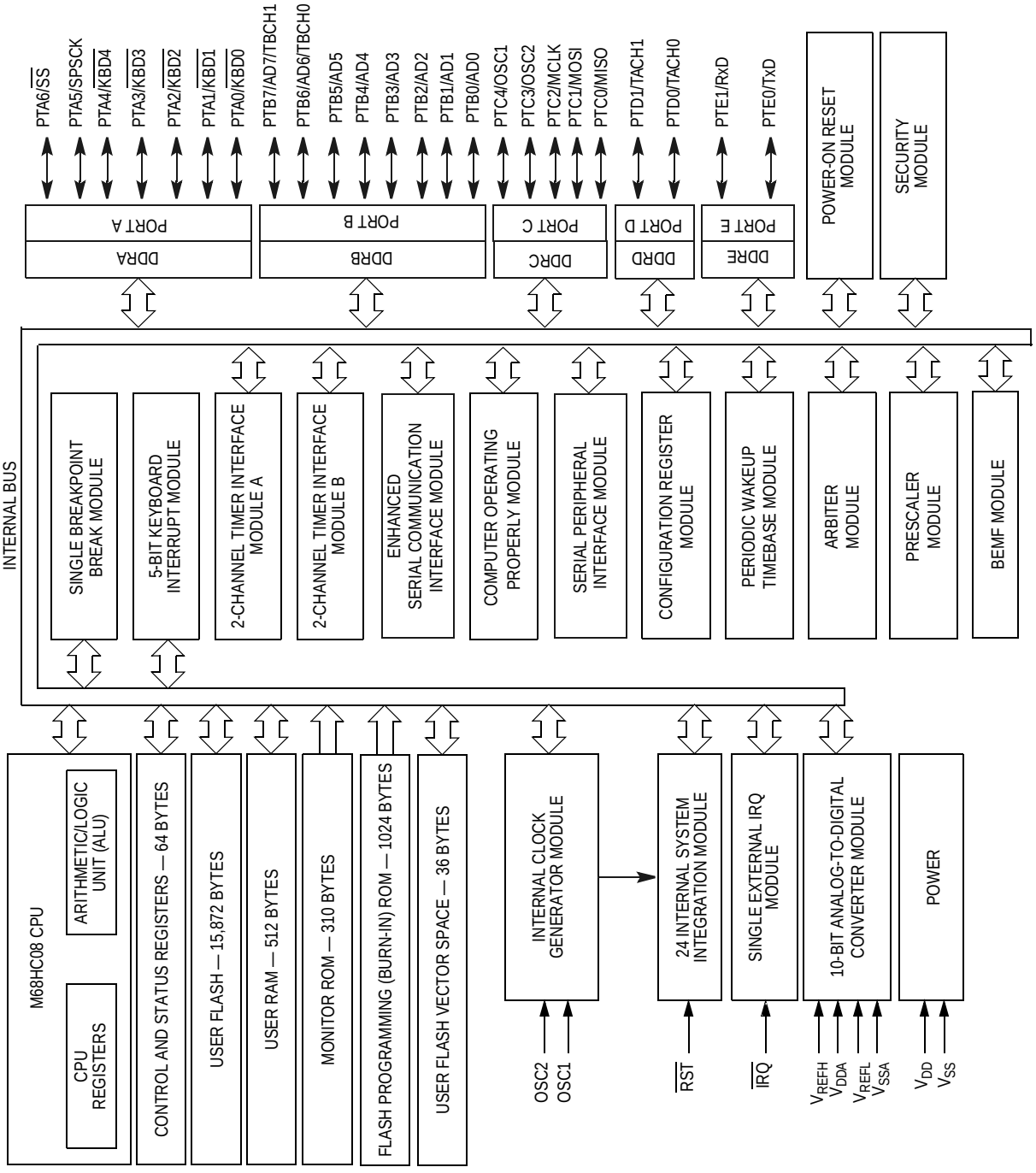


Figure 1-1. MCU Block Diagram

1.5 Pin Assignments

Figure 1-2 shows the pin assignments for the MC68HC908EY16.

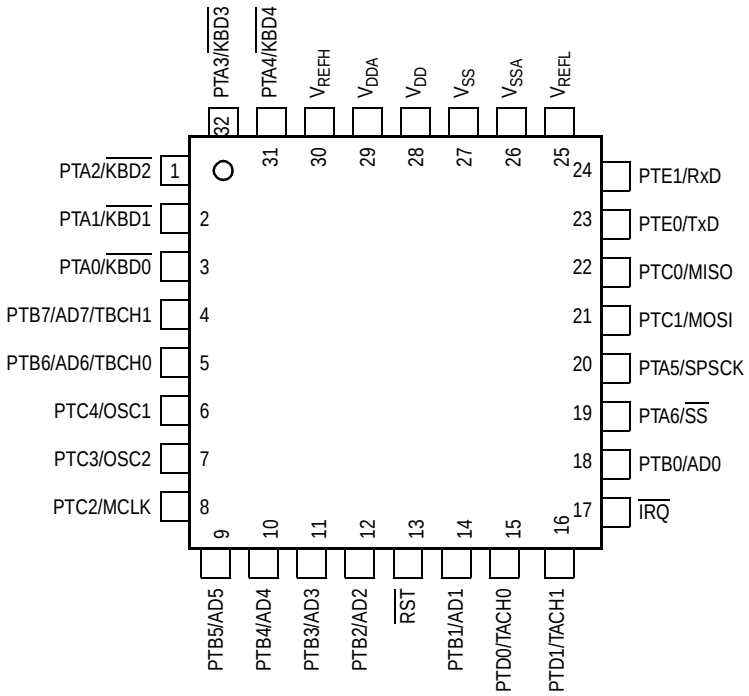


Figure 1-2. Pin Assignments

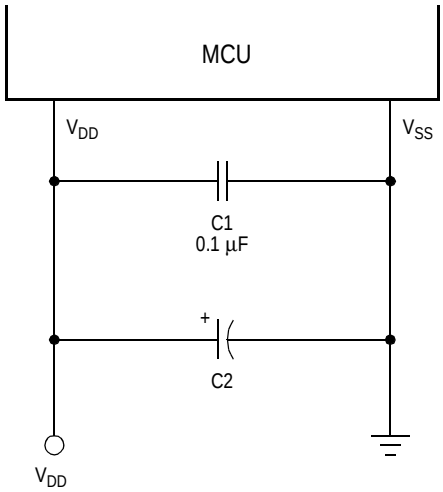
1.6 Pin Functions

Descriptions of the pin functions are provided here.

1.6.1 Power Supply Pins (V_{DD} and V_{SS})

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

Fast signal transitions on MCU pins place high, short-duration current demands on the power supply. To prevent noise problems, take special care to provide power supply bypassing at the MCU as Figure 1-3 shows. Place the C1 bypass capacitor as close to the MCU as possible. Use a high-frequency-response ceramic capacitor for C1. C2 is an optional bulk current bypass capacitor for use in applications that require the port pins to source high current levels.



Note: Component values shown represent typical applications.

Figure 1-3. Power Supply Bypassing

1.6.2 Oscillator Pins (PTC4/OSC1 and PTC3/OSC2)

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

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

1.6.3 External Reset Pin ($\overline{\text{RST}}$)

A logic 0 on the $\overline{\text{RST}}$ pin forces the MCU to a known startup state. $\overline{\text{RST}}$ is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted. This pin contains an internal pullup resistor that is always activated, even when the reset pin is pulled low. See **Section 6. System Integration Module (SIM)**.

1.6.4 External Interrupt Pin ($\overline{\text{IRQ}}$)

$\overline{\text{IRQ}}$ is an asynchronous external interrupt pin. This pin contains an internal pullup resistor that is always activated, even when the $\overline{\text{IRQ}}$ pin is pulled low. See **Section 13. External Interrupt (IRQ)**.

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

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

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

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

See **Section 21. Analog-to-Digital Converter (ADC) Module**.

1.6.6 Port A I/O Pins (PTA6/$\overline{SS}$, PTA5/SPSCK, PTA4/$\overline{KBD4}$–PTA0/$\overline{KBD0}$)

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

See **Section 22. Input/Output (I/O) Ports** and **Section 13. External Interrupt (IRQ)**.

1.6.7 Port B I/O Pins (PTB7/AD7/TBCH1, PTB6/AD6/TBCH0, PTB5/AD5–PTB0/AD0)

PTB7/AD7/TBCH1, PTB6/AD6/TBCH0, and PTB5/AD5–PTB0/AD0 are special-function, bidirectional I/O port pins that can also be used for ADC inputs. PTB7/AD7/TBCH1 and PTB6/AD6/TBCH0 are special function bidirectional I/O port pins that can also be used for timer interface pins.

See **Section 16. Timer Interface A (TIMA) Module** and **Section 21. Analog-to-Digital Converter (ADC) Module**.

1.6.8 Port C I/O Pins (PTC4/OSC1, PTC3/OSC2, PTC2/MCLK, PTC1/MOSI, PTC0/MISO)

PTC4/OSC1–PTC0/MISO are special-function, bidirectional I/O port pins. See **Section 22. Input/Output (I/O) Ports**. PTC3/OSC2 and PTC4/OSC1 are shared with the on-chip oscillator circuit through configuration options. See **Section 7. Internal Clock Generator (ICG) Module**.

When applications require:

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

PTC2/MCLK is software selectable to be MCLK, or bus clock out. PTC1/MOSI can be programmed to be the MOSI signal for the SPI. PTC0/MISO can be programmed to be the MISO signal for the SPI.

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

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

See **Section 16. Timer Interface A (TIMA) Module** and **Section 22. Input/Output (I/O) Ports**.

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

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

See **Section 14. Enhanced Serial Communications Interface (ESCI) Module** and **Section 22. Input/Output (I/O) Ports**.

NOTE: *Any unused inputs and I/O ports should be tied to an appropriate logic level (either V_{DD} or V_{SS}). Although the I/O ports of the MC68HC908EY16 do not require termination, termination is recommended to reduce the possibility of electro-static discharge damage.*

Section 2. Memory Map

2.1 Contents

2.2 Introduction

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

- 16 Kbytes of FLASH memory, 15, 872 bytes of user space
- 512 bytes of random-access memory (RAM)
- 36 bytes of user-defined vectors
- 310 bytes of monitor routines in read-only memory (ROM)
- 1024 bytes of integrated FLASH burn-in routines in ROM

2.3 Unimplemented Memory Locations

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

2.4 Reserved Memory Locations

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

2.5 Input/Output (I/O) Section

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

- $FE00; SIM break status register, SBSR
- $FE01; SIM reset status register, SRSR
- $FE03; SIM break flag control register, SBFCR
- $FE08; FLASH control register, FLCR
- $FE09; break address register high, BRKH
- $FE0A; break address register low, BRKL
- $FE0B; break status and control register, BRKSCR
- $FE0C; LVI status register, LVISR
- $FF7E; FLASH block protect register, FLBPR

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

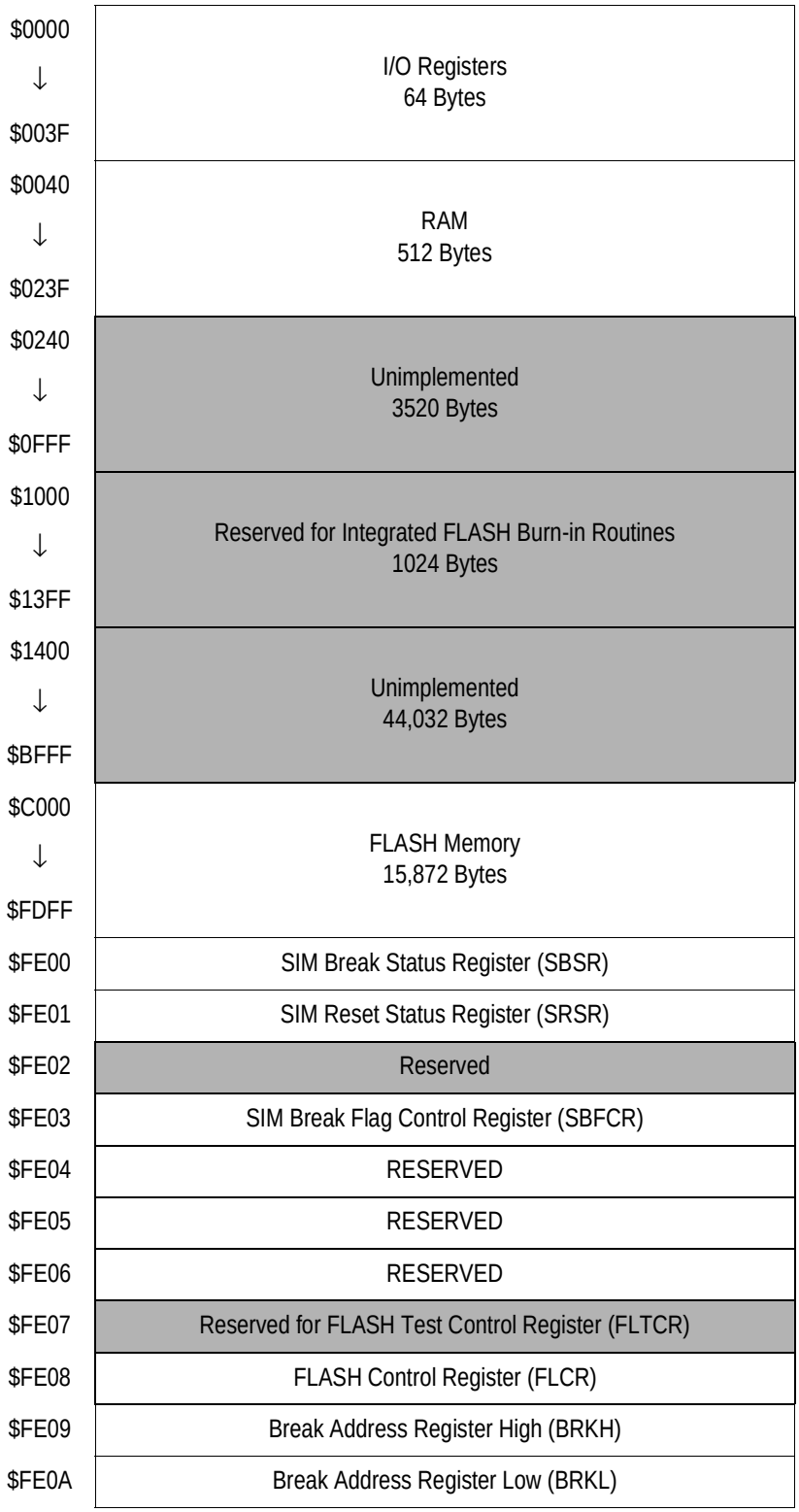
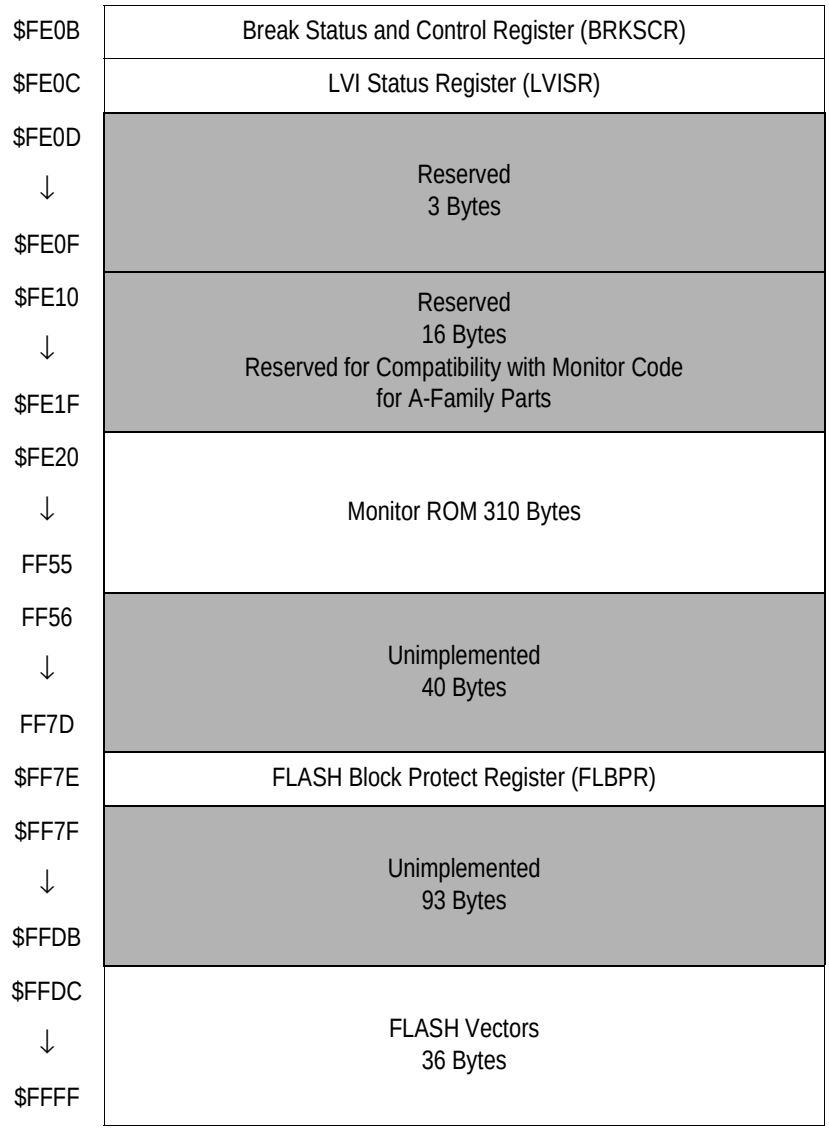


Figure 2-1. Memory Map



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

Figure 2-1. Memory Map (Continued)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA) See 363.	Read:	0	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB) See 366.	Read:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Write:								
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC) See 369.	Read:	0	0	0	PTC4	PTC3	PTC2	PTC1	PTC0
		Write:								
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD) See 372.	Read:	0	0	0	0	0	0	PTD1	PTD0
		Write:								
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA) See 364.	Read:	0	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB) See 367.	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC) See 370.	Read:	MCLKEN	0	0	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD) See 373.	Read:	0	0	0	0	0	0	DDRD1	DDRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE) See 375.	Read:	0	0	0	0	0	0	PTE1	PTE0
		Write:								
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0009	Reserved		R	R	R	R	R	R	R	R
$000A	Data Direction Register E (DDRE) See 376.	Read:	0	0	0	0	0	0	DDRE1	DDRE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000B	BEMF Register (BEMF) See 326.	Read:	BEMF7	BEMF6	BEMF5	BEMF4	BEMF3	BEMF2	BEMF1	BEMF0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000C	Reserved		R	R	R	R	R	R	R	R
$000D	SPI Control Register (SPCR) See 269.	Read:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Write:								
		Reset:	0	0	1	0	1	0	0	0
$000E	SPI Status and Control Register (SPSCR) See 272.	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	1	0	0	0
$000F	SPI Data Register (SPDR) See 275.	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Indeterminate after reset							
$0010	ESCI Control Register 1 (SCC1) See 220.	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0011	ESCI Control Register 2 (SCC2) See 223.	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0012	ESCI Control Register 3 (SCC3) See 225.	Read:	R8	T8	R	R	ORIE	NEIE	FEIE	PEIE
		Write:								
		Reset:	U	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0013	ESCI Status Register 1 (SCS1) See 227.	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$0014	ESCI Status Register 2 (SCS2) See 231.	Read:	0	0	0	0	0	0	BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0015	ESCI Data Register (SCDR) See 232.	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0016	ESCI Baud Rate Register (SCBR) See 232.	Read:	R	LINR	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0017	ESCI Prescale Register (SCPSC) See 234.	Read:	PDS2	PDS1	PDS0	PSSB4	PSSB3	PSSB2	PSSB1	PSSB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	ESCII Arbiter Control Register (SCIACTL) See 239.	Read:	AM1	ALOST	AM0	ACLK	AFIN	ARUN	AROVFL	ARD8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0019	ESCI Arbiter Data Register (SCIACTL) See 241.	Read:	ARD7	ARD6	ARD5	ARD4	ARD3	ARD2	ARD1	ARD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001A	Keyboard Status and Control Register (INTKBSCR) See 333.	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
		Write:						ACKK		
		Reset:	0	0	0	0	0	0	0	0
$001B	Keyboard Interrupt Enable Register (INTKBIER) See 334.	Read:	0	0	0	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		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 9)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001C	Timebase Control Register (TBCR) See 341.	Read:	TBIF	TBR2	TBR1	TBR0	0	TBIE	TBON	R
		Write:					TACK			
		Reset:	0	0	0	0	0	0	0	0
$001D	IRQ Status and Control Register (INTSCR) See 194.	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0
$001E	Configuration Register 2 (CONFIG2) See 148.	Read:	R	ESCI BDSRC	EXT-XTALEN	EXT-SLOW	EXT-CLKEN	TMB-CLKSEL	OSCENIN-STOP	SSB-PUENB
		Write:								
		Reset:	0	0	0	0	0	0	0	1
$001F	Configuration Register 1 (CONFIG1) See 148.	Read:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVI5OR3(1)	SSREC	STOP	COPD
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0020	Timer A Status and Control Register (TASC) See 291.	Read:	TOF	TOIE	TSTOP	0	R	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0021	Timer A Counter Register High (TACNTH) See 293.	Read:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0022	Timer A Counter Register Low (TACNTL) See 293.	Read:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0023	Timer A Counter Modulo Register High (TAMODH) See 294.	Read:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1

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

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0024	Timer A Counter Modulo Register Low (TAMODL) See 294.	Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Reset:	1	1	1	1	1	1	1	1
$0025	Timer A Channel 0 Status and Control Register (TASC0) See 295.	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0026	Timer A Channel 0 Register High (TACH0H) See 300.	Read: Write:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Reset:	Indeterminate after reset							
$0027	Timer A Channel 0 Register Low (TACH0L) See 300.	Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Reset:	Indeterminate after reset							
$0028	Timer A Channel 1 Status and Control Register (TASC1) See 300.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$0029	Timer A Channel 1 Register High (TACH1H) See 300.	Read: Write:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Reset:	Indeterminate after reset							
$002A	Timer A Channel 1 Register Low (TACH1L) See 300.	Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Reset:	Indeterminate after reset							
$002B	Timer B Status and Control Register (TBSC) See 316.	Read:	TOF	TOIE	TSTOP	0	R	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$002C	Timer B Counter Register High (TBCNTH) See 318.	Read:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002D	Timer B Counter Register Low (TBCNTL) See 318.	Read:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002E	Timer B Counter Modulo Register High (TBMODH) See 319.	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
$002F	Timer B Counter Modulo Register Low (TBMODL) See 319.	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
$0030	Timer B Channel 0 Status and Control Register (TBSC0) See 320.	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0031	Timer B Channel 0 Register High (TBCH0H) See 324.	Read:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Write:								
		Reset:	Indeterminate after reset							
$0032	Timer B Channel 0 Register Low (TBCH0L) See 324.	Read:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Write:								
		Reset:	Indeterminate after reset							
$0033	Timer B Channel 1 Status and Control Register (TBSC1) See 320.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$0034	Timer B Channel 1 Register High (TBCH1H) See 324.	Read:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Write:								
		Reset:	Indeterminate after reset							
$0035	Timer B Channel 1 Register Low (TBCH1L) See 324.	Read:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Write:								
		Reset:	Indeterminate after reset							

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0036	ICG Control Register (ICGCR) See 142.	Read:	CMIE	CMF	CMON	CS	ICGON	ICGS	ECGON	ECGS
		Write:		0						
		Reset:	0	0	0	0	1	0	0	0
$0037	ICG Multiplier Register (ICGMR) See 144.	Read:		N6	N5	N4	N3	N2	N1	N0
		Write:								
		Reset:	0	0	0	1	0	1	0	1
$0038	ICG Trim Register (ICGTR) See 145.	Read:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
		Write:								
		Reset:	1	0	0	0	0	0	0	0
$0039	ICG Divider Control Register (ICGDVR) See 145.	Read:					DDIV3	DDIV2	DDIV1	DDIV0
		Write:								
		Reset:	0	0	0	0	U	U	U	U
$003A	ICG DCO Stage Control Register (ICGDSR) See 146.	Read:	DSTG7	DSTG6	DSTG5	DSTG4	DDSTG3	DSTG2	DSTG1	DSTG0
		Write:	R	R	R	R	R	R	R	R
		Reset:	U	U	U	U	U	U	U	U
$003B	Reserved		R	R	R	R	R	R	R	R
$003C	Analog-to-Digital Status and Control Register (ADSCR) See 352.	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:								
		Reset:	0	0	0	1	1	1	1	1
$003D	Analog-to-Digital Data Register High (ADRH) See 355.	Read:	0	0	0	0	0	0	ADCH9	ADCH8
		Write:								
		Reset:	Unaffected by reset							
$003E	Analog-to-Digital Data Register Low (ADRL) See 359.	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:								
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$003F	Analog-to-Digital Clock Register (ADCLK) See 359.	Read:	ADIV2	ADIV1	ADIV0	ADICLK	MODE1	MODE0	R	0
		Write:								
		Reset:	0	0	0	0	0	1	0	0
$FE00	SIM Break Status Register (SBSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							NOTE	
		Reset:	0	0	0	0	0	0	0	0

Note: Writing a logic 0 clears SBSW.

$FE01	SIM Reset Status Register (SRSR) See 107.	Read:	POR	PIN	COP	ILOP	ILAD	MENRST	LVI	0
		Write:								
		POR:	1	0	0	0	0	0	0	0
$FE02	Reserved									
$FE03	SIM Break Flag Control Register (SBFCR)		BCFE	R	R	R	R	R	R	R
$FE04	Reserved									
↓										
$FE07	Reserved									
$FE08	FLASH Control Register (FLCR) See 60.	Read:	0	0	0	0	HVEN	MASS	ERASE	PGM
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE09	Break Address Register High (BRKH) See 161.	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 Register Low (BRKL) See 161.	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 R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE0B	Break Status and Control Register (BRKSCR) See 160.	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 186.	Read:	LVIOUT	0	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FF7E	FLASH Block Protect Register (FLBPR)(1) See 67.	Read:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
		Write:								
		Reset:	Unaffected by reset							
$FFFF	COP Control Register (COPCTL) See 181.	Read:	Low byte of reset vector							
		Write:	Writing clears COP counter (any value)							
		Reset:	Unaffected by reset							

1. Non-volatile FLASH register

= Unimplemented R = Reserved U = Unaffected

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

Table 2-1. Vector Addresses

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

Section 3. Random Access Memory (RAM)

3.1 Contents

3.2 Introduction

This section describes the 512 bytes of random-access memory (RAM).

3.3 Functional Description

Addresses $0040–$00FF and $0100–$023F are RAM locations. The location of the stack RAM is programmable with the reset stack pointer instruction (RSP). The 16-bit stack pointer allows the stack RAM to be anywhere in the 64K-byte memory space.

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

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

Before processing an interrupt, the CPU uses five bytes of the stack to save the contents of the central processor unit (CPU) registers.

NOTE: *For M6805, M146805, and M68HC05 compatibility, the H register is not stacked.*

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

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

Section 4. FLASH Memory

4.1 Contents

4.2 Introduction

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

4.3 Functional Description

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

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

The program and erase operations are facilitated through control bits in the FLASH control register (FLCR). See **4.4 FLASH Control Register**.

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

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

4.4 FLASH Control Register

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

Address: $FE08

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

= Unimplemented

Figure 4-1. FLASH Control Register (FLCR)

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

HVEN — High-Voltage Enable Bit

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

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

MASS — Mass Erase Control Bit

Setting this read/write bit configures the 16-Kbyte FLASH array for mass or page erase operation.

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

ERASE — Erase Control Bit

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

1 = Erase operation selected
0 = Erase operation unselected

PGM — Program Control Bit

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

1 = Program operation selected
0 = Program operation unselected

4.5 FLASH Page Erase Operation

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

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

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

NOTE: *Due to the security feature (**Section 10. Monitor ROM (MON)**) the last page of the FLASH (0xFFDC–0xFFFF), which contains the security bytes, cannot be erased by Page Erase Operation. It can only be erased with the Mass Erase Operation.*

4.6 FLASH Mass Erase Operation

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

1. Set both the ERASE bit and the MASS bit in the FLASH control register.
2. Read from the FLASH block protect register.
3. Write any data to any FLASH address[1] within the FLASH memory address range.
4. Wait for a time, t_{NVS} (minimum of 10 μs).
5. Set the HVEN bit.
6. Wait for a time, t_{MErase} (minimum of 4 ms).
7. Clear the ERASE bit.
8. Wait for a time, t_{NVHL} (minimum of 100 μs).
9. Clear the HVEN bit.
10. After a time, t_{RCV} (minimum of 1 μs), the memory can be accessed again in read mode.

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.*

1. When in monitor mode, with security sequence failed (see **Section 10. Monitor ROM (MON)**), write to the FLASH block protect register instead of any FLASH address.

4.7 FLASH Program/Read Operation

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

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

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

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

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.*

4.8 FLASH Block Protection

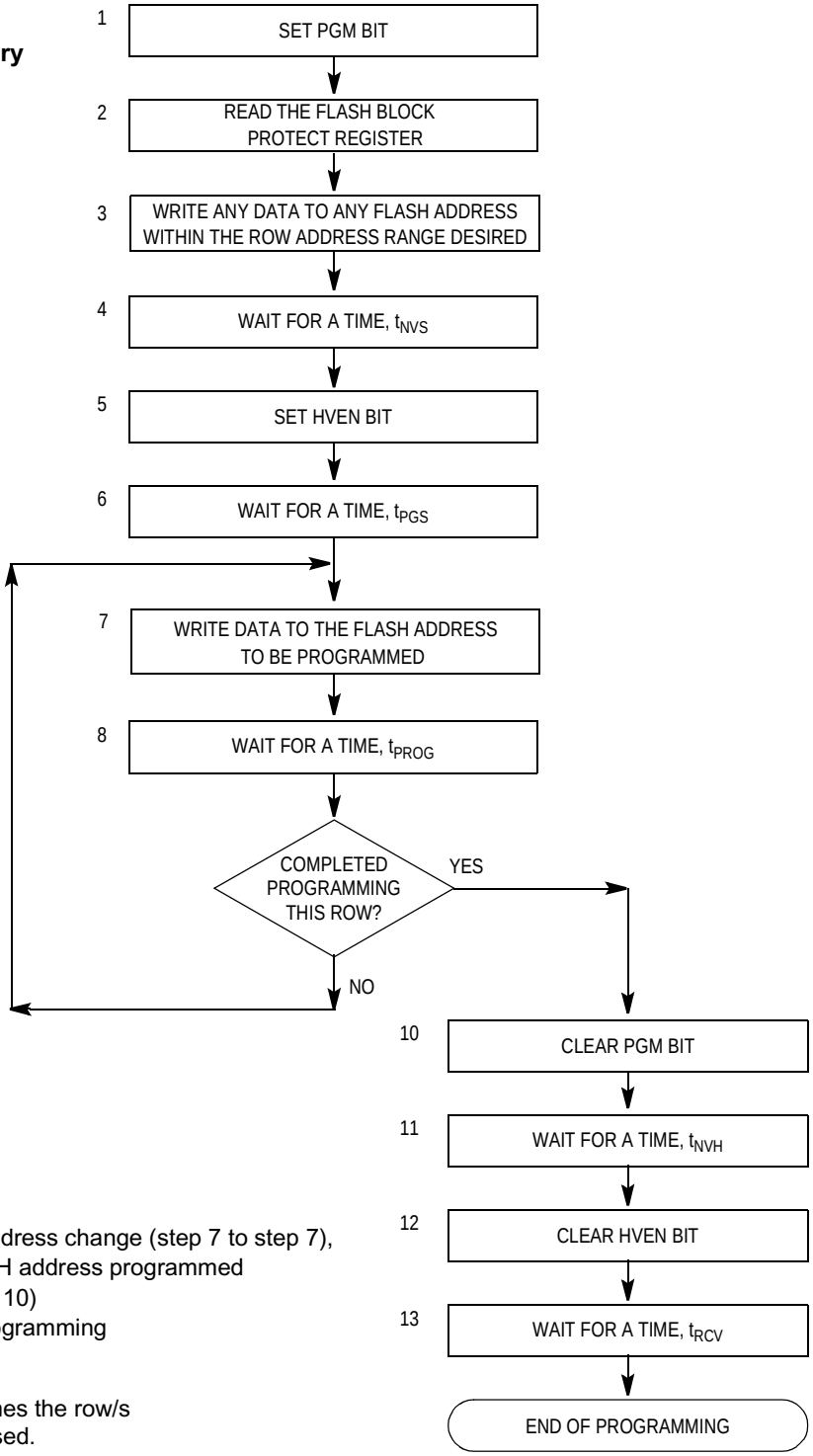
Due to the ability of the on-board charge pump to erase and program the FLASH memory in the target application, provision is made for protecting a block of memory from unintentional erase or program operations due to system malfunction. This protection is done by using the FLASH block protect register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

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

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

When bits within the FLBPR are programmed, they lock a block of memory address ranges as shown in **4.9 FLASH Block Protect Register**. If 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. FLBPR itself can then be erased or programmed only with an external voltage V_{tst} present on the IRQ pin.

Algorithm for programming a row (32 bytes) of FLASH memory



Notes:

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

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

Figure 4-2. FLASH Programming Flowchart

4.9 FLASH Block Protect Register

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

Address: $FF7E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
Reset:	U	U	U	U	U	U	U	U

U = Unaffected by reset. Initial value from factory is $FF.
Write to this register is by a programming sequence to the FLASH memory.

Figure 4-3. FLASH Block Protect Register (FLBPR)

BPR7–BPR0 — FLASH Block Protect Bits

These eight bits represent bits 13–6 of a 16-bit memory address. Bits 15 and 14 are logic 1s and bits 5–0 are logic 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, etc., (64 bytes page boundaries) within the FLASH memory. See Figure 4-4 and Table 4-1.

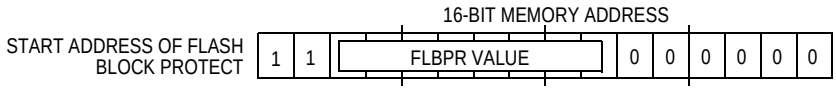


Figure 4-4. FLASH Block Protect Start Address

Table 4-1. Protect Start Address Examples

BPR7–BPR0	Start of Address of Protect Range(1)
$00	The entire FLASH memory is protected.
$01 (**0000 0001**)	$C040 (110**0 0000 01**00 0000)
$02 (**0000 0010**)	$C080 (110**0 0000 10**00 0000)
and so on...	
$FE (**1111 1110**)	$FF80 (111**1 1111 10**00 0000)
$FF	The entire FLASH memory is not protected.

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

4.10 Wait Mode

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

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

4.11 Stop Mode

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

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

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

Section 5. Central Processor Unit (CPU)

5.1 Contents

5.2 Introduction

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

5.3 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

5.4 CPU Registers

Figure 5-1 shows the five CPU registers. CPU registers are not part of the memory map.

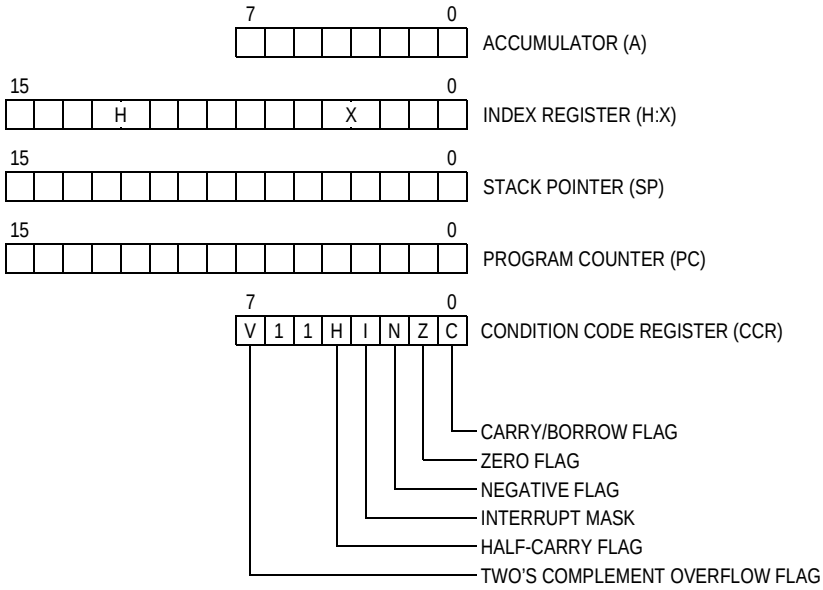


Figure 5-1. CPU Registers

5.4.1 Accumulator

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

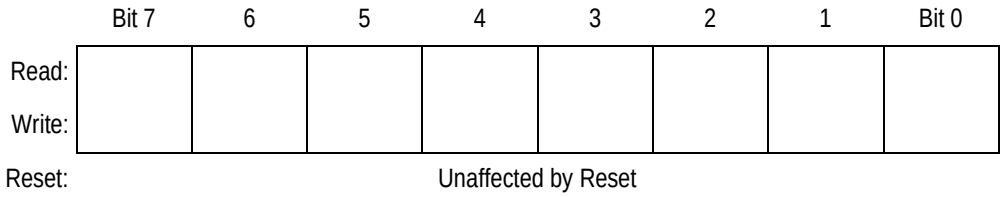


Figure 5-2. Accumulator (A)

5.4.2 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 5-3. Index Register (H:X)

5.4.3 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 (LSB) to $FF and does not affect the most significant byte (MSB). The stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

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

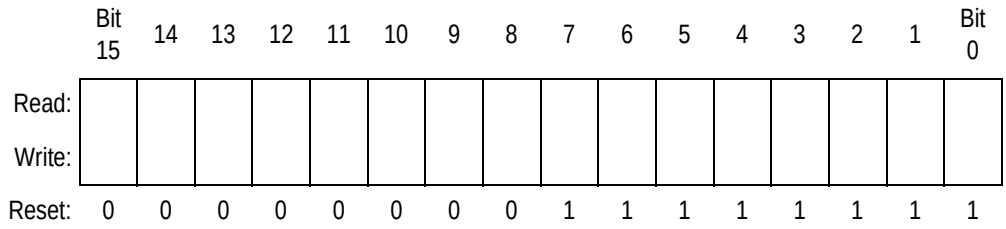


Figure 5-4. Stack Pointer (SP)

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.*

5.4.4 Program Counter

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

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

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

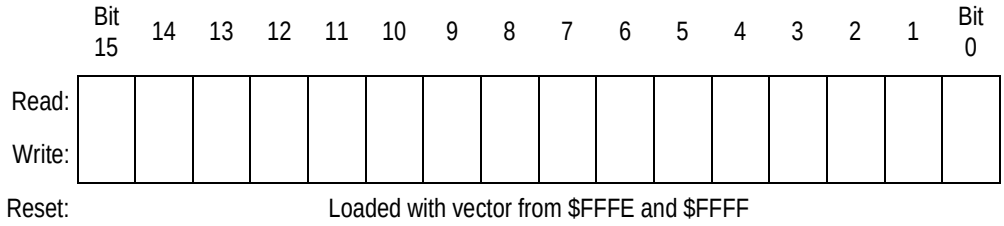


Figure 5-5. Program Counter (PC)

5.4.5 Condition Code Register

The 8-bit condition code register (CCR) contains the interrupt mask and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to logic 1. The functions of the CCR are described here.

	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 5-6. 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 decimal adjust A (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 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 push H onto stack (PSHH) and pull H from stack (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 only be cleared 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 produce 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

5.5 Arithmetic/Logic Unit (ALU)

The arithmetic/logic unit (ALU) performs the arithmetic and logic operations defined by the instruction set.

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

5.6 Low-Power Modes

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

5.6.1 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.

5.6.2 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.

5.7 CPU During Break Interrupts

If the break module is enabled, a break interrupt causes the CPU to execute the software interrupt instruction (SWI) at the completion of the current CPU instruction (see **Section 9. Break Module (BRK)**). The program counter vectors to $FFFC–$FFFD ($FEFC–$FEFD in monitor mode).

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.

5.8 Instruction Set Summary

Table 5-1 provides a summary of the M68HC08 instruction set.

Table 5-1. Instruction Set Summary (Sheet 1 of 8)

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

Table 5-1. Instruction Set Summary (Sheet 2 of 8)

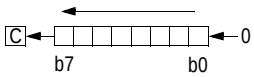
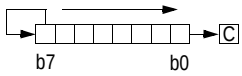
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ASL *opr* ASLA ASLX ASL *opr*,X ASL ,X ASL *opr*,SP	Arithmetic Shift Left (Same as LSL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
ASR *opr* ASRA ASRX ASR *opr*,X 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
BHS *rel*	Branch if Higher or Same (Same as BCC)	PC ← (PC) + 2 + *rel* ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if $\overline{\text{IRQ}}$ Pin High	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if $\overline{\text{IRQ}}$ Pin Low	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #*opr* BIT *opr* BIT *opr* BIT *opr*,X BIT *opr*,X BIT ,X BIT *opr*,SP BIT *opr*,SP	Bit Test	(A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A5 B5 C5 D5 E5 F5 9EE5 9ED5	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
BLE *opr*	Branch if Less Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) | (N ⊕ V) = 1	–	–	–	–	–	–	REL	93	rr	3
BLO *rel*	Branch if Lower (Same as BCS)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3

Table 5-1. Instruction Set Summary (Sheet 3 of 8)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BLS *rel*	Branch if Lower or Same	PC ← (PC) + 2 + *rel* ? (C) | (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT *opr*	Branch if Less Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) =1	–	–	–	–	–	–	REL	91	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	PC ← (PC) + 2 + *rel* ? (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI *rel*	Branch if Minus	PC ← (PC) + 2 + *rel* ? (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	PC ← (PC) + 2 + *rel* ? (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + 2 + *rel* ? (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL *rel*	Branch if Plus	PC ← (PC) + 2 + *rel* ? (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA *rel*	Branch Always	PC ¨ (PC) + 2 + *rel*	–	–	–	–	–	–	REL	20	rr	3
BRCLR *n*,*opr*,*rel*	Branch if Bit *n* in M Clear	PC ← (PC) + 3 + *rel* ? (Mn) = 0	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	PC ← (PC) + 2	–	–	–	–	–	–	REL	21	rr	3
BRSET *n*,*opr*,*rel*	Branch if Bit *n* in M Set	PC ← (PC) + 3 + *rel* ? (Mn) = 1	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n*,*opr*	Set Bit *n* in M	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	4
CBEQ *opr*,*rel* CBEQA #*opr*,*rel* CBEQX #*opr*,*rel* CBEQ *opr*,X+,*rel* CBEQ X+,*rel* CBEQ *opr*,SP,*rel*	Compare and Branch if Equal	PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (X) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 2 + rel ? (A) – (M) = $00 PC ← (PC) + 4 + rel ? (A) – (M) = $00	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 4 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	I ← 0	–	–	0	–	–	–	INH	9A		2

Table 5-1. Instruction Set Summary (Sheet 4 of 8)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
CLR *opr* CLRA CLRX CLRH CLR *opr*,X CLR ,X CLR *opr*,SP	Clear	M ← $00 A ← $00 X ← $00 H ← $00 M ← $00 M ← $00 M ← $00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	3 1 1 1 3 2 4
CMP #*opr* CMP *opr* CMP *opr* CMP *opr*,X CMP *opr*,X CMP ,X CMP *opr*,SP CMP *opr*,SP	Compare A with M	(A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A1 B1 C1 D1 E1 F1 9EE1 9ED1	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
COM *opr* COMA COMX COM *opr*,X COM ,X COM *opr*,SP	Complement (One's Complement)	M ← ($\overline{M}$) = $FF – (M) A ← ($\overline{A}$) = $FF – (M) X ← ($\overline{X}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M)	0	–	–	↕	↕	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff	4 1 1 4 3 5
CPHX #*opr* CPHX *opr*	Compare H:X with M	(H:X) – (M:M + 1)	↕	–	–	↕	↕	↕	IMM DIR	65 75	ii ii+1 dd	3 4
CPX #*opr* CPX *opr* CPX *opr* CPX ,X CPX *opr*,X CPX *opr*,X CPX *opr*,SP CPX *opr*,SP	Compare X with M	(X) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A3 B3 C3 D3 E3 F3 9EE3 9ED3	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
DAA	Decimal Adjust A	$(A)_{10}$	U	–	–	↕	↕	↕	INH	72		2
DBNZ *opr,rel* DBNZA *rel* DBNZX *rel* DBNZ *opr*,X,*rel* DBNZ X,*rel* DBNZ *opr*,SP,*rel*	Decrement and Branch if Not Zero	A ← (A) – 1 or M ← (M) – 1 or X ←(X) – 1 PC ← (PC) + 3 + *rel* ? (result) ¼ 0 PC ← (PC) + 2 + *rel* ? (result) ¼ 0 PC ← (PC) + 2 + *rel* ? (result) ¼ 0 PC ← (PC) + 3 + *rel* ? (result) ¼ 0 PC ← (PC) + 2 + *rel* ? (result) ¼ 0 PC ← (PC) + 4 + *rel* ? (result) ¼ 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

Table 5-1. Instruction Set Summary (Sheet 5 of 8)

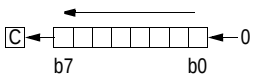
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X EOR *opr*,SP EOR *opr*,SP	Exclusive OR M with A	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9EE8 9ED8	ii 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
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

Table 5-1. Instruction Set Summary (Sheet 6 of 8)

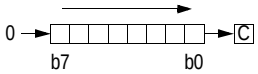
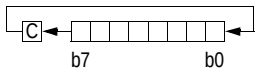
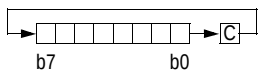
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LSR *opr* LSRA LSR*X* LSR *opr*,X LSR ,X LSR *opr*,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	4 1 1 4 3 5
MOV *opr,opr* MOV *opr*,X+ MOV #*opr,opr* MOV X+,*opr*	Move	$(M)_{Destination} \leftarrow (M)_{Source}$ H:X ← (H:X) + 1 (IX+D, DIX+)	0	–	–	↕	↕	–	DD DIX+ IMD IX+D	4E 5E 6E 7E	dd dd dd ii dd dd	5 4 4 4
MUL	Unsigned multiply	X:A ¨ (X) × (A)	–	0	–	–	–	0	INH	42		5
NEG *opr* NEGA NEGX NEG *opr*,X NEG ,X NEG *opr*,SP	Negate (Two's Complement)	M ← –(M) = $00 – (M) A ← –(A) = $00 – (A) X ← –(X) = $00 – (X) M ← –(M) = $00 – (M) M ← –(M) = $00 – (M)	↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	4 1 1 4 3 5
NOP	No Operation	None	–	–	–	–	–	–	INH	9D		1
NSA	Nibble Swap A	A ← (A[3:0]:A[7:4])	–	–	–	–	–	–	INH	62		3
ORA #*opr* ORA *opr* ORA *opr* ORA *opr*,X ORA *opr*,X ORA ,X ORA *opr*,SP ORA *opr*,SP	Inclusive OR A and M	A ← (A) | (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AA BA CA DA EA FA 9EEA 9EDA	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
PSHA	Push A onto Stack	Push (A); SP ← (SP) – 1	–	–	–	–	–	–	INH	87		2
PSHH	Push H onto Stack	Push (H); SP ← (SP) – 1	–	–	–	–	–	–	INH	8B		2
PSHX	Push X onto Stack	Push (X); SP ← (SP) – 1	–	–	–	–	–	–	INH	89		2
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

Table 5-1. Instruction Set Summary (Sheet 7 of 8)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
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 $\overline{\text{IRQ}}$ Pin; Stop Oscillator	I ← 0; Stop Oscillator	–	–	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
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

Table 5-1. Instruction Set Summary (Sheet 8 of 8)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
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; Stop Processor	I bit ← 0	–	–	0	–	–	–	INH	8F		1

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

5.9 Opcode Map

The opcode map is provided in Table 5-2.

Table 5-2. Opcode Map

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

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

*Pre-byte for stack pointer indexed instructions

MSB / LSB	
0	High Byte of Opcode in Hexadecimal
Low Byte of Opcode in Hexadecimal	Cycles Opcode Mnemonic Number of Bytes / Addressing Mode

Example: 5 / BRSET0 / 3 DIR

Section 6. System Integration Module (SIM)

6.1 Contents

6.2 Introduction

This section describes the system integration module (SIM), which supports up to 24 external and/or internal interrupts. The SIM is a system state controller that coordinates the central processor unit (CPU) and exception timing. Together with the CPU, the SIM controls all microcontroller unit (MCU) activities. A block diagram of the SIM is shown in Figure 6-1.

The SIM is responsible for:

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

Table 6-1 shows the internal signal names used in this section.

Table 6-1. Signal Name Conventions

Signal Name	Description
CGMXCLK	Selected clock source from internal clock generator module (ICG)
CGMOUT	Clock output from ICG module (bus clock = CGMOUT divided by two)

Table 6-1. Signal Name Conventions

Signal Name	Description
IAB	Internal address bus
IDB	Internal data bus
PORRST	Signal from the power-on reset (POR) module to the SIM
IRST	Internal reset signal
R/$\overline{W}$	Read/write signal

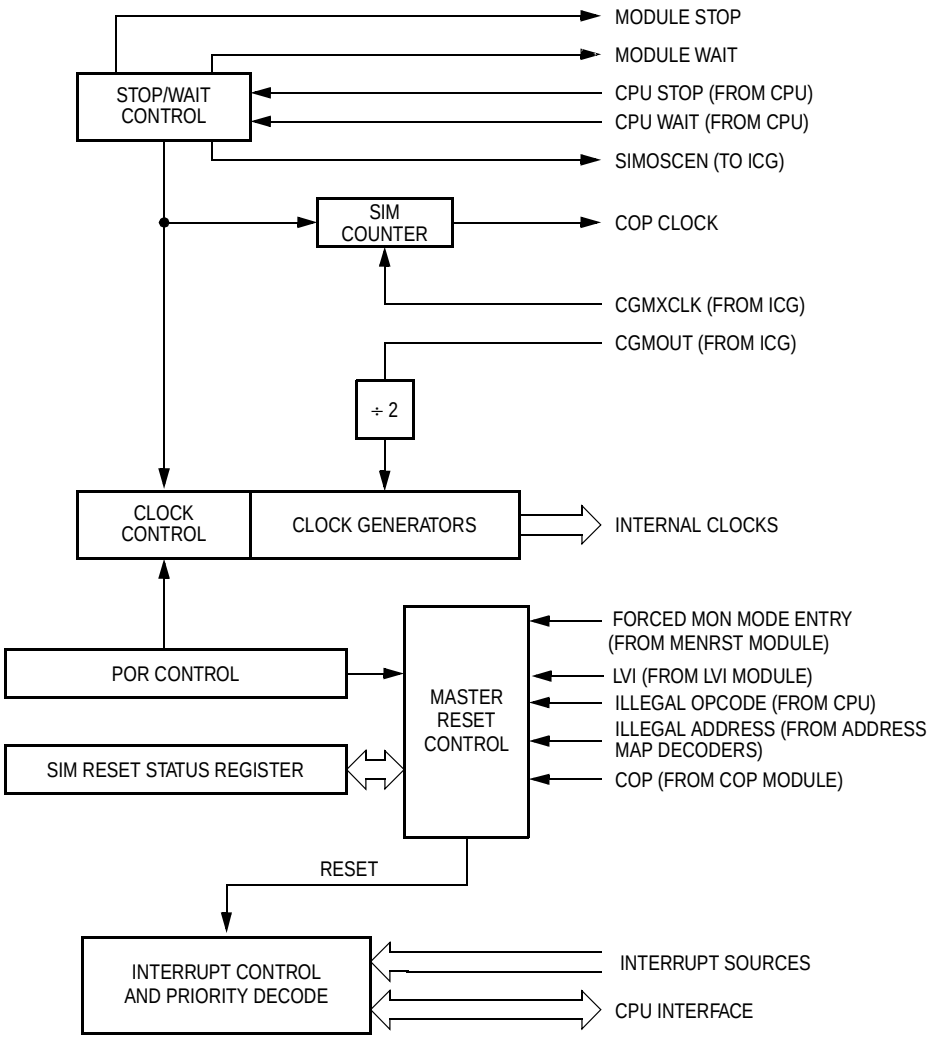


Figure 6-1. SIM Block Diagram

6.3 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, CGMOUT, as shown in Figure 6-2. This clock originates from either an external oscillator or from the internal clock generator.

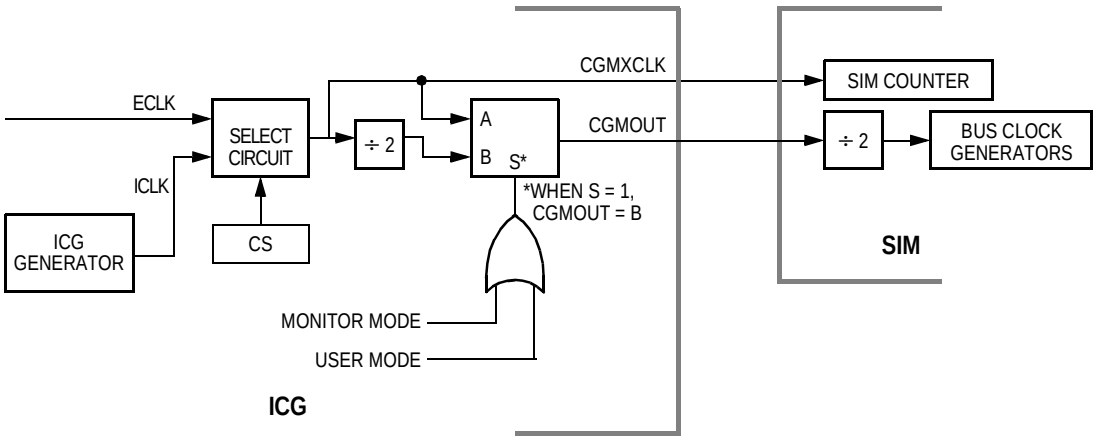


Figure 6-2. System Clock Signals

6.3.1 Bus Timing

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

6.3.2 Clock Startup from POR or LVI Reset

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

6.3.3 Clocks in Stop Mode and Wait Mode

Upon exit from stop mode by an interrupt or reset, the SIM allows CGMXCLK to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay timeout. Stop mode recovery timing is discussed in detail in **6.7.2 Stop Mode**.

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

6.4 Reset and System Initialization

The MCU has these internal reset sources:

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

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

These internal resets clear the SIM counter and set a corresponding bit in the SIM reset status register (SRSR). See **6.5 SIM Counter** and **6.8.2 SIM Reset Status Register**.

6.4.1 External Pin Reset

The $\overline{\text{RST}}$ pin circuit includes 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 a minimum of 67 CGMXCLK cycles, assuming that neither the POR nor the LVI was the source of the reset. See Table 6-2 for details. Figure 6-3 shows the relative timing.

Table 6-2. PIN Bit Set Timing

Reset Type	Number of Cycles Required to set PIN
POR/LVI	4163 (4096 + 64 + 3)
All Others	67 (64 + 3)

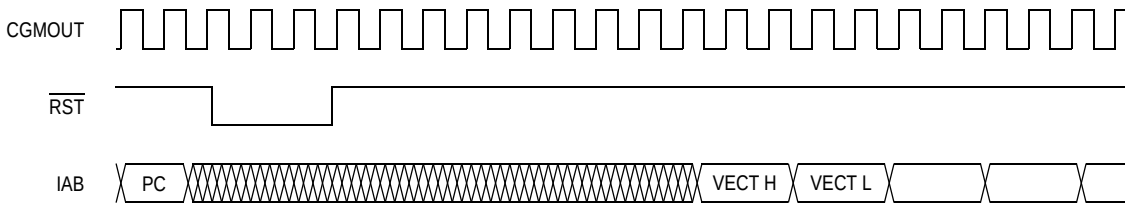


Figure 6-3. External Reset Timing

6.4.2 Active Resets from Internal Sources

An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, POR, or MENRST as shown in Figure 6-4.

NOTE: *For LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM asserts IRST. The internal reset signal then follows with the 64-cycle phase as shown in Figure 6-5.*

The COP reset is asynchronous to the bus clock.

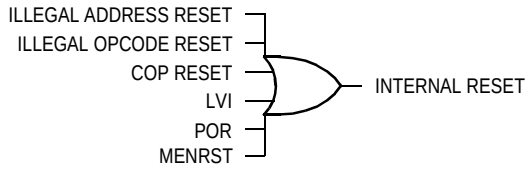


Figure 6-4. Sources of Internal Reset

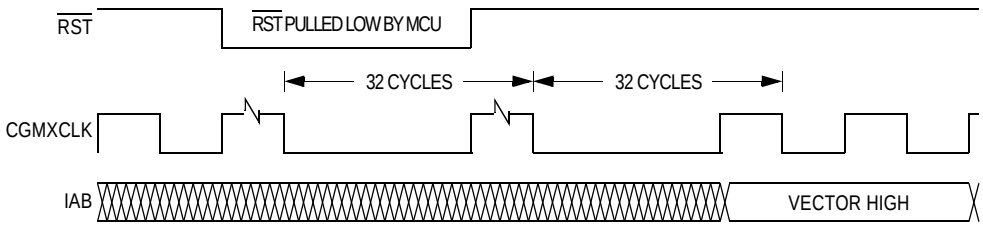


Figure 6-5. Internal Reset Timing

6.4.2.1 Power-On Reset

When power is first applied to the MCU, the power-on reset (POR) module generates a pulse to indicate that power-on has occurred. The MCU is held in reset while the SIM counter counts out 4096 CGMXCLK cycles. Another 64 CGMXCLK cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur.

At power-on, these events occur:

- A POR pulse is generated.
- The internal reset signal is asserted.
- The SIM enables CGMOUT.
- Internal clocks to the CPU and modules are held inactive for 4096 CGMXCLK cycles to allow stabilization of the internal clock generator.
- The POR bit of the SIM reset status register (SRSR) is set and all other bits in the register are cleared.

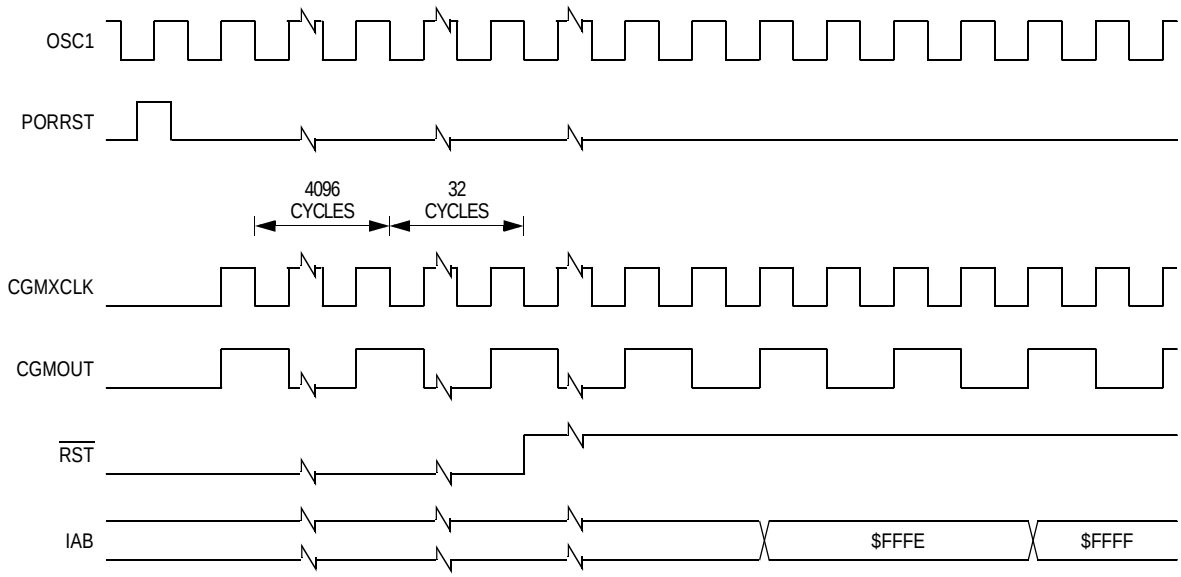


Figure 6-6. POR Recovery

6.4.2.2 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 reset status register (SRSR).

To prevent a COP module timeout, write any value to location $FFFF. Writing to location $FFFF clears the COP counter and stages 12–5 of the SIM counter. The SIM counter output, which occurs at least every 2^{12}–2^4 CGMXCLK cycles, drives the COP counter. The COP should be serviced as soon as possible out of reset to guarantee the maximum amount of time before the first timeout.

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

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

6.4.2.4 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.

6.4.2.5 Forced Monitor Mode Entry Reset (MENRST)

The MENRST module is monitoring the reset vector fetches and will assert an internal reset if it detects that the reset vectors are erased ($00). When the MCU comes out of reset, it is forced into monitor mode. See **Section 10. Monitor ROM (MON)**.

6.4.2.6 Low-Voltage Inhibit (LVI) Reset

The low-voltage inhibit module (LVI) asserts its output to the SIM when the V_{DD} voltage falls to the V_{TRIPF} voltage. The LVI bit in the SIM reset status register (SRSR) is set and a chip reset is asserted if the LVIPWRD and LVIRSTD bits in the CONFIG register are at logic 0. The MCU is held in reset until V_{DD} rises above $V_{TRIPR.}$ The MCU remains in reset until the SIM counts 4096 CGMXCLK to begin a reset recovery. Another 64 CGMXCLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. See **Section 12. Low-Voltage Inhibit (LVI) Module**.

6.5 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 overflow supplies the clock for the COP module. The SIM counter is 12 bits long and is clocked by the falling edge of CGMXCLK.

6.5.1 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 internal clock generator to drive the bus clock state machine.

6.5.2 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 or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the configuration register. If the SSREC bit is a logic 1, then the stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32 CGMXCLK cycles.

6.5.3 SIM Counter and Reset States

The SIM counter is free-running after all reset states. See **6.4.2 Active Resets from Internal Sources** for counter control and internal reset recovery sequences.

6.6 Program Exception Control

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

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

6.6.1 Interrupts

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the return-from-interrupt (RTI) instruction recovers the CPU register contents from the stack so that normal processing can resume. Figure 6-7 shows interrupt entry timing. Figure 6-8 shows interrupt recovery timing.

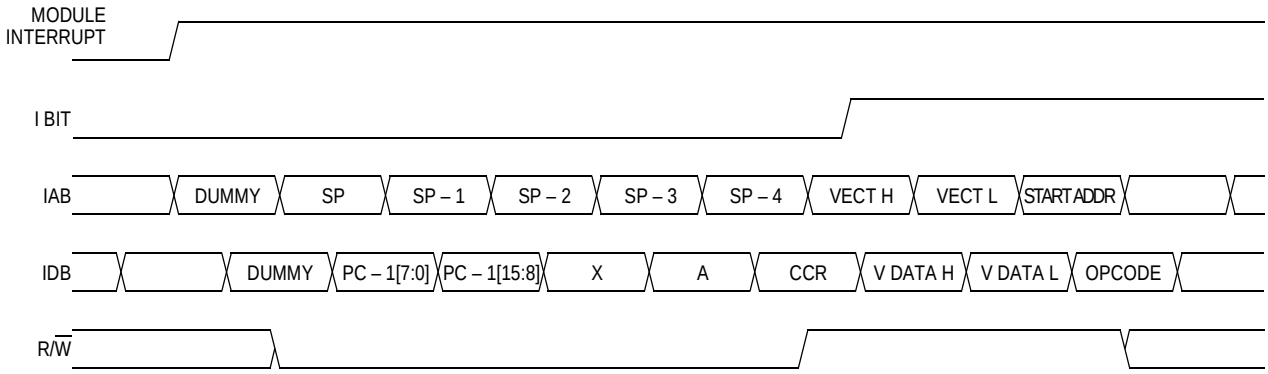


Figure 6-7. Interrupt Entry

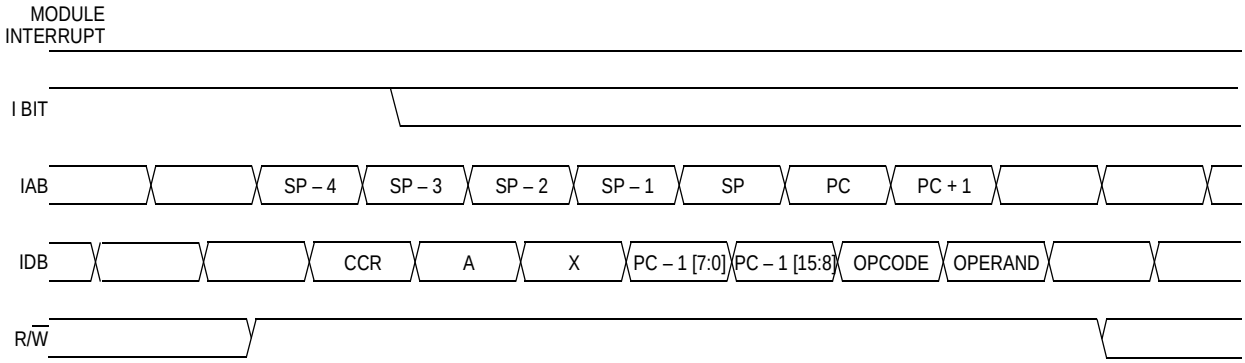


Figure 6-8. Interrupt Recovery

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. As shown in Figure 6-9, once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced or the I bit is cleared.

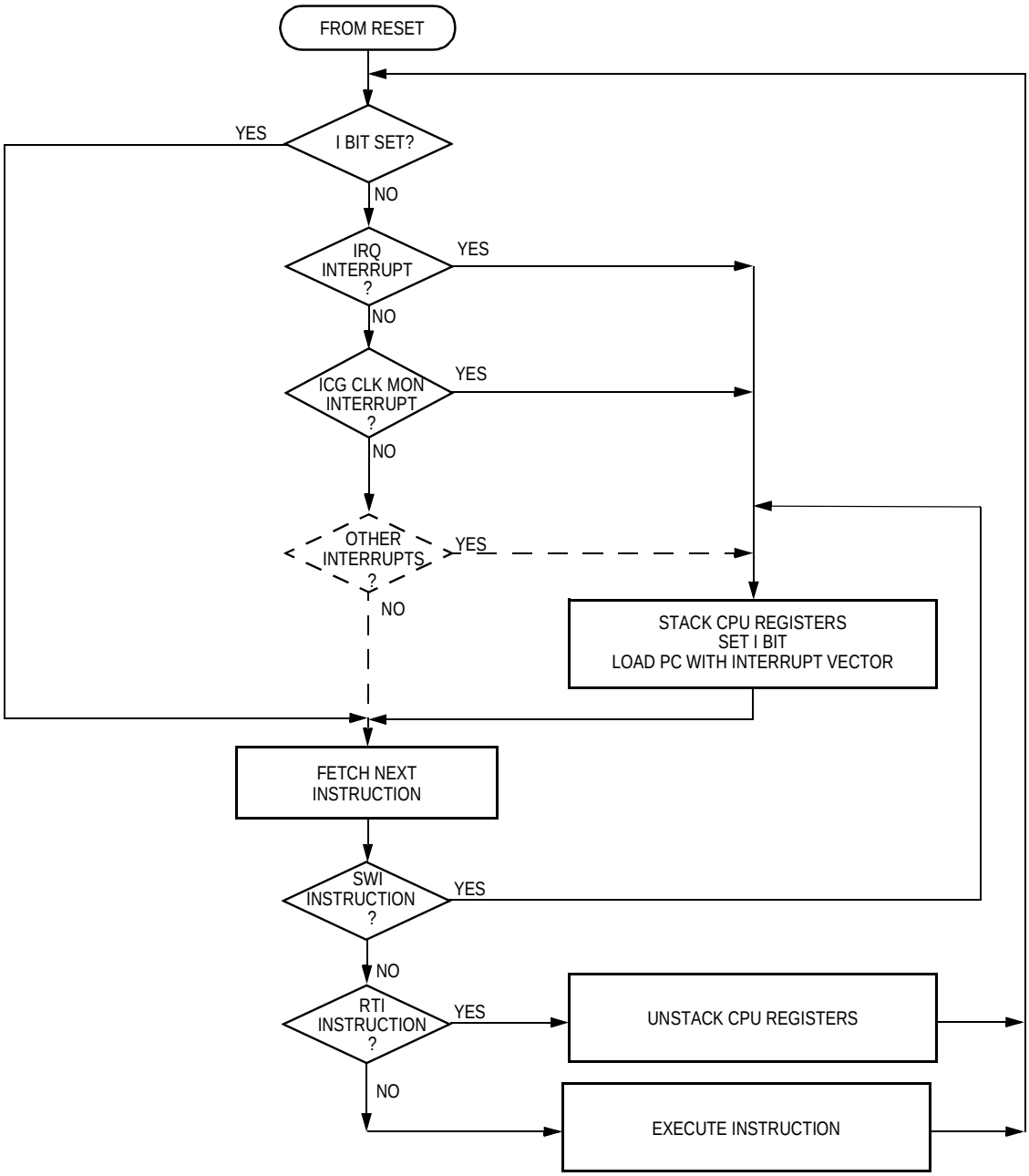


Figure 6-9. Interrupt Processing

6.6.1.1 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 6-10 demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the load-accumulator-from-memory (LDA) instruction is executed.

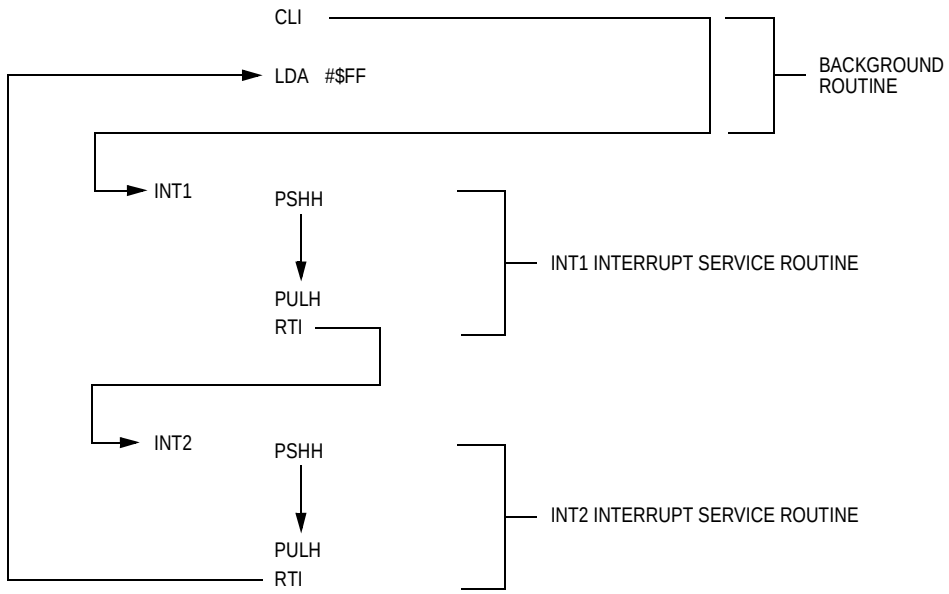


Figure 6-10. Interrupt Recognition Example

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

NOTE: *To maintain compatibility with the M6805, M146805, and MC68HC05 Families the H register is not pushed on the stack during interrupt entry. If the interrupt service routine modifies the H register or uses the indexed addressing mode, software should save the H register and then restore it prior to exiting the routine.*

6.6.1.2 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.*

6.6.2 Reset

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

6.6.3 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. See **Section 22. Break Module (BRK)**. The SIM puts the CPU into the break state by forcing it to the SWI vector location. Refer to the break interrupt subsection of each module to see how each module is affected by the break state.

6.6.4 Status Flag Protection in Break Mode

The SIM controls whether status flags contained in other modules can be cleared during break mode. The user can select whether flags are protected from being cleared by properly initializing the break clear flag enable bit (BCFE) in the SIM break flag control register (SBFCR).

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

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

6.7 Low-Power Modes

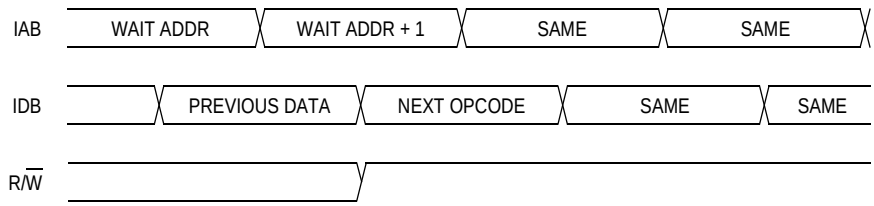
Executing the WAIT or STOP instruction puts the MCU in a low power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur. Low-power modes are exited via an interrupt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

6.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while one set of peripheral clocks continues to run. Figure 6-11 shows the timing for wait mode entry.

A module that is active during wait mode can wake up the CPU with an interrupt if the interrupt is enabled. Stacking for the interrupt begins one cycle after the WAIT instruction during which the interrupt occurred. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

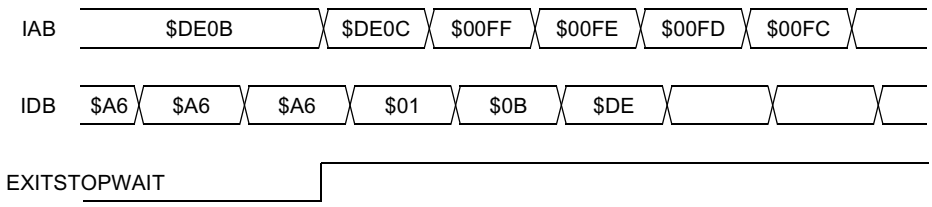
Wait mode can also be exited by a reset. If the COP disable bit, COPD, in the configuration register is logic 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.



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

Figure 6-11. Wait Mode Entry Timing

Figure 6-12 and Figure 6-13 show the timing for WAIT recovery.



Note: EXITSTOPWAIT = CPU interrupt

Figure 6-12. Wait Recovery from Interrupt

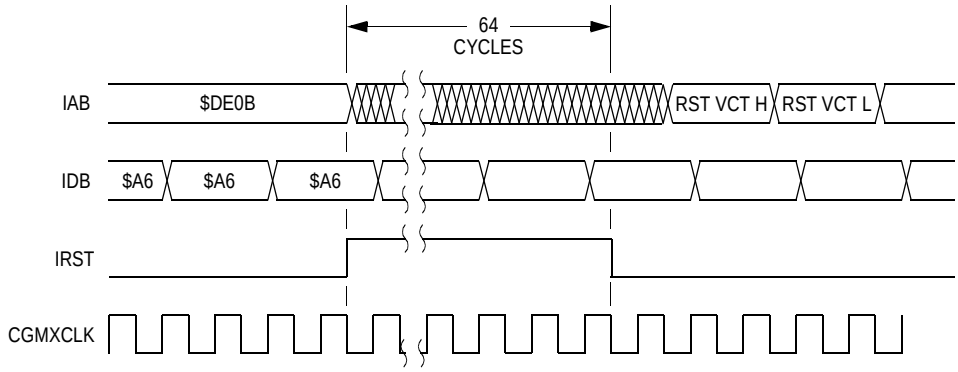


Figure 6-13. Wait Recovery from Internal Reset

6.7.2 Stop Mode

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

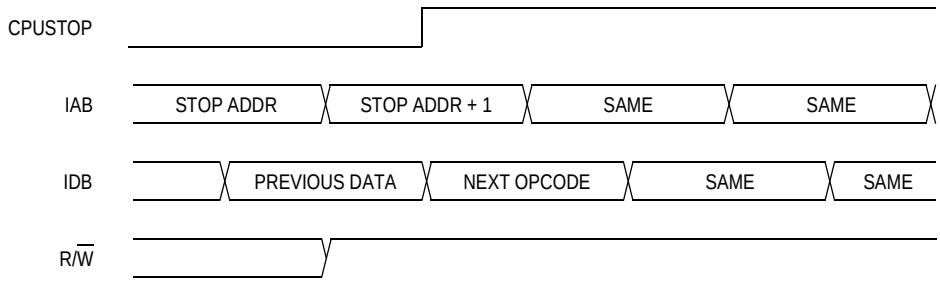
The CPU and peripheral clocks do not become active until after the stop delay timeout. Stop mode is exited via an interrupt request from a module that is still active in stop mode or from a system reset.

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

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

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

The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. Figure 6-14 shows stop mode entry timing.



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

Figure 6-14. Stop Mode Entry Timing

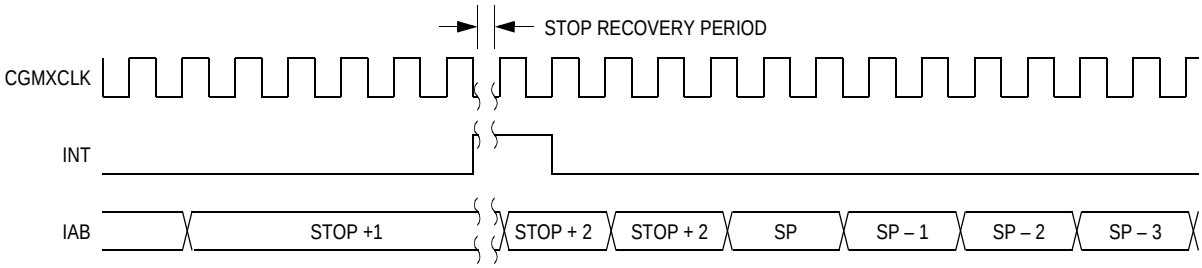


Figure 6-15. Stop Mode Recovery from Interrupt

6.8 SIM Registers

The SIM has three memory mapped registers. **Table 6-3** shows the mapping of these registers.

Table 6-3. SIM Registers

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

6.8.1 SIM Break Status Register

The SIM break status register (SBSR) contains a flag to indicate that a break caused an exit from stop or wait mode.

Address: $FE00

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

R = Reserved

Note: 1. Writing a logic 0 clears SBSW

Figure 6-16. SIM Break Status Register (SBSR)

SBSW — SIM Break STOP/WAIT

This status bit is useful in applications requiring a return to stop or wait mode after exiting from a break interrupt. SBSW can be cleared by writing a logic 0 to it. Reset clears SBSW.

1 = Stop or wait mode was exited by break interrupt
0 = Stop or wait mode was not exited by break interrupt

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.

6.8.2 SIM Reset Status Register

This register contains seven bits that show the source of the last reset. The status register will clear automatically after reading it. A power-on reset sets the POR bit and clears all other bits in the register.

Address: $FE01

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

= Unimplemented

Figure 6-17. SIM Reset Status Register (SRSR)

POR — Power-On Reset Bit
- 1 = Last reset caused by POR circuit
- 0 = Read of SRSR

PIN — External Reset Bit
- 1 = Last reset caused by external reset pin $\overline{\text{RST}}$
- 0 = POR or read of SPSR

COP — Computer Operating Properly Reset Bit
- 1 = Last reset caused by COP counter
- 0 = POR or read of SRSR

ILOP — Illegal Opcode Reset Bit
- 1 = Last reset caused by an illegal opcode
- 0 = POR or read of SRSR

ILAD — Illegal Address Reset Bit (opcode fetches only)
- 1 = Last reset caused by an opcode fetch from an illegal address
- 0 = POR or read of SRSR

MENRST — Forced Monitor Mode Entry Reset Bit
- 1 = Last reset was caused by the MENRST circuit
- 0 = POR or read of SRSR

LVI — Low-Voltage Inhibit Reset Bit

1 = Last reset was caused by the LVI circuit

0 = POR or read of SRSR

6.8.3 SIM Break Flag Control Register

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

Address: $FE03

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

R = Reserved

Figure 6-18. SIM Break Flag Control Register (SBFCR)

BCFE — Break Clear Flag Enable Bit

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

1 = Status bits clearable during break

0 = Status bits not clearable during break

Section 7. Internal Clock Generator (ICG) Module

7.1 Contents

7.2 Introduction

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

7.3 Features

The ICG has these features:

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

NOTE: *For the MC68HC908EY16, do not exceed the maximum bus frequency of 8 MHz at 5.0 V.*

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

7.4 Functional Description

The ICG, shown in Figure 7-1, contains these major submodules:

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

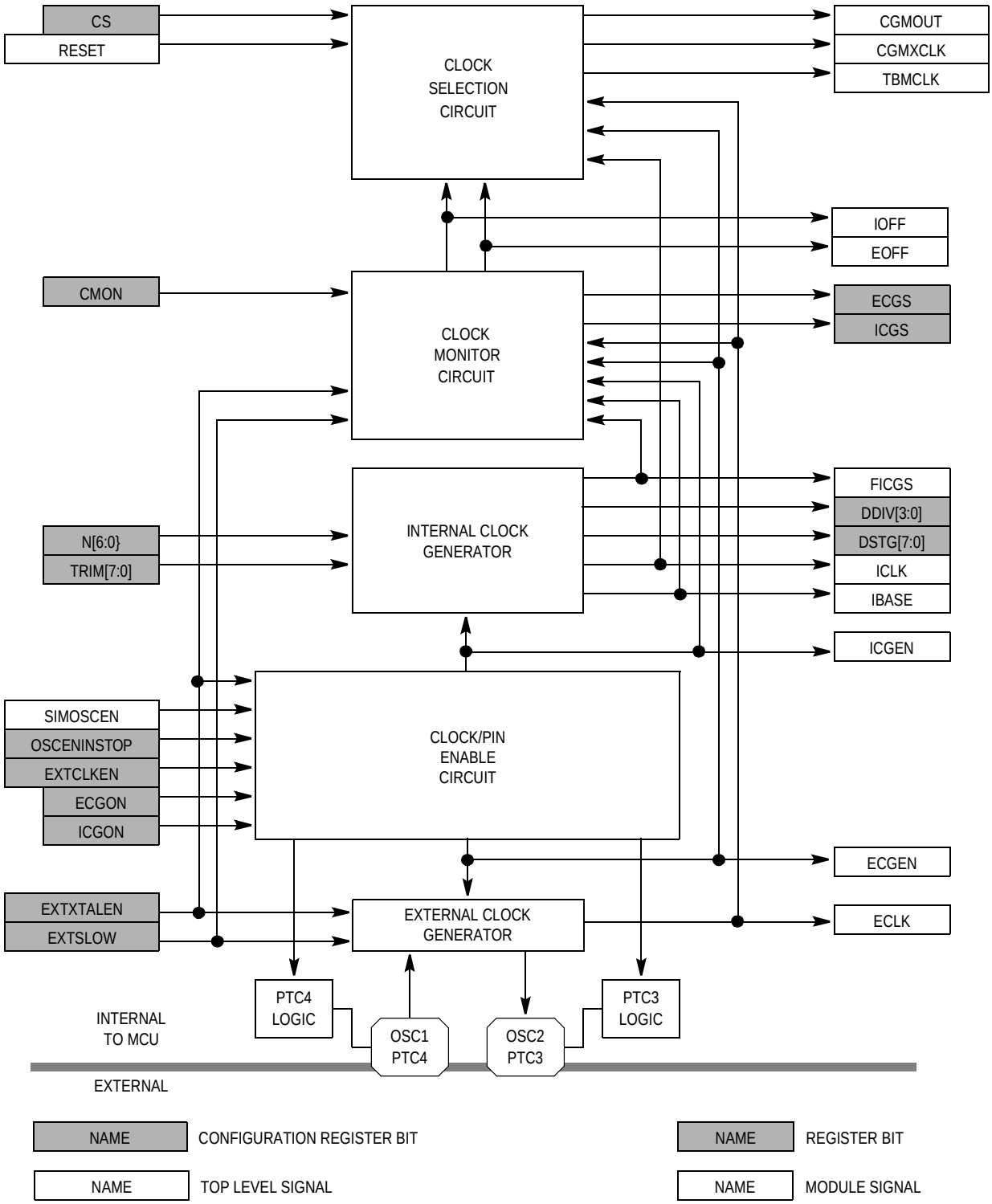


Figure 7-1. ICG Module Block Diagram

7.4.1 Clock Enable Circuit

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

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

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

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

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

7.4.2 Internal Clock Generator

The internal clock generator, shown in Figure 7-2, creates a low frequency base clock (IBASE), which operates at a nominal frequency (f_{NOM}) of 307.2 kHz ± 25 percent, and an internal clock (ICLK) which is an integer multiple of IBASE. This multiple is the ICG multiplier factor (N), which is programmed in the ICG multiplier register (ICGMR). The internal clock generator is turned off and the output clocks (IBASE and ICLK) are held low when the internal clock generator enable signal (ICGEN) is clear.

The internal clock generator contains:

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

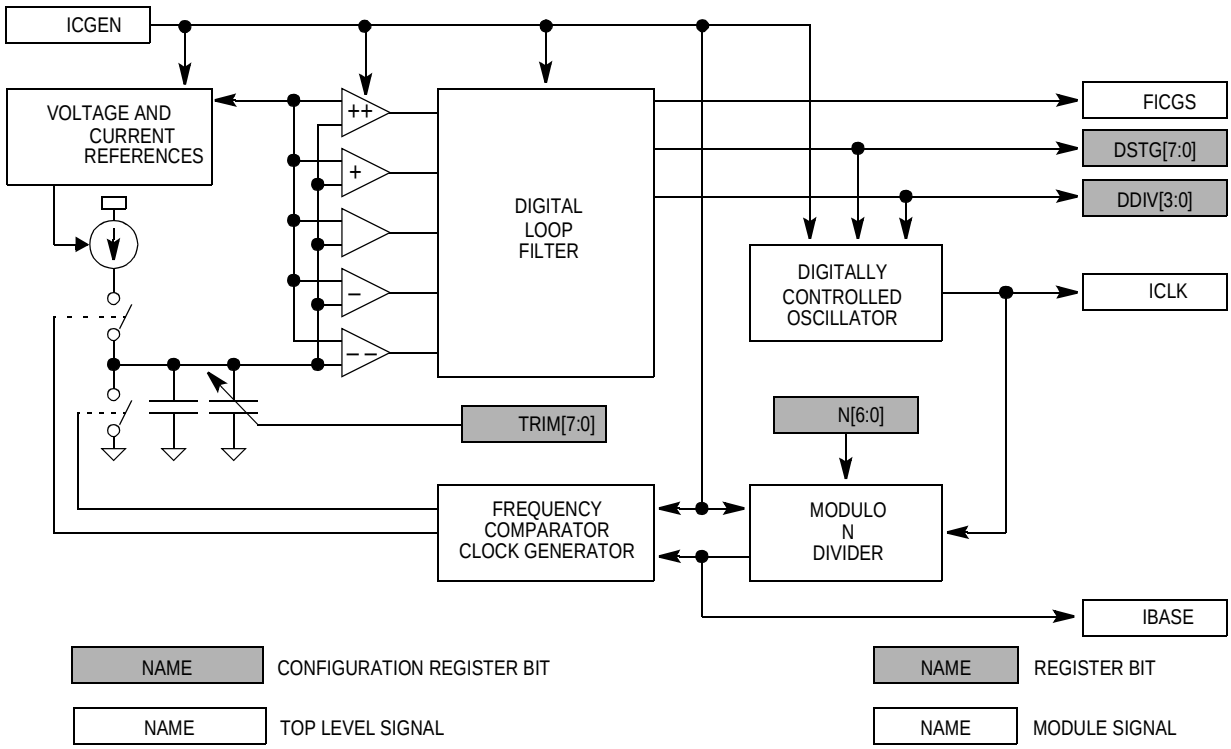


Figure 7-2. Internal Clock Generator Block Diagram

7.4.2.1 Digitally Controlled Oscillator

The digitally controlled oscillator (DCO) is an inaccurate oscillator which generates the internal clock (ICLK). The clock period of ICLK is dependent on the digital loop filter outputs (DSTG[7:0] and DDIV[3:0]). Because of only a limited number of bits in DDIV and DSTG, the precision of the output (ICLK) is restricted to a precision of approximately ±0.202 percent to ±0.368 percent when measured over several cycles (of the desired frequency). Additionally, since the propagation delays of the devices used in the DCO ring oscillator are a measurable fraction of the bus clock period, reaching the long-term precision may require alternately running faster and slower than desired, making the worst case cycle-to-cycle frequency variation ±6.45 percent to ±11.8 percent (of the desired frequency). The valid values of DDIV:DSTG range from $000 to $9FF. For more information on the quantization error in the DCO, see **7.5.4 Quantization Error in DCO Output**.

7.4.2.2 Modulo N Divider

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

7.4.2.3 Frequency Comparator

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

7.4.2.4 Digital Loop Filter

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

Table 7-1. Correction Sizes from DLF to DCO

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

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

7.4.3 External Clock Generator

The ICG also provides for an external oscillator or external clock source, if desired. The external clock generator, shown in Figure 7-3, contains an external oscillator amplifier and an external clock input path.

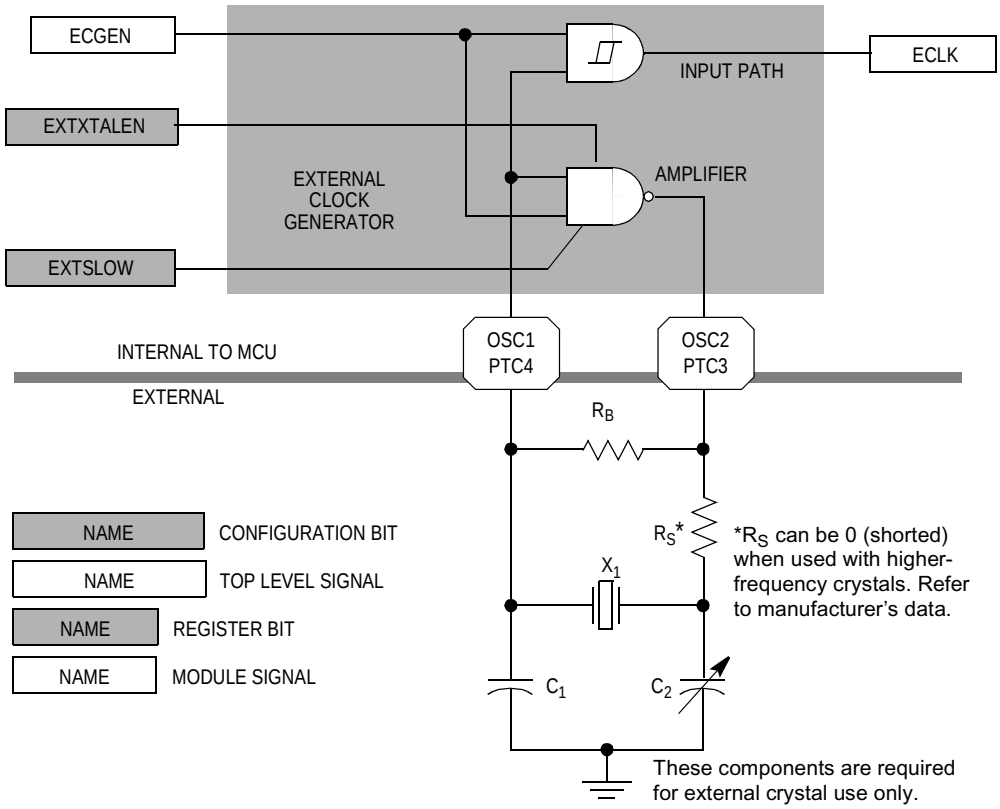


Figure 7-3. External Clock Generator Block Diagram

7.4.3.1 External Oscillator Amplifier

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

The amplifier is enabled when the external clock generator enable (ECGEN) signal is set and when the external crystal enable (EXTXTALEN) bit in the CONFIG is set. ECGEN is controlled by the clock enable circuit (see **7.4.1 Clock Enable Circuit**) and indicates that the external clock function is desired. When enabled, the amplifier will be connected between the PTC4/OSC1 and PTC3/OSC2 pins. Otherwise, the PTC3/OSC2 pin reverts to its port function.

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

1. Crystal, X_1
2. Fixed capacitor, C_1
3. Tuning capacitor, C_2 (can also be a fixed capacitor)
4. Feedback resistor, RB
5. Series resistor, R_S (included in Figure 7-3 to follow strict Pierce oscillator guidelines and may not be required for all ranges of operation, especially with high frequency crystals. Refer to the crystal manufacturer's data for more information.)

7.4.3.2 External Clock Input Path

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

7.4.4 Clock Monitor Circuit

The ICG contains a clock monitor circuit which, when enabled, will continuously monitor both the external clock (ECLK) and the internal clock (ICLK) to determine if either clock source has been corrupted. The clock monitor circuit, shown in Figure 7-4, contains these blocks:

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

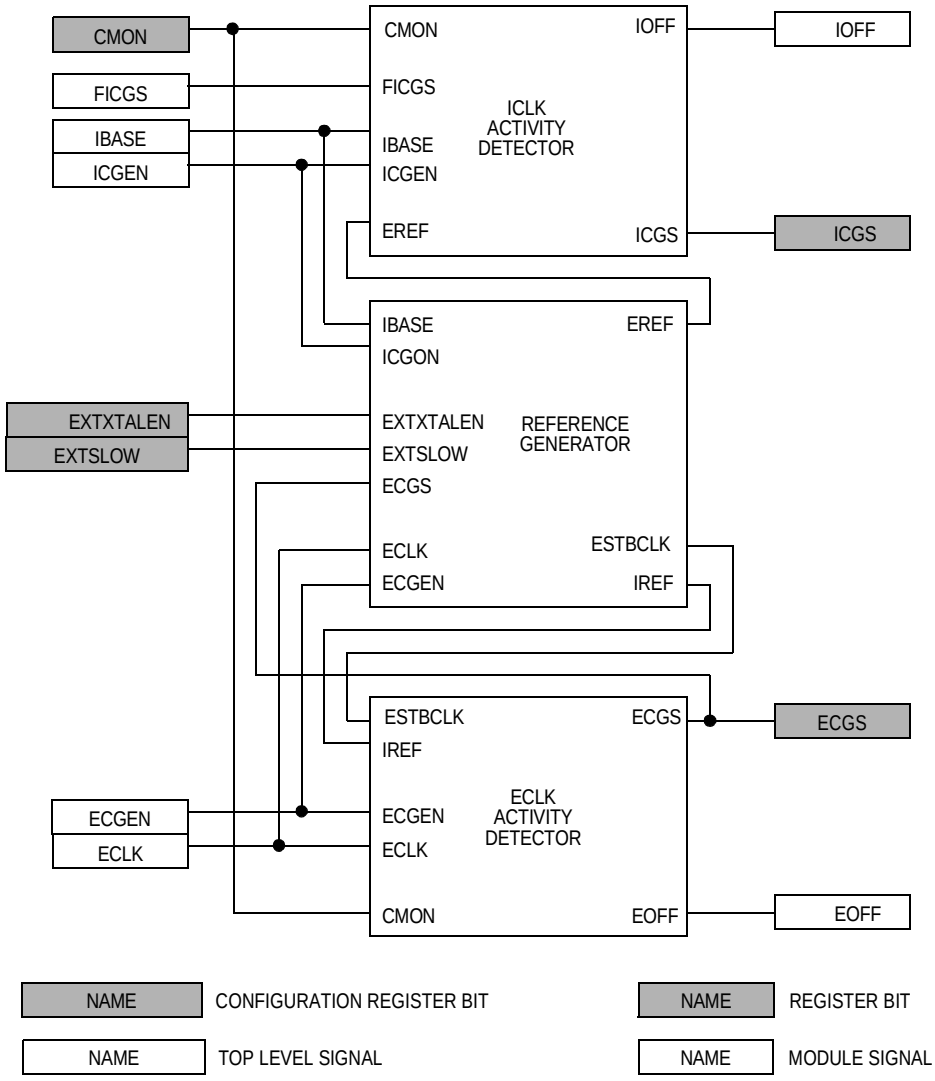


Figure 7-4. Clock Monitor Block Diagram

7.4.4.1 Clock Monitor Reference Generator

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

To guarantee that IREF is slower than ECLK and EREF is slower than IBASE, one of the signals is divided down. Which signal is divided and by how much is determined by the external slow (EXTSLOW) and external crystal enable (EXTXTALEN) bits in the CONFIG, according to the rules in Table 7-2.

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

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

7.4.4.2 Internal Clock Activity Detector

The internal clock activity detector, shown in Figure 7-5, looks for at least one falling edge on the low-frequency base clock (IBASE) every time the external reference (EREF) is low. Since EREF is less than half the frequency of IBASE, this should occur every time. If it does not occur two consecutive times, the internal clock inactivity indicator (IOFF) is set. IOFF will be cleared the next time there is a falling edge of IBASE while EREF is low.

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

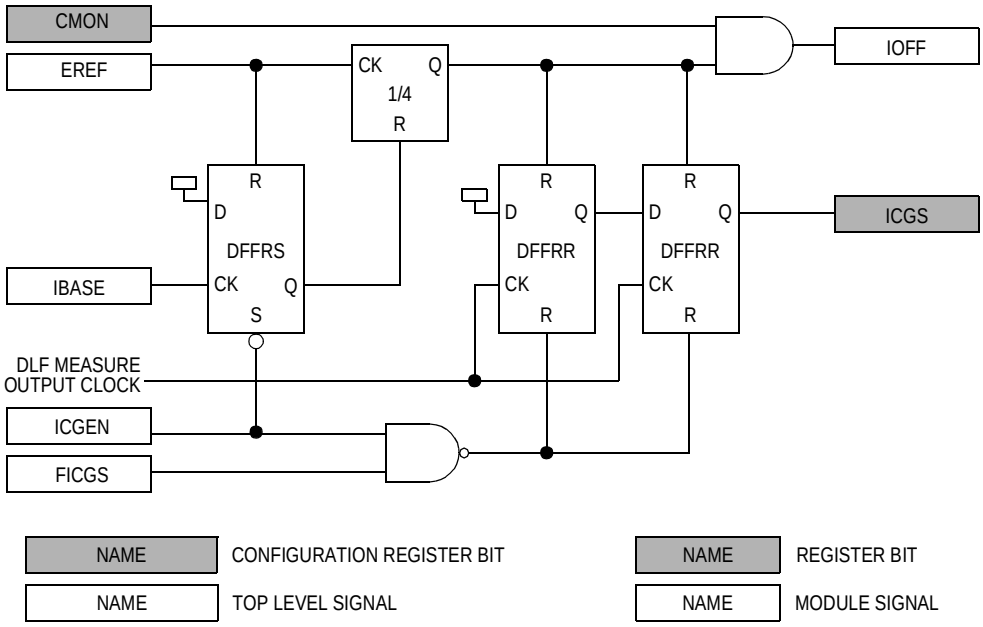


Figure 7-5. Internal Clock Activity Detector

7.4.4.3 External Clock Activity Detector

The external clock activity detector, shown in Figure 7-6, looks for at least one falling edge on the external clock (ECLK) every time the internal reference (IREF) is low. Since IREF is less than half the frequency of ECLK, this should occur every time. If it does not occur two consecutive times, the external clock inactivity indicator (EOFF) is set. EOFF will be cleared the next time there is a falling edge of ECLK while IREF is low.

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

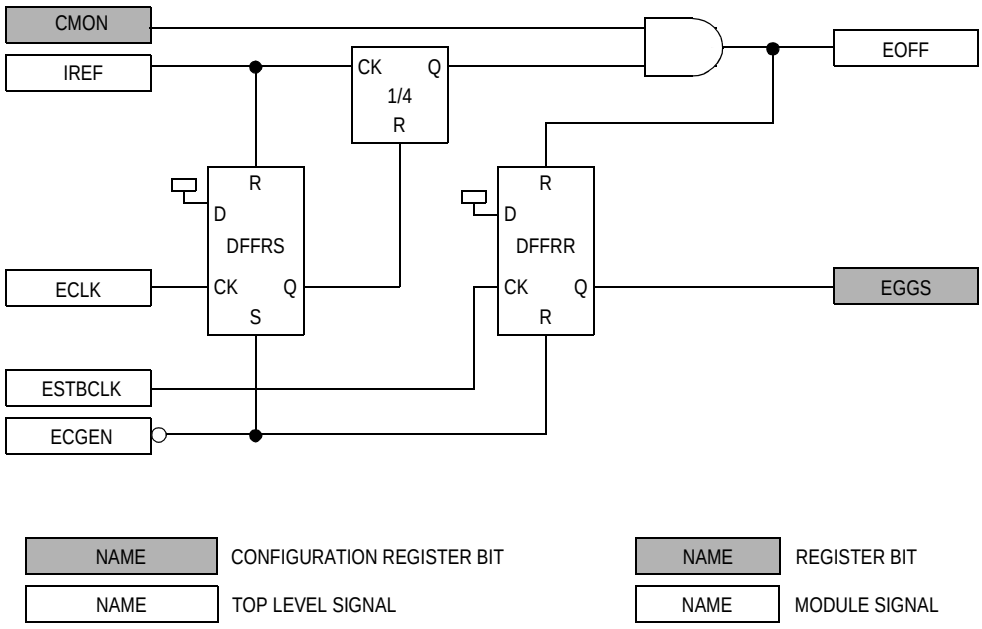


Figure 7-6. External Clock Activity Detector

7.4.5 Clock Selection Circuit

The clock selection circuit, shown in Figure 7-7, contains two clock switches which generate the oscillator output clock (CGMXCLK) and the timebase clock (TBMCLK) from either the internal clock (ICLK) or the external clock (ECLK). The clock selection circuit also contains a divide-by-two circuit which creates the clock generator output clock (CGMOUT), which generates the bus clocks.

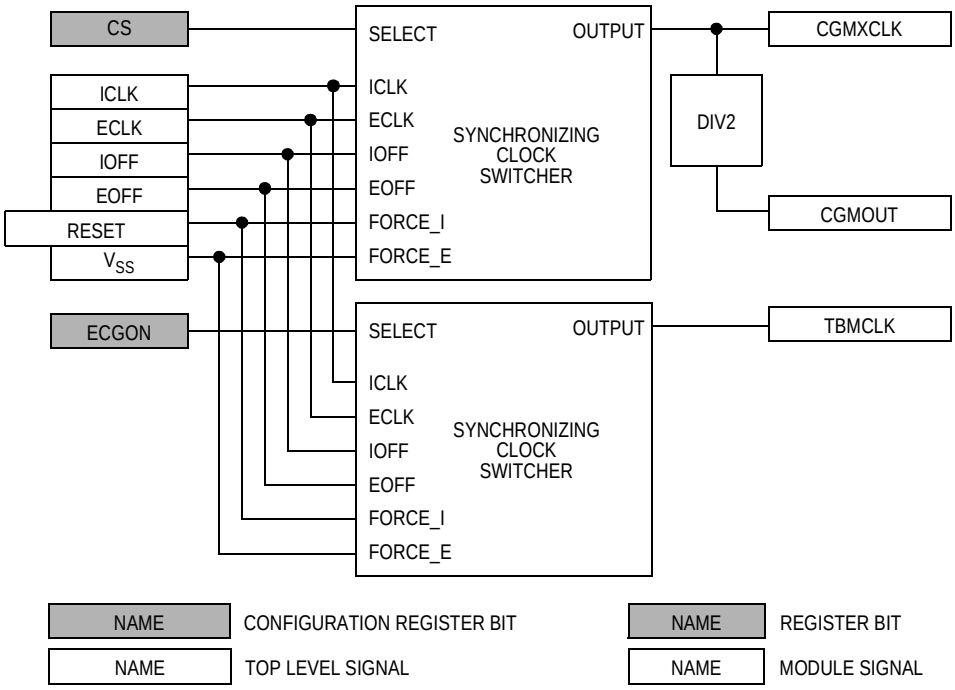


Figure 7-7. Clock Selection Circuit Block Diagram

7.4.5.1 Clock Selection Switches

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

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

7.4.5.2 Clock Switching Circuit

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

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

7.5 Usage Notes

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

These notes include:

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

7.5.1 Switching Clock Sources

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

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

The key point to remember in this flow is that the clock source cannot be switched (CS cannot be written) unless the desired clock is on and stable. A short assembly code example of how to employ this flow is shown in Figure 7-8. This code is for illustrative purposes only and does not represent valid syntax for any particular assembler.

```
                        ;Clock Switching Code Example
                        ;This code switches from Internal to External clock
                        ;Clock Monitor and interrupts are not enabled
start   lda     #$13    ;Mask for CS, ECGON, ECGS
                        ; If switching from External to Internal, mask is $0C.
loop    **      **      ;Other code here, such as writing the COP, since ECGS may
                        ; take some time to set
        sta     icgcr   ;Try to set CS, ECGON and clear ICGON. ICGON will not
                        ; clear until CS is set, and CS will not set until
                        ; ECGON and ECGS are set.
        cmpa    icgcr   ;Check to see if ECGS set, then CS set, then ICGON clear
        bne     loop    ;Keep looping until ICGON is clear.
```

Figure 7-8. Code Example for Switching Clock Sources

7.5.2 Enabling the Clock Monitor

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

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

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

These events must happen in sequence. A short assembly code example of how to employ this flow is shown in Figure 7-9. This code is for illustrative purposes only and does not represent valid syntax for any particular assembler.

```
                                ;Clock Monitor Enabling Code Example
                                ;This code turns on both clocks, selects the desired
                                ; one, then turns on the Clock Monitor and Interrupts
start   lda     #$AF            ;Mask for CMIE, CMON, ICGON, ICGS, ECGON, ECGS
                                ; If Internal Clock desired, mask is $AF
                                ; If External Clock desired, mask is $BF
                                ; If interrupts not desired mask is $2F int; $3F ext
loop    **      **              ;Other code here, such as writing the COP, since ECGS
                                ; and ICGS may take some time to set.
        sta     icgcr           ;Try to set CMIE. CMIE wont set until CMON set; CMON
                                ; won't set until ICGON, ICGS, ECGON, ECGS set.
        brset   6,ICGCR,error   ;Verify CMF is not set
        cmpa    icgcr           ;Check if ECGS set, then CMON set, then CMIE set
        bne     loop            ;Keep looping until CMIE is set.
```

Figure 7-9. Code Example for Enabling the Clock Monitor

7.5.3 Using Clock Monitor Interrupts

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

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

7.5.4 Quantization Error in DCO Output

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

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

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

7.5.4.1 Digitally Controlled Oscillator

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

Table 7-2. Quantization Error in ICLK

DDIV[3:0]	ICLK Cycles	Bus Cycles	τ_{ICLK} Q-ERR
%0000 (min)	1	NA	6.45%–11.8%
%0000 (min)	4	1	1.61%–2.94%
%0000 (min)	≥ 32	≥ 8	0.202%–0.368%
%0001	1	NA	3.23%–5.88%
%0001	4	1	0.806%–1.47%
%0001	≥ 16	≥ 4	0.202%–0.368%

Table 7-2. Quantization Error in ICLK (Continued)

DDIV[3:0]	ICLK Cycles	Bus Cycles	τ_{ICLK} Q-ERR
%0010	1	NA	1.61%–2.94%
%0010	4	1	0.403%–0.735%
%0010	≥ 8	≥ 2	0.202%–0.368%
%0011	1	NA	0.806%–1.47%
%0011	≥ 4	≥ 1	0.202%–0.368%
%0100	1	NA	0.403%–0.735%
%0100	≥ 2	≥ 1	0.202%–0.368%
%0101–%1001 (max)	≥ 1	≥ 1	0.202%–0.368%

7.5.4.2 Binary Weighted Divider

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

7.5.4.3 Variable-Delay Ring Oscillator

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

7.5.4.4 Ring Oscillator Fine-Adjust Circuit

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

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

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

7.5.5 Switching Internal Clock Frequencies

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

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

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

- Verify there is no clock monitor interrupt by reading the CMF bit.
- Turn off the clock monitor.
- If desired, switch to the external clock (see **7.5.1 Switching Clock Sources**).
- Change the value of N.
- Switch back to internal (see **7.5.1 Switching Clock Sources**), if desired.
- Turn on the clock monitor (see **7.5.2 Enabling the Clock Monitor**), if desired.

7.5.6 Nominal Frequency Settling Time

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

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

7.5.6.1 Settling to Within 15 Percent

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

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

Note that increasing speed takes much longer than decreasing speed since N is higher. This can be expressed in terms of the initial clock period (τ_1) minus the final clock period (τ_2) as such:

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

7.5.6.2 Settling to Within 5 Percent

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

usable, although the error will be as high as 5 percent. The total time to this point is:

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

7.5.6.3 Total Settling Time

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

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

The equations for τ_{15}, τ_5, and τ_{tot} are dependent on the actual initial and final clock periods τ_1 and τ_2, not the nominal. This means the variability in the ICLK frequency due to process, temperature, and voltage must be considered. Additionally, other process factors and noise can affect the actual tolerances of the points at which the filter changes modes. This means a worst case adjustment of up to 35 percent (ICLK clock period tolerance plus 10 percent) must be added. This adjustment can be reduced with trimming. Table 7-3 shows some typical values for settling time.

Table 7-3. Typical Settling Time Examples

τ_1	τ_2	N	τ_{15}	τ_5	τ_{tot}
1/ (6.45 MHz)	1/ (25.8 MHz)	84	430 µs	535 µs	850 µs
1/ (25.8 MHz)	1/ (6.45 MHz)	21	107 µs	212 µs	525 µs
1/ (25.8 MHz)	1/ (307.2 kHz)	1	141 µs	246 µs	560 µs
1/ (307.2 kHz)	1/ (25.8 MHz)	84	11.9 ms	12.0 ms	12.3 ms

7.5.7 Trimming Frequency on the Internal Clock Generator

The unadjusted frequency of the low-frequency base clock (IBASE), when the comparators in the frequency comparator indicate zero error, will vary as much as ±25 percent due to process, temperature, and voltage dependencies. These dependencies are in the voltage and current references, the offset of the comparators, and the internal capacitor. The voltage and temperature dependencies have been designed to be a maximum of approximately ±1 percent error. The process dependencies account for the rest.

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

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

7.6 Low-Power Modes

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

7.6.1 Wait Mode

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

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

7.6.2 Stop Mode

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

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

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

7.7 CONFIG Options

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

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

All CONFIG options will have a default setting. Refer to **Section 8. Configuration Registers (CONFIG1 & CONFIG2)** on how the CONFIG is used.

7.7.1 External Clock Enable (EXTCLKEN)

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

The default state for this option is clear.

7.7.2 External Crystal Enable (EXTXTALEN)

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

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

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

The default state for this option is clear.

7.7.3 Slow External Clock (EXTSLOW)

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

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

The default state for this option is clear.

7.7.4 Oscillator Enable In Stop (OSCENINSTOP)

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

The default state for this option is clear.

7.8 Input/Output (I/O) Registers

The ICG contains five registers, summarized in Figure 7-10. These registers are:

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

Several of the bits in these registers have interaction where the state of one bit may force another bit to a particular state or prevent another bit from being set or cleared. A summary of this interaction is shown in Table 7-4.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0036	ICG Control Register (ICGCR) See 142.	Read:	CMIE	CMF	CMON	CS	ICGON	ICGS	ECGON	ECGS
		Write:		0*						
		Reset:	0	0	0	0	1	0	0	0
$0037	ICG Multiply Register (ICGMR) See 144.	Read:		N6	N5	N4	N3	N2	N1	N0
		Write:								
		Reset:	0	0	0	1	0	1	0	1
$0038	ICG Trim Register (ICGTR) See 145.	Read:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
		Write:								
		Reset:	1	0	0	0	0	0	0	0
$0039	ICG Divider Control Register (ICGDVR) See 145.	Read:					DDIV3	DDIV2	DDIV1	DDIV0
		Write:								
		Reset:	0	0	0	0	U	U	U	U
$003A	ICG DCO Stage Control Register (ICGDSR) See 146.	Read:	DSTG7	DSTG6	DSTG5	DSTG4	DSTG3	DSTG2	DSTG1	DSTG0
		Write:	R	R	R	R	R	R	R	R
		Reset:	U	U	U	U	U	U	U	U

= Unimplemented R = Reserved U = Unaffected

*See **7.8.1 ICG Control Register** for method of clearing the CMF bit.

Figure 7-10. ICG Module I/O Register Summary

Table 7-4. ICG Module Register Bit Interaction Summary

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

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

7.8.1 ICG Control Register

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

Address: $0036

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

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

= Unimplemented

Figure 7-11. ICG Control Register (ICGCR)

CMIE — Clock Monitor Interrupt Enable Bit

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

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

CMF — Clock Monitor Interrupt Flag

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

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

CMON — Clock Monitor On Bit

This read/write bit enables the clock monitor. CMON can be set when both ICLK and ECLK have been on and stable for at least one bus cycle. (ICGON, ECGON, ICGS, and ECGS are all set.) CMON is

forced set when CMF is set, to avoid inadvertent clearing of CMF. CMON is forced clear when either ICGON or ECGON is clear, during stop mode with OSCENINSTOP low, or during reset.

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

CS — Clock Select Bit

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

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

ICGON — Internal Clock Generator On Bit

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

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

ICGS — Internal Clock Generator Stable Bit

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

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

ECGON — External Clock Generator On Bit

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

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

ECGS — External Clock Generator Stable Bit

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

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

7.8.2 ICG Multiplier Register

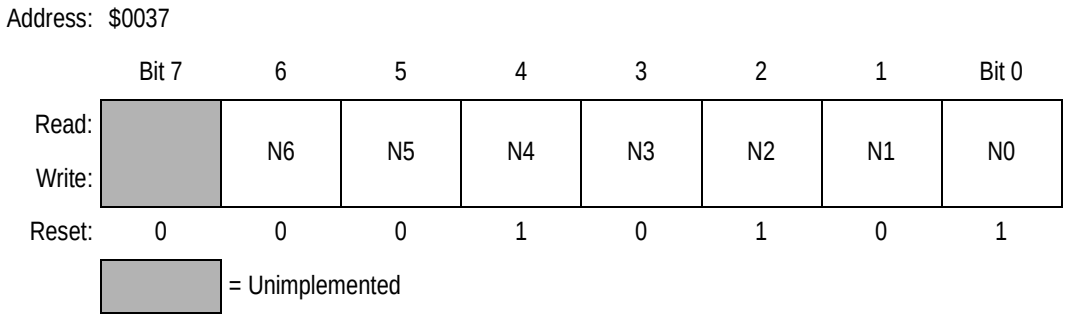


Figure 7-12. ICG Multiplier Register (ICGMR)

N6:N0 — ICG Multiplier Factor Bits

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

(307.2 kHz ± 25 percent) * N

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

7.8.3 ICG Trim Register

Address: $0038

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

Figure 7-13. ICG Trim Register (ICGTR)

TRIM7:TRIM0 — ICG Trim Factor Bits

These read/write bits change the size of the internal capacitor used by the internal clock generator. By testing the frequency of the internal clock and incrementing or decrementing this factor accordingly, the accuracy of the internal clock can be improved to ± 2 percent. Incrementing this register by one decreases the frequency by 0.195 percent of the unadjusted value. Decrementing this register by one increases the frequency by 0.195 percent. This register cannot be written when the CMON bit is set. Reset sets these bits to $80, centering the range of possible adjustment.

7.8.4 ICG DCO Divider Register

Address: $0039

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

= Unimplemented U = Unaffected

Figure 7-14. ICG DCO Divider Control Register (ICGDVR)

DDIV3:DDIV0 — ICG DCO Divider Control Bits

These bits indicate the number of divide-by-twos (DDIV) that follow the digitally controlled oscillator. When ICGON is set, DDIV is controlled by the digital loop filter. The range of valid values for DDIV

is from $0 to $9. Values of $A through $F are interpreted the same as $9. Since the DCO is active during reset, reset has no effect on DSTG and the value may vary.

7.8.5 ICG DCO Stage Register

Address: $003A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	DSTG7	DSTG6	DSTG5	DSTG4	DSTG3	DSTG2	DSTG1	DSTG0
Write:	R	R	R	R	R	R	R	R
Reset:	U	U	U	U	U	U	U	U

R = Reserved U = Unaffected

Figure 7-15. ICG DCO Stage Control Register (ICGDSR)

DSTG7:DSTG0 — ICG DCO Stage Control Bits

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

Section 8. Configuration Registers (CONFIG1 & CONFIG2)

8.1 Contents

8.2 Introduction

This section describes the configuration registers, CONFIG1 and CONFIG2. The configuration registers control these options:

- Stop mode recovery time, 32 CGMXCLK cycles or 4096 CGMXCLK cycles
- Computer operating properly (COP) timeout period, 2^{18}–2^4 or 2^{13}–2^4 CGMXCLK cycles
- STOP instruction
- Computer operating properly (COP) module
- Low-voltage inhibit (LVI) module control and voltage trip point selection
- Enable/disable the oscillator (OSC) during stop mode
- External clock/crystal source control
- Enhanced SCI clock source selection

8.3 Functional Description

The configuration registers are used in the initialization of various options and can be written once after each reset. All of the configuration register bits are cleared during reset. Since the various options affect the operation of the microcontroller unit (MCU), it is recommended that these registers be written immediately after reset. The configuration registers are located at $001E and $001F. For compatibility, a write to a read-only memory (ROM) version of the MCU at this location will have no effect. The configuration register may be read at anytime.

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

On a FLASH device, the CONFIG registers are special registers containing one-time writable latches after each reset. Upon a reset, the CONFIG registers default to predetermined settings as shown in Figure 8-1 and Figure 8-2.

Address: $001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	R	ESCIBD-SRC	EXT-XTALEN	EXT-SLOW	EXT-CLKEN	TMB-CLKSEL	OSCENIN-STOP	SSB-PUENB
Reset:	0	0	0	0	0	0	0	1

= Unimplemented R = Reserved

Figure 8-1. Configuration Register 2 (CONFIG2)

Address: $001F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVI5OR3(1)	SSREC	STOP	COPD
Reset:	0	0	0	0	1	0	0	0

= Unimplemented R = Reserved

Figure 8-2. Configuration Register 1 (CONFIG1)

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

ESCIBDSRC — ESCI Baud Rate Clock Source Bit

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

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

EXTXTALEN — External Crystal Enable Bit

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

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

Clearing the EXTXTALEN bit (default setting) allows the PTC3/OSC2 pin to function as a general-purpose I/O pin. Refer to Table 8-1 for configuration options for the external source. See **Section 7. Internal Clock Generator (ICG) Module** for a more detailed description of the external clock operation.

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

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

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

EXTSLOW — Slow External Crystal Enable Bit

The EXTSLOW bit has two functions. It configures the ICG module for a fast (1 MHz to 8 MHz) or slow (30 kHz to 100 kHz) speed crystal. The option also configures the clock monitor operation in the ICG

module to expect an external frequency higher (307.2 kHz to 32 MHz) or lower (60 Hz to 307.2 kHz) than the base frequency of the internal oscillator. See **Section 7. Internal Clock Generator (ICG) Module**.

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

Table 8-1. External Clock Option Settings

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

EXTCLKEN — External Clock Enable Bit

EXTCLKEN enables an external clock source or crystal/ceramic resonator to be used as a clock input. Setting this bit enables PTC4/OSC1 pin to be a clock input pin. Clearing this bit (default setting) allows the PTC4/OSC1 and PTC3/OSC2 pins to function as general-purpose input/output (I/O) pins. Refer to Table 8-1 for configuration options for the external source. See **Section 7. Internal Clock Generator (ICG) Module** for a more detailed description of the external clock operation.

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

TMBCLKSEL — Timebase Clock Select Bit

TMBCLKSEL enables an enable the extra divide by 128 prescaler in the timebase module. Setting this bit enables the extra prescaler and clearing this bit disables it. Refer to Table 20-1 for timebase divider selection details.

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

OSCENINSTOP — Oscillator Enable In Stop Mode Bit

OSCENINSTOP, when set, will enable the internal clock generator module to continue to generate clocks (either internal, ICLK, or external, ECLK) in stop mode. See **Section 7. Internal Clock Generator (ICG) Module**. This function is used to keep the timebase running while the rest of the microcontroller stops. When clear, all clock generation will cease and both ICLK and ECLK will be forced low during stop mode. The default state for this option is clear, disabling the ICG in stop mode.

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

NOTE: *This bit has the same functionality as the OSCSTOPENB CONFIG bit in MC68HC908GP20 and MC68HC908GR8 parts.*

SSBPUENB — $\overline{SS}$ Pull-up Enable Bit

Clearing SSBPUENB enables the $\overline{SS}$ pull-up resistor.

1 = Disables $\overline{SS}$ pull-up resistor.
0 = Enables $\overline{SS}$ pull-up resistor.

COPRS — COP Rate Select Bit

COPRS selects the COP timeout period. Reset clears COPRS. See **Section 11. Computer Operating Properly (COP) Module**.

1 = COP timeout period = $2^{13} – 2^4$ CGMXCLK cycles
0 = COP timeout period = $2^{18} – 2^4$ CGMXCLK cycles

LVISTOP — LVI Enable in Stop Mode Bit

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

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

LVIRSTD — LVI Reset Disable Bit

LVIRSTD disables the reset signal from the LVI module. See **Section 12. Low-Voltage Inhibit (LVI) Module**.

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

LVIPWRD — LVI Power Disable Bit

LVIPWRD disables the LVI module. See **Section 12. Low-Voltage Inhibit (LVI) Module**.

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

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

LVI5OR3 selects the voltage operating mode of the LVI module. See **Section 12. Low-Voltage Inhibit (LVI) Module**. The voltage mode selected for the LVI will typically be 5V. However, users may choose to operate the LVI in 3V mode if desired. See **Section 23. Electrical Specifications** for the LVI's voltage trip points for each of the modes.

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

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

SSREC — Short Stop Recovery Bit

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

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

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

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

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

STOP — STOP Instruction Enable Bit

STOP enables the STOP instruction.

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

COPD — COP Disable Bit

COPD disables the COP module. See **Section 11. Computer Operating Properly (COP) Module**.

1 = COP module disabled
0 = COP module enabled

Configuration Registers (CONFIG1 & CONFIG2)

Section 9. Break Module (BRK)

9.1 Contents

9.2 Introduction

The break module (BRK) can generate a break interrupt that stops normal program flow at a defined address to enter a background program.

9.3 Features

Features include:

- Accessible input/output (I/O) registers during break interrupts
- Central processor unit (CPU) generated break interrupts
- Software generated break interrupts
- Computer operating properly (COP) disabling during break interrupts

9.4 Functional Description

When the internal address bus matches the value written in the break address registers, the break module issues a breakpoint signal to the CPU. The CPU then loads the instruction register with a software interrupt instruction (SWI) after completion of the current CPU instruction. The program counter vectors to $FFFC and $FFFD ($FEFC and $FEFD in monitor mode).

These 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 logic 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 begins after the CPU completes its current instruction. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. Figure 9-1 shows the structure of the break module.

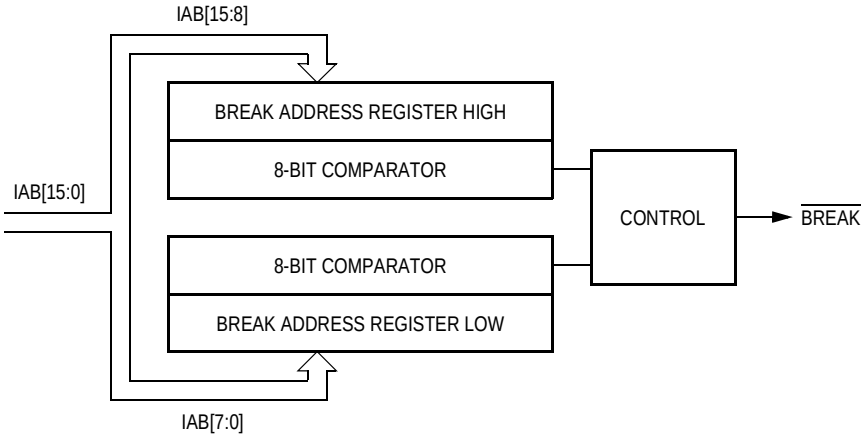


Figure 9-1. Break Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	SIM Break Status Register (SBSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							NOTE	
		Reset:	0	0	0	0	0	0	0	0
$FE03	SIM Break Flag COntrol Register (SBFCR)		BCFE	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE09	Break Address Register High (BRKH) See 161.	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 Register Low (BRKL) See 161.	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 (BSCR) See 160.	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 9-2. I/O Register Summary

9.4.1 Flag Protection During Break Interrupts

The BCFE bit in the break flag control register (SBFCR) enables software to clear status bits during the break state.

9.4.2 CPU During Break Interrupts

The CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with $FFFC:$FFFD ($FEFC:$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.

9.4.3 TIM During Break Interrupts

A break interrupt stops the timer counter.

9.4.4 COP During Break Interrupts

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

9.5 Low-Power Modes

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

9.5.1 Wait Mode

If enabled, the break module is active in wait mode. The SIM break stop/wait bit (SBSW) in the SIM break status register indicates whether wait was exited by a break interrupt. If so, the user can modify the return address on the stack by subtracting one from it. See **9.6.1 Break Status and Control Register**.

9.5.2 Stop Mode

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

9.6 Break Module Registers

These registers control and monitor operation of the break module:

- Break address register high, BRKH
- Break address register low, BRKL
- Break status and control register, BSCR

9.6.1 Break Status and Control Register

The break status and control register contains break module enable and status bits.

Address: $FE0B

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

= Unimplemented

Figure 9-3. Break Status and Control Register (BSCR)

BRKE — Break Enable Bit

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

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

BRKA — Break Active Bit

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

1 = (When read) Break address match
0 = (When read) No break address match

9.6.2 Break Address Registers

The break address registers contain the high and low bytes of the desired breakpoint address. Reset clears the break address registers.

Address: Break Address Register High — $FE09
Break Address Register Low — $FE0A

	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
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 9-4. Break Address Registers (BRKH and BRKL)

9.6.3 SIM Break Status Register

The SIM break status register (SBSR) contains a flag to indicate that a break caused an exit from wait mode.The flag is useful in applications requiring a return to wait mode after exiting from a break interrupt.

Address: $FE00

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

R = Reserved

Note: 1. Writing a logic 0 clears SBSW

Figure 9-5. SIM Break Status Register (SBSR)

SBSW — SIM Break STOP/WAIT

This status bit is useful in applications requiring a return to stop or wait mode after exiting from a break interrupt. SBSW can be cleared by writing a logic 0 to it. Reset clears SBSW.

1 = Stop or wait mode was exited by break interrupt
0 = Stop or wait mode was not exited by break interrupt

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

9.6.4 SIM Break Flag Control Register

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

Address: $FE03

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

R = Reserved

Figure 9-6. SIM Break Flag Control Register (SBFCR)

BCFE — Break Clear Flag Enable Bit

This read/write bit is 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

Section 10. Monitor ROM (MON)

10.1 Contents

10.2 Introduction

This section describes the monitor read-only memory (ROM). The monitor ROM (MON) allows complete testing of the microcontroller unit (MCU) through a single-wire interface with a host computer.

10.3 Features

Features of the monitor ROM include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between monitor ROM and host computer
- Standard mark/space non-return-to-zero (NRZ) communication with host computer
- Execution of code in random-access memory (RAM) or FLASH
- FLASH memory security[1]
- FLASH memory programming interface

10.4 Functional Description

The monitor ROM receives and executes commands from a host computer via a standard RS-232 interface. Simple monitor commands can access any memory address. In monitor mode, the microcontroller unit (MCU) can execute host-computer code in RAM while all 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.

10.5 Monitor Mode Entry

There are two methods for entering monitor mode. The first is the traditional M68HC08 method where V_{TST} is applied to $\overline{IRQ}$ and the mode pins are configured appropriately. A second method, intended for in-circuit programming applications, will force entry into monitor mode without requiring high voltage on the $\overline{IRQ}$ pin when the reset vector locations of the FLASH are erased ($FF).

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

Both of these methods require that the PTA1 pin be pulled low for the first 24 CGMXCLK cycles after the part comes out of reset. This check is used by the monitor code to configure the MCU for serial communication.

In forced monitor mode, the IGC will be selected to clock the device if $\overline{IRQ}$ = VDD.

The mode selection conditions are summarised in Table 10-1.

Table 10-1. Mode Selection

Mode	IRQ	RST	$FFFE/FFFF	ICG	PTB4	PTB3	EXT CLK	BUS FREQ	COP	For Serial Comm.		Comments
										PTA0	Baud Rate	
Reset	X	GND	X	X	X	X	X	0	Disabled	X	0	No operation until $\overline{RST}$ goes high
Monitor	V_{TST}	V_{DD} or V_{TST}	X	OFF	1	0	9.8304 MHz	2.4576 MHz	Disabled	1	9600	
Forced Monitor	V_{DD}	V_{DD}	$FF (blank)	OFF	X	X	9.8304 MHz	2.4576 MHz	Disabled	1	9600	External frequency always divided by 4
	GND	V_{DD}	$FF (blank)	ON	X	X	X	Nominal 2.45 MHz	Disabled	1	Nominal 9600	ICG enabled
User	V_{DD} or GND	V_{TST}	$FF (blank)	OFF	X	X	X	-	Enabled	X	–	Enters user mode – will encounter an illegal address reset
	V_{DD} or GND	V_{DD} or V_{TST}	Not $FF (programmed)	ON	X	X	X	Nominal 1.6 MHz	Enabled	X	–	Enters User mode

10.5.1 Normal Monitor Mode

Normal monitor mode is useful for MCU evaluation, factory testing, and development tool programming operation. Figure 10-1 shows an example circuit used for normal monitor mode. Table 10-2 shows the pin conditions for entering this mode.

Table 10-2. Monitor Mode Entry

$\overline{\text{IRQ}}$ Pin	PTB3 Pin (PTXMOD1)	PTB4 Pin (PTXMOD0)	PTA1 Pin	PTA0 Pin	CGMOUT	Bus Frequency (f_{OP})
V_{TST}	0	1	0	1	$\frac{\text{CGMXCLK}}{2}$	$\frac{\text{CGMOUT}}{2}$

NOTE: *PTA1 = 0 and PTA0 = 1 allow normal serial communications. Parallel communication is available for factory test only.*

The MCU initially comes out of reset using the external clock for its clock source. This overrides the user mode operation of the oscillator circuits where the part comes up using the internally generated oscillator. Running from an external clock allows the MCU, using an appropriate frequency clock source, to communicate with host software at standard baud rates.

NOTE: *While the voltage on $\overline{IRQ}$ is at V_{TST}, the internal clock generator (ICG) module is bypassed and the external square-wave clock becomes the clock source. Dropping $\overline{IRQ}$ to below V_{TST} will remove the bypass and the MCU will revert to the clock source selected by the ICG (as determined by the settings in the ICG registers).*

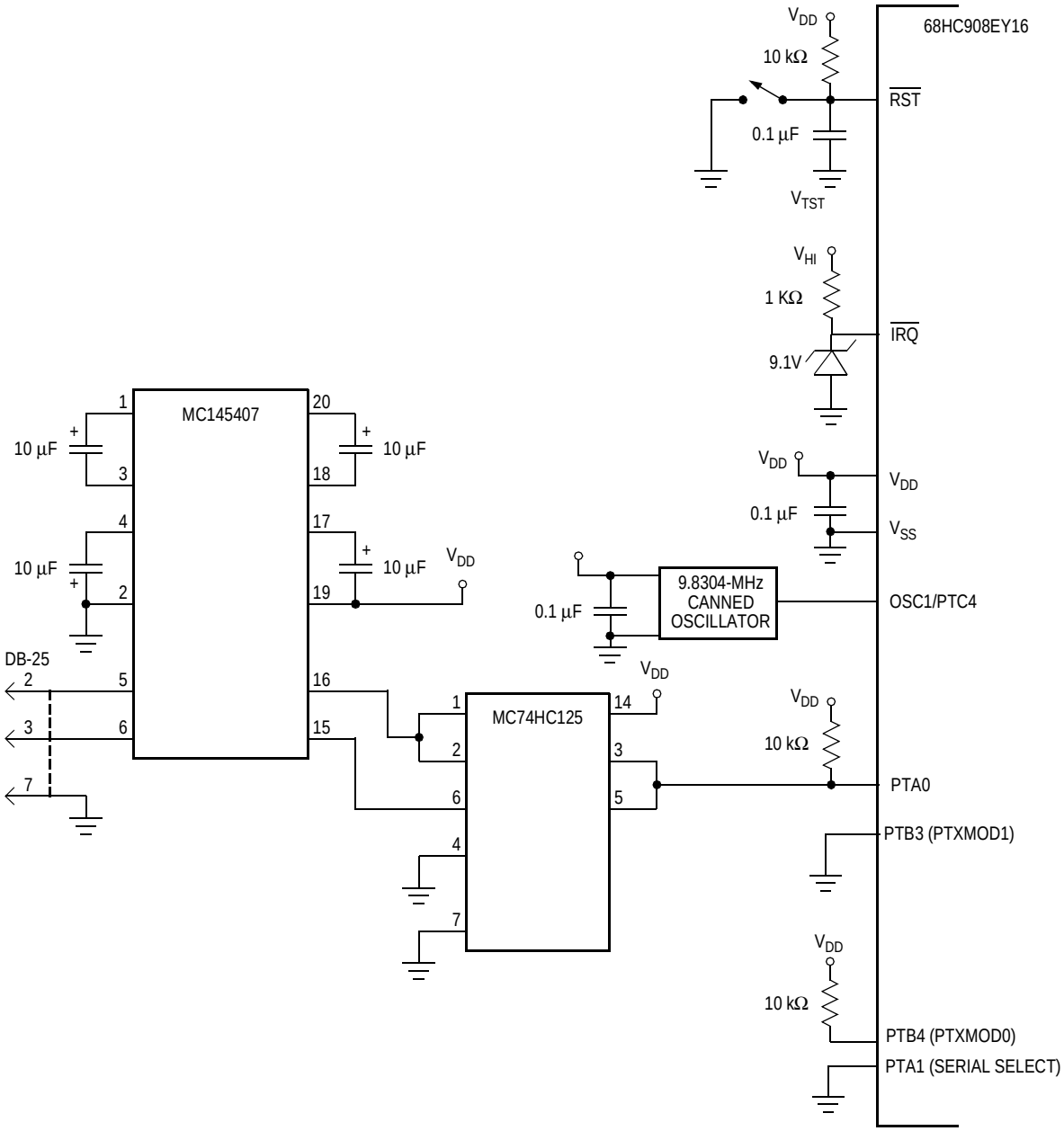


Figure 10-1. Normal Monitor Mode Circuit

The computer operating properly (COP) module is disabled in normal monitor mode whenever V_{TST} is applied to the $\overline{IRQ}$ pin. If the voltage on $\overline{IRQ}$ is less than V_{TST}, the COP module is controlled by the COPD configuration bit.

10.5.2 Forced Monitor Mode

If the voltage applied to the $\overline{\text{IRQ}}$ is less than V_{TST}, the MCU will come out of reset in user mode. The MENRST module is monitoring the reset vector fetches and will assert an internal reset if it detects that the reset vectors are erased ($FF). When the MCU comes out of reset, it is forced into monitor mode without requiring high voltage on the $\overline{\text{IRQ}}$ pin.

Once out of reset, the monitor code is initially executing off the internal clock at its default frequency. The monitor code reconfigures the ICG module to use the external square-wave clock source. Switching to an external clock source allows the MCU, using an appropriate clock frequency, to communicate with host software at standard baud rates.

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.

10.6 Monitor Mode Vectors

Monitor mode uses alternate vectors for reset and SWI interrupts. The alternate vectors are in the $FE page instead of the $FF page and allow code execution from the internal monitor firmware instead of user code. Table 10-3 shows vector differences between user mode and monitor mode.

Table 10-3. Monitor Mode Vector Relocation

Modes	Reset Vector High	Reset Vector Low	SWI Vector High	SWI Vector Low
User	$FFFE	$FFFF	$FFFC	$FFFD
Monitor	$FEFE	$FEFF	$FEFC	$FEFD

10.7 Data Format

The MCU waits for the host to send eight security bytes (see **10.11 Security**). After the security bytes, the MCU sends a break signal (10 consecutive logic 0s) to the host computer, indicating that it is ready to receive a command.

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.

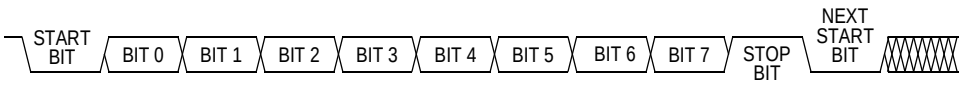


Figure 10-2. Monitor Data Format

10.8 Break Signal

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

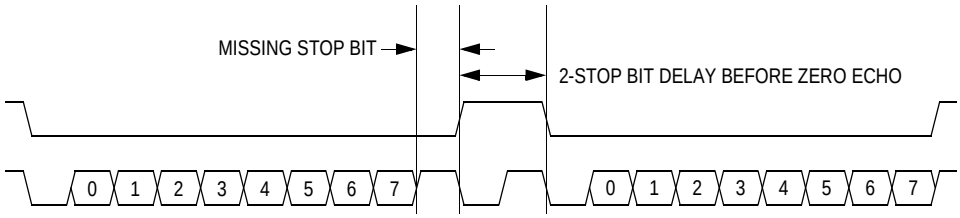


Figure 10-3. Break Transaction

10.9 Baud Rate

The communication baud rate is controlled by the CGMXCLK frequency output of the internal clock generator module.

10.9.1 Force Monitor Mode

In forced monitor mode, the baud rate is fixed at CGMXCLK/1024. A CMGXCLK frequency of 4.9152 MHz results in a 4800 baud rate. A 9.8304-MHz frequency produces a 9600 baud rate.

10.9.2 Normal Monitor Mode

In normal monitor mode, the communication baud rate is controlled by the CGMXCLK frequency output of the internal clock generator module. Table 10-4 lists CGMXCLK frequencies required to achieve standard baud rates. Other standard baud rates can be accomplished using other clock frequencies. The internal clock can be used as the clock source by programming the internal clock generator registers however, monitor mode will always be entered using the external clock as the clock source.

Table 10-4. Normal Monitor Mode Baud Rate Selection

CGMXCLK Frequency (MHz)	Baud Rate
9.8304	9600

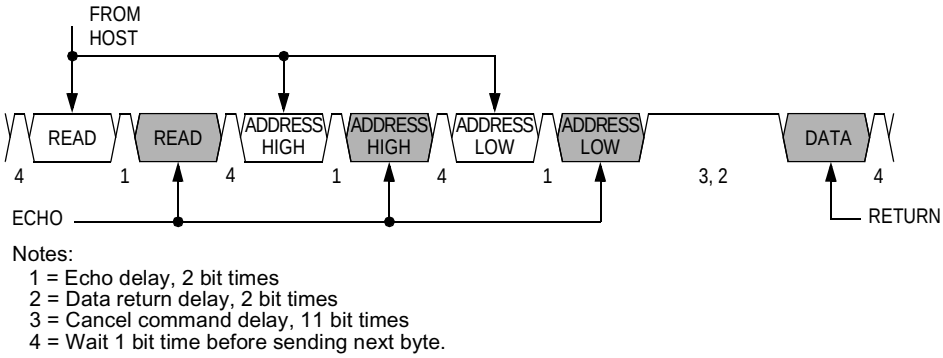
10.10 Commands

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

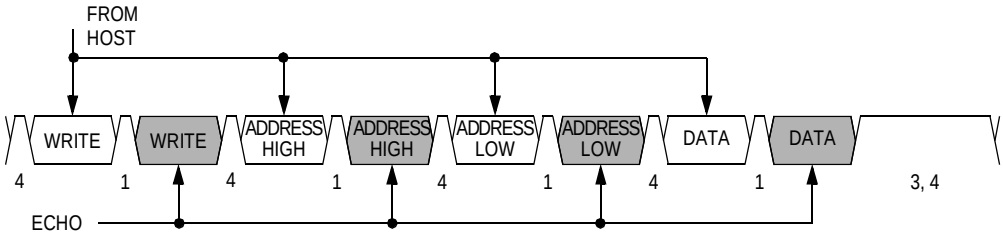
The monitor ROM firmware echoes each received byte back to the PTA0 pin for error checking. An 11-bit delay at the end of each command allows the host to send a break character to cancel the command. A delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. 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, 2 bit times
2 = Data return delay, 2 bit times
3 = Cancel command delay, 11 bit times
4 = Wait 1 bit time before sending next byte.

Figure 10-4. Read Transaction



Notes:
1 = Echo delay, 2 bit times
3 = Cancel command delay, 11 bit times
4 = Wait 1 bit time before sending next byte.

Figure 10-5. Write Transaction

A brief description of each monitor mode command is given here.

Table 10-5. 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

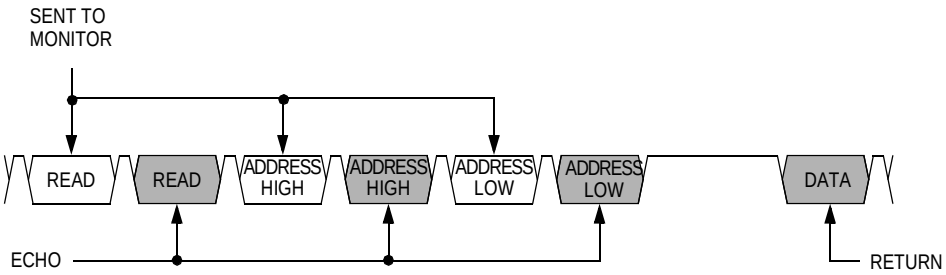


Table 10-6. WRITE (Write Memory) Command

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

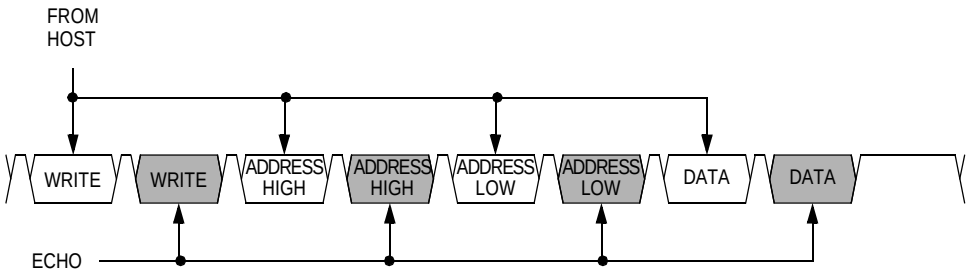


Table 10-7. IREAD (Indexed Read) Command

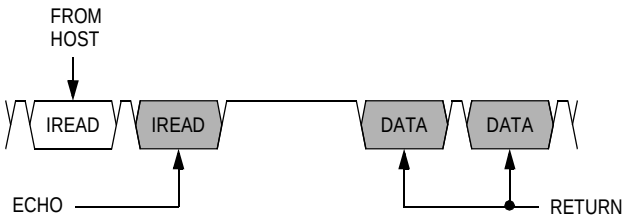
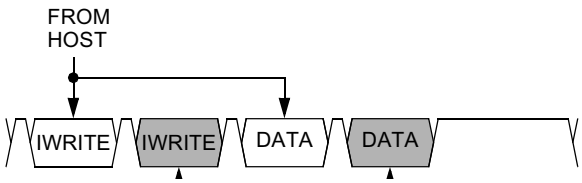
Description	Read next 2 bytes in memory from last address accessed
Operand	2-byte address in high byte:low byte order
Data returned	Returns contents of next two addresses
Opcode	$1A
Command Sequence	

Table 10-8. 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 10-9. READSP (Read Stack Pointer) Command

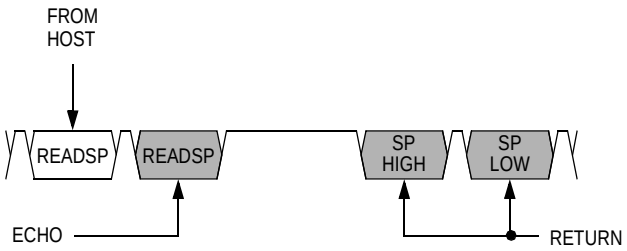
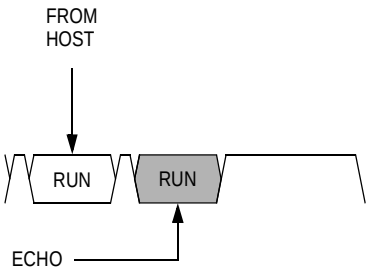
Description	Reads stack pointer
Operand	None
Data returned	Returns incremented stack pointer value (SP + 1) in high byte:low byte order
Opcode	$0C
Command Sequence	

Table 10-10. RUN (Run User Program) Command

Description	Executes PULH and RTI instructions
Operand	None
Data returned	None
Opcode	$28
Command Sequence	

The MCU executes the SWI and PSHH instructions when it enters monitor mode. The RUN command tells the MCU to execute the PULH and RTI instructions. Before sending the RUN command, the host can modify the stacked CPU registers to prepare to run the host program. The READSP command returns the incremented stack pointer value, SP + 1. The high and low bytes of the program counter are at addresses SP + 5 and SP + 6.

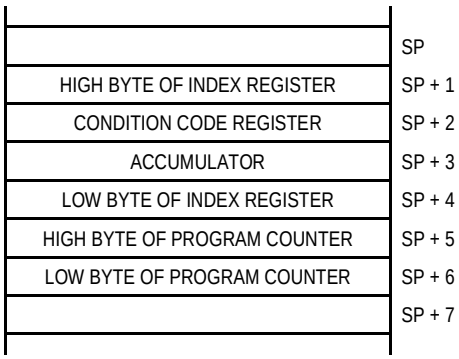


Figure 10-6. Stack Pointer at Monitor Mode Entry

10.11 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. If FLASH is erased, the eight security byte values to be sent to the MCU are $FF, the unprogrammed state of the FLASH.*

During monitor mode entry, a reset must be asserted. PTA1 must be held low during the reset and 24 CGMXCLK cycles after the end of the reset. Then the MCU will wait for eight security bytes on PTA0. Each byte will be echoed back to the host. See Figure 10-7.

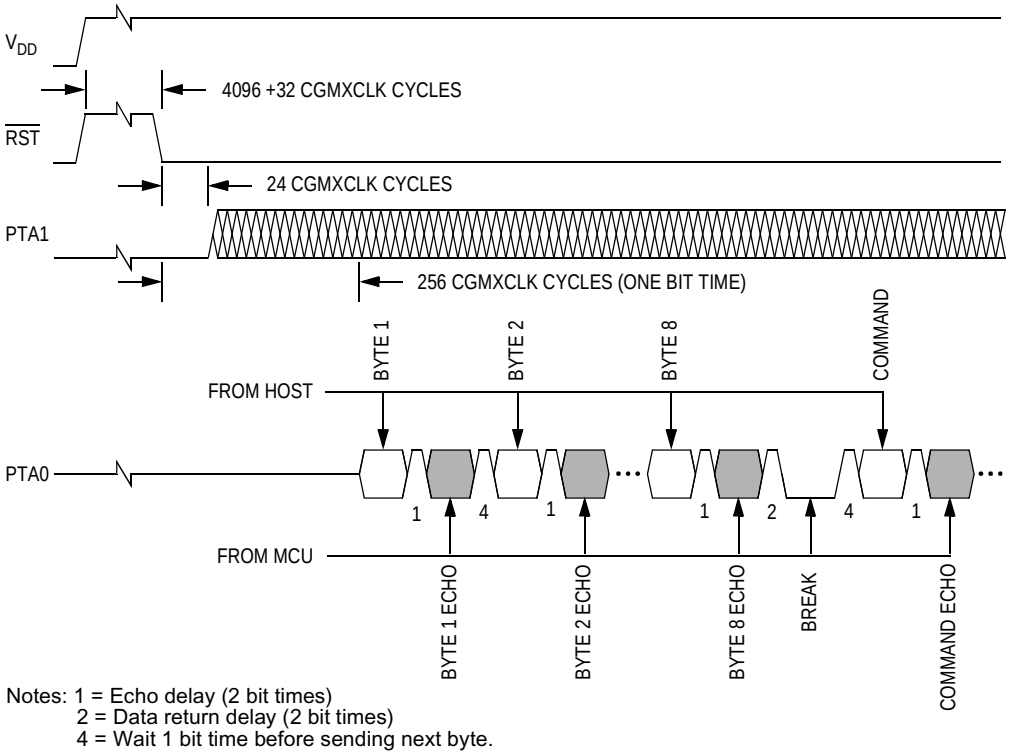


Figure 10-7. Monitor Mode Entry Timing

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 reset occurs. After any reset, security will be locked. To bypass security again, the host must resend the eight security bytes on PTA0.

If the received bytes 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 FLASH locations returns undefined data, 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 signalling 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.*

Section 11. Computer Operating Properly (COP) Module

11.1 Contents

11.2 Introduction

The computer operating properly (COP) module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by periodically clearing the COP counter.

11.3 Functional Description

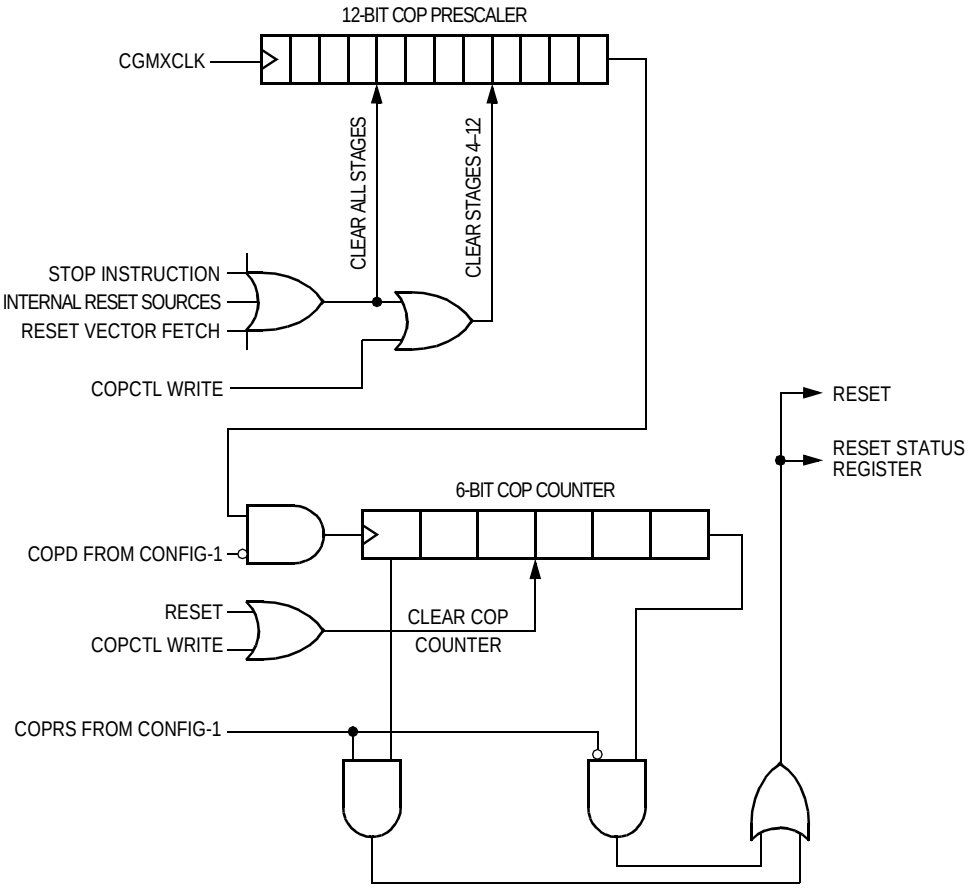


Figure 11-1. COP Block Diagram

The COP counter is a free-running 6-bit counter preceded by a 12-bit prescaler. If not cleared by software, the COP counter overflows and generates an asynchronous reset after $2^{13} - 2^4$ or $2^{18} - 2^4$ CGMXCLK cycles, depending on the state of the COP rate select bit, COPRS, in the CONFIG-1. When COPRS = 1, a 4.9152-MHz crystal gives a COP timeout period of 53.3 ms. Writing any value to location $FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 4–12 of the SIM counter.

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

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

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

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

11.4 I/O Signals

The following paragraphs describe the signals shown in Figure 11-1.

11.4.1 CGMXCLK

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

11.4.2 STOP Instruction

The STOP instruction signal clears the COP prescaler.

11.4.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see **11.5 COP Control Register**) clears the COP counter and clears stages 12 through 4 of the COP prescaler. Reading the COP control register returns the reset vector.

11.4.4 Power-On Reset

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

11.4.5 Internal Reset

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

11.4.6 Reset Vector Fetch

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

11.4.7 COPD

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register. See **Section 8. Configuration Registers (CONFIG1 & CONFIG2)**.

11.4.8 COPRS

The COPRS signal reflects the state of the COP rate select bit (COPRS) in the configuration register. See **Section 8. Configuration Registers (CONFIG1 & CONFIG2)**.

11.5 COP Control Register

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

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 11-2. COP Control Register (COPCTL)

11.6 Interrupts

The COP does not generate CPU interrupt requests.

11.7 Monitor Mode

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

11.8 Low-Power Modes

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

11.8.1 Wait Mode

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

11.8.2 Stop Mode

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

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

11.9 COP Module During Break Interrupts

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

Section 12. Low-Voltage Inhibit (LVI) Module

12.1 Contents

12.2 Introduction

This section describes the low-voltage inhibit (LVI) module, which monitors the voltage on the V_{DD} pin and can force a reset when the V_{DD} voltage falls to the LVI trip voltage.

12.3 Features

Features of the LVI module include:

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

12.4 Functional Description

Figure 12-1 shows the structure of the LVI module. The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. The LVI power bit, LVIPWRD, enables the LVI to monitor V_{DD} voltage. The LVI reset bit, LVIRSTD, enables the LVI module to generate a reset when V_{DD} falls below a voltage, LVI_{TRIPF}. LVISTOP, enables the LVI module during stop mode. This will ensure when the STOP instruction is implemented, the LVI will continue to monitor the voltage level on V_{DD}. LVIPWRD, LVISTOP, and LVIRSTD are in the configuration register (CONFIG-1). See **Section 8. Configuration Registers (CONFIG1 & CONFIG2)**.

Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises above a voltage, LVI_{TRIPR}. V_{DD} must be above LVI_{TRIPR} for only one CPU cycle to bring the MCU out of reset (see **12.4.2 Forced Reset Operation**). The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

An LVI reset also drives the $\overline{RST}$ pin low to provide low-voltage protection to external peripheral devices.

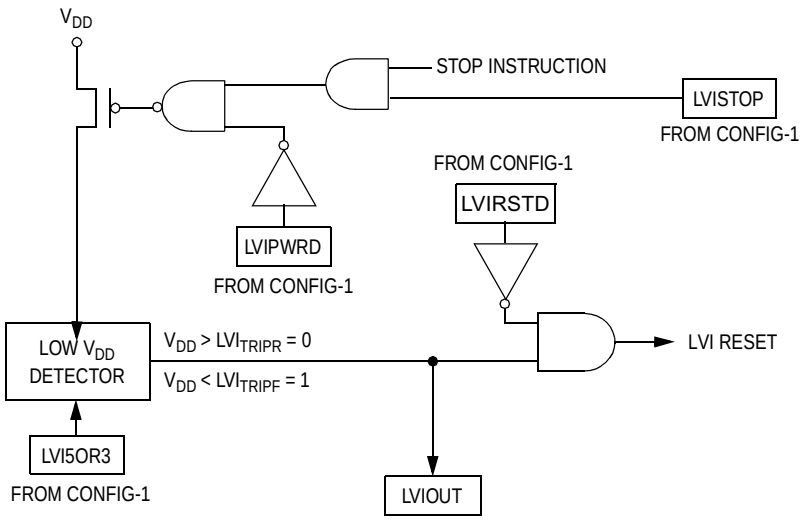


Figure 12-1. LVI Module Block Diagram

12.4.1 Polled LVI Operation

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

12.4.2 Forced Reset Operation

In applications that require V_{DD} to remain above the LVI_{TRIPF} level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls to the LVI_{TRIPF} level. In the configuration register, the LVIPWRD and LVIRSTD bits must be at logic 1 to enable the LVI module and to enable LVI resets.

12.4.3 False Reset Protection

False reset protection is provided by the hysteresis in the LVI trip circuit. Please refer to Table 12-1. Please refer to the Electrical Specifications for hysteresis value (VHYS) and rising and falling LVI trip values.

12.4.4 LVI Status Register

The LVI status register flags V_{DD} voltages below the LVI_{TRIPF} level.

Address: $FE0C

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

= Unimplemented

Figure 12-2. LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

This read-only flag becomes set when the V_{DD} voltage falls below the LVI_{TRIPF} voltage for 32 to 40 CGMXCLK cycles. (See **Table 12-1**.) Reset clears the LVIOUT bit.

Table 12-1. LVIOUT Bit Indication

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

12.5 LVI Interrupts

The LVI module does not generate interrupt requests.

12.6 Wait Mode

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

With the LVIPWRD bit in the configuration register programmed to logic 1, the LVI module is active after a WAIT instruction.

With the LVIRSTD bit in the configuration register programmed to logic 1, the LVI module can generate a reset and bring the MCU out of wait mode.

12.7 Stop Mode

The STOP instruction puts the MCU in low power-consumption mode.

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

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

Low-Voltage Inhibit (LVI) Module

Section 13. External Interrupt (IRQ)

13.1 Contents

13.2 Introduction

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

13.3 Features

Features include:

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

13.4 Functional Description

A logic 0 applied to the external interrupt pin can latch a central processor unit (CPU) interrupt request. Figure 13-1 shows the structure of the IRQ module.

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

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

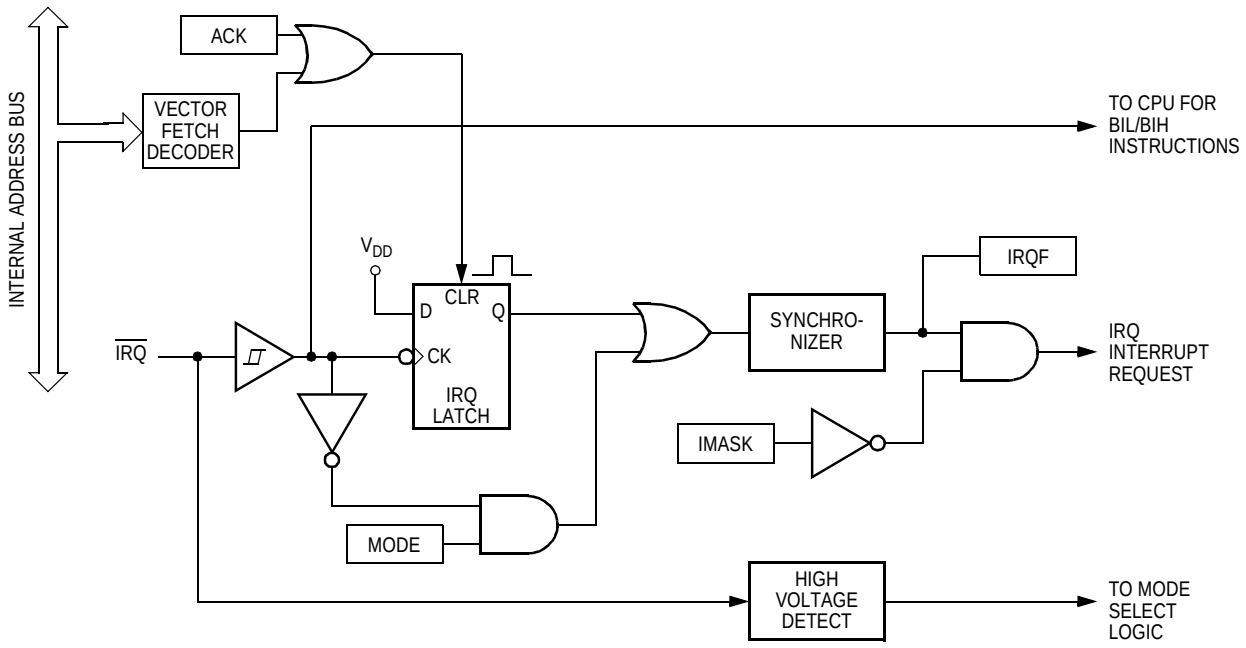


Figure 13-1. IRQ Block Diagram

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

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

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

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

The vector fetch or software clear may occur before or after the interrupt pin returns to logic 1. As long as the pin is low, the interrupt request remains pending. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

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

NOTE: *The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests.*
See Figure 13-2.

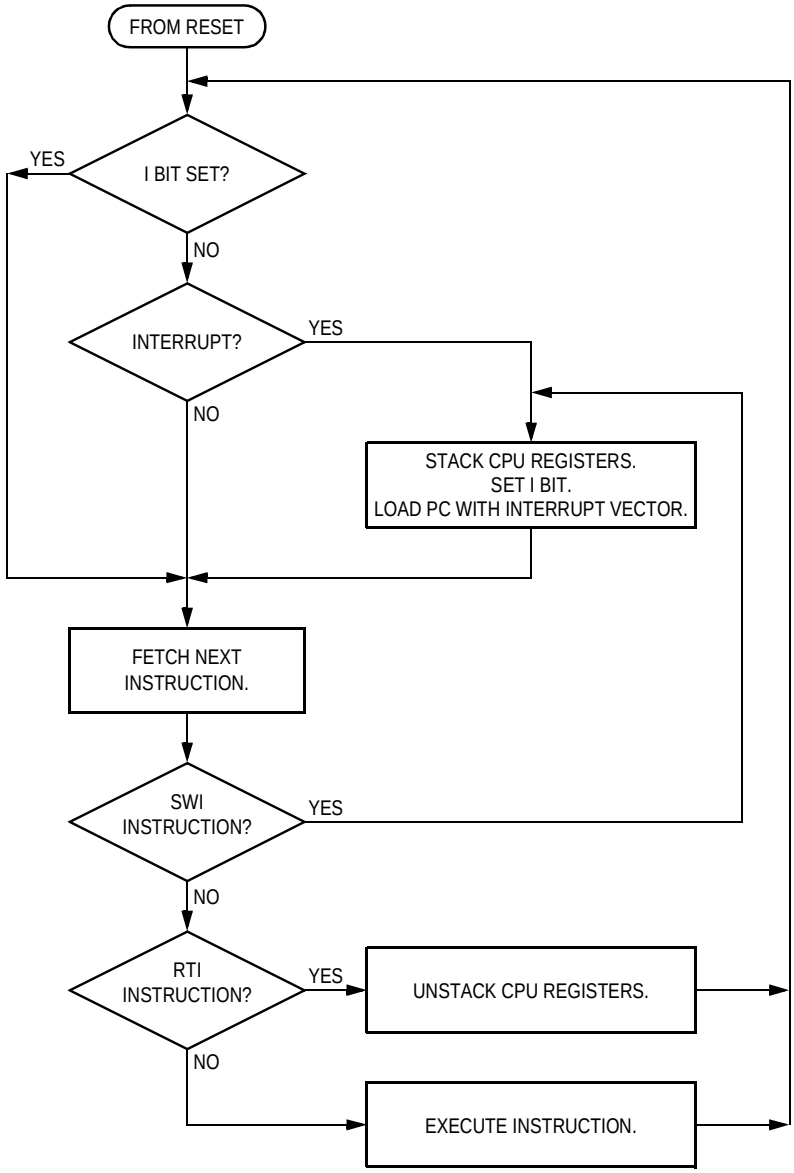


Figure 13-2. IRQ Interrupt Flowchart

13.5 $\overline{\text{IRQ}}$ Pin

A logic 0 on the $\overline{\text{IRQ}}$ pin can latch an interrupt request into the IRQ latch. A vector fetch, software clear, or reset clears the IRQ latch.

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

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the latch. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACK bit in the interrupt status and control register (ISCR). The ACK bit is useful in applications that poll the $\overline{\text{IRQ}}$ pin and require software to clear the IRQ latch. Writing to the ACK bit can also prevent spurious interrupts due to noise. Setting ACK does not affect subsequent transitions on the $\overline{\text{IRQ}}$ pin. A falling edge on $\overline{\text{IRQ}}$ that occurs after writing to the ACK bit latches another interrupt request. If the IRQ mask bit, IMASK, is clear, the CPU loads the program counter with the vector address at locations $FFFA and $FFFB.
- Return of the $\overline{\text{IRQ}}$ pin to logic 1 — As long as the $\overline{\text{IRQ}}$ pin is at logic 0, the IRQ latch remains set.

The vector fetch or software clear and the return of the $\overline{\text{IRQ}}$ pin to logic 1 can occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ}}$ pin is at logic 0. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

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

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

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

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

13.6 IRQ Module During Break Interrupts

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

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

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

13.7 IRQ Status and Control Register

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

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

Address: $001D

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

R = Reserved

Figure 13-3. IRQ Status and Control Register (ISCR)

IRQF — IRQ Flag Bit

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

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

ACK — IRQ Interrupt Request Acknowledge Bit

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

IMASK — $\overline{\text{IRQ}}$ Interrupt Mask Bit

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

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

MODE — IRQ Edge/Level Select Bit

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

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

External Interrupt (IRQ)

Section 14. Enhanced Serial Communications Interface (ESCI) Module

14.1 Contents

14.2 Introduction

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

14.3 Features

Features include:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- Programmable baud rates
- Programmable 8-bit or 9-bit character length
- Separately enabled transmitter and receiver
- Separate receiver and transmitter central processor unit (CPU) interrupt requests
- Programmable transmitter output polarity
- Two receiver wakeup methods:
  - Idle line wakeup
  - address mark wakeup
- Interrupt-driven operation with eight interrupt flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

14.4 Pin Name Conventions

The generic names of the ESCI input/output (I/O) pins are:

- RxD (receive data)
- TxD (transmit data)

ESCI I/O lines are implemented by sharing parallel I/O port pins. The full name of an ESCI input or output reflects the name of the shared port pin. Table 14-1 shows the full names and the generic names of the ESCI I/O pins. The generic pin names appear in the text of this section.

Table 14-1. Pin Name Conventions

Generic Pin Names	RxD	TxD
Full Pin Names	PTE1/RxD	PTE0/TxD

14.5 Functional Description

Figure 14-1 shows the structure of the ESCI module. The ESCI allows full-duplex, asynchronous, NRZ serial communication between the MCU and remote devices, including other MCUs. The transmitter and receiver of the ESCI operate independently, although they use the same baud rate generator. During normal operation, the CPU monitors the status of the ESCI, writes the data to be transmitted, and processes received data.

The baud rate clock source for the ESCI can be selected via the configuration bit, ESCIBDSRC, of the CONFIG2 register ($001E).

For reference, a summary of the ESCI module input/output registers is provided in Figure 14-2.

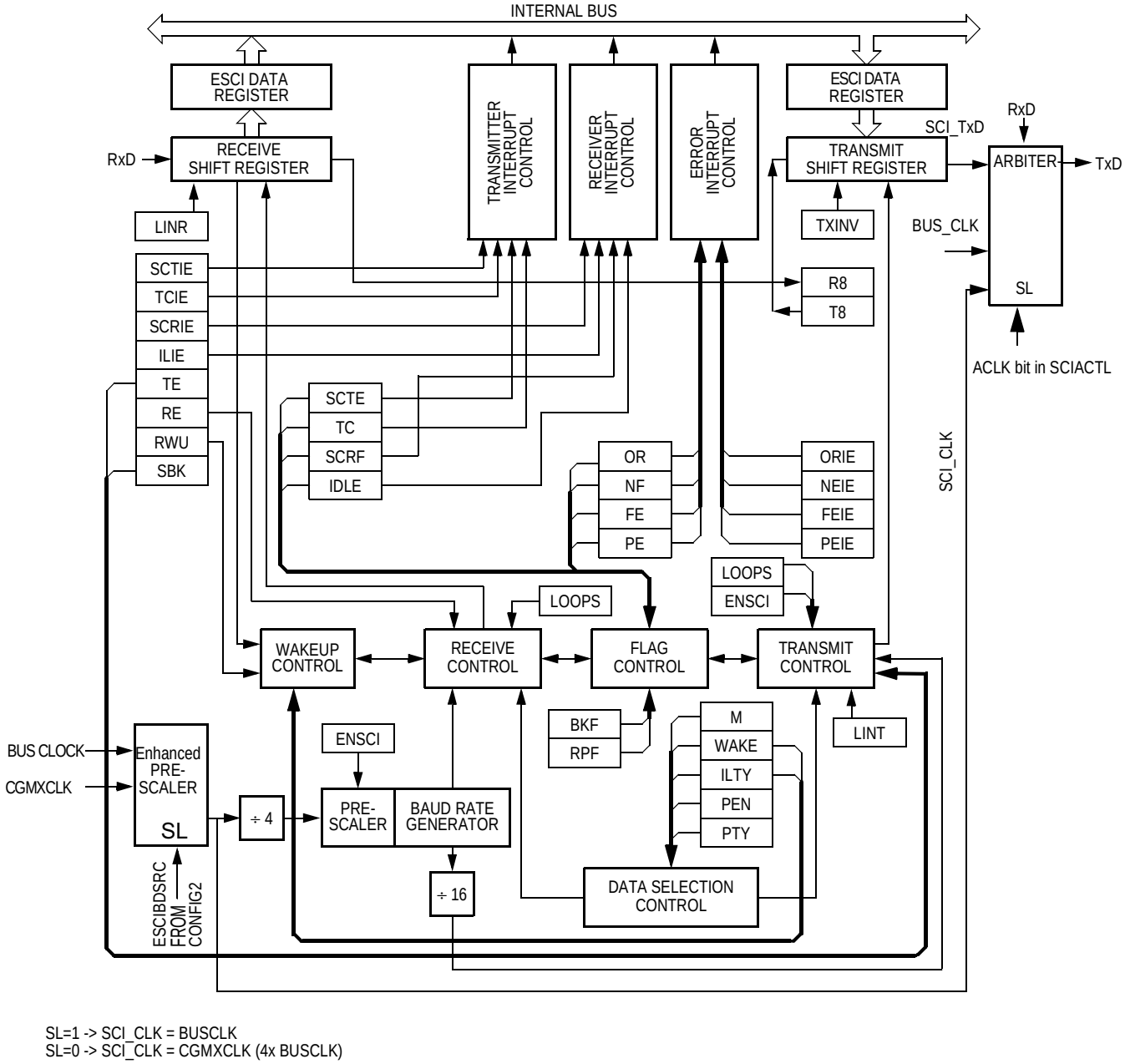


Figure 14-1. ESCI Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0010	ESCI Control Register 1 (SCC1) See 220.	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0011	ESCI Control Register 2 (SCC2) See 223.	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0012	ESCI Control Register 3 (SCC3) See 225.	Read:	R8	T8	R	R	ORIE	NEIE	FEIE	PEIE
		Write:								
		Reset:	U	0	0	0	0	0	0	0
$0013	ESCI Status Register 1 (SCS1) See 227.	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$0014	ESCI Status Register 2 (SCS2) See 231.	Read:	0	0	0	0	0	0	BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0015	ESCI Data Register (SCDR) See 232.	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0016	ESCI Baud Rate Register (SCBR) See 232.	Read:	R	LINR	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0017	ESCI Prescaler Register (SCPSC) See 234.	Read:	PDS2	PDS1	PDS0	PSSB4	PSSB3	PSSB2	PSSB1	PSSB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	ESCI Arbiter Control Register (SCIACTL) See 239.	Read:	AM1	ALOST	AM0	ACLK	AFIN	ARUN	AROVFL	ARD8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0019	ESCI Arbiter Data Register (SCIADAT) See 241.	Read:	ARD7	ARD6	ARD5	ARD4	ARD3	ARD2	ARD1	ARD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Figure 14-2. ESCI I/O Register Summary

14.5.1 Data Format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in Figure 14-3.

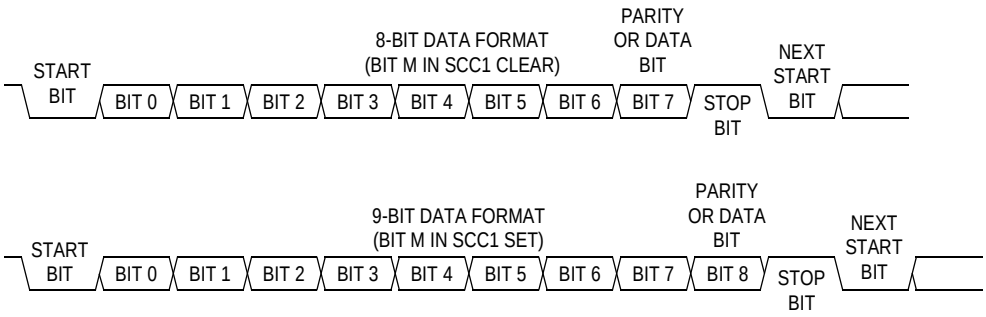


Figure 14-3. SCI Data Formats

14.5.2 Transmitter

Figure 14-4 shows the structure of the SCI transmitter and the registers are summarized in Figure 14-2. The baud rate clock source for the ESCI can be selected via the configuration bit, ESCIBDSRC.

14.5.2.1 Character Length

The transmitter can accommodate either 8-bit or 9-bit data. The state of the M bit in ESCI control register 1 (SCC1) determines character length. When transmitting 9-bit data, bit T8 in ESCI control register 3 (SCC3) is the ninth bit (bit 8).

14.5.2.2 Character Transmission

During an ESCI transmission, the transmit shift register shifts a character out to the TxD pin. The ESCI data register (SCDR) is the write-only buffer between the internal data bus and the transmit shift register.

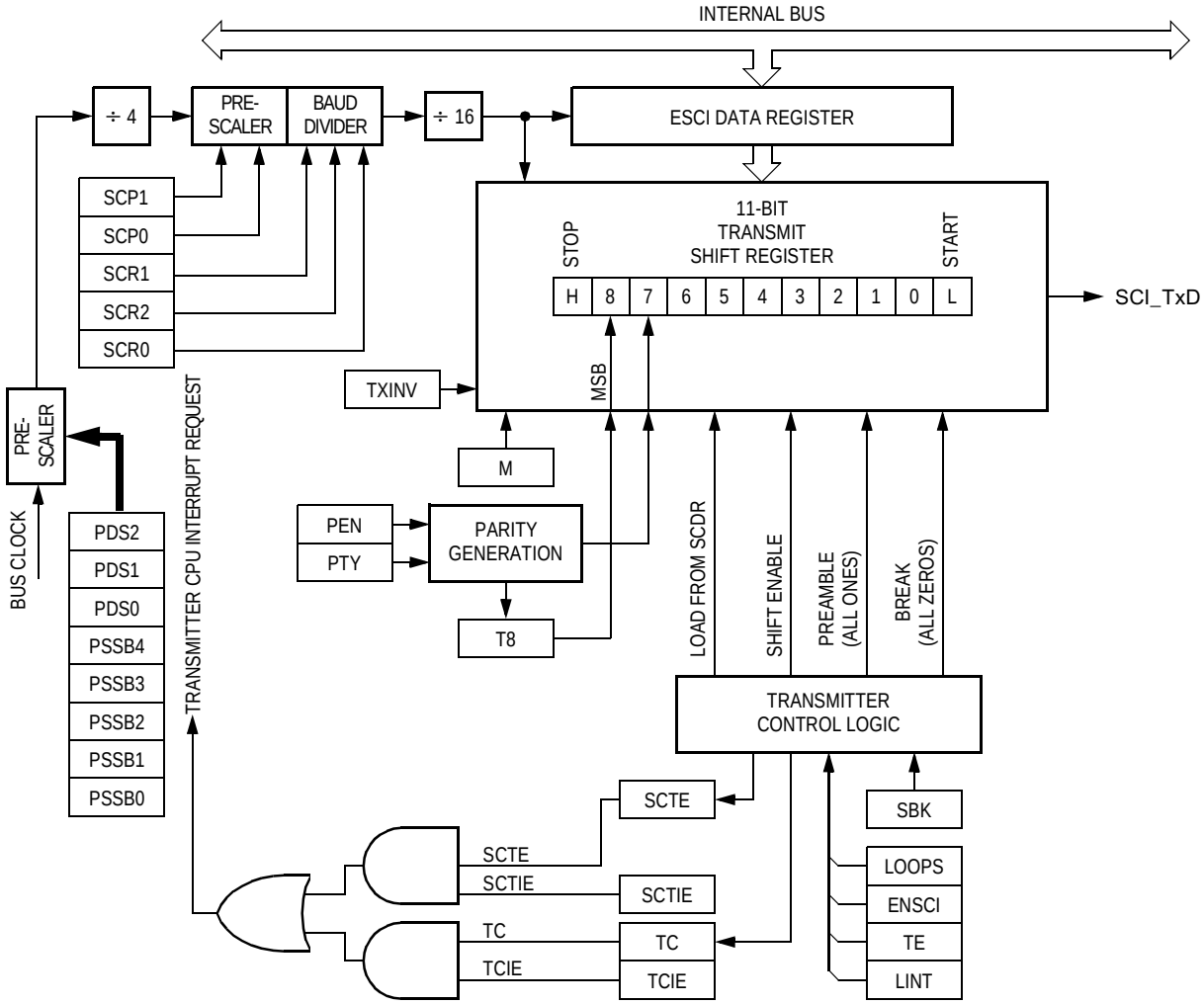


Figure 14-4. ESCI Transmitter

To initiate an ESCI transmission:

1. Enable the ESCI by writing a logic 1 to the enable ESCI bit (ENSCI) in ESCI control register 1 (SCC1).
2. Enable the transmitter by writing a logic 1 to the transmitter enable bit (TE) in ESCI control register 2 (SCC2).
3. Clear the ESCI transmitter empty bit (SCTE) by first reading ESCI status register 1 (SCS1) and then writing to the SCDR. For 9-bit data, also write the T8 bit in SCC3.
4. Repeat step 3 for each subsequent transmission.

At the start of a transmission, transmitter control logic automatically loads the transmit shift register with a preamble of logic 1s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit (LSB) position of the transmit shift register. A logic 1 stop bit goes into the most significant bit (MSB) position.

The ESCI transmitter empty bit, SCTE, in SCS1 becomes set when the SCDR transfers a byte to the transmit shift register. The SCTE bit indicates that the SCDR can accept new data from the internal data bus. If the ESCI transmit interrupt enable bit, SCTIE, in SCC2 is also set, the SCTE bit generates a transmitter CPU interrupt request.

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

14.5.2.3 Break Characters

Writing a logic 1 to the send break bit, SBK, in SCC2 loads the transmit shift register with a break character. For TXINV = 0 (output not inverted), a transmitted break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in SCC1 and the LINR bits in SCBR. As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next character.

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

Receiving a break character has these effects on ESCI registers:

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

14.5.2.4 Idle Characters

For TXINV = 0 (output not inverted), a transmitted idle character contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in SCC1. The preamble is a synchronizing idle character that begins every transmission.

If the TE bit is cleared during a transmission, the TxD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the character currently being transmitted.

NOTE: *When a break sequence is followed immediately by an idle character, this SCI design exhibits a condition in which the break character length is reduced by one half bit time. In this instance, the break sequence will consist of a valid start bit, eight or nine data bits (as defined by the M bit in SCC1) of logic 0 and one half data bit length of logic 0 in the stop bit position followed immediately by the idle character. To ensure a break character of the proper length is transmitted, always queue up a byte of data to be transmitted while the final break sequence is in progress.*

NOTE: *When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current character shifts out to the TxD pin. Setting TE after the stop bit appears on TxD causes data previously written to the SCDR to be lost. A good time to toggle the TE bit for a queued idle character is when the SCTE bit becomes set and just before writing the next byte to the SCDR.*

14.5.2.5 Inversion of Transmitted Output

The transmit inversion bit (TXINV) in ESCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values including idle, break, start, and stop bits, are inverted when TXINV is at logic 1. See **14.9.1 ESCI Control Register 1**.

14.5.2.6 Transmitter Interrupts

These conditions can generate CPU interrupt requests from the ESCI transmitter:

- ESCI transmitter empty (SCTE) — The SCTE bit in SCS1 indicates that the SCDR has transferred a character to the transmit shift register. SCTE can generate a transmitter CPU interrupt request. Setting the ESCI transmit interrupt enable bit, SCTIE, in SCC2 enables the SCTE bit to generate transmitter CPU interrupt requests.
- Transmission complete (TC) — The TC bit in SCS1 indicates that the transmit shift register and the SCDR are empty and that no break or idle character has been generated. The transmission complete interrupt enable bit, TCIE, in SCC2 enables the TC bit to generate transmitter CPU interrupt requests.

14.5.3 Receiver

Figure 14-5 shows the structure of the ESCI receiver. The receiver I/O registers are summarized in Figure 14-2.

14.5.3.1 Character Length

The receiver can accommodate either 8-bit or 9-bit data. The state of the M bit in ESCI control register 1 (SCC1) determines character length. When receiving 9-bit data, bit R8 in ESCI control register 3 (SCC3) is the ninth bit (bit 8). When receiving 8-bit data, bit R8 is a copy of the eighth bit (bit 7).

14.5.3.2 Character Reception

During an ESCI reception, the receive shift register shifts characters in from the RxD pin. The ESCI data register (SCDR) is the read-only buffer between the internal data bus and the receive shift register.

After a complete character shifts into the receive shift register, the data portion of the character transfers to the SCDR. The ESCI receiver full bit, SCRF, in ESCI status register 1 (SCS1) becomes set, indicating that the

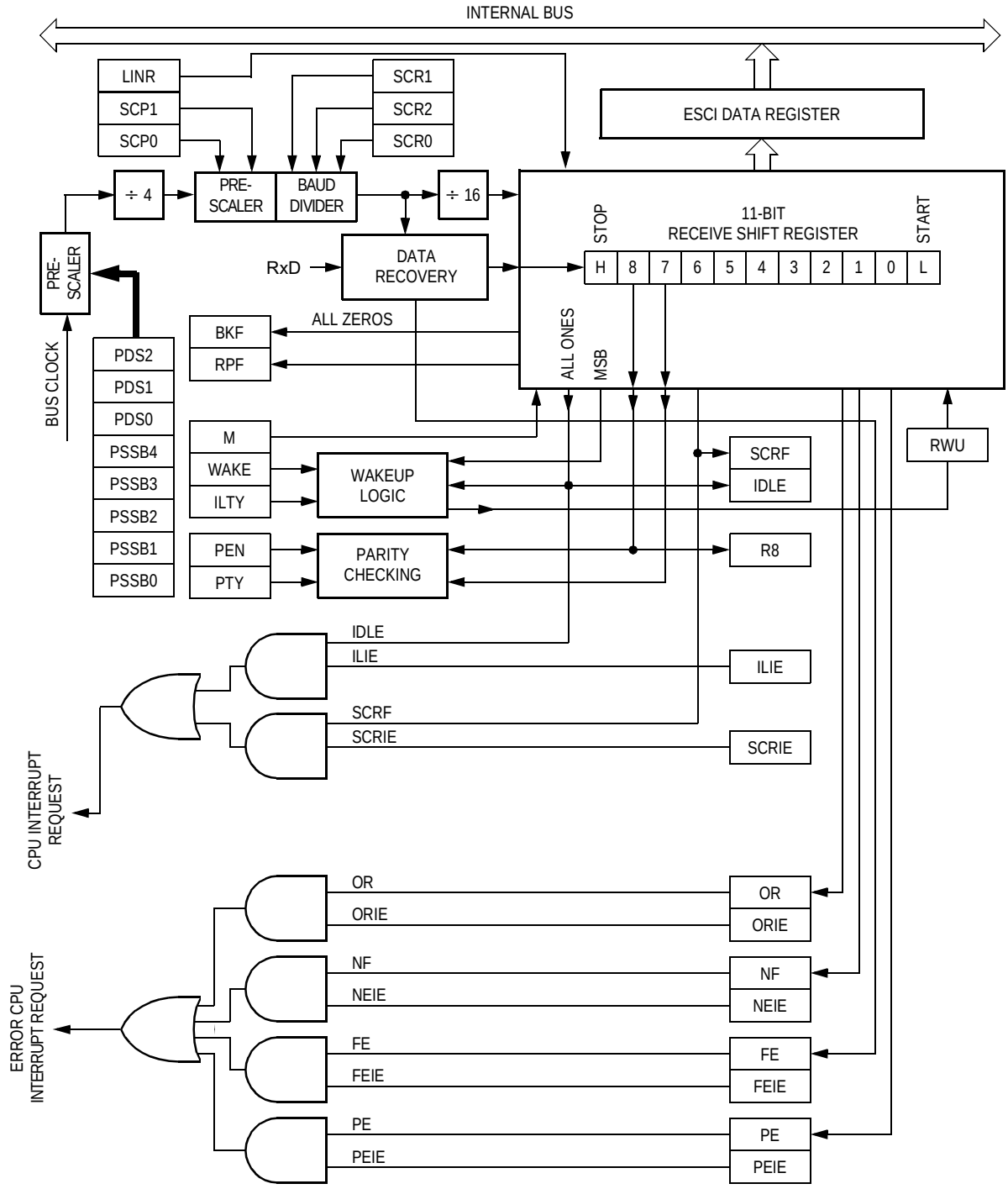


Figure 14-5. ESCI Receiver Block Diagram

received byte can be read. If the ESCI receive interrupt enable bit, SCRIE, in SCC2 is also set, the SCRF bit generates a receiver CPU interrupt request.

14.5.3.3 Data Sampling

The receiver samples the RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at these times (see Figure 14-6):

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0)

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

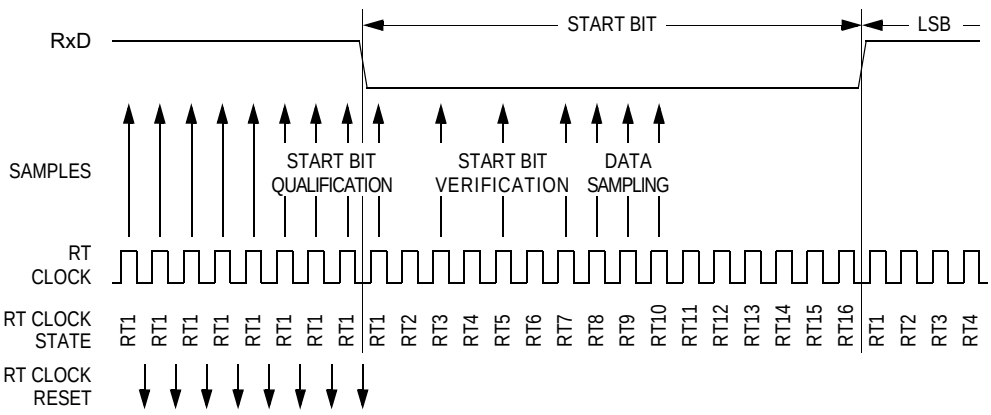


Figure 14-6. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 14-2 summarizes the results of the start bit verification samples.

Table 14-2. Start Bit Verification

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
000	Yes	0
001	Yes	1
010	Yes	1
011	No	0
100	Yes	1
101	No	0
110	No	0
111	No	0

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 14-3 summarizes the results of the data bit samples.

Table 14-3. Data Bit Recovery

RT8, RT9, and RT10 Samples	Data Bit Determination	Noise Flag
000	0	0
001	0	1
010	0	1
011	1	1
100	0	1
101	1	1
110	1	1
111	1	0

NOTE: *The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit.*

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 14-4 summarizes the results of the stop bit samples.

Table 14-4. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

14.5.3.4 Framing Errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming character, it sets the framing error bit, FE, in SCS1. A break character also sets the FE bit because a break character has no stop bit. The FE bit is set at the same time that the SCRF bit is set.

14.5.3.5 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples to fall outside the actual stop bit. Then a noise error occurs. If more than one of the samples is outside the stop bit, a framing error occurs. In most applications, the baud rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming character, it resynchronizes the RT clock on any valid falling edge within the character. Resynchronization within characters corrects misalignments between transmitter bit times and receiver bit times.

Slow Data Tolerance

Figure 14-7 shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

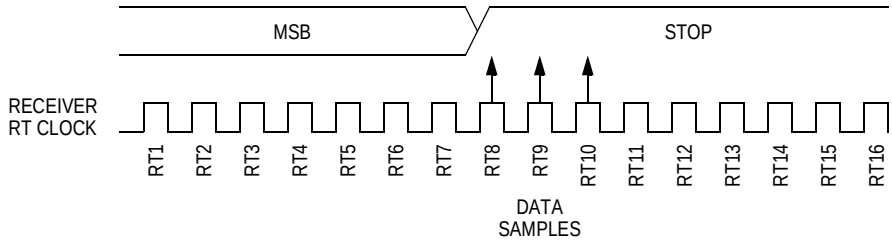


Figure 14-7. Slow Data

For an 8-bit character, data sampling of the stop bit takes the receiver 9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.

With the misaligned character shown in Figure 14-7, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit times × 16 RT cycles + 3 RT cycles = 147 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit character with no errors is:

$$\left|\frac{154 - 147}{154}\right| \times 100 = 4.54\%$$

For a 9-bit character, data sampling of the stop bit takes the receiver 10 bit times × 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in Figure 14-7, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is:

$$\left|\frac{170 - 163}{170}\right| \times 100 = 4.12\%$$

Fast Data Tolerance

Figure 14-8 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

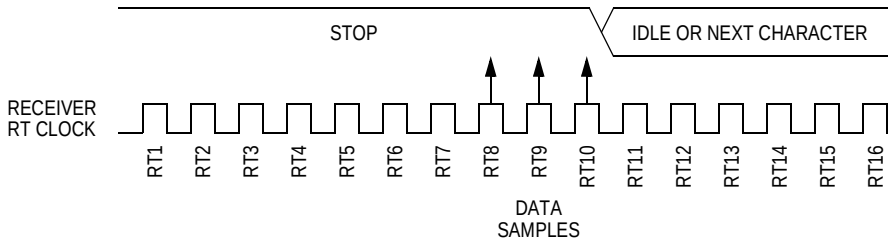


Figure 14-8. Fast Data

For an 8-bit character, data sampling of the stop bit takes the receiver 9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.

With the misaligned character shown in Figure 14-8, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles = 160 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is

$$\left|\frac{154 - 160}{154}\right| \times 100 = 3.90\%.$$

For a 9-bit character, data sampling of the stop bit takes the receiver 10 bit times × 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in Figure 14-8, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times × 16 RT cycles = 176 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is:

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

14.5.3.6 Receiver Wakeup

So that the MCU can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCC2 puts the receiver into a standby state during which receiver interrupts are disabled.

Depending on the state of the WAKE bit in SCC1, either of two conditions on the RxD pin can bring the receiver out of the standby state:

1. Address mark — An address mark is a logic 1 in the MSB position of a received character. When the WAKE bit is set, an address mark wakes the receiver from the standby state by clearing the RWU bit. The address mark also sets the ESCI receiver full bit, SCRF. Software can then compare the character containing the address mark to the user-defined address of the receiver. If they are the same, the receiver remains awake and processes the characters that follow. If they are not the same, software can set the RWU bit and put the receiver back into the standby state.
2. Idle input line condition — When the WAKE bit is clear, an idle character on the RxD pin wakes the receiver from the standby state by clearing the RWU bit. The idle character that wakes the receiver does not set the receiver idle bit, IDLE, or the ESCI receiver full bit, SCRF. The idle line type bit, ILTY, determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit.

NOTE: *With the WAKE bit clear, setting the RWU bit after the RxD pin has been idle will cause the receiver to wake up.*

14.5.3.7 Receiver Interrupts

These sources can generate CPU interrupt requests from the ESCI receiver:

- ESCI receiver full (SCRF) — The SCRF bit in SCS1 indicates that the receive shift register has transferred a character to the SCDR. SCRF can generate a receiver CPU interrupt request. Setting the ESCI receive interrupt enable bit, SCRIE, in SCC2 enables the SCRF bit to generate receiver CPU interrupts.
- Idle input (IDLE) — The IDLE bit in SCS1 indicates that 10 or 11 consecutive logic 1s shifted in from the RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables the IDLE bit to generate CPU interrupt requests.

14.5.3.8 Error Interrupts

These receiver error flags in SCS1 can generate CPU interrupt requests:

- Receiver overrun (OR) — The OR bit indicates that the receive shift register shifted in a new character before the previous character was read from the SCDR. The previous character remains in the SCDR, and the new character is lost. The overrun interrupt enable bit, ORIE, in SCC3 enables OR to generate ESCI error CPU interrupt requests.
- Noise flag (NF) — The NF bit is set when the ESCI detects noise on incoming data or break characters, including start, data, and stop bits. The noise error interrupt enable bit, NEIE, in SCC3 enables NF to generate ESCI error CPU interrupt requests.
- Framing error (FE) — The FE bit in SCS1 is set when a logic 0 occurs where the receiver expects a stop bit. The framing error interrupt enable bit, FEIE, in SCC3 enables FE to generate ESCI error CPU interrupt requests.
- Parity error (PE) — The PE bit in SCS1 is set when the ESCI detects a parity error in incoming data. The parity error interrupt enable bit, PEIE, in SCC3 enables PE to generate ESCI error CPU interrupt requests.

14.6 Low-Power Modes

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

14.6.1 Wait Mode

The ESCI module remains active in wait mode. Any enabled CPU interrupt request from the ESCI module can bring the MCU out of wait mode.

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

14.6.2 Stop Mode

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

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

14.7 ESCI During Break Module Interrupts

The BCFE bit in the break flag control register (SBFCR) enables software to clear status bits during the break state. See **Section 9. Break Module (BRK)**.

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

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

14.8 I/O Signals

Port E shares two of its pins with the ESCI module. The two ESCI I/O pins are:

- PTE0/TxD — transmit data
- PTE1/RxD — receive data

14.8.1 PTE0/TxD (Transmit Data)

The PTE0/TxD pin is the serial data output from the ESCI transmitter. The ESCI shares the PTE0/TxD pin with port E. When the ESCI is enabled, the PTE0/TxD pin is an output regardless of the state of the DDRE0 bit in data direction register E (DDRE).

14.8.2 PTE1/RxD (Receive Data)

The PTE1/RxD pin is the serial data input to the ESCI receiver. The ESCI shares the PTE1/RxD pin with port E. When the ESCI is enabled, the PTE1/RxD pin is an input regardless of the state of the DDRE1 bit in data direction register E (DDRE).

14.9 I/O Registers

These I/O registers control and monitor ESCI operation:

- ESCI control register 1, SCC1
- ESCI control register 2, SCC2
- ESCI control register 3, SCC3
- ESCI status register 1, SCS1
- ESCI status register 2, SCS2
- ESCI data register, SCDR
- ESCI baud rate register, SCBR
- ESCI prescaler register, SCPSC
- ESCI arbiter control register, SCIACTL
- ESCI arbiter data register, SCIADAT

14.9.1 ESCI Control Register 1

ESCI control register 1 (SCC1):

- Enables loop mode operation
- Enables the ESCI
- Controls output polarity
- Controls character length
- Controls ESCI wakeup method
- Controls idle character detection
- Enables parity function
- Controls parity type

Address: $0013

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
Reset:	0	0	0	0	0	0	0	0

Figure 14-9. ESCI Control Register 1 (SCC1)

LOOPS — Loop Mode Select Bit

This read/write bit enables loop mode operation. In loop mode the RxD pin is disconnected from the ESCI, and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use loop mode. Reset clears the LOOPS bit.

1 = Loop mode enabled
0 = Normal operation enabled

ENSCI — Enable ESCI Bit

This read/write bit enables the ESCI and the ESCI baud rate generator. Clearing ENSCI sets the SCTE and TC bits in ESCI status register 1 and disables transmitter interrupts. Reset clears the ENSCI bit.

1 = ESCI enabled
0 = ESCI disabled

TXINV — Transmit Inversion Bit

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

1 = Transmitter output inverted
0 = Transmitter output not inverted

NOTE: *Setting the TXINV bit inverts all transmitted values including idle, break, start, and stop bits.*

M — Mode (Character Length) Bit

This read/write bit determines whether ESCI characters are eight or nine bits long (See Table 14-5).The ninth bit can serve as a receiver wakeup signal or as a parity bit. Reset clears the M bit.

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

Table 14-5. Character Format Selection

Control Bits		Character Format				
M	**PEN:PTY**	**Start Bits**	**Data Bits**	**Parity**	**Stop Bits**	**Character Length**
0	0 X	1	8	None	1	10 bits
1	0 X	1	9	None	1	11 bits
0	1 0	1	7	Even	1	10 bits
0	1 1	1	7	Odd	1	10 bits
1	1 0	1	8	Even	1	11 bits
1	1 1	1	8	Odd	1	11 bits

WAKE — Wakeup Condition Bit

This read/write bit determines which condition wakes up the ESCI: a logic 1 (address mark) in the MSB position of a received character or an idle condition on the RxD pin. Reset clears the WAKE bit.

1 = Address mark wakeup
0 = Idle line wakeup

ILTY — Idle Line Type Bit

This read/write bit determines when the ESCI starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. Reset clears the ILTY bit.

1 = Idle character bit count begins after stop bit
0 = Idle character bit count begins after start bit

PEN — Parity Enable Bit

This read/write bit enables the ESCI parity function (see Table 14-5). When enabled, the parity function inserts a parity bit in the MSB position (see Table 14-3). Reset clears the PEN bit.

1 = Parity function enabled
0 = Parity function disabled

PTY — Parity Bit

This read/write bit determines whether the ESCI generates and checks for odd parity or even parity (see Table 14-5). Reset clears the PTY bit.

1 = Odd parity
0 = Even parity

NOTE: *Changing the PTY bit in the middle of a transmission or reception can generate a parity error.*

14.9.2 ESCI Control Register 2

ESCI control register 2 (SCC2):

- Enables these CPU interrupt requests:
  - SCTE bit to generate transmitter CPU interrupt requests
  - TC bit to generate transmitter CPU interrupt requests
  - SCRF bit to generate receiver CPU interrupt requests
  - IDLE bit to generate receiver CPU interrupt requests
- Enables the transmitter
- Enables the receiver
- Enables ESCI wakeup
- Transmits ESCI break characters

Address: $0014

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
Reset:	0	0	0	0	0	0	0	0

Figure 14-10. ESCI Control Register 2 (SCC2)

SCTIE — ESCI Transmit Interrupt Enable Bit

This read/write bit enables the SCTE bit to generate ESCI transmitter CPU interrupt requests. Setting the SCTIE bit in SCC2 enables the SCTE bit to generate CPU interrupt requests. Reset clears the SCTIE bit.

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

TCIE — Transmission Complete Interrupt Enable Bit

This read/write bit enables the TC bit to generate ESCI transmitter CPU interrupt requests. Reset clears the TCIE bit.

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

SCRIE — ESCI Receive Interrupt Enable Bit

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

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

ILIE — Idle Line Interrupt Enable Bit

This read/write bit enables the IDLE bit to generate ESCI receiver CPU interrupt requests. Reset clears the ILIE bit.

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

TE — Transmitter Enable Bit

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

1 = Transmitter enabled
0 = Transmitter disabled

NOTE: *Writing to the TE bit is not allowed when the enable ESCI bit (ENSCI) is clear. ENSCI is in ESCI control register 1.*

RE — Receiver Enable Bit

Setting this read/write bit enables the receiver. Clearing the RE bit disables the receiver but does not affect receiver interrupt flag bits. Reset clears the RE bit.

1 = Receiver enabled
0 = Receiver disabled

NOTE: *Writing to the RE bit is not allowed when the enable ESCI bit (ENSCI) is clear. ENSCI is in ESCI control register 1.*

RWU — Receiver Wakeup Bit

This read/write bit puts the receiver in a standby state during which receiver interrupts are disabled. The WAKE bit in SCC1 determines whether an idle input or an address mark brings the receiver out of the standby state and clears the RWU bit. Reset clears the RWU bit.

1 = Standby state
0 = Normal operation

SBK — Send Break Bit

Setting and then clearing this read/write bit transmits a break character followed by a logic 1. The logic 1 after the break character guarantees recognition of a valid start bit. If SBK remains set, the transmitter continuously transmits break characters with no logic 1s between them. Reset clears the SBK bit.

1 = Transmit break characters
0 = No break characters being transmitted

NOTE: *Do not toggle the SBK bit immediately after setting the SCTE bit. Toggling SBK before the preamble begins causes the ESCI to send a break character instead of a preamble.*

14.9.3 ESCI Control Register 3

ESCI control register 3 (SCC3):

- Stores the ninth ESCI data bit received and the ninth ESCI data bit to be transmitted.
- Enables these interrupts:
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error

Address: $0015

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R8	T8	R	R	ORIE	NEIE	FEIE	PEIE
Write:								
Reset:	U	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Figure 14-11. ESCI Control Register 3 (SCC3)

R8 — Received Bit 8

When the ESCI is receiving 9-bit characters, R8 is the read-only ninth bit (bit 8) of the received character. R8 is received at the same time that the SCDR receives the other 8 bits.

When the ESCI is receiving 8-bit characters, R8 is a copy of the eighth bit (bit 7). Reset has no effect on the R8 bit.

T8 — Transmitted Bit 8

When the ESCI is transmitting 9-bit characters, T8 is the read/write ninth bit (bit 8) of the transmitted character. T8 is loaded into the transmit shift register at the same time that the SCDR is loaded into the transmit shift register. Reset clears the T8 bit.

ORIE — Receiver Overrun Interrupt Enable Bit

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

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

NEIE — Receiver Noise Error Interrupt Enable Bit

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

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

FEIE — Receiver Framing Error Interrupt Enable Bit

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

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

PEIE — Receiver Parity Error Interrupt Enable Bit

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

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

14.9.4 ESCI Status Register 1

ESCI status register 1 (SCS1) contains flags to signal these conditions:

- Transfer of SCDR data to transmit shift register complete
- Transmission complete
- Transfer of receive shift register data to SCDR complete
- Receiver input idle
- Receiver overrun
- Noisy data
- Framing error
- Parity error

Address: $0016

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
Write:								
Reset:	1	1	0	0	0	0	0	0

= Unimplemented

Figure 14-12. ESCI Status Register 1 (SCS1)

SCTE — ESCI Transmitter Empty Bit

This clearable, read-only bit is set when the SCDR transfers a character to the transmit shift register. SCTE can generate an ESCI transmitter CPU interrupt request. When the SCTIE bit in SCC2 is set, SCTE generates an ESCI transmitter CPU interrupt request. In normal operation, clear the SCTE bit by reading SCS1 with SCTE set and then writing to SCDR. Reset sets the SCTE bit.

1 = SCDR data transferred to transmit shift register
0 = SCDR data not transferred to transmit shift register

TC — Transmission Complete Bit

This read-only bit is set when the SCTE bit is set, and no data, preamble, or break character is being transmitted. TC generates an ESCI transmitter CPU interrupt request if the TCIE bit in SCC2 is also set. TC is cleared automatically when data, preamble, or break is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queueing data, preamble, and break and the transmission actually starting. Reset sets the TC bit.

1 = No transmission in progress
0 = Transmission in progress

SCRF — ESCI Receiver Full Bit

This clearable, read-only bit is set when the data in the receive shift register transfers to the ESCI data register. SCRF can generate an ESCI receiver CPU interrupt request. When the SCRIE bit in SCC2 is set the SCRF generates a CPU interrupt request. In normal operation, clear the SCRF bit by reading SCS1 with SCRF set and then reading the SCDR. Reset clears SCRF.

1 = Received data available in SCDR
0 = Data not available in SCDR

IDLE — Receiver Idle Bit

This clearable, read-only bit is set when 10 or 11 consecutive logic 1s appear on the receiver input. IDLE generates an ESCI error CPU interrupt request if the ILIE bit in SCC2 is also set. Clear the IDLE bit by reading SCS1 with IDLE set and then reading the SCDR. After the receiver is enabled, it must receive a valid character that sets the SCRF bit before an idle condition can set the IDLE bit. Also, after the IDLE bit has been cleared, a valid character must again set the SCRF bit before an idle condition can set the IDLE bit. Reset clears the IDLE bit.

1 = Receiver input idle
0 = Receiver input active (or idle since the IDLE bit was cleared)

OR — Receiver Overrun Bit

This clearable, read-only bit is set when software fails to read the SCDR before the receive shift register receives the next character. The OR bit generates an ESCI error CPU interrupt request if the ORIE

bit in SCC3 is also set. The data in the shift register is lost, but the data already in the SCDR is not affected. Clear the OR bit by reading SCS1 with OR set and then reading the SCDR. Reset clears the OR bit.

1 = Receive shift register full and SCRF = 1

0 = No receiver overrun

Software latency may allow an overrun to occur between reads of SCS1 and SCDR in the flag-clearing sequence. Figure 14-13 shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

In applications that are subject to software latency or in which it is important to know which byte is lost due to an overrun, the flag-clearing routine can check the OR bit in a second read of SCS1 after reading the data register.

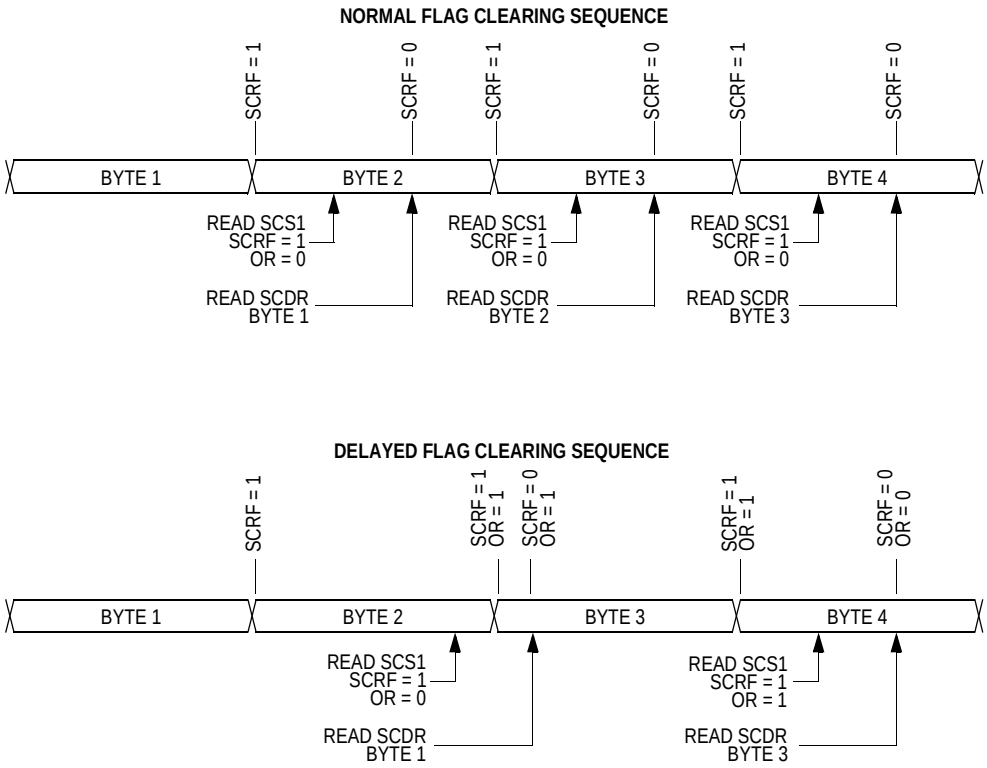


Figure 14-13. Flag Clearing Sequence

NF — Receiver Noise Flag Bit

This clearable, read-only bit is set when the ESCI detects noise on the RxD pin. NF generates an NF CPU interrupt request if the NEIE bit in SCC3 is also set. Clear the NF bit by reading SCS1 and then reading the SCDR. Reset clears the NF bit.

1 = Noise detected
0 = No noise detected

FE — Receiver Framing Error Bit

This clearable, read-only bit is set when a logic 0 is accepted as the stop bit. FE generates an ESCI error CPU interrupt request if the FEIE bit in SCC3 also is set. Clear the FE bit by reading SCS1 with FE set and then reading the SCDR. Reset clears the FE bit.

1 = Framing error detected
0 = No framing error detected

PE — Receiver Parity Error Bit

This clearable, read-only bit is set when the ESCI detects a parity error in incoming data. PE generates a PE CPU interrupt request if the PEIE bit in SCC3 is also set. Clear the PE bit by reading SCS1 with PE set and then reading the SCDR. Reset clears the PE bit.

1 = Parity error detected
0 = No parity error detected

14.9.5 ESCI Status Register 2

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

- Break character detected
- Incoming data

Address: $0017

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

= Unimplemented

Figure 14-14. ESCI Status Register 2 (SCS2)

BKF — Break Flag Bit

This clearable, read-only bit is set when the ESCI detects a break character on the RxD pin. In SCS1, the FE and SCRF bits are also set. In 9-bit character transmissions, the R8 bit in SCC3 is cleared. BKF does not generate a CPU interrupt request. Clear BKF by reading SCS2 with BKF set and then reading the SCDR. Once cleared, BKF can become set again only after logic 1s again appear on the RxD pin followed by another break character. Reset clears the BKF bit.

1 = Break character detected
0 = No break character detected

RPF — Reception in Progress Flag Bit

This read-only bit is set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RPF does not generate an interrupt request. RPF is reset after the receiver detects false start bits (usually from noise or a baud rate mismatch), or when the receiver detects an idle character. Polling RPF before disabling the ESCI module or entering stop mode can show whether a reception is in progress.

1 = Reception in progress
0 = No reception in progress

14.9.6 ESCI Data Register

The ESCI data register (SCDR) is the buffer between the internal data bus and the receive and transmit shift registers. Reset has no effect on data in the ESCI data register.

Address: $0018

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

Figure 14-15. ESCI Data Register (SCDR)

R7/T7:R0/T0 — Receive/Transmit Data Bits

Reading address $0018 accesses the read-only received data bits, R7:R0. Writing to address $0018 writes the data to be transmitted, T7:T0. Reset has no effect on the ESCI data register.

NOTE: *Do not use read-modify-write instructions on the ESCI data register.*

14.9.7 ESCI Baud Rate Register

The ESCI baud rate register (SCBR) together with the ESCI prescaler register selects the baud rate for both the receiver and the transmitter.

NOTE: *There are two prescalers available to adjust the baud rate. One in the ESCI baud rate register and one in the ESCI prescaler register.*

Address: $0019

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	LINR	SCP1	SCP0	R	SCR2	SCR1	SCR0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 14-16. ESCI Baud Rate Register (SCBR)

LINR — LIN Receiver Bits

This read/write bit selects the enhanced ESCI features for slave nodes in the local interconnect network (LIN) protocol as shown in Table 14-6. Reset clears LINR.

Table 14-6. ESCI LIN Control Bits

LINR	M	Functionality
0	X	Normal ESCI functionality
1	0	11-bit break detect enabled for LIN receiver
1	1	12-bit break detect enabled for LIN receiver

In LIN (version 1.2) systems, the master node transmits a break character which will appear as 11.05–14.95 dominant bits to the slave node. A data character of 0x00 sent from the master might appear as 7.65–10.35 dominant bit times. This is due to the oscillator tolerance requirement that the slave node must be within ±15% of the master node's oscillator. Since a slave node cannot know if it is running faster or slower than the master node (prior to synchronization), the LINR bit allows the slave node to differentiate between a 0x00 character of 10.35 bits and a break character of 11.05 bits. The break symbol length must be verified in software in any case, but the LINR bit serves as a filter, preventing false detections of break characters that are really 0x00 data characters.

SCP1 and SCP0 — ESCI Baud Rate Register Prescaler Bits

These read/write bits select the baud rate register prescaler divisor as shown in Table 14-7. Reset clears SCP1 and SCP0.

Table 14-7. ESCI Baud Rate Prescaling

SCP[1:0]	Baud Rate Register Prescaler Divisor (BPD)
0 0	1
0 1	3
1 0	4
1 1	13

SCR2–SCR0 — ESCI Baud Rate Select Bits

These read/write bits select the ESCI baud rate divisor as shown in Table 14-8. Reset clears SCR2–SCR0.

Table 14-8. ESCI Baud Rate Selection

SCR[2:1:0]	Baud Rate Divisor (BD)
0 0 0	1
0 0 1	2
0 1 0	4
0 1 1	8
1 0 0	16
1 0 1	32
1 1 0	64
1 1 1	128

14.9.8 ESCI Prescaler Register

The ESCI prescaler register (SCPSC) together with the ESCI baud rate register selects the baud rate for both the receiver and the transmitter.

NOTE: *There are two prescalers available to adjust the baud rate. One in the ESCI baud rate register and one in the ESCI prescaler register.*

Address: $0017

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PDS2	PDS1	PDS0	PSSB4	PSSB3	PSSB2	PSSB1	PSSB0
Reset:	0	0	0	0	0	0	0	0

Figure 14-17. ESCI Prescaler Register (SCPSC)

PDS2–PDS0 — Prescaler Divisor Select Bits

These read/write bits select the prescaler divisor as shown in Table 14-9. Reset clears PDS2–PDS0.

NOTE: *The setting of '000' will bypass this prescaler. It is not recommended to bypass the prescaler while ENSCI is set, because the switching is not glitch free.*

Table 14-9. ESCI Prescaler Division Ratio

PDS[2:1:0]	Prescaler Divisor (PD)
0 0 0	Bypass this prescaler
0 0 1	2
0 1 0	3
0 1 1	4
1 0 0	5
1 0 1	6
1 1 0	7
1 1 1	8

PSSB4–PSSB0 — Clock Insertion Select Bits

These read/write bits select the number of clocks inserted in each 32 output cycle frame to achieve more timing resolution on the **average** prescaler frequency as shown in Table 14-10. Reset clears PSSB4–PSSB0.

Table 14-10. ESCI Prescaler Divisor Fine Adjust

PSSB[4:3:2:1:0]	Prescaler Divisor Fine Adjust (PDFA)
0 0 0 0 0	0/32 = 0
0 0 0 0 1	1/32 = 0.03125
0 0 0 1 0	2/32 = 0.0625
0 0 0 1 1	3/32 = 0.09375
0 0 1 0 0	4/32 = 0.125
0 0 1 0 1	5/32 = 0.15625
0 0 1 1 0	6/32 = 0.1875
0 0 1 1 1	7/32 = 0.21875

Table 14-10. ESCI Prescaler Divisor Fine Adjust (Continued)

PSSB[4:3:2:1:0]	Prescaler Divisor Fine Adjust (PDFA)
0 1 0 0 0	8/32 = 0.25
0 1 0 0 1	9/32 = 0.28125
0 1 0 1 0	10/32 = 0.3125
0 1 0 1 1	11/32 = 0.34375
0 1 1 0 0	12/32 = 0.375
0 1 1 0 1	13/32 = 0.40625
0 1 1 1 0	14/32 = 0.4375
0 1 1 1 1	15/32 = 0.46875
1 0 0 0 0	16/32 = 0.5
1 0 0 0 1	17/32 = 0.53125
1 0 0 1 0	18/32 = 0.5625
1 0 0 1 1	19/32 = 0.59375
1 0 1 0 0	20/32 = 0.625
1 0 1 0 1	21/32 = 0.65625
1 0 1 1 0	22/32 = 0.6875
1 0 1 1 1	23/32 = 0.71875
1 1 0 0 0	24/32 = 0.75
1 1 0 0 1	25/32 = 0.78125
1 1 0 1 0	26/32 = 0.8125
1 1 0 1 1	27/32 = 0.84375
1 1 1 0 0	28/32 = 0.875
1 1 1 0 1	29/32 = 0.90625
1 1 1 1 0	30/32 = 0.9375
1 1 1 1 1	31/32 = 0.96875

Use the following formula to calculate the ESCI baud rate:

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

Frequency of the SCI clock source = f_{Bus} or CGMXCLK (selected by ESCIBDSRC in the CONFIG2 register)

BPD = Baud rate register prescaler divisor

BD = Baud rate divisor

PD = Prescaler divisor

PDFA = Prescaler divisor fine adjust

Table 14-11 shows the ESCI baud rates that can be generated with a 4.9152-MHz clock frequency.

Table 14-11. ESCI Baud Rate Selection Examples

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

14.10 ESCI Arbiter

The ESCI module comprises an arbiter module designed to support software for communication tasks as bus arbitration, baud rate recovery and break time detection. The arbiter module consists of an 9-bit counter with 1-bit overflow and control logic. The CPU can control operation mode via the ESCI arbiter control register (SCIACTL).

14.10.1 ESCI Arbiter Control Register

Address: $0018

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	AM1	ALOST	AM0	ACLK	AFIN	ARUN	AROVFL	ARD8
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-18. ESCI Arbiter Control Register (SCIACTL)

AM1 and AM0 — Arbiter Mode Select Bits

These read/write bits select the mode of the arbiter module as shown in Table 14-12. Reset clears AM1 and AM0.

Table 14-12. ESCI Arbiter Selectable Modes

AM[1:0]	ESCI Arbiter Mode
0 0	Idle / counter reset
0 1	Bit time measurement
1 0	Bus arbitration
1 1	Reserved / do not use

ALOST — Arbitration Lost Flag

This read-only bit indicates loss of arbitration. Clear ALOST by writing a logic 0 to AM1. Reset clears ALOST.

ACLK — Arbiter Counter Clock Select Bit

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

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

NOTE: *For ACLK=1, the Arbiter input clock is driven from the ESCI prescaler. The prescaler can be clocked by either the bus clock or CGMXCLK depending on the state of the ESCIBDSRC bit in CONFIG2.*

AFIN— Arbiter Bit Time Measurement Finish Flag

This read-only bit indicates bit time measurement has finished. Clear AFIN by writing any value to SCIACTL. Reset clears AFIN.

1 = Bit time measurement has finished
0 = Bit time measurement not yet finished

ARUN— Arbiter Counter Running Flag

This read-only bit indicates the arbiter counter is running. Reset clears ARUN.

1 = Arbiter counter running
0 = Arbiter counter stopped

AROVFL— Arbiter Counter Overflow Bit

This read-only bit indicates an arbiter counter overflow. Clear AROVFL by writing any value to SCIACTL. Writing logic 0s to AM1 and AM0 resets the counter keeps it in this idle state. Reset clears AROVFL.

1 = Arbiter counter overflow has occurred
0 = No arbiter counter overflow has occurred

ARD8— Arbiter Counter MSB

This read-only bit is the MSB of the 9-bit arbiter counter. Clear ARD8 by writing any value to SCIACTL. Reset clears ARD8.

14.10.2 ESCI Arbiter Data Register

Address: $0019

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ARD7	ARD6	ARD5	ARD4	ARD3	ARD2	ARD1	ARD0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-19. ESCI Arbiter Data Register (SCIADAT)

ARD7–ARD0 — Arbiter Least Significant Counter Bits

These read-only bits are the eight LSBs of the 9-bit arbiter counter. Clear ARD7–ARD0 by writing any value to SCIACTL. Writing logic 0s to AM1 and AM0 permanently resets the counter and keeps it in this idle state. Reset clears ARD7–ARD0.

14.10.3 Bit Time Measurement

Two bit time measurement modes, described here, are available according to the state of ACLK.

1. ACLK = 0 — The counter is clocked with one quarter of the bus clock. The counter is started when a falling edge on the RxD pin is detected. The counter will be stopped on the next falling edge. ARUN is set while the counter is running, AFIN is set on the second falling edge on RxD (for instance, the counter is stopped). This mode is used to recover the received baud rate. See Figure 14-20.
2. ACLK = 1 — The counter is clocked with one half of the ESCI input clock generated by the ESCI prescaler. The counter is started when a logic 0 is detected on RxD (see Figure 14-21). A logic 0 on RxD on enabling the bit time measurement with ACLK = 1 leads to immediate start of the counter (see Figure 14-22). The counter will be stopped on the next rising edge of RxD. This mode is used to measure the length of a received break.

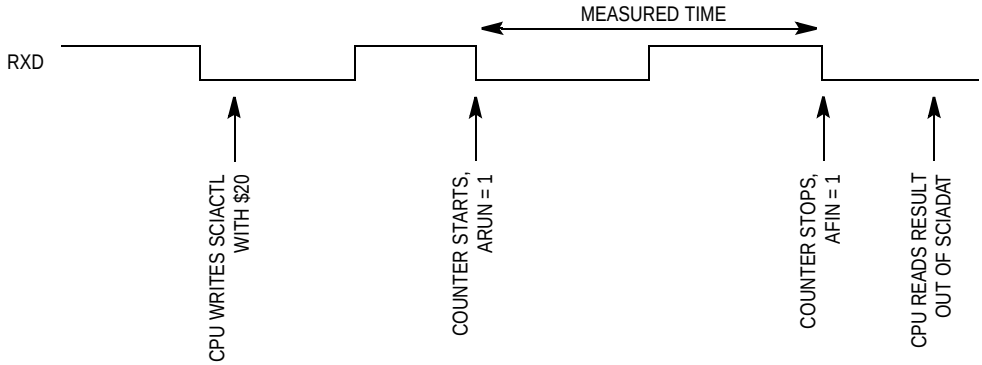


Figure 14-20. Bit Time Measurement with ACLK = 0

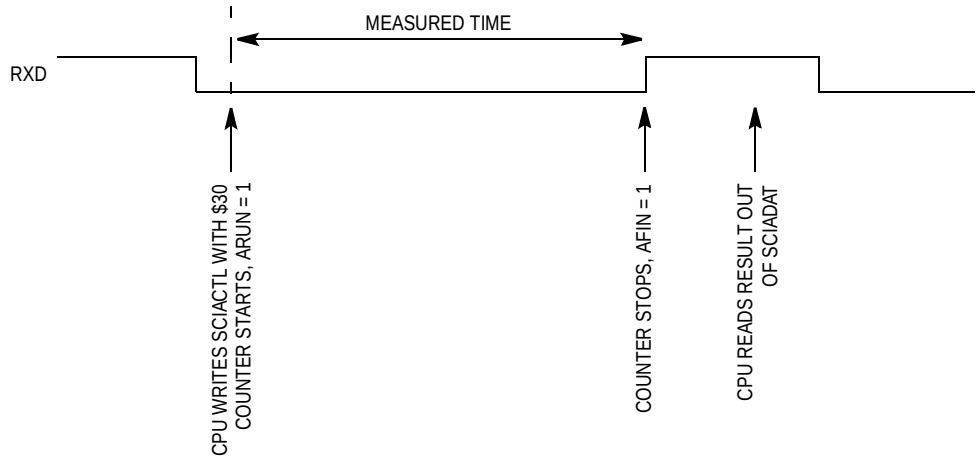


Figure 14-21. Bit Time Measurement with ACLK = 1, Scenario A

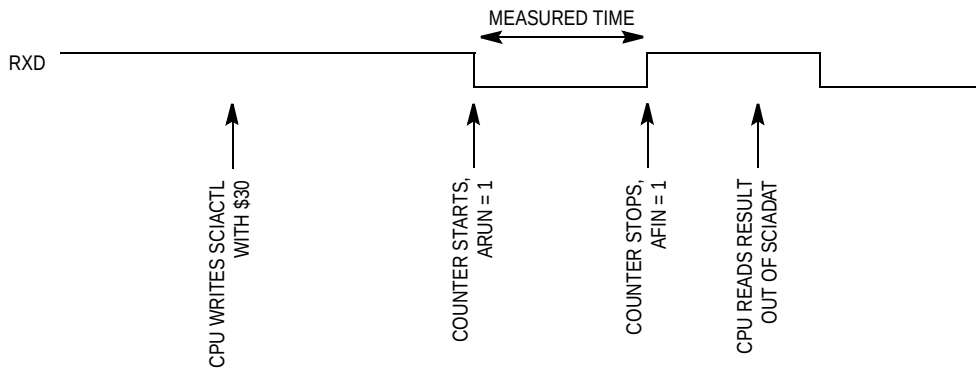


Figure 14-22. Bit Time Measurement with ACLK = 1, Scenario B

14.10.4 Arbitration Mode

If AM[1:0] is set to 10, the arbiter module operates in arbitration mode. On every rising edge of SCI_TxD (output of the ESCI module, internal chip signal), the counter is started. When the counter reaches $38 (ACLK = 0) or $08 (ACLK = 1), RxD is statically sensed. If in this case, RxD is sensed low (for example, another bus is driving the bus dominant) ALOST is set. As long as ALOST is set, the TxD pin is forced to 1, resulting in a seized transmission.

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

Section 15. Serial Peripheral Interface (SPI) Module

15.1 Contents

15.2 Introduction

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

15.3 Features

Features of the SPI module include:

- Full-Duplex Operation
- Master and Slave Modes
- Double-Buffered Operation with Separate Transmit and Receive Registers
- Four Master Mode Frequencies (Maximum = Bus Frequency ÷ 2)
- Maximum Slave Mode Frequency = Bus Frequency
- Serial Clock with Programmable Polarity and Phase
- Two Separately Enabled Interrupts with CPU Service:
  - SPRF (SPI Receiver Full)
  - SPTE (SPI Transmitter Empty)
- Mode Fault Error Flag with CPU Interrupt Capability
- Overflow Error Flag with CPU Interrupt Capability
- Programmable Wired-OR Mode
- I^2C (Inter-Integrated Circuit) Compatibility

15.4 Pin Name and Register Name Conventions

The generic names of the SPI input/output (I/O) pins are:

- $\overline{SS}$ (slave select)
- SPSCK (SPI serial clock)
- MOSI (master out slave in)
- MISO (master in slave out)

The SPI shares four I/O pins with a parallel I/O port. The full name of an SPI pin reflects the name of the shared port pin. Table 15-1 shows the full names of the SPI I/O pins. The generic pin names appear in the text that follows.

Table 15-1. Pin Name Conventions

SPI Generic Pin Name	MISO	MOSI	$\overline{SS}$	SPSCK
Full SPI Pin Name	PTC0/MISO	PTC1/MOSI	PTA6/$\overline{SS}$	PTA5/SPSCK

The generic names of the SPI I/O registers are:

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

Table 15-2 shows the names and the addresses of the SPI I/O registers.

Table 15-2. I/O Register Addresses

Register Name	Address
SPI Control Register (SPCR)	$000D
SPI Status and Control Register (SPSCR)	$000E
SPI Data Register (SPDR)	$000F

15.5 Functional Description

Table 15-3 summarizes the SPI I/O registers and Figure 15-1 shows the structure of the SPI module.

Table 15-3. SPI I/O Register Summary

Addr	Register Name	R/W	Bit 7	6	5	4	3	2	1	Bit 0
$000D	SPI Control Register (SPCR)	Read:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Write:								
		Reset:	0	0	1	0	1	0	0	0
$000E	SPI Status and Control Register (SPSCR)	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	1	0	0	0
$000F	SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by Reset							

R = Reserved = Unimplemented

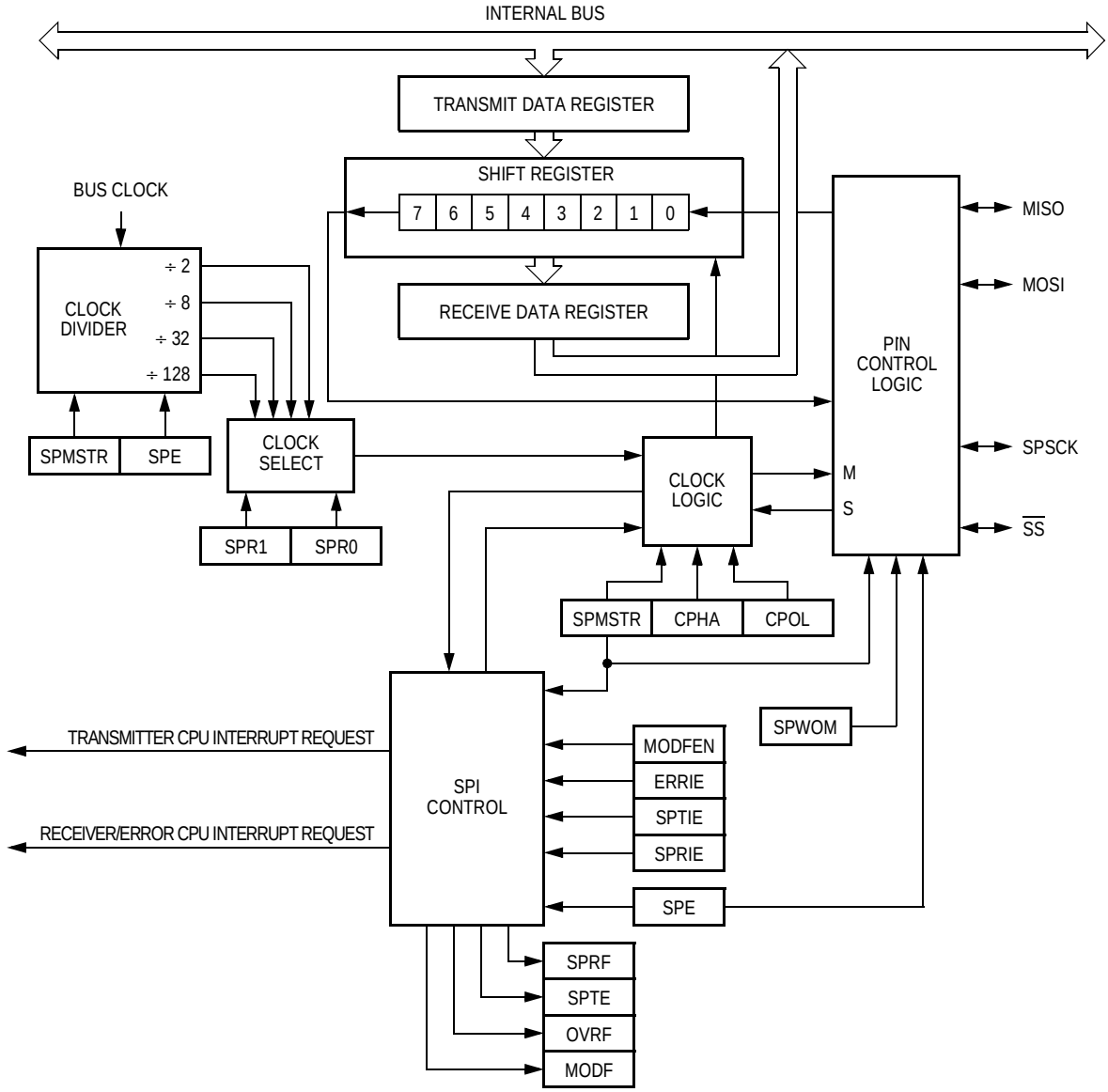


Figure 15-1. SPI Module Block Diagram

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

The following paragraphs describe the operation of the SPI module.

15.5.1 Master Mode

The SPI operates in master mode when the SPI master bit, SPMSTR (SPCR $0010), is set.

NOTE: *Configure the SPI modules as master and slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. See* ***SPI Control Register*** *on page 269.*

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the SPI data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE (SPSCR $0011). The byte begins shifting out on the MOSI pin under the control of the serial clock. (See Table 15-4).

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See **SPI Status and Control Register** on page 272). Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

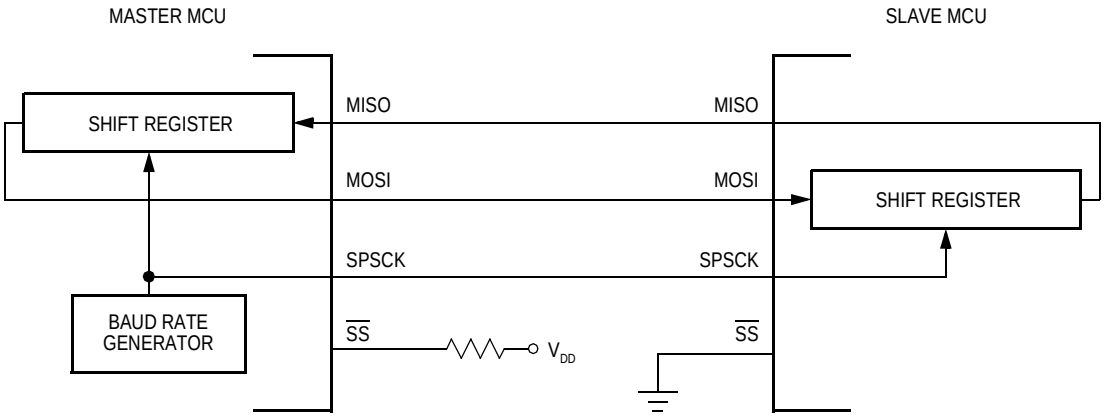


Figure 15-2. Full-Duplex Master-Slave Connections

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

15.5.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit (SPCR, $0010) is clear. In slave mode the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{SS}$ pin of the slave MCU must be at logic 0. $\overline{SS}$ must remain low until the transmission is complete. (See **Mode Fault Error** on page 259).

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

The maximum frequency of the SPSCK for an SPI configured as a slave is the bus clock speed, which is twice as fast as the fastest master SPSCK clock that can be generated. The frequency of the SPSCK for an SPI configured as a slave does not have to correspond to any SPI baud rate. The baud rate only controls the speed of the SPSCK generated by an SPI configured as a master. Therefore, the frequency of the SPSCK for an SPI configured as a slave can be any frequency less than or equal to the bus speed.

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

Data written to the slave shift register during a a transmission remains in a buffer until the end of the transmission.

When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transmission. (See **Transmission Formats** on page 252).

If the write to the data register is late, the SPI transmits the data already in the shift register from the previous transmission.

NOTE: *To prevent SPSCK from appearing as a clock edge, SPSCK must be in the proper idle state before the slave is enabled.*

15.6 Transmission Formats

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

15.6.1 Clock Phase and Polarity Controls

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

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

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

15.6.2 Transmission Format When CPHA = 0

Figure 15-3 shows an SPI transmission in which CPHA (SPCR) is logic 0. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI (see **Mode Fault Error** on page 259). When CPHA = 0, the first SPSCK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SPSCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the transmission. The $\overline{SS}$ pin must be toggled high and then low again between each byte transmitted.

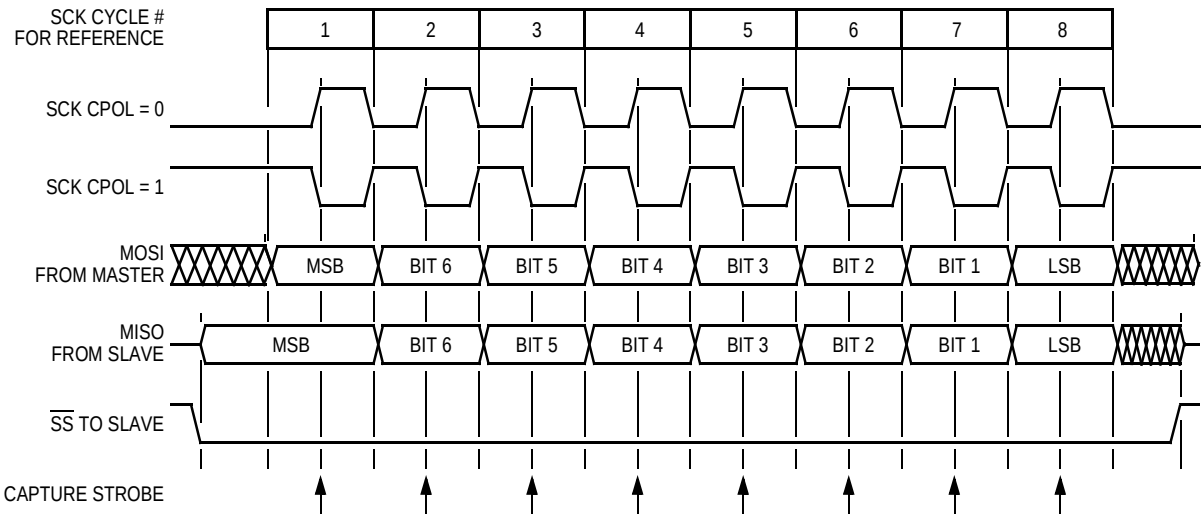


Figure 15-3. Transmission Format (CPHA = 0)

15.6.3 Transmission Format When CPHA = 1

Figure 15-4 shows an SPI transmission in which CPHA (SPCR) is logic 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See **Mode Fault Error** on page 259). When CPHA = 1, the master begins driving its MOSI pin on the first SPSCK edge. Therefore, the slave uses the first SPSCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

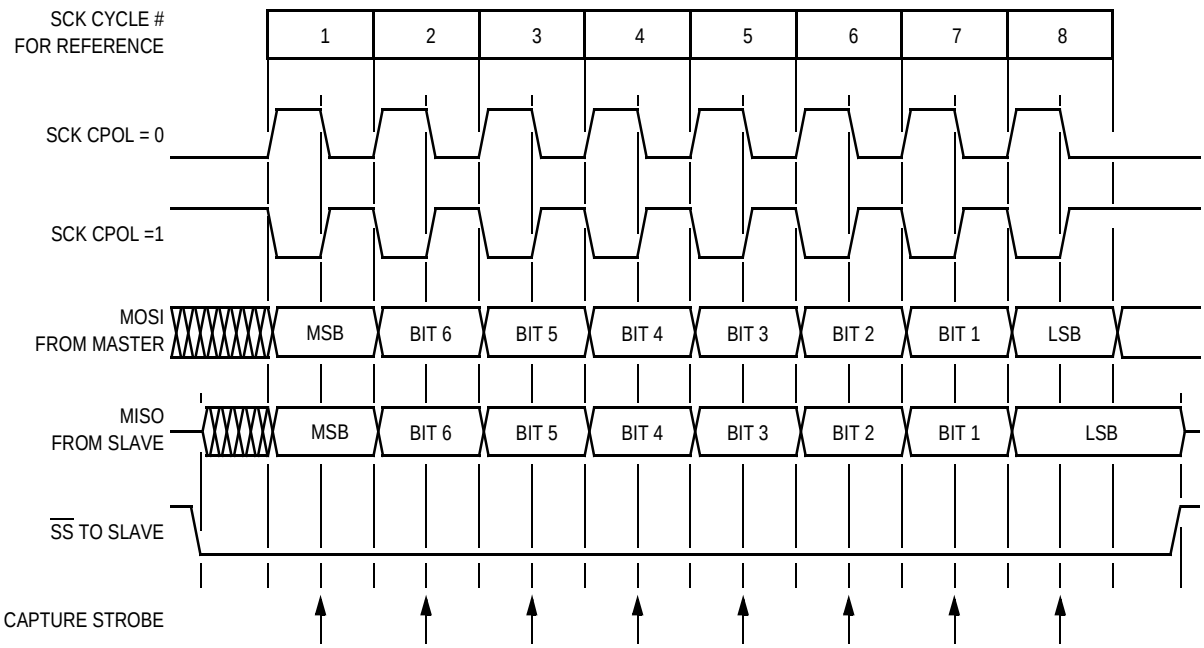


Figure 15-4. Transmission Format (CPHA = 1)

15.6.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), transmissions are started by a software write to the SPDR ($0012). CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SCK signal. When CPHA = 0, the SCK signal remains inactive for the first half of the first SCK cycle. When CPHA = 1, the first SCK cycle begins with an edge on the SCK line from its inactive to its active level. The SPI clock rate (selected by SPR1–SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See Figure 15-5). The internal SPI clock in the master is a free-running derivative of the internal MCU clock. It is only enabled when both the SPE and SPMSTR bits (SPCR) are set to conserve power. SCK edges occur half way through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR will occur relative to the slower SCK. This uncertainty causes the variation in the initiation delay shown in Figure 15-5. This delay will be no longer than a single SPI bit time. That is, the maximum delay between the write to SPDR and the start of the SPI transmission is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

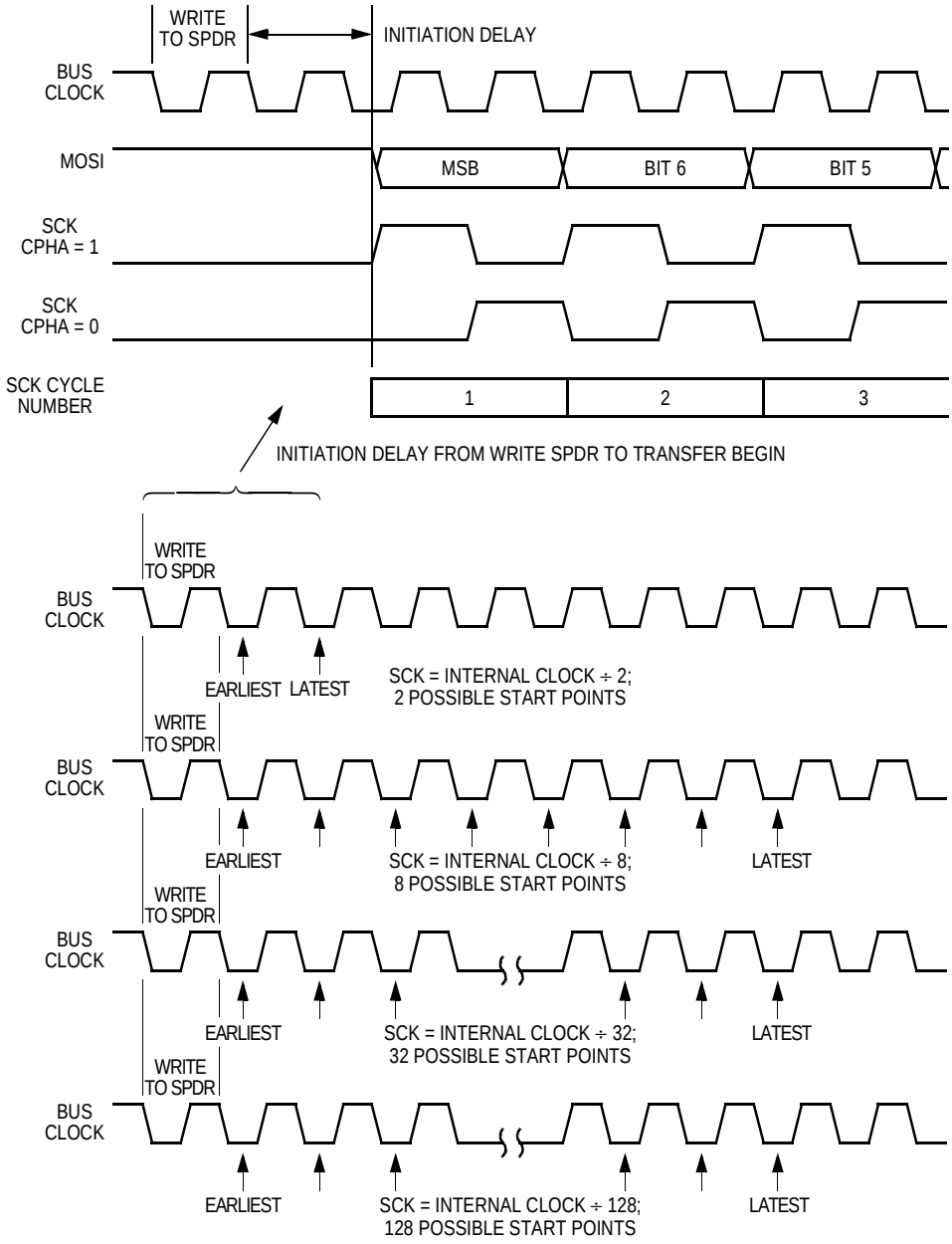


Figure 15-5. Transmission Start Delay (Master)

15.7 Error Conditions

Two flags signal SPI error conditions:

1. Overflow (OVRF in SPSCR) — Failing to read the SPI data register before the next byte enters the shift register sets the

OVRF bit. The new byte does not transfer to the receive data register, and the unread byte still can be read by accessing the SPI data register. OVRF is in the SPI status and control register.

2. Mode fault error (MODF in SPSCR) — The MODF bit indicates that the voltage on the slave select pin ($\overline{SS}$) is inconsistent with the mode of the SPI. MODF is in the SPI status and control register.

15.7.1 Overflow Error

The overflow flag (OVRF in SPSCR) becomes set if the SPI receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. (See Figure 15-3 and Figure 15-4.) If an overflow occurs, the data being received is not transferred to the receive data register so that the unread data can still be read. Therefore, an overflow error always indicates the loss of data.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE in SPSCR) is also set. MODF and OVRF can generate a receiver/error CPU interrupt request. (See Figure 15-8). It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

If an end-of-block transmission interrupt was meant to pull the MCU out of wait, having an overflow condition without overflow interrupts enabled causes the MCU to hang in wait mode. If the OVRF is enabled to generate an interrupt, it can pull the MCU out of wait mode instead.

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. Figure 15-6 shows how it is possible to miss an overflow.

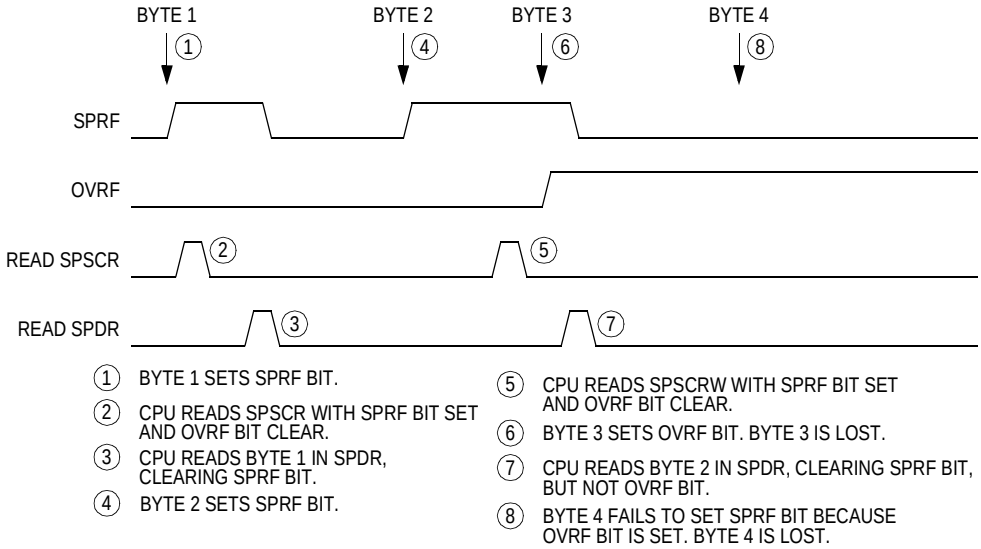


Figure 15-6. Missed Read of Overflow Condition

The first part of Figure 15-6 shows how to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF flag can be set in between the time that SPSCR and SPDR are read.

In this case, an overflow can be easily missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it will not be obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR after the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions will complete with an SPRF interrupt. Figure 15-7 illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit (SPSCR).

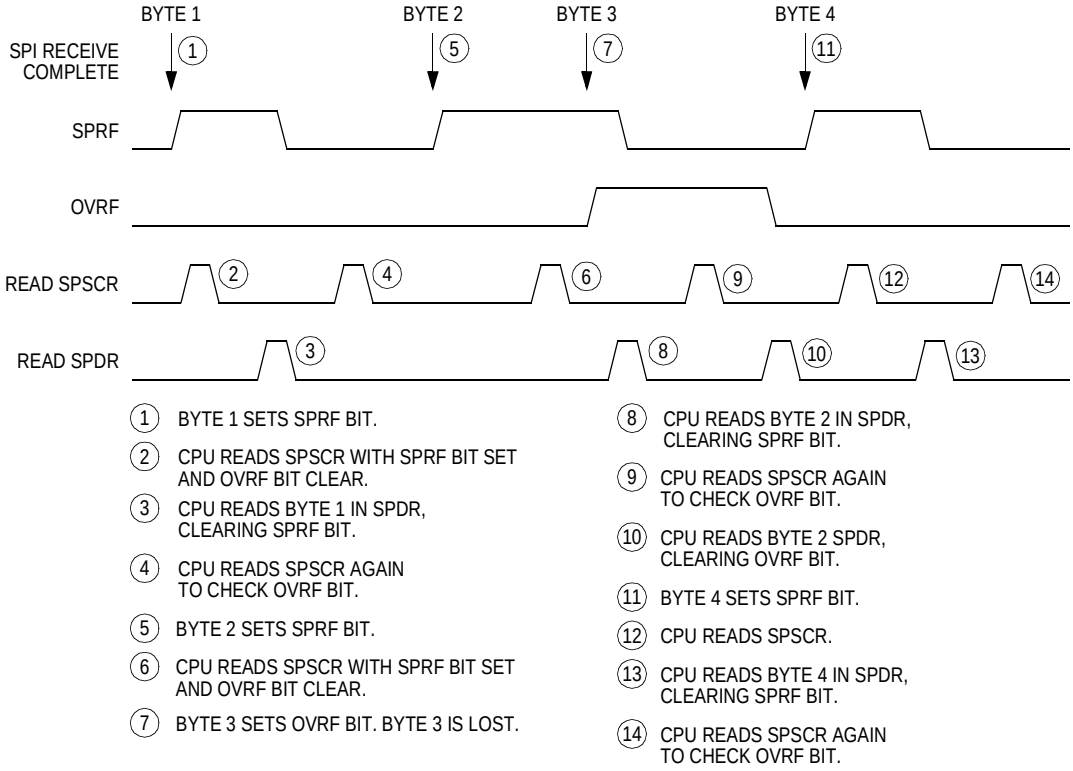


Figure 15-7. Clearing SPRF When OVRF Interrupt Is Not Enabled

15.7.2 Mode Fault Error

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

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE in SPSCR) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. MODF and OVRF can generate a receiver/error CPU interrupt request. (See Figure 15-8). It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes to logic 0. A mode fault in a master SPI causes the following events to occur:

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

NOTE: *To prevent bus contention with another master SPI after a mode fault error, clear all data direction register (DDR) bits associated with the SPI shared port pins.*

NOTE: *Setting the MODF flag (SPSCR) does not clear the SPMSTR bit. Reading SPMSTR when MODF = 1 will indicate a MODE fault error occurred in either master mode or slave mode.*

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{SS}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{SS}$ goes low and ends once the incoming SPSCK returns to its idle level after the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SPSCK returns to its IDLE level after the shift of the last data bit. (See **Transmission Formats** on page 252).

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

In a slave SPI (MSTR = 0), the MODF bit generates an SPI receiver/error CPU interrupt request if the ERRIE bit is set. The MODF

bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transmission by toggling the SPE bit of the slave.

NOTE: *A logic 1 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if a transmission has begun.*

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

15.8 Interrupts

Four SPI status flags can be enabled to generate CPU interrupt requests:

Table 15-4. SPI Interrupts

Flag	Request
SPTE (Transmitter Empty)	SPI Transmitter CPU Interrupt Request (SPTIE = 1)
SPRF (Receiver Full)	SPI Receiver CPU Interrupt Request (SPRIE = 1)
OVRF (Overflow)	SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1)
MODF (Mode Fault)	SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1, MODFEN = 1)

The SPI transmitter interrupt enable bit (SPTIE) enables the SPTE flag to generate transmitter CPU interrupt requests.

The SPI receiver interrupt enable bit (SPRIE) enables the SPRF bit to generate receiver CPU interrupt, provided that the SPI is enabled (SPE = 1).

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

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

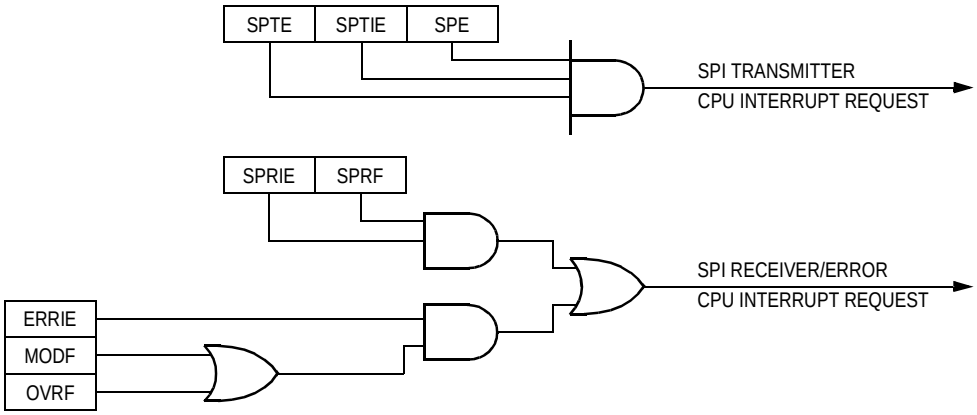


Figure 15-8. SPI Interrupt Request Generation

Two sources in the SPI status and control register can generate CPU interrupt requests:

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

15.9 Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE in SPSCR) indicates when the transmit data buffer is ready to accept new data. Write to the SPI data register only when the SPTE bit is high. Figure 15-9 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA:CPOL = 1:0).

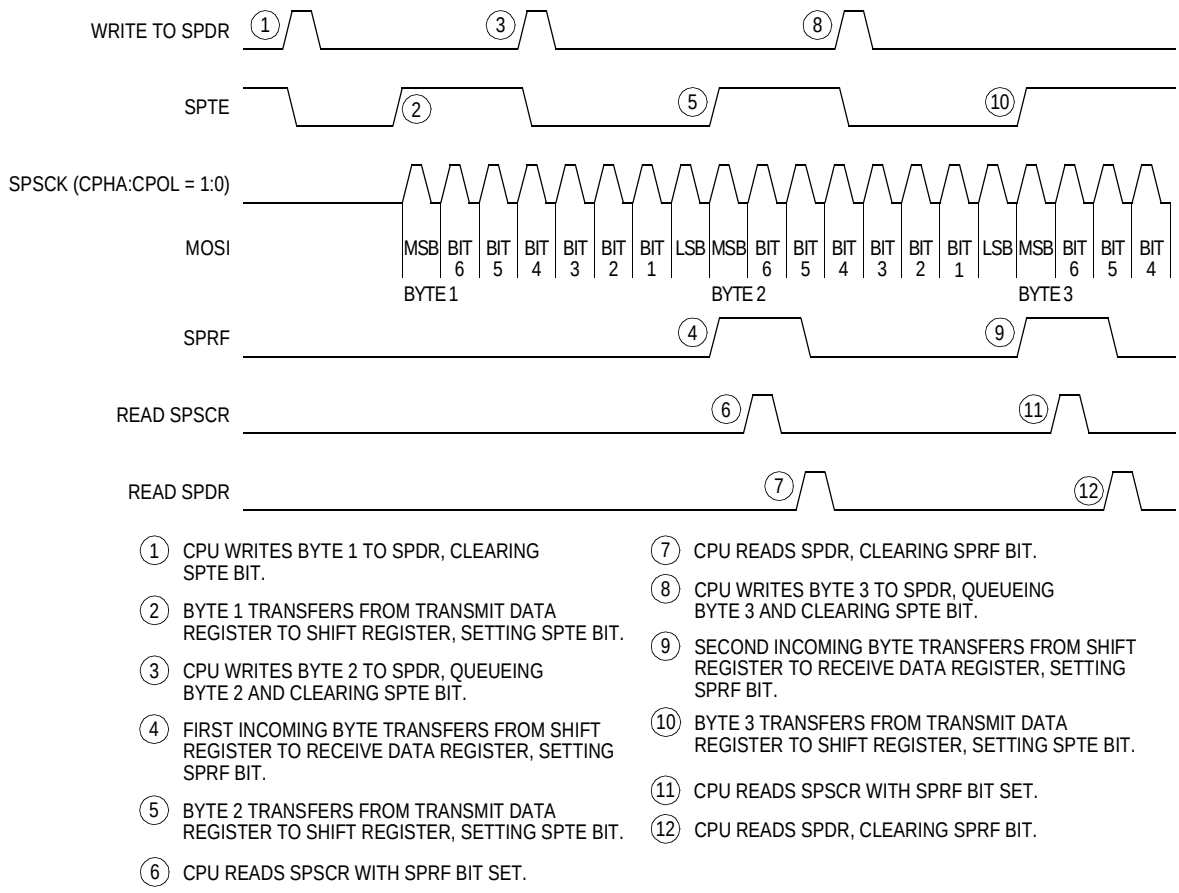


Figure 15-9. SPRF/SPTE CPU Interrupt Timing

For a slave, the transmit data buffer allows back-to-back transmissions to occur without the slave having to time the write of its data between the transmissions. Also, if no new data is written to the data buffer, the last value contained in the shift register will be the next data word transmitted.

15.10 Resetting the SPI

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

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

The following additional items are reset only by a system reset:

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

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

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

15.11 Low-Power Modes

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

15.11.1 Wait Mode

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

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

To exit wait mode when an overflow condition occurs, enable the OVRF bit to generate CPU interrupt requests by setting the error interrupt enable bit (ERRIE). (See **Interrupts** on page 261).

15.11.2 Stop Mode

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

15.12 SPI During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR, $FE03) enables software to clear status bits during the break state. (See **Flag Protection During Break Interrupts** on page 158).

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

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

Since the SPTE bit cannot be cleared during a break with the BCFE bit cleared, a write to the data register in break mode will not initiate a transmission nor will this data be transferred into the shift register.

Therefore, a write to the SPDR in break mode with the BCFE bit cleared has no effect.

15.13 I/O Signals

The SPI module has four I/O pins and shares three of them with a parallel I/O port.

- MISO — Data received
- MOSI — Data transmitted
- SPSCK — Serial clock
- $\overline{SS}$ — Slave select
- V_{SS} — Clock ground

The SPI has limited inter-integrated circuit (I^2C) capability (requiring software support) as a master in a single-master environment. To communicate with I^2C peripherals, MOSI becomes an open-drain output when the SPWOM bit in the SPI control register is set. In I^2C communication, the MOSI and MISO pins are connected to a bidirectional pin from the I^2C peripheral and through a pullup resistor to V_{DD}.

15.13.1 MISO (Master In/Slave Out)

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

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is logic 0 and its $\overline{SS}$ pin is at logic 0. To support a multiple-slave system, a logic 1 on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

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

15.13.2 MOSI (Master Out/Slave In)

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

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

15.13.3 SPSCK (Serial Clock)

The serial clock synchronizes data transmission between master and slave devices. In a master MCU, the SPSCK pin is the clock output. In a slave MCU, the SPSCK pin is the clock input. In full duplex operation, the master and slave MCUs exchange a byte of data in eight serial clock cycles.

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

15.13.4 $\overline{SS}$ (Slave Select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{SS}$ is used to select a slave. For CPHA = 0, the $\overline{SS}$ is used to define the start of a transmission. (See **15.6 Transmission Formats**.) Since it is used to indicate the start of a transmission, the $\overline{SS}$ must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low throughout the transmission for the CPHA = 1 format. See Figure 15-10.

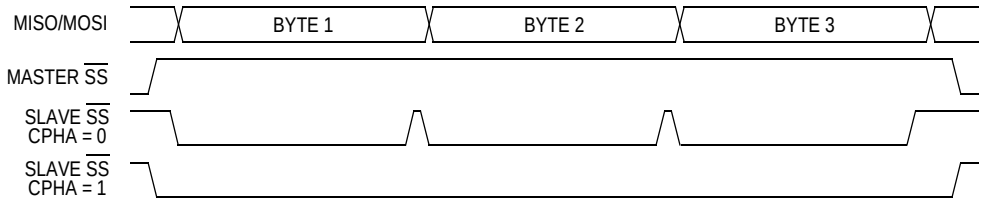


Figure 15-10. CPHA/SS Timing

When an SPI is configured as a slave, the SS pin is always configured as an input. It cannot be used as a general-purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of the SS from creating a MODF error. (See **SPI Status and Control Register** on page 272).

NOTE: *A logic 1 voltage on the SS pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCK clocks, even if a transmission already has begun.*

When an SPI is configured as a master, the SS input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. (See **Mode Fault Error** on page 259). For the state of the SS pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is low for an SPI master, the SS pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{SS}$ pin by configuring the appropriate pin as an input and reading the data register. (See Table 15-5).

Table 15-5. SPI Configuration

SPE	SPMSTR	MODFEN	SPI Configuration	State of $\overline{SS}$ Logic
0	X	X	Not Enabled	General-Purpose I/O; $\overline{SS}$ Ignored by SPI
1	0	X	Slave	Input-Only to SPI
1	1	0	Master without MODF	General-Purpose I/O; $\overline{SS}$ Ignored by SPI
1	1	1	Master with MODF	Input-Only to SPI

X = don't care

15.13.5 V_{SS} (Clock Ground)

V_{SS} is the ground return for the serial clock pin, SPSCK, and the ground for the port output buffers. To reduce the ground return path loop and minimize radio frequency (RF) emissions, connect the ground pin of the slave to the V_{SS} pin.

15.14 I/O Registers

Three registers control and monitor SPI operation:

- SPI control register (SPCR $0010)
- SPI status and control register (SPSCR $0011)
- SPI data register (SPDR $0012)

15.14.1 SPI Control Register

The SPI control register:

- Enables SPI module interrupt requests
- Selects CPU interrupt requests

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

Address: $000D

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

R = Reserved

Figure 15-11. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

This read/write bit enables CPU interrupt requests generated by the SPRF bit. The SPRF bit is set when a byte transfers from the shift register to the receive data register. Reset clears the SPRIE bit.

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

SPMSTR — SPI Master Bit

This read/write bit selects master mode operation or slave mode operation. Reset sets the SPMSTR bit.

1 = Master mode
0 = Slave mode

CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the SPSCK pin between transmissions. (See Figure 15-3 and Figure 15-4.) To transmit data between SPI modules, the SPI modules must have identical CPOL bits. Reset clears the CPOL bit.

CPHA — Clock Phase Bit

This read/write bit controls the timing relationship between the serial clock and SPI data. (See Figure 15-3 and Figure 15-4.) To transmit data between SPI modules, the SPI modules must have identical CPHA bits. When CPHA = 0, the $\overline{SS}$ pin of the slave SPI module must be set to logic 1 between bytes. (See Figure 15-10). Reset sets the CPHA bit.

When CPHA = 0 for a slave, the falling edge of $\overline{SS}$ indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. Once the transmission begins, no new data is allowed into the shift register from the data register. Therefore, the slave data register must be loaded with the desired transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the data register and transferred to the shift register at the current transmission.

When CPHA = 1 for a slave, the first edge of the SPSCK indicates the beginning of the transmission. The same applies when $\overline{SS}$ is high for a slave. The MISO pin is held in a high-impedance state, and the incoming SPSCK is ignored. In certain cases, it may also cause the MODF flag to be set. (See **Mode Fault Error** on page 259). A logic 1 on the $\overline{SS}$ pin does not in any way affect the state of the SPI state machine.

SPWOM — SPI Wired-OR Mode Bit

This read/write bit disables the pullup devices on pins SPSCK, MOSI, and MISO so that those pins become open-drain outputs.

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

SPE — SPI Enable Bit

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI (see **Resetting the SPI** on page 263). Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE — SPI Transmit Interrupt Enable Bit

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

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

15.14.2 SPI Status and Control Register

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

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

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

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

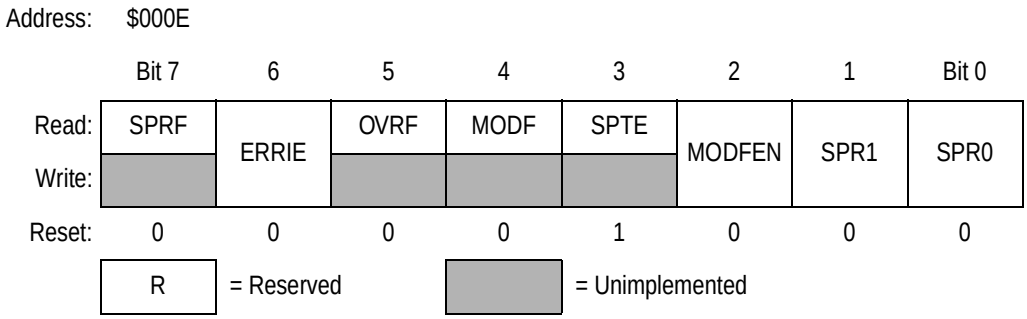


Figure 15-12. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

This clearable, read-only flag is set each time a byte transfers from the shift register to the receive data register. SPRF generates a CPU interrupt request if the SPRIE bit in the SPI control register is set also.

During an SPRF CPU interrupt, the CPU clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register. Any read of the SPI data register clears the SPRF bit.

Reset clears the SPRF bit.

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

ERRIE — Error Interrupt Enable Bit

This read-only bit enables the MODF and OVRF flags to generate CPU interrupt requests. Reset clears the ERRIE bit.

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

OVRF — Overflow Bit

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

1 = Overflow
0 = No overflow

MODF — Mode Fault Bit

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

1 = $\overline{SS}$ pin at inappropriate logic level
0 = $\overline{SS}$ pin at appropriate logic level

SPTE — SPI Transmitter Empty Bit

This clearable, read-only flag is set each time the transmit data register transfers a byte into the shift register. SPTE generates an SPTE CPU interrupt request if the SPTIE bit in the SPI control register is set also.

NOTE: *Do not write to the SPI data register unless the SPTE bit is high.*

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

Reset sets the SPTE bit.

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

MODFEN — Mode Fault Enable Bit

This read/write bit, when set to 1, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag. If the SPI is enabled as a master and the MODFEN bit is low, then the $\overline{SS}$ pin is available as a general-purpose I/O.

If the MODFEN bit is set, then this pin is not available as a general purpose I/O. When the SPI is enabled as a slave, the $\overline{SS}$ pin is not available as a general-purpose I/O regardless of the value of MODFEN. (See **SS (Slave Select)** on page 267).

If the MODFEN bit is low, the level of the $\overline{SS}$ pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation. (See **Mode Fault Error** on page 259).

SPR1 and SPR0 — SPI Baud Rate Select Bits

In master mode, these read/write bits select one of four baud rates as shown in Table 15-6. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 15-6. SPI Master Baud Rate Selection

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

Use this formula to calculate the SPI baud rate:

$$\text{Baud rate} = \frac{\text{CGMOUT}}{2 \times \text{BD}}$$

where:

CGMOUT = base clock output of the internal clock generator module (ICG), see **Internal Clock Generator (ICG) Module** on page 109.

BD = baud rate divisor

15.14.3 SPI Data Register

The SPI data register is the read/write buffer for the receive data register and the transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate buffers that can contain different values. See Figure 15-1

Address: $000F

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

Figure 15-13. SPI Data Register (SPDR)

R7–R0/T7–T0 — Receive/Transmit Data Bits

NOTE: *Do not use read-modify-write instructions on the SPI data register since the buffer read is not the same as the buffer written.*

Section 16. Timer Interface A (TIMA) Module

16.1 Contents

16.2 Introduction

This section describes the timer interface module (TIMA). The TIMA is a 2-channel timer that provides a timing reference with input capture, output compare, and pulse width modulation (PWM) functions. Figure 16-1 is a block diagram of the TIMA and Figure 16-2 provides a summary of the input/output (I/O) registers.

For further information regarding timers on M68HC08 family devices, please consult the HC08 Timer Reference Manual, TIM08RM/AD.

16.3 Features

Features of the TIMA include:

- Two input capture/output compare channels
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
- Buffered and unbuffered PWM signal generation
- Programmable TIMA clock input
  - 7-frequency internal bus clock prescaler selection
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMA counter stop and reset bits

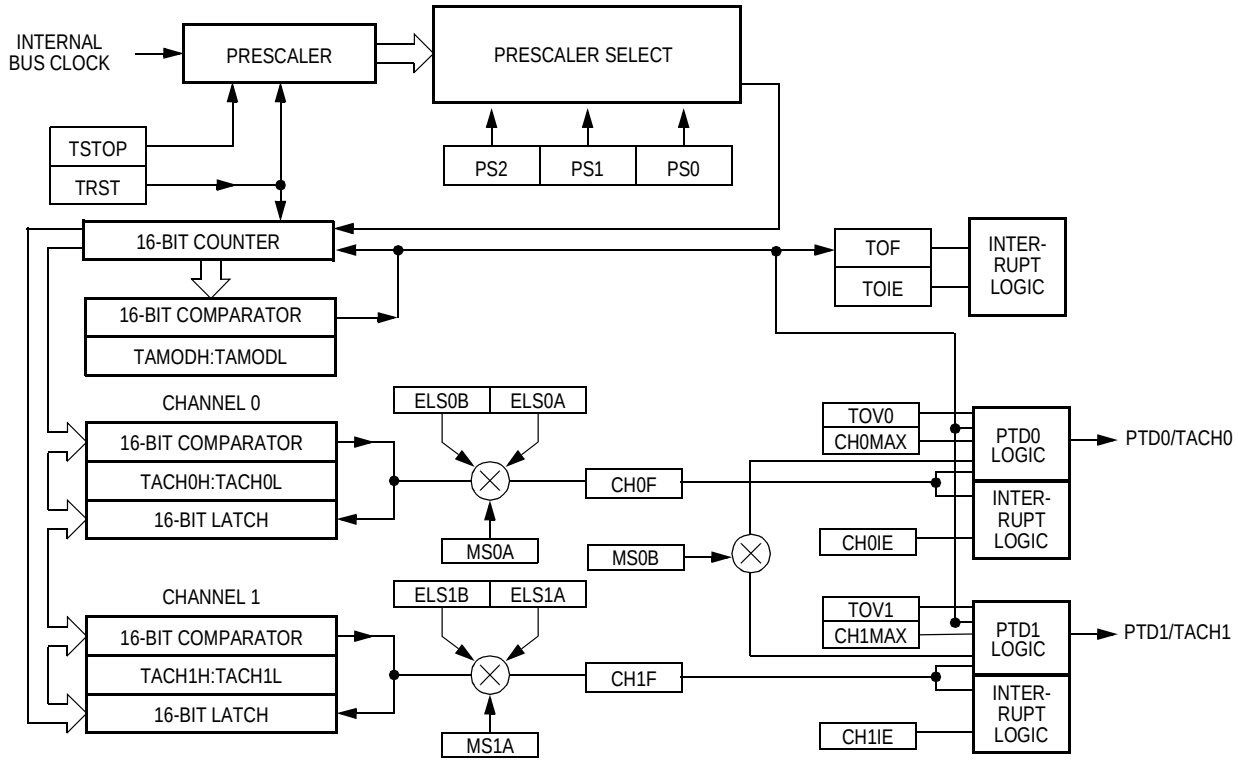


Figure 16-1. TIMA Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0020	TIMA Status and Control Register (TASC) See 291.	Read:	TOF	TOIE	TSTOP	0	R	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0021	TIMA Counter Register High (TACNTH) See 293.	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	TIMA Counter Register Low (TACNTL) See 293.	Read:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

Figure 16-2. TIMA I/O Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0023	TIMA Counter Modulo Register High (TAMODH) See 294.	Read: Write:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Reset:	1	1	1	1	1	1	1	1
$0024	TIMA Counter Modulo Register Low (TAMODL) See 294.	Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Reset:	1	1	1	1	1	1	1	1
$0025	TIMA Channel 0 Status and Control Register (TASC0) See 295.	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0026	TIMA Channel 0 Register High (TACH0H) See 300.	Read: Write:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Reset:	Indeterminate after reset							
$0027	TIMA Channel 0 Register Low (TACH0L) See 300.	Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Reset:	Indeterminate after reset							
$0028	TIMA Channel 1 Status and Control Register (TASC1) See 295.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$0029	TIMA Channel 1 Register High (TACH1H) See 300.	Read: Write:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Reset:	Indeterminate after reset							
$002A	TIMA Channel 1 Register Low (TACH1L) See 300.	Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Reset:	Indeterminate after reset							

R = Reserved = Unimplemented

Figure 16-2. TIMA I/O Register Summary (Continued)

16.4 Functional Description

Figure 16-1 shows the TIMA structure. The central component of the TIMA is the 16-bit TIMA counter that can operate as a free-running counter or a modulo up-counter. The TIMA counter provides the timing reference for the input capture and output compare functions. The TIMA counter modulo registers, TAMODH–TAMODL, control the modulo value of the TIMA counter. Software can read the TIMA counter value at any time without affecting the counting sequence.

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

16.4.1 TIMA Counter Prescaler

The TIMA clock source can be one of the seven prescaler outputs. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMA status and control register select the TIMA clock source.

16.4.2 Input Capture

An input capture function has three basic parts: edge select logic, an input capture latch, and a 16-bit counter. Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TASC0 through TASC1 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMA latches the contents of the TIMA counter into the TIMA channel registers, TACHxH–TACHxL. Input captures can generate TIMA central processor unit (CPU) interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The free-running counter contents are transferred to the TIMA channel status and control register (TACHxH–TACHxL, see **16.9.5 TIMA Channel Registers**) on each proper signal transition regardless of whether the TIMA channel flag (CH0F–CH1F in TASC0–TASC1 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register 2 bus cycles after the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see **16.9.5 TIMA Channel Registers**). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the input capture channel register (TACHxH–TACHxL).

16.4.3 Output Compare

With the output compare function, the TIMA can generate a periodic pulse with a programmable polarity, duration, and frequency. When the counter reaches the value in the registers of an output compare channel, the TIMA can set, clear, or toggle the channel pin. Output compares can generate TIMA CPU interrupt requests.

16.4.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in **16.4.3 Output Compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMA channel registers.

An unsynchronized write to the TIMA channel registers to change an output compare value could cause incorrect operation for up to two counter overflow periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that counter overflow period. Also, using a TIMA overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIMA may pass the new value before it is written.

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

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

16.4.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTD0/TACH0 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The output compare value in the TIMA channel 0 registers initially controls the output on the PTD0/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTD1/TACH1, is available as a general-purpose I/O pin.

NOTE: *In buffered output compare operation, do not write new output compare values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered output compares.*

16.4.4 Pulse Width Modulation (PWM)

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

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

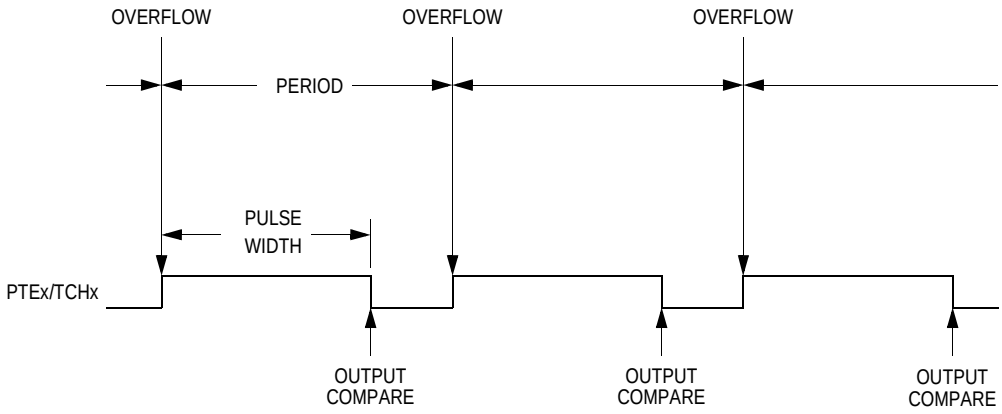


Figure 16-3. PWM Period and Pulse Width

The value in the TIMA counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIMA counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000 (see **16.9.1 TIMA Status and Control Register**).

The value in the TIMA channel registers determines the pulse width of the PWM output. The pulse width of an 8-bit PWM signal is variable in 256 increments. Writing $0080 (128) to the TIMA channel registers produces a duty cycle of 128/256 or 50%.

16.4.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **16.4.4 Pulse Width Modulation (PWM)**. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIMA channel registers.

An unsynchronized write to the TIMA channel registers to change a pulse width value could cause incorrect operation for up to two PWM periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that PWM period. Also, using a TIMA overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIMA may pass the new value before it is written to the TIMA channel registers.

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

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare also can cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

16.4.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTD0/TACH0 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The TIMA channel 0 registers

initially control the pulse width on the PTD0/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the pulse width are the ones written to last. TASC0 controls and monitors the buffered PWM function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTD1/TACH1, is available as a general-purpose I/O pin.

NOTE: *In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.*

16.4.4.3 PWM Initialization

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

1. In the TIMA status and control register (TASC):
   a. Stop the TIMA counter by setting the TIMA stop bit, TSTOP.
   b. Reset the TIMA counter prescaler by setting the TIMA reset bit, TRST.
2. In the TIMA counter modulo registers (TAMODH–TAMODL), write the value for the required PWM period.
3. In the TIMA channel x registers (TACHxH–TACHxL), write the value for the required pulse width.
4. In TIMA channel x status and control register (TASCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA. See Table 16-2.
   b. Write 1 to the toggle-on-overflow bit, TOVx.

c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB–ELSxA. The output action on compare must force the output to the complement of the pulse width level. See Table 16-2.

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

5. In the TIMA status control register (TASC), clear the TIMA stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMA channel 0 registers (TACH0H–TACH0L) initially control the buffered PWM output. TIMA status control register 0 (TASC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIMA overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and setting the TOVx bit generates a 100% duty cycle output. See **16.9.4 TIMA Channel Status and Control Registers**.

16.5 Interrupts

These TIMA sources can generate interrupt requests:

- TIMA overflow flag (TOF) — The TOF bit is set when the TIM counter reaches the modulo value programmed in the TIMA counter modulo registers. The TIMA overflow interrupt enable bit, TOIE, enables TIMA overflow CPU interrupt requests. TOF and TOIE are in the TIMA status and control register.

- TIMA channel flags (CH1F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMA CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

16.6 Low-Power Modes

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

16.6.1 Wait Mode

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

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

16.6.2 Stop Mode

The TIMA is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions or the state of the TIMA counter. TIMA operation resumes when the MCU exits stop mode.

16.7 TIMA During Break Interrupts

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

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

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

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

16.8 I/O Signals

Port D shares two of its pins with the TIMA. There is no external clock input to the TIMA prescaler. The two TIMA channel I/O pins are PTD0/TACH0 and PTD1/TACH1.

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTD0/TACH0 and PTD1/TACH1 can be configured as buffered output compare or buffered PWM pins.

16.9 I/O Registers

These I/O registers control and monitor TIMA operation:

- TIMA status and control register, TASC
- TIMA control registers, TACNTH–TACNTL
- TIMA counter modulo registers, TAMODH–TAMODL
- TIMA channel status and control registers, TASC0 and TASC1
- TIMA channel registers, TACH0H–TACH0L and TACH1H–TACH1L

16.9.1 TIMA Status and Control Register

The TIMA status and control register (TASC):

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

Address: $$0020

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

R = Reserved

Figure 16-4. TIMA Status and Control Register (TASC)

TOF — TIMA Overflow Flag Bit

This read/write flag is set when the TIMA counter reaches the modulo value programmed in the TIMA counter modulo registers. Clear TOF by reading the TIMA status and control register when TOF is set and then writing a logic 0 to TOF. If another TIMA overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

1 = TIMA counter has reached modulo value
0 = TIMA counter has not reached modulo value

TOIE — TIMA Overflow Interrupt Enable Bit

This read/write bit enables TIMA overflow interrupts when the TOF bit becomes set. Reset clears the TOIE bit.

1 = TIMA overflow interrupts enabled
0 = TIMA overflow interrupts disabled

TSTOP — TIMA Stop Bit

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

1 = TIMA counter stopped
0 = TIMA counter active

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

TRST — TIMA Reset Bit

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

1 = Prescaler and TIMA counter cleared
0 = No effect

NOTE: *Setting the TSTOP and TRST bits simultaneously stops the TIMA counter at a value of $0000.*

PS[2:0] — Prescaler Select Bits

These read/write bits select one of the seven prescaler outputs as the input to the TIMA counter as Table 16-1 shows. Reset clears the PS[2:0] bits.

Table 16-1. Prescaler Selection

PS[2:0]	TIMA Clock Source
0 0 0	Internal bus clock ÷ 1
0 0 1	Internal bus clock ÷ 2
0 1 0	Internal bus clock ÷ 4
0 1 1	Internal bus clock ÷ 8
1 0 0	Internal bus clock ÷ 16
1 0 1	Internal bus clock ÷ 32
1 1 0	Internal bus clock ÷ 64
1 1 1	Unused

16.9.2 TIMA Counter Registers

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

NOTE: *If TACNTH is read during a break interrupt, be sure to unlatch TACNTL by reading TACNTL before exiting the break interrupt. Otherwise, TACNTL retains the value latched during the break.*

Register Name and Address TACNTH — $0021

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

Register Name and Address TACNTL — $0022

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

R = Reserved

Figure 16-5. TIMA Counter Registers (TACNTH and TACNTL)

16.9.3 TIMA Counter Modulo Registers

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

Register Name and Address TAMODH — $0023

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

Register Name and Address TAMODL — $0024

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

Figure 16-6. TIMA Counter Modulo Registers (TMODH and TMODL)

NOTE: *Reset the TIMA counter before writing to the TIMA counter modulo registers.*

16.9.4 TIMA Channel Status and Control Registers

Each of the TIMA channel status and control registers:

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

Register Name and Address TASC0 — $0025

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Register Name and Address TASC1 — $0028

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

R R = Reserved

Figure 16-7. TIMA Channel Status and Control Registers (TASC0–TASC1)

CHxF — Channel x Flag Bit

When channel x is an input capture channel, this read/write bit is set when an active edge occurs on the channel x pin. When channel x is an output compare channel, CHxF is set when the value in the TIMA counter registers matches the value in the TIMA channel x registers.

When CHxIE = 1, clear CHxF by reading TIMA channel x status and control register with CHxF set, and then writing a logic 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a logic 1 to CHxF has no effect.

1 = Input capture or output compare on channel x
0 = No input capture or output compare on channel x

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMA CPU interrupts on channel x.

Reset clears the CHxIE bit.

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

MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIMA channel 0.

Setting MS0B disables the channel 1 status and control register and reverts TACH1 to general-purpose I/O.

Reset clears the MSxB bit.

1 = Buffered output compare/PWM operation enabled
0 = Buffered output compare/PWM operation disabled

MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. See Table 16-2.

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

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin once PWM, input capture, or output compare operation is enabled (see Table 16-2). Reset clears the MSxA bit.

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

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

MSxB:MSxA	ELSxB:ELSxA	Mode	Configuration
X0	00	Output Preset	Pin under Port Control; Initialize Timer Output Level High
X1	00		Pin under Port Control; Initialize Timer Output Level Low
00 00 00	01 10 11	Input Capture	Capture on Rising Edge Only Capture on Falling Edge Only Capture on Rising or Falling Edge
01 01 01	01 10 11	Output Compare or PWM	Toggle Output on Compare Clear Output on Compare Set Output on Compare
1X 1X 1X	01 10 11	Buffered Output Compare or Buffered PWM	Toggle Output on Compare Clear Output on Compare Set Output on Compare

NOTE: *Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIMA status and control register (TASC).*

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port B or port D, and pin PTD0/TACH0 or pin PTD1/TACH1 is available as a general-purpose I/O pin. However, channel x is at a state determined by these bits and becomes transparent to the respective pin when PWM, input capture, or output compare mode is enabled. Table 16-2 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

NOTE: *Before enabling a TIMA channel register for input capture operation, make sure that the PTDx/TACHx pin is stable for at least two bus clocks.*

TOVx — Toggle-On-Overflow Bit

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

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

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

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic 1 and clear output on compare is selected, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100 percent. As Figure 16-8 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at 100 percent duty cycle level until the cycle after CHxMAX is cleared.

NOTE: *The PWM 100 percent duty cycle is defined as output high all of the time. To generate the 100 percent duty cycle, use the CHxMAX bit in the TSCx register. The PWM 0 percent duty cycle is defined as output low all of the time. To generate the 0 percent duty cycle, select clear output on compare and then clear the TOVx bit (CHxMAX = 0).*

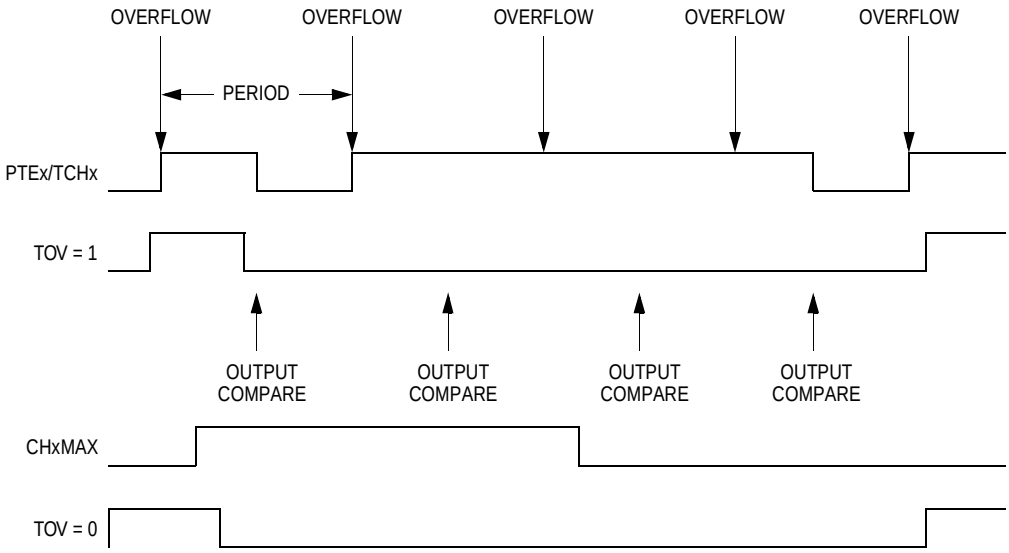


Figure 16-8. CHxMAX Latency

16.9.5 TIMA Channel Registers

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

In input capture mode (MSxB–MSxA = 0:0), reading the high byte of the TIMA channel x registers (TACHxH) inhibits input captures until the low byte (TACHxL) is read.

In output compare mode (MSxB–MSxA $\neq$ 0:0), writing to the high byte of the TIMA channel x registers (TACHxH) inhibits output compares and the CHxF bit until the low byte (TACHxL) is written.

Register Name and Address TACH0H — $0026

	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							

Register Name and Address TACH0L — $0027

	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							

Register Name and Address TACH1H — $0029

	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							

Register Name and Address TACH1L — $002A

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

Figure 16-9. TIMA Channel Registers (TACH0H/L–TACH1H/L)

Section 17. Timer Interface B (TIMB) Module

17.1 Contents

17.2 Introduction

This section describes the timer interface module (TIMB). The TIMB is a 2-channel timer that provides a timing reference with input capture, output compare, and pulse width modulation (PWM) functions. Figure 17-1 is a block diagram of the TIMB and Figure 17-2 provides a summary of the input/output (I/O) registers.

For further information regarding timers on M68HC08 family devices, please consult the HC08 Timer Reference Manual, TIM08RM/AD.

17.3 Features

Features of the TIMB include:

- Two input capture/output compare channels
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
- Buffered and unbuffered PWM signal generation
- Programmable TIMB clock input
  - 7-frequency internal bus clock prescaler selection
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMB counter stop and reset bits

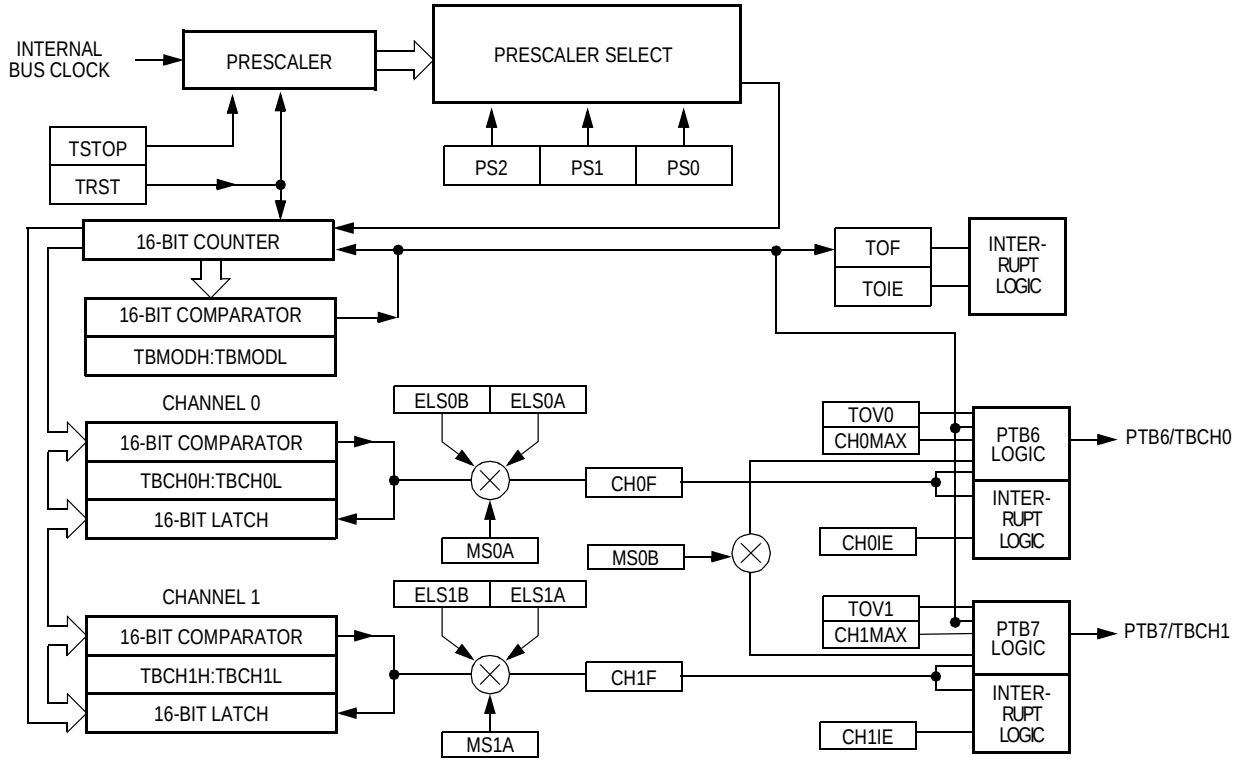


Figure 17-1. TIMB Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002B	TIMB Status and Control Register (TBSC) See 316.	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	R			
		Reset:	0	0	1	0	0	0	0	0
$002C	TIMB Counter Register High (TBCNTH) See 318.	Read:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002D	TIMB Counter Register Low (TBCNTL) See 318.	Read:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

Figure 17-2. TIMB I/O Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002E	TIMB Counter Modulo Register High (TBMODH) See 319.	Read: Write:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Reset:	1	1	1	1	1	1	1	1
$002F	TIMB Counter Modulo Register Low (TBMODL) See 319.	Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Reset:	1	1	1	1	1	1	1	1
$0030	TIMB Channel 0 Status and Control Register (TBSC0) See 320.	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0031	TIMB Channel 0 Register High (TBCH0H) See 324.	Read: Write:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Reset:	Indeterminate after reset							
$0032	TIMB Channel 0 Register Low (TBCH0L) See 324.	Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Reset:	Indeterminate after reset							
$0033	TIMB Channel 1 Status and Control Register (TBSC1) See 320.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$0034	TIMB Channel 1 Register High (TBCH1H) See 324.	Read: Write:	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
		Reset:	Indeterminate after reset							
$0035	TIMB Channel 1 Register Low (TBCH1L) See 324.	Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
		Reset:	Indeterminate after reset							

R = Reserved = Unimplemented

Figure 17-2. TIMB I/O Register Summary (Continued)

17.4 Functional Description

Figure 17-1 shows the TIMB structure. The central component of the TIMB is the 16-bit TIMB counter that can operate as a free-running counter or a modulo up-counter. The TIMB counter provides the timing reference for the input capture and output compare functions. The TIMB counter modulo registers, TBMODH–TBMODL, control the modulo value of the TIMB counter. Software can read the TIMB counter value at any time without affecting the counting sequence.

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

17.4.1 TIMB Counter Prescaler

The TIMB clock source can be one of the seven prescaler outputs. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMB status and control register select the TIMB clock source.

17.4.2 Input Capture

An input capture function has three basic parts: edge select logic, an input capture latch, and a 16-bit counter. Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TBSC0 through TBSC1 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMB latches the contents of the TIMB counter into the TIMB channel registers, TCHxH–TCHxL. Input captures can generate TIMB CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The free-running counter contents are transferred to the TIMB channel status and control register (TBCHxH–TBCHxL, see **17.9.5 TIMB Channel Registers**) on each proper signal transition regardless of whether the TIMB channel flag (CH0F–CH1F in TBSC0–TBSC1 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register 2 bus cycles after the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see **17.9.5 TIMB Channel Registers**). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the input capture channel register (TBCHxH–TBCHxL).

17.4.3 Output Compare

With the output compare function, the TIMB can generate a periodic pulse with a programmable polarity, duration, and frequency. When the counter reaches the value in the registers of an output compare channel, the TIMB can set, clear, or toggle the channel pin. Output compares can generate TIMB CPU interrupt requests.

17.4.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in **17.4.3 Output Compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMB channel registers.

An unsynchronized write to the TIMB channel registers to change an output compare value could cause incorrect operation for up to two counter overflow periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that counter overflow period. Also, using a TIMB overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIMB may pass the new value before it is written.

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

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

17.4.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTB6/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The output compare value in the TIMB channel 0 registers initially controls the output on the PTB6/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the output after the TIMB overflows. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTB7/TBCH1, is available as a general-purpose I/O pin.

NOTE: *In buffered output compare operation, do not write new output compare values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered output compares.*

17.4.4 Pulse Width Modulation (PWM)

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

As Figure 17-3 shows, the output compare value in the TIMB channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMB to clear the channel pin on output compare if the state of the PWM pulse is logic 1. Program the TIMB to set the pin if the state of the PWM pulse is logic 0.

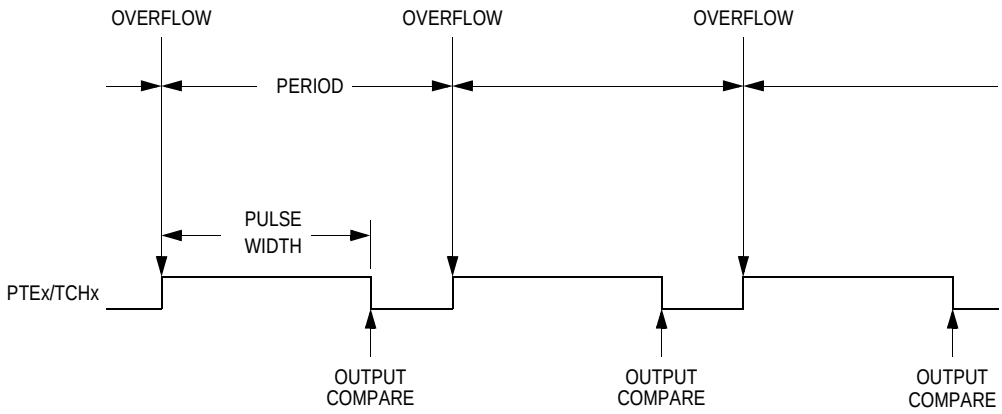


Figure 17-3. PWM Period and Pulse Width

The value in the TIMB counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIMB counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000 (see **17.9.1 TIMB Status and Control Register**).

The value in the TIMB channel registers determines the pulse width of the PWM output. The pulse width of an 8-bit PWM signal is variable in 256 increments. Writing $0080 (128) to the TIMB channel registers produces a duty cycle of 128/256 or 50%.

17.4.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **17.4.4 Pulse Width Modulation (PWM)**. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIMB channel registers.

An unsynchronized write to the TIMB channel registers to change a pulse width value could cause incorrect operation for up to two PWM periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that PWM period. Also, using a TIMB overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIMB may pass the new value before it is written to the TIMB channel registers.

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

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare also can cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

17.4.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTB6/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The TIMB channel 0 registers initially control the pulse width on the PTB6/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the pulse width are the ones written to last. TBSC0 controls and monitors the buffered PWM function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTB7/TBCH1, is available as a general-purpose I/O pin.

NOTE: *In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.*

17.4.4.3 PWM Initialization

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

1. In the TIMB status and control register (TBSC):
   a. Stop the TIMB counter by setting the TIMB stop bit, TSTOP.
   b. Reset the TIMB counter prescaler by setting the TIMB reset bit, TRST.
2. In the TIMB counter modulo registers (TBMODH–TBMODL), write the value for the required PWM period.
3. In the TIMB channel x registers (TBCHxH–TBCHxL), write the value for the required pulse width.

4. In TIMB channel x status and control register (TBSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA. See Table 17-2.
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB–ELSxA. The output action on compare must force the output to the complement of the pulse width level. See Table 17-2.

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

5. In the TIMB status control register (TBSC), clear the TIMB stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMB channel 0 registers (TBCH0H–TBCH0L) initially control the buffered PWM output. TIMB status control register 0 (TBSC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIMB overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and setting the TOVx bit generates a 100% duty cycle output. See **17.9.4 TIMB Channel Status and Control Registers**.

17.5 Interrupts

These TIMB sources can generate interrupt requests:

- TIMB overflow flag (TOF) — The TOF bit is set when the TIMB counter reaches the modulo value programmed in the TIMB counter modulo registers. The TIMB overflow interrupt enable bit, TOIE, enables TIMB overflow CPU interrupt requests. TOF and TOIE are in the TIMB status and control register.
- TIMB channel flags (CH1F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMB CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

17.6 Low-Power Modes

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

17.6.1 Wait Mode

The TIMB remains active after the execution of a WAIT instruction. In wait mode, the TIMB registers are not accessible by the central processor unit (CPU). Any enabled CPU interrupt request from the TIMB can bring the MCU out of wait mode.

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

17.6.2 Stop Mode

The TIMB is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions or the state of the TIMB counter. TIMB operation resumes when the MCU exits stop mode.

17.7 TIMB During Break Interrupts

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

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

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

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

17.8 I/O Signals

Port B shares two of its pins with the TIMB. There is no external clock input to the TIMB prescaler. The two TIMB channel I/O pins are PTB6/TBCH0 and PTB7/TBCH1. See **Section 22. Input/Output (I/O) Ports**.

17.8.1 TIMB Channel I/O Pins (PTB7/TBCH1–PTB6/TBCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTB6/TBCH0 and PTB7/TBCH1 can be configured as buffered output compare or buffered PWM pins.

17.9 I/O Registers

These I/O registers control and monitor TIMB operation:

- TIMB status and control register, TBSC
- TIMB control registers, TBCNTH–TBCNTL
- TIMB counter modulo registers, TBMODH–TBMODL
- TIMB channel status and control registers, TBSC0 and TBSC1
- TIMB channel registers, TBCH0H–TBCH0L and TBCH1H–TBCH1L

17.9.1 TIMB Status and Control Register

The TIMB status and control register:

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

Address: $002B

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

R = Reserved

Figure 17-4. TIMB Status and Control Register (TBSC)

TOF — TIMB Overflow Flag Bit

This read/write flag is set when the TIMB counter reaches the modulo value programmed in the TIMB counter modulo registers. Clear TOF by reading the TIMB status and control register when TOF is set and then writing a logic 0 to TOF. If another TIMB overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

1 = TIMB counter has reached modulo value
0 = TIMB counter has not reached modulo value

TOIE — TIMB Overflow Interrupt Enable Bit

This read/write bit enables TIMB overflow interrupts when the TOF bit becomes set. Reset clears the TOIE bit.

1 = TIMB overflow interrupts enabled
0 = TIMB overflow interrupts disabled

TSTOP — TIMB Stop Bit

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

1 = TIMB counter stopped
0 = TIMB counter active

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

TRST — TIMB Reset Bit

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

1 = Prescaler and TIMB counter cleared
0 = No effect

NOTE: *Setting the TSTOP and TRST bits simultaneously stops the TIMB counter at a value of $0000.*

PS[2:0] — Prescaler Select Bits

These read/write bits select one of the seven prescaler outputs as the input to the TIMB counter as Table 17-1 shows. Reset clears the PS[2:0] bits.

Table 17-1. Prescaler Selection

PS[2:0]	TIMB Clock Source
0 0 0	Internal bus clock ÷ 1
0 0 1	Internal bus clock ÷ 2
0 1 0	Internal bus clock ÷ 4
0 1 1	Internal bus clock ÷ 8
1 0 0	Internal bus clock ÷ 16
1 0 1	Internal bus clock ÷ 32
1 1 0	Internal bus clock ÷ 64
1 1 1	Unused

17.9.2 TIMB Counter Registers

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

NOTE: *If TBCNTH is read during a break interrupt, be sure to unlatch TBCNTL by reading TBCNTL before exiting the break interrupt. Otherwise, TBCNTL retains the value latched during the break.*

Register Name and Address TBCNTH — $002C

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

Register Name and Address TBCNTL — $002D

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

R = Reserved

Figure 17-5. TIMB Counter Registers (TBCNTH and TBCNTL)

17.9.3 TIMB Counter Modulo Registers

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

Register Name and Address TBMODH — $002E

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

Register Name and Address TBMODL — $002F

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

Figure 17-6. TIMB Counter Modulo Registers (TMODH and TMODL)

NOTE: *Reset the TIMB counter before writing to the TIMB counter modulo registers.*

17.9.4 TIMB Channel Status and Control Registers

Each of the TIMB channel status and control registers:

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

Register Name and Address TBSC0 — $0030

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Register Name and Address TBSC1 — $0033

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

R = Reserved

Figure 17-7. TIMB Channel Status and Control Registers (TBSC0–TBSC1)

CHxF — Channel x Flag Bit

When channel x is an input capture channel, this read/write bit is set when an active edge occurs on the channel x pin. When channel x is an output compare channel, CHxF is set when the value in the TIMB counter registers matches the value in the TIMB channel x registers.

When CHxIE = 1, clear CHxF by reading TIMB channel x status and control register with CHxF set, and then writing a logic 0 to CHxF. If

another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a logic 1 to CHxF has no effect.

1 = Input capture or output compare on channel x
0 = No input capture or output compare on channel x

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMB CPU interrupts on channel x.

Reset clears the CHxIE bit.

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

MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIMB channel 0.

Setting MS0B disables the channel 1 status and control register and reverts TBCH1 to general-purpose I/O.

Reset clears the MSxB bit.

1 = Buffered output compare/PWM operation enabled
0 = Buffered output compare/PWM operation disabled

MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. See Table 17-2.

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

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin once PWM, input capture, or output compare operation is enabled. See Table 17-2. Reset clears the MSxA bit.

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

NOTE: *Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIMB status and control register (TBSC).*

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port B, and pin PTBx/TBCHx is available as a general-purpose I/O pin. However, channel x is at a state determined by these bits and becomes transparent to the respective pin when PWM, input capture, or output compare mode is enabled. Table 17-2 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 17-2. Mode, Edge, and Level Selection

MSxB:MSxA	ELSxB:ELSxA	Mode	Configuration
X0	00	Output Preset	Pin under Port Control; Initialize Timer Output Level High
X1	00		Pin under Port Control; Initialize Timer Output Level Low
00 00 00	01 10 11	Input Capture	Capture on Rising Edge Only Capture on Falling Edge Only Capture on Rising or Falling Edge
01 01 01	01 10 11	Output Compare or PWM	Toggle Output on Compare Clear Output on Compare Set Output on Compare
1X 1X 1X	01 10 11	Buffered Output Compare or Buffered PWM	Toggle Output on Compare Clear Output on Compare Set Output on Compare

NOTE: *Before enabling a TIMB channel register for input capture operation, make sure that the* PTBx/TBCHx *pin is stable for at least two bus clocks.*

TOVx — Toggle-On-Overflow Bit

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

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

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

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic 1 and clear output on compare is selected, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100 percent. As Figure 17-8 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at 100 percent duty cycle level until the cycle after CHxMAX is cleared.

NOTE: *The PWM 100 percent duty cycle is defined as output high all of the time. To generate the 100 percent duty cycle, use the CHxMAX bit in the TSCx register. The PWM 0 percent duty cycle is defined as output low all of the time. To generate the 0 percent duty cycle, select clear output on compare and then clear the TOVx bit (CHxMAX = 0).*

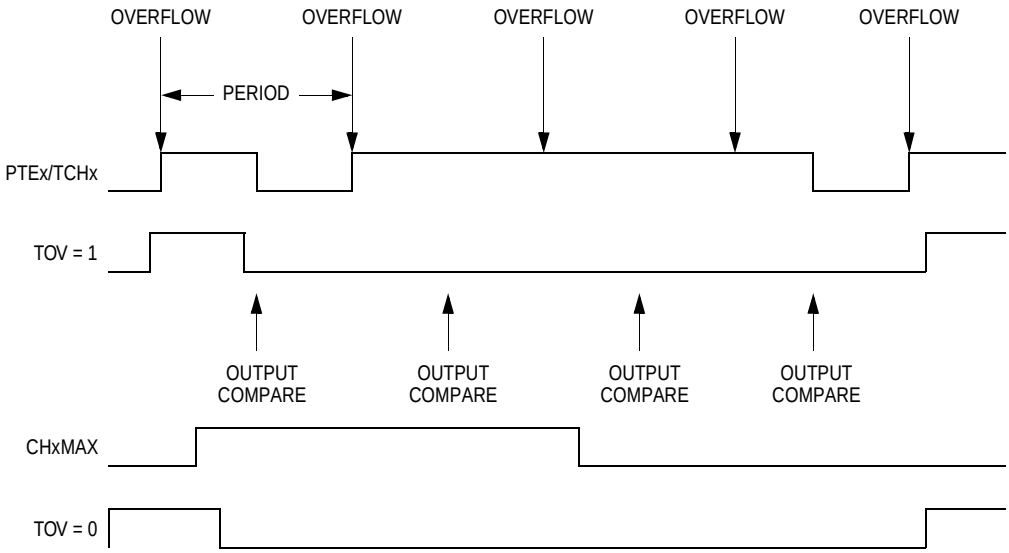


Figure 17-8. CHxMAX Latency

17.9.5 TIMB Channel Registers

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

In input capture mode (MSxB–MSxA = 0:0), reading the high byte of the TIMB channel x registers (TBCHxH) inhibits input captures until the low byte (TBCHxL) is read.

In output compare mode (MSxB–MSxA ≠ 0:0), writing to the high byte of the TIMB channel x registers (TBCHxH) inhibits output compares and the CHxF bit until the low byte (TBCHxL) is written.

Register Name and Address TBCH0H — $0031

	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							

Register Name and Address TBCH0L — $0032

	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							

Register Name and Address TBCH1H — $0034

	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							

Register Name and Address TBCH1L — $0035

	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 17-9. TIMB Channel Registers (TBCH0H/L–TBCH1H/L)

Section 18. BEMF Module

18.1 Contents

18.2 Introduction

This section describes the BEMF Module. The BEMF counter integrates over time, while the PTD0/TACH0 pin is active.

This function is useful for measuring recirculation currents in motors occurring on switching of inductive loads.

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

18.3 Functional Description

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

18.4 BEMF Register

The BEMF Register contains the 8 read-only bits of the BEMF Counter, showing its actual value. A read access to the BEMF Register resets all counter bits to logic zero.

Address: $000B

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

Figure 18-1. BEMF Register (BEMF)

18.5 Input Signal

Port D shares the PTD0/TACH0 pin with the BEMF module. To measure an external signal with the BEMF module, PTD0/ATD8 must be configured as an input (DDRD0=0).

18.6 Low Power Modes

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

18.6.1 Wait Mode

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

18.6.2 Stop Mode

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

Section 19. Keyboard Interrupt (KBD) Module

19.1 Contents

19.2 Introduction

The keyboard interrupt (KBD) module provides five independently maskable external interrupt pins.

19.3 Features

KBD features include:

- Five keyboard interrupt pins (PTA4/$\overline{\text{KBD4}}$–PTA0/$\overline{\text{KBD0}}$) with internal pullups, with separate keyboard interrupt enable bits and one keyboard interrupt mask
- Hysteresis buffers
- Programmable edge-only or edge- and level- interrupt sensitivity
- Automatic interrupt acknowledge
- Exit from low-power modes

19.4 Functional Description

Writing to the KBIE4–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its internal pullup device. A logic 0 applied to an enabled keyboard interrupt pin latches a keyboard interrupt request.

A keyboard interrupt is latched when one or more keyboard pins goes low after all were high. The MODEK bit in the keyboard status and control register controls the triggering mode of the keyboard interrupt.

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

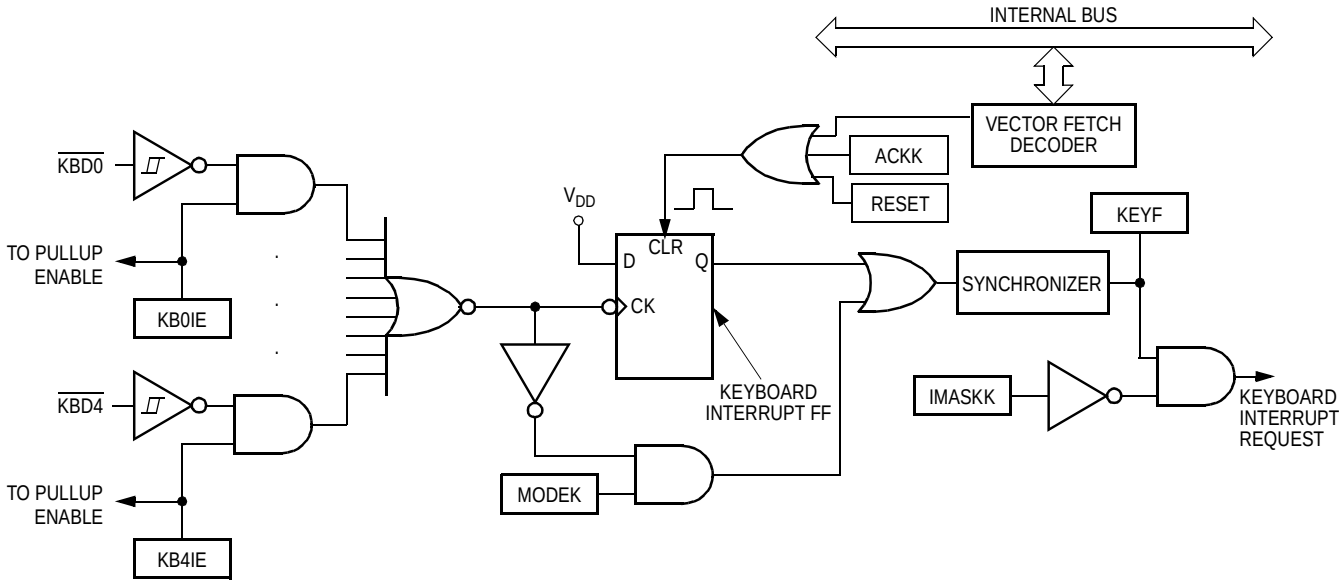


Figure 19-1. Keyboard Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001A	Keyboard Status and Control Register (KBSCR) See 333.	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
		Write:						ACKK		
		Reset:	0	0	0	0	0	0	0	0
$001B	Keyboard Interrupt Enable Register (KBIER) See 334.	Read:	0	0	0	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 19-2. KBD I/O Register Summary

If the MODEK bit is set, the keyboard interrupt pins are both falling edge- and low level-sensitive, and both of the following actions must occur to clear a keyboard interrupt request:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACKK bit in the keyboard status and control register (KBSCR). The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit prior to leaving an interrupt service routine also can prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the vector address at locations $FFE4 and $FFE5.
- Return of all enabled keyboard interrupt pins to logic 1. As long as any enabled keyboard interrupt pin is at logic 0, the keyboard interrupt remains set.

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

If the MODEK bit is clear, the keyboard interrupt pin is falling edge-sensitive only. With MODEK clear, a vector fetch or software clear immediately clears the keyboard interrupt request.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt pin stays at logic 0.

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

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

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

19.5 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup to reach a logic 1. Therefore, a false interrupt can occur as soon as the pin is enabled.

To prevent a false interrupt on keyboard initialization:

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

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

Another way to avoid a false interrupt:

1. Configure the keyboard pins as outputs by setting the appropriate DDRA bits in data direction register A.
2. Write logic 1s to the appropriate port A data register bits.
3. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.

19.6 Low-Power Modes

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

19.6.1 Wait Mode

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

19.6.2 Stop Mode

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

19.7 Keyboard Module During Break Interrupts

The BCFE bit in the break flag control register (SBFCR) enables software to clear status bits during the break state.

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

To protect the KEYF bit during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0, writing to the keyboard acknowledge bit (ACKK) in the keyboard status and control register during the break state has no effect. See **19.8.1 Keyboard Status and Control Register**.

19.8 I/O Registers

These registers control and monitor operation of the keyboard module:

- Keyboard status and control register, KBSCR
- Keyboard interrupt enable register, KBIER

19.8.1 Keyboard Status and Control Register

The keyboard status and control register:

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

Address: $001A

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

= Unimplemented

Figure 19-3. Keyboard Status and Control Register (KBSCR)

Bits 7–4 — Not used

These read-only bits always read as logic 0s.

KEYF — Keyboard Flag Bit

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

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

ACKK — Keyboard Acknowledge Bit

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

IMASKK — Keyboard Interrupt Mask Bit

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

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

MODEK — Keyboard Triggering Sensitivity Bit

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

1 = Keyboard interrupt requests on falling edges and low levels
0 = Keyboard interrupt requests on falling edges only

19.8.2 Keyboard Interrupt Enable Register

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

Address: $001B

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

= Unimplemented

Figure 19-4. Keyboard Interrupt Enable Register (KBIER)

KBIE4–KBIE0 — Keyboard Interrupt Enable Bits

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

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

Section 20. Timebase Module (TBM)

20.1 Contents

20.2 Introduction

This section describes the timebase module (TBM). The TBM will generate periodic interrupts at user selectable rates using a counter clocked by either the internal or external clock sources. This TBM version uses 15 divider stages, eight of which are user selectable.

NOTE: *The TBM on this device differs from that of the MC68HC908KX8 in that it has an additional divide-by-128 at the front end of the divider chain.*

For further information regarding timers on M68HC08 family devices, please consult the HC08 Timer Reference Manual, TIM08RM/AD.

20.3 Features

Features of the TBM module include:

- Software configurable periodic interrupts with divide-by-1024, 2048, 4096, 8192, 16384, 262144, 1048576 and 4194304 taps of the selected clock source
- Configurable for operation during stop mode to allow periodic wake up from stop

20.4 Functional Description

This module can generate a periodic interrupt by dividing the clock source supplied from the internal clock generator module, TBMCLK. Note that this clock source is the external clock ECLK when the ECGON bit in the ICG control register (ICGCR) is set. Otherwise, TBMCLK is driven at the internally generated clock frequency (ICLK). In other words, if the external clock is enabled it will be used as the TBMCLK, even if the MCU bus clock is based on the internal clock.

The counter is initialized to all 0s when TBON bit is cleared. The counter, shown in Figure 20-1, starts counting when the TBON bit is set. When the counter overflows at the tap selected by TBR2–TBR0, the TBIF bit gets set. If the TBIE bit is set, an interrupt request is sent to the CPU. The TBIF flag is cleared by writing a 1 to the TACK bit. The first time the TBIF flag is set after enabling the timebase module, the interrupt is generated at approximately half of the overflow period. Subsequent events occur at the exact period.

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

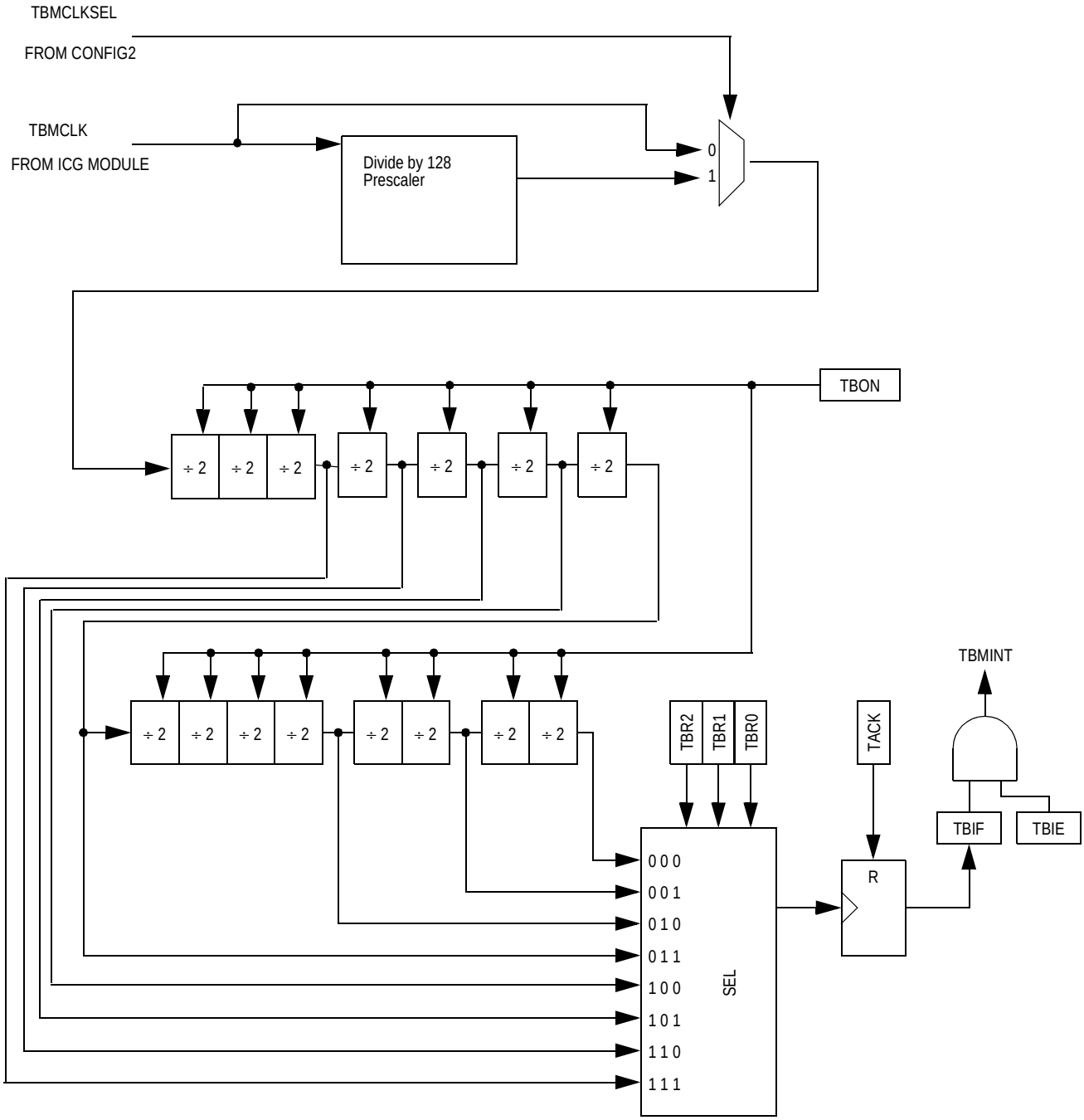


Figure 20-1. Timebase Block Diagram

20.5 Interrupts

The timebase module can periodically interrupt the CPU with a rate defined by the selected TBMCLK and the select bits TBR2–TBR0. When the timebase counter chain rolls over, the TBIF flag is set. If the TBIE bit is set, enabling the timebase interrupt, the counter chain overflow will generate a CPU interrupt request.

Interrupts must be acknowledged by writing a logic 1 to the TACK bit.

20.6 TBM Interrupt Rate

The interrupt rate is determined by the equation:

$$t_{TBMRATE} = \frac{1}{f_{TBMRATE}} = \frac{Divider}{f_{TBMCLK}}$$

where:

f_{TBMCLK} = Frequency supplied from the internal clock generator (ICG) module

Divider = Divider value as determined by TBR2–TBR0 settings. See **Table 20-1.**

As an example, a clock source of 4.9152 MHz and the TBR2–TBR0 set to {011}, the divider tap is 128 and the interrupt rate calculates to $128/4.9152 \times 10^6 = 26\ \mu s$.

Table 20-1. Timebase Divider Selection

TBR2	TBR1	TBR0	Divider Tap	
			TMBCLKSEL	
			0	1
0	0	0	32,768	4,194,304
0	0	1	8192	1,048,576
0	1	0	2048	262144
0	1	1	128	16,384
1	0	0	64	8192
1	0	1	32	4096

Table 20-1. Timebase Divider Selection

TBR2	TBR1	TBR0	Divider Tap	
			TMBCLKSEL	
			0	1
1	1	0	16	2048
1	1	1	8	1024

NOTE: Do not change TBR2–TBR0 bits while the timebase is enabled (TBON = 1).

20.7 Low-Power Modes

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

20.7.1 Wait Mode

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

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

20.7.2 Stop Mode

The timebase module may remain active after execution of the STOP instruction if the internal clock generator has been enabled to operate during stop mode through the OSCENINSTOP bit in the configuration register. The timebase module can be used in this mode to generate a periodic wake up from stop mode.

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

If the timebase functions are not required during stop mode, reduce power consumption by disabling the timebase module before executing the STOP instruction.

20.8 Timebase Control Register

The timebase has one register, the timebase control register (TBCR), which is used to enable the timebase interrupts and set the rate.

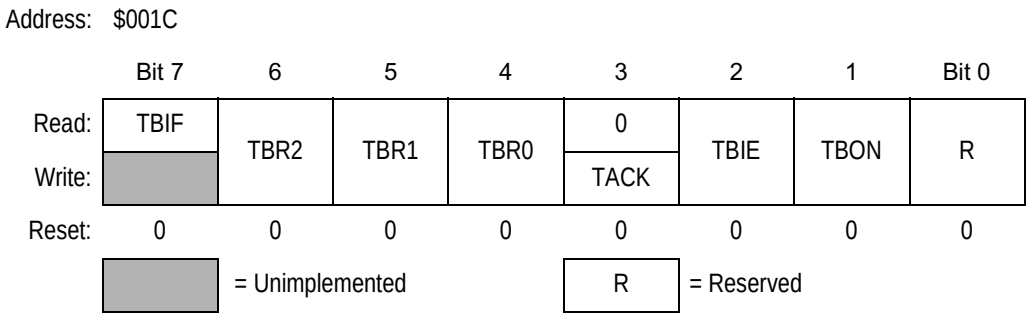


Figure 20-2. Timebase Control Register (TBCR)

TBIF — Timebase Interrupt Flag

This read-only flag bit is set when the timebase counter has rolled over.

1 = Timebase interrupt pending
0 = Timebase interrupt not pending

TBR2–TBR0 — Timebase Divider Selection Bits

These read/write bits select the tap in the counter to be used for timebase interrupts as shown in Table 20-1.

NOTE: *Do not change TBR2–TBR0 bits while the timebase is enabled (TBON = 1).*

TACK— Timebase ACKnowledge Bit

The TACK bit is a write-only bit and always reads as 0. Writing a logic 1 to this bit clears TBIF, the timebase interrupt flag bit. Writing a logic 0 to this bit has no effect.

1 = Clear timebase interrupt flag
0 = No effect

TBIE — Timebase Interrupt Enabled Bit

This read/write bit enables the timebase interrupt when the TBIF bit becomes set. Reset clears the TBIE bit.

1 = Timebase interrupt is enabled.
0 = Timebase interrupt is disabled.

TBON — Timebase Enabled Bit

This read/write bit enables the timebase. Timebase may be turned off to reduce power consumption when its function is not necessary. The counter can be initialized by clearing and then setting this bit. Reset clears the TBON bit.

1 = Timebase is enabled.
0 = Timebase is disabled and the counter initialized to 0s.

NOTE: *Clearing TBON has no effect on the TBIF flag.*

Section 21. Analog-to-Digital Converter (ADC) Module

21.1 Contents

21.2 Introduction

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

For further information regarding analog-to-digital converters on Motorola microcontrollers, please consult the HC08 ADC Reference Manual, ADCRM/AD.

21.3 Features

Features of the ADC module include:

- 8 channels with multiplexed input
- Linear successive approximation
- 10-bit resolution, 8-bit accuracy
- Single or continuous conversion
- Conversion complete flag or conversion complete interrupt
- Selectable ADC clock
- Left or right justified result
- Left justified sign data mode
- High impedance buffered ADC input

21.4 Functional Description

Eight ADC channels are available for sampling external sources at pins PTB7:PTB0. To achieve the best possible accuracy, these pins are implemented as input-only pins when the analog-to-digital (A/D) feature is enabled. An analog multiplexer allows the single ADC to select one of the 8 ADC channels as ADC voltage IN (ADCVIN). ADCVIN is converted by the successive approximation algorithm. When the conversion is completed, the ADC places the result in the ADC data register (ADRH and ADRL) and sets a flag or generates an interrupt. See Figure 21-1.

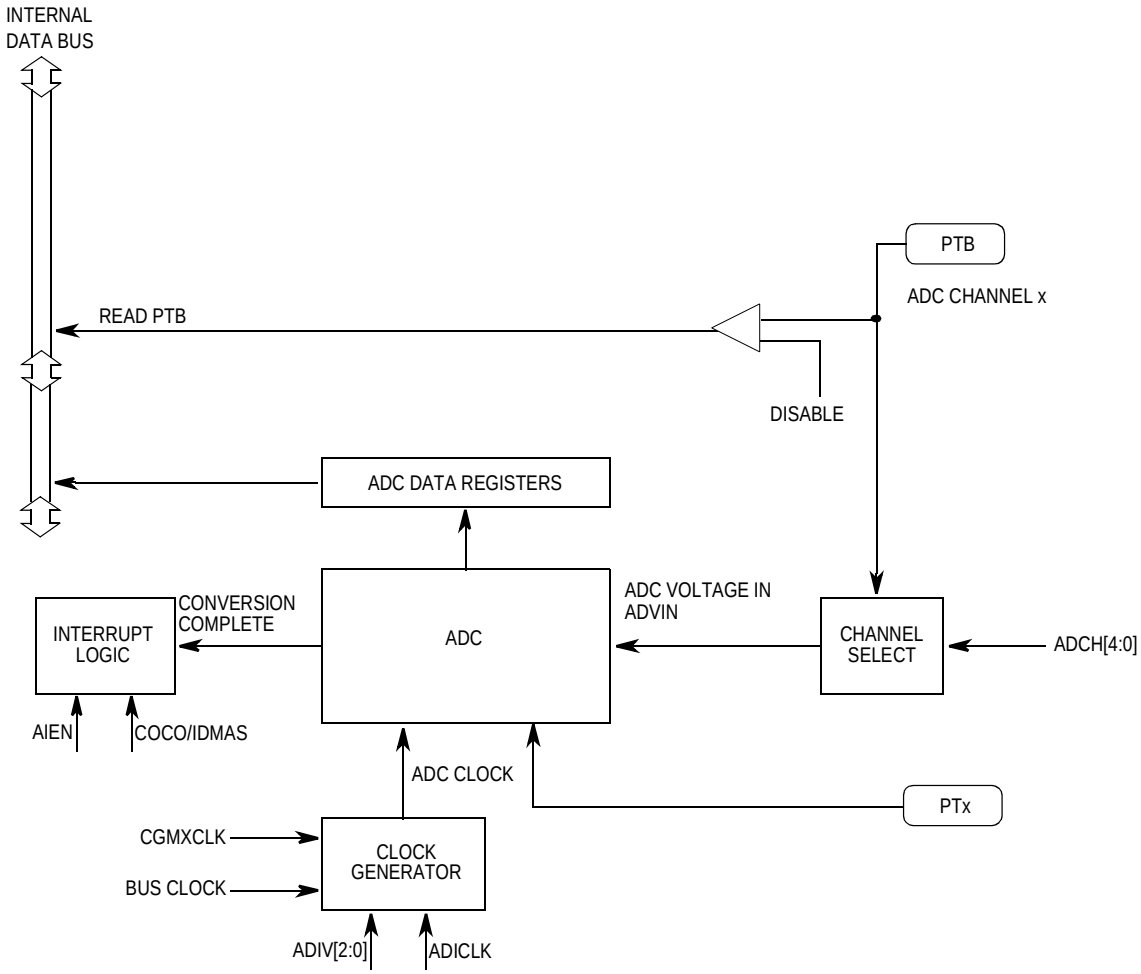


Figure 21-1. ADC Block Diagram

21.4.1 ADC Port I/O Pins

PTB7:PTB0 are general-purpose I/O pins that are shared with the ADC channels. See **Section 22. Input/Output (I/O) Ports**.

The channel select bits define which ADC channel/port pin will be used as the input signal. The ADC overrides the port logic when that port is selected by the ADC multiplexer. The remaining ADC channels/port pins are controlled by the port logic and can be used as general-purpose input/output (I/O) pins. Writes to the port register or DDR will not have any effect on the port pin that is selected by the ADC. Read of a port pin which is in use by the ADC will return a logic 0.

21.4.2 Voltage Conversion

When the input voltage to the ADC equals V_{REFH}, the ADC converts the signal to $3FF (full scale). If the input voltage equals V_{REFL}, the ADC converts it to $000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. All other input voltages will result in $3FF if greater than V_{REFH} and $000 if less than V_{REFL}.

NOTE: *Input voltage should not exceed the analog supply voltages. See* ***23.11 Analog-to-Digital Converter (ADC) Characteristics****.*

21.4.3 Conversion Time

Conversion starts after a write to the ADSCR. A conversion is between 16 and 17 ADC clock cycles, therefore:

$$\text{Conversion time} = \frac{\text{16 to17 ADC Cycles}}{\text{ADC Frequency}}$$

Number of Bus Cycles = Conversion Time x Bus Frequency

The ADC conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is either the bus clock or CGMXCLK and is selectable by ADICLK located in the ADC clock register. For example, if CGMXCLK is 4 MHz and is selected as the ADC input clock source, the ADC input clock divide-by-2 prescale is selected and the bus frequency is 8 MHz:

$$\text{Conversion Time} = \frac{\text{16 to17 ADC Cycles}}{\text{4 MHz/2}} = 8 \text{ to } 8.5\ \mu s$$

Number of bus cycles = (8 to 8.5µs) x 8 MHz = 64 to 68 cycles

NOTE: *The ADC frequency must be between f_{ADIC} minimum and f_{ADIC} maximum to meet A/D specifications. See* ***23.11 Analog-to-Digital Converter (ADC) Characteristics****.*

Since an ADC cycle may be comprised of several bus cycles (four in the previous example) and the start of a conversion is initiated by a bus cycle write to the ADSCR, from zero to four additional bus cycles may occur

before the start of the initial ADC cycle. This results in a fractional ADC cycle and is represented as the 17th cycle.

21.4.4 Continuous Conversion

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

21.4.5 Result Justification

The conversion result may be formatted in four different ways:

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

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

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

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

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

NOTE: *Quantization error is affected when only the most significant eight bits are used as a result. See Figure 21-2.*

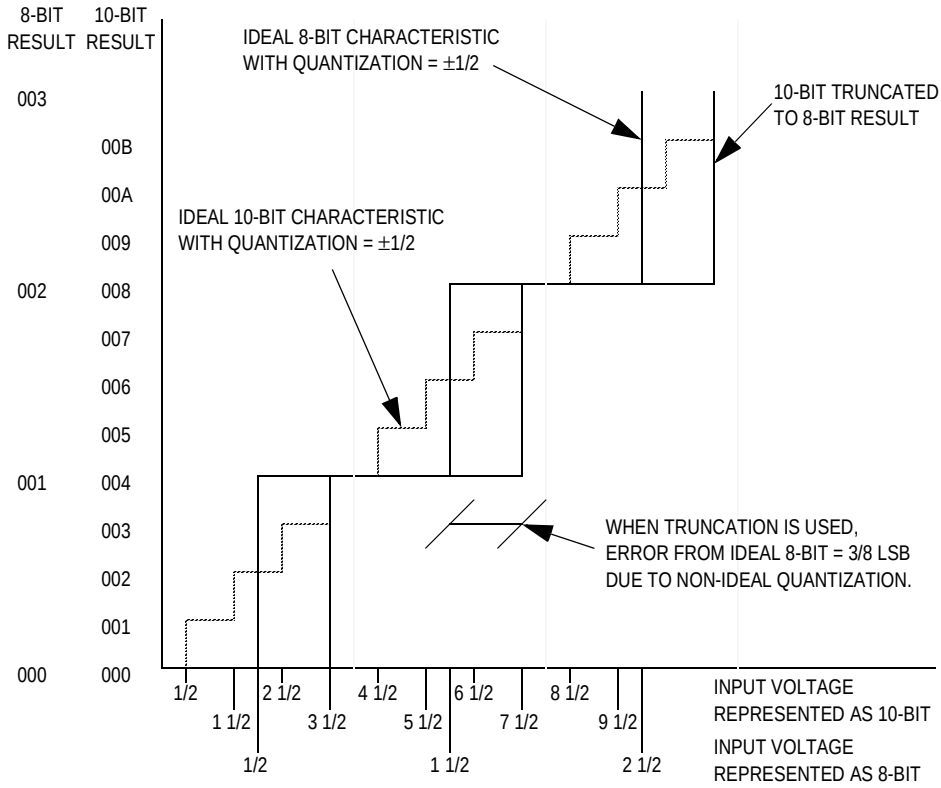


Figure 21-2. 8-Bit Truncation Mode Error

21.4.6 Monotonicity

The conversion process is monotonic and has no missing codes.

21.5 Interrupts

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

21.6 Wait Mode

The WAIT instruction can put the MCU in low power-consumption standby mode.

The ADC continues normal operation during wait mode. Any enabled CPU interrupt request from the ADC can bring the MCU out of wait mode. If the ADC is not required to bring the MCU out of wait mode, power down the ADC by setting ADCH[4:0] in the ADC status and control register before executing the WAIT instruction.

21.7 I/O Signals

The ADC module has 8 input signals.

21.7.1 ADC Analog Power Pin (V_{DDA})

The ADC analog portion uses V_{DDA} as its power pin. 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: *Route V_{DDA} carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.*

21.7.2 ADC Analog Ground Pin (V_{SSA})

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

21.7.3 ADC Voltage Reference Pin (V_{REFH})

V_{REFH} is the power supply for setting the reference voltage V_{REFH}. Connect the V_{REFH} pin to the same voltage potential as V_{DDA}. There will be a finite current associated with V_{REFH}. See **Section 23. Electrical Specifications**.

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

21.7.4 ADC Voltage Reference Low Pin (V_{REFL})

V_{REFL} is the lower reference supply for the ADC. Connect the V_{REFL} pin to the same voltage potential as V_{SSA}. A finite current will be associated with V_{REFL}. See **Section 23. Electrical Specifications**.

21.7.5 ADC Voltage In (ADVIN)

ADVIN is the input voltage signal from one of the 8 ADC channels to the ADC module.

21.7.6 ADC External Connections

This section describes the ADC external connections: V_{REFH} and V_{REFL}, ANx, and grounding.

21.7.6.1 V_{REFH} and V_{REFL}

Both ac and dc current are drawn through the V_{REFH} and V_{REFL} loop. The AC current is in the form of current spikes required to supply charge to the capacitor array at each successive approximation step. The current flows through the internal resistor string. The best external

component to meet both these current demands is a capacitor in the 0.01 μF to 1 μF range 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 dc current will cause a voltage drop which could result in conversion errors.

21.7.6.2 ANx

Empirical data shows that capacitors from the analog inputs to V_{REFL} improve ADC performance. 0.01-μF and 0.1-μF capacitors with good high-frequency characteristics are sufficient. These capacitors must be placed as close as possible to the package pins.

21.7.6.3 Grounding

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. Connect the V_{REFL} pin to the same potential as V_{SSA} at the single point ground location.

21.8 I/O Registers

These I/O registers control and monitor operation of the ADC:

- ADC status and control register, ADSCR
- ADC data registers, ADRH and ARDL
- ADC clock register, ADCLK

21.8.1 ADC Status and Control Register

This section describes the function of the ADC status and control register (ADSCR). Writing ADSCR aborts the current conversion and initiates a new conversion.

Address: $003C

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

Figure 21-3. ADC Status and Control Register (ADSCR)

COCO — Conversions Complete Bit

When AIEN bit is a logic 0, the COCO is a read-only bit which is set each time a conversion is completed except in the continuous conversion mode where it is set after the first conversion. This bit is cleared whenever the ADC status and control register is written or whenever the ADC data register is read.

If AIEN bit is a logic 1, the COCO is a read/write bit. Reset clears this bit.

1 = Conversion completed (AIEN = 0)
0 = Conversion not completed (AIEN = 0)/CPU interrupt (AIEN = 1)

AIEN — ADC Interrupt Enable Bit

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

1 = ADC interrupt enabled
0 = ADC interrupt disabled

ADCO — ADC Continuous Conversion Bit

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

1 = Continuous ADC conversion
0 = One ADC conversion

ADCH[4:0] — ADC Channel Select Bits

ADCH4, ADCH3, ADCH2, ADCH1, and ADCH0 form a 5-bit field which is used to select one of 8 ADC channels. The ADC channels are detailed in Table 21-1.

NOTE: *Take care to prevent switching noise from corrupting the analog signal when simultaneously using a port pin as both an analog and digital input.*

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

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

The voltage levels supplied from internal reference nodes as specified in Table 21-1 are used to verify the operation of the ADC both in production test and for user applications.

Table 21-1. Mux Channel Select

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	Input Select
0	0	0	0	0	PTB0
0	0	0	0	1	PTB1
0	0	0	1	0	PTB2
0	0	0	1	1	PTB3
0	0	1	0	0	PTB4
0	0	1	0	1	PTB5
0	0	1	1	0	PTB6
0	0	1	1	1	PTB7
0	1	0	0	0	Unused *
to					
1	1	0	1	0	
1	1	0	1	1	Reserved **
1	1	1	0	0	Unused *
1	1	1	0	1	V_{REFH}
1	1	1	1	0	V_{REFL}
1	1	1	1	1	ADC power off

* If any unused channels are selected, the resulting ADC conversion will be unknown.
** Used for factory testing.

21.8.2 ADC Data Register High (ADRH) and Data Register Low (ADRL)

21.8.2.1 Left Justified Mode

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

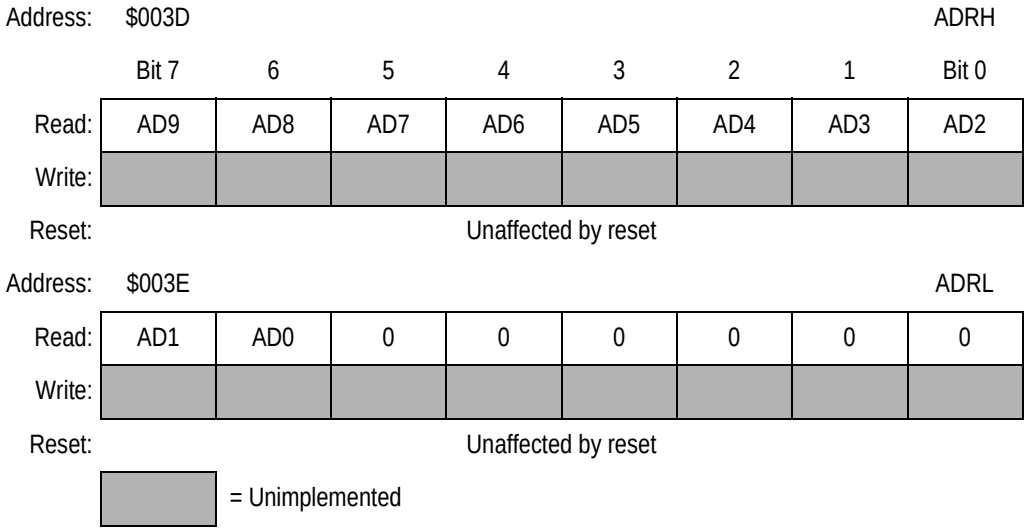


Figure 21-4. ADC Data Register High (ADRH) and Low (ADRL)

21.8.2.2 Right Justified Mode

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

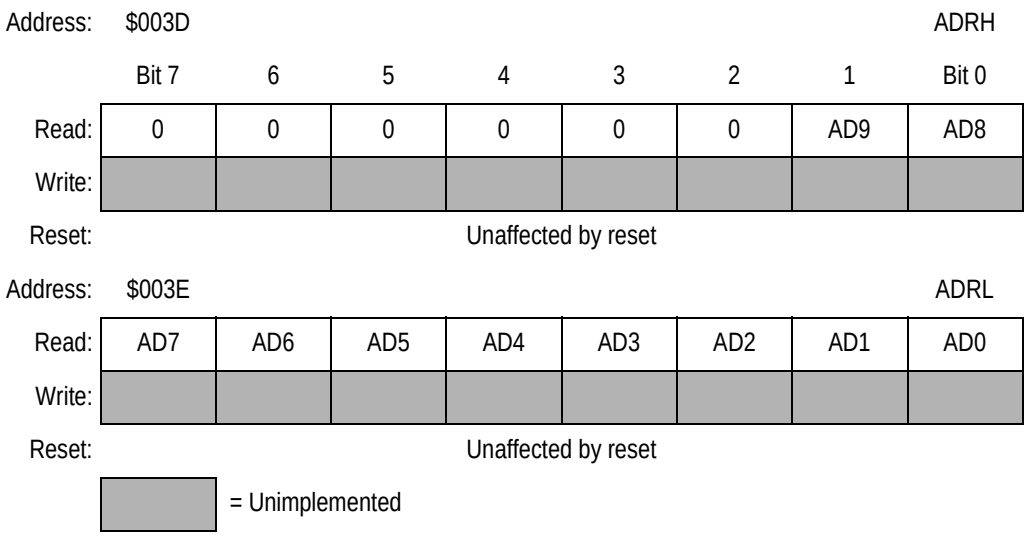


Figure 21-5. ADC Data Register High (ADRH) and Low (ADRL)

21.8.2.3 Left Justified Signed Data Mode

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

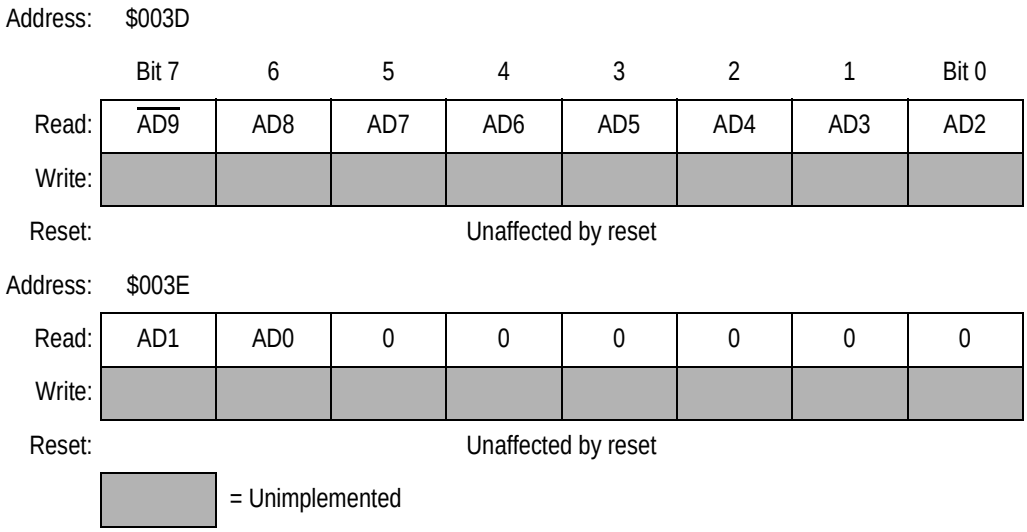


Figure 21-6. ADC Data Register High (ADRH) and Low (ADRL)

21.8.2.4 Eight Bit Truncation Mode

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

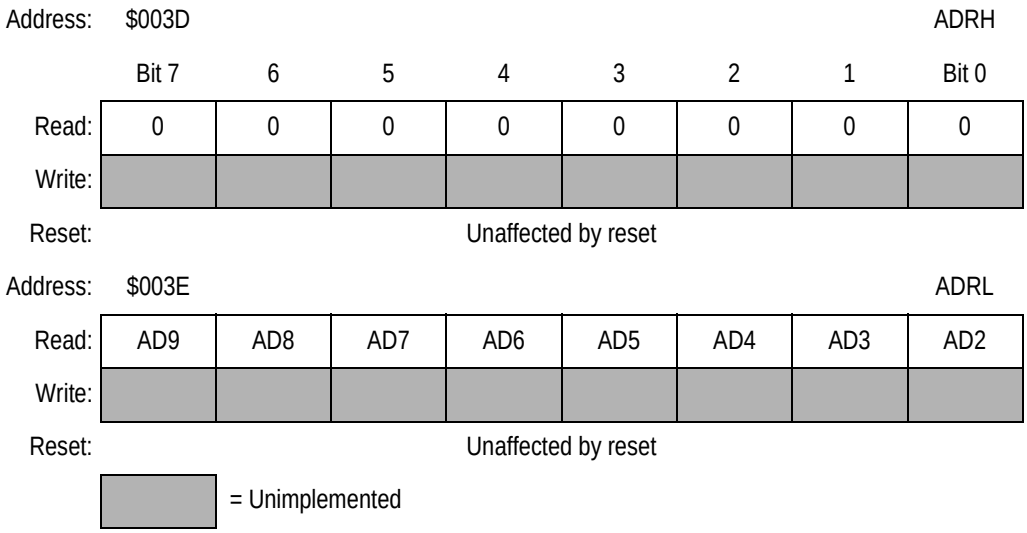


Figure 21-7. ADC Data Register High (ADRH) and Low (ADRL)

21.8.3 ADC Clock Register

This register selects the clock frequency for the ADC, selecting between modes of operation.

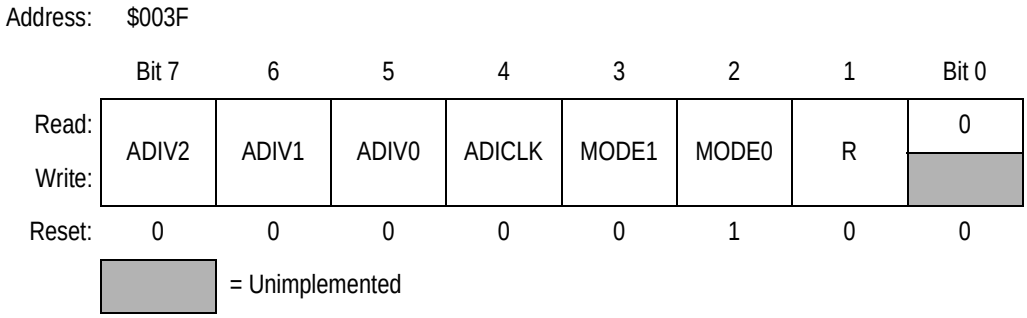


Figure 21-8. ADC Clock Register (ADCLK)

ADIV2:ADIV0 — ADC Clock Prescaler Bits

ADIV2, ADIV1, and ADIV0 form a 3-bit field which selects the divide ratio used by the ADC to generate the internal ADC clock. Table 21-2 shows the available clock configurations.

Table 21-2. ADC Clock Divide Ratio

ADIV2	ADIV1	ADIV0	ADC Clock Rate
0	0	0	ADC input clock /1
0	0	1	ADC input clock /2
0	1	0	ADC input clock /4
0	1	1	ADC input clock /8
1	X	X	ADC input clock /16

X = don’t care

ADICLK — ADC Input Clock Select Bit

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

If the external clock (CGMXCLK) is equal to or greater than 1 MHz, CGMXCLK can be used as the clock source for the ADC. If CGMXCLK is less than 1 MHz, use the PLL-generated bus clock as the clock source. As long as the internal ADC clock is at f_{ADIC}, correct operation can be guaranteed. See **23.11 Analog-to-Digital Converter (ADC) Characteristics**.

1 = Internal bus clock
0 = External clock, CGMXCLK

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

MODE1:MODE0 — Modes of Result Justification Bits

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

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

Section 22. Input/Output (I/O) Ports

22.1 Contents

22.2 Introduction

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

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA) See 363.	Read:	0	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB) See 366.	Read:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Write:								
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC) See 369.	Read:	0	0	0	PTC4	PTC3	PTC2	PTC1	PTC0
		Write:								
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD) See 372.	Read:	0	0	0	0	0	0	PTD1	PTD0
		Write:								
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA) See 364.	Read:	0	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB) See 367.	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC) See 370.	Read:	MCLKEN	0	0	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD) See 373.	Read:	0	0	0	0	0	0	DDRD1	DDRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE) See 375.	Read:	0	0	0	0	0	0	PTE1	PTE0
		Write:								
		Reset:	Unaffected by reset							

= Unimplemented

Figure 22-1. MC68HC908EY16 I/O Port Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$000A	Data Direction Register E (DDRE) See 376.	Read:	0	0	0	0	0	0	DDRE1	DDRE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000B	BEMF Register (BEMF) See 326.	Read:	BEMF7	BEMF6	BEMF5	BEMF4	BEMF3	BEMF2	BEMF1	BEMF0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 22-1. MC68HC908EY16 I/O Port Register Summary (Continued)

22.3 Port A

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

22.3.1 Port A Data Register

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

Address: $0000

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

= Unimplemented

Figure 22-2. Port A Data Register (PTA)

PTA[6:0] — Port A Data Bits

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

22.3.2 Data Direction Register A

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

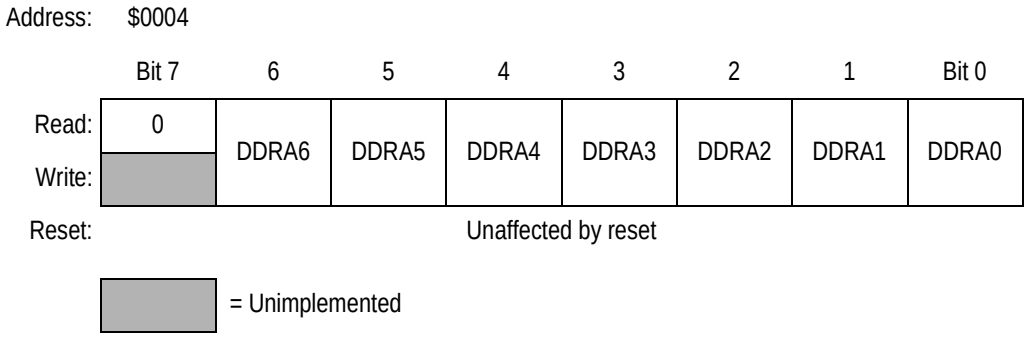


Figure 22-3. Data Direction Register A (DDRA)

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

These read/write bits control port A data direction. Reset clears DDRA[6:0], configuring all port A pins as inputs.

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

NOTE: *Avoid glitches on port A pins by writing to the port A data register before changing data direction register A bits from 0 to 1.*

Figure 22-4 shows the port A I/O logic.

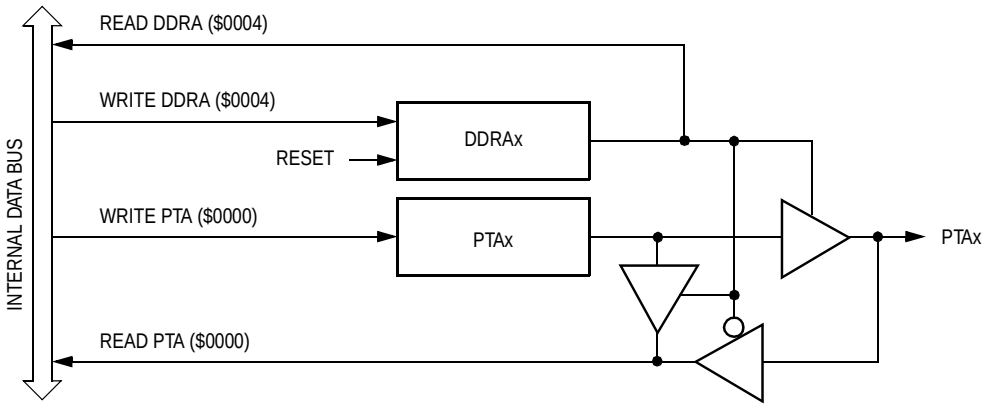


Figure 22-4. Port A I/O Circuit

When bit DDRAx is a logic 1, reading address $0000 reads the PTAx data latch. When bit DDRAx is a logic 0, reading address $0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-1 summarizes the operation of the port A pins.

Table 22-1. Port A Pin Functions

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

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

22.4 Port B

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

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

22.4.1 Port B Data Register

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

Address: $0001

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
Reset:	Unaffected by reset							
Alternate Functions:	ATD7	ATD6	ATD5	ATD4	ATD3	ATD2	ATD1	ATD0
Alternate Function:	TBCH1	TBCH0						

Figure 22-5. 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.

ATD[7:0] — ADC Channels

PTB7–PTB0 are eight of the 15 analog-to-digital converter channels. The ADC channel select bits, CH[4:0], determine whether the PTB7–PTB0 pins are ADC channels or general-purpose I/O pins. If an ADC channel is selected and a read of this corresponding bit in the port B data register occurs, the data will be 0 if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch (see **Section 21. Analog-to-Digital Converter (ADC) Module**). Data direction register B (DDRB) does not affect the data direction of port B pins that are being used by the ADC. However, the DDRB bits always determine whether reading port B returns to the states of the latches or logic 0.

TBCH[1:0] — Timer Channel I/O Bits

The PTB7/TBCH1–PTB6/TBCH0 pins are the TIMB input capture/output compare pins. The edge/level select bits, ELSxB–ELSxA, determine whether the PTB7/TBCH1–PTB6/TBCH0 pins are timer channel I/O pins or general-purpose I/O pins. (See **17.9.1 TIMB Status and Control Register**.)

22.4.2 Data Direction Register B

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

Address: $0005

	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 22-6. 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 22-7 shows the port B I/O logic.

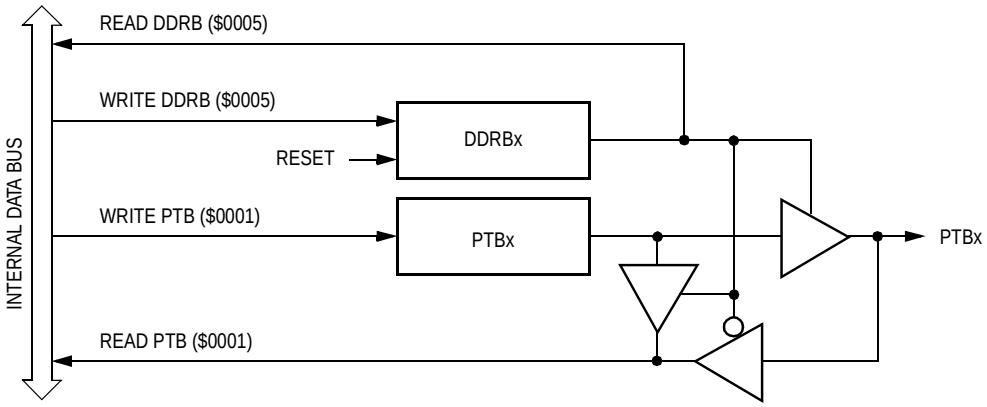


Figure 22-7. Port B I/O Circuit

When bit DDRBx is a logic 1, reading address $0001 reads the PTBx data latch. When bit DDRBx is a logic 0, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-2 summarizes the operation of the port B pins.

Table 22-2. Port B Pin Functions

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

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

22.5 Port C

Port C is an 5-bit general-purpose bidirectional I/O port that shares pin functions with the ICG and SPI modules.

22.5.1 Port C Data Register

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

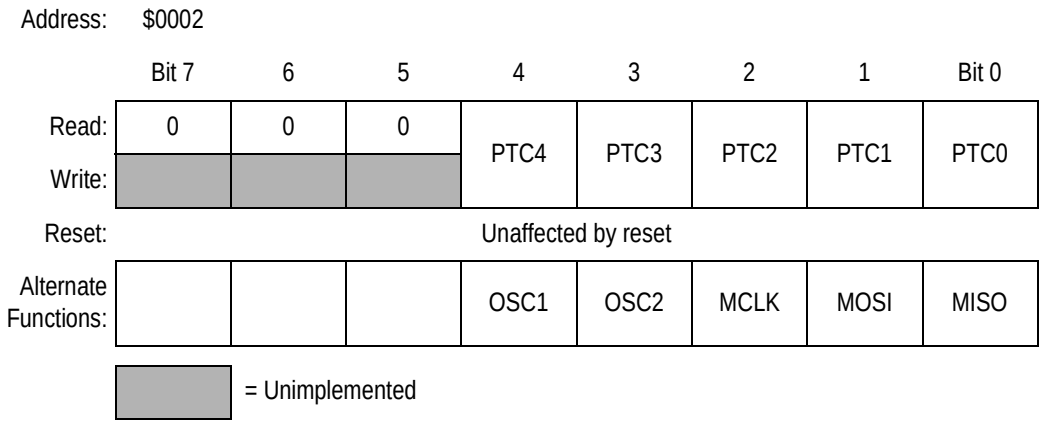


Figure 22-8. Port C Data Register (PTC)

PTC[4:0] — Port C Data Bits

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

MCLK — T12 System Clock Bit

The system clock is driven out of PTC2 when enabled by MCLKEN bit in PTCDDR7.

22.5.2 Data Direction Register C

Data direction register C determines whether each port C pin is an input or an output. Writing a logic 1 to a DDRC bit enables the output buffer for the corresponding port C pin; a logic 0 disables the output buffer.

Address: $0006

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	MCLKEN	0	0	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 22-9. Data Direction Register C (DDRC)

MCLKEN — MCLK Enable Bit

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

1 = MCLK output enabled
0 = MCLK output disabled

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

These read/write bits control port C data direction. Reset clears DDRC[4:0] and MCLKEN, configuring all port C pins as inputs.

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

NOTE: *Avoid glitches on port C pins by writing to the port C data register before changing data direction register C bits from 0 to 1.*

Figure 22-10 shows the port C I/O logic.

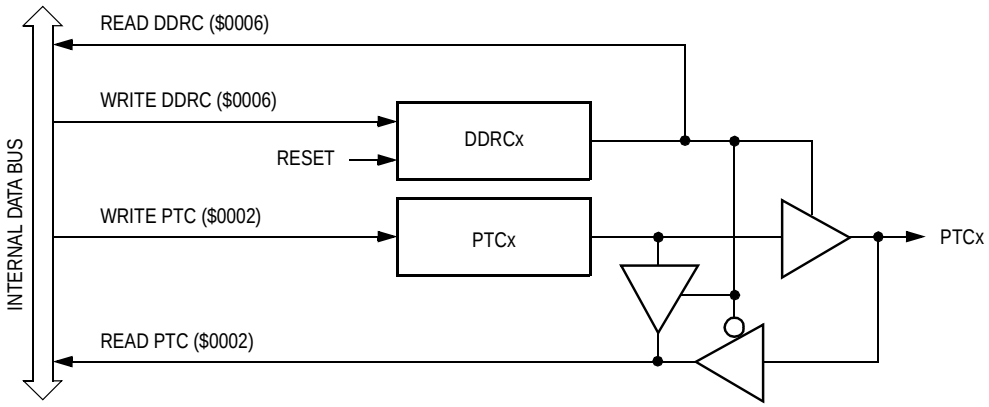


Figure 22-10. Port C I/O Circuit

When bit DDRCx is a logic 1, reading address $0002 reads the PTCx data latch. When bit DDRCx is a logic 0, reading address $0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-3 summarizes the operation of the port C pins.

Table 22-3. Port C Pin Functions

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

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

22.6 Port D

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

22.6.1 Port D Data Register

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

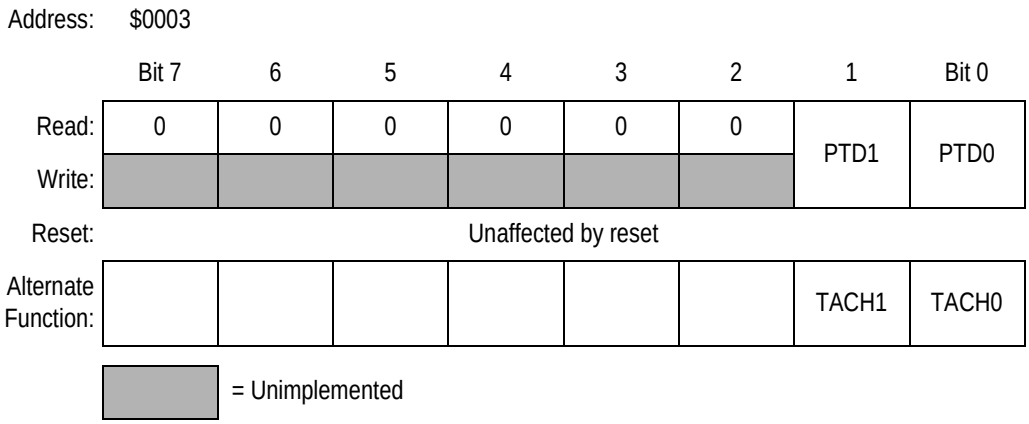


Figure 22-11. Port D Data Register (PTD)

PTD[1:0] — Port D Data Bits

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

TACH[1:0] — Timer Channel I/O Bits

The PTD1/TACH1–PTD0/TACH0 pins are the TIMA input capture/output compare pins. The edge/level select bits, ELSxB–ELSxA, determine whether the PTD1/TACH1–PTD0/TACH0 pins are timer channel I/O pins or general-purpose I/O pins. (See **16.9.1 TIMA Status and Control Register**.)

NOTE: *Data direction register D (DDRD) does not affect the data direction of port D pins that are being used by the TIMA. However, the DDRD bits always determine whether reading port D returns the states of the latches or the states of the pins. (See **Table 22-4**.)*

22.6.2 Data Direction Register D

Data direction register D determines whether each port D pin is an input or an output. Writing a logic 1 to a DDRD bit enables the output buffer for the corresponding port D pin; a logic 0 disables the output buffer.

Address: $0007

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	DDRD1	DDRD0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 22-12. Data Direction Register D (DDRD)

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

These read/write bits control port D data direction. Reset clears DDRD[1:0], configuring all port D pins as inputs.

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

NOTE: *Avoid glitches on port D pins by writing to the port D data register before changing data direction register D bits from 0 to 1.*

Figure 22-13 shows the port D I/O logic.

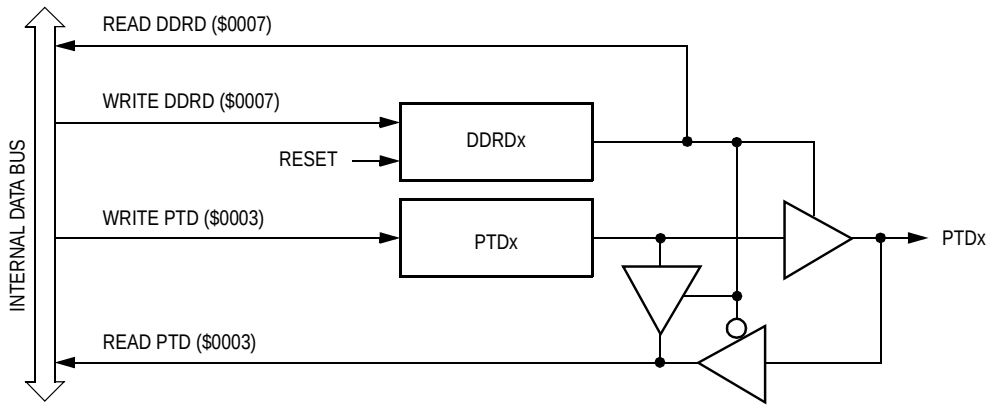


Figure 22-13. Port D I/O Circuit

When bit DDRDx is a logic 1, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic 0, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-4 summarizes the operation of the port D pins.

Table 22-4. Port D Pin Functions

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

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

22.7 Port E

Port E is a 2-bit special function port that shares its pins with the Enhanced Serial Communications Interface module (ESCI).

22.7.1 Port E Data Register

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

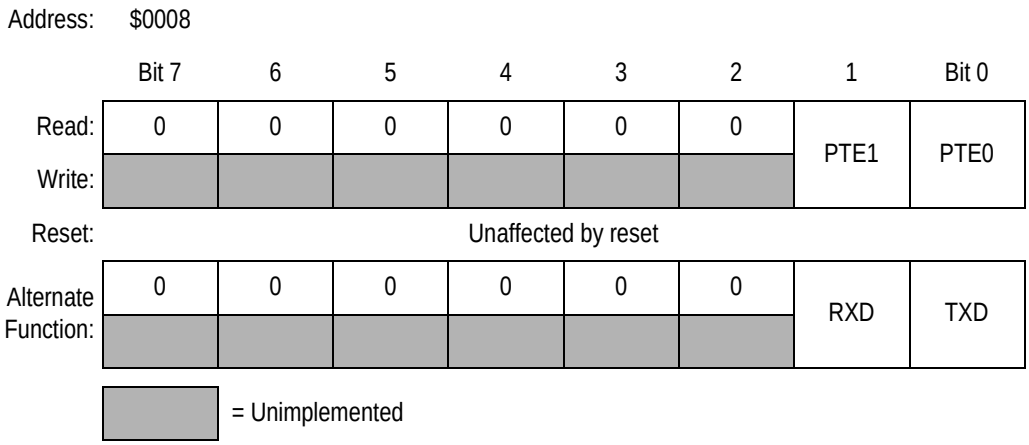


Figure 22-14. Port E Data Register (PTE)

PTE[1:0] — Port E Data Bits

These read/write bits are software programmable. Data direction of each port E pin is under the control of the corresponding bit in data direction register E. Reset has no effect on PTE[1:0].

RxD — SCI Receive Data Input Bit

The PTE1/RxD pin is the receive data input for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE1/RxD pin is available for general-purpose I/O. See **14.9.1 ESCI Control Register 1**.

TxD — SCI Transmit Data Output

The PTE0/TxD pin is the transmit data output for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE0/TxD pin is available for general-purpose I/O. See **14.9.1 ESCI Control Register 1**.

NOTE: *Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the ESCI. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See **Table 22-5**.)*

22.7.2 Data Direction Register E

Data direction register E determines whether each port E pin is an input or an output. Writing a logic 1 to a DDRE bit enables the output buffer for the corresponding port E pin; a logic 0 disables the output buffer.

Address: $000A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	DDRE1	DDRE0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 22-15. Data Direction Register E (DDRE)

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

These read/write bits control port E data direction. Reset clears DDRE[1:0], configuring all port E pins as inputs.

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

NOTE: *Avoid glitches on port E pins by writing to the port E data register before changing data direction register E bits from 0 to 1.*

Figure 22-16 shows the port E I/O logic.

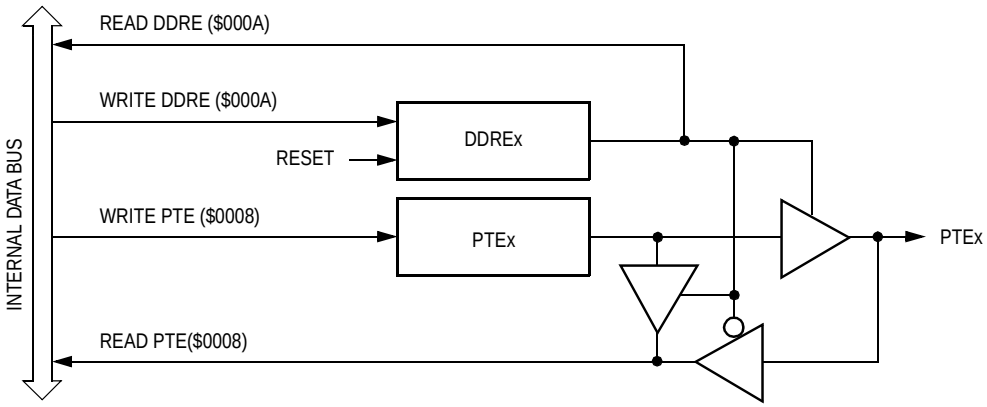


Figure 22-16. Port E I/O Circuit

When bit DDREx is a logic 1, reading address $0008 reads the PTEx data latch. When bit DDREx is a logic 0, reading address $0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-5 summarizes the operation of the port E pins.

Table 22-5. Port E Pin Functions

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

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

Input/Output (I/O) Ports

Section 23. Electrical Specifications

23.1 Contents

23.2 Introduction

This section contains preliminary electrical and timing specifications. These values are design targets and have not yet been fully tested.

23.3 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* ***23.6 DC Electrical Characteristics*** *for guaranteed operating conditions.*

Characteristic[1]	Symbol	Value	Unit
Supply voltage	V_{DD}	–0.3 to +6.0	V
Input voltage	V_{In}	V_{SS} –0.3 to V_{DD} +0.3	V
Maximum current per pin excluding V_{DD}, V_{SS}, and PTA0–PTA6 and PTC0-PTC1	I	±15	mA
Maximum current for pins PTA0–PTA6 and PTC0-PTC1	I_{PTA0}–I_{PTA6}, I_{PTC0}–I_{PTC1}	±25	mA
Maximum current out of V_{SS}	I_{MVSS}	100	mA
Maximum current into V_{DD}	I_{MVDD}	100	mA
Storage temperature	T_{STG}	–55 to +150	°C

1. Voltages referenced to V_{SS}

NOTE: *This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. For proper operation, it is recommended that V_{In} and V_{Out} be constrained to the range $V_{SS} \le (V_{In}$ or $V_{Out}) \le V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either V_{SS} or V_{DD}).*

23.4 Functional Operating Range

Characteristic	Symbol	Value	Unit
Operating temperature range	T_A	–40 to 125	°C
Operating voltage range	V_{DD}	5.0 ± 10%	V

23.5 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance QFP (32 pins)	θ_{JA}	100	C/W
I/O pin power dissipation	$P_{I/O}$	User determined	W
Power dissipation(1)	P_D	$P_D = (I_{DD} \times V_{DD}) + P_{I/O} = K/(T_J + 273°C)$	W
Constant(2)	K	$P_D \times (T_A + 273°C) + P_D^2 \times \theta_{JA}$	W/C
Average junction temperature	T_J	$T_A + (P_D \times \theta_{JA})$	°C

1. Power dissipation is a function of temperature.
2. K is a constant unique to the device. K can be determined for a known T_A and measured P_D. With this value of K, P_D and T_J can be determined for any value of T_A.

23.6 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} = –5.0 mA, all I/O pins I_{Load} = –10.0 mA, all I/O pins I_{Load} = –15.0 mA, PTA0–PTA6/$\overline{SS}$ and PTC0–PTC1 only	V_{OH}	 V_{DD} –0.7 V_{DD} –1.1 V_{DD} –1.7 V_{DD} –1.5	 V_{DD} –0.54 V_{DD} –0.91 V_{DD} –1.51 V_{DD} –0.81	 — — — —	V
Output low voltage I_{Load} = 1.6 mA, all I/O pins I_{Load} = 5.0 mA, all I/O pins I_{Load} = 10.0 mA, all I/O pins I_{Load} = 15.0 mA, PTA0–PTA6/$\overline{SS}$ and PTC0–PTC1 only	V_{OL}	 — — — —	 0.31 0.56 0.99 1.44	 0.4 1 1.5 1.8	V
Input high voltage — all ports, $\overline{IRQ}$, $\overline{RST}$	V_{IH}	0.7 x V_{DD}	—	V_{DD} + 0.3	V
Input low voltage — all ports, $\overline{IRQ}$, $\overline{RST}$	V_{IL}	V_{SS}	—	0.3 x V_{DD}	V
dc injection current, all ports[3]	I_{INJ}	– 2.0	—	+ 2.0	mA
Total dc current injection (sum of all I/O)	I_{INJTOT}	– 25		+ 25	mA
V_{DD} + V_{DDA} supply current Run[4],[5] Wait[4], [6] Stop (LVI off) @ 25°C[4], [7] Stop (LVI on) @ 25°C Stop (LVI off), –40°C to 125°C Stop (LVI on), –40°C to 125°C	I_{DD}	 — — — — — —	 18 5.2 0.83 0.19 3.0 0.19	 25 7.0 2.00 0.24 30 0.30	 mA mA µA mA µA mA
I/O ports Hi-Z leakage current[8]	I_{IL}	–10	—	+10	µA
Input current – RESET, OSC1	I_{In}	–1	—	+1	µA
Capacitance Ports (as input or output)	C_{Out} C_{In}	— —	— —	12 8	pF
POR rearm voltage[9]	V_{POR}	0	—	100	mV
POR reset voltage[10]	V_{POR}	0	700	800	mV
POR rise time ramp rate	R_{POR}	0.035	—	—	V/ms
Monitor mode entry voltage	V_{TST}	V_{DD}+ 3.5		V_{DD}+ 4.5	V

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Low-voltage inhibit reset, trip falling voltage[11]	V_{TRIPF}	3.90	4.30	4.50	V
Low-voltage inhibit reset, trip rising voltage[12]	V_{TRIPR}	4.00	4.40	4.60	V
Low-voltage inhibit reset/recover hysteresis[13]	V_{HYS}	—	0.09	—	V
Pullup resistor — PTA0–PTA6/$\overline{SS}$[14], $\overline{IRQ}$, $\overline{RST}$	R_{PU}	24	—	48	kΩ

1. V_{DD} = 5.5 Vdc to 4.5 Vdc, V_{SS} = 0 Vdc, T_A = –40°C to +125°C, unless otherwise noted
2. Typical values reflect average measurements at midpoint of voltage range, 25°C only.
3. Some disturbance of the ADC accuracy is possible during any injection event and is dependent on board layout and power supply decoupling.
4. Run (operating) I_{DD} measured using internal oscillator at its 32-MHz rate. V_{DD} = 5.5 Vdc. All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. All ports configured as inputs. Measured with all modules enabled.
5. All measurements taken with LVI enabled.
6. Wait I_{DD} measured using internal oscillator at its 1-MHz rate. All inputs 0.2 V from rail; no dc loads; less than 100 pF on all outputs. All ports configured as inputs.
7. Stop I_{DD} is measured with no port pin sourcing current; all modules are disabled. OSCSTOPEN option is not selected.
8. Pullups and pulldowns are disabled.
9. Maximum is highest voltage that power-on reset (POR) is guaranteed.
10. Maximum is highest voltage that POR is possible.
11. These values assume the LVI is operating in 5V mode (i.e. LVI5OR3 bit is set to 1). For 3V mode (LVI5OR3=0), values become Min: 2.45, Typ: 2.60, Max: 2.80
12. These values assume the LVI is operating in 5V mode (i.e. LVI5OR3 bit is set to 1). For 3V mode (LVI5OR3=0), values become Min: 2.55, Typ: 2.66, Max: 2.80
13. These values assume the LVI is operating in 5V mode (i.e. LVI5OR3 bit is set to 1). For 3V mode (LVI5OR3=0), values become Typ: 60
14. PTA0–PTA4 pull-up resistors are for interrupts only and are only enabled when the keyboard is in use.

23.7 Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Frequency of operation[2] Crystal option External clock option[3]	f_{osc}	32 dc[4]	100 16	kHz MHz
Internal operating frequency	f_{op}	—	8	MHz
Internal clock period ($1/f_{OP}$)	t_{cyc}	125	—	ns
$\overline{\text{RESET}}$ input pulse width low[5]	t_{IRL}	50	—	ns
$\overline{\text{IRQ}}$ interrupt pulse width low[6] (edge-triggered)	t_{ILIH}	50	—	ns
$\overline{\text{IRQ}}$ interrupt pulse period	t_{ILIL}	Note 8	—	t_{cyc}
16-bit timer[7] Input capture pulse width Input capture period	 t_{TH},t_{TL} t_{TLTL}	 Note 8	 — —	 ns t_{cyc}

Notes:

1. V_{SS} = 0 Vdc; timing shown with respect to 20% V_{DD} and 70% V_{SS} unless otherwise noted.
2. See Internal Clock Generator (ICG) Module for more information.
3. No more than 10% duty cycle deviation from 50%
4. Some modules may require a minimum frequency greater than dc for proper operation. See appropriate table for this information.
5. Minimum pulse width reset is guaranteed to be recognized. It is possible for a smaller pulse width to cause a reset.
6. Minimum pulse width is for guaranteed interrupt. It is possible for a smaller pulse width to be recognized.
7. Minimum pulse width is for guaranteed interrupt. It is possible for a smaller pulse width to be recognized.
8. The minimum period, t_{ILIL} or t_{TLTL}, should not be less than the number of cycles it takes to execute the interrupt service routine plus t_{cyc}.

23.8 Internal Oscillator Characteristics

Characteristic[1]	Symbol	Min	Typ	Max	Unit
Internal oscillator base frequency[2], [3]	f_{INTOSC}	230.4	307.2	384	kHz
Internal oscillator tolerance	f_{OSC_TOL}	–25	—	+25	%
Internal oscillator multiplier[4]	N	1	—	127	—

1. V_{DD} = 4.5 to 5.5 Vdc, V_{SS} = 0 Vdc, T_A = –40°C to +125°C, unless otherwise noted
2. Internal oscillator is selectable through software for a maximum frequency. Actual frequency will be multiplier (N) x base frequency.
3. $f_{Bus} = (f_{INTOSC} / 4) \times N$ when internal clock source selected
4. Multiplier must be chosen to limit the maximum bus frequency of 8 MHz for 4.5-V operation.

23.9 External Oscillator Characteristics

Characteristic[1]	Symbol	Min	Typ	Max	Unit
External clock option[2][3] With ICG clock disabled With ICG clock enabled EXTSLOW = 1[4] EXTSLOW = 0[4]	f_{EXTOSC}	 dc[5] 60 307.2 k	 — — —	 32 M[6] 307.2 k 32 M[6]	Hz
External crystal options[7][8] EXTSLOW = 1[4] EXTSLOW = 0[4]	f_{EXTOSC}	 30 k 1 M	 — —	 100 k 8 M	Hz
Crystal load capacitance[9]	C_L	—	—	—	pF
Crystal fixed capacitance[9]	C_1	—	2 x C_L	—	pF
Crystal tuning capacitance[9]	C_2	—	2 x C_L	—	pF
Feedback bias resistor[9]	R_B	—	10	—	MΩ
Series resistor [9][10]	R_S	—	—	—	MΩ

1. V_{DD} = 4.5 to 5.5 Vdc, V_{SS} = 0 Vdc, T_A = –40°C to +125°C, unless otherwise noted
2. Setting EXTCLKEN configuration option enables OSC1 pin for external clock square-wave input.
3. No more than 10% duty cycle deviation from 50%
4. EXTSLOW configuration option configures external oscillator for a slow speed crystal and sets the clock monitor circuits of the ICG module to expect an external clock frequency that is higher/lower than the internal oscillator base frequency, f_{INTOSC}.
5. Some modules may require a minimum frequency greater than dc for proper operation. See appropriate table for this information.
6. MCU speed derates from 32 MHz at V_{DD} = 4.5 Vdc
7. Setting EXTCLKEN and EXTXTALEN configuration options enables OSC1 and OSC2 pins for external crystal option.
8. $f_{Bus} = (f_{EXTOSC} / 4)$ when external clock source is selected.
9. Consult crystal vendor data sheet, see **Figure 7-2. Internal Clock Generator Block Diagram**.
10. Not required for high-frequency crystals

23.10 Trimmed Accuracy of the Internal Clock Generator

The unadjusted frequency of the low-frequency base clock (IBASE), when the comparators in the frequency comparator indicate zero error, can vary as much as ±25% due to process, temperature, and voltage. The trimming capability exists to compensate for process affects. The remaining variation in frequency is due to temperature, voltage, and change in target frequency (multiply register setting). These affects are designed to be minimal, however variation does occur. Better performance is seen with lower settings of N.

23.10.1 Trimmed Internal Clock Generator Characteristics

Characteristic[1]	Symbol	Min	Typ	Max	Unit
Absolute trimmed internal oscillator tolerance[2],[3] –40°C to 85°C –40°C to 125°C	F_{abs_tol}	 — —	 4.0 5.0	 7.0 10.0	%
Variation over temperature[3], [4]	V_{ar_temp}	—	0.05	0.08	%/°C
Variation over voltage[3], [5] 25°C –40°C to 85°C –40°C to 125°C	V_{ar_volt}	 — — —	 1.0 1.0 1.0	 2.0 2.0 2.0	%/V

1. These specifications concern long-term frequency variation. Each measurement is taken over a 1-ms period.
2. Absolute value of variation in ICG output frequency, trimmed at nominal V_{DD} and temperature, as temperature and V_{DD} are allowed to vary for a single given setting of N.
3. Specification is characterized but not tested.
4. Variation in ICG output frequency for a fixed N and voltage
5. Variation in ICG output frequency for a fixed N

23.11 Analog-to-Digital Converter (ADC) Characteristics

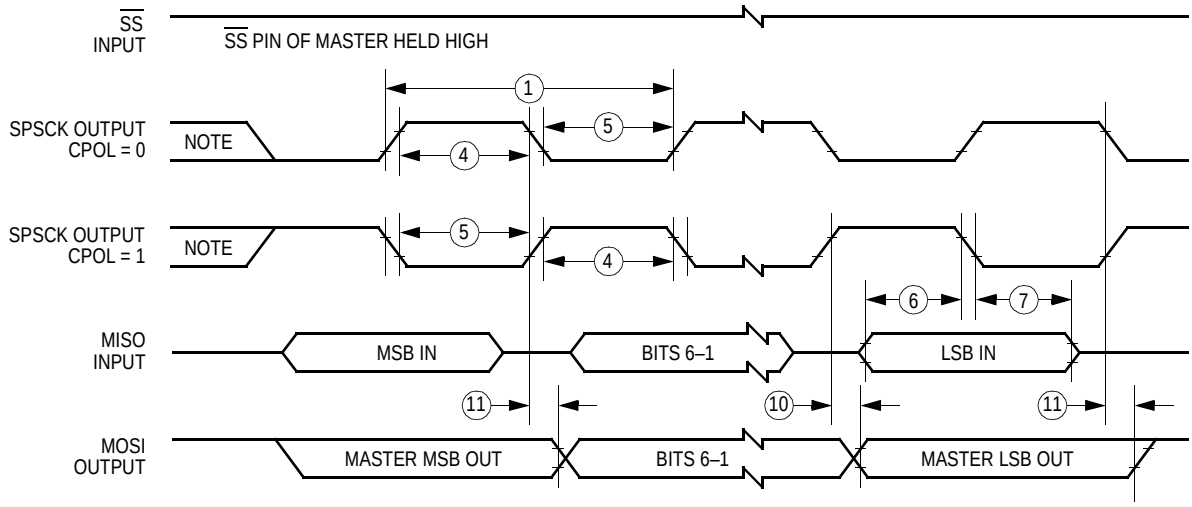
Characteristic	Symbol	Min	Typ	Max	Unit	Notes
Supply voltage	V_{DDA}	4.5	—	5.5	V	V_{DDA} should be tied to the same potential as V_{DD} via separate traces
Input voltages	V_{ADIN}	0	—	V_{DDA}	V	V_{ADIN} <= V_{DDA}
Resolution	B_{AD}	10	—	10	Bits	
Absolute accuracy	A_{AD}	–4	—	+4	LSB	Includes quantization
ADC internal clock	f_{ADIC}	500 k	—	1.048 M	Hz	$t_{AIC} = 1/f_{ADIC}$
Conversion range	R_{AD}	V_{SSA}	—	V_{DDA}	V	
Power-up time	t_{ADPU}	16	—	—	t_{AIC} cycles	
Conversion time	t_{ADC}	16	—	17	t_{AIC} cycles	
Sample time	t_{ADS}	5	—	—	t_{AIC} cycles	
Monotonicity	M_{AD}	Guaranteed				
Zero input reading	Z_{ADI}	000	—	003	Hex	
Full-scale reading	F_{ADI}	3FC	—	3FF	Hex	
Input capacitance	C_{ADI}	—	—	30	pF	Not tested
V_{REFH}/V_{REFL} current	I_{VREF}	—	1.6	—	mA	
Absolute accuracy (8-bit truncated mode)	A_{AD}	–1	—	+1	LSB	Includes quantization
Zero input reading (8-bit truncated mode)	Z_{ADI}	00	—	01	Hex	
Full-scale reading (8-bit truncated mode)	F_{ADI}	FE	—	FF	Hex	
Quantization error (8-bit truncated mode)	—	—	—	+7/8 –1/8	LSB	

23.12 SPI Characteristics

Diagram Number[1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	 $f_{OP(M)}$ $f_{OP(S)}$	 $f_{OP}/128$ DC	 $f_{OP}/2$ f_{OP}	 MHz MHz
1	Cycle time Master Slave	 $t_{CYC(M)}$ $t_{CYC(S)}$	 2 1	 128 —	 t_{cyc} t_{cyc}
2	Enable lead time	$t_{Lead(S)}$	1	—	t_{cyc}
3	Enable lag time	$t_{Lag(S)}$	1	—	t_{cyc}
4	Clock (SPSCK) high time Master Slave	 $t_{SCKH(M)}$ $t_{SCKH(S)}$	 t_{cyc} –25 1/2 t_{cyc} –25	 64 t_{cyc} —	 ns ns
5	Clock (SPSCK) low time Master Slave	 $t_{SCKL(M)}$ $t_{SCKL(S)}$	 t_{cyc} –25 1/2 t_{cyc} –25	 64 t_{cyc} —	 ns ns
6	Data setup time (inputs) Master Slave	 $t_{SU(M)}$ $t_{SU(S)}$	 30 30	 — —	 ns ns
7	Data hold time (inputs) Master Slave	 $t_{H(M)}$ $t_{H(S)}$	 30 30	 — —	 ns ns
8	Access time, slave[3] CPHA = 0 CPHA = 1	 $t_{A(CP0)}$ $t_{A(CP1)}$	 0 0	 40 40	 ns ns
9	Disable time, slave[4]	$t_{DIS(S)}$	—	40	ns
10	Data valid time, after enable edge Master Slave[5]	 $t_{V(M)}$ $t_{V(S)}$	 — —	 50 50	 ns ns
11	Data hold time, outputs, after enable edge Master Slave	 $t_{HO(M)}$ $t_{HO(S)}$	 0 0	 — —	 ns ns

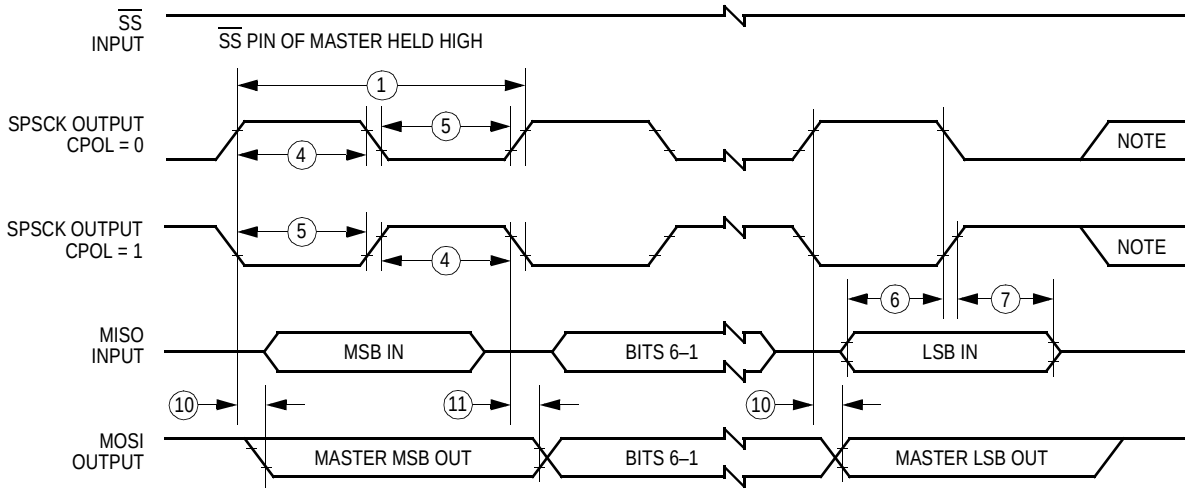
Notes:

1. Numbers refer to dimensions in Figure 23-1 and Figure 23-2.
2. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; 100 pF load on all SPI pins.
3. Time to data active from high-impedance state
4. Hold time to high-impedance state
5. With 100 pF on all SPI pins



Note: This first clock edge is generated internally, but is not seen at the SPSCK pin.

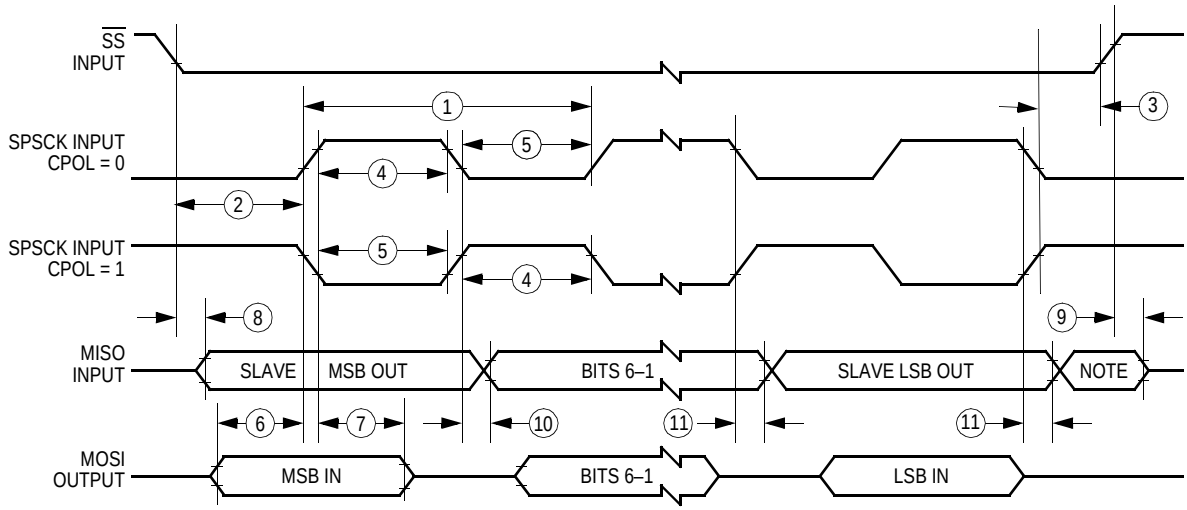
a) SPI Master Timing (CPHA = 0)



Note: This last clock edge is generated internally, but is not seen at the SPSCK pin.

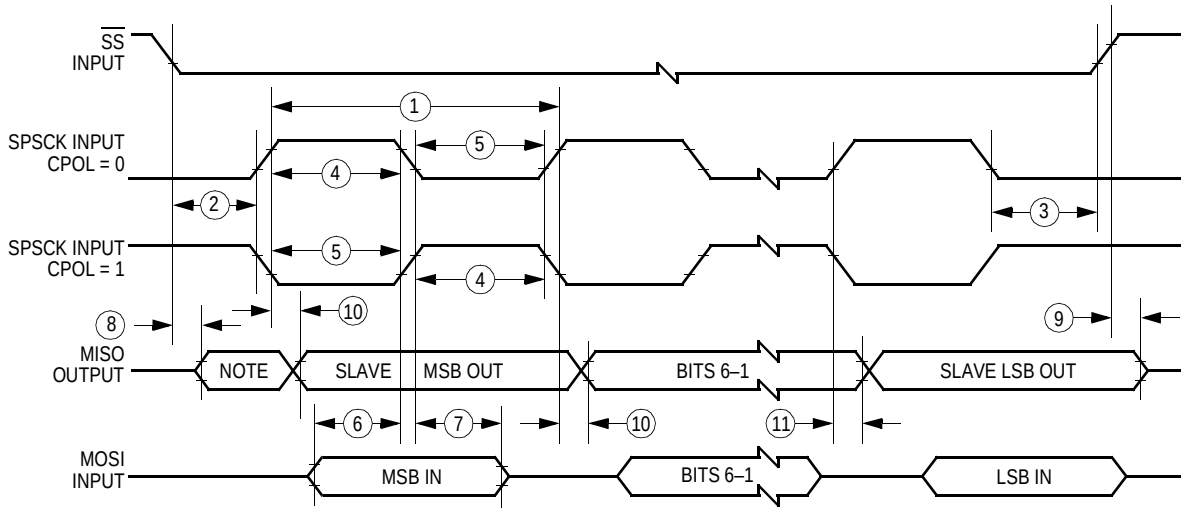
b) SPI Master Timing (CPHA = 1)

Figure 23-1 SPI Master Timing



Note: Not defined but normally MSB of character just received

a) SPI Slave Timing (CPHA = 0)



Note: Not defined but normally LSB of character previously transmitted

b) SPI Slave Timing (CPHA = 1)

Figure 23-2 SPI Slave Timing

23.13 Memory Characteristics

Characteristic	Symbol/ Description	Min	Max	Units
RAM data retention voltage[1]	V_{RDR}	1.3	—	V
FLASH program bus clock frequency	—	1	—	MHz
FLASH read bus clock frequency	f_{Read}[2]	32 k	8 M	Hz
FLASH page erase time	t_{Erase}[3]	4	—	ms
FLASH mass erase time	t_{MErase}[4]	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}[5]	1	—	µs
FLASH cumulative program HV period	t_{HV}[6]	—	4	ms
FLASH row erase endurance[7]	—	10 K	—	Cycles
FLASH row program endurance[6]	—	10 K	—	Cycles
FLASH data retention time[8]	—	10	—	Years

1. Specification is characterized but not tested.
2. f_{Read} is defined as the frequency range for which the FLASH memory can be read.
3. If the page erase time is longer than t_{Erase} (min), there is no erase-disturb, but it reduces the endurance of the FLASH memory.
4. If the mass erase time is longer than t_{MErase} (min), there is no erase-disturb, but it reduces the endurance of the FLASH memory.
5. 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 logic 0.
6. t_{HV} is defined as the cumulative high voltage programming time to the same row before next erase. t_{HV} must satisfy this condition: $t_{NVS} + t_{NVH} + t_{PGS} + (t_{PROG} \times 64) \leq t_{HV}$ max.
7. The minimum row endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many erase/program cycles.
8. The FLASH is guaranteed to retain data over the entire operating temperature range for at least the minimum time specified.

Electrical Specifications

Section 24. Mechanical Specifications

24.1 Contents

24.2 Introduction

This section gives the dimensions for the 32-pin quad flat pack (QFP).

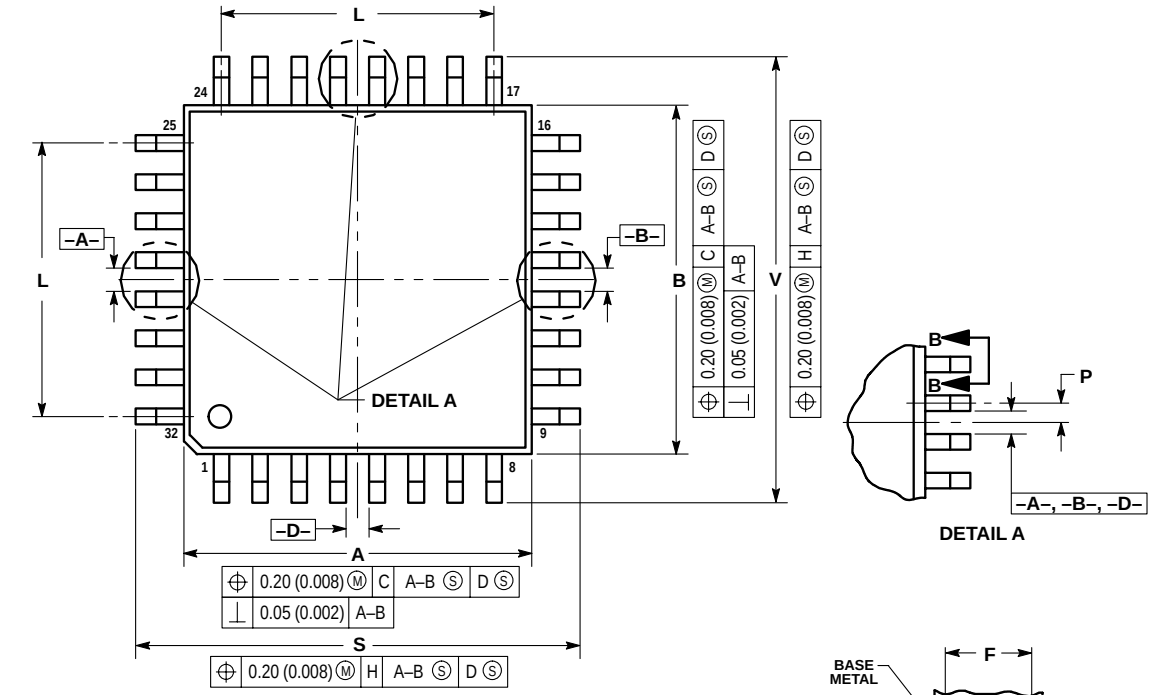
The following figure shows the latest package drawings at the time of this publication. To make sure that you have the latest package specifications, contact one of the following:

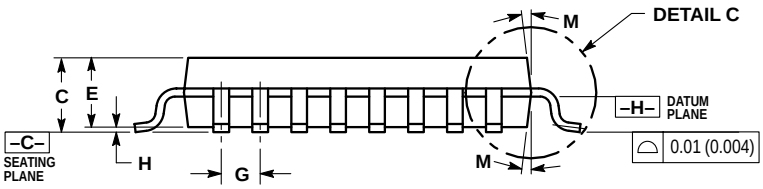
- Local Motorola Sales Office
- Worldwide Web at

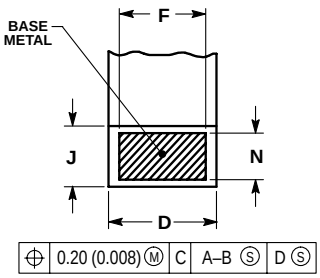
  http://www.motorola.com/semiconductors/

Follow Worldwide Web on-line instructions to retrieve the current mechanical specifications.

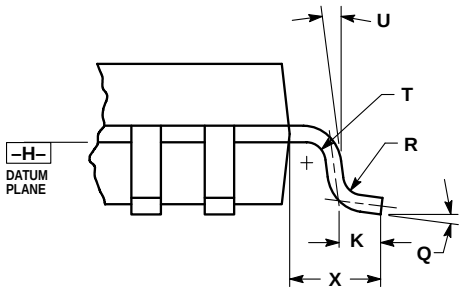
24.3 32-Pin QFP (Case Number 873)







SECTION B–B
VIEW ROTATED 90 ° CLOCKWISE



DETAIL C

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE –H– IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS –A–, –B– AND –D– TO BE DETERMINED AT DATUM PLANE –H–.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE –C–.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 (0.010) PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE –H–.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 (0.003) TOTAL IN EXCESS OF THE D DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	6.95	7.10	0.274	0.280
B	6.95	7.10	0.274	0.280
C	1.40	1.60	0.055	0.063
D	0.273	0.373	0.010	0.015
E	1.30	1.50	0.051	0.059
F	0.273	—	0.010	—
G	0.80 BSC		0.031 BSC	
H	—	0.20	—	0.008
J	0.119	0.197	0.005	0.008
K	0.33	0.57	0.013	0.022
L	5.6 REF		0.220 REF	
M	6 °	8 °	6 °	8 °
N	0.119	0.135	0.005	0.005
P	0.40 BSC		0.016 BSC	
Q	5 °	10°	5 °	10°
R	0.15	0.25	0.006	0.010
S	8.85	9.15	0.348	0.360
T	0.15	0.25	0.006	0.010
U	5 °	11°	5 °	11°
V	8.85	9.15	0.348	0.360
X	1.00 REF		0.039 REF	

Section 25. Ordering Information

25.1 Contents

25.2 Introduction

This section contains ordering numbers for the MC68HC908EY16.

25.3 MC Order Numbers

Table 25-1. MC Order Numbers

MC Order Number[1]	Operating Temperature Range
MC68HC908EY16MFA	–40°C to +125°C
MC68HC908EY16VFA	–40°C to +105°C
MC68HC908EY16CFA	–40°C to +85°C

1. FA = Quad flat pack

Ordering Information

Revision History

Contents

Introduction

This section contains the revision history for the 68HC908EY16 advance information data book.

Revision History

Changes from Rev 3.0 published in November 2002 to Rev 4.0 published in February 2003

Section	Page (in Rev 3.0)	Description of change
Electrical Specifications	382	Updated parameters for output high voltage (V_{OH}), output low voltage (V_{OL}) and supply current (I_{DD})
	383	Updated parameters for low voltage inhibit reset: V_{TRIPF}, V_{TRIPR} and V_{HYS}.
	387	Updated parameters for ADC absolute accuracy, zero input reading, full-scale reading, zero input reading (8-bit truncated mode) and full-scale reading (8-bit truncated mode).

Changes from Rev 2.0 published in May 2002 to Rev 3.0 published in November 2002

Section	Page (in Rev 3.0)	Description of change
Memory Map	50	LVI5OR3 bit added to CONFIG1
	56	ESCI vectors re-ordered
FLASH Memory	62	Minimum changed to 4ms in step 6.
System Integration Module (SIM)	93	Figure 6-5 updated
	94	Figure 6-6 updated
	106	Code example removed from SBSW description
Internal Clock Generator (ICG) Module	118	PTB6/OSC1 and PTB7/OSC2 corrected to PTC4/OSC1 and PTC3/OSC2 respectively
	137	
Configuration Registers (CONFIG1 & CONFIG2)	151	COPD corrected to COPRS in COPRS bit description
	148	LVI5OR3 bit added to CONFIG1 and default reset state changed to 0
	152	LVI5OR3 description added
Break Module (BRK)	162	Code example removed from SBSW description
Computer Operating Properly (COP) Module	178	COPL corrected to COPRS in Figure 11-1
	179	COPL corrected to COPRS in top paragraph
	180	COPL corrected to COPRS in Section 11.4.8
	181	$\overline{\text{IRQ1}}$ corrected to $\overline{\text{IRQ}}$
Low-Voltage Inhibit (LVI) Module	183	3rd bullet added to features
External Interrupt (IRQ)	190	$\overline{\text{IRQ1}}$ corrected to $\overline{\text{IRQ}}$
	191	

Section	Page (in Rev 3.0)	Description of change
Enhanced Serial Communications Interface (ESCI) Module	233 237	Extra paragraph added describing LINR bit functionality in LIN version 1.2 systems 'SCI clock source' changed to 'Frequency of the SCI clock source' in baud rate equation and description
Electrical Specifications	382 383	Output high voltage, I_{LOAD}=–10.0mA changed to –5.0mA Output low voltage, I_{LOAD}=10.0mA changed to 5.0mA Footnotes 11, 12 and 13 added

Changes from Rev 1.0 published on 17 April 2002 to Rev 2.0 published in May 2002

3V option removed.
PTB5 frequency divider function removed.
BEMF section moved from appendix to Section 18.

Section	Page (in Rev 2.0)	Description of change
Memory Map	50	ESCIBDSRC bit added to CONFIG2
Configuration Registers (CONFIG1 & CONFIG2)	148	ESCIBDSRC bit added to CONFIG2 with bit description
Enhanced Serial Communications Interface (ESCI) Module	200	Baud rate selection sentence added to sections 14.5 and 14.5.2 and after Table 14-10.

Changes from Rev 0.4 published internally on 9 April 2002 to Rev 1.0 published on 17 April 2002

Change in revision number only to denote external release version.

Changes from Rev 0.3 published on 6 September 2001 to Rev 0.4 published internally on 9 April 2002

Section	Page (in Rev 0.4)	Description of change
Memory Map	43	Reserved port register bits redefined as unimplemented
FLASH Memory	62	Note added about erasing last FLASH page
	65	Removed last two sentences of FLASH Block Protection description
System Integration Module (SIM)	92	$\overline{\text{RST}}$ description added
Configuration Registers (CONFIG1 & CONFIG2)	149	PTB7 changed to PTC3
Monitor ROM (MON)	169	**Table 10-1 . Mode Selection** added
	170	Added sentence about forced monitor mode Updated first note in section 10.5.1
	170	PTB5 column added to Table 10-2
	171	PTB5 pin added to Figure 10-1
Enhanced Serial Communications Interface (ESCI) Module	210 238 243 and 244	Note added regarding length of break character when followed by an idle. Prescale bits renamed ACLK = 0 description changed
Timer Interface A (TIMA) Module	279 – 301	Several updates for clarification
Timer Interface B (TIMB) Module	303 – 326	Several updates for clarification
Analog-to-Digital Converter (ADC) Module	345 354 355	Reserved register bits redefined as unimplemented Table 20-1 updated to show all unused combinations Left Justified Mode description corrected
Input/Output (I/O) Ports	361 – 376 366	Reserved register bits redefined as unimplemented Port B description updated
Preliminary Electrical Specifications	 382 and 384 382 and 384 382 and 384 386 391	Changes to: Hi-Z leakage current Input current Monitor mode entry voltage External clock frequency of operation FLASH page erase time

Changes from Rev 0.2 published on 1 August 2001 to Rev 0.3 published on 6 September 2001

Section	Page (in Rev 0.3)	Description of change
Memory Map	50	$001E TMBCLKSEL and SSBPUENB bits added
Configuration Registers (CONFIG1 & CONFIG2)	150	$001E TMBCLKSEL and SSBPUENB bits added
	152	Corrections to Table 8-1: PTB6 to PTC4 and PTB7 to PTC3
Low-Voltage Inhibit (LVI) Module	188	Figure 12-1 updated, digital filter removed
	189	False reset protection text updated
	190	Table 12-1 updated
	191	References to digital filter removed
Timer Interface A (TIMA) Module	280	External clock input removed from features
Timer Interface B (TIMB) Module	304	External clock input removed from features
Timebase Module (TBM)	336	Divide-by-128 replaced by divide-by-1024
	337	Figure 19-1 updated
	338	Note added after Table 19-1
Analog-to-Digital Converter (ADC) Module	345	PTC and Cx removed from Figure 20-1
	347	ADCR changed to ADCLK
Preliminary Electrical Specifications	384 and 386	Control Timing specifications added
	388 and 392	SPI characteristics added
	391	FLASH read bus clock frequency changed to 8 MHz

Changes from Rev 0.0 published on 17 July 2001 to Rev 0.2 published on 1 August 2001

Section	Page (in Rev 0.2)	Description of change
General Description	34	Third bullet in standard features list changes to: 8-MHz internal bus frequency at 5V, 4MHZ at 3V
	37	BEMF module added to block diagram
Memory Map	48	BEMF register added
	53	Register addresses changed for ADC: $003B is now reserved ADSCR is now $003C ADRH is now $003D ADRL is now $003E
	54	Reset value of $003F corrected to $04
FLASH Memory	68	Several corrections made to Table 4-1
Monitor ROM (MON)	168, 172	Erased Flash locations corrected to $FF
Keyboard Interrupt (KBD) Module	330	Keyboard interrupt vector corrected to $FFE4 and $FFE5
Analog-to-Digital Converter (ADC) Module	352	Address of ADSCR is now $003C
	355	Address of ADRH is now $003D Register description now includes left justified mode
	358	Address of ADRL is now $003E for right justified mode as well as 8-bit mode Register description now includes left justified mode
	359	Reset value of $003F corrected to $04
Input/Output (I/O) Ports	361	BEMF register added to Figure 21-1
Preliminary Electrical Specifications	383	dc injection current specifications corrected
BEMF Module	401	New appendix

Glossary

A — See “accumulator (A).”

accumulator (A) — An 8-bit general-purpose register in the CPU08. The CPU08 uses the accumulator to hold operands and results of arithmetic and logic operations.

acquisition mode — A mode of PLL operation during startup before the PLL locks on a frequency. Also see "tracking mode."

address bus — The set of wires that the CPU or DMA uses to read and write memory locations.

addressing mode — The way that the CPU determines the operand address for an instruction. The M68HC08 CPU has 16 addressing modes.

ALU — See “arithmetic logic unit (ALU).”

arithmetic logic unit (ALU) — The portion of the CPU that contains the logic circuitry to perform arithmetic, logic, and manipulation operations on operands.

asynchronous — Refers to logic circuits and operations that are not synchronized by a common reference signal.

baud rate — The total number of bits transmitted per unit of time.

BCD — See “binary-coded decimal (BCD).”

binary — Relating to the base 2 number system.

binary number system — The base 2 number system, having two digits, 0 and 1. Binary arithmetic is convenient in digital circuit design because digital circuits have two permissible voltage levels, low and high. The binary digits 0 and 1 can be interpreted to correspond to the two digital voltage levels.

Glossary

binary-coded decimal (BCD) — A notation that uses 4-bit binary numbers to represent the 10 decimal digits and that retains the same positional structure of a decimal number. For example,

234 (decimal) = 0010 0011 0100 (BCD)

bit — A binary digit. A bit has a value of either logic 0 or logic 1.

branch instruction — An instruction that causes the CPU to continue processing at a memory location other than the next sequential address.

break module — A module in the M68HC08 Family. The break module allows software to halt program execution at a programmable point in order to enter a background routine.

breakpoint — A number written into the break address registers of the break module. When a number appears on the internal address bus that is the same as the number in the break address registers, the CPU executes the software interrupt instruction (SWI).

break interrupt — A software interrupt caused by the appearance on the internal address bus of the same value that is written in the break address registers.

bus — A set of wires that transfers logic signals.

bus clock — The bus clock is derived from the CGMOUT output from the CGM. The bus clock frequency, f_{op}, is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

byte — A set of eight bits.

C — The carry/borrow bit in the condition code register. The CPU08 sets the carry/borrow bit when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the carry/borrow bit (as in bit test and branch instructions and shifts and rotates).

CCR — See “condition code register.”

central processor unit (CPU) — The primary functioning unit of any computer system. The CPU controls the execution of instructions.

CGM — See “clock generator module (CGM).”

clear — To change a bit from logic 1 to logic 0; the opposite of set.

clock — A square wave signal used to synchronize events in a computer.

clock generator module (CGM) — A module in the M68HC08 Family. The CGM generates a base clock signal from which the system clocks are derived. The CGM may include a crystal oscillator circuit and or phase-locked loop (PLL) circuit.

comparator — A device that compares the magnitude of two inputs. A digital comparator defines the equality or relative differences between two binary numbers.

computer operating properly module (COP) — A counter module in the M68HC08 Family that resets the MCU if allowed to overflow.

condition code register (CCR) — An 8-bit register in the CPU08 that contains the interrupt mask bit and five bits that indicate the results of the instruction just executed.

control bit — One bit of a register manipulated by software to control the operation of the module.

control unit — One of two major units of the CPU. The control unit contains logic functions that synchronize the machine and direct various operations. The control unit decodes instructions and generates the internal control signals that perform the requested operations. The outputs of the control unit drive the execution unit, which contains the arithmetic logic unit (ALU), CPU registers, and bus interface.

COP — See "computer operating properly module (COP)."

counter clock — The input clock to the TIM counter. This clock is the output of the TIM prescaler.

CPU — See “central processor unit (CPU).”

CPU08 — The central processor unit of the M68HC08 Family.

CPU clock — The CPU clock is derived from the CGMOUT output from the CGM. The CPU clock frequency is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

CPU cycles — A CPU cycle is one period of the internal bus clock, normally derived by dividing a crystal oscillator source by two or more so the high and low times will be equal. The length of time required to execute an instruction is measured in CPU clock cycles.

Glossary

CPU registers — Memory locations that are wired directly into the CPU logic instead of being part of the addressable memory map. The CPU always has direct access to the information in these registers. The CPU registers in an M68HC08 are:

- A (8-bit accumulator)
- H:X (16-bit index register)
- SP (16-bit stack pointer)
- PC (16-bit program counter)
- CCR (condition code register containing the V, H, I, N, Z, and C bits)

CSIC — customer-specified integrated circuit

cycle time — The period of the operating frequency: $t_{CYC} = 1/f_{OP}$.

decimal number system — Base 10 numbering system that uses the digits zero through nine.

direct memory access module (DMA) — A M68HC08 Family module that can perform data transfers between any two CPU-addressable locations without CPU intervention. For transmitting or receiving blocks of data to or from peripherals, DMA transfers are faster and more code-efficient than CPU interrupts.

DMA — See "direct memory access module (DMA)."

DMA service request — A signal from a peripheral to the DMA module that enables the DMA module to transfer data.

duty cycle — A ratio of the amount of time the signal is on versus the time it is off. Duty cycle is usually represented by a percentage.

EEPROM — Electrically erasable, programmable, read-only memory. A nonvolatile type of memory that can be electrically reprogrammed.

EPROM — Erasable, programmable, read-only memory. A nonvolatile type of memory that can be erased by exposure to an ultraviolet light source and then reprogrammed.

exception — An event such as an interrupt or a reset that stops the sequential execution of the instructions in the main program.

external interrupt module (IRQ) — A module in the M68HC08 Family with both dedicated external interrupt pins and port pins that can be enabled as interrupt pins.

fetch — To copy data from a memory location into the accumulator.

firmware — Instructions and data programmed into nonvolatile memory.

free-running counter — A device that counts from zero to a predetermined number, then rolls over to zero and begins counting again.

full-duplex transmission — Communication on a channel in which data can be sent and received simultaneously.

H — The upper byte of the 16-bit index register (H:X) in the CPU08.

H — The half-carry bit in the condition code register of the CPU08. This bit indicates a carry from the low-order four bits of the accumulator value to the high-order four bits. The half-carry bit is required for binary-coded decimal arithmetic operations. The decimal adjust accumulator (DAA) instruction uses the state of the H and C bits to determine the appropriate correction factor.

hexadecimal — Base 16 numbering system that uses the digits 0 through 9 and the letters A through F.

high byte — The most significant eight bits of a word.

illegal address — An address not within the memory map

illegal opcode — A nonexistent opcode.

I — The interrupt mask bit in the condition code register of the CPU08. When I is set, all interrupts are disabled.

index register (H:X) — A 16-bit register in the CPU08. The upper byte of H:X is called H. The lower byte is called X. In the indexed addressing modes, the CPU uses the contents of H:X to determine the effective address of the operand. H:X can also serve as a temporary data storage location.

input/output (I/O) — Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal and writes to an output to change the level on an external signal.

instructions — Operations that a CPU can perform. Instructions are expressed by programmers as assembly language mnemonics. A CPU interprets an opcode and its associated operand(s) and instruction.

Glossary

interrupt — A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine.

interrupt request — A signal from a peripheral to the CPU intended to cause the CPU to execute a subroutine.

I/O — See “input/output (I/0).”

IRQ — See "external interrupt module (IRQ)."

jitter — Short-term signal instability.

latch — A circuit that retains the voltage level (logic 1 or logic 0) written to it for as long as power is applied to the circuit.

latency — The time lag between instruction completion and data movement.

least significant bit (LSB) — The rightmost digit of a binary number.

logic 1 — A voltage level approximately equal to the input power voltage (V_{DD}).

logic 0 — A voltage level approximately equal to the ground voltage (V_{SS}).

low byte — The least significant eight bits of a word.

low voltage inhibit module (LVI) — A module in the M68HC08 Family that monitors power supply voltage.

LVI — See "low voltage inhibit module (LVI)."

M68HC08 — A Motorola family of 8-bit MCUs.

mark/space — The logic 1/logic 0 convention used in formatting data in serial communication.

mask — 1. A logic circuit that forces a bit or group of bits to a desired state. 2. A photomask used in integrated circuit fabrication to transfer an image onto silicon.

mask option — A optional microcontroller feature that the customer chooses to enable or disable.

mask option register (MOR) — An EPROM location containing bits that enable or disable certain MCU features.

MCU — Microcontroller unit. See “microcontroller.”

memory location — Each M68HC08 memory location holds one byte of data and has a unique address. To store information in a memory location, the CPU places the address of the location on the address bus, the data information on the data bus, and asserts the write signal. To read information from a memory location, the CPU places the address of the location on the address bus and asserts the read signal. In response to the read signal, the selected memory location places its data onto the data bus.

memory map — A pictorial representation of all memory locations in a computer system.

microcontroller — Microcontroller unit (MCU). A complete computer system, including a CPU, memory, a clock oscillator, and input/output (I/O) on a single integrated circuit.

modulo counter — A counter that can be programmed to count to any number from zero to its maximum possible modulus.

monitor ROM — A section of ROM that can execute commands from a host computer for testing purposes.

MOR — See "mask option register (MOR)."

most significant bit (MSB) — The leftmost digit of a binary number.

multiplexer — A device that can select one of a number of inputs and pass the logic level of that input on to the output.

N — The negative bit in the condition code register of the CPU08. The CPU sets the negative bit when an arithmetic operation, logical operation, or data manipulation produces a negative result.

nibble — A set of four bits (half of a byte).

object code — The output from an assembler or compiler that is itself executable machine code, or is suitable for processing to produce executable machine code.

opcode — A binary code that instructs the CPU to perform an operation.

open-drain — An output that has no pullup transistor. An external pullup device can be connected to the power supply to provide the logic 1 output voltage.

operand — Data on which an operation is performed. Usually a statement consists of an operator and an operand. For example, the operator may be an add instruction, and the operand may be the quantity to be added.

oscillator — A circuit that produces a constant frequency square wave that is used by the computer as a timing and sequencing reference.

OTPROM — One-time programmable read-only memory. A nonvolatile type of memory that cannot be reprogrammed.

overflow — A quantity that is too large to be contained in one byte or one word.

page zero — The first 256 bytes of memory (addresses $0000–$00FF).

parity — An error-checking scheme that counts the number of logic 1s in each byte transmitted. In a system that uses odd parity, every byte is expected to have an odd number of logic 1s. In an even parity system, every byte should have an even number of logic 1s. In the transmitter, a parity generator appends an extra bit to each byte to make the number of logic 1s odd for odd parity or even for even parity. A parity checker in the receiver counts the number of logic 1s in each byte. The parity checker generates an error signal if it finds a byte with an incorrect number of logic 1s.

PC — See “program counter (PC).”

peripheral — A circuit not under direct CPU control.

phase-locked loop (PLL) — A oscillator circuit in which the frequency of the oscillator is synchronized to a reference signal.

PLL — See "phase-locked loop (PLL)."

pointer — Pointer register. An index register is sometimes called a pointer register because its contents are used in the calculation of the address of an operand, and therefore points to the operand.

polarity — The two opposite logic levels, logic 1 and logic 0, which correspond to two different voltage levels, V_{DD} and V_{SS}.

polling — Periodically reading a status bit to monitor the condition of a peripheral device.

port — A set of wires for communicating with off-chip devices.

prescaler — A circuit that generates an output signal related to the input signal by a fractional scale factor such as 1/2, 1/8, 1/10 etc.

program — A set of computer instructions that cause a computer to perform a desired operation or operations.

program counter (PC) — A 16-bit register in the CPU08. The PC register holds the address of the next instruction or operand that the CPU will use.

pull — An instruction that copies into the accumulator the contents of a stack RAM location. The stack RAM address is in the stack pointer.

pullup — A transistor in the output of a logic gate that connects the output to the logic 1 voltage of the power supply.

pulse-width — The amount of time a signal is on as opposed to being in its off state.

pulse-width modulation (PWM) — Controlled variation (modulation) of the pulse width of a signal with a constant frequency.

push — An instruction that copies the contents of the accumulator to the stack RAM. The stack RAM address is in the stack pointer.

PWM period — The time required for one complete cycle of a PWM waveform.

RAM — Random access memory. All RAM locations can be read or written by the CPU. The contents of a RAM memory location remain valid until the CPU writes a different value or until power is turned off.

RC circuit — A circuit consisting of capacitors and resistors having a defined time constant.

read — To copy the contents of a memory location to the accumulator.

register — A circuit that stores a group of bits.

reserved memory location — A memory location that is used only in special factory test modes. Writing to a reserved location has no effect. Reading a reserved location returns an unpredictable value.

reset — To force a device to a known condition.

ROM — Read-only memory. A type of memory that can be read but cannot be changed (written). The contents of ROM must be specified before manufacturing the MCU.

SCI — See "serial communication interface module (SCI)."

serial — Pertaining to sequential transmission over a single line.

serial communications interface module (SCI) — A module in the M68HC08 Family that supports asynchronous communication.

serial peripheral interface module (SPI) — A module in the M68HC08 Family that supports synchronous communication.

set — To change a bit from logic 0 to logic 1; opposite of clear.

shift register — A chain of circuits that can retain the logic levels (logic 1 or logic 0) written to them and that can shift the logic levels to the right or left through adjacent circuits in the chain.

signed — A binary number notation that accommodates both positive and negative numbers. The most significant bit is used to indicate whether the number is positive or negative, normally logic 0 for positive and logic 1 for negative. The other seven bits indicate the magnitude of the number.

software — Instructions and data that control the operation of a microcontroller.

software interrupt (SWI) — An instruction that causes an interrupt and its associated vector fetch.

SPI — See "serial peripheral interface module (SPI)."

stack — A portion of RAM reserved for storage of CPU register contents and subroutine return addresses.

stack pointer (SP) — A 16-bit register in the CPU08 containing the address of the next available storage location on the stack.

start bit — A bit that signals the beginning of an asynchronous serial transmission.

status bit — A register bit that indicates the condition of a device.

stop bit — A bit that signals the end of an asynchronous serial transmission.

subroutine — A sequence of instructions to be used more than once in the course of a program. The last instruction in a subroutine is a return from subroutine (RTS) instruction. At each place in the main program where the subroutine instructions are needed, a jump or branch to subroutine (JSR or BSR) instruction is used to call the subroutine. The CPU leaves the flow of the main program to execute the instructions in the subroutine. When the RTS instruction is executed, the CPU returns to the main program where it left off.

synchronous — Refers to logic circuits and operations that are synchronized by a common reference signal.

TIM — See "timer interface module (TIM)."

timer interface module (TIM) — A module used to relate events in a system to a point in time.

timer — A module used to relate events in a system to a point in time.

toggle — To change the state of an output from a logic 0 to a logic 1 or from a logic 1 to a logic 0.

tracking mode — Mode of low-jitter PLL operation during which the PLL is locked on a frequency. Also see "acquisition mode."

two's complement — A means of performing binary subtraction using addition techniques. The most significant bit of a two's complement number indicates the sign of the number (1 indicates negative). The two's complement negative of a number is obtained by inverting each bit in the number and then adding 1 to the result.

unbuffered — Utilizes only one register for data; new data overwrites current data.

unimplemented memory location — A memory location that is not used. Writing to an unimplemented location has no effect. Reading an unimplemented location returns an unpredictable value. Executing an opcode at an unimplemented location causes an illegal address reset.

V —The overflow bit in the condition code register of the CPU08. The CPU08 sets the V bit when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow bit.

variable — A value that changes during the course of program execution.

VCO — See "voltage-controlled oscillator."

vector — A memory location that contains the address of the beginning of a subroutine written to service an interrupt or reset.

voltage-controlled oscillator (VCO) — A circuit that produces an oscillating output signal of a frequency that is controlled by a dc voltage applied to a control input.

waveform — A graphical representation in which the amplitude of a wave is plotted against time.

wired-OR — Connection of circuit outputs so that if any output is high, the connection point is high.

word — A set of two bytes (16 bits).

write — The transfer of a byte of data from the CPU to a memory location.

Glossary

X — The lower byte of the index register (H:X) in the CPU08.

Z — The zero bit in the condition code register of the CPU08. The CPU08 sets the zero bit when an arithmetic operation, logical operation, or data manipulation produces a result of $00.

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku, Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://motorola.com/semiconductors

MC68HC908EY16/D