MC68HC908W32/H
REV. 1.0

MC68HC908W32

HCMOS Microcontroller Unit

TECHNICAL DATA

MOTOROLA

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it 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.

List of Sections

List of Sections

Table of Contents

Section 1. General Description

Section 2. Memory

Section 3. Random-Access Memory (RAM)

Section 4. FLASH Memory

Section 5. Configuration Register (CONFIG)

Section 6. Central Processor Unit (CPU)

Section 7. System Integration Module (SIM)

Section 8. Clock Generation Module (CGMB)

Section 9. Monitor ROM (MON)

Section 10. Time Base Module

Section 11. Timer Interface Module (TIM)

Section 12. Reloadable Timer Module

Section 13. Serial Peripheral Interface Module (SPI)

Section 14. Ring Detector Module (RING)

Section 15. DTMF/Melody Generator Module

Section 16. I/O Ports

Section 17. External Interrupt (IRQ)

Section 18. Keyboard Interrupt Module (KBI)

Section 19. Computer Operating Properly (COP)

Section 20. Low Voltage Inhibit (LVI)

Section 21. Break Module (BREAK)

Section 22. Electrical Specifications

Section 23. Mechanical Specifications

Table of Contents

List of Figures

Figure	Title	Page

Figure	Title	Page

List of Figures

List of Tables

Table	Title	Page

Section 1. General Description

1.1 Contents

1.2 Introduction

The MC68HC908W32 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). The M68HC08 Family is based on the customer-specified integrated circuit (CSIC) design strategy. 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

Features of the MC68HC908W32 include the following:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 families
- 2.5 MHz internal bus frequency at 3.0V
- 32K bytes of on-chip FLASH
- 768 bytes of on-chip RAM
- On-chip monitor ROM firmware for use with host personal computer
- 16-bit auto-reloadable timer
- 16-bit, 4-channel timer interface module (TIM)
  - Each channel selectable as input capture, output compare, or PWM
- DTMF/melody generator
- Power ring detector
- Two external vectored interrupt pins
- Single clock source using one 32.768 KHz external crystal
- On-chip PLL clock generator
- Serial peripheral interface modules (SPI)
- System protection features
  - Optional computer operating properly (COP) reset
  - Program selectable low voltage interrupt (LVI)
  - Low voltage reset
  - Illegal opcode detection
  - Illegal address detection
- 44-pin quad flat package (QFP) or 28-pin small outline integrated circuit package (SOIC)

- Low-power design (fully static with stop and wait modes)
- Master reset pin and power-on reset
- 36 or 20 general purpose I/O pins including
  - 13 shared function I/O pins
  - 8-bit keyboard wake-up port pins
  - Two light emitting diode (LED) direct drive port pins (sink and source)

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 MC68HC908W32 block diagram.

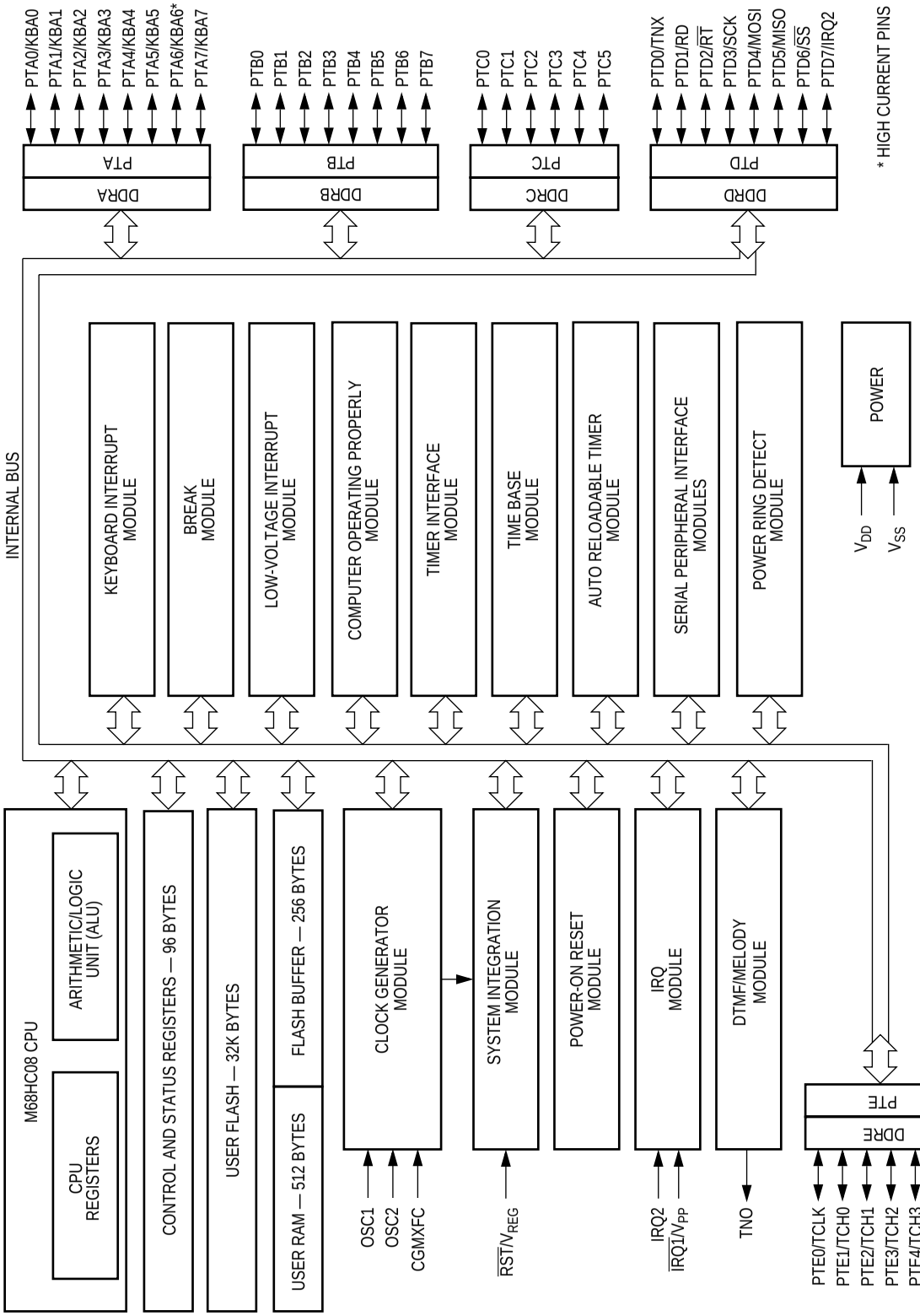


Figure 1-1. MC68HC908W32 Block Diagram

1.5 Pin Assignment

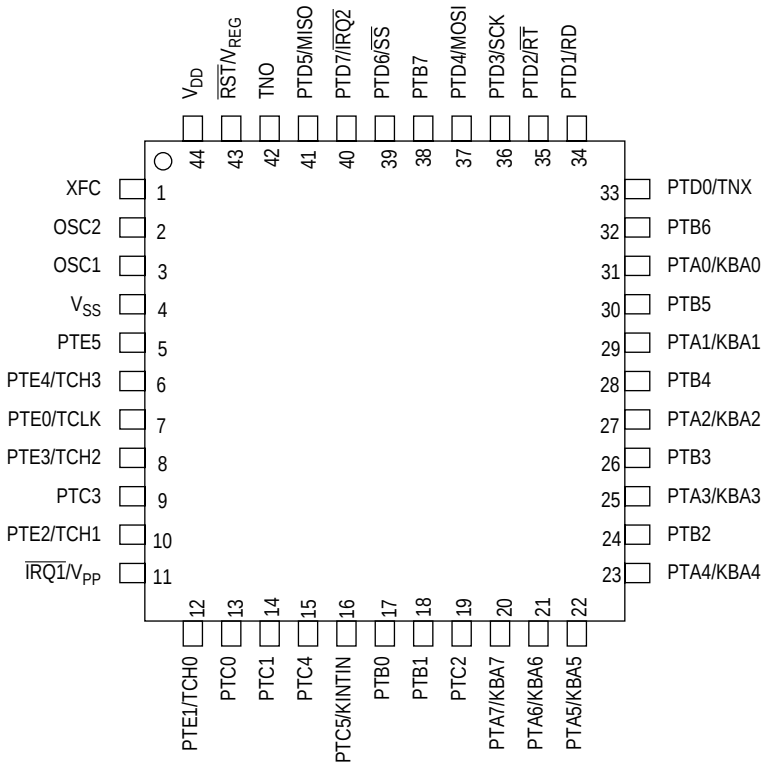


Figure 1-2. 44-Pin QFP Pin Assignments

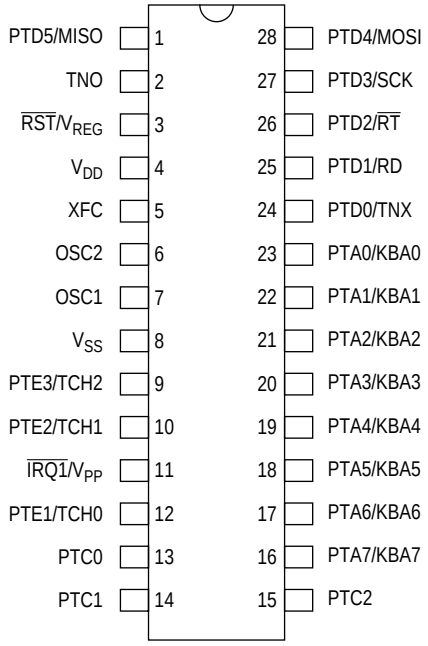


Figure 1-3. 28-Pin SOIC Pin Assignments

1.6 Pin Functions

Descriptions of the pin functions are provided here.

1.6.1 Power Supply Pins (V_{DD}, 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 1-4** 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.

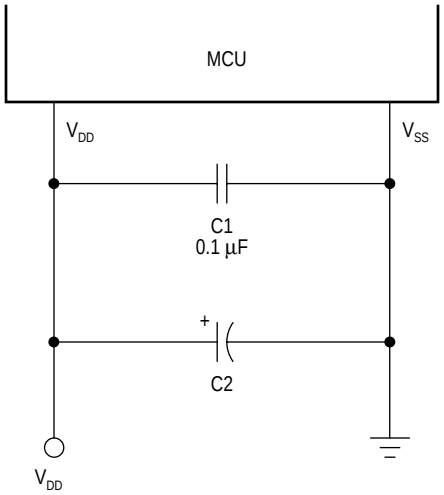


Figure 1-4. Power Supply Bypassing

1.6.2 Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. (See **Section 8. Clock Generation Module (CGMB)**.)

OSC1 is also used as a high voltage input for test purpose.

1.6.3 External Reset Pin ($\overline{RST}$/V_{REG})

A logic zero on the $\overline{RST}$ pin forces the MCU to a known start-up state. $\overline{RST}$ is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted, therefore use open drain outputs with resistive pullups on this pin. (See **Section 7. System Integration Module (SIM)**.)

$\overline{RST}$ pin shares with V_{REG} which is used as DTMF regulator. (See **Section 15. DTMF/Melody Generator Module**.)

1.6.4 External Interrupt Pins ($\overline{IRQ1}/V_{PP}$, $\overline{IRQ2}$)

$\overline{IRQ1}/V_{PP}$ is the asynchronous external interrupt pin. $\overline{IRQ2}$ is a second asynchronous external interrupt. (See **Section 7. System Integration Module (SIM)** and **Section 17. External Interrupt (IRQ)**.)

1.6.5 External Filter Capacitor Pin (XFC)

CGMXFC is an external filter capacitor connection for the CGM. (See **Section 8. Clock Generation Module (CGMB)**.)

1.6.6 Port A Input/Output (I/O) Pins (PTA7/KBA7–PTA0/KBA0)

Port A is an 8-bit bidirectional I/O port. Any or all of the port pins can be programmed to serve as external interrupt pins. (See **Section 16. I/O Ports** and **Section 18. Keyboard Interrupt Module (KBI)**.)

1.6.7 Port B I/O Pins (PTB7–PTB0)

Port B is an 8-bit bidirectional I/O port. (See **Section 16. I/O Ports**.)

1.6.8 Port C I/O Pins (PTC5–PTC0)

Port C is a 6-bit bidirectional I/O port that shares one pin with the keyboard interrupt function when it is used in an emulator. User should consider port C as a simple 6-bit bidirectional I/O port. (See **Section 16. I/O Ports**.)

1.6.9 Port D I/O Pins (PTD7–PTD0)

Port D is an 8-bit bidirectional I/O port that shares its pins with the serial peripheral interface module (SPI), ring detector module (RING), and DTMF/Melody generator module. (See **Section 16. I/O Ports**, **Section 13. Serial Peripheral Interface Module (SPI)**, **Section 14. Ring Detector Module (RING)** and **Section 15. DTMF/Melody Generator Module**.)

1.6.10 DTMF/Melody Generator Pins (TN0, TNX)

TNO is the analog output pin and TNX is the digital output pin of the DTMF/Melody generator. (See **Section 15. DTMF/Melody Generator Module**.)

1.6.11 Port E I/O Pins (PTE5–PTE0)

Port E is a 6-bit bidirectional I/O port that shares five of its pins with the Timer channels. (See **Section 16. I/O Ports** and **Section 11. Timer Interface Module (TIM)**.)

1.7 Clock Distribution

Each module in the MC68HC908W32 that requires a clock uses either the buffered raw oscillator clock CGMXCLK, the DTMF clock CGMOUTX or the system bus clock CGMOUT. CGMOUT is a divide-by-2 of either the PLL (if engaged) or CGMXCLK. CGMOUT is derived from CGMVCLK through a software selectable prescaler of either 1, 1/2, 1/4 or 1/8. The internal bus clock is a divide-by-2 of CGMOUT.

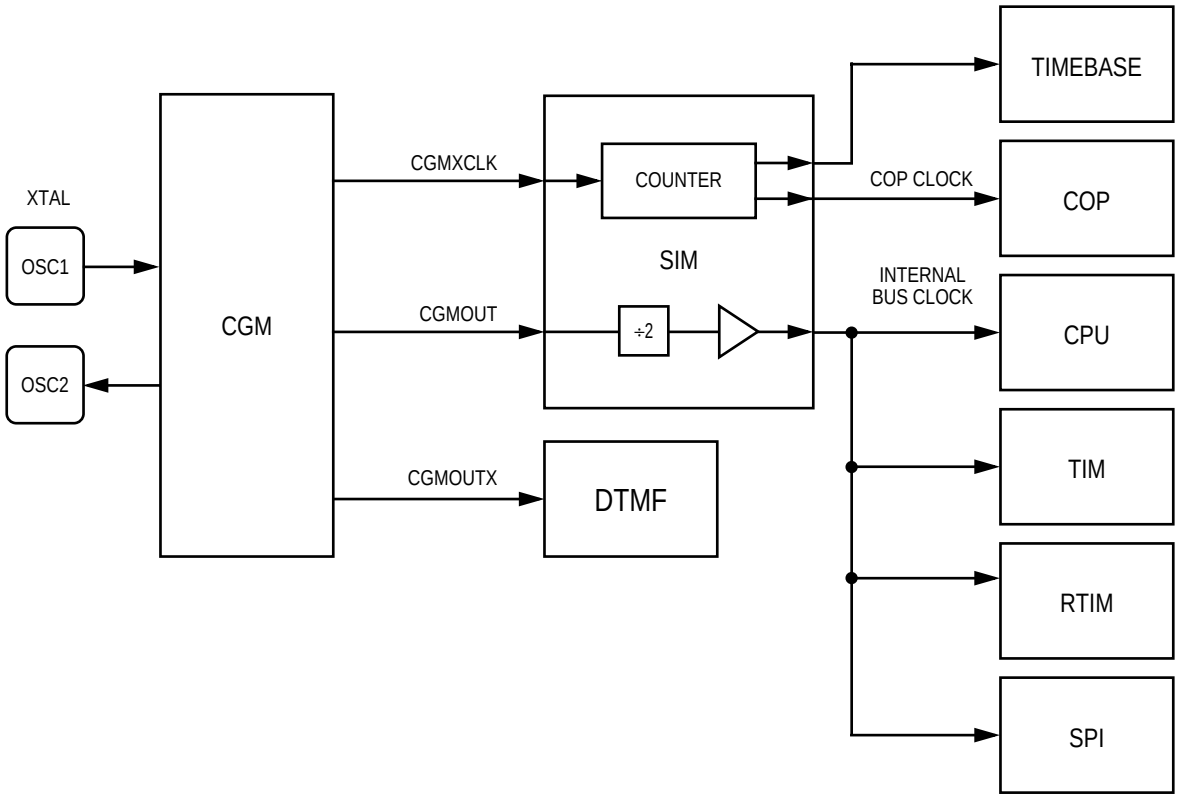


Figure 1-5. Clock Distribution Block Diagram

Section 2. Memory

2.1 Contents

2.2 Introduction

The CPU08 can address 64K-bytes of memory space. The memory map, shown in **Figure 2-1**, includes:

- 32488 bytes of FLASH
- 768 bytes of RAM
- 24 bytes of user-defined vectors
- 240 bytes of monitor ROM

Figure 2-1. Memory Map

2.3 I/O Section

Addresses $0000–$003F, shown in **Figure 2-2**, contain most of the control, status, and data registers. Additional I/O registers have the following addresses:

- $FE00; SIM break status register, SBSR
- $FE01; SIM reset status register, SRSR
- $FE02; reserved
- $FE03; SIM break flag control register, SBFCR
- $FE04; reserved
- $FE05; reserved
- $FE06; reserved
- $FE07; FLASH Control Register, FLCR
- $FE08; FLASH Block Protect Register, FLBPR
- $FE09; reserved
- $FE0A; reserved
- $FE0B; reserved
- $FE0C; break address register high, BRKH
- $FE0D; break address register low, BRKL
- $FE0E; break status and control register, BRKSCR
- $FFFF; COP control register, COPCTL

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	R:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		W:								
$0001	Port B Data Register (PTB)	R:	PTB7	PTB6	PTB25	PTB4	PTB3	PTB2	PTB1	PTB0
		W:								
$0002	Port C Data Register (PTC)	R:	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		W:								
$0003	Port D Data Register (PTD)	R:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		W:								
$0004	Data Direction Register A (DDRA)	R:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		W:								
$0005	Data Direction Register B (DDRB)	R:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		W:								
$0006	Data Direction Register C (DDRC)	R:	0	0	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		W:								
$0007	Data Direction Register D (DDRD)	R:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		W:								
$0008	Port E Data Register (PTE)	R:	0	0	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		W:								
$0009	Data Direction Register E (DDRE)	R:	0	0	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		W:								
$000A	Reserved	R:								
		W:								
$000B	Reserved	R:								
		W:								
$000C	Port A Keyboard Status and Control Register (KBDSCR)	R:	0	0	0	0	KEYDF	0	IMASKD	MODED
		W:						ACKD		
$000D	Port A Keyboard Interrupt Enable Register (KBDIER)	R:	KBAIE7	KBAIE6	KBAIE5	KBAIE4	KBAIE3	KBAIE2	KBAIE1	KBAIE0
		W:								
$000E	Reserved	R:								
		W:								
$000F	Reserved	R:								
		W:								
$0010	Reloadable Timer Control Register (RLTCR)	R:	RLTCKS		RLTPS1	RLTPS0	RLTEN	RLTIE	0	RLTUF
		W:							CRTUF	
$0011	Reloadable Timer Preset 1 (RLTPR1)	R:	TP15	TP14	TP13	TP12	TP11	TP10	TP9	TP8
		W:								
$0012	Reloadable Timer Preset 2 (RLTPR2)	R:	TP7	TP6	TP5	TP4	TP3	TP2	TP1	TP0
		W:								
$0013	Reloadable Timer Count 1 (RLTCNT1)	R:	CNT15	CNT14	CNT13	CNT12	CNT11	CNT10	CNT9	CNT8
		W:								
$0014	Reloadable Timer Count 2 (RLTCNT2)	R:	CNT7	CNT6	CNT5	CN4	CNT3	CN2	CNT1	CNT0
		W:								

= Unimplemented

Figure 2-2. Control, Status, and Data Registers

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0015	Time Base Control Register (TBCR)	R:	TBIF	TBIE	TBR1	TBR0	0	0	TBON	0
		W:					TACK			
$0016	Reserved	R:								
		W:								
$0017	Reserved	R:								
		W:								
$0018	SPI Control Register	R:	SPRIE		SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		W:								
$0019	SPI Status Register	R:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		W:								
$001A	SPI Data Register	R:	R7	R6	R5	R4	R3	R2	R1	R0
		W:	T7	T6	T5	T4	T3	T2	T1	T0
$001B	Reserved	R:								
		W:								
$001C	Reserved	R:								
		W:								
$001D	Port Option Control Register (POC)	R:	OSCSTOPEN	IRQ2MUX	SPIMUX	RDMUX	TNXMUX	PCP	PBP	PAP
		W:								
$001E	LVI Control Register (LVICR)	R:	LVIOUT	LVIF	0	LVREN	LVIT1	LVIT0	$\overline{\text{LVISTOPEN}}$	LVIEN
		W:			LVIACK					
$001F	Configuration Register (CONFIG)	R:	0	0	0	0	SSREC	COPRS	STOP	COPD
		W:								
$0020	Timer Status and Control Register (TSC)	R:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		W:	0			TRST				
$0021	Reserved	R:								
		W:								
$0022	Timer Counter Register High (TCNTH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0023	Timer Counter Register Low (TCNTL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0024	Timer Modulo Register High (TMODH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0025	Timer Modulo Register Low (TMODL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0026	Timer Channel 0 Status and Control Register (TSC0)	R:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		W:	0							
$0027	Timer Channel 0 Register High (TCH0H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0028	Timer Channel 0 Register Low (TCH0L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0029	Timer Channel 1 Status and Control Register (TSC1)	R:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		W:	0							

= Unimplemented

Figure 2-2. Control, Status, and Data Registers (Continued)

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$002A	Timer Channel 1 Register High (TCH1H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$002B	Timer Channel 1 Register Low (TCH1L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$002C	Timer Channel 2 Status and Control Register (TSC2)	R:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		W:	0							
$002D	Timer Channel 2 Register High (TCH2H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$002E	Timer Channel 2 Register Low (TCH2L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$002F	Timer Channel 3 Status and Control Register (TSC3)	R:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		W:	0							
$0030	Timer Channel 3 Register High (TCH3H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0031	Timer Channel 3 Register Low (TCH3L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0032	DTMF Row Frequency Control Register (FCR)	R:	FCR7	FCR6	FCR5	FCR4	FCR3	FCR2	FCR1	FCR0
		W:								
$0033	DTMF Column Frequency Control Register (FCC)	R:	FCC7	FCC6	FCC5	FCC4	FCC3	FCC2	FCC1	FCC0
		W:								
$0034	DTMF Tone Control Register (TNCR)	R:	0	0	0	0	TNE	TNXE	TNOE	TNOM
		W:								
$0035	Tone Data Register (DTSUM)	R:	0	0	DT5	DT4	DT3	DT2	DT1	DT0
		W:								
$0036	PLL Control Register (PCTL)	R:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
		W:								
$0037	PLL Bandwidth Control (PBWC)	R:	AUTO	LOCK	$\overline{\text{ACQ}}$	0	0	0	0	0
		W:								
$0038	PLL Multiplier Select Register High (PMSH)	R:	0	0	0	0	MUL11	MUL10	MUL9	MUL8
		W:								
$0039	PLL Multiplier Select Register Low (PMSL)	R:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
		W:								
$003A	PLL VCO Range Select Register (PMRS)	R:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
		W:								
$003B	PLL Reference Divider Select Register (PMDS)	R:	0	0	0	0	RDS3	RDS2	RDS1	RDS0
		W:								
$003C	CGM Clock Selection Register (PCKS)	R:	0	0	0	0	0	0	CGMS1	CGMS0
		W:								
$003D	IRQ Status/Control Register (ISCR)	R:	IRQF2	0	IMASK2	MODE2	IRQF1	0	IMASK1	MODE1
		W:		ACK2				ACK1		
$003E	Ring Detector Control Register (RDCR)	R:					RVALID	RDIF	0	RDE
		W:							RDIACK	

= Unimplemented

Figure 2-2. Control, Status, and Data Registers (Continued)

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$003F	Reserved	R:								
		W:								
$FE00	SIM Break Status Register (SBSR)	R:	R	R	R	R	R	R	SBSW	R
		W:								
$FE01	SIM Reset Status Register (SRSR)	R:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
		W:								
$FE02	Reserved	R:								
		W:								
$FE03	SIM Break Flag Control Register (SBFCR)	R:	BCFE	R	R	R	R	R	R	R
		W:								
$FE04	Reserved	R:								
		W:								
$FE05	Reserved	R:								
		W:								
$FE06	Reserved	R:								
		W:								
$FE07	FLASH Control Register (FLCR)	R:	0	0	0	0	HVEN	MASS	ERASE	PGM
		W:								
$FE08	FLASH Protection Register (FLSPR)	R:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
		W:								
$FE09	Reserved	R:								
		W:								
$FE0A	Reserved	R:								
		W:								
$FE0B	Reserved	R:								
		W:								
$FE0C	Break Address Register High (BRKH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$FE0D	Break Address Register Low (BRKL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$FE0E	Break Status and Control Register (BRKSCR)	R:	BRKE	BRKA	0	0	0	0	0	0
		W:								
$FE0F	Reserved	R:								
		W:								
$FFFF	COP Control Register (COPCTL)	R:	LOW BYTE OF RESET VECTOR							
		W:	WRITING TO $FFFF CLEARS COP COUNTER							

= Unimplemented

Figure 2-2. Control, Status, and Data Registers (Continued)

Table 2-1. Vector Addresses

Address	Vector
$FFE8:$FFE9	PLL Vector (High:Low)
$FFEA:$FFEB	Keyboard Interrupt Vector (High:Low)
$FFEC:$FFED	Ring Detector Vector (High:Low)
$FFEE:$FFEF	SPI Vector (High:Low)
$FFF0:$FFF1	Real Time Clock Vector (High:Low)
$FFF2:$FFF3	TIM Vector (High:Low)
$FFF4:$FFF5	Reloadable Timer Vector (High:Low)
$FFF6:$FFF7	IRQ2 Vector (High:Low)
$FFF8:$FFF9	IRQ1 Vector (High:Low)
$FFFA:$FFFB	LVI Vector (High:Low)
$FFFC:$FFFD	SWI Vector (High:Low)
$FFFE:$FFFF	Reset Vector (High:Low)

Low ↑ Priority ↓ High

2.4 Monitor ROM

The 240 bytes at addresses $FE10-$FEFF are reserved ROM area that contains the instructions for the monitor functions. (See **Section 9. Monitor ROM (MON)**.)

Section 3. Random-Access Memory (RAM)

3.1 Contents

3.2 Introduction

This section describes the 768 bytes of RAM (random-access memory).

3.3 Functional Description

Addresses $0080 through $037F are RAM locations. The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in the 64-Kbyte memory space.

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

Within page zero ($0000–$00FF) are 128 bytes of RAM. Because the stack RAM is programmable, all page zero RAM locations can be used for I/O control and user data or code. When the stack pointer is moved from its reset location at $00FF out of page zero, direct addressing mode instructions can efficiently access all page zero RAM locations. 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 CPU registers.

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

During a subroutine call, the CPU uses 2 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 may 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. The FLASH memory can be read, programmed, and erased from a single external supply through the use of the internal charge pump for program and erase.

NOTE: *The Charge Pump for the FLASH on this device is optimized for 3V programming.*

4.3 Functional Description

The FLASH memory is an array of 32,256 bytes with an additional 232 bytes of user memory and 24 bytes of user vectors. *An erased bit reads as logic 1 and a programmed bit reads as a logic 0*. Program and erase operations are facilitated through control bits in a memory mapped FLASH control register (FLCR). The FLASH memory and the associated charge pump is designed to operate in the voltage range 2.7V to 3.6V. The address ranges for the user memory and vectors are:

- $8000–$FDFF; user memory
- $FE07; FLASH control register
- $FE08; FLASH block protect register
- $FF00–$FFE7; user memory
- $FFE8–$FFFF; these locations are reserved for user-defined interrupt and reset vectors

Programming tools are available from Motorola. Contact your local Motorola representative for more information.

4.4 FLASH Control Register (FLCR)

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

Address: $FE07

	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)

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 only be set if either PGM = 1 or ERASE = 1 and the proper sequence for program or erase is followed.

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

MASS — Mass Erase Control Bit

Setting this read/write bit configures the 32 Kbytes FLASH array for mass erase operation.

1 = Mass Erase operation selected
0 = Mass Erase operation not 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 not selected

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 not selected

4.5 FLASH Mass Erase Operation

Use the following procedure to erase entire FLASH memory:

1. Set both the ERASE bit, and the MASS bit in the FLASH control register.
2. Write to any FLASH address with any data within the FLASH memory address range.
3. Wait for a time, t_{nvs} (10µs).

4. Set the HVEN bit.
5. Wait for a time, t_{ERASE} (4ms).
6. Clear the ERASE bit.
7. Wait for a time, t_{nvhl} (100µs).
8. Clear the HVEN bit.
9. After time, t_{rcv} (1µs), 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 as shown, but other unrelated operations may occur between the steps.*

4.6 FLASH Program Operation

Programming of the FLASH memory is done on a page basis. A page consists of 64 consecutive bytes starting from addresses $XX00, $XX40, $XX80, and $XXC0. Use this step-by-step procedure to program a page of FLASH memory:

1. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
2. Write any FLASH address with any data within the page address range desired.
3. Wait for a time, t_{nvs} (10µs).
4. Set the HVEN bit.
5. Wait for a time, t_{pgs} (5µs).
6. Write to the FLASH address with data to the byte desired to be programmed.
7. Wait for time, t_{PROG} (30µs).
8. Repeat step 6 and 7 until all the bytes within the page are programmed.

9. Clear the PGM bit.
10. Wait for time, t_{nvh} (5µs).
11. Clear the HVEN bit.
12. After time, t_{rcv} (1µs), the memory can be accessed in read mode again.

This program sequence is repeated throughout the memory until all data is programmed. For minimum overall programming time and least program disturb effect, the sequence should be part of an intelligent operation which iterates per page.

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.*

4.7 FLASH Protection

Due to the ability of the on-board charge pump to erase and program the FLASH memory in the target application, provision is made for protecting blocks of memory from unintentional erase or program operations due to system malfunction. This protection is done by use of a FLASH Block Protect Register (FLBPR). The FLBPR determines the range of the FLASH memory which is to be protected. The range of the protected area starts from a location defined by FLBPR and ends 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.

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

4.8 FLASH Block Protect Register (FLBPR)

The FLASH block protect register is implemented as an 8-bit I/O register. The BPR bit content of the register determines the starting location of the protected range within the FLASH memory.

Address: $FE08

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

Figure 4-2. FLASH Block Protect Register (FLBPR)

BPR[7:0] — FLASH Block Protect Bits

These eight bits specify the 8 most significant bits of the addressable 32K FLASH memory location from where the memory gets protected. Memory is protected starting from this location to the end of the FLASH memory. The start address is on a block boundary (128 bytes).

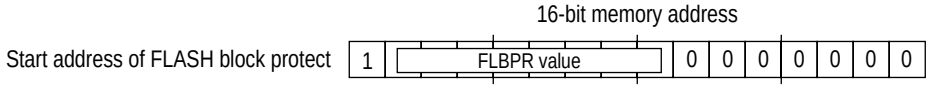


Figure 4-3. FLASH Block Protect Start Address

Examples of protected range:

$00 = The entire FLASH memory is protected: $8000–$FFFF.

$01 = Protected range: $8080–$FFFF.

$02 = Protected range: $8100–$FFFF.

... and so on.

$FE = Protected range: $FF00–$FFFF.

$FF = The entire FLASH memory is not protected.

Section 5. Configuration Register (CONFIG)

5.1 Contents

5.2 Introduction

This section describes the configuration register (CONFIG). The configuration register enables or disables the following options:

- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- STOP instruction
- Computer operating properly module (COP)

5.3 Functional Description

The configuration register is used in the initialization of various options. The configuration register 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 MCU it is recommended that this register be written immediately after reset. The configuration register is located at $001F. For compatibility, a write to the 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. For references in the documentation which refer to the MOR (Mask Option Register), the CONFIG would be applicable for the FLASH/OTPROM version of the device.*

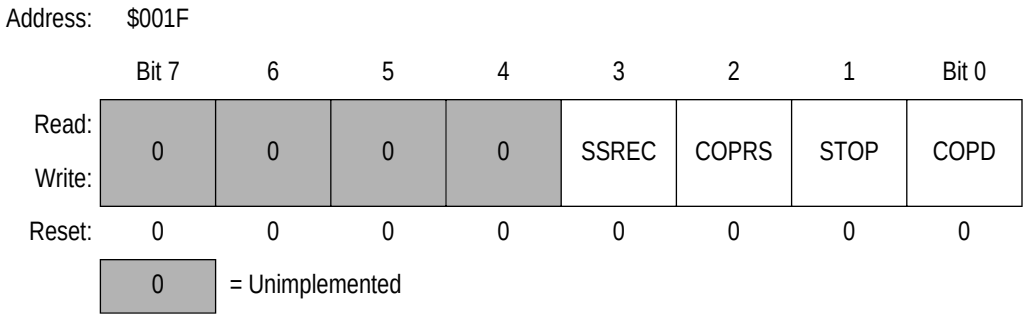


Figure 5-1. Configuration Register (CONFIG)

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 CGMXCLK cycles

NOTE: *If using an external crystal oscillator, do not set the SSREC bit.*

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 19. Computer Operating Properly (COP)**.)

1 = COP module disabled
0 = COP module enabled

COPPS — COP reset Period Selection Bit

COPPS selects the timeout period of the COP module. (See **Section 19. Computer Operating Properly (COP)**.)

1 = Selects an 13-bit divider for COP
0 = Selects a 18-bit divider for COP

Section 6. Central Processor Unit (CPU)

6.1 Contents

6.2 Introduction

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

6.3 Features

- 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

6.4 CPU Registers

Figure 6-1 shows the five CPU registers. CPU registers are not part of the memory map.

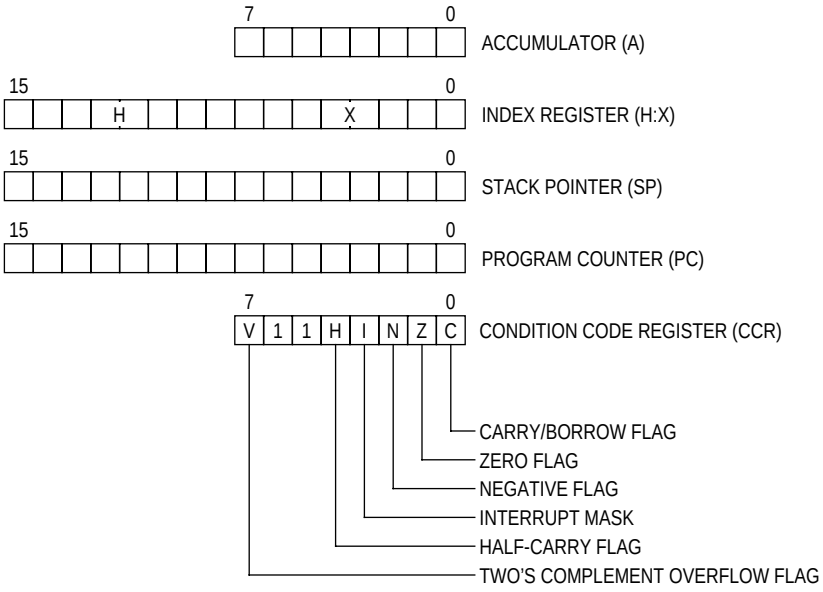


Figure 6-1. CPU Registers

6.4.1 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.

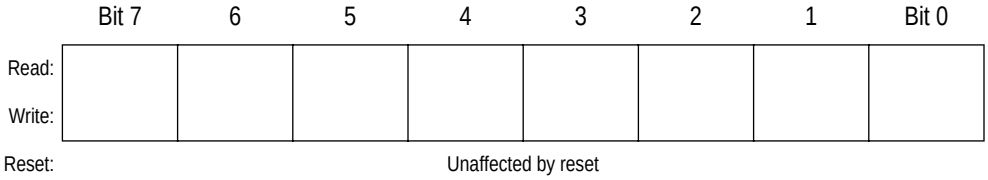


Figure 6-2. Accumulator (A)

6.4.2 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.

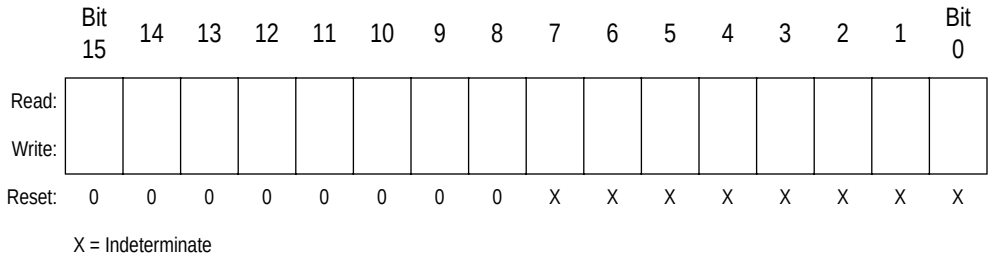


Figure 6-3. Index Register (H:X)

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

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

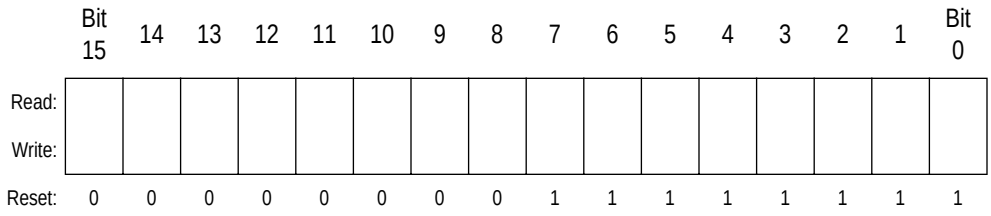


Figure 6-4. Stack Pointer (SP)

NOTE: *The location of the stack is arbitrary and may be relocated anywhere in 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.*

6.4.4 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.

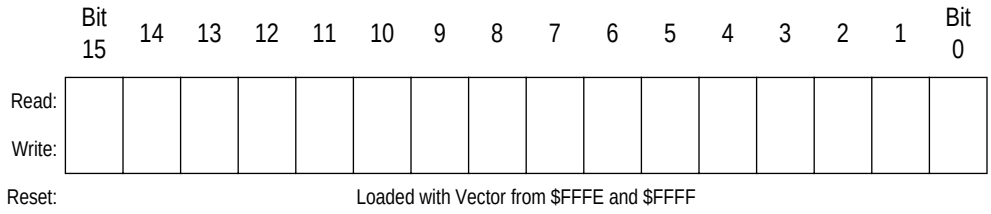


Figure 6-5. Program Counter (PC)

6.4.5 Condition Code Register

The 8-bit condition code register contains the interrupt mask and five flags that indicate the results of the instruction just executed. Bits 6 and

5 are set permanently to logic 1. The following paragraphs describe the functions of the condition code register.

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

X = Indeterminate

Figure 6-6. Condition Code Register (CCR)

V — Overflow Flag

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

1 = Overflow
0 = No overflow

H — Half-Carry Flag

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

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

I — Interrupt Mask

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

1 = Interrupts disabled
0 = Interrupts enabled

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

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

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

N — Negative flag

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

1 = Negative result
0 = Non-negative result

Z — Zero flag

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

1 = Zero result
0 = Non-zero result

C — Carry/Borrow Flag

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

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

6.5 Arithmetic/Logic Unit (ALU)

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

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

6.6 Low-Power Modes

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

6.6.1 Wait Mode

The 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

6.6.2 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.

6.7 CPU During Break Interrupts

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

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

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

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

6.8 Instruction Set Summary

6.9 Opcode Map

See **Table 6-2**.

Table 6-1. Instruction Set Summary

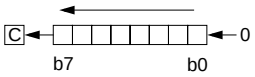
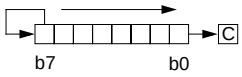
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X ADC *opr*,SP ADC *opr*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9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

Table 6-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
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
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

Table 6-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
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
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

Table 6-1. Instruction Set Summary (Continued)

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

Table 6-1. Instruction Set Summary (Continued)

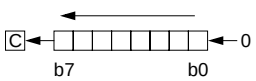
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X EOR *opr*,SP EOR *opr*,SP	Exclusive OR M with A	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9EE8 9ED8	ii 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 (Same as ASL) LSL ,X LSL *opr*,SP	Logical Shift Left  		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5

Table 6-1. Instruction Set Summary (Continued)

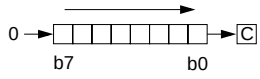
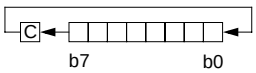
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LSR *opr* LSRA LSR*X* LSR *opr*,X LSR ,X LSR *opr*,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	4 1 1 4 3 5
MOV *opr,opr* MOV *opr*,X+ MOV #*opr,opr* MOV 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

Table 6-1. Instruction Set Summary (Continued)

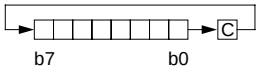
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ROR *opr* RORA RORX ROR *opr*,X ROR ,X ROR *opr*,SP	Rotate Right through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	4 1 1 4 3 5
RSP	Reset Stack Pointer	SP ← $FF	–	–	–	–	–	–	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 1; Pull (CCR) SP ← (SP) + 1; Pull (A) SP ← (SP) + 1; Pull (X) SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	↕	↕	↕	↕	↕	↕	INH	80		7
RTS	Return from Subroutine	SP ← SP + 1; Pull (PCH) SP ← SP + 1; Pull (PCL)	–	–	–	–	–	–	INH	81		4
SBC #*opr* SBC *opr* SBC *opr* SBC *opr*,X SBC *opr*,X SBC ,X SBC *opr*,SP SBC *opr*,SP	Subtract with Carry	A ← (A) – (M) – (C)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A2 B2 C2 D2 E2 F2 9EE2 9ED2	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
SEC	Set Carry Bit	C ← 1	–	–	–	–	–	1	INH	99		1
SEI	Set Interrupt Mask	I ← 1	–	–	1	–	–	–	INH	9B		2
STA *opr* STA *opr* STA *opr*,X STA *opr*,X STA ,X STA *opr*,SP STA *opr*,SP	Store A in M	M ← (A)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	B7 C7 D7 E7 F7 9EE7 9ED7	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 5
STHX *opr*	Store H:X in M	(M:M + 1) ← (H:X)	0	–	–	↕	↕	–	DIR	35	dd	4
STOP	Enable $\overline{\text{IRQ}}$ 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

Table 6-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
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
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

Table 6-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				

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

Table 6-2. 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

Low Byte of Opcode in Hexadecimal

MSB / LSB	0	High Byte of Opcode in Hexadecimal
0	5 BRSET0 3 DIR	Cycles Opcode Mnemonic Number of Bytes / Addressing Mode

Section 7. System Integration Module (SIM)

7.1 Contents

7.2 Introduction

This section describes the system integration module (SIM16, Version E), which supports up to 16 external and/or internal interrupts. Together with the CPU, the SIM controls all MCU activities. A block diagram of the SIM is shown in **Figure 7-1**. **Table 7-1** shows a summary of the SIM I/O registers. The SIM is a system state controller that coordinates CPU and exception timing. The SIM is responsible for:

- Bus clock generation and control for CPU and peripherals
- Stop/wait/reset/break entry and recovery
- Internal clock control
- Master reset control, including power-on reset (POR) and COP timeout
- Interrupt control:
- Acknowledge timing
- Arbitration control timing
- Vector address generation
- CPU enable/disable timing
- Modular architecture expandable to 128 interrupt sources
- All references to DMA mode operation in this chapter should be ignored.

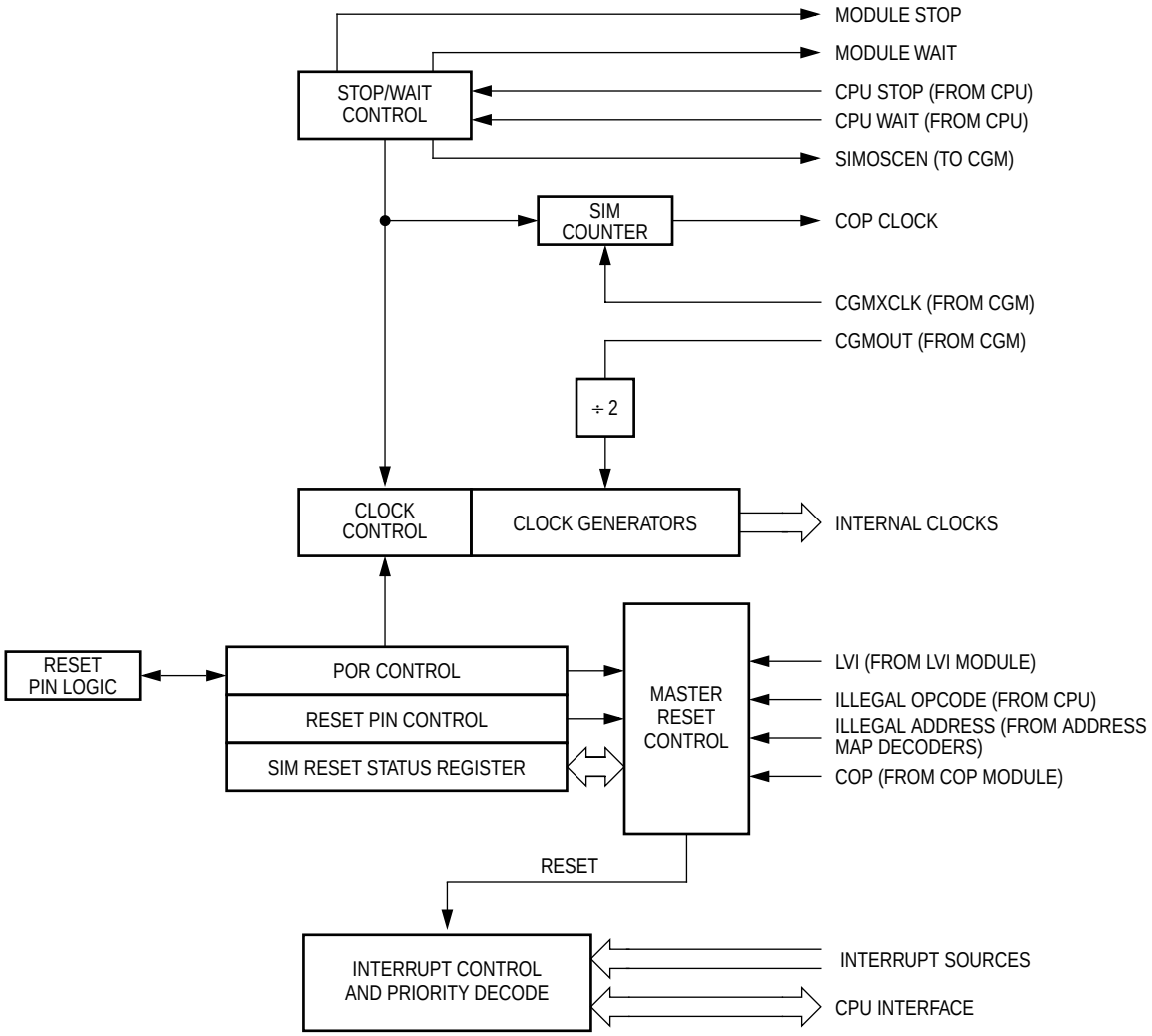


Figure 7-1. SIM Block Diagram

Table 7-1. SIM I/O Register Summary

Addr.	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$FE00	SIM Break Status Register (SBSR)	R	R	R	R	R	R	SBSW	R
$FE01	SIM Reset Status Register (SRSR)	POR	PIN	COP	ILOP	ILAD	0	LVI	0
$FE03	SIM Break Flag Control Register (SBFCR)	BCFE	0	0	0	0	0	0	0

R = Reserved for factory test

Table 7-2 shows the internal signal names used in this section.

Table 7-2. Signal Name Conventions

Signal Name	Description
CGMXCLK	Buffered version of OSC1 from clock generator module (CGM)
CGMVCLK	PLL output
CGMOUT	PLL-based or OSC1-based clock output from CGM module (Bus clock = CGMOUT divided by two)
IAB	Internal address bus
IDB	Internal data bus
PORRST	Signal from the power-on reset module to the SIM
IRST	Internal reset signal
R/$\overline{W}$	Read/write signal

7.3 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 7-2**. This clock can come from either an external oscillator or from the on-chip PLL. (See **Section 8. Clock Generation Module (CGMB)**.)

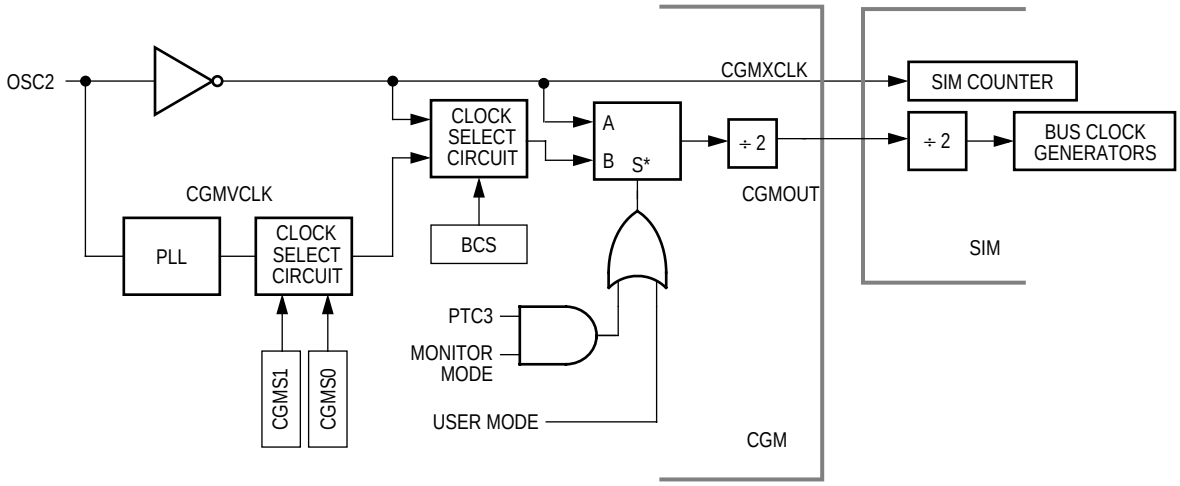


Figure 7-2. CGM Clock Signals

7.3.1 Bus Timing

In user mode, the internal bus clock is either from the crystal oscillator output (CGMXCLK) or from the PLL output (CGMVCLK). The clock first goes through a programmable divider (divided by 1, 2, 4 or 8) and followed by a divide-by-four divider. (See **Section 8. Clock Generation Module (CGMB)**.)

7.3.2 Clock Start-Up from POR or LVI Reset

When the power-on reset module or the low-voltage interrupt module generates a reset, the clocks to the CPU and peripherals are inactive and held in an inactive phase until after the 4096 CGMXCLK cycle POR timeout has completed. The IBUS clocks start upon completion of the timeout.

7.3.3 Clocks in Stop Mode and Wait Mode

Upon exit from stop mode (by an interrupt, break, 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. This timeout is selectable as 4096 or 32 CGMXCLK cycles. (See **7.7.2 Stop Mode**.)

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

7.4 Reset and System Initialization

The MCU has the following reset sources:

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

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

An internal reset clears the SIM counter (see **7.5 SIM Counter**), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR) (see **7.8 SIM Registers**).

7.4.1 External Pin Reset

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 7-3. PIN Bit Set Timing)**. **Figure 7-3** shows the relative timing.

Table 7-3. PIN Bit Set Timing

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

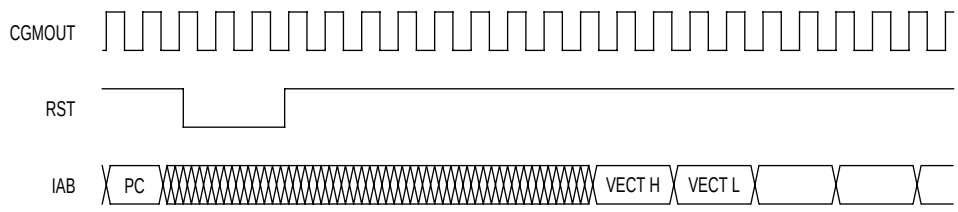


Figure 7-3. External Reset Timing

7.4.2 Active Resets from Internal Sources

SIM module in HC08 has the capability to drive the $\overline{\text{RST}}$ pin low when internal reset events occur.

All internal reset sources actively pull the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles (see **Figure 7-4. Internal Reset Timing)**. An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR (see **Figure 7-5. Sources of Internal Reset)**. Note that for LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM forces the $\overline{\text{RST}}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{\text{RST}}$ shown in **Figure 7-4**.

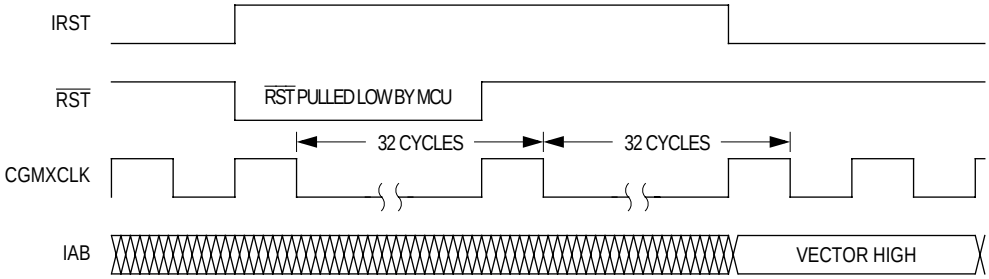


Figure 7-4. Internal Reset Timing

The COP reset is asynchronous to the bus clock.

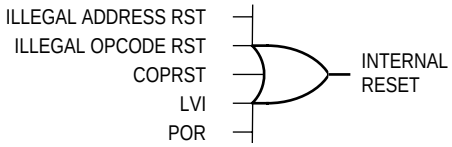


Figure 7-5. Sources of Internal Reset

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

7.4.2.1 Power-On Reset

When power is first applied to the MCU, the power-on reset module (POR) generates a pulse to indicate that power-on has occurred. The external reset pin ($\overline{RST}$) is held low while the SIM counter counts out 4096 CGMXCLK cycles. Sixty-four CGMXCLK cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur.

At power-on, the following events occur:

- A POR pulse is generated.
- The internal reset signal is asserted.
- The SIM enables CGMOUT.

- Internal clocks to the CPU and modules are held inactive for 4096 CGMXCLK cycles to allow stabilization of the oscillator.
- The $\overline{RST}$ pin is driven low during the oscillator stabilization time.
- The POR bit of the SIM reset status register (SRSR) is set and all other bits in the register are cleared.

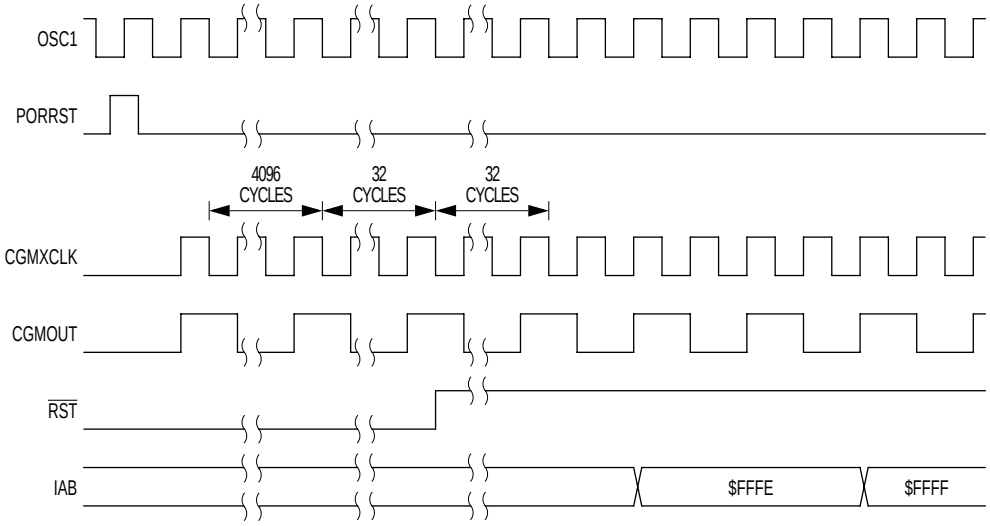


Figure 7-6. POR Recovery

7.4.2.2 Computer Operating Properly (COP) Reset

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

To prevent a COP module timeout, write any value to location $FFFF. Writing to location $FFFF clears the COP counter and bits 12 through 4 of the SIM counter. The SIM counter output, which occurs at least every $2^{13} - 2^4$ 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 OSC1 pin or the $\overline{IRQ1}/V_{PP}$ pin is held at $V_{DD} + V_{HI}$ 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 OSC1 or the $\overline{IRQ1}/V_{pp}$ pin. This prevents the COP from becoming disabled as a result of external noise. During a break state, $V_{DD} + V_{HI}$ on the OSC1 pin disables the COP module.

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

7.4.2.4 Illegal Address Reset

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

7.4.2.5 Low-Voltage Interrupt (LVI) Reset

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

7.5 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 13 bits long and is clocked by the falling edge of CGMXCLK.

7.5.1 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 clock generation module (CGM) to drive the bus clock state machine.

7.5.2 SIM Counter During Stop Mode Recovery

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

7.5.3 SIM Counter and Reset States

External reset has no effect on the SIM counter (see **7.7.2 Stop Mode**). The SIM counter is free-running after all reset states (see **7.4.2 Active Resets from Internal Sources** for counter control and internal reset recovery sequences).

7.6 Exception Control

Normally, sequential program execution can be changed in three different ways:

- Interrupts
  - Maskable hardware CPU interrupts
  - Non-maskable software interrupt instruction (SWI)
- Reset
- Break interrupts

7.6.1 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 RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. **Figure 7-7** shows interrupt entry timing. **Figure 7-9** shows interrupt recovery timing.

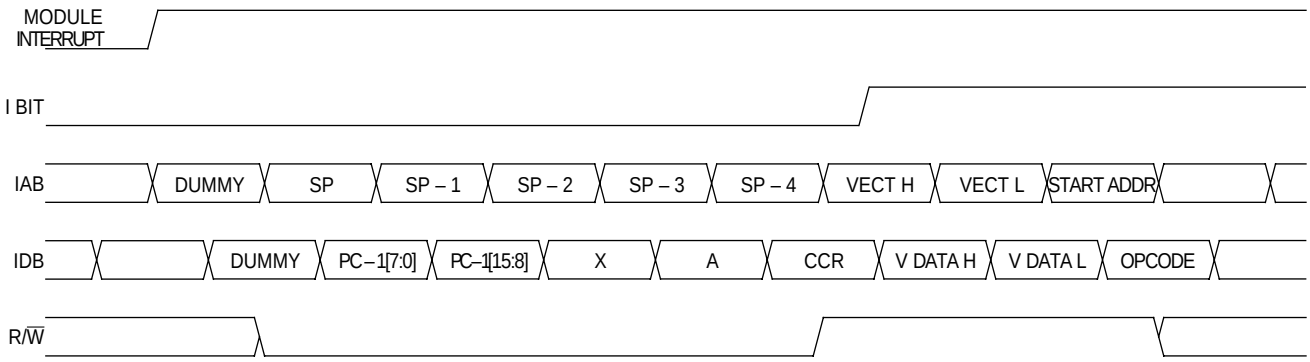


Figure 7-7 . Interrupt Entry

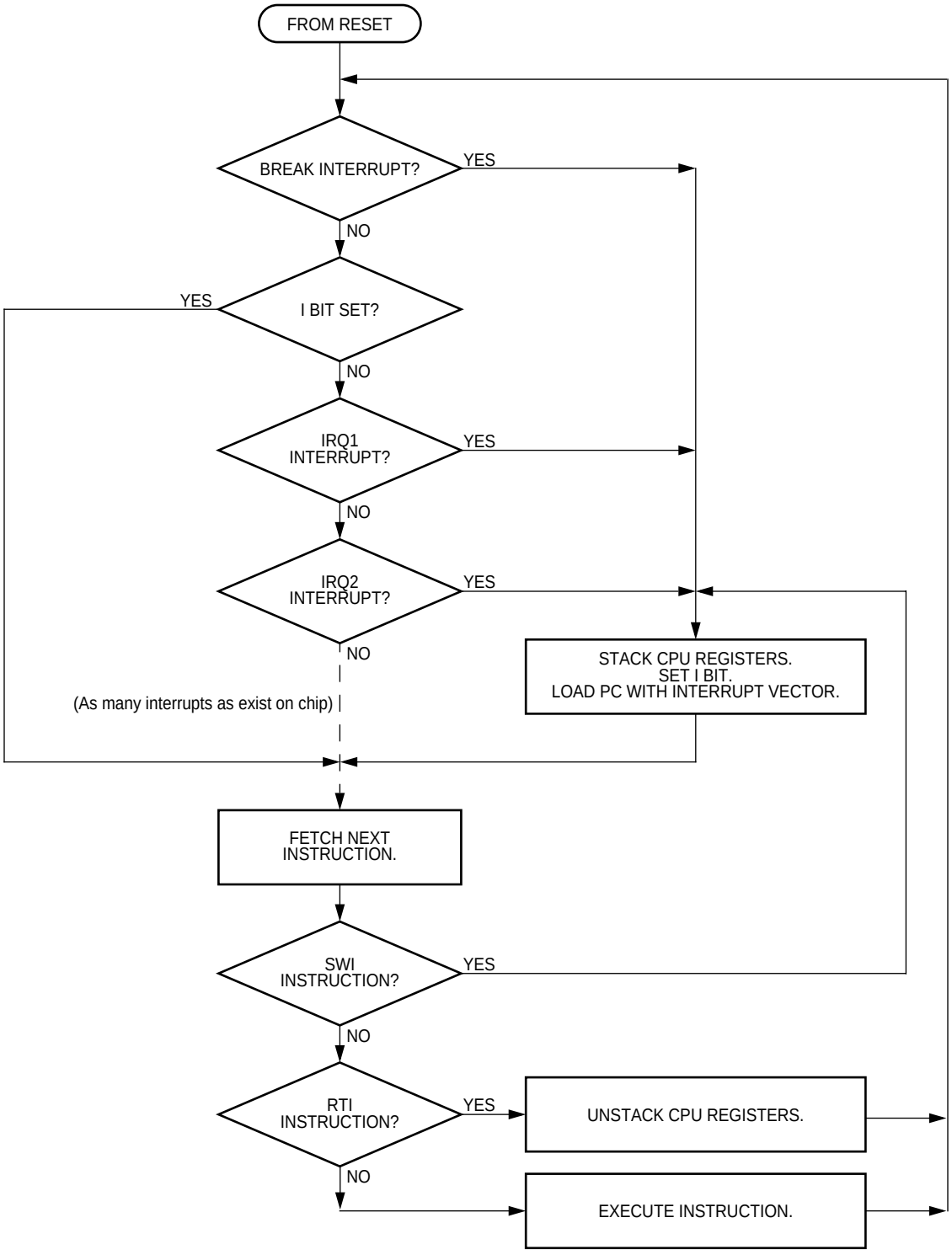


Figure 7-8. Interrupt Processing

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt may take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared). (See **Figure 7-9. Interrupt Recovery**.)

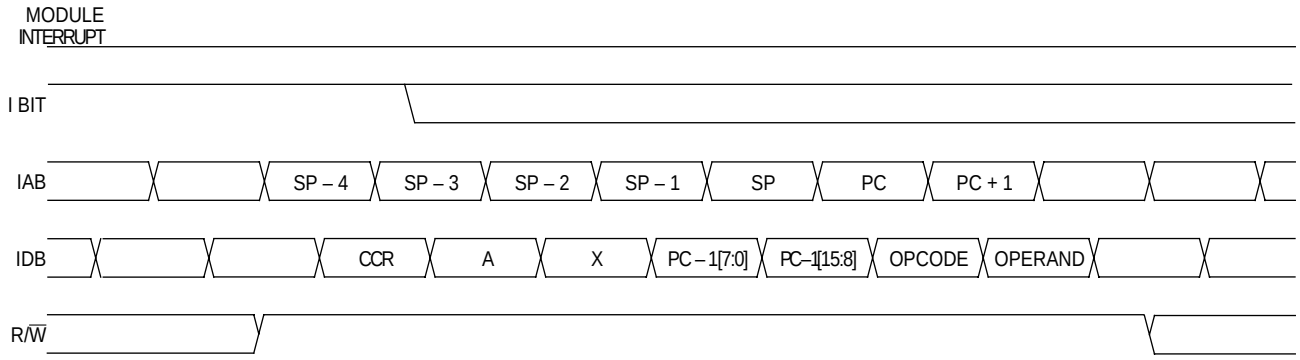


Figure 7-9. Interrupt Recovery

7.6.1.1 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 7-10** demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

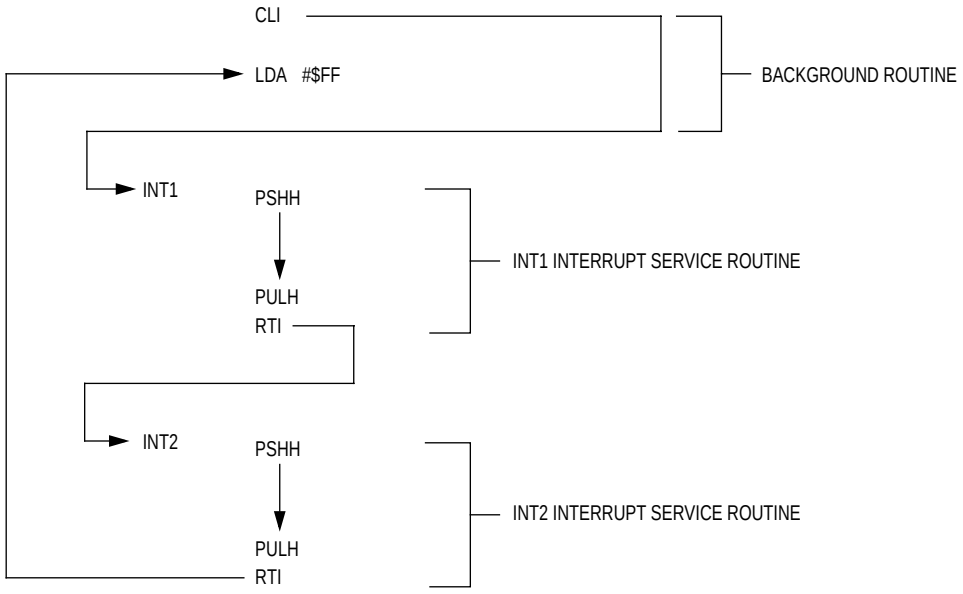


Figure 7-10 . Interrupt Recognition Example

The LDA opcode is pre-fetched by both the INT1 and INT2 RTI instructions. However, in the case of the INT1 RTI pre-fetch, this is a redundant operation.

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

7.6.1.2 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.*

7.6.2 Reset

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

7.6.3 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output (see **Section 21. Break Module (BREAK)**). 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.

7.6.4 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.

7.7 Low-Power Modes

Executing the WAIT or STOP instruction puts the MCU in a low-power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is

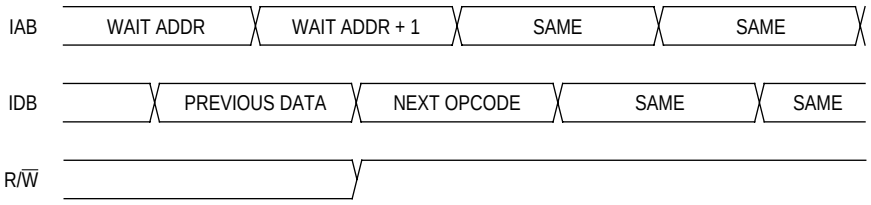
described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

7.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. **Figure 7-11** 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. In wait mode, the CPU clocks are inactive. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

Wait mode can also be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the mask option register is logic zero, 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 7-11. Wait Mode Entry Timing

Figure 7-12 and **Figure 7-13** show the timing for WAIT recovery.

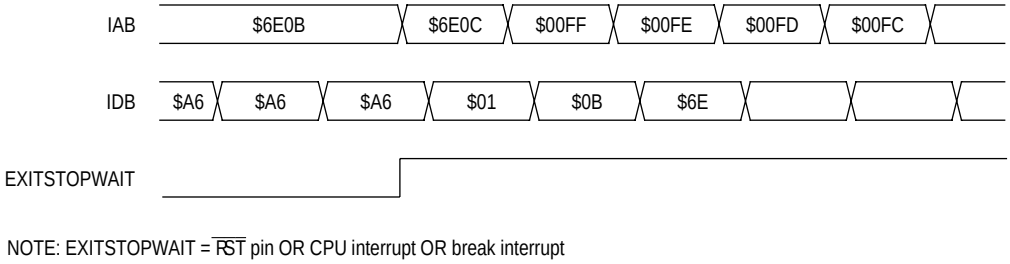


Figure 7-12. Wait Recovery from Interrupt or Break

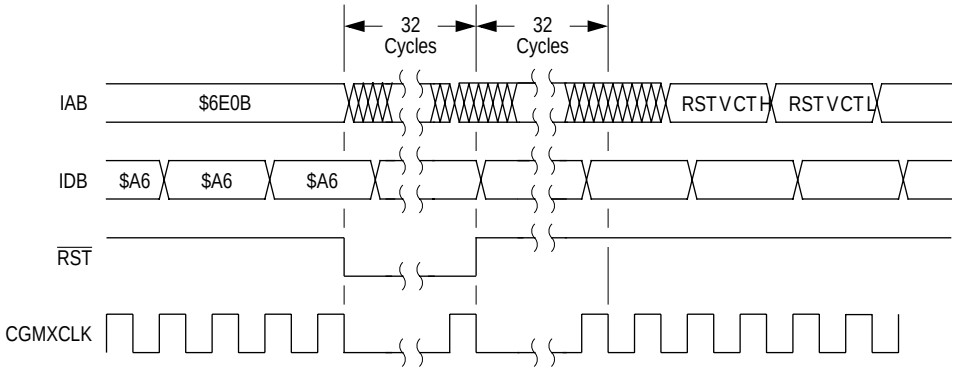


Figure 7-13. Wait Recovery from Internal Reset

7.7.2 Stop Mode

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

The SIM disables the clock generator module outputs (CGMOUT and CGMXCLK) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in the mask option register (MOR). If SSREC is set, stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32. This is ideal for applications using canned oscillators that do not require long start-up times from stop mode.

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

A break interrupt during stop mode sets the SIM break stop/wait bit (SBSW) in the SIM break status register (SBSR).

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 7-14** shows stop mode entry timing.

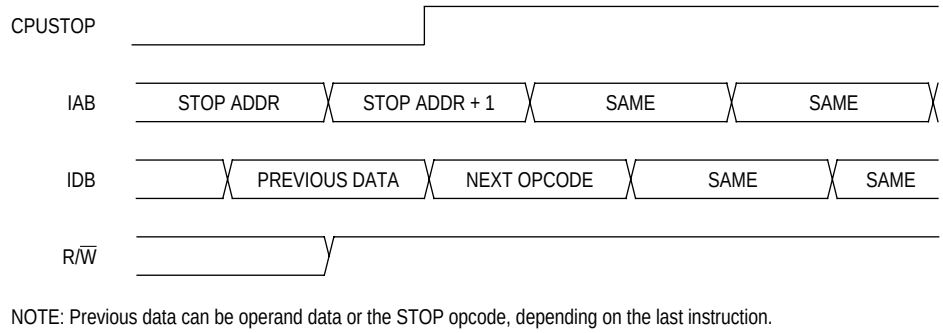


Figure 7-14. Stop Mode Entry Timing

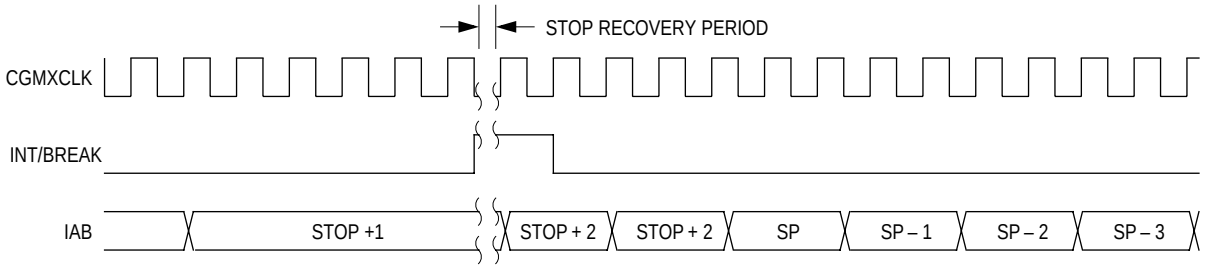


Figure 7-15. Stop Mode Recovery from Interrupt or Break

7.8 SIM Registers

The SIM has three memory mapped registers. **Table 7-4** shows the mapping of these registers.

Table 7-4. SIM Registers Summary

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

7.8.1 SIM Break Status Register (SBSR)

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

		Bit 7	6	5	4	3	2	1	Bit 0
SBSR $FE00	Read:	R	R	R	R	R	R	SBSW	R
	Write:							Note(1)	
	Reset:							0	

R = Reserved for factory test

1. Writing a logic zero clears SBSW.

Figure 7-16. SIM Break Status Register (SBSR)

SBSW — SIM Break Stop/Wait

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

1 = Stop mode or wait mode was exited by break interrupt
0 = Stop mode 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. The following code is an example of this. Writing zero to the SBSW bit clears it.

```
; This code works if the H register has been pushed onto the stack
; in the break service routine software. This code should be
; executed at the end of the break service routine software.

  HIBYTE   EQU    5

  LOBYTE   EQU    6

;          If not SBSW, do RTI

           BRCLR  SBSW,SBSR,    ; See if wait mode or stop mode
                  RETURN        ; was exited by break.

           TST    LOBYTE,SP     ; If RETURNLO is not zero,

           BNE    DOLO          ; then just decrement low byte.

           DEC    HIBYTE,SP     ; Else deal with high byte, too.

  DOLO     DEC    LOBYTE,SP     ; Point to WAIT/STOP opcode.

  RETURN   PULH                 ; Restore H register.
           RTI
```

7.8.2 SIM Reset Status Register (SRSR)

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

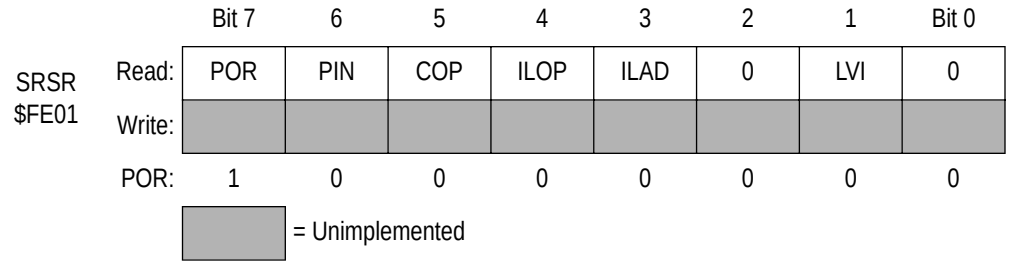


Figure 7-17. SIM Reset Status Register (SRSR)

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 SRSR

COP — Computer Operating Properly Reset Bit

1 = Last reset caused by COP counter

0 = POR or read of SRSR

ILOP — Illegal Opcode Reset Bit

1 = Last reset caused by an illegal opcode

0 = POR or read of SRSR

ILAD — Illegal Address Reset Bit (opcode fetches only)

1 = Last reset caused by an opcode fetch from an illegal address

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

7.8.3 SIM Break Flag Control Register (SBFCR)

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

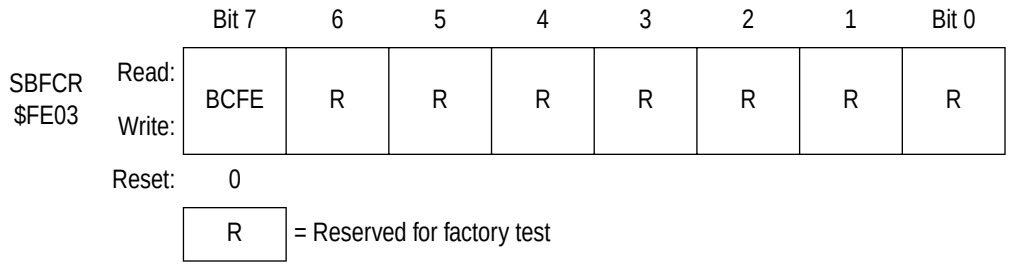


Figure 7-18. SIM Break Flag Control Register (SBFCR)

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 8. Clock Generation Module (CGMB)

8.1 Contents

8.2 Introduction

This section describes the clock generator module (CGM, Version B). The CGM generates the crystal clock signal, CGMXCLK, which operates at the frequency of the crystal. The CGM also generates two frequencies CGMOUTX and CGMOUT. CGMOUTX is based on the phase-locked loop (PLL) clock, CGMVCLK, divided by two. This is the clock for the DTMF/Melody generator module (see **Section 15. DTMF/Melody Generator Module**). CGMOUT is derived from either the crystal clock or CGMVCLK by using a programmable divider. This divider provides the user with options to operate the MCU in a reduced clock speed in order to conserve power consumption. The SIM derives the system clocks, including the bus clock, which is at a frequency of CGMOUT/2. The PLL is a fully functional frequency generator designed for use with crystals or ceramic resonators. The PLL can generate a 2.49MHz bus frequency by using a low cost 32.768KHz crystal.

8.3 Features

Features of the CGMB include the following:

- Phase-Locked Loop with Output Frequency in Integer Multiples of an Integer Dividend of the Crystal Reference
- Low Frequency Crystal Operation with Low Power Operation and High-Output Frequency Resolution
- Programmable Reference Divider For Greater Resolution
- Programmable Prescaler for Power-of-Two Increases in Frequency
- Programmable Hardware Voltage-Controlled Oscillator (VCO) for Low-Jitter Operation
- Automatic Bandwidth Control Mode for Low-Jitter Operation
- Automatic Frequency Lock Detector
- CPU Interrupt on Entry or Exit from Locked Condition

8.4 Functional Description

The CGMB consists of three major sub-modules:

- Crystal oscillator circuit — The crystal oscillator circuit generates the constant crystal frequency clock, CGMXCLK.
- Phase-locked loop (PLL) — The PLL generates the programmable VCO frequency clock CGMVCLK.
- Base clock selector circuit — This software-controlled circuit selects either CGMXCLK divided by two or the VCO clock, CGMVCLK, divided by the programmable divider and further divided by two as the base clock, CGMOUT. The SIM derives the system clocks from CGMOUT.

Figure 8-1 shows the structure of the CGM.

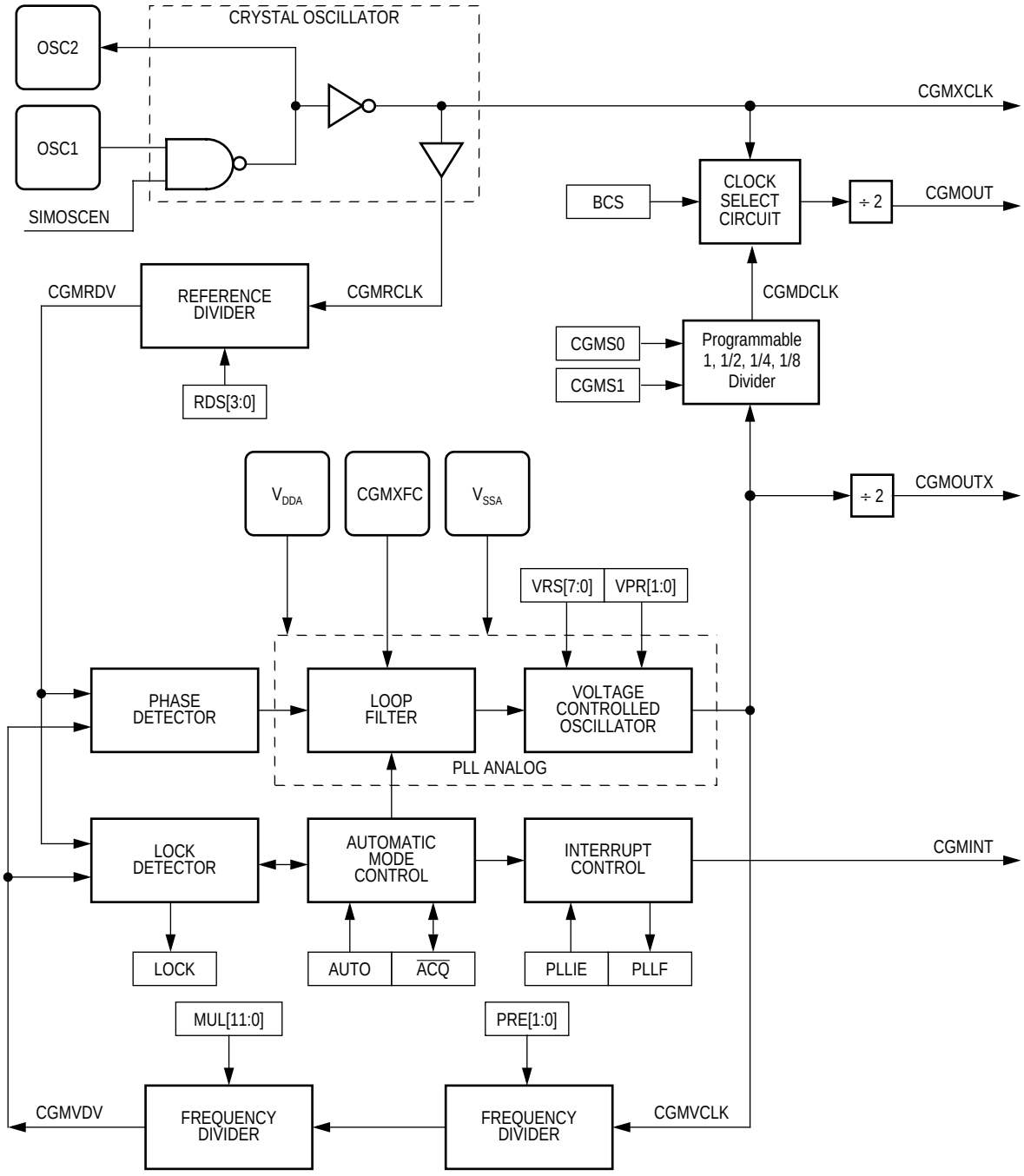


Figure 8-1. CGMB Block Diagram

8.4.1 Crystal Oscillator Circuit

The crystal oscillator circuit consists of an inverting amplifier and an external crystal. The OSC1 pin is the input to the amplifier and the OSC2 pin is the output. The SIMOSCEN signal from the system integration module (SIM) enables the crystal oscillator circuit.

The CGMXCLK signal is the output of the crystal oscillator circuit and runs at a rate equal to the crystal frequency. CGMXCLK is then buffered to produce CGMRCLK, the PLL reference clock.

CGMXCLK can be used by other modules which require precise timing for operation. The duty cycle of CGMXCLK is not guaranteed to be 50% and depends on external factors, including the crystal and related external components.

An externally generated clock also can feed the OSC1 pin of the crystal oscillator circuit. Connect the external clock to the OSC1 pin and let the OSC2 pin float.

8.4.2 Phase-Locked Loop Circuit (PLL)

The PLL is a frequency generator that can operate in either acquisition mode or tracking mode, depending on the accuracy of the output frequency. The PLL can change between acquisition and tracking modes either automatically or manually.

8.4.3 PLL Circuits

The PLL consists of the following circuits:

- Voltage-controlled oscillator (VCO)
- Reference divider
- Frequency prescaler
- Modulo VCO frequency divider
- Phase detector
- Loop filter
- Lock detector

The operating range of the VCO is programmable for a wide range of frequencies and for maximum immunity to external noise, including supply and CGM/XFC noise. The VCO frequency is bound to a range from roughly one-half to twice the center-of-range frequency, f_{VRS}. Modulating the voltage on the CGM/XFC pin changes the frequency within this range. By design, f_{VRS} is equal to the nominal center-of-range frequency, f_{NOM}, (38.4 KHz) times a linear factor L and a power-of-two factor E, or $(L \times 2^E)f_{NOM}$.

CGMRCLK is the PLL reference clock, a buffered version of CGMXCLK. CGMRCLK runs at a frequency, f_{RCLK}, and is fed to the PLL through a programmable modulo reference divider, which divides f_{RCLK} by a factor R. This feature allows frequency steps of higher resolution. The divider's output is the final reference clock, CGMRDV, running at a frequency $f_{RDV} = f_{RCLK}/R$.

The VCO's output clock, CGMVCLK, running at a frequency f_{VCLK}, is fed back through a programmable prescaler divider and a programmable modulo divider. The prescaler divides the VCO clock by a power-of-two factor P and the modulo divider reduces the VCO clock by a factor, N. The dividers' output is the VCO feedback clock, CGMVDV, running at a frequency $f_{VDV} = f_{VCLK}/(N \times 2^P)$. (See **8.4.6 Programming the PLL**.)

The phase detector then compares the VCO feedback clock, CGMVDV, with the final reference clock, CGMRDV. A correction pulse is generated based on the phase difference between the two signals. The loop filter then slightly alters the DC voltage on the external capacitor connected to CGM/XFC based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, described in (see **8.4.4 Acquisition and Tracking Modes**). The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

The lock detector compares the frequencies of the VCO feedback clock, CGMVDV, and the final reference clock, CGMRDV. Therefore, the speed of the lock detector is directly proportional to the final reference frequency f_{RDV}. The circuit determines the mode of the PLL and the lock condition based on this comparison.

8.4.4 Acquisition and Tracking Modes

The PLL filter is manually or automatically configurable into one of two operating modes:

- Acquisition mode — In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL start-up or when the PLL has suffered a severe noise hit and the VCO frequency is far off the desired frequency. When in acquisition mode, the $\overline{ACQ}$ bit is clear in the PLL bandwidth control register. (See **8.6.2 PLL Bandwidth Control Register (PBWC)**.)
- Tracking mode — In tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source (see **8.4.8 Base Clock Selector Circuit**). The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{ACQ}$ bit is set.

8.4.5 Manual and Automatic PLL Bandwidth Modes

The PLL can change the bandwidth or operational mode of the loop filter manually or automatically. Automatic mode is recommended for most users.

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT (see **8.6.2 PLL Bandwidth Control Register (PBWC)**). If PLL interrupts are enabled, the software can wait for a PLL interrupt request and then check the LOCK bit. If interrupts are disabled, software can poll the LOCK bit continuously (during PLL start-up, usually) or at periodic intervals. In either case, when the LOCK bit is set, the VCO clock is safe to use as the source for the base clock (see **8.4.8 Base Clock Selector Circuit**). If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe

noise hit and the software must take appropriate action, depending on the application (See **8.7 Interrupts** for information and precautions on using interrupts). The following conditions apply when the PLL is in automatic bandwidth control mode:

- The $\overline{ACQ}$ bit (see **8.6.2 PLL Bandwidth Control Register (PBWC)**) is a read-only indicator of the mode of the filter (see **8.4.4 Acquisition and Tracking Modes**).
- The $\overline{ACQ}$ bit is set when the VCO frequency is within a certain tolerance Δ_{TRK} and is cleared when the VCO frequency is out of a certain tolerance Δ_{UNT}. (See **8.9 Acquisition/Lock Time Specifications**.)
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance Δ_{LOCK} and is cleared when the VCO frequency is out of a certain tolerance Δ_{UNL}. (See **8.9 Acquisition/Lock Time Specifications**.)
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. (See **8.6.1 PLL Control Register (PCTL)**.)

The PLL also may operate in manual mode (AUTO = 0). Manual mode is used by systems that do not require an indicator of the lock condition for proper operation. Such systems typically operate well below f_{BUSMAX}. The following conditions apply when in manual mode:

- $\overline{ACQ}$ is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the $\overline{ACQ}$ bit must be clear.
- Before entering tracking mode ($\overline{ACQ}$ = 1), software must wait a given time, t_{ACQ} (see **8.9 Acquisition/Lock Time Specifications**), after turning on the PLL by setting PLLON in the PLL control register (PCTL).
- Software must wait a given time, t_{AL}, after entering tracking mode before selecting the PLL as the clock source to CGMOUT (BCS = 1).

- The LOCK bit is disabled.
- CPU interrupts from the CGMB are disabled.

8.4.6 Programming the PLL

The following procedure shows how to program the PLL.

NOTE: *The round function in the following equations means that the real number should be rounded to the nearest integer number.*

1. Choose the desired bus frequency, f_{BUSDES}.
2. Calculate the desired VCO frequency (four times the desired bus frequency).

$$f_{VCLKDES} = 4 \times f_{BUSDES}$$

3. Choose a practical PLL (crystal) reference frequency, f_{RCLK}, and the reference clock divider, R.
   Frequency errors to the PLL are corrected at a rate of f_{RCLK}/R. For stability and lock time reduction this rate must be as fast as possible. The VCO frequency must be an integer multiple of this rate. The relationship between the VCO frequency f_{VCLK} and the reference frequency f_{RCLK} is

$$f_{VCLK} = \frac{2^P N}{R}(f_{RCLK})$$

   P, the power of two multiplier and N, the range multiplier are integers.
   In cases where desired bus frequency has some tolerance, choose f_{RCLK} to a value determined either by other module requirements (i.e. modules which are clocked by CGMXCLK), cost requirements, or ideally, as high as the specified range allows. (See **Section 22. Electrical Specifications**.) Choose the reference divider R = 1. After choosing N, and P (as shown below) The actual bus frequency can be determined using equation 2 above.
   When the tolerance on the bus frequency is tight, choose f_{RCLK} to

an integer divisor of f_{BUSDES}, and R = 1. If f_{RCLK} cannot meet this requirement, use the following equation to solve for R with practical choices of f_{RCLK}, and choose the f_{RCLK} that gives the lowest R.

$$R = \text{round}\left[R_{MAX} \times \left[\left(\frac{f_{VCLKDES}}{f_{RCLK}}\right) - \text{integer}\left(\frac{f_{VCLKDES}}{f_{RCLK}}\right)\right]\right]$$

4. Select a VCO frequency multiplier, N.

$$N = \text{round}\left(\frac{R \times f_{VCLKDES}}{f_{RCLK}}\right)$$

Reduce N/R to the lowest possible R.

5. If N is < N_{max}, use P = 0. If N > N_{max}, choose P using the table below:

Current N value	P
$0 < N \leq N_{max}$	0
$N_{max} < N \leq N_{max} \times 2$	1
$N_{max} \times 2 < N \leq N_{max} \times 4$	2
$N_{max} \times 4 < N \leq N_{max} \times 8$	3

Then recalculate N:

$$N = \text{round}\left(\frac{R \times f_{VCLKDES}}{f_{RCLK} \times 2^P}\right)$$

6. Calculate and verify the adequacy of the VCO and bus frequencies f_{VCLK} and f_{BUS}.

$$f_{VCLK} = \left(2^P \times \frac{N}{R}\right) \times f_{RCLK}$$

$$f_{BUS} = \frac{f_{VCLK}}{4}$$

7. Select the VCO's power-of-two range multiplier E, according to the following table:

Frequency Range	E
$0 < f_{VCLK} \leq f_{VRSMAX}/4$	0
$f_{VRSMAX}/4 < f_{VCLK} \leq f_{VRSMAX}/2$	1
$f_{VRSMAX}/2 < f_{VCLK} \leq f_{VRSMAX}$	2

NOTE: Do not program E to a value of 3.

8. Select a VCO linear range multiplier, L, where f_{NOM} = 38.4 KHz

$$L = \text{round}\left(\frac{f_{VCLK}}{2^E \times f_{NOM}}\right)$$

9. Calculate and verify the adequacy of the VCO programmed center-of-range frequency f_{VRS}. The center-of-range frequency is the midpoint between the minimum and maximum frequencies attainable by the PLL.

$$f_{VRS} = (L \times 2^E) f_{NOM}$$

For proper operation,

$$|f_{VRS} - f_{VCLK}| \leq \frac{f_{NOM} \times 2^E}{2}$$

10. Verify the choice of P, R, N, E, and L by comparing f_{VCLK} to f_{VRS} and $f_{VCLKDES}$. For proper operation, f_{VCLK} must be within the application's tolerance of $f_{VCLKDES}$, and f_{VRS} must be as close as possible to f_{VCLK}.

NOTE: *Exceeding the recommended maximum bus frequency or VCO frequency can crash the MCU.*

11. Program the PLL registers accordingly:
    a. In the PRE bits of the PLL Control Register (PCTL), program the binary equivalent of P.

b. In the VPR bits of the PLL Control Register (PCTL), program the binary equivalent of E.

c. In the PLL Multiplier Select Register Low (PMSL) and the PLL Multiplier Select Register High (PMSH), program the binary equivalent of N.

d. In the PLL VCO Range Select Register (PVRS), program the binary coded equivalent of L.

e. In the PLL Reference Divider Select Register (PRDS), program the binary coded equivalent of R.

Numeric Example (Numbers are in Hexadecimal notation):

Table 8-1. Numeric Example

f_{BUS}	f_{RCLK}	E	P	N	L	R
8MHz	32.768KHz	2	0	3d1	d0	1

8.4.7 Special Programming Exceptions

The programming method described in (see **8.4.6 Programming the PLL**) does not account for three possible exceptions. A value of zero for R, N, or L is meaningless when used in the equations given. To account for these exceptions:

- A zero value for R or N is interpreted exactly the same as a value of one. A zero value for L disables the PLL and prevents its selection as the source for the base clock. (See **8.4.8 Base Clock Selector Circuit**.)

8.4.8 Base Clock Selector Circuit

This circuit is used to select either the crystal clock, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the base clock, CGMOUT. The two input clocks go through a transition control circuit that waits up to three CGMXCLK cycles and three CGMVCLK cycles to change from one clock source to the other. During this time, CGMOUT is held in stasis. The output of the transition control circuit is then divided by two

to correct the duty cycle. Therefore, the bus clock frequency, which is one-half of the base clock frequency, is one-fourth the frequency of the selected clock (CGMXCLK or CGMVCLK).

The BCS bit in the PLL control register (PCTL) selects which clock drives CGMOUT. The VCO clock cannot be selected as the base clock source if the PLL is not turned on. The PLL cannot be turned off if the VCO clock is selected. The PLL cannot be turned on or off simultaneously with the selection or deselection of the VCO clock. The VCO clock also cannot be selected as the base clock source if the factor L is programmed to a zero. This value would set up a condition inconsistent with the operation of the PLL, so that the PLL would be disabled and the crystal clock would be forced as the source of the base clock.

8.4.9 CGMB External Connections

In its typical configuration, the CGMB requires six external components. Five of these are for the crystal oscillator and one is for the PLL.

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in **Figure 8-2**. The figure shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

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

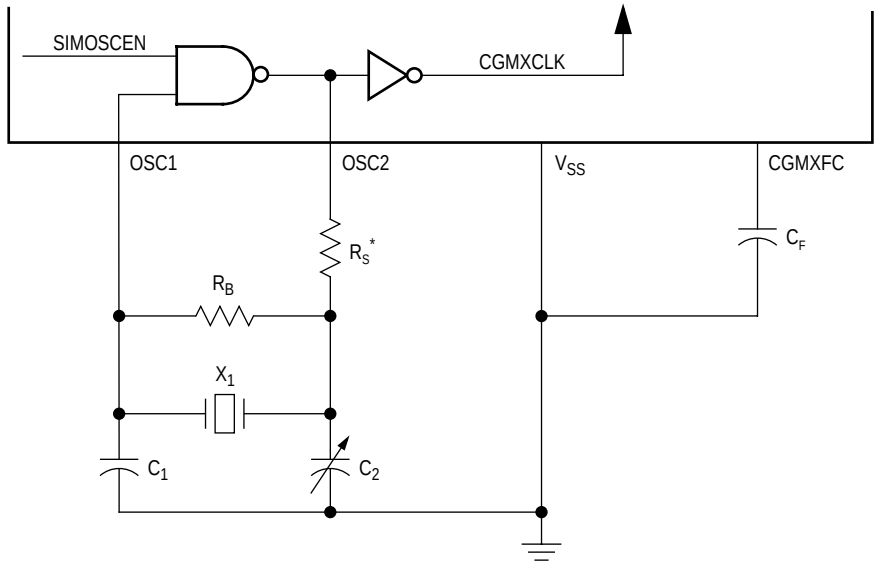
The series resistor (R_S) is included in the diagram to follow strict Pierce oscillator guidelines and may not be required for all ranges of operation, especially with high frequency crystals. Refer to the crystal manufacturer's data for more information.

Figure 8-2 also shows the external component for the PLL:

- Filter capacitor, C_F

Routing should be done with great care to minimize signal cross talk and noise.

See **Section 22. Electrical Specifications** for capacitor and resistor values.



*R_S can be zero (shorted) when used with higher-frequency crystals. Refer to manufacturer's data.

Figure 8-2. CGMB External Connections

8.5 I/O Signals

The following paragraphs describe the CGMB I/O signals.

8.5.1 Crystal Amplifier Input Pin (OSC1)

The OSC1 pin is an input to the crystal oscillator amplifier.

8.5.2 Crystal Amplifier Output Pin (OSC2)

The OSC2 pin is the output of the crystal oscillator inverting amplifier.

8.5.3 External Filter Capacitor Pin (CGMXFC)

The CGMXFC pin is required by the loop filter to filter out phase corrections. A small external capacitor is connected to this pin.

NOTE: *To prevent noise problems, C_F should be placed as close to the CGMXFC pin as possible, with minimum routing distances and no routing of other signals across the C_F connection.*

8.5.4 Buffered Crystal Clock Output (CGMVOUT)

CGMVOUT buffers the OSC1 clock for external use.

8.5.5 CGMVSEL

CGMVSEL must be tied low or floated.

8.5.6 Oscillator Enable Signal (SIMOSCEN)

The SIMOSCEN signal comes from the system integration module (SIM) and enables the oscillator and PLL.

8.5.7 Crystal Output Frequency Signal (CGMXCLK)

CGMXCLK is the crystal oscillator output signal. It runs at the full speed of the crystal (f_{XCLK}) and comes directly from the crystal oscillator circuit. **Figure 8-2** shows only the logical relation of CGMXCLK to OSC1 and OSC2 and may not represent the actual circuitry. The duty cycle of CGMXCLK is unknown and may depend on the crystal and other external factors. Also, the frequency and amplitude of CGMXCLK can be unstable at start-up.

8.5.8 CGMB Base Clock Output (CGMOUT)

CGMOUT is the clock output of the CGMB. This signal goes to the SIM, which generates the MCU clocks. CGMOUT is a 50% duty cycle clock running at twice the bus frequency. CGMOUT is software programmable

to be either the oscillator output, CGMXCLK, divided by two or the VCO clock, CGMVCLK, divided by the programmable divider and further divided by two.

8.5.9 CGMB CPU Interrupt (CGMINT)

CGMINT is the interrupt signal generated by the PLL lock detector.

8.6 CGMB Registers

The following registers control and monitor operation of the CGMB:

- PLL control register (PCTL) (see **8.6.1 PLL Control Register (PCTL)**)
- PLL bandwidth control register (PBWC) (see **8.6.2 PLL Bandwidth Control Register (PBWC)**)
- PLL Multiplier Select Register High (PMSH) (see **8.6.3 PLL Multiplier Select Register High (PMSH)**)
- PLL Multiplier Select Register Low (PMSL) (see **8.6.4 PLL Multiplier Select Register Low (PMSL)**)
- PLL VCO Range Select Register (PVRS) (see **8.6.5 PLL VCO Range Select Register (PMRS)**)
- PLL Reference Divider Select Register (PRDS) (see **8.6.6 PLL Reference Divider Select Register (PMDS)**)

		Bit 7	6	5	4	3	2	1	Bit 0
PCTL $0036	Read:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
	Write:								
PBWC $0037	Read:	AUTO	LOCK	$\overline{\text{ACQ}}$	0	0	0	0	COE
	Write:								
PMSH $0038	Read:	0	0	0	0	MUL11	MUL10	MUL9	MUL8
	Write:								
PMSL $0039	Read:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
	Write:								
PMRS $003A	Read:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
	Write:								
PMDS $003B	Read:	0	0	0	0	RDS3	RDS2	RDS1	RDS0
	Write:								
PCKS $003C	Read:	0	0	0	0	0	0	CGMS1	CGMS0
	Write:								

= Unimplemented

NOTES:

1. When AUTO = 0, PLLIE is forced clear and is read-only.

Figure 8-3. CGMB Registers

8.6.1 PLL Control Register (PCTL)

The PLL control register contains the interrupt enable and flag bits, the on/off switch, the base clock selector bit, the prescaler bits, and the VCO power of two range selector bits.

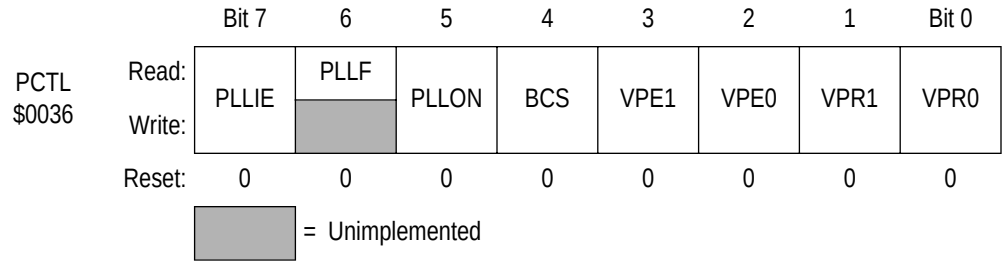


Figure 8-4. PLL Control Register (PCTL)

PLLIE — PLL Interrupt Enable Bit

This read/write bit enables the PLL to generate an interrupt request when the LOCK bit toggles, setting the PLL flag, PLLF. When the AUTO bit in the PLL bandwidth control register (PBWC) is clear, PLLIE cannot be written and reads as logic zero. Reset clears the PLLIE bit.

1 = PLL interrupts enabled
0 = PLL interrupts disabled

PLLF — PLL Interrupt Flag Bit

This read-only bit is set whenever the LOCK bit toggles. PLLF generates an interrupt request if the PLLIE bit also is set. PLLF always reads as logic zero when the AUTO bit in the PLL bandwidth control register (PBWC) is clear. Clear the PLLF bit by reading the PLL control register. Reset clears the PLLF bit.

1 = Change in lock condition
0 = No change in lock condition

NOTE: *Do not inadvertently clear the PLLF bit. Any read or read-modify-write operation on the PLL control register clears the PLLF bit.*

PLLON — PLL On Bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1) (see **8.4.8 Base Clock Selector Circuit**). Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

BCS — Base Clock Select Bit

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, divided by the programmable divider, CGMDCLK as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis (see **8.4.8 Base Clock Selector Circuit**). Reset clear the BCS bit.

1 = CGMDCLK divided by two drives CGMOUT
0 = CGMXCLK divided by two drives CGMOUT

NOTE: *PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMVCLK requires two writes to the PLL control register. (See **8.4.8 Base Clock Selector Circuit**.)*

PRE1 and PRE0 — Prescaler program bits

These read/write bits control a prescaler that selects the prescaler power-of-two multiplier P (see 8.4.3 and **8.4.6 Programming the PLL**). PRE1:PRE0 cannot be written when the PLLON bit is set. Reset clears these bits.

VPR[1:0] — VCO power-of-two range select bits

These read/write bits control the VCO's hardware power-of-two range multiplier E that, in conjunction with L (see 8.4.3, **8.4.6 Programming the PLL**, and **8.6.5 PLL VCO Range Select Register (PMRS)**), controls the hardware center-of-range frequency f_{VRS}. VPR1:VPR0 cannot be written when the PLLON bit is set. Reset clears these bits.

8.6.2 PLL Bandwidth Control Register (PBWC)

The PLL bandwidth control register does the following:

- Selects automatic or manual (software-controlled) bandwidth control mode
- Indicates when the PLL is locked
- In automatic bandwidth control mode, indicates when the PLL is in acquisition or tracking mode
- In manual operation, forces the PLL into acquisition or tracking mode

		Bit 7	6	5	4	3	2	1	Bit 0
PBWC $0037	Read:	AUTO	LOCK	$\overline{ACQ}$	0	0	0	0	COE
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 8-5. PLL Band Width Control Register (PBWC)

AUTO — Automatic Bandwidth Control Bit

This read/write bit selects automatic or manual bandwidth control. When initializing the PLL for manual operation (AUTO = 0), clear the $\overline{ACQ}$ bit before turning on the PLL. Reset clears the AUTO bit.

1 = Automatic bandwidth control
0 = Manual bandwidth control

LOCK — Lock Indicator Bit

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock, CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as logic zero and has no meaning. The write one function of this bit is reserved for test, so this bit must ALWAYS be written a zero. Reset clears the LOCK bit.

1 = VCO frequency correct or locked
0 = VCO frequency incorrect or unlocked

$\overline{ACQ}$ — Acquisition Mode Bit

When the AUTO bit is set, $\overline{ACQ}$ is a read-only bit that indicates whether the PLL is in acquisition mode or tracking mode. When the AUTO bit is clear, $\overline{ACQ}$ is a read/write bit that controls whether the PLL is in acquisition or tracking mode.

In automatic bandwidth control mode (AUTO = 1), the last-written value from manual operation is stored in a temporary location and is recovered when manual operation resumes. Reset clears this bit, enabling acquisition mode.

1 = Tracking mode
0 = Acquisition mode

8.6.3 PLL Multiplier Select Register High (PMSH)

The PLL Multiplier Select Register High contains the programming information for the high byte of the modulo feedback divider.

		Bit 7	6	5	4	3	2	1	Bit 0
PMSH $0038	Read:	0	0	0	0	MUL11	MUL10	MUL9	MUL8
	Write:								
	Reset:	0	1	0	0	0	0	0	0

= Unimplemented

Figure 8-7. PLL Multiplier Select Register High (PMSH)

MUL[11:8] — Multiplier select bits

These read/write bits control the high byte of the modulo feedback divider that selects the VCO frequency multiplier N. (see **8.4.3 PLL Circuits**) and (see **8.4.6 Programming the PLL**). A value of $0000 in the Multiplier Select Registers configures the modulo feedback divider the same as a value of $0001. Reset initializes the registers to $0040, for a default multiply value of 64.

NOTE: *The multiplier select bits have built in protection such that they cannot be written when the PLL is on (PLLON = 1).*

PMSH[7:4] — Unimplemented bits

These bits have no function and always read as logic zeros.

8.6.4 PLL Multiplier Select Register Low (PMSL)

The PLL multiplier select register low contains the programming information for the low byte of the modulo feedback divider.

		Bit 7	6	5	4	3	2	1	Bit 0
PMSL $0039	Read: Write:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
	Reset:	0	1	0	0	0	0	0	0

Figure 8-8. PLL Multiplier Select Register Low (PMSL)

MUL[7:0] — Multiplier select bits

These read/write bits control the low byte of the modulo feedback divider that selects the VCO frequency multiplier, N.(see **8.4.3 PLL Circuits**) and (see **8.4.6 Programming the PLL**). MUL[7:0] cannot be written when the PLLON bit in the PCTL is set. A value of $0000 in the Multiplier Select Registers configures the modulo feedback divider the same as a value of $0001. Reset initializes the register to $40, for a default multiply value of 64.

NOTE: *The multiplier select bits have built-in protection such that they cannot be written when the PLL is on (PLLON = 1).*

8.6.5 PLL VCO Range Select Register (PMRS)

The PLL VCO Range Select Register contains the programming information required for the hardware configuration of the VCO.

		Bit 7	6	5	4	3	2	1	Bit 0
PMRS $003A	Read: / Write:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
	Reset:	0	1	0	0	0	0	0	0

Figure 8-9. PLL VCO Range Select Register (PMRS)

VRS[7:0] - VCO range select bits

These read/write bits control the hardware center-of-range linear multiplier L which, in conjunction with E (see **8.4.3 PLL Circuits**, **8.4.6 Programming the PLL**, and **8.6.1 PLL Control Register (PCTL)**), controls the hardware center-of-range frequency, f_{VRS}. VRS[7:0] cannot be written when the PLLON bit in the PCTL is set (see **8.4.7 Special Programming Exceptions**). A value of $00 in the VCO Range Select Register disables the PLL and clears the BCS bit in the PLL Control Register (PCTL) (see **8.4.8 Base Clock Selector Circuit**) and **8.4.7 Special Programming Exceptions**). Reset initializes the register to $40 for a default range multiply value of 64.

NOTE: *The VCO range select bits have built in protection such that they cannot be written when the PLL is on (PLLON = 1) and such that the VCO clock cannot be selected as the source of the base clock (BCS=1) if the VCO range select bits are all clear.*

NOTE: *The PLL VCO Range Select Register must be programmed correctly. Incorrect programming may result in failure of the PLL to achieve lock.*

8.6.6 PLL Reference Divider Select Register (PMDS)

The PLL Reference Divider Select Register contains the programming information for the modulo reference divider.

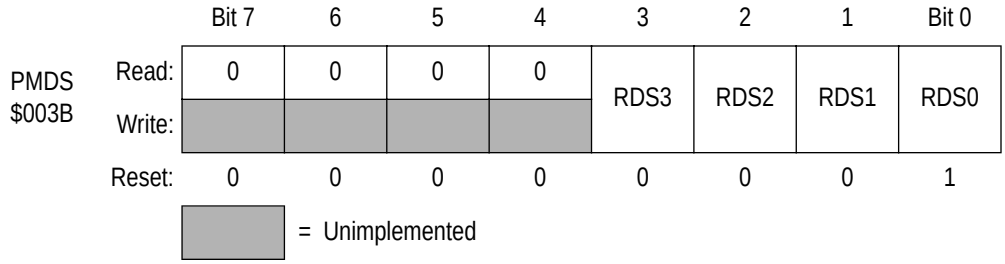


Figure 8-10. PLL Reference Divider Select Register (PMDS)

RDS[3:0] — Reference Divider Select Bits

These read/write bits control the modulo reference divider that selects the reference division factor R (see **8.4.3 PLL Circuits** and **8.4.6 Programming the PLL**). RDS[7:0] cannot be written when the PLLON bit in the PCTL is set. A value of $00 in the Reference Divider Select Register configures the reference divider the same as a value of $01 (see **8.4.7 Special Programming Exceptions**). Reset initializes the register to $01, for a default divide value of 1.

NOTE: *The Reference Divider Select bits have built-in protection such that they cannot be written when the PLL is on (PLLON = 1).*

8.6.7 CGM Clock Output Select Register (PCKS)

The PCKS determines which clock signal should be used as CGMOUT.

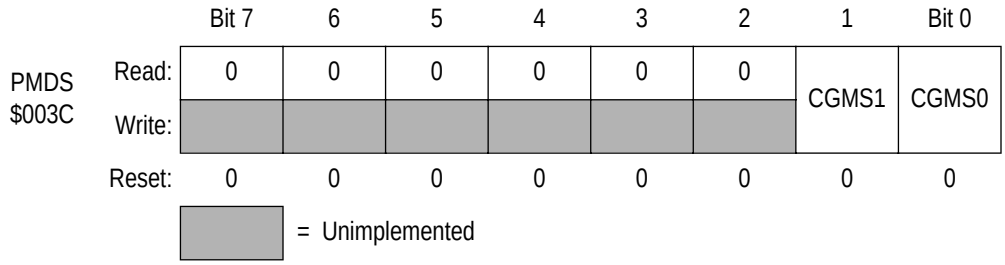


Figure 8-11. CGM Clock Output Select Register (PCKS)

CGMS1, CGMS0 — CGMOUT Programmable Divider Selection

CGMS1	CGMS0	Divider	CGMOUT Frequency @ 4.98MHz CGMOUTX
0	0	1	4.98MHz
0	1	2	2.49MHz
1	0	4	1.245MHz
1	1	8	622.5KHz

Table 8-4. Programmable Divider Selection

8.7 Interrupts

When the AUTO bit is set in the PLL bandwidth control register (PBWC), the PLL can generate a CPU interrupt request every time the LOCK bit changes state. The PLLIE bit in the PLL control register (PCTL) enables CPU interrupts from the PLL. PLLF, the interrupt flag in the PCTL, becomes set whether interrupts are enabled or not. When the AUTO bit is clear, CPU interrupts from the PLL are disabled and PLLF reads as logic zero.

Software should read the LOCK bit after a PLL interrupt request to see if the request was due to an entry into lock or an exit from lock. When the PLL enters lock, the VCO clock, CGMVCLK, divided by two can be selected as the CGMOUT source by setting BCS in the PCTL. When the PLL exits lock, the VCO clock frequency is corrupt, and appropriate precautions should be taken. If the application is not frequency-sensitive, interrupts should be disabled to prevent PLL interrupt service routines from impeding software performance or from exceeding stack limitations.

NOTE: *Software can select the CGMDCLK divided by two as the CGMOUT source even if the PLL is not locked (LOCK = 0). Therefore, software should make sure the PLL is locked before setting the BCS bit.*

8.8 Special Modes

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

8.8.1 Wait Mode

The WAIT instruction does not affect the CGMB. Before entering wait mode, software can disengage and turn off the PLL by clearing the BCS and PLLON bits in the PLL control register (PCTL) to save power. Less power-sensitive applications can disengage the PLL without turning it off, so that the PLL clock is immediately available at WAIT exit. This would also be the case when the PLL is to wake the MCU from wait mode, such as when the PLL is first enabled and waiting for LOCK, or LOCK is lost.

8.8.2 CGMB 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) enables software to clear status bits during the break state. (See **7.8.3 SIM Break Flag Control Register (SBFCR)**.)

To allow software to clear status bits during a break interrupt, write a logic one 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 the PLLF bit during the break state, write a logic zero to the BCFE bit. With BCFE at logic zero (its default state), software can read and write the PLL control register during the break state without affecting the PLLF bit.

8.9 Acquisition/Lock Time Specifications

The acquisition and lock times of the PLL are, in many applications, the most critical PLL design parameters. Proper design and use of the PLL ensures the highest stability and lowest acquisition/lock times.

8.9.1 Acquisition/Lock Time Definitions

Typical control systems refer to the acquisition time or lock time as the reaction time, within specified tolerances, of the system to a step input. In a PLL, the step input occurs when the PLL is turned on or when it suffers a noise hit. The tolerance is usually specified as a percent of the step input or when the output settles to the desired value plus or minus a percent of the frequency change. Therefore, the reaction time is constant in this definition, regardless of the size of the step input. For example, consider a system with a 5% acquisition time tolerance. If a command instructs the system to change from 0 Hz to 1 MHz, the acquisition time is the time taken for the frequency to reach 1 MHz ±50 kHz. Fifty kHz = 5% of the 1 MHz step input. If the system is operating at 1 MHz and suffers a –100 kHz noise hit, the acquisition time is the time taken to return from 900 kHz to 1 MHz ±5 kHz. Five kHz = 5% of the 100 kHz step input.

Other systems refer to acquisition and lock times as the time the system takes to reduce the error between the actual output and the desired output to within specified tolerances. Therefore, the acquisition or lock time varies according to the original error in the output. Minor errors may not even be registered. Typical PLL applications prefer to use this definition because the system requires the output frequency to be within a certain tolerance of the desired frequency regardless of the size of the initial error.

The discrepancy in these definitions makes it difficult to specify an acquisition or lock time for a typical PLL. Therefore, the definitions for acquisition and lock times for this module are as follows:

- Acquisition time, t_{ACQ}, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the tracking mode entry tolerance, Δ_{TRK}. Acquisition time is based on an initial frequency error, $(f_{DES} - f_{ORIG})/f_{DES}$, of not more than ±100%. In automatic bandwidth control mode (see **8.4.5 Manual and Automatic PLL Bandwidth Modes**), acquisition time expires when the $\overline{ACQ}$ bit becomes set in the PLL bandwidth control register (PBWC).

- Lock time, t_{LOCK}, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the lock mode entry tolerance, Δ_{LOCK}. Lock time is based on an initial frequency error, $(f_{DES} - f_{ORIG})/f_{DES}$, of not more than ±100%. In automatic bandwidth control mode, lock time expires when the LOCK bit becomes set in the PLL bandwidth control register (PBWC). (See **8.4.5 Manual and Automatic PLL Bandwidth Modes**.)

Obviously, the acquisition and lock times can vary according to how large the frequency error is and may be shorter or longer in many cases.

8.9.2 Parametric Influences on Reaction Time

Acquisition and lock times are designed to be as short as possible while still providing the highest possible stability. These reaction times are not constant, however. Many factors directly and indirectly affect the acquisition time.

The most critical parameter which affects the reaction times of the PLL is the reference frequency, f_{RDV}. This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, the corrections must be small compared to the desired frequency, so several corrections are required to reduce the frequency error. Therefore, the slower the reference the longer it takes to make these corrections. This parameter is under user control via the choice of crystal frequency $f_{XCLK}$and the R value programmed in the reference divider. (See **8.4.3 PLL Circuits**, **8.4.6 Programming the PLL**, and **8.6.6 PLL Reference Divider Select Register (PMDS)**.)

Another critical parameter is the external filter capacitor. The PLL modifies the voltage on the VCO by adding or subtracting charge from this capacitor. Therefore, the rate at which the voltage changes for a given frequency error (thus change in charge) is proportional to the capacitor size. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor is too large, the PLL may not be able to adjust the voltage in a reasonable time. (See **8.9.3 Choosing a Filter Capacitor**.)

Also important is the operating voltage potential applied to V_{DDA}. The power supply potential alters the characteristics of the PLL. A fixed value is best. Variable supplies, such as batteries, are acceptable if they vary within a known range at very slow speeds. Noise on the power supply is not acceptable, because it causes small frequency errors which continually change the acquisition time of the PLL.

Temperature and processing also can affect acquisition time because the electrical characteristics of the PLL change. The part operates as specified as long as these influences stay within the specified limits. External factors, however, can cause drastic changes in the operation of the PLL. These factors include noise injected into the PLL through the filter capacitor, filter capacitor leakage, stray impedances on the circuit board, and even humidity or circuit board contamination.

8.9.3 Choosing a Filter Capacitor

As described in **8.9.2 Parametric Influences on Reaction Time**, the external filter capacitor, C_F, is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage. The value of the capacitor must, therefore, be chosen with supply potential and reference frequency in mind. For proper operation, the external filter capacitor must be chosen according to the following equation:

$$C_F = C_{FACT}\left(\frac{V_{DDA}}{f_{RDV}}\right)$$

For the value of V_{DDA}, choose the voltage potential at which the MCU is operating. If the power supply is variable, choose a value near the middle of the range of possible supply values.

This equation does not always yield a commonly available capacitor size, so round to the nearest available size. If the value is between two different sizes, choose the higher value for better stability. Choosing the lower size may seem attractive for acquisition time improvement, but the PLL may become unstable. Also, always choose a capacitor with a tight tolerance (±20% or better) and low dissipation.

8.9.4 Reaction Time Calculation

The actual acquisition and lock times can be calculated using the equations below. These equations yield nominal values under the following conditions:

- Correct selection of filter capacitor, C_F (see **8.9.3 Choosing a Filter Capacitor**)
- Room temperature operation
- Negligible external leakage on CGMXFC
- Negligible noise

The K factor in the equations is derived from internal PLL parameters. K_{ACQ} is the K factor when the PLL is configured in acquisition mode, and K_{TRK} is the K factor when the PLL is configured in tracking mode (see **8.4.4 Acquisition and Tracking Modes**). Reaction time is based on an initial frequency error, $(f_{DES} - f_{ORIG})/f_{DES}$, of not more than ±100%.

$$t_{ACQ} = \left(\frac{V_{DDA}}{f_{RDV}}\right)\left(\frac{8}{K_{ACQ}}\right)$$

$$t_{AL} = \left(\frac{V_{DDA}}{f_{RDV}}\right)\left(\frac{4}{K_{TRK}}\right)$$

$$t_{LOCKMAX} = t_{ACQ} + t_{AL} + 256t_{VRDV}$$

Note the inverse proportionality between the lock time and the reference frequency.

In automatic bandwidth control mode the acquisition and lock times are quantized into units based on the reference frequency (see **8.4.5 Manual and Automatic PLL Bandwidth Modes**). A certain number of clock cycles, n_{ACQ}, is required to ascertain that the PLL is within the tracking mode entry tolerance, Δ_{TRK}, before exiting acquisition mode. A certain number of clock cycles, n_{TRK}, is required to ascertain that the PLL is within the lock mode entry tolerance, Δ_{LOCK}. Therefore, the acquisition time, t_{ACQ}, is an integer multiple of n_{ACQ}/f_{RDV}, and the acquisition to lock time, t_{AL}, is an integer multiple of n_{TRK}/f_{RDV}.

In manual mode, it is usually necessary to wait considerably longer than $t_{LOCKMAX}$ before selecting the PLL clock (see **8.4.8 Base Clock Selector**

Circuit), because the factors described in (see **8.9.2 Parametric Influences on Reaction Time**) may slow the lock time considerably. Automatic Bandwidth mode is recommended for most users.

8.10 Numerical Electrical Specifications

Table 8-5 contains numerical specification limits on environmental inputs and module outputs.

Table 8-5. Electrical Specifications

Description	Symbol	Min	Typ	Max	Notes
Operating Voltage	VDD	2.7V	—	3.6V	
Operating Temperature	T	0°C	25°C	70°C	
Crystal Reference Frequency	f_{RCLK}		32.768KHz		
Range Nom. Multiplier (Hz)	f_{NOM}		32.768KHz		2.7 to 3.6V V_{DD}
VCO Center-of-Range Frequency (Hz)	f_{VRS}	32.768K		20.0M	2.7 to 3.6V V_{DD}
VCO Nominal Frequency (Hz)		32.768K		10.0M	2.7 to 3.6V V_{DD}
VCO Range Linear Range Multiplier	L	1	64	255	
VCO Power-of-Two Range Multiplier	2^E	1	1	8	
VCO Multiply Factor	N	1	64	4095	
VCO Prescale Multiplier	2^P	1	1	8	
Reference Divider Factor	R	1	1	15	
VCO Operating Frequency	f_{VCLK}	f_{VRSMIN}		f_{VRSMAX}	
Bus Operating Frequency (Hz)	f_{BUS}		2.49M		2.7 to 3.6V V_{DD} only

8.11 Acquisition/Lock Time Specifications

Table 8-6 provides specifications for the entry and exit of Acquisition and Tracking Modes, as well as required Manual Mode delay times.

Table 8-6. Acquisition/Lock Time Specifications

Description	Symbol	Min	Typ	Max	Notes
Filter Capacitor Multiply Factor	C_{FACT}		0.0145		F/sV
Acquisition Mode Time Factor	K_{ACQ}		0.117		V
Tracking Mode Time Factor	K_{TRK}		0.021		V
Manual Mode Time to Stable	t_{ACQ}	—	$(8 \times V_{DDA}) \div (f_{RDV} \times K_{ACQ})$	—	If C_F chosen correctly.
Manual Stable to Lock Time	t_{AL}	—	$(4 \times V_{DDA}) \div (f_{RDV} \times K_{TRK})$	—	If C_F chosen correctly.
Manual Acquisition Time	t_{LOCK}		$t_{ACQ} + t_{AL}$		
Tracking Mode Entry Frequency Tolerance	Δ_{TRK}	0	—	± 3.6%	
Acquisition Mode Entry Frequency Tolerance	Δ_{ACQ}	± 6.3%	—	± 7.2%	
LOCK Entry Freq. Tolerance	Δ_{LOCK}	0	—	± 0.9%	
LOCK Exit Freq. Tolerance	Δ_{UNL}	± 0.9%		± 1.8%	
Reference cycles per Acquisition Mode Measurement	n_{ACQ}		32		
Reference cycles per Tracking Mode Measurement	n_{TRK}		128		
Automatic Mode Time to Stable	t_{ACQ}	$n_{ACQ} \div f_{RDV}$	$(8 \times V_{DDA}) \div (f_{RDV} \times K_{ACQ})$		If C_F chosen correctly.
Automatic Stable to Lock Time	t_{AL}	$n_{TRK} \div f_{RDV}$	$(4 \times V_{DDA}) \div (f_{RDV} \times K_{TRK})$	—	If C_F chosen correctly.
Automatic Lock Time	t_{LOCK}	—	$t_{ACQ} + t_{AL}$	—	

Section 9. Monitor ROM (MON)

9.1 Contents

9.2 Introduction

This section describes the monitor ROM (MON08). The monitor ROM allows complete testing of the MCU through a single-wire interface with a host computer.

9.3 Features

Features of the monitor ROM include the following:

- 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
- 4800 Baud to 28.8 kBaud Communication with Host Computer
- Execution of Code in RAM or ROM
- FLASH Programming

9.4 Functional Description

The monitor ROM receives and executes commands from a host computer. **Figure 9-1** shows a sample circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

Simple monitor commands can access any memory address. In monitor mode, the MCU can execute 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/KBA0 pin. A level-shifting and multiplexing interface is required between PTA0/KBA0 and the host computer. PTA0/KBA0 is used in a wired-OR configuration and requires a pullup resistor.

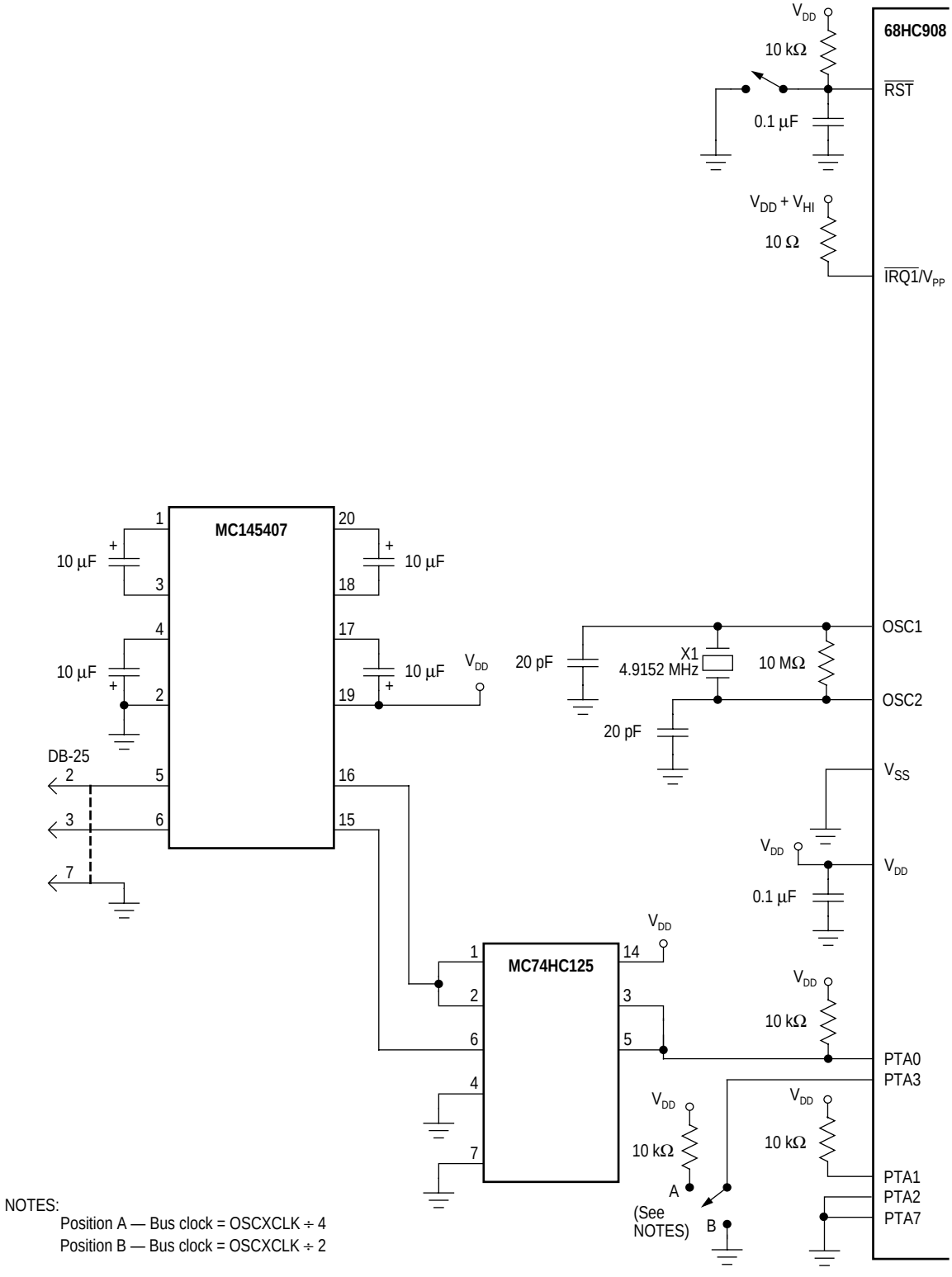


Figure 9-1. Monitor Mode Circuit

9.4.1 Entering Monitor Mode

Table 9-1 shows the pin conditions for entering monitor mode.

Table 9-1. Mode Selection

$\overline{IRQ1}/V_{PP}$ Pin	PTA7 Pin	PTA2 Pin	PTA1 Pin	PTA0 Pin	PTA3 Pin	Mode	CGMOUT	Bus Frequency
$V_{DD} + V_{HI}$	0	0	1	1	1	Monitor	$\frac{CGMXCLK}{2}$ or $\frac{CGMDCLK}{2}$	$\frac{CGMOUT}{2}$
$V_{DD} + V_{HI}$	0	0	1	1	0	Monitor	CGMXCLK	$\frac{CGMOUT}{2}$

Enter monitor mode by either

- Executing a software interrupt instruction (SWI) or
- Applying a logic zero and then a logic one to the $\overline{RST}$ pin.

The MCU sends a break signal (10 consecutive logic zeros) to the host computer, indicating that it is ready to receive a command. The break signal also provides a timing reference to allow the host to determine the necessary baud rate.

Monitor mode uses alternate vectors for reset, SWI, and break interrupt. 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. The COP module is disabled in monitor mode as long as $V_{DD} + V_{HI}$ is applied to either the $\overline{IRQ1}/V_{PP}$ pin or the OSC1 pin. (See **Section 7. System Integration Module (SIM)** for more information on modes of operation.)

NOTE: *Holding the PTA3 pin low when entering monitor mode causes a bypass of a divide-by-two stage at the oscillator. The CGMOUT frequency is equal to the CGMXCLK frequency, and the OSC1 input directly generates internal bus clocks. In this case, the OSC1 signal must have a 50% duty cycle at maximum bus frequency.*

Table 9-2 is a summary of the differences between user mode and monitor mode.

Table 9-2. Mode Differences

Modes	Functions						
	COP	Reset Vector High	Reset Vector Low	Break Vector High	Break Vector Low	SWI Vector High	SWI Vector Low
User	Enabled	$FFFE	$FFFF	$FFFC	$FFFD	$FFFC	$FFFD
Monitor	Disabled(1)	$FEFE	$FEFF	$FEFC	$FEFD	$FEFC	$FEFD

Notes:
1. If the high voltage (V_{DD} + V_{HI}) is removed from the $\overline{IRQ1}$/V_{PP} pin or the OSC1 pin, the SIM asserts its COP enable output. The COP is a mask option enabled or disabled by the COPD bit in the mask option register.

9.4.2 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. (See **Figure 9-2** and **Figure 9-3**.)

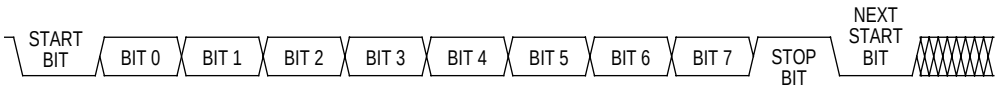


Figure 9-2. Monitor Data Format

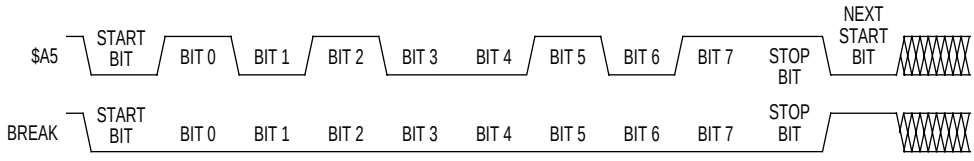


Figure 9-3. Sample Monitor Waveforms

The data transmit and receive rate can be anywhere from 4800 baud to 28.8 kbaud. Transmit and receive baud rates must be identical.

9.4.3 Echoing

As shown in **Figure 9-4**, the monitor ROM immediately echoes each received byte back to the PTA0/KBA0 pin for error checking.

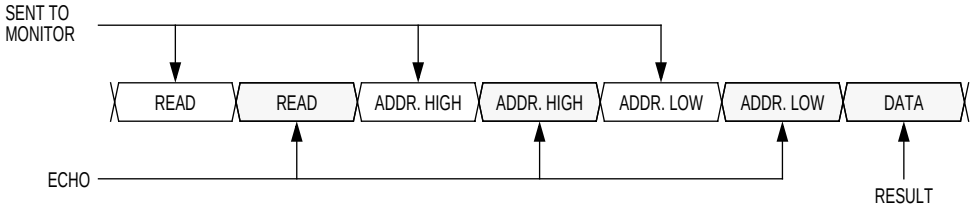


Figure 9-4. Read Transaction

Any result of a command appears after the echo of the last byte of the command.

9.4.4 Break Signal

A start bit followed by nine low bits is a break signal (see **Figure 9-5**). When the monitor receives a break signal, it drives the PTA0/KBA0 pin high for the duration of two bits before echoing the break signal.

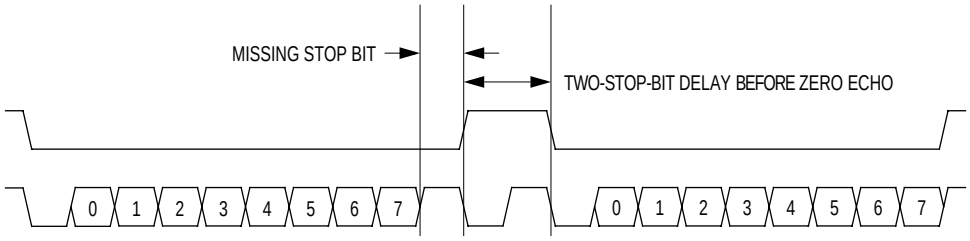


Figure 9-5. Break Transaction

Commands

The monitor ROM uses the following commands:

- READ (read memory)
- WRITE (write memory)
- IREAD (indexed read)
- IWRITE (indexed write)
- READSP (read stack pointer)
- RUN (run user program)

Table 9-3. READ (Read Memory) Command

Description	Read byte from memory
Operand	Specifies 2-byte address in high byte:low byte order
Data Returned	Returns contents of specified address
Opcode	$4A

Command Sequence

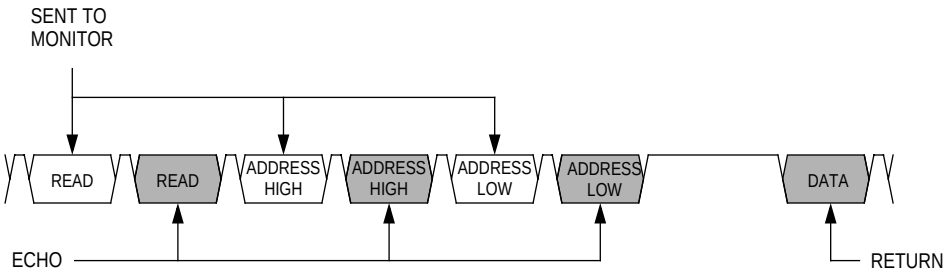


Table 9-4. WRITE (Write Memory) Command

Description	Write byte to memory
Operand	Specifics 2-byte address in high byte:low byte order; low byte followed by data byte
Data Returned	None
Opcode	$49

Command Sequence

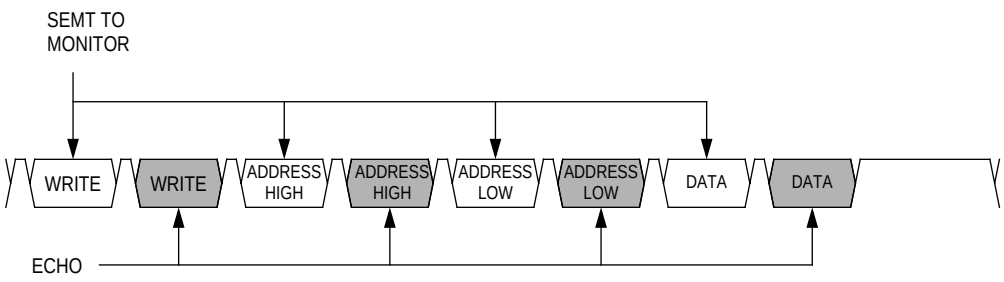


Table 9-5. IREAD (Indexed Read) Command

Description	Read Next 2 Bytes in Memory from Last Address Accessed
Operand	Specifies 2-byte address in high byte:low byte order
Data Returned	Returns contents of next two addresses
Opcode	$1A

Command Sequence

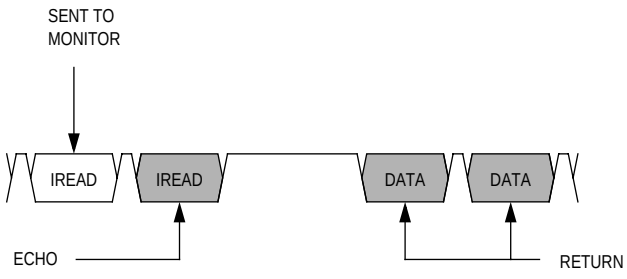
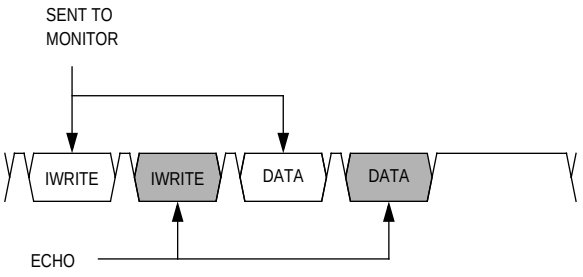


Table 9-6. IWRITE (Indexed Write) Command

Description	Write to last address accessed + 1
Operand	Specifies 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 9-7. READSP (Read Stack Pointer) Command

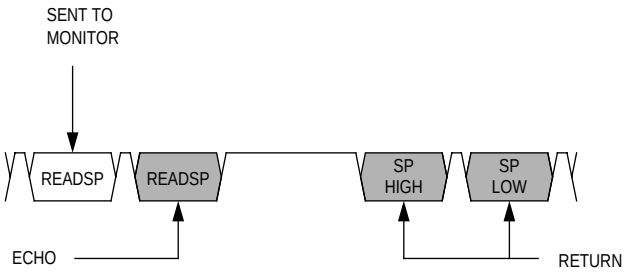
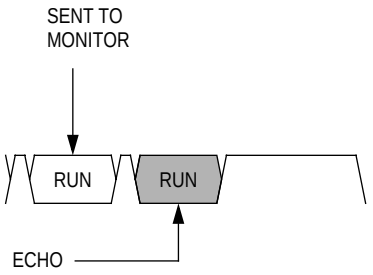
Description	Reads stack pointer
Operand	None
Data Returned	Returns stack pointer in high byte:low byte order
Opcode	$0C
Command Sequence	

Table 9-8. RUN (Run User Program) Command

Description	Executes RTI instruction
Operand	None
Data Returned	None
Opcode	$28
Command Sequence	

9.4.5 Baud Rate

With a 4.9152-MHz external clock and the PTA3 pin at logic one during reset, data is transferred between the monitor and host at 4800 baud. If the PTA3 pin is at logic zero during reset, the monitor baud rate is 9600. When the CGM output, CGMOUT, is driven by the PLL, the baud rate is determined by the MUL[7:4] bits in the PLL programming register (PPG). (See **Section 8. Clock Generation Module (CGMB)**.)

Table 9-9. Monitor Baud Rate Selection

	VCO Frequency Multiplier (N)					
	1	**2**	**3**	**4**	**5**	**6**
Monitor Baud Rate	4800	9600	14,400	19,200	24,000	28,800

Section 10. Time Base Module

10.1 Contents

10.2 Introduction

This section describes the Time Base module which generates real time interrupt signals for the implementation of a real time clock. The Time Base module consists of a counter clocked by the crystal clock, which will generate periodic interrupts at a user selectable rates. The module can be programmed to function even when the CPU is in STOP mode.

10.3 Features

- Can be disabled for power saving mode.
- All counters set to zero when disabled.
- Software programmable 1Hz, 4Hz, 16Hz, and 256Hz periodic interrupt using 32.768KHz crystal.
- Option to operate when CPU is in STOP mode

10.4 Functional Description

NOTE: *This module is designed for a 32.768 kHz oscillator.*

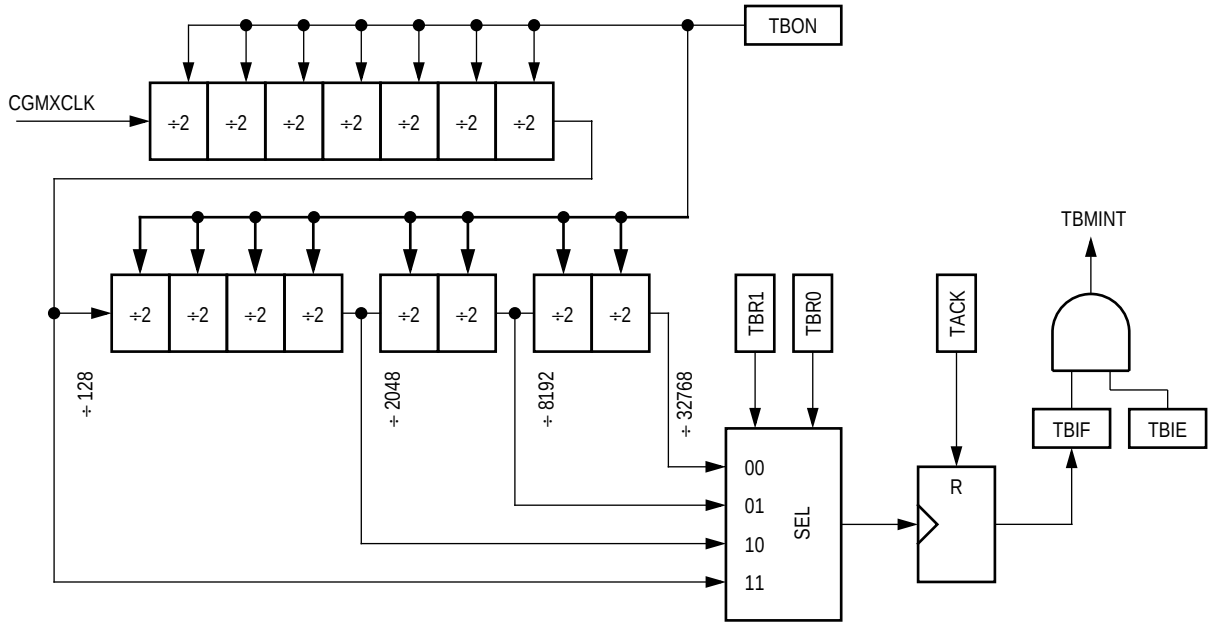


Figure 10-1. Time Base Block Diagram

This module can generate a periodic interrupt by dividing the crystal frequency, CGMXCLK. The counter is initialized to all zeros when TBON bit is cleared. The counter, shown in **Figure 10-1**, starts counting when the TBON bit is set. When the counter overflows at the tap selected by TBR1:TBR0, the TBIF bit is 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 Time Base Module, the interrupt is generated at approximately half of the overflow period. Subsequent events occur at the exact period.

10.5 Time Base Register Description

The Time Base has one register, the TBCR, which is used to enable the Time Base interrupts and set the rate.

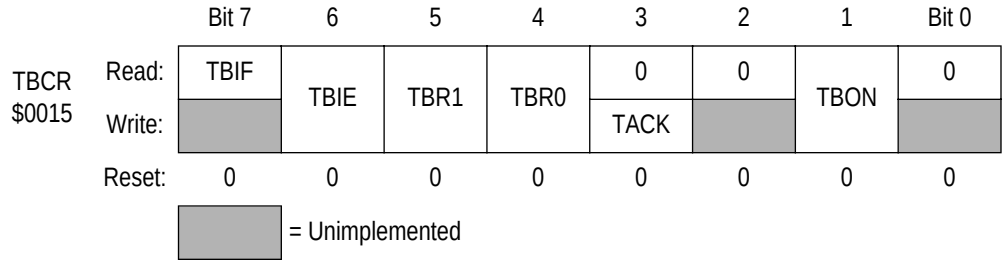


Figure 10-2. Time Base Control Register

TBIF — Time Base Interrupt Flag

This read only flag bit is set when the Time Base counter has rolled over.

1 = Time Base Interrupt pending
0 = Time Base Interrupt not pending

TBIE — Time Base Interrupt Enabled

This read/write bit enables the Time Base interrupt when the TBIF bit becomes set. Reset clears the TBIE bit

1 = Time Base Interrupt is enabled
0 = Time Base Interrupt is disabled

TBR1:TBR0 — Time Base Rate selection

These read/write bits are used to select the rate of Time Base interrupts.

Table 10-1. Time Base Rate Selection for OSC1 = 32.768 KHz

TBR1:TBR0	Divider	TIME BASE INTERRUPT RATE	
		(Hz)	(ms)
00	1/32768	1	1000
01	1/8192	4	250
10	1/2048	16	62.5
11	1/128	256	~ 3.9

NOTE: *Do not change TBR1:TBR0 bits while the Time Base is enabled (TBON=1).*

TACK — Time base ACKnowledge

This write only bit always read as 0. Writing a logic one to this bit clears TBIF. Writing a logic zero to this bit has no effect.

1 = Clear Time Base Interrupt Flag
0 = no effect

TBON — Time Base Enabled

This read/write bit enables the Time Base. Time Base may be turned off to reduce power consumption when its function is not necessary. The counter may be initialized by clearing and then setting this bit. Reset clears the TBON bit.

1 = Time Base is enabled
0 = Time Base is disabled and counter initialized to zeros

10.6 Interrupts

The Time Base module can interrupt the CPU on a regular basis with a rate defined by TBR1 and TBR0. When the Time Base counter chain rolls over, the TBIF flag is set. If the TBIE bit is set, enabling the time base interrupt, the counter chain overflow will generate a CPU interrupt request.

Interrupts must be acknowledged by writing a logic one to TACK bit.

10.7 Low-Power Modes

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

10.7.1 Wait Mode

The Time Base module remains active after execution of the WAIT instruction. In WAIT mode the Time base register is not accessible by the CPU.

If the Time Base functions are not required during wait mode, reduce the power consumption by stopping the Time Base before enabling the WAIT instruction.

10.7.2 Stop Mode

When the OSCSTOPEN bit is set to logic one, the Time Base module will continue to operate even when the CPU is in STOP mode. It will be able to generate interrupts to bring the CPU out of STOP mode.

When the OSCSTOPEN bit is set to logic zero, the Time Base becomes inactive after execution of the STOP instruction. The STOP instruction does not affect register conditions or the state of the Time Base counter. The Time Base operation continues when the MCU exits stop mode with an external interrupt, after the system clock resumes.

If the Time Base functions are not required during STOP mode, reduce the power consumption by stopping the Time Base and setting OSCSTOPEN to zero before enabling the STOP instruction.

Section 11. Timer Interface Module (TIM)

11.1 Contents

11.2 Introduction

This section describes the timer interface module (TIM). The TIM is a four-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. **Figure 11-1** is a block diagram of the TIM.

11.3 Features

Features of the TIM include the following:

- Four Input Capture/Output Compare Channels
  - Rising-Edge, Falling-Edge, or Any-Edge Input Capture Trigger
  - Set, Clear, or Toggle Output Compare Action
- Buffered and Unbuffered Pulse Width Modulation (PWM) Signal Generation
- Programmable TIM Clock Input
- Seven-Frequency Internal Bus Clock Prescaler Selection
- Free-Running or Modulo Up-Count Operation
- Toggle Any Channel Pin on Overflow
- TIM Counter Stop and Reset Bits
- External TIM CLK input

11.4 Functional Description

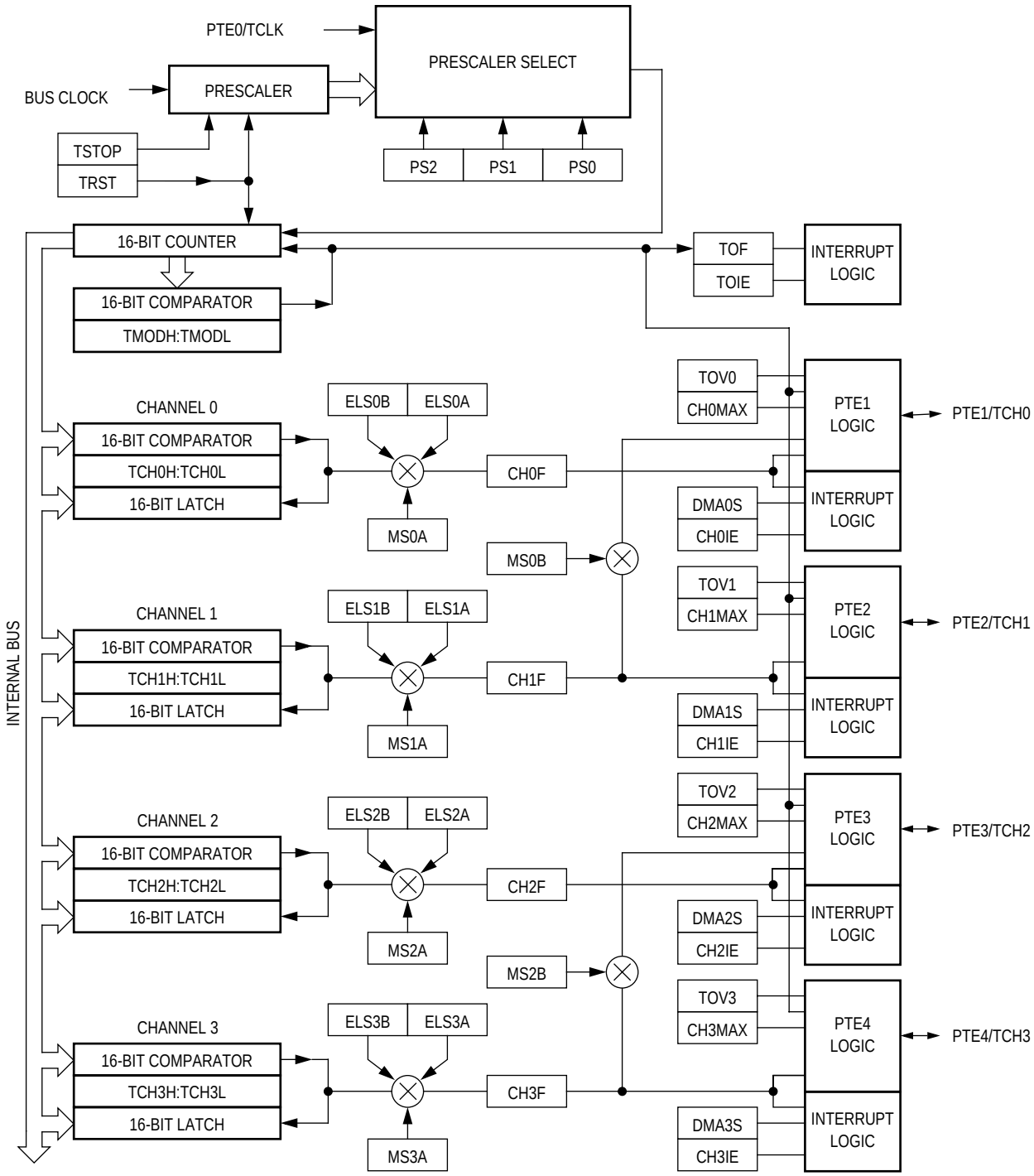


Figure 11-1. TIM Block Diagram

Table 11-1. TIM I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
TIM Status/Control Register (TSC)	TOF	TOIE	TSTOP	TRST	0	PS2	PS1	PS0	$0020
TIM Counter Register High (TCNTH)	Bit 15	14	13	12	11	10	9	Bit 8	$0022
TIM Counter Register Low (TCNTL)	Bit 7	6	5	4	3	2	1	Bit 0	$0023
TIM Counter Modulo Reg. High (TMODH)	Bit 15	14	13	12	11	10	9	Bit 8	$0024
TIM Counter Modulo Reg. Low (TMODL)	Bit 7	6	5	4	3	2	1	Bit 0	$0025
TIM Ch. 0 Status/Control Register (TSC0)	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX	$0026
TIM Ch. 0 Register High (TCH0H)	Bit 15	14	13	12	11	10	9	Bit 8	$0027
TIM Ch. 0 Register Low (TCH0L)	Bit 7	6	5	4	3	2	1	Bit 0	$0028
TIM Ch. 1 Status/Control Register (TSC1)	CH1F	CH1IE		MS1A	ELS1B	ELS1A	TOV1	CH1MAX	$0029
TIM Ch. 1 Register High (TCH1H)	Bit 15	14	13	12	11	10	9	Bit 8	$002A
TIM Ch. 1 Register Low (TCH1L)	Bit 7	6	5	4	3	2	1	Bit 0	$002B
TIM Ch. 2 Status/Control Register (TSC2)	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX	$002C
TIM Ch. 2 Register High (TCH2H)	Bit 15	14	13	12	11	10	9	Bit 8	$002D
TIM Ch. 2 Register Low (TCH2L)	Bit 7	6	5	4	3	2	1	Bit 0	$002E
TIM Ch. 3 Status/Control Register (TSC3)	CH3F	CH3IE		MS3A	ELS3B	ELS3A	TOV3	CH3MAX	$002F
TIM Ch. 3 Register High (TCH3H)	Bit 15	14	13	12	11	10	9	Bit 8	$0030
TIM Ch. 3 Register Low (TCH3L)	Bit 7	6	5	4	3	2	1	Bit 0	$0031

= Unimplemented

Figure 11-1 shows the structure of the TIM. The central component of the TIM is the 16-bit TIM counter that can operate as a free-running counter or a modulo up-counter. The TIM counter provides the timing reference for the input capture and output compare functions. The TIM counter modulo registers, TMODH:TMODL, control the modulo value of the TIM counter. Software can read the TIM counter value at any time without affecting the counting sequence.

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

11.4.1 TIM Counter Prescaler

The TIM clock source is from 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 TIM status and control register select the TIM clock source.

11.4.2 Input Capture

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

11.4.3 Output Compare

With the output compare function, the TIM can generate a periodic pulse with a programmable polarity, duration, and frequency. When the counter reaches the value in the registers of an output compare channel, the TIM can set, clear, or toggle the channel pin. Output compares can generate TIM CPU interrupt requests.

11.4.4 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in **11.4.3 Output Compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM channel registers.

An unsynchronized write to the TIM channel registers to change an output compare value could cause incorrect operation for up to two counter overflow periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that counter overflow period. Also, using a TIM overflow interrupt routine to write a new, smaller output

compare value may cause the compare to be missed. The TIM may pass the new value before it is written.

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

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable channel x TIM overflow interrupts and write the new value in the TIM overflow interrupt routine. The TIM overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

11.4.5 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTE1/TCH0 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The output compare value in the TIM channel 0 registers initially controls the output on the PTE1/TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE2/TCH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered output compare channel whose output appears on the PTE3/TCH2 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIM channel 2 status and control register (TSC2) links channel 2 and channel 3. The output compare value in the TIM channel 2 registers initially controls the output on the PTE3/TCH2 pin. Writing to the TIM channel 3 registers enables the TIM channel 3 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM channel registers (2 or 3) that control the output are the ones written to last. TSC2 controls and monitors the buffered output compare function, and TIM channel 3 status and control register (TSC3) is unused. While the MS2B bit is set, the channel 3 pin, PTE4/TCH3, is available as a general-purpose I/O pin.

NOTE: *In buffered output compare operation, do not write new output compare values to the currently active channel registers. Writing to the active channel registers is the same as generating unbuffered output compares.*

11.4.6 Pulse Width Modulation (PWM)

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

As **Figure 11-2** shows the output compare value in the TIM channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIM to clear the channel pin on output compare if the state of the PWM pulse is logic one. Program the TIM to set the pin if the state of the PWM pulse is logic zero.

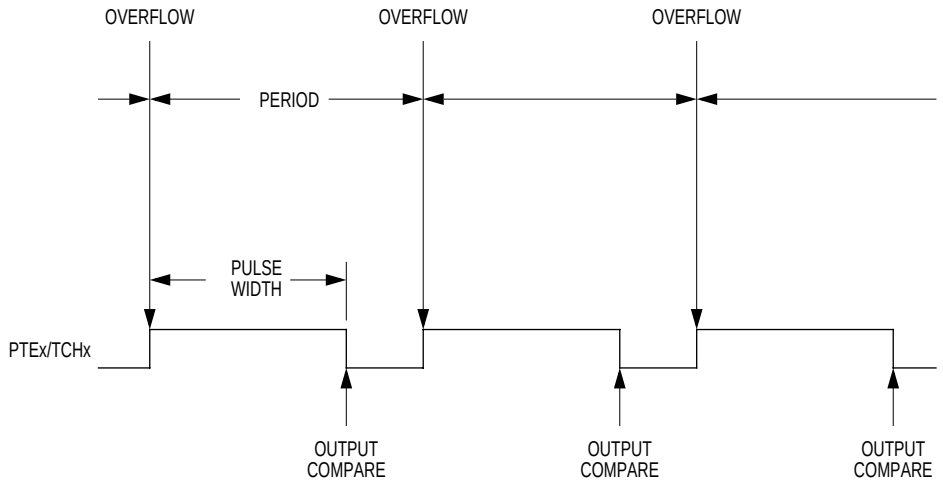


Figure 11-2. PWM Period and Pulse Width

The value in the TIM counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIM counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $0000. See (See **11.9.3 TIM Counter Modulo Registers (TMODH:TMODL)**.).

The value in the TIM channel registers determines the pulse width of the PWM output. The pulse width of an 8-bit PWM signal is variable in 256 increments. Writing $0080 (128) to the TIM channel registers produces a duty cycle of 128/256 or 50%.

11.4.7 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **11.4.6 Pulse Width Modulation (PWM)**. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

An unsynchronized write to the TIM channel registers to change a pulse width value could cause incorrect operation for up to two PWM periods.

For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that PWM period. Also, using a TIM overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIM may pass the new value before it is written.

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

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable channel x TIM overflow interrupts and write the new value in the TIM overflow interrupt routine. The TIM overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare also can cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

11.4.8 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTE1/TCH0 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The TIM channel 0 registers initially control the pulse width on the PTE1/TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the pulse width at the beginning of the next PWM

period. At each subsequent overflow, the TIM channel registers (0 or 1) that control the pulse width are the ones written to last. TSC0 controls and monitors the buffered PWM function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE2/TCH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered PWM channel whose output appears on the PTE3/TCH2 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS2B bit in TIM channel 2 status and control register (TSC2) links channel 2 and channel 3. The TIM channel 2 registers initially control the pulse width on the PTE3/TCH2 pin. Writing to the TIM channel 3 registers enables the TIM channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (2 or 3) that control the pulse width are the ones written to last. TSC2 controls and monitors the buffered PWM function, and TIM channel 3 status and control register (TSC3) is unused. While the MS2B bit is set, the channel 3 pin, PTE4/TCH3, is available as a general-purpose I/O pin.

NOTE: *In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. Writing to the active channel registers is the same as generating unbuffered PWM signals.*

11.4.9 PWM Initialization

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

1. In the TIM status and control register (TSC):
   a. Stop the TIM counter by setting the TIM stop bit, TSTOP.
   b. Reset the TIM counter by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH:TCHxL), write the value for the required pulse width.

4. In TIM channel x status and control register (TSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. See **Table 11-3**.
   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 11-3**.)

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

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

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIM channel 0 registers (TCH0H:TCH0L) initially control the buffered PWM output. TIM status control register 0 (TSCR0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Setting MS2B links channels 2 and 3 and configures them for buffered PWM operation. The TIM channel 2 registers (TCH2H:TCH2L) initially control the PWM output. TIM status control register 2 (TSCR2) controls and monitors the PWM signal from the linked channels. MS2B takes priority over MS2A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIM overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100% duty cycle output. (See **11.9.4 TIM Channel Status and Control Registers (TSC0–TSC3)**.)

11.5 Interrupts

The following TIM sources can generate interrupt requests:

- TIM overflow flag (TOF) — The TOF bit is set when the TIM counter value rolls over to $0000 after matching the value in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow CPU interrupt requests. TOF and TOIE are in the TIM status and control register.
- TIM channel flags (CH3F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM CPU interrupt requests are controlled by the channel x interrupt enable bit CHxIE. Channel x TIM CPU interrupt requests are enabled when CHxIE = 1. CHxF and CHxIE are in the TIM channel x status and control register.

11.6 Low-Power Modes

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

11.6.1 Wait Mode

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

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

11.6.2 Stop Mode

The TIM is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions or the state of the TIM counter. TIM operation resumes when the MCU exits stop mode after an external interrupt.

11.7 TIM During Break Interrupts

A break interrupt stops the TIM counter.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **7.8.3 SIM Break Flag Control Register (SBFCR)**.)

To allow software to clear status bits during a break interrupt, write a logic one 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 zero to the BCFE bit. With BCFE at logic zero (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 zero. After the break, doing the second step clears the status bit.

11.8 I/O Signals

Port E shares five of its pins with the TIM. PTE0/TCLK is an external clock input to the TIM prescaler. The four TIM channel I/O pins are PTE1/TCH0, PTE2/TCH1, PTE3/TCH2, and PTE4/TCH3.

11.8.1 TIM Clock Pin (PTE0/TCLK)

PTE0/TCLK is an external clock input that can be the clock source for the TIM counter instead of the prescaled internal bus clock. Select the PTE0/TCLK input by writing logic ones to the three prescaler select bits, PS[2:0]. See **11.9.1 TIM Status and Control Register (TSC)**. The minimum TCLK pulse width, $TCLK_{LMIN}$ or $TCLK_{HMIN}$, is:

$$\frac{1}{\text{bus frequency}} + t_{SU}$$

The maximum TCLK frequency is:

bus frequency ÷ 2

PTE0/TCLK is available as a general-purpose I/O pin when not used as the TIM clock input. When the PTE0/TCLK pin is the TIM clock input, it is an input regardless of the state of the DDRE3 bit in data direction register E.

11.8.2 TIM Channel I/O Pins (PTE1/TCH0–PTE4/TCH3)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTE1/TCH0 and PTE3/TCH2 can be configured as buffered output compare or buffered PWM pins.

11.9 I/O Registers

The following I/O registers control and monitor operation of the TIM:

- TIM status and control register (TSC)
- TIM control registers (TCNTH:TCNTL)
- TIM counter modulo registers (TMODH:TMODL)
- TIM channel status and control registers (TSC0, TSC1, TSC2, and TSC3)
- TIM channel registers (TCH0H:TCH0L, TCH1H:TCH1L, TCH2H:TCH2L, and TCH3H:TCH3L)

11.9.1 TIM Status and Control Register (TSC)

The TIM status and control register does the following:

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

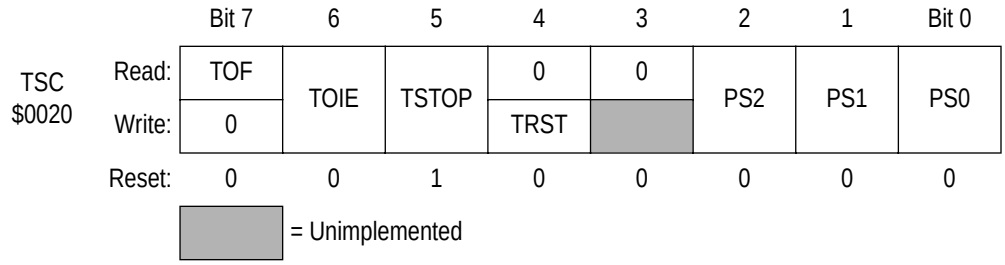


Figure 11-3. TIM Status and Control Register (TSC)

TOF — TIM Overflow Flag Bit

This read/write flag is set when the TIM counter resets to $0000 after reaching the modulo value programmed in the TIM counter modulo registers. Clear TOF by reading the TIM status and control register when TOF is set and then writing a logic zero to TOF. If another TIM overflow occurs before the clearing sequence is complete, then writing logic zero 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 one to TOF has no effect.

1 = TIM counter has reached modulo value
0 = TIM counter has not reached modulo value

TOIE — TIM Overflow Interrupt Enable Bit

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

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

TSTOP — TIM Stop Bit

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

1 = TIM counter stopped
0 = TIM counter active

NOTE: *Do not set the TSTOP bit before entering wait mode if the TIM is required to exit wait mode.*

TRST — TIM Reset Bit

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

1 = Prescaler and TIM counter cleared
0 = No effect

NOTE: *Setting the TSTOP and TRST bits simultaneously stops the TIM 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 TIM counter as **Table 11-2** shows. Reset clears the PS[2:0] bits.

Table 11-2. Prescaler Selection

PS[2:0]	TIM Clock Source
000	Internal Bus Clock ÷1
001	Internal Bus Clock ÷ 2
010	Internal Bus Clock ÷ 4
011	Internal Bus Clock ÷ 8
100	Internal Bus Clock ÷ 16
101	Internal Bus Clock ÷ 32
110	Internal Bus Clock ÷ 64
111	PTE0/TCLK

11.9.2 TIM Counter Registers (TCNTH:TCNTL)

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

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

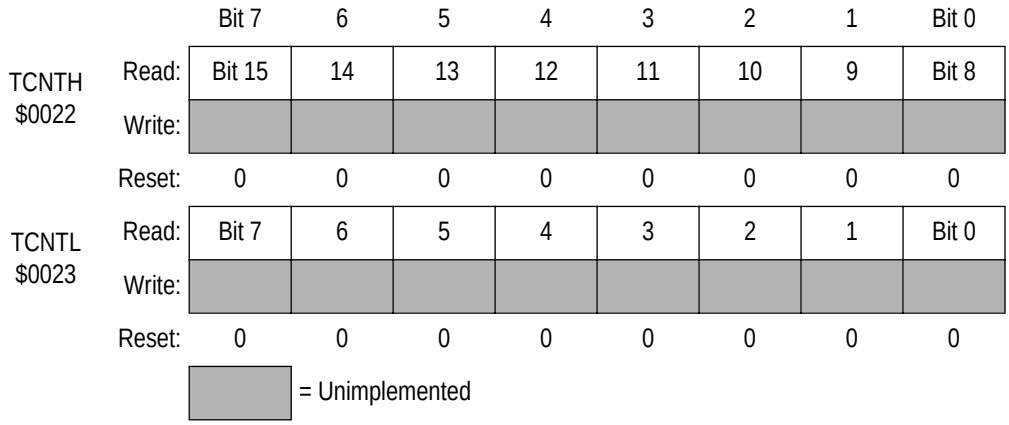


Figure 11-4. TIM Counter Registers (TCNTH:TCNTL)

11.9.3 TIM Counter Modulo Registers (TMODH:TMODL)

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

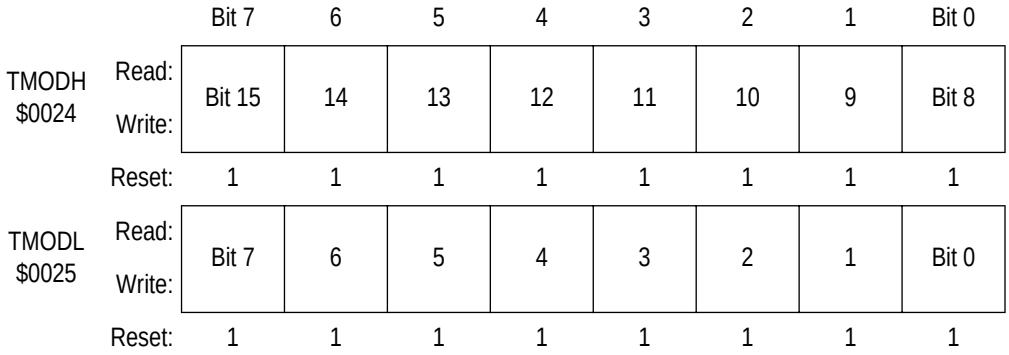


Figure 11-5. TIM Counter Modulo Registers (TMODH:TMODL)

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

11.9.4 TIM Channel Status and Control Registers (TSC0–TSC3)

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

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

		Bit 7	6	5	4	3	2	1	Bit 0
TSC0 $0026	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
TSC1 $0029	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
TSC2 $002C	Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
TSC3 $002F	Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-6. TIM Channel Status and Control Registers (TSC0–TSC3)

CHxF— Channel x Flag Bit

When channel x is an input capture channel, this read/write bit is set when an active edge occurs on the channel x pin. When channel x is an output compare channel, CHxF is set when the value in the TIM counter registers matches the value in the TIM channel x registers.

When TIM CPU interrupt requests are enabled (CHxIE= 1), clear CHxF by reading TIM channel x status and control register with CHxF set and then writing a logic zero to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic zero 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 one to CHxF has no effect.

1 = Input capture or output compare on channel x
0 = No input capture or output compare on channel x

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIM CPU interrupt requests 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 TIM channel 0 and TIM channel 2 status and control registers.

Setting MS0B disables the channel 1 status and control register and reverts TCH1 to general-purpose I/O.

Setting MS2B disables the channel 3 status and control register and reverts TCH3 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 11-3**.

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

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin (see Table 11-3). Reset clears the MSxA bit.
1 = Initial output level low
1 = Initial output level high

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

ELSxB and ELSxA — Edge/Level Select Bits

When channel x is an input capture channel, these read/write bits control the active edge-sensing logic on channel x.
When channel x is an output compare channel, ELSxB and ELSxA control the channel x output behavior when an output compare occurs.
When ELSxB and ELSxA are both clear, channel x is not connected to port E, and pin PTEx/TCHx is available as a general-purpose I/O pin. **Table 11-3** shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 11-3. Mode, Edge, and Level Selection

MSxB:MSxA	ELSxB:ELSxA	Mode	Configuration
X0	00	Output Preset	Pin under Port Control; Initial Output Level High
X1	00		Pin under Port Control; Initial Output Level Low
00	01	Input Capture	Capture on Rising Edge Only
00	10		Capture on Falling Edge Only
00	11		Capture on Rising or Falling Edge
01	01	Output Compare or PWM	Toggle Output on Compare
01	10		Clear Output on Compare
01	11		Set Output on Compare
1X	01	Buffered Output Compare or Buffered PWM	Toggle Output on Compare
1X	10		Clear Output on Compare
1X	11		Set Output on Compare

NOTE: *Before enabling a TIM channel register for input capture operation, make sure that the PTE/TCHx 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 TIM 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 TIM counter overflow.
0 = Channel x pin does not toggle on TIM counter overflow.

NOTE: *When TOVx is set, a TIM 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 zero, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As **Figure 11-7** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.

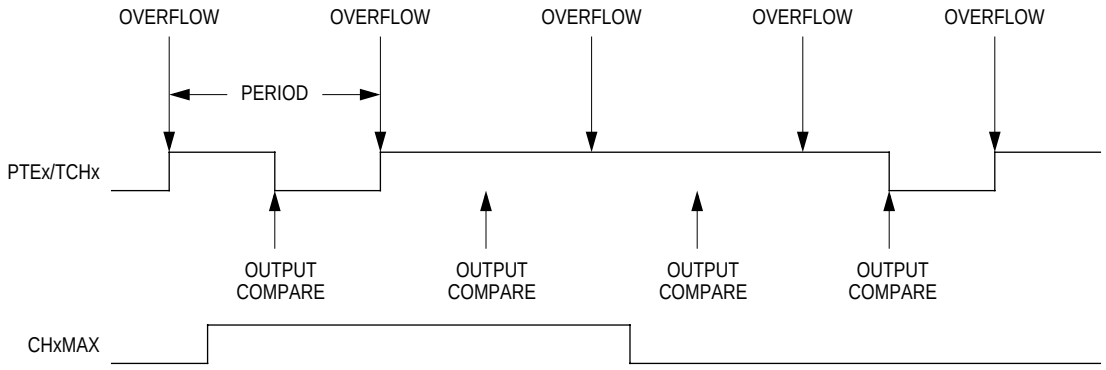


Figure 11-7. CHxMAX Latency

11.9.5 TIM Channel Registers (TCH0H/L–TCH3H/L)

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

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

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

		Bit 7	6	5	4	3	2	1	Bit 0
TCH0H $0027	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after reset							
TCH0L $0028	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
TCH1H $002A	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after reset							
TCH1L $002B	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
TCH2H $002D	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after reset							
TCH2L $002E	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
TCH3H $0030	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after reset							
TCH3L $0031	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after reset							

Figure 11-8. TIM Channel Registers (TCH0H/L–TCH3H/L)

Section 12. Reloadable Timer Module

12.1 Contents

12.2 Introduction

The Reloadable Timer Module is a 16-bit counter which will reload with a user programmable preset value, every time when an underflow occurs. This Reloadable Timer is convenient for generating periodic interrupts without the aid of software. Using the bus clock (2.5MHz), the periodic interrupt ranges from 1.19Hz to 625kHz. If CGMXCLK is used, the periodic interrupt ranges from 0.015Hz to 8.2kHz.

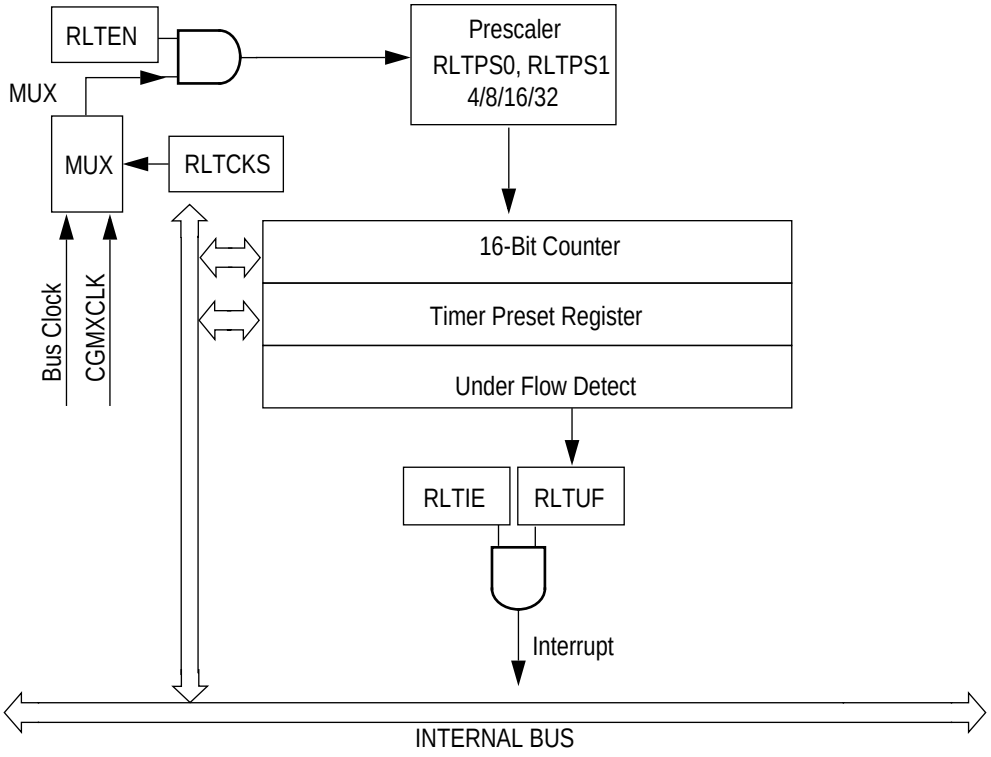


Figure 12-1. Reloadable Timer Block Diagram

Table 12-1. Reloadable Timer I/O Register Summary

Register Name		Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Timer Control Register (RLTCR)	R:	RLTCKS		RLTPS1	RLTPS0	RLTEN	RLTIE	0	RLTUF	$0010
	W:							CRTUF		
Timer Preset Register 1 (RLTPR1)	R:	TP15	TP14	TP13	TP12	TP11	TP10	TP9	TP8	$0011
	W:									
Timer Preset Register 2 (RLTPR2)	R:	TP7	TP6	TP5	TP4	TP3	TP2	TP1	TP0	$0012
	W:									
Timer Counter Register 1 (RLTCNT1)	R:	CNT15	CNT14	CNT13	CNT12	CNT11	CNT10	CNT9	CNT8	$0013
	W:									
Timer Counter Register 2 (RLTCNT2)	R:	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0	$0014
	W:									

= Unimplemented

12.3 Reloadable Timer I/O Registers

The reloadable timer has five I/O registers:

- Timer preset registers 1 and 2
- Timer counter registers 1 and 2
- Timer control register

12.3.1 Timer Preset Registers (RLTPR1, RLTPR2)

		Bit 7	6	5	4	3	2	1	Bit 0
RLTPR1 $0011	Read: Write:	TP15	TP14	TP13	TP12	TP11	TP10	TP9	TP8
	Reset:	0	0	0	0	0	0	0	0
RLTPR2 $0012	Read: Write:	TP7	TP6	TP5	TP4	TP3	TP2	TP1	TP0
	Reset:	0	0	0	0	0	0	0	0

Figure 12-2. Timer Preset Registers (RLTPR1, RLTPR1)

TP0–TP15

The two timer preset registers can be programmed independently. The value of these two registers will be loaded into the 16-bit counter and downcount the value until underflow occurs. RLTPR1 contains the most significant byte and RLTPR2 contains the least significant byte of the 16-bit count. After underflow occurs, the preset value is then reloaded for next counting cycle.

The value on the preset registers will be loaded into the 16-bit counter in the following two conditions:

- The first low to high transition of input clock after RLTEN is set to “1”
- The underflow condition occurs at the counter

12.3.2 Timer Counter Registers (RLTCNT1, RLTCNT2)

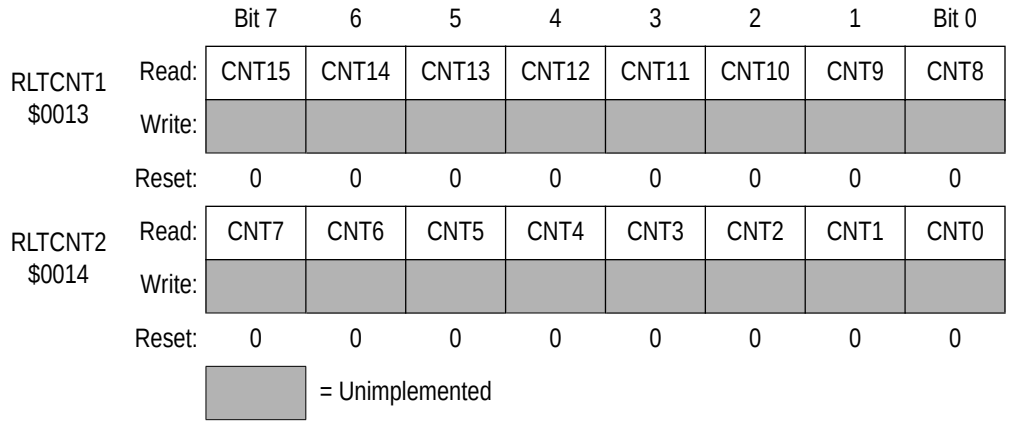


Figure 12-3. Timer Counter Registers (RLTCNT1, RLTCNT2)

The counter value on the timer can be read from the counter registers. Reset clears the counter registers.

- Timer Counter Register1 (RLTCNT1) is the High Byte Counter Register.
- Timer Counter Register2 (RLTCNT2) is the Low Byte Counter Register.

Reading the high byte (RLTCNT1) latches the content of the low byte (RLTCNT2) into a buffer. Subsequent reads of RLTCNT1 do not affect the latched RLTCNT2 value until RLTCNT2 is read.

12.3.3 Timer Control Register (RLTCR)

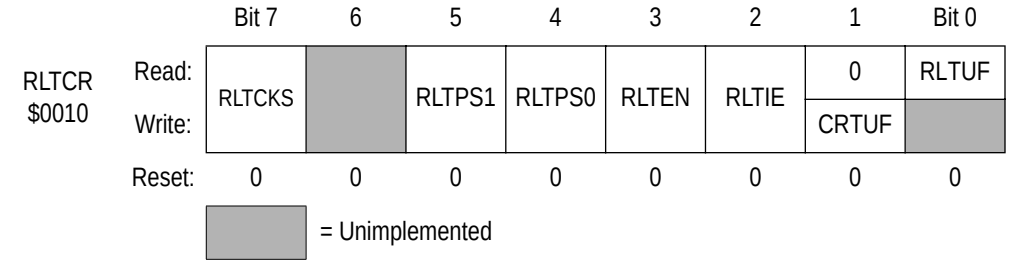


Figure 12-4. Timer Control Register (RLTCR)

RLTCKS — Timer Input Clock Select

This read/write bit is used to selected the clock source to drive the reloadable timer. Reset clears the RLTCKS bit.

1 = Internal Bus Clock
0 = CGMXCLK

RLTPS1, RLTPS0 — Prescaler Bits

These read/write bits select the prescaler divisor. Reset clears RLTPS1 and RLTPS0.

Table 12-2. Selection on the Prescaler Divisor

RLTPS1:0	Prescaler Divisor (PD)
00	4
01	8
10	16
11	32

RLTEN — Reloadable Timer Enable Bit

This read/write bit enables the clock source to drive the counters in this reloadable timer module. After this bit is set to ’1’, the first low to high transition of the clock will load the reloadable timer with the value on the Timer Preset Register. Reset clears the RLTEN bit.

1 = Enable Reloadable Timer
0 = Disable and Clear the Reloadable Timer

RLTIE — Reloadable Timer Interrupt Enable Bit

This read/write bit enables Reloadable Timer Underflow (RLTUF) CPU interrupt requests. Reset clears the RLTIE bit.

1 = Enable Reloadable Timer Underflow Interrupt
0 = Disable Reloadable Timer Underflow Interrupt

CRTUF — Clear Timer Under Flow Flag Bit

This read/write bit is used to clear the RLTUF bit in Reloadable Timer Register. Write "1" to this bit will clear the RLTUF bit, write "0" to this bit has no effect. A read from this bit will always give a "0".

1 = Clear the Reloadable Timer Underflow flag
0 = No effect

RLTUF — Reloadable Timer Underflow Flag

This read only bit is set when the counter of the reloadable timer underflow after reaching $00. Upon this bit being set, the value in the preset register will be reloaded to the counter for next cycle down counting. The CPU interrupt request will be generated if the RLTIE bit is set. This bit is cleared by writing a "1" to CRTUF bit.

1 = Reloadable Timer underflow
0 = Reloadable Timer not underflow.

12.4 Interrupts

The Reloadable Timer can generate an interrupt to the CPU upon underflow when the interrupt is enabled by setting the appropriate bit in the timer control register.

12.5 Low-Power Modes

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

12.5.1 Wait Mode

The reloadable timer module remains active after execution of the WAIT instruction. In WAIT mode, the reloadable timer registers are not accessible by the CPU.

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

12.5.2 Stop Mode

The reloadable timer is inactive after execution of the STOP instruction. The STOP instruction does not affect register conditions or the state of the time base counter. The reloadable timer operation continues when the MCU exits stop mode with an external interrupt, after the system clock resumes.

Section 13. Serial Peripheral Interface Module (SPI)

13.1 Contents

13.2 Introduction

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

13.3 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 Interrupt Services:
  - 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

13.4 Pin Name Conventions and I/O Register Addresses

The text that follows describes the SPI. The SPI I/O pin names are $\overline{SS}$ (slave select), SCK (SPI serial clock), MOSI (master out slave in), and MISO (master in slave out). The SPI shares four I/O pins with four parallel I/O ports.

The full names of the SPI I/O pins are shown in **Table 13-1**. The generic pin names appear in the text that follows.

Table 13-1. Pin Name Conventions

SPI Generic Pin Names:	MISO	MOSI	$\overline{SS}$	SCK
Full SPI Pin Names:	PTD5/MISO	PTD4/MOSI	PTD6/$\overline{SS}$	PTD3/SCK

13.5 Functional Description

Figure 13-1 summarizes the SPI I/O registers and **Figure 13-2** shows the structure of the SPI module.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0018	SPI Control Register (SPCR)	Read:	SPRIE		SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Write:								
		Reset:	0	0	1	0	0	0	0	0
$0019	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
$001A	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							

Figure 13-1. SPI I/O Register Summary

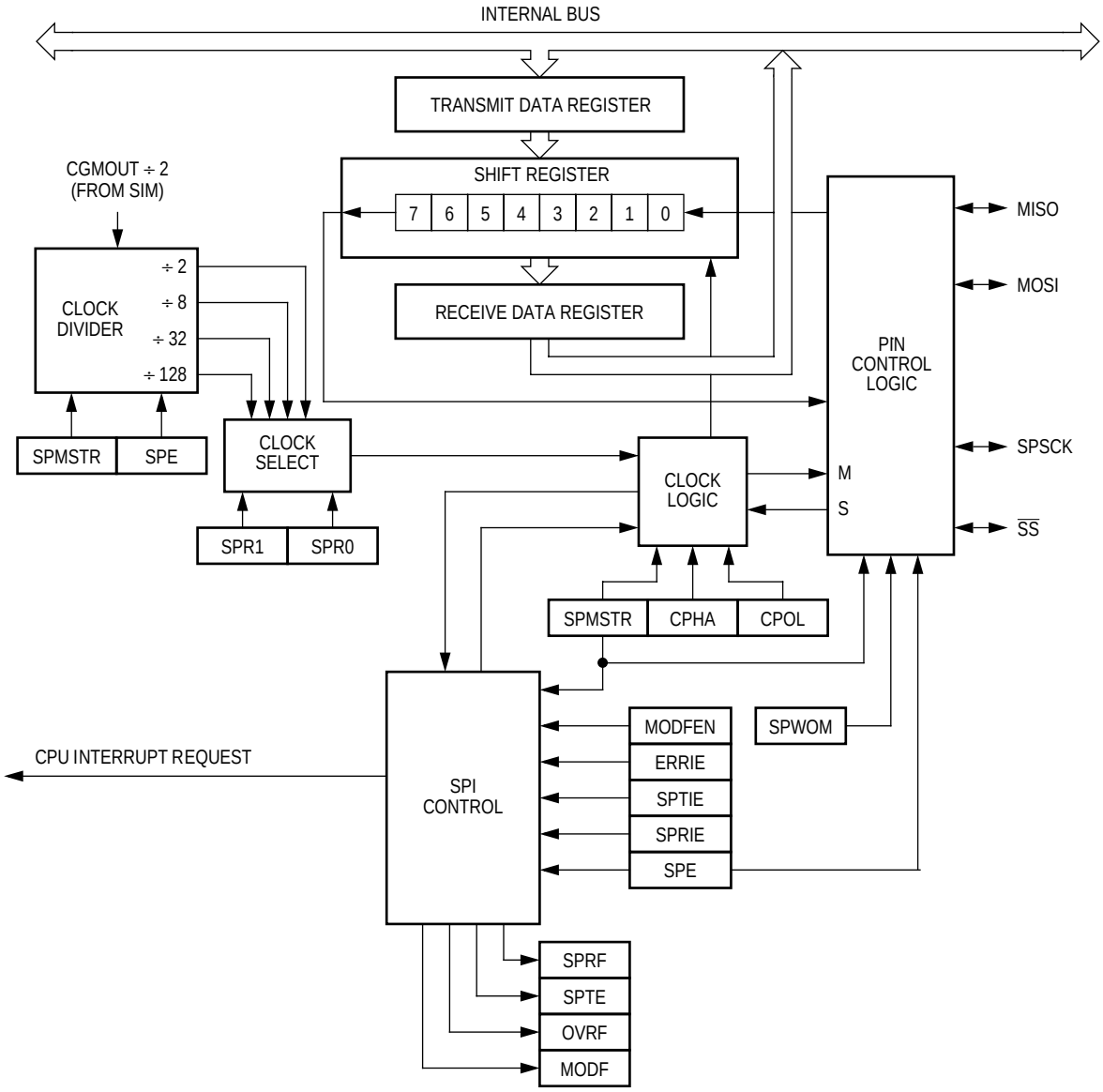


Figure 13-2. SPI Module Block Diagram

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 are serviced by the CPU.

The following paragraphs describe the operation of the SPI module.

13.5.1 Master Mode

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

NOTE: *Configure the SPI modules as master or slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. (See **13.14.1 SPI Control Register (SPCR)**.)*

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. The byte begins shifting out on the MOSI pin under the control of the serial clock (**See Figure 13-3**.)

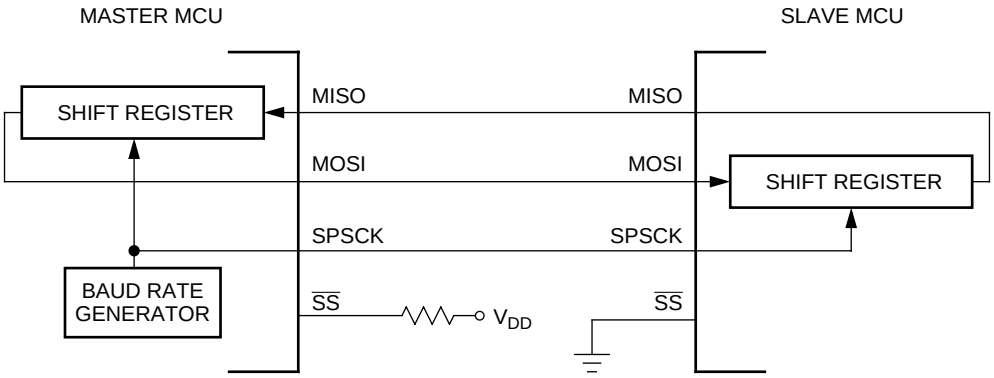


Figure 13-3. Full-Duplex Master-Slave Connections

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register (see **13.14.2 SPI Status and Control Register (SPSCR)**). Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

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, 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 with SPRF set and then reading the SPI data register. Writing to the SPI data register clears the SPTE bit.

13.5.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit is clear. In slave mode the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be at logic zero. $\overline{SS}$ must remain low until the transmission is complete (see **13.7.2 Mode Fault Error**).

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

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

A slave SPI must complete the write to the data register at least one bus cycle before the master SPI starts a 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 **13.6 Transmission Formats**).

If the write to the data register is late, the SPI transmits the data already in the shift register from the previous transmission.

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

13.6 Transmission Formats

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

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

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

13.6.2 Transmission Format When CPHA = 0

Figure 13-4 shows an SPI transmission in which CPHA is logic zero. 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 zero, 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 **13.7.2 Mode Fault Error**). When CPHA = 0, the first SCK edge is the MSB capture strobe. Therefore the slave must begin driving its data before the first SCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the slave data transmission. The slave's $\overline{SS}$ pin must be toggled back to high and then low again between each byte transmitted, as shown in **Figure 13-5**.

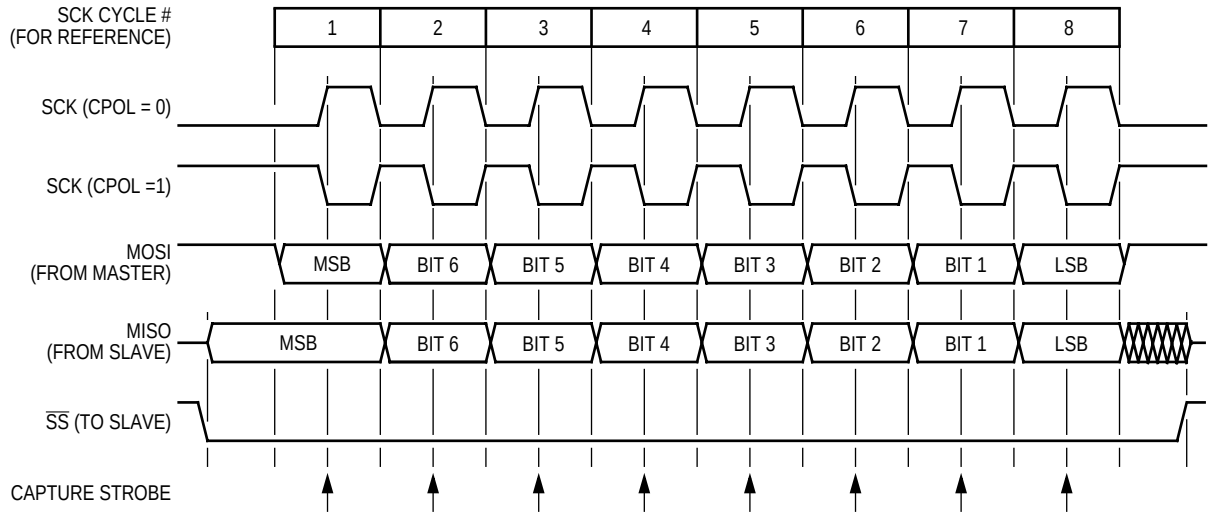


Figure 13-4. Transmission Format (CPHA = 0)

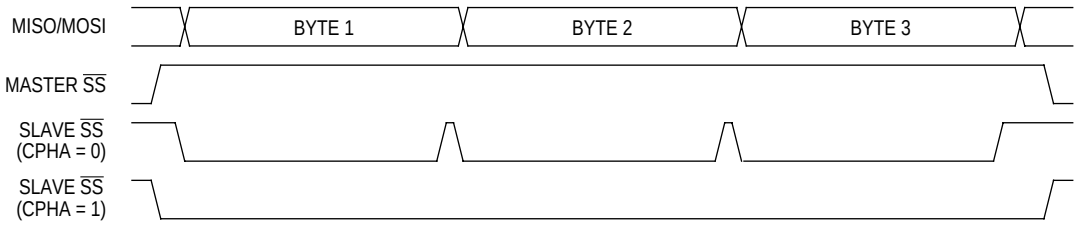


Figure 13-5. CPHA/$\overline{SS}$ Timing

13.6.3 Transmission Format When CPHA = 1

Figure 13-6 shows an SPI transmission in which CPHA is logic one. 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 zero, 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 **13.7.2 Mode Fault Error**). When CPHA = 1, the master begins driving its MOSI pin on the first SCK edge. Therefore the slave uses the first SCK 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.

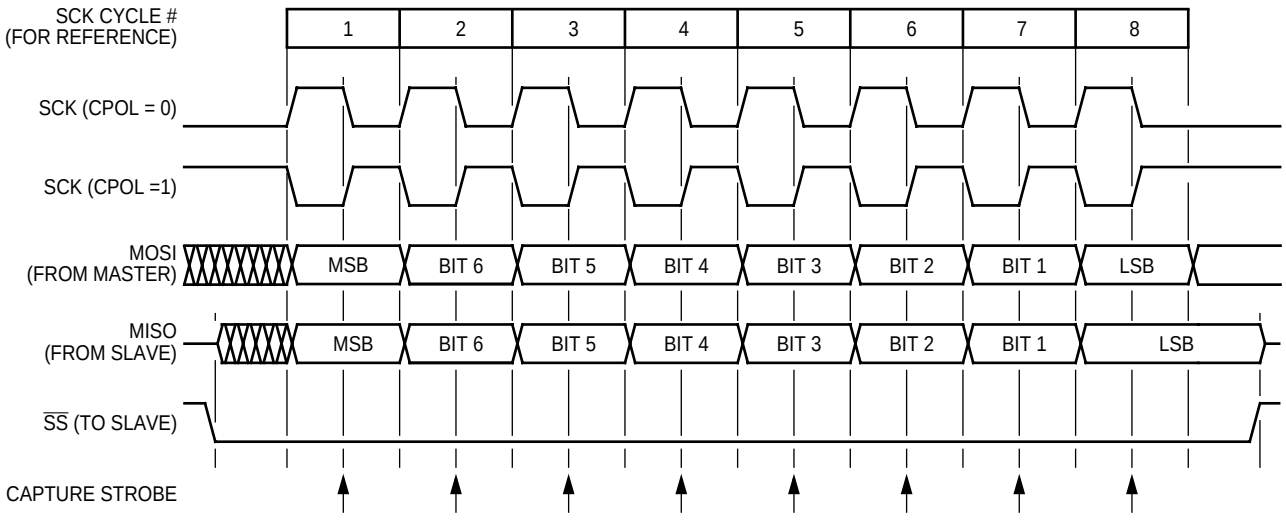


Figure 13-6. Transmission Format (CPHA = 1)

13.6.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), transmissions are started by a software write to the SPDR. 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 13-7. Transmission Start Delay (Master))**. 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 are set to conserve power. SCK edges occur halfway 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 13-7**. This delay will be no longer than a single SPI bit time. That is, the maximum delay is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

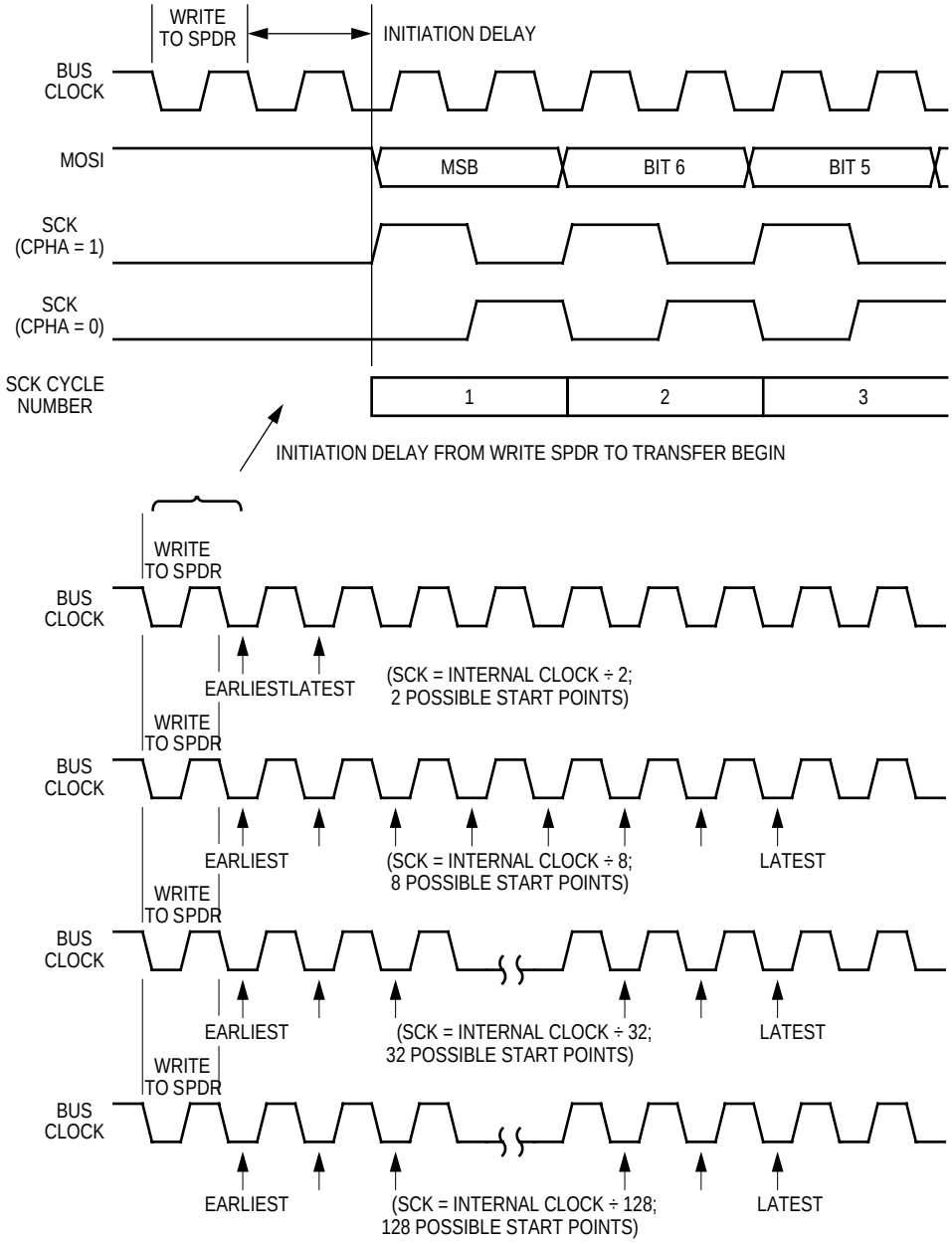


Figure 13-7. Transmission Start Delay (Master)

13.7 Error Conditions

The following flags signal SPI error conditions:

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

13.7.1 Overflow Error

The overflow flag (OVRF) 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 13-4** and **Figure 13-6**). 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) is set. It is not possible to enable MODF or OVRF individually to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. **Figure 13-8** shows how it is possible to miss an overflow.

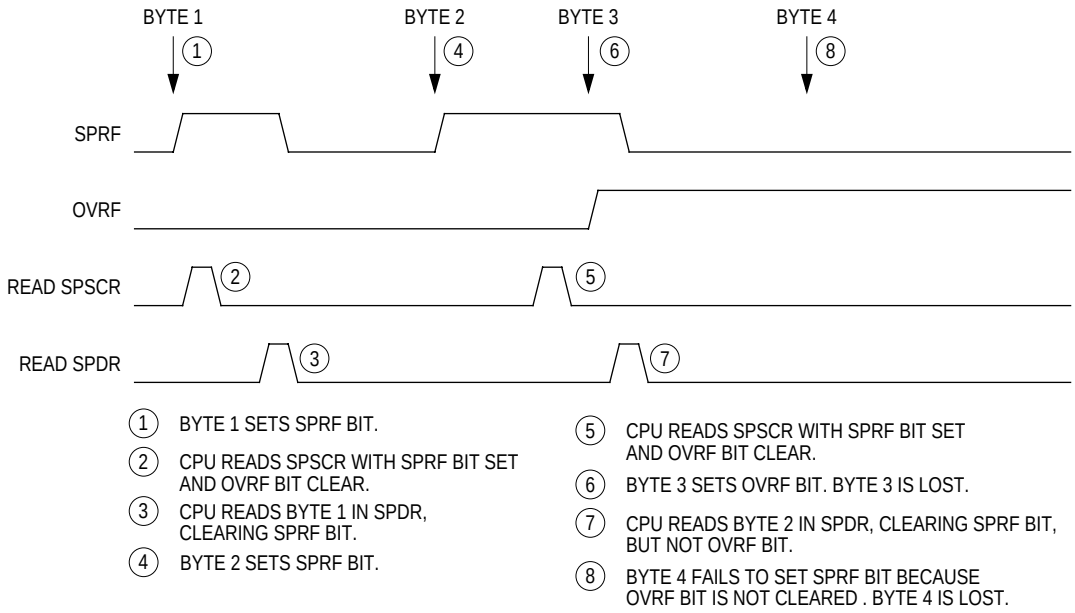


Figure 13-8. Missed Read of Overflow Condition

The first part of **Figure 13-8** 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 easily be 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 following 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 13-9** illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit.

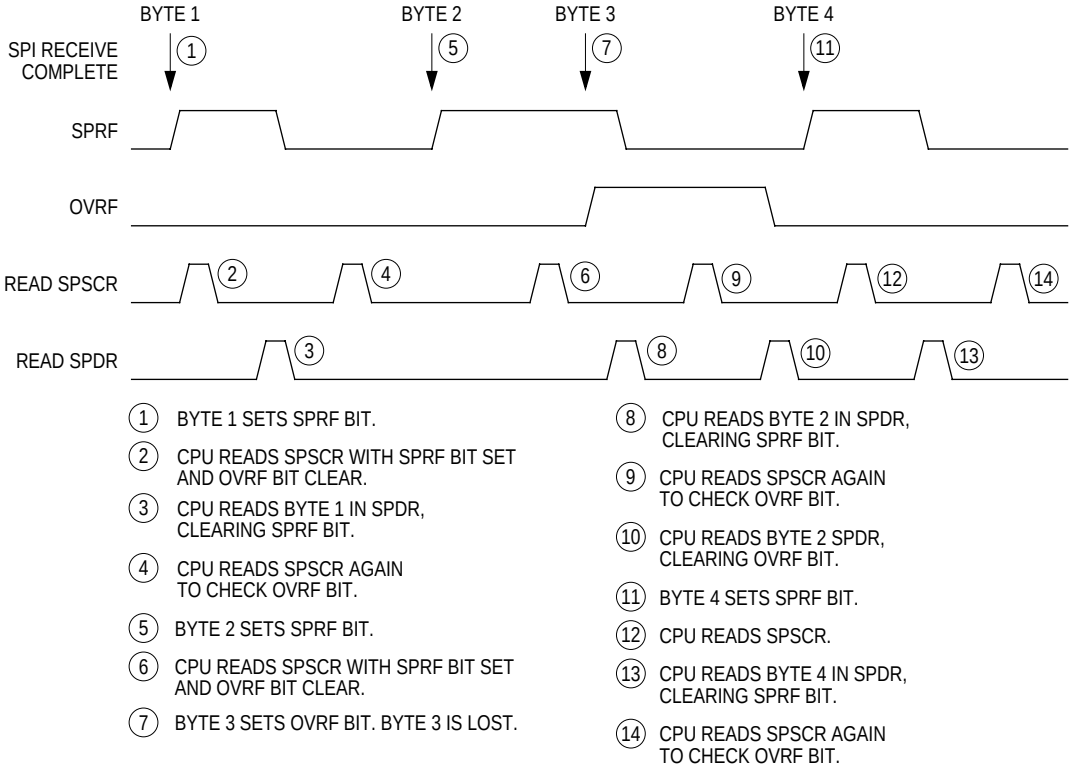


Figure 13-9. Clearing SPRF When OVRF Interrupt Is Not Enabled

13.7.2 Mode Fault Error

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

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is set. It is not possible to enable MODF or OVRF individually 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 zero. 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 SPI bits of the data direction register of the shared I/O port before enabling the SPI.*

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{SS}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{SS}$ goes low and ends once the incoming SPSCK goes back to its idle level following the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SPSCK returns to its IDLE level following the shift of the last data bit (see **13.6 Transmission Formats**).

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

NOTE: *When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is at logic 0) and later unselected ($\overline{SS}$ 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 unselected 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 clearing the SPE bit of the slave.

NOTE: *A logic one 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 it was already in the middle of a transmission.*

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

13.8 Interrupts

Four SPI status flags can be enabled to generate CPU interrupt requests:

Table 13-2. SPI Interrupts

Flag	Request
SPTE (TransmitterEmpty)	SPI Transmitter CPU Interrupt Request (SPTIE = 1) SPI Transmitter DMA Service Request (SPTIE = 1)
SPRF (Receiver Full)	SPI Receiver CPU Interrupt Request (SPRIE = 1) SPI Receiver DMA Service Request (SPRIE = 1)
OVRF (Overflow)	SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1)
MODF (ModeFault)	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 requests, 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.

The following sources in the SPI status and control register can generate CPU interrupt requests or DMA service requests:

- 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 will generate an SPI receiver/error CPU interrupt request.
- 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 will generate an SPTE CPU interrupt request.

13.9 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) 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 13-10** shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA: CPOL = 1:0).

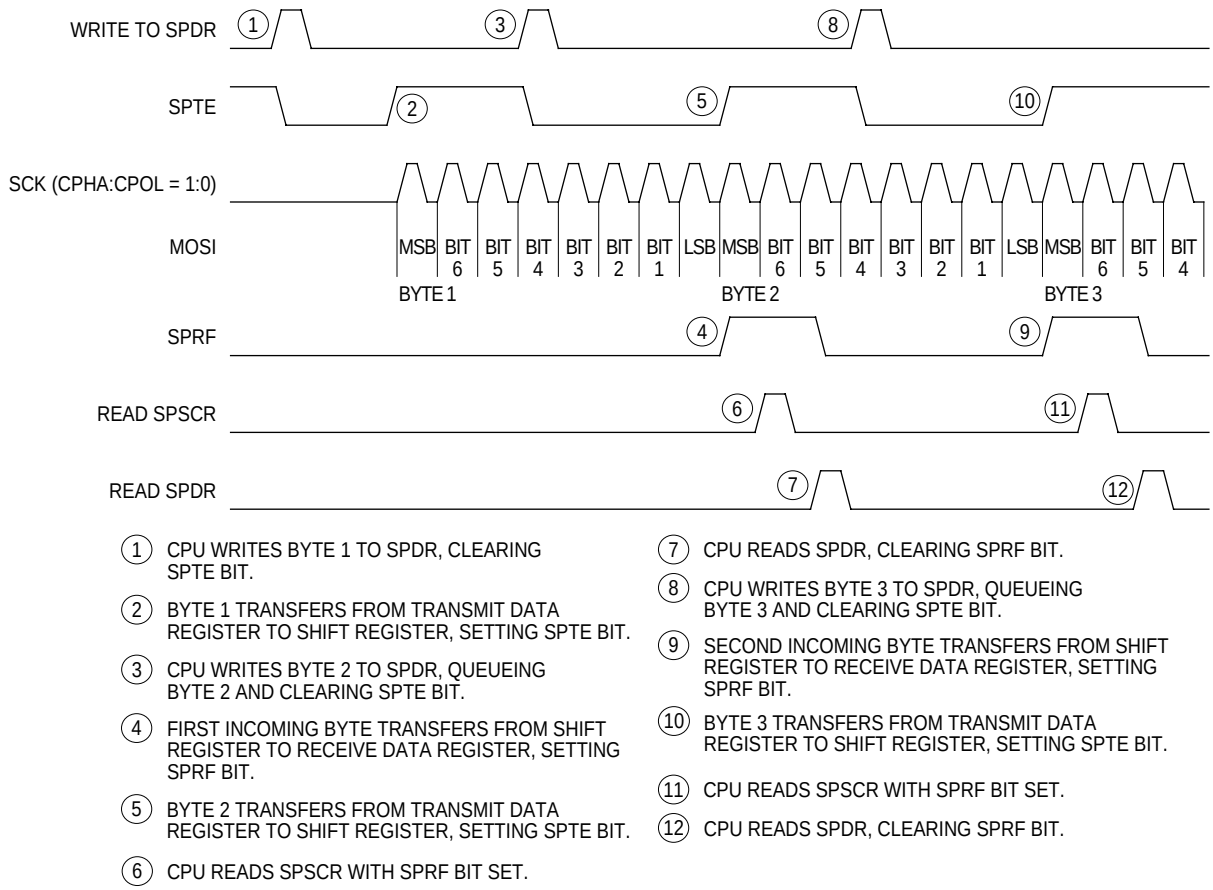


Figure 13-10. SPRF/SPTE CPU Interrupt Timing

For a slave, the transmit data buffer allows back-to-back transmissions to occur without the slave having to precisely time the write of its data between the transmissions, as would be the case for a system with a single data buffer. Also, if no new data is written to the data buffer, the last value contained in the shift register will be the next data word transmitted.

13.10 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur 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 items are reset only by a system reset:

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

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

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

13.11 Low-Power Modes

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

13.11.1 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.

13.11.2 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 an external interrupt. If stop mode is exited by reset, any transfer in progress is aborted, and the SPI is reset.

13.12 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) enables software to clear status bits during the break state. (See **Section 7. System Integration Module (SIM)**.)

To allow software to clear status bits during a break interrupt, write a logic one 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 zero to the BCFE bit. With BCFE at logic zero (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 zero. 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.

13.13 I/O Signals

The SPI module has four I/O pins and shares with a parallel I/O port.

- MISO — Data received
- MOSI — Data transmitted
- SCK — Serial clock
- $\overline{SS}$ — Slave select

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}.

13.13.1 MISO (Master In/Slave Out)

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

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is logic zero and its $\overline{SS}$ pin is at logic zero. To support a multiple-slave system, a logic one 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.

13.13.2 MOSI (Master Out/Slave In)

MOSI is one of the two SPI module pins that transmits serial data. In full duplex operation, the MOSI pin of the master SPI module is connected

to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin.

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

13.13.3 SCK (Serial Clock)

The serial clock synchronizes data transmission between master and slave devices. In a master MCU, the SCK pin is the clock output. In a slave MCU, the SCK 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 SCK pin regardless of the state of the data direction register of the shared I/O port.

13.13.4 $\overline{SS}$ (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 **13.6 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 13-11 . CPHA/SS Timing**).

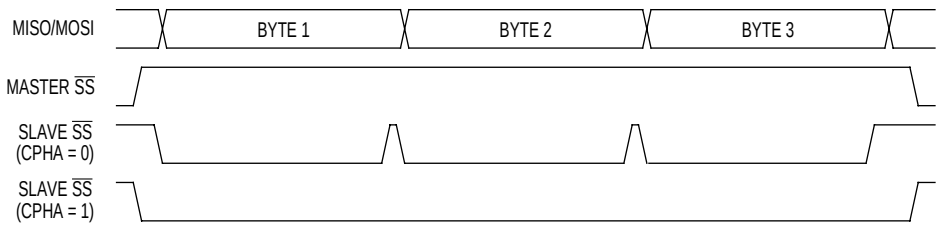


Figure 13-11. CPHA/$\overline{SS}$ Timing

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. It cannot be used as a general purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of the $\overline{SS}$ from creating a MODF error (see **13.14.2 SPI Status and Control Register (SPSCR)**).

NOTE: *A logic one voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SCK clocks, even if it was already in the middle of a transmission.*

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SCK (see **13.7.2 Mode Fault Error**). For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SCK register must be set. If the MODFEN bit is low for an SPI master, the $\overline{SS}$ pin can be used as a general purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{SS}$ pin by configuring the appropriate pin as an input and reading the data register (see **Table 13-3**).

Table 13-3. SPI Configuration

SPE	SPMSTR	MODFEN	SPI CONFIGURATION	STATE OF $\overline{SS}$ LOGIC
0	X(1)	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

Notes:
1. X = don't care

13.14 I/O Registers

Three registers control and monitor SPI operation:

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

13.14.1 SPI Control Register (SPCR)

The SPI control register does the following:

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

Enables the SPI module

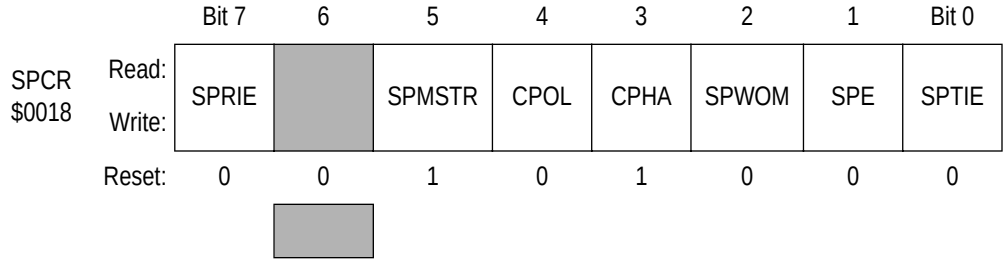


Figure 13-12. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

This read/write bit enables CPU interrupt requests generated by the SPRF bit. The SPRF bit is set when a byte transfers from the shift register to the receive data register. Reset clears the SPRIE bit.

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

SPMSTR — SPI Master Bit

This read/write bit selects master mode operation or slave mode operation. Reset sets the SPMSTR bit.

1 = Master mode
0 = Slave mode

CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the SPSCK pin between transmissions (See **Figure 13-4** and **Figure 13-6**). To transmit data between SPI modules, the SPI modules must have identical CPOL values. 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 13-4** and **Figure 13-6**). To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the $\overline{SS}$ pin of the slave SPI module must be set to logic one between bytes (see **Figure 13-11 . CPHA/SS Timing**). 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.

When CPHA = 1 for a slave, the first edge of the SPSCK indicates the beginning of the transmission. A logic one on the $\overline{SS}$ pin does not in any way affect the state of the SPI state machine.

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 after the current transmission.

SPWOM — SPI Wired-OR Mode Bit

This read/write bit disables the pull-up devices on pins SCK, MOSI, and MISO so that those pins become open-drain outputs.

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

SPE — SPI Enable

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI (see **13.10 Resetting the SPI**). Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE— SPI Transmit Interrupt Enable

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

13.14.2 SPI Status and Control Register (SPSCR)

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 the following functions:

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

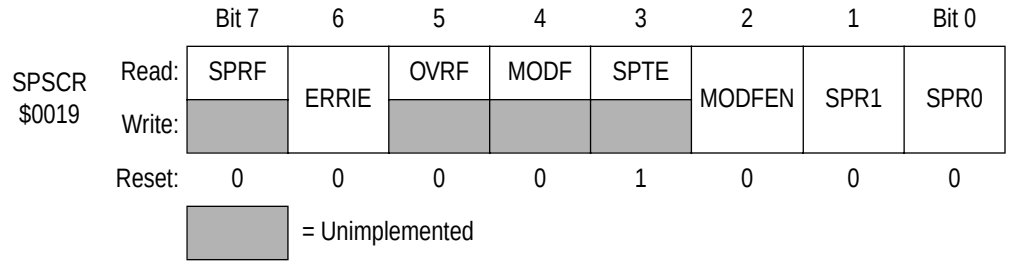


Figure 13-13. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

This clearable, read-only flag is set each time a byte transfers from the shift register to the receive data register. SPRF generates a CPU interrupt request if the SPRIE bit in the SPI control register is set also.

Reset clears the SPRF bit.

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

ERRIE — Error Interrupt Enable Bit

This read/write bit enables the MODF and OVRF 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, ready-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 SPCR 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 **13.13.4 SS (Slave Select)**)

If the MODFEN bit is low, the level of the $\overline{SS}$ pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation (see **13.7.2 Mode Fault Error**).

SPR1, SPR0 — SPI Baud Rate Select Bits

In master mode, these read/write bits select one of four baud rates as shown in **Table 13-4**. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 13-4. SPI Master Baud Rate Selection

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

Use the following formula to calculate the SPI baud rate:

$$\text{Baud rate} = \frac{\text{CGMOUT}}{2 \times \text{BD}}$$

Where:
CGMOUT = base clock output of the clock generator module (CGM)
BD = baud rate divisor

13.14.3 SPI Data Register (SPDR)

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 13-2 . SPI Module Block Diagram**.

		Bit 7	6	5	4	3	2	1	Bit 0
SPDR $001A	Read:	R7	R6	R5	R4	R3	R2	R1	R0
	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	Reset:	Indeterminate after reset							

Figure 13-14. SPI Data Register (SPDR)

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

NOTE: *Do not use read-modify-write instructions on the SPI data register since the buffer read is not the same as the buffer written.*

Section 14. Ring Detector Module (RING)

14.1 Contents

14.2 Introduction

This section describes the Ring Detector module (RING). It detects the presence of a power ring signal from the line interface circuit and verifies that the signal has correct properties on signal level, frequency and duration. If the signal detected is valid, the module will generate an interrupt to the CPU. This module shares two general purpose I/O pins on port D (see **Section 16. I/O Ports**).

14.3 Features

Features of the Ring Detector module include:

- Checks three parameters of the power ring signal; amplitude, minimum frequency and signal duration
- Generates an interrupt to the CPU upon detection of a valid ring signal
- Software control interrupt enable/disable

14.4 Functional Description

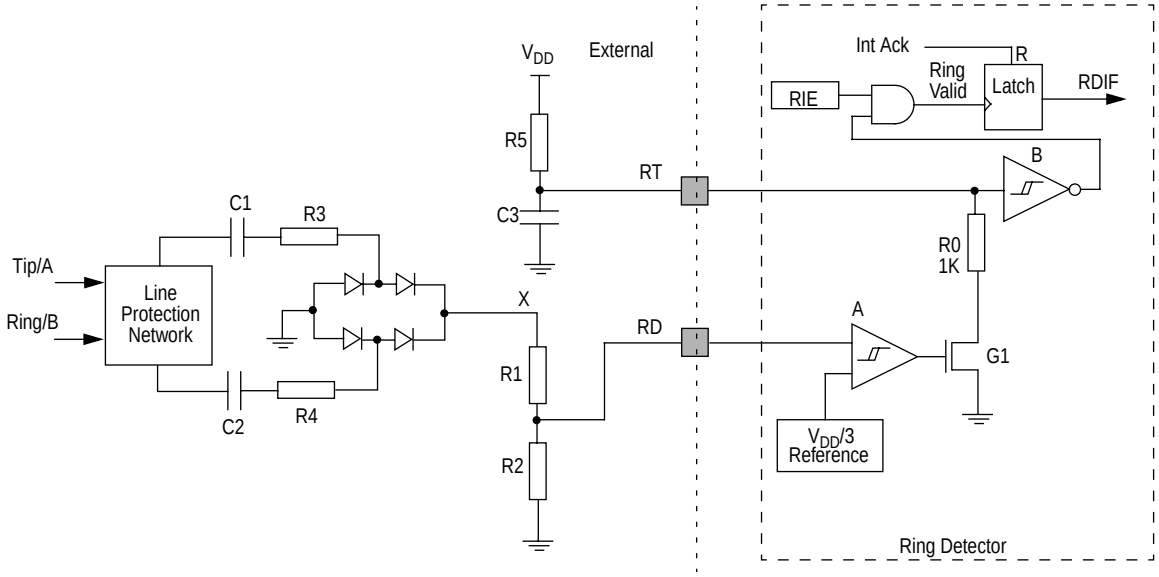


Figure 14-1. Ring Detector Block Diagram

The Ring Detector is used to check whether a power ring signal is present or not. The frequency of the ring signal must be greater than 15Hz.

When V_{DD} is applied to the circuit, capacitor C3 will be charged to V_{DD} through resistor R5.

Resistors R1 and R2 attenuate the incoming ring signal. The values of these resistors should be chosen to provide a sufficient voltage at pin

RD. The signal is then compared with a 1.2V reference voltage. If the input voltage is greater than the threshold, the comparator will turn on gate G1 to discharge capacitor C3 through resistor R0.

For each ring frequency cycle, capacitor C3 will be charged up through resistor R5 and discharged through resistor R0. The combination of these two time constants will determine the minimum period of a valid ring signal (t_1).

The time constant formed by R5 and C3 should be chosen that the voltage at pin RT is kept below the threshold (V_T) of Schmitt trigger "B" for a ring signal with frequency greater than 15Hz, which satisfies the minimum requirement.

In order to reduce the operating current, the Ring Detector can be turned off by setting the Ring Detect Enable (RDE) bit to zero. (See **14.6 Ring Detector Control Register (RDCR)**.)

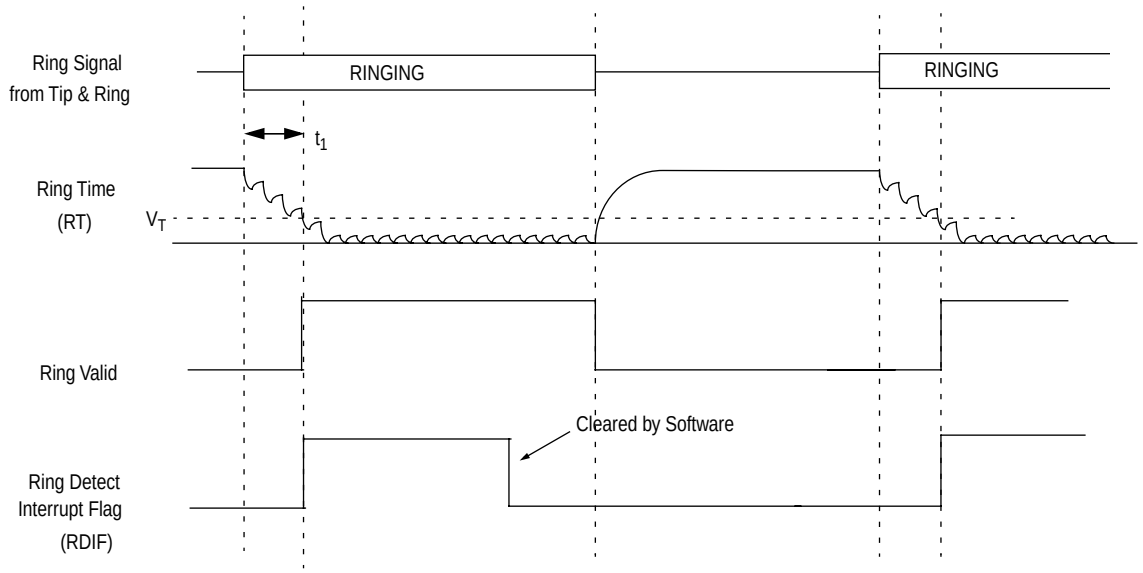


Figure 14-2. Timing Diagram

14.5 Interrupt Circuit

When the Ring Detector module is enabled (RDE = 1) and a valid ring signal is detected on the input pin RD, the module will generate a signal

to clock the output latch. The output from this latch is the interrupt signal (RDIF) to the CPU. CPU can clear the RDIF signal by writing a "0" into RDIACK of the Ring Detector Control Register (RDCR).

14.6 Ring Detector Control Register (RDCR)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0	Addr
Ring Detector Control Register (RDCR)	R:					RVALID	RDIF	0	RDE	$003E
	W:							RDIACK		
Reset:		0	0	0	0	0	0	0	0	

= Unimplemented

Figure 14-3. Ring Dector Control Register (RDCR)

RDE — Ring Detector Enable Bit

1 = Ring Detector Enabled

0 = Ring Detector Disabled (Ring Detector Module will be in low power mode)

RDIACK — Ring Detector Interrupt Acknowledge Bit

1 = Clear the RDIF latch

0 = No effect

RDIF — Ring Detector Interrupt Flag Bit

1 = Interrupt pending

0 = No interrupt pending

RVALID — Ring Valid

When this bit is high, it indicates a valid ring signal has been detected. The software can poll this bit to determine when the ring cycle stops.

1 = Valid ring signal detected

0 = No valid ring signal detected

14.7 Power Management

To reduce the operating current, the Ring Detector can be turned off by setting the Ring Detect Enable (RDE) bit to zero. (See **14.6 Ring Detector Control Register (RDCR)**.)

14.8 Low Power Modes

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

14.8.1 Wait Mode

When the ring detector module is enabled, it will continue to operate even in WAIT Mode. It will generate an interrupt to the CPU upon detection of a valid ring signal.

14.8.2 Stop Mode

When the ring detector module is enabled, it will continue to operate even in STOP Mode. It will generate an interrupt to the CPU upon detection of a valid ring signal.

Section 15. DTMF/Melody Generator Module

15.1 Contents

15.2 Introduction

The DTMF/Melody Generator (DMG) is a multi-function tone generator built into the MC68HC908W32 MCU, supporting DTMF dialling, melody-on-hold, and pacifier tone functions. The associated output pins are TNO and TPD0/TNX.

15.3 Features

- Four row and four column frequencies for DTMF dialling
- Two octaves of musical scale in steps of semitones
- Twelve modem frequencies for data rates between 300bps and 1200bps
- All frequencies can be used for pacifier tone
- Power saving mechanism for no tone condition
- 1-bit Sigma-Delta D/A converter
- 32 time steps for sine wave generation
- Sine wave or square wave selectable output for melody or DTMF
- Single or dual tone capability for melody or DTMF

15.4 General Operation

The DMG consists of a row tone and a column tone generation path. The tone frequency of each path is controlled by their respective frequency control registers: Row Frequency Control register (FCR) and Column Frequency Control register (FCC). At the TNO output, single/dual sine/square wave tones of DTMF and melody frequencies are possible, whereas at the TNX output, only single square wave tones are possible.

To generate a sine wave tone with programmable frequency in a path, the internal clock (4.980736MHz) is first divided by a frequency divider, whose value is set by the frequency control register (FCR or FCC). The output of the divider is a periodic pulse train whose frequency is the sampling rate of the desired “staircase sine wave”. This pulse train then clocks a sine wave synthesizer with 32 time-slots. the resolution of the sine wave synthesizer is 6-bit. The resulting digital sine wave bits from both paths (row and column) are combined together to drive a 1-bit Sigma-Delta digital-analog converter (DAC).

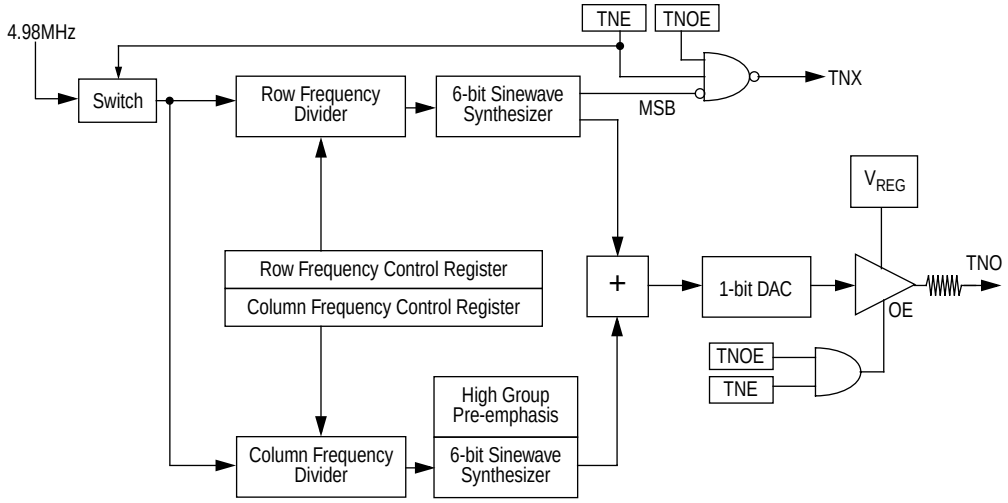


Figure 15-1. DTMF/Melody Generator Block Diagram

The method for generating a square wave tone in a path is similar to that of a sine wave tone except that only the most significant bit of the combined digital sine wave is used to drive the output driver. The resulting square wave tone has exactly the same frequency and phase as a sine wave tone for the same frequency control register value.

The pre-emphasis of the column frequency group is built into the sine wave synthesizer table therefore no additional hardware block is required. To reduce the total harmonic distortion, only an external capacitor is required to connect to the TNO output. The capacitor together with the built-in resistor forms a simple low pass filter with cutoff frequency at around 7KHz.

The generator provides not only DTMF and melody but also a square wave tone output (TNX). This signal is extracted from the most significant bit of the sine wave synthesizer of the row path.

15.5 DMG Registers

The DMG has four registers, Row Frequency Control register (FCR), Column Frequency Control register (FCC), Tone Control register (TNCR) and Tone Data Register (DTSUM)[1]. FCR and FCC select the

1. Tone Data Register (DTSUM) is for IC testing only. User should not read or write to this register.

row and column frequencies respectively. Tone Control register selects and controls the operating modes of the DMG module.

15.5.1 Frequency Control Registers — Row and Column

Register Name		Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Row Frequency Control Register (FCR)	R:	FCR7	FCR6	FCR5	FCR4	FCR3	FCR2	FCR1	FCR0	$0032
	W:									
Reset:		0	0	0	0	0	0	0	0	
Column Frequency Control Register (FCC)	R:	FCC7	FCC6	FCC5	FCC4	FCC3	FCC2	FCC1	FCC0	$0033
	W:									
Reset:		0	0	0	0	0	0	0	0	

Figure 15-2. Frequency Control Registers

FCR0-7 and FCC0-7 control the frequencies of the tone signals on the row and the column paths respectively. The register values for the generation of DTMF frequencies, modem frequencies and musical scale frequencies are shown in section **15.7 Modem Frequencies** and **15.8 Musical Scales Frequencies** respectively.

The relationship between the value of the frequency control registers and the tone output frequency is as following:

$$\text{Count} = [4980736 \div \{32 \times \text{target freq (Hz)}\}] - 1$$

If a zero is loaded into either the row or the column register, the corresponding path will be disabled and no tone will be generated from that path.

15.5.2 Tone Control Register (TNCR)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Tone Control Register (TNCR)	R:	0	0	0	0	TNE	TNXE	TNOE	TNOM	$0034
	W:									
Reset:		0	0	0	0	0	0	0	0	

Figure 15-3. Tone Control Register (TNCR)

This register controls the internal configuration and tone output timing of the DTMF/Melody Generator.

TNOM — TNO Mode Select

This bit controls the operating mode of the TNO output pin
1 = TNO outputs a square wave signal
0 = TNO outputs a sine wave signal

TNOE — TNO Output Enable

This bit controls the output stage of the TNO output pin.
1 = TNO output pin is in active mode
0 = TNO output pin is in tristate mode

TNXE — TNX Enable

This bit controls the output stage of the TNX output pin.
1 = TNX output pin is in active mode.
0 = TNX output pin is disabled. (output remains in logic “1” stage)

TNE — DTMF/Melody Generator Enable

This bit controls whether the DTMF/Melody generator is active or in power save mode.
1 = DTMF/Melody generator is active.
0 = DTMF/Melody generator is disabled and put in power save mode. TNO output pin is put into tristate mode.

15.6 DTMF Frequencies

Assuming the frequency input to the DTMF/Melody generator module is 4.980736MHz, the DTMF frequencies can be generated with the following values in the frequency control registers.

Table 15-1. DTMF Frequencies Generation

FCR/FCC	TONE	Standard Frequency (Hz)	Tone Output Frequency (Hz)	Frequency Deviation (Hz)	Frequency Deviation (%)
$DE	f_{R1}	697	698.0	1	0.14
$C9	f_{R2}	770	770.5	0.5	0.06
$B6	f_{R3}	852	850.5	1.5	–0.18
$A4	f_{R4}	941	943.3	2.3	0.24
$80	f_{C1}	1209	1206.6	2.4	–0.20
$74	f_{C2}	1336	1330.3	5.7	–0.43
$68	f_{C3}	1477	1482.4	5.4	–0.37
$5E	f_{C4}	1633	1638.4	5.5	–0.33

Table 15-2 shows the combinations of DTMF frequencies for the telephone keyboard symbols.

Table 15-2. DTMF Frequency Pairs for Telephone

Telephone Keypad Symbols	DTMF Frequency Pairs (Hz)	FCR	FCC
0	(941, 1336)	$A4	$74
1	(697, 1209)	$DE	$80
2	(697, 1336)	$DE	$74
3	(697, 1477)	$DE	$68
4	(770, 1209)	$C9	$80
5	(770, 1336)	$C9	$74
6	(770, 1477)	$C9	$68
7	(852, 1209)	$52	$80
8	(852, 1336)	$52	$74
9	(852, 1477)	$52	$68
A	(697, 1633)	$DE	$5E
B	(770, 1633)	$C9	$5E
C	(852, 1633)	$52	$5E
D	(941, 1633)	$A4	$5E
*	(941, 1209)	$A4	$80
#	(941, 1477)	$A4	$68

15.7 Modem Frequencies

Assuming the frequency input to the DTMF/Melody generator module is 4.980736MHz, the modem frequencies can be generated with the following values in the frequency control registers.

Standards	FCR/FCC	Standard Frequency (Hz)	Tone Output Frequency (Hz)	Frequency Deviation (Hz)	Frequency Deviation (%)
V.21	$9E	980	978.9	1.1	–0.11
	$83	1180	1179.2	0.8	–0.07
	$5D	1650	1655.8	5.8	0.35
	$53	1850	1853.0	3.0	0.16
Bell 103	$90	1070	1073.4	3.4	0.32
	$7A	1270	1265.4	4.6	–0.36
	$4C	2025	2021.4	3.6	–0.18
	$45	2225	2223.5	1.5	–0.07
Bell 202	$81	1200	1197.3	2.7	–0.23
	$46	2200	2192.2	7.8	–0.35
V.23	$77	1300	1297.1	2.9	–0.22
	$49	2100	2103.4	3.4	0.16

Table 15-3. Modem Frequencies Generation

15.8 Musical Scales Frequencies

Assuming the frequency input to the DTMF/Melody generator module is 4.980736MHz, the two octaves of musical scale in semitones can be generated with the following values in the frequency control register.

Table 15-4. Musical Scale Frequencies Generation

Note	FCR/FCC	Frequency (Hz)		Deviation (%)
		Standard	Generated	
D#5	$F9	622.3	622.3	0.05
E5	$EB	659.3	659.5	0.03
F5	$DE	698.5	698.0	–0.07
F#5	$D1	740.0	741.2	0.16
G5	$C6	784.0	782.2	–0.23
G#5	$BA	830.6	832.3	0.20
A5	$B0	880.0	879.4	–0.07
A#5	$A8	923.3	921.0	–0.25
B5	$9D	987.8	985.1	–0.27
C6	$94	1046.5	1044.6	–0.18
C#6	$8B	1108.7	1111.8	0.28
D6	$84	1174.7	1170.3	–0.37
D#6	$7C	1244.5	1245.2	0.06
E6	$75	1318.5	1319.1	0.05
F6	$6E	1396.9	1402.2	0.38
F#6	$68	1480.0	1482.4	0.16
G6	$62	1568.0	1572.2	0.27
G#6	$5D	1661.2	1655.8	–0.33
A6	$57	1760.0	1768.7	0.49
A#6	$52	1864.7	1875.3	0.57
B6	$4E	1975.5	1970.2	–0.27
C7	$49	2093.0	2103.4	0.50
C#7	$45	2217.5	2223.5	0.27
D7	$41	2349.3	2358.3	0.38
D#7	$3E	2489.0	2470.6	–0.74

15.9 DMG Programming

This section gives some examples on how to program the DMG registers for different types of applications.

15.9.1 DTMF Generation

The following sequence shows how to generate the DTMF for key “0”.

- Load $02 into TNCR to disable DMG and TNX output, and select sine wave output from TNO.
- load $A4 into FCR to select row frequency of 941Hz.

- load $74 into FCC to select column frequency of 1336Hz.
- load $0A into TNCR to enable the DMG.

15.9.2 Single Tone Sine Wave Generation

The following sequence shows how to generate a single tone sine wave (D#5) from TNO. Either the row or the column frequency can be used. However, due to the pre-emphasis built into the column path, the tone generated from this path will have a slightly larger amplitude than the row path.

- Load $02 into TNCR to disable DMG and TNX output, and select sine wave output from TNO.
- load $00 into FCR to disable the row path.
- load $F9 into FCC to select column frequency of 622.3Hz.
- load $0A into TNCR to enable the DMG.

15.9.3 Single Tone Square Wave Generation from TNX

The following sequence shows how to generate a single tone sine wave (D#5) from TNX.

- Load $00 into TNCR to disable DMG, TNX output and TNO output.
- load $F9 into FCR to disable the row path.
- load $00 into FCC to disable the column frequency path. (This step can be ignored if TNO is disabled)
- load $0C into TNCR to enable the DMG.

15.10 Low-Power Modes

The WAIT and STOP instructions put the MCU in low-power-consumption standby modes. By setting TNE bit to ”0” will disable the DMG to reduce power consumption.

15.10.1 Wait/Stop Mode

The DMG module is inactive after the execution of a wait/stop mode. The WAIT/STOP instruction does not affect DMG register states. DMG module resumes after external interrupt.

Section 16. I/O Ports

16.1 Contents

16.2 Introduction

36 bidirectional input-output (I/O) pins form 5 parallel ports (20 bidirectional I/O pins for 28-pin package). 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.*

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Port A Data Register (PTA)	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0	$0000
Port B Data Register (PTB)	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0	$0001
Port C Data Register (PTC)	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0	$0002
Port D Data Register (PTD)	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0	$0003
Data Direction Register A (DDRA)	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0	$0004
Data Direction Register B (DDRB)	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0	$0005
Data Direction Register C (DDRC)	0	0	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0	$0006
Data Direction Register D (DDRD)	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0	$0007
Port E Data Register (PTE)	0	0	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0	$0008
Data Direction Register E (DDRE)	0	0	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0	$000A

Figure 16-1. I/O Port Register Summary

Table 16-1. Port Control Register Bits Summary

Port	bit	DDR	Module Control		Pin
A	0	DDRA0	KBAIE	KBAIE0	PTA0/KBA0
	1	DDRA1		KBAIE1	PTA1/KBA1
	2	DDRA2		KBAIE2	PTA2/KBA2
	3	DDRA3		KBAIE3	PTA3/KBA3
	4	DDRA4		KBAIE4	PTA4/KBA4
	5	DDRA5		KBAIE5	PTA5/KBA5
	6	DDRA6		KBAIE6	PTA6/KBA6
	7	DDRA7		KBAIE7	PTA7/KBA7
B	0	DDRB0			PTB0
	1	DDRB1			PTB1
	2	DDRB2			PTB2
	3	DDRB3			PTB3
	4	DDRB4			PTB4
	5	DDRB5			PTB5
	6	DDRB6			PTB6
	7	DDRB7			PTB7
C	0	DDRC0			PTC0
	1	DDRC1			PTC1
	2	DDRC2			PTC2
	3	DDRC3			PTC3
	4	DDRC4			PTC4
	5	DDRC5			PTC5
D	0	DDRD0	POC	TNXMUX	PTD0/TNX
	1	DDRD1		RDMUX	PTD1/RD
	2	DDRD2			PTD2/$\overline{\text{RT}}$
	3	DDRD3		SPIMUX	PTD3/SCK
	4	DDRD4			PTD4/MOSI
	5	DDRD5			PTD5/MISO
	6	DDRD6			PTD6/$\overline{\text{SS}}$
	7	DDRD7		IRQ2MUX	PTD7/$\overline{\text{IRQ2}}$
E	0	DDRE0	TIM	PS[2:0]	PTE0/TCLK
	1	DDRE1		ELS0A/B	PTE1/TCH0
	2	DDRE2		ELS1A/B	PTE2/TCH1
	3	DDRE3		ELS2A/B	PTE3/TCH2
	4	DDRE4		ELS3A/B	PTE4/TCH3
	5	DDRE5			PTE5

16.3 Port A

Port A is an 8-bit general purpose bidirectional I/O port.

16.3.1 Port A Data Register (PTA)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTA $0000	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
	Reset:	Unaffected by reset							
	Alternate Function:	KBA7	KBA6	KBA5	KBA4	KBA3	KBA2	KBA1	KBA0

Figure 16-2. Port A Data Register (PTA)

PTA[7:0] — 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.

KBA[7:0] — Keyboard Inputs

The keyboard interrupt enable bits, KBAIE[7:0], in the keyboard interrupt control register (KBICR), enable the port A pins as external interrupt pins. (See **Section 18. Keyboard Interrupt Module (KBI)**.)

16.3.2 Data Direction Register A (DDRA)

Data direction register A determines whether each port A pin is an input or an output. Writing a logic one to a DDRA bit enables the output buffer for the corresponding port A pin; a logic zero disables the output buffer.

		Bit 7	6	5	4	3	2	1	Bit 0
DDRA $0004	Read: / Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
	Reset:	0	0	0	0	0	0	0	0

Figure 16-3. Data Direction Register A (DDRA)

DDRA[7:0] — Data Direction Register A Bits

These read/write bits control port A data direction. Reset clears DDRA[7:0], 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 16-4 shows the port A I/O logic.

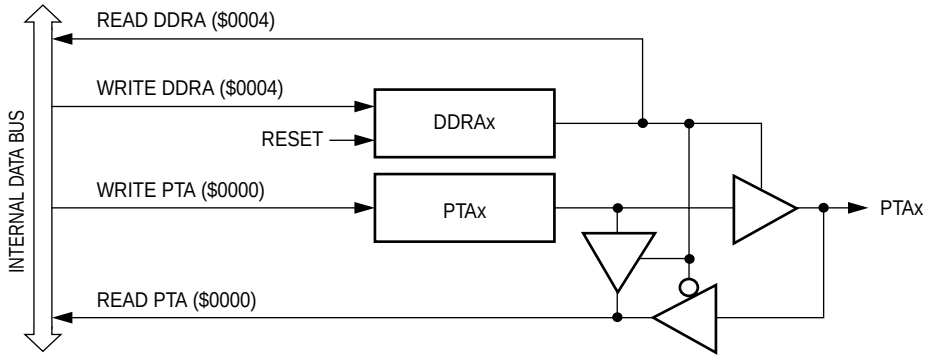


Figure 16-4. Port A I/O Circuit

When bit DDRAx is a logic one, reading address $0000 reads the PTAx data latch. When bit DDRAx is a logic zero, 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 16-1** summarizes the operation of the port A pins.

Table 16-2. Port A Pin Functions

DDRA Bit	PTA Bit	I/O Pin Mode	Accesses to DDRA	Accesses to PTA	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRA[7:0]	Pin	PTA[7:0](3)
1	X	Output	DDRA[7:0]	PTA[7:0]	PTA[7:0]

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

16.4 Port B

Port B is an 8-bit general purpose bidirectional I/P port.

16.4.1 Port B Data Register (PTB)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTB $0001	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
	Reset:	Unaffected by reset							

Figure 16-5. 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.

16.4.2 Data Direction Register B (DDRB)

Data direction register B determines whether each port B pin is an input or an output. Writing a logic one to a DDRB bit enables the output buffer for the corresponding port B pin; a logic zero disables the output buffer.

		Bit 7	6	5	4	3	2	1	Bit 0
DDRB $0005	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
	Reset:	0	0	0	0	0	0	0	0

Figure 16-6. 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 16-7 shows the port B I/O logic.

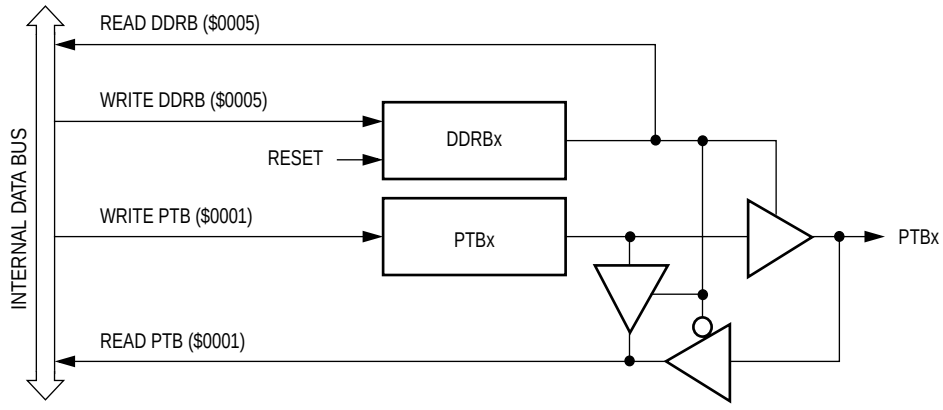


Figure 16-7. Port B I/O Circuit

When bit DDRBx is a logic one, reading address $0001 reads the PTBx data latch. When bit DDRBx is a logic zero, 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 16-3** summarizes the operation of the port B pins.

Table 16-3. Port B Pin Functions

DDRB Bit	PTB Bit	I/O Pin Mode	Accesses to DDRB	Accesses to PTB	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRB[7:0]	Pin	PTB[7:0](3)
1	X	Output	DDRB[7:0]	PTB[7:0]	PTB[7:0]

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

16.5 Port C

Port C is an 6-bit general purpose bidirectional I/O port.

16.5.1 Port C Data Register (PTC)

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

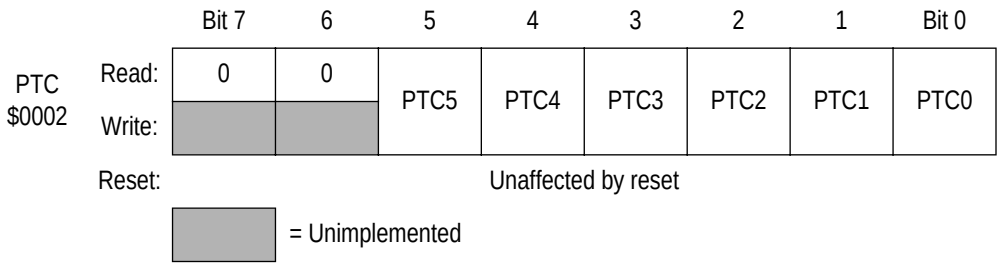


Figure 16-8. Port C Data Register (PTC)

PTC[5:0] — 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.

16.5.2 Data Direction Register C (DDRC)

Data direction register C determines whether each port C pin is an input or an output. Writing a logic one to a DDRC bit enables the output buffer for the corresponding port C pin; a logic zero disables the output buffer.

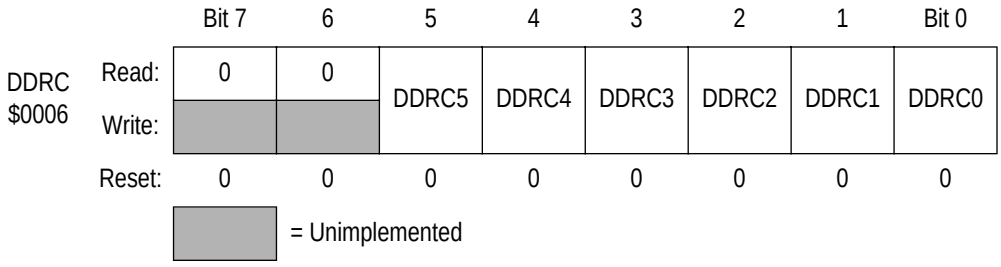


Figure 16-9. Data Direction Register C (DDRC)

DDRC[5:0] — Data Direction Register C Bits

These read/write bits control port C data direction. Reset clears DDRC[4:0], 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 16-10 shows the port C I/O logic.

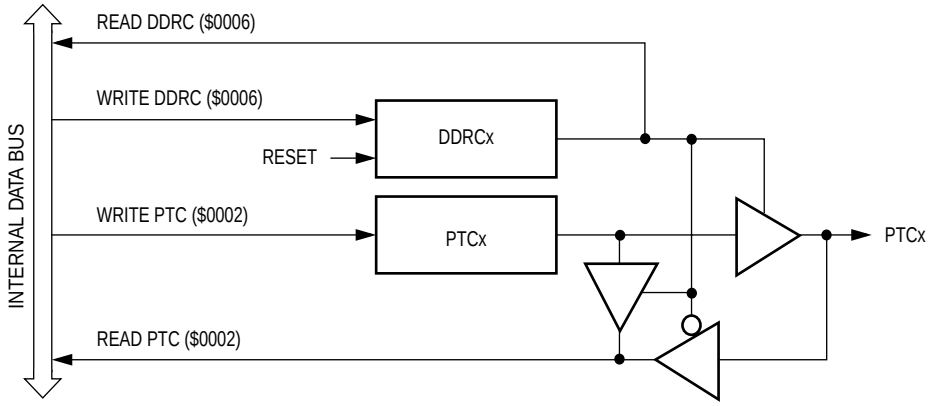


Figure 16-10. Port C I/O Circuit

When bit DDRCx is a logic one, reading address $0002 reads the PTCx data latch. When bit DDRCx is a logic zero, 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 16-4** summarizes the operation of the port C pins.

Table 16-4. Port C Pin Functions

DDRC Bit	PTC Bit	I/O Pin Mode	Accesses to DDRC	Accesses to PTC	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRC[4:0]	Pin	PTC[4:0](3)
1	X	Output	DDRC[4:0]	PTC[4:0]	PTC[4:0]

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

16.6 Port D

Port D is an 8-bit special function port that shares all eight of its pins with DTMF/Melody Generator, Ring Detector, External Interrupt and Serial Peripheral Interface module (SPI).

16.6.1 Port D Data Register (PTD)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTD $0003	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
	Reset:	Unaffected by reset							
	Alternate Function:	$\overline{IRQ2}$	$\overline{SS}$	MISO	MOSI	SCK	$\overline{RT}$	RD	TNX

Figure 16-11. Port D Data Register (PTD)

PTD[7:0] — Port D Data Bits

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

SCK — Serial Clock

The PTD3/SCK pin is the serial clock of the SPI module. When SPIMUX in POC is cleared, the SPI module is disabled, and PTD3/SCK pin is available for general-purpose I/O.

The serial clock synchronizes data transmission between master and slave devices. In master mode, the SCK pin is the clock output. In slave mode, the SCK pin is the clock input.

MISO — Master In/Slave Out

The PTD5/MISO pin is the master in/slave out terminal of the SPI module. When SPIMUX in POC is cleared, the SPI module is disabled, and PTD5/MISO pin is available for general-purpose I/O.

Data direction register D (DDRD) does not affect the data direction of port D pins that are being used by the SPI module. However, the DDRD bits always determine whether reading port D returns the states of the latches or the states of the pins. (See **Table 16-4 . Port C Pin Functions**.)

MOSI — Master Out/Slave In

The PTD4/MOSI pin is the master out/slave in terminal of the SPI module. When SPIMUX in POC is cleared, PTD4/MOSI pin is available for general-purpose I/O.

$\overline{SS}$ — Slave Select

The PTD6/$\overline{SS}$ pin is the slave select input of the SPI module. When SPIMUX in POC is cleared, or when the SPI master bit, SPMSTR, is set, PTD6/$\overline{SS}$ pin is available for general-purpose I/O. When the SPI is enabled, the DDRB2 bit in data direction register B (DDRB) has no effect on the PTD6/$\overline{SS}$ pin.

$\overline{IRQ2}$ — External Interrupt Request 2

The PTD7/$\overline{IRQ2}$ pin is the second external interrupt request input . When IRQ2MUX in POC is cleared, PTD7/$\overline{IRQ2}$ pin is available for general-purpose I/O.

$\overline{RT}$ — Ring Time

The PTD2/$\overline{RT}$ pin is the input to control the timing of the ring detector module. When the RDMUX in POC is cleared, PTD2/$\overline{RT}$ pin is available for general-purpose I/O.

RD — Ring Detect

The PTD1/RDI pin is the input pin to the ring detector module. When the RDMUX in POC is cleared, PTD1/RDI pin is available for general-purpose I/O.

TNX — Melody Output

The PTD0/TNX pin is the melody output from the DTMF/Melody generator module. When the TNXMUX in POC is cleared, PTD0/TNX pin is available for general-purpose I/O.

16.6.2 Data Direction Register D (DDRD)

Data direction register D determines whether each port D pin is an input or an output. Writing a logic one to a DDRD bit enables the output buffer for the corresponding port D pin; a logic zero disables the output buffer.

		Bit 7	6	5	4	3	2	1	Bit 0
DDRD $0007	Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
	Reset:	0	0	0	0	0	0	0	0

Figure 16-12. Data Direction Register D (DDRD)

DDRD[7:0] — Data Direction Register D Bits

These read/write bits control port D data direction. Reset clears DDRD[7:0], 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 16-13 shows the port D I/O logic.

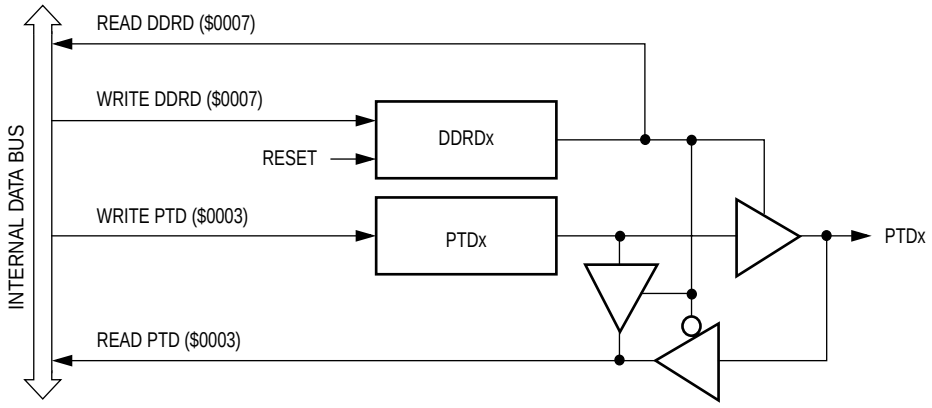


Figure 16-13. Port D I/O Circuit

When bit DDRDx is a logic one, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic zero, 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 16-5** summarizes the operation of the port D pins.

Table 16-5. Port D Pin Functions

DDRD Bit	PTD Bit	I/O Pin Mode	Accesses to DDRD	Accesses to PTD	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRD[7:0]	Pin	PTD[7:0](3)
1	X	Output	DDRD[7:0]	PTD[7:0]	PTD[7:0]

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

16.7 Port E

Port E is a 6-bit special function port that shares four of its pins with the timer interface (TIM) module.

16.7.1 Port E Data Register (PTE)

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

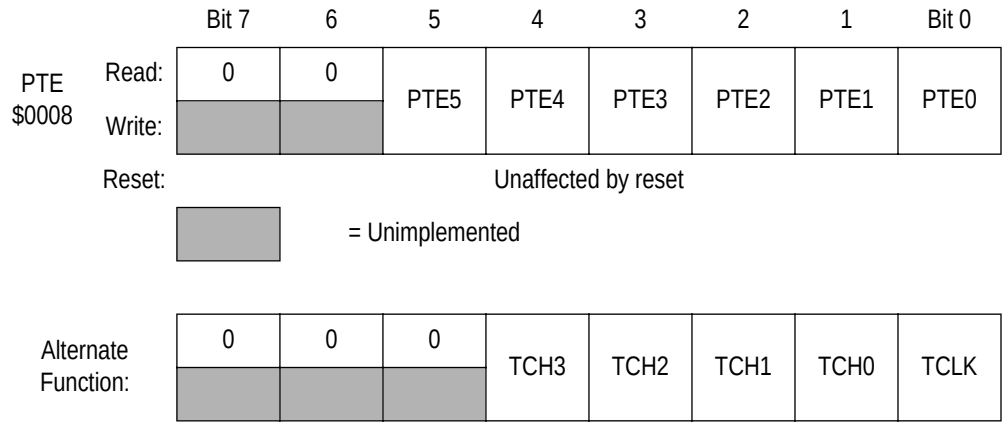


Figure 16-14. Port E Data Register (PTE)

PTE[5:0] — Port E Data Bits

PTE[5:0] are read/write, software programmable bits. Data direction of each port E pin is under the control of the corresponding bit in data direction register E.

NOTE: *Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the TIM. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See* ***Table 16-6 . Port E Pin Functions****.).*

TCH[3:0] — Timer Channel I/O Bits

The PTE3/TCH3–PTE0/TCH0 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTE4/TCH3–PTE1/TCH0 pins are timer channel I/O pins or general-purpose I/O pins. (See **Section 11. Timer Interface Module (TIM)**.)

TCLK — Timer Clock Pin

PTE0/TCLK is an external clock input that can be the clock source for the TIM counter instead of the prescaled internal bus clock. Select the PTE0/TCLK input by writing logic ones to the three prescaler select bits, PS[2:0]. See **11.9.1 TIM Status and Control Register (TSC)**. The minimum TCLK pulse width, $TCLK_{LMIN}$ or $TCLK_{HMIN}$, is:

$$\frac{1}{\text{bus frequency}} + t_{SU}$$

The maximum TCLK frequency is: bus frequency ÷ 2

PTE0/TCLK is available as a general-purpose I/O pin when not used as the TIM clock input. When the PTE0/TCLK pin is the TIM clock input, it is an input regardless of the state of the DDRE3 bit in data direction register E.

16.7.2 Data Direction Register E (DDRE)

Data direction register E determines whether each port E pin is an input or an output. Writing a logic one to a DDRE bit enables the output buffer for the corresponding port E pin; a logic zero disables the output buffer.

		Bit 7	6	5	4	3	2	1	Bit 0
DDRE $0009	Read:	0	0	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

Figure 16-15. Data Direction Register E (DDRE)

DDRE[5:0] — Data Direction Register E Bits

These read/write bits control port E data direction. Reset clears DDRE[5:0], 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 16-16 shows the port E I/O logic.

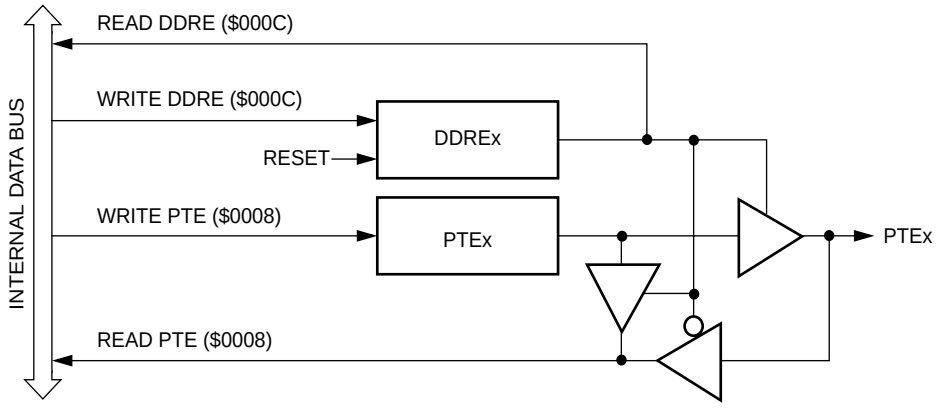


Figure 16-16. Port E I/O Circuit

When bit DDREx is a logic one, reading address $0008 reads the PTEx data latch. When bit DDREx is a logic zero, 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. **Figure 16-8** summarizes the operation of the port E pins.

Table 16-6. Port E Pin Functions

DDRE Bit	PTE Bit	I/O Pin Mode	Accesses to DDRE	Accesses to PTE	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRE[5:0]	Pin	PTE[5:0](3)
1	X	Output	DDRE[5:0]	PTE[5:0]	PTE[5:0]

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

16.8 Port Handling for Small Package

When this chip is put into a smaller package using bonding options, special care must be taken in handling the I/O ports in order to minimize the leakage current and the total current consumption. All the ports that are not bonded out should be programmed as output ports with a value of logic zero.

Section 17. External Interrupt (IRQ)

17.1 Contents

17.2 Introduction

The IRQ provides two maskable interrupt inputs.

17.3 Features

Features of the IRQ module include the following:

- Two External Interrupt Pins ($\overline{IRQ1}$ and $\overline{IRQ2}$)
- Separate IRQ1 and IRQ2 Interrupt Control Bits
- Hysteresis Buffers
- Programmable Edge-only or Edge and Level Interrupt Sensitivity
- Automatic Interrupt Acknowledge

17.4 Functional Description

A logic zero applied to any of the external interrupt pins can latch a CPU interrupt request. **Figure 17-1** shows the structure of the IRQ module.

Interrupt signals on the $\overline{\text{IRQ1}}$ pin are latched into the IRQ1 latch. Interrupt signals on the $\overline{\text{IRQ2}}$ pin are latched into the IRQ2 interrupt latch. An interrupt latch remains set until one of the following 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 one to the ACK1 bit clears the IRQ1 latch. Writing a logic one to the ACK2 bit clears the IRQ2 interrupt latch.
- Reset — A reset automatically clears both interrupt latches.

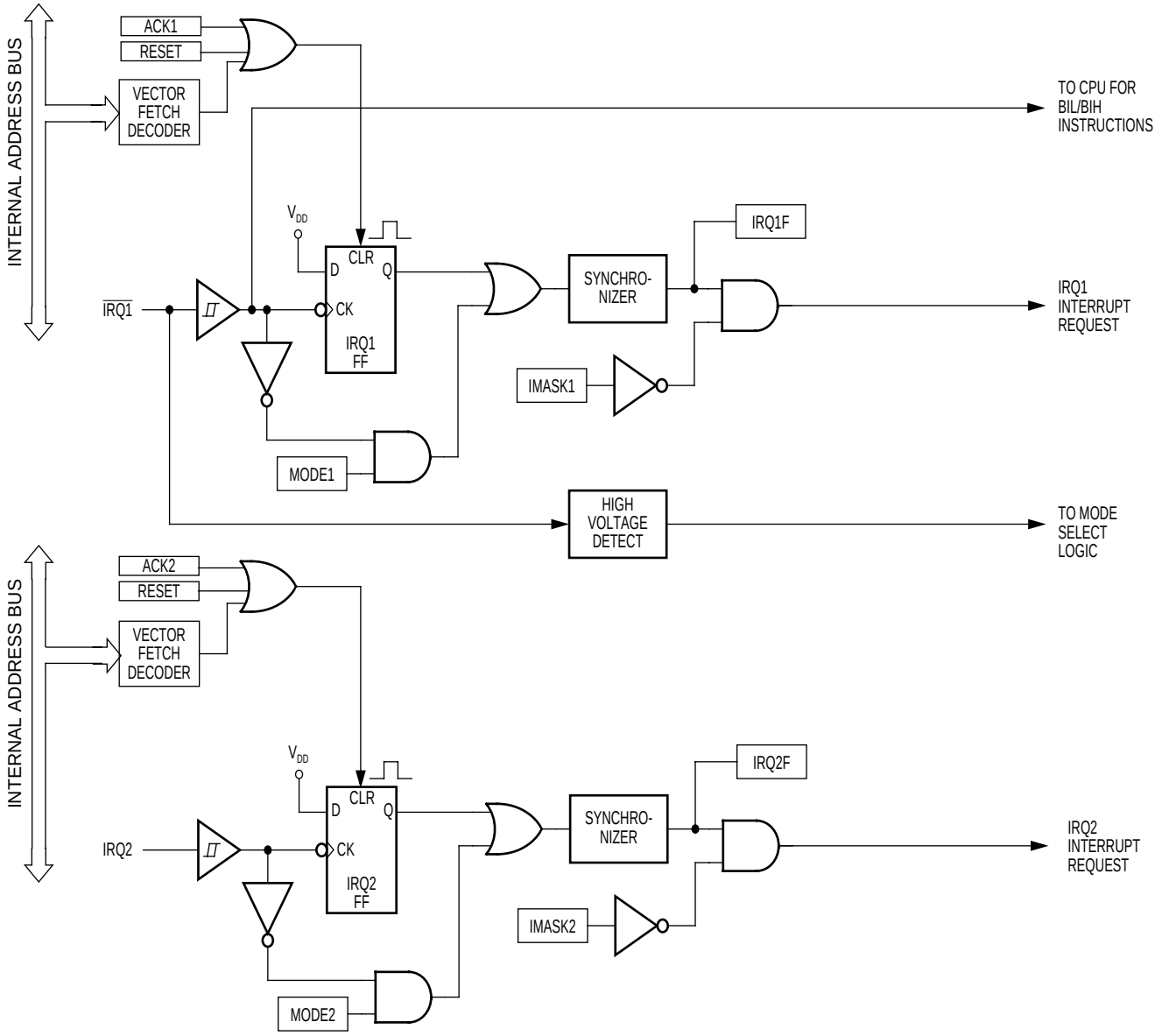


Figure 17-1. IRQ Module Block Diagram

Register Name		Bit 7	6	5	4	3	2	1	Bit 0	Addr.
IRQ Status/Control Register (ISCR)	R:	IRQF2	0	IMASK2	MODE2	IRQF1	0	IMASK1	MODE1	$003D
	W:		ACK2				ACK1			

= Unimplemented

Figure 17-2. IRQ I/O Register Summary

All of the external interrupt pins are falling-edge-triggered and are software configurable to be both falling-edge and low-level-triggered. The MODE1 bit in the ISCR controls the triggering sensitivity of the $\overline{\text{IRQ1}}$ pin. The MODE2 bit controls the triggering sensitivity of the $\overline{\text{IRQ2}}$ pin.

When an interrupt pin is edge-triggered only, the interrupt 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 remains set until both of the following occur:

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

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

When set, the IMASK1 and IMASK2 bits in the ISCR mask 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 17-3 . IRQ Interrupt Flowchart****.)*

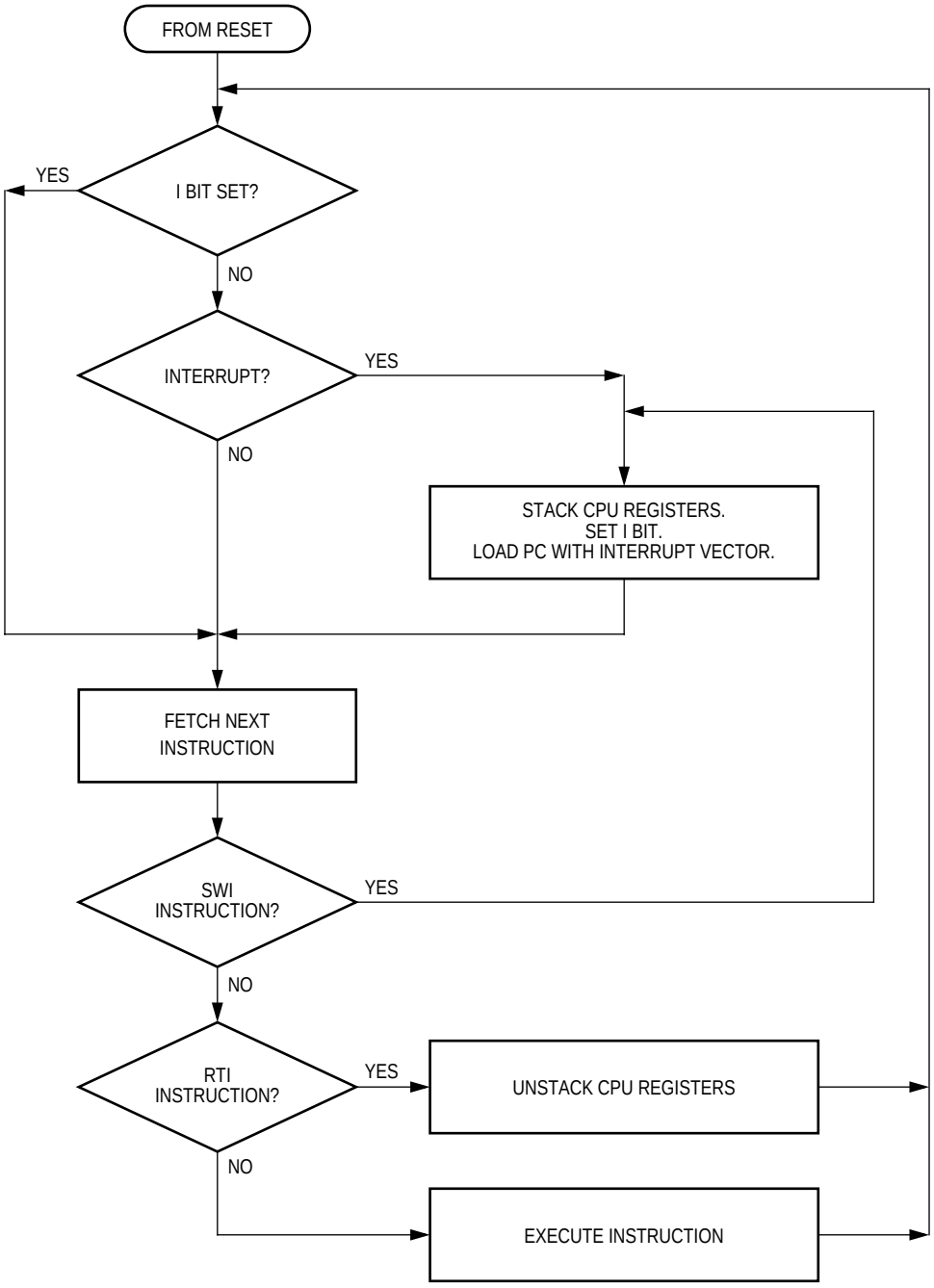


Figure 17-3. IRQ Interrupt Flowchart

17.4.1 $\overline{\text{IRQ1}}$ /V_{pp} Pin

A logic zero on the $\overline{\text{IRQ1}}$ pin can latch an interrupt request into the IRQ1 latch. A vector fetch, software clear, or reset clears the IRQ1 latch.

- If the MODE1 bit is set, the $\overline{\text{IRQ1}}$ pin is both falling-edge-sensitive and low-level-sensitive. With MODE1 set, both of the following actions must occur to clear IRQ1:
- 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 one to the ACK1 bit in the interrupt status and control register (ISCR). The ACK1 bit is useful in applications that poll the $\overline{\text{IRQ1}}$ pin and require software to clear the IRQ1 latch. Writing to the ACK1 bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACK1 does not affect subsequent transitions on the $\overline{\text{IRQ1}}$ pin. A falling edge that occurs after writing to the ACK1 bit latches another interrupt request. If the IRQ1 mask bit, IMASK1, is clear, the CPU loads the program counter with the vector address at locations $FFF8 and $FFF9.
- Return of the $\overline{\text{IRQ1}}$ pin to logic one — As long as the $\overline{\text{IRQ1}}$ pin is at logic zero, IRQ1 remains active.

The vector fetch or software clear and the return of the $\overline{\text{IRQ1}}$ pin to logic one may occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ1}}$ pin is at logic zero. A reset will clear the latch and the MODEx control bit, thereby clearing the interrupt even if the pin stays low.

If the MODE1 bit is clear, the $\overline{\text{IRQ1}}$ pin is falling-edge-sensitive only. With MODE1 clear, a vector fetch or software clear immediately clears the IRQ1 latch.

The IRQF1 bit in the ISCR register can be used to check for pending interrupts. The IRQF1 bit us not affected by the IMASK1 bit, which makes it useful in applications where polling is preferred.

Use the BIH or BIL instruction to read the logic level on the $\overline{\text{IRQ1}}$ pin.

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

17.4.2 $\overline{\text{IRQ2}}$ Pin

A logic zero on the $\overline{\text{IRQ2}}$ pin can latch an interrupt request into the IRQ2 interrupt latch. A vector fetch, software clear, or reset clears the IRQ2 interrupt latch.

If the MODE2 bit is set, the $\overline{\text{IRQ2}}$ pin is both falling-edge-sensitive and low-level-sensitive. With MODE2 set, both of the following actions must occur to clear IRQ2:

- Vector fetch or software clear, or reset — A vector fetch generates an interrupt acknowledge signal to clear the latch. Software may generate the interrupt acknowledge signal by writing a logic one to the ACK2 bit in the interrupt status and control register (ISCR). The ACK2 bit is useful in applications that poll the $\overline{\text{IRQ2}}$ pin and require software to clear the IRQ2 interrupt latch. Writing to the ACK2 bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACK2 does not affect subsequent transitions on the $\overline{\text{IRQ2}}$ pin. A falling edge that occurs after writing to the ACK2 bit latches another interrupt request. If the IRQ2 mask bit, IMASK2, is clear, the CPU loads the program counter with the vector address at locations $FFF6 and $FFF7.
- Return of the $\overline{\text{IRQ2}}$ pin to logic one — As long as the $\overline{\text{IRQ2}}$ pin is at logic zero, IRQ2 remains active.

The vector fetch or software clear and the return of the $\overline{\text{IRQ2}}$ pin to logic one may occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ2}}$ pin is at logic zero. A reset will clear the latch and the MODEx control bit, thereby clearing the interrupt even if the pin stays low.

If the MODE2 bit is clear, the $\overline{\text{IRQ2}}$ pin is falling-edge-sensitive only. With MODE2 clear, a vector fetch or software clear immediately clears the IRQ2 interrupt latch.

There is no direct way to determine the logic level on the $\overline{IRQ2}$ pin. However, it is possible to use the IRQF2 bit in the ISCR to infer the state of the $\overline{IRQ2}$ pin. By writing a one to the MODE2 bit, the IRQF2 bit in the ISCR will be the opposite value of the $\overline{IRQ2}$ pin as long as the $\overline{IRQ2}$ latch is cleared. (See **Figure 17-1**.) The $\overline{IRQ2}$ latch can be cleared by writing a one to the acknowledge bit. Recall, however, that every falling edge on the $\overline{IRQ2}$ pin will set the $\overline{IRQ2}$ latch, so an additional acknowledge is necessary after each falling edge on $\overline{IRQ2}$ to maintain the opposite relationship between IRQF2 and the $\overline{IRQ2}$ pin. The user may want to set the IMASK2 bit in the ISCR to prevent the IRQF2 from generating interrupts when used in this manner.

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

17.5 IRQ Module During Break Interrupts

The system integration module (SIM) controls whether the IRQ1 and IRQ2 interrupt latches 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. (See **Section 7. System Integration Module (SIM)**.)

To allow software to clear the IRQ1 and the IRQ2 interrupt latches during a break interrupt, write a logic one 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 latches during the break state, write a logic zero to the BCFE bit. With BCFE at logic zero (its default state), writing to the ACK1 and ACK2 bits in the IRQ status and control register during the break state has no effect on the IRQ latches.

17.6 IRQ Status and Control Register (ISCR)

The IRQ Status and Control Register (ISCR) controls and monitors operation of the IRQ module. The ISCR has the following functions:

- Shows the state of the IRQ1 and IRQ2 interrupt flags
- Clears the IRQ1 and IRQ2 interrupt latches
- Masks IRQ1 and IRQ2 interrupt requests
- Controls triggering sensitivity of the $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ interrupt pins

ISCR $003D	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IRQF2	0	IMASK2	MODE2	IRQF1	0	IMASK1	MODE1
Write:		ACK2				ACK1		
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-4. IRQ Status and Control Register (ISCR)

IRQF2 — $\overline{\text{IRQ2}}$ Flag

This read-only status bit is high when the $\overline{\text{IRQ2}}$ interrupt is pending.

1 = $\overline{\text{IRQ2}}$ interrupt pending
0 = $\overline{\text{IRQ2}}$ interrupt not pending

ACK2 — IRQ2 Interrupt Request Acknowledge Bit

Writing a logic one to this write-only bit clears the IRQ2 interrupt latch. ACK2 always reads as logic zero. Reset clears ACK2.

IMASK2 — IRQ2 Interrupt Mask Bit

Writing a logic one to this read/write bit prevents the output of the IRQ2 interrupt latch from generating interrupt requests. Reset clears IMASK2.

1 = $\overline{\text{IRQ2}}$ pin interrupt requests disabled
0 = $\overline{\text{IRQ2}}$ pin interrupt requests enabled

MODE2 — IRQ2 Interrupt Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the IRQ2 interrupt pins. Reset clears MODE2.

1 = $\overline{\text{IRQ2}}$ interrupt requests on falling edges and low levels
0 = $\overline{\text{IRQ2}}$ interrupt requests on falling edges only

IRQF1 — IRQ1 Flag

This read-only status bit is high when the IRQ1 interrupt is pending.
1 = $\overline{\text{IRQ1}}$ interrupt pending
0 = $\overline{\text{IRQ1}}$ interrupt not pending

ACK1 — IRQ1 Interrupt Request Acknowledge Bit

Writing a logic one to this write-only bit clears the IRQ1 latch. ACK1 always reads as logic zero. Reset clears ACK1.

IMASK1 — IRQ1 Interrupt Mask Bit

Writing a logic one to this read/write bit disables IRQ1 interrupt requests. Reset clears IMASK1.
1 = IRQ1 interrupt requests disabled
0 = IRQ1 interrupt requests enabled

MODE1 — IRQ1 Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ1}}$ pin. Reset clears MODE1.
1 = $\overline{\text{IRQ1}}$ interrupt requests on falling edges and low levels
0 = $\overline{\text{IRQ1}}$ interrupt requests on falling edges only

Section 18. Keyboard Interrupt Module (KBI)

18.1 Contents

18.2 Introduction

The keyboard interrupt module provides eight independently maskable external interrupt pins.

18.3 Features

Features of the keyboard interrupt module include the following:

- Eight Keyboard Interrupt Pins and Interrupt Masks
- Selectable triggering polarity

18.4 Functional Description

Writing to the KBAIE7–KBAIE0 bits in the keyboard interrupt enable register independently enables or disables each port A pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its internal pull-up device. A logic zero applied to an enabled keyboard interrupt pin will latch a keyboard interrupt request.

The keyboard interrupt latch becomes set when one or more keyboard pins goes low after all were high. The MODED 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 edge- and level-sensitive, an interrupt request is present as long as any keyboard pin is low.

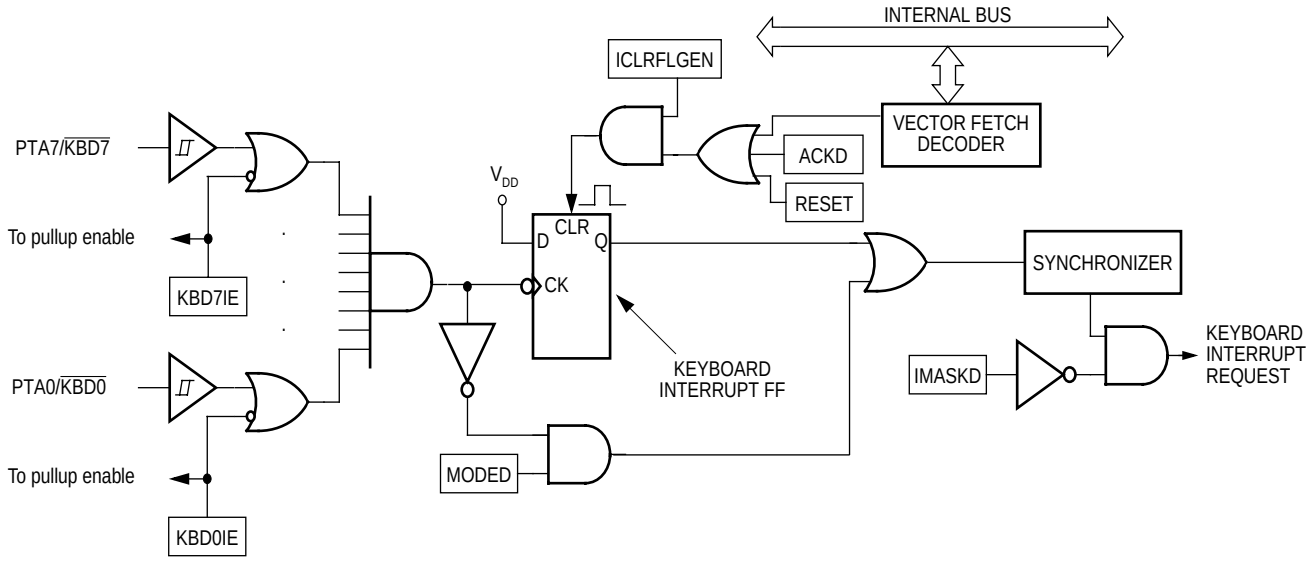


Figure 18-1. Keyboard Module Block Diagram

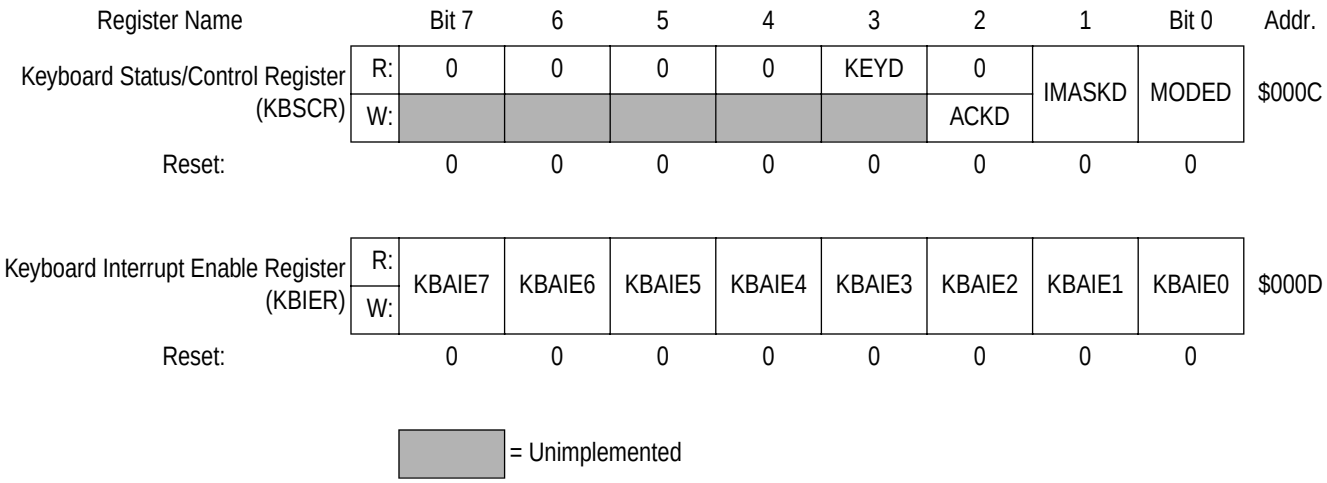


Figure 18-2. KB I/O Register Summary

The MODED bit in the keyboard status and control register controls the triggering sensitivity of the keyboard interrupt latch. If the MODED 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:

Software clear — Software may generate the interrupt acknowledge signal by writing a logic one to the ACKD bit in the keyboard status and control register (KBSCR). The ACKD bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt latch. Writing to the ACKD bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACKD does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKD bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKD, is clear, the CPU loads the program counter with the vector address at locations $FFEA and $FFEB.

- Return of all enabled keyboard interrupt pins to logic one — As long as any enabled keyboard interrupt pin is at logic zero, the keyboard interrupt remains set.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to logic one may occur in any order. The interrupt request

remains pending as long as any enabled keyboard interrupt pin is at logic zero.

If the MODED bit is clear, the keyboard interrupt pin is falling-edge-sensitive only. With MODED clear, a vector fetch or software clear immediately clears the keyboard interrupt latch.

Reset clears the keyboard interrupt latch and the MODED bit, clearing the interrupt request even if a keyboard interrupt pin stays at logic zero.

The keyboard flag bit (KEYDF) in the keyboard status and control register can be used to see if a pending interrupt exists. The KEYDF bit is not affected by the keyboard interrupt mask bit (IMASKD) 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 (KBAIEx) 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 zero for software to read the pin.*

18.5 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pull-up to reach a logic one, so it is possible that an interrupt could occur when the pin is initially enabled.

To prevent this, it is recommended that the keyboard interrupt be masked with the IMASKD bit in the KBSCR register before enabling the pin, and the ACKD bit be used to acknowledge the potential false interrupt before setting IMASKD back to zero. An edge-only type of interrupt can be acknowledged immediately after enabling the pin. An edge and level triggered interrupt must be acknowledged after a delay which is dependent on the external load.

Another way to avoid a false interrupt is to set the appropriate bit of port A to an output driving a logic one, before enabling the keyboard input.

18.6 I/O Registers

The following registers control and monitor operation of the keyboard interrupt module:

- Keyboard status and control register (KBSCR)
- Keyboard interrupt enable register (KBIER)

18.6.1 Keyboard Status and Control Register (KBSCR)

The keyboard status and control register performs the following functions:

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

		Bit 7	6	5	4	3	2	1	Bit 0
KBSCR $000C	Read:	0	0	0	0	KEYD	0	IMASKD	MODED
	Write:						ACKD		
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 18-3. Keyboard Status and Control Register (KBDSCR)

Bits [7:4] — Not used

These read-only bits always read as logic zeros.

KEYDF — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending. Reset clears the KEYDF bit.

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

ACKD — Keyboard Acknowledge Bit

Writing a logic one to this read/write bit clears the keyboard interrupt latch. ACKD always reads as logic zero. Reset clears ACKD.

IMASKD — Keyboard Interrupt Mask Bit

Writing a logic one to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests. Reset clears the IMASKD bit.

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

MODED — Keyboard Triggering Sensitivity Bit

This read/write bit controls the triggering sensitivity of the keyboard interrupt pins. Reset clears MODE2.

1 = Keyboard interrupt requests on falling edges and low levels
0 = Keyboard interrupt requests on falling edges only

18.6.2 Keyboard Interrupt Enable Register (KBIER)

The keyboard interrupt enable register enables or disables each port A pin to operate as a keyboard interrupt pin.

		Bit 7	6	5	4	3	2	1	Bit 0
KBIER $000D	R: W:	KBAIE7	KBAIE6	KBAIE5	KBAIE4	KBAIE3	KBAIE2	KBAIE1	KBAIE0
	Reset:	0	0	0	0	0	0	0	0

Figure 18-4. Keyboard Interrupt Enable Register (KBIER)

KBAIE[7:0] — 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 = PAx pin enabled as keyboard interrupt pin
0 = PAx pin not enabled as keyboard interrupt pin

18.7 Keyboard Module During Break Interrupts

The system integration module (SIM) controls whether the keyboard 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 latch during the break state.

To allow software to clear the keyboard interrupt latch during a break interrupt, write a logic one 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 zero to the BCFE bit. With BCFE at logic zero (its default state), writing during the break state to the keyboard acknowledge bit (ACKD) in the keyboard status and control register has no effect. (See **18.6.1 Keyboard Status and Control Register (KBSCR)**.)

Section 19. Computer Operating Properly (COP)

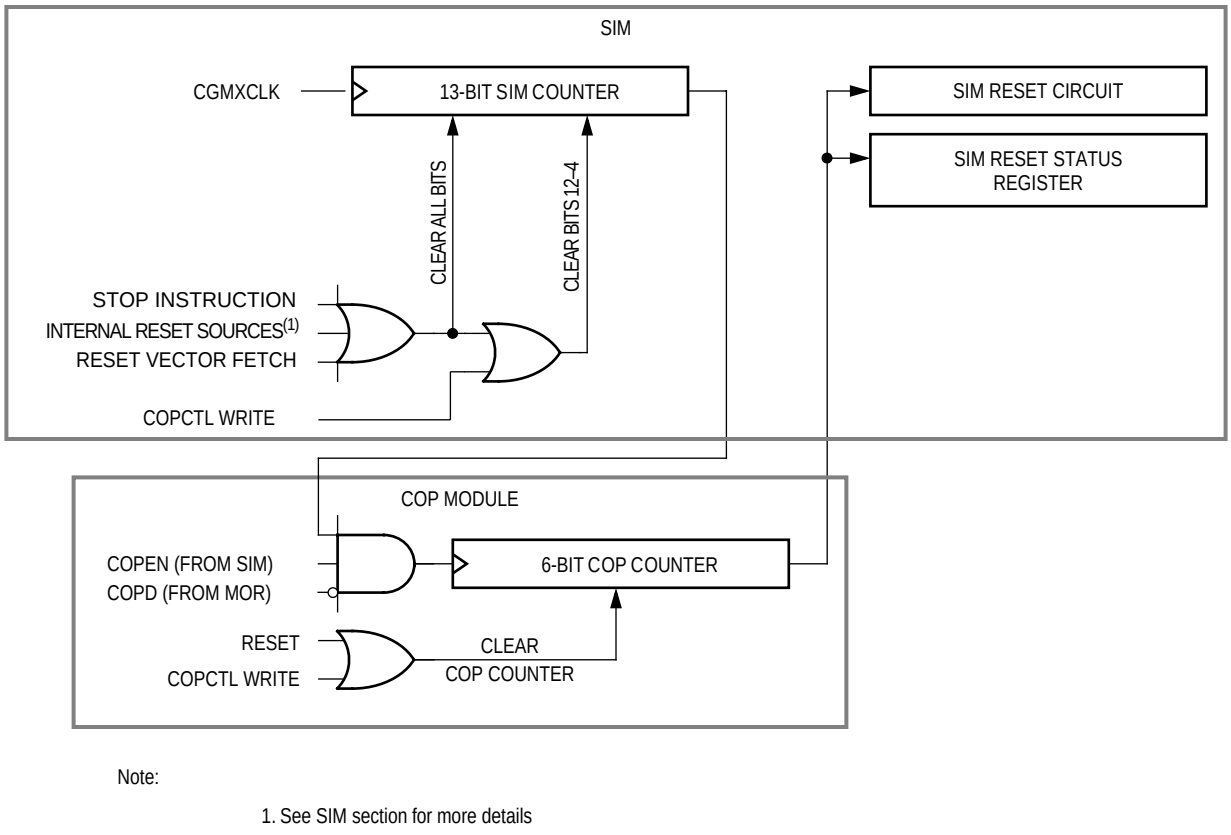
19.1 Contents

19.2 Introduction

This section describes the computer operating properly module (COP, Version B), 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.

19.3 Functional Description

Figure 19-1 shows the structure of the COP module.



Note:

1. See SIM section for more details

Figure 19-1. COP Block Diagram

The COP counter is a free-running 6-bit counter preceded by the 12-bit system integration module (SIM) counter. If not cleared by software, the COP counter overflows and generates an asynchronous reset after $2^{18} - 2^4$ CGMXCLK cycles. With a 32.768kHz crystal, the COP timeout period is eight second. Writing any value to location $FFFF before overflow occurs clears the COP counter and prevents reset.

A COP reset pulls the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles. The COP bit in the SIM reset status register (SRSR) will also be set (see **Section 7. System Integration Module (SIM)**). Clear the COP immediately before entering or after exiting stop mode to assure a full COP timeout

period after entering or exiting stop mode. A CPU interrupt routine can be used to clear 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.*

19.4 I/O Signals

The following paragraphs describe the signals shown in **Figure 19-1**.

19.4.1 CGMXCLK

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

19.4.2 STOP Instruction

The STOP instruction clears the SIM counter.

19.4.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see **19.5 COP Control Register (COPCTL)**) clears the COP counter and clears bits 12 through 4 of the SIM counter. Reading the COP control register returns the reset vector.

19.4.4 Power-On Reset

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

19.4.5 Internal Reset

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

19.4.6 Reset Vector Fetch

A reset vector fetch occurs when the vector address appears on the data bus. A reset vector fetch clears the SIM counter.

19.4.7 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the mask option register (MOR). (See **Section 5. Configuration Register (CONFIG)**.)

19.5 COP Control Register (COPCTL)

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.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0	Addr.
COP Control Register (COPCT:)	R:	Low byte of reset vector								$FFFF
	W:	Clear COP counter								
Reset:		Unaffected by reset								

Figure 19-2. COP Control Register (COPCTL)

19.6 Interrupts

The COP does not generate CPU interrupt requests or DMA service requests.

19.7 Monitor Mode

The COP is disabled in monitor mode when V_{DD} + V_{HI} is present on the $\overline{IRQ1}/V_{PP}$ pin or on the OSC1 pin.

19.8 Low-Power Modes

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

19.8.1 Wait Mode

The COP continues to operate during wait mode. To prevent a COP reset during wait mode, periodically clear the COP counter in a CPU interrupt routine or a DMA service routine.

19.8.2 Stop Mode

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

The STOP bit in the configuration register (CONFIG) or the mask option register (MOR) enables the STOP instruction. To prevent inadvertently turning off the COP with a STOP instruction, disable the STOP instruction by programming the STOP bit to logic zero.

19.9 COP Module During Break Interrupts

The COP is disabled during a break interrupt when $V_{DD} + V_{HI}$ is present on the OSC1 pin.

Section 20. Low Voltage Inhibit (LVI)

20.1 Contents

20.2 Introduction

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

20.3 Features

Features of the LVI module include the following:

- Selectable LVI trip voltage
- Programmable LVI Reset
- Programmable LVI Interrupt
- Programmable Power Consumption

20.4 Functional Description

Figure 20-1 shows the structure of the LVI module. The LVI module contains a bandgap reference circuit and comparator. The LVI enable bit (LVIEN) enables the LVI to monitor V_{DD} voltage. The LVI trip voltage selection bits (LVIT1, LVIT0) determines at which V_{DD} level the LVI module should take actions.

The LVI module has three output signals:

- Interrupt — once the LVI module is enabled, the interrupt will be enabled automatically. When V_{DD} drops below LVI_{TRIP} an interrupt will be generated to inform the CPU.
- Reset — an reset signal will be generated to reset the CPU when V_{DD} drops to below V_{LVR}. A user can enable such reset operation by setting LVREN to logic zero and enabling the LVI module.
- LVIOUT — this is the output status of the voltage comparator. A program can monitor the status by continuously polling this bit in LVICR.

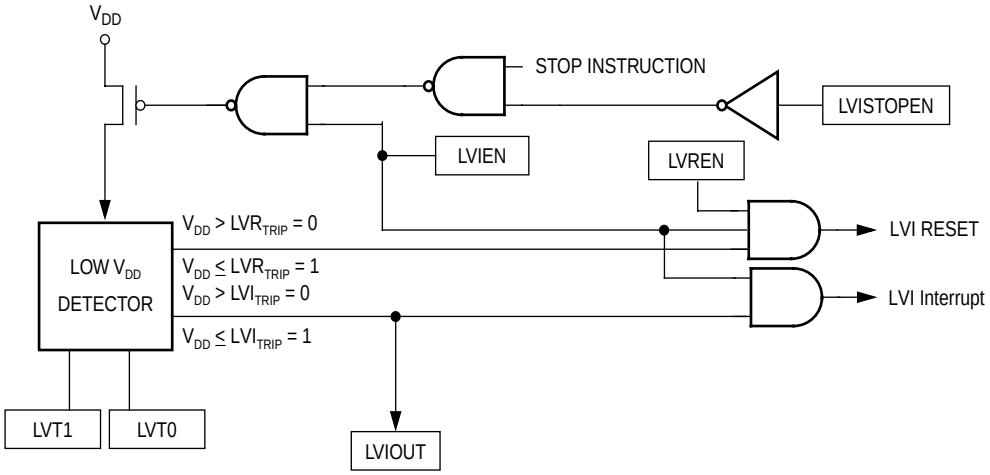


Figure 20-1. LVI Module Block Diagram

20.5 LVI Control Register (LVICR)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0	Addr.
LVI Control Register (LVICR)	R:	LVIOUT	LVIF	0	LVREN	LVIT1	LVIT0	$\overline{\text{LVISTO}}$ $\overline{\text{PEN}}$	LVIEN	$003C
	W:			LVIACK						
Reset:		0	0	0	0	0	0	0	0	

Figure 20-2. LVI Control Register (LVICR)

LVIEN — LVI Enable

1 = LVI enabled, Interrupt enabled
0 = LVI disabled, Interrupt enabled

$\overline{\text{LVISTOPEN}}$ — Enable LVI in STOP Mode

1 = LVI not operating in STOP mode
0 = LVI continues to operate in STOP mode when LVI module enabled

LVIT1, LVIT0 — LVI Trip Voltage Selection

These two bits determine at which level of V_{DD} the LVI module will come into action.

LIVT1	LVIT0	Trip Voltage
0	0	V_{LVI1}
0	1	V_{LVI2}
1	0	V_{LVI3}
1	1	V_{LVI4}

LVREN — Low Voltage Reset Enable

When this bit is enabled, the LVI module will generate a reset to the CPU when V_{DD} drops below V_{LVR}. This bit will be effective only when the LVI module is enabled by setting LVIEN to logic one.

1 = Low voltage reset enabled
0 = Low voltage reset disabled

LVIF — Interrupt Flag Status

This read only bit indicates the status of the LVI interrupt signal.
1 = LVI interrupt pending
0 = No LVI interrupt pending

LVIACK — Interrupt Acknowledge Bit

The LVI interrupt flag can be cleared by writing a logic one into this bit.

LVIOUT — LVI Output Bit

This read-only flag becomes set when the V_{DD} voltage falls below the V_{LVR} trip voltage (see **Table 20-1**). Reset clears the LVIOUT bit.

Table 20-1. LVIOUT Bit Indication

V_{DD}	LVIOUT
$V_{DD} > LVI_{Trip}$	0
$V_{DD} < LVI_{Trip}$	1
$LVI_{Trip} < V_{DD} < LVI_{Trip} + H_{LVI}$	Previous Value

20.6 Low-Power Modes

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

20.6.1 Wait Mode

With the LVIEN bit is set to logic one, the LVI module will continue to operate in WAIT mode.

To minimize the current consumption, the LVI module should be disabled by writing a logic zero to LVIEN when LVI function is not required in an application.

20.6.2 Stop Mode

With the LVIEN bit is set to logic one and $\overline{\text{LIVSTOPEN}}$ set to logic zero, the LVI module will continue to operate in STOP mode.

If LVI function is not required during STOP mode, the $\overline{\text{LIVSTOPEN}}$ should be set to logic one. To minimize the overall system current consumption, the LVI module should be disabled when LVI function is not required in an application.

Section 21. Break Module (BREAK)

21.1 Contents

21.2 Introduction

This section describes the break module (Break, Version B). The break module can generate a break interrupt that stops normal program flow at a defined address to enter a background program.

21.3 Features

Features of the break module include the following:

- Accessible I/O Registers during the Break Interrupt
- CPU-Generated Break Interrupts
- Software-Generated Break Interrupts
- COP Disabling during Break Interrupts

21.4 Functional Description

When the internal address bus matches the value written in the break address registers, the break module issues a breakpoint signal ($\overline{\text{BKPT}}$) to the SIM. The SIM then causes the CPU to load 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).

The following events can cause a break interrupt to occur:

- A CPU-generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a logic one 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 21-1** shows the structure of the break module.

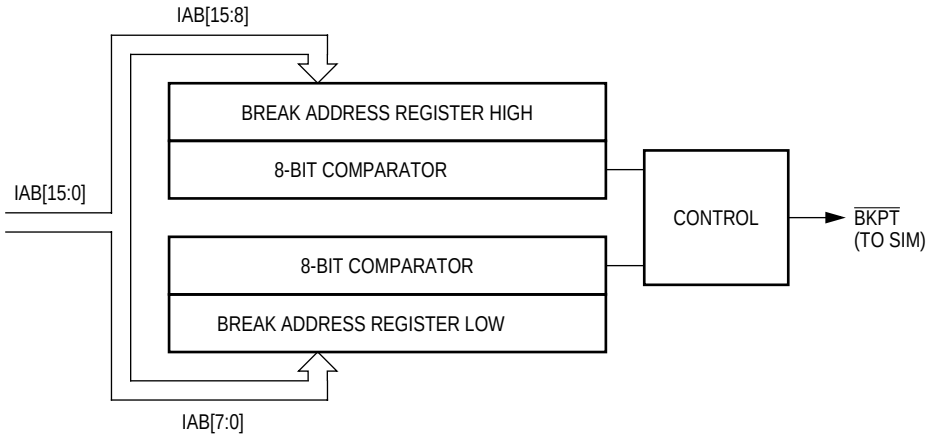


Figure 21-1. Break Module Block Diagram

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Break Address Register High (BRKH)	Bit 15	14	13	12	11	10	9	Bit 8	$FE0C
Break Address Register Low (BRKL)	Bit 7	6	5	4	3	2	1	Bit 0	$FE0D
Break Status/Control Register (BRKSCR)	BRKE	BRKA							$FE0E

= Unimplemented

Table 21-1. Break I/O Register Summary

21.4.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether or not module status bits can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **7.8.3 SIM Break Flag Control Register (SBFCR)** and **7.6.3 Break Interrupts** subsection for each module.)

21.4.2 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.

21.4.3 TIM During Break Interrupts

A break interrupt stops the timer counter.

21.4.4 COP During Break Interrupts

The COP is disabled during a break interrupt when V_{DD} + V_{HI} is present on the OSC1 pin.

21.5 Break Module Registers

Three registers control and monitor operation of the break module:

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

21.5.1 Break Status and Control Register (BRKSCR)

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

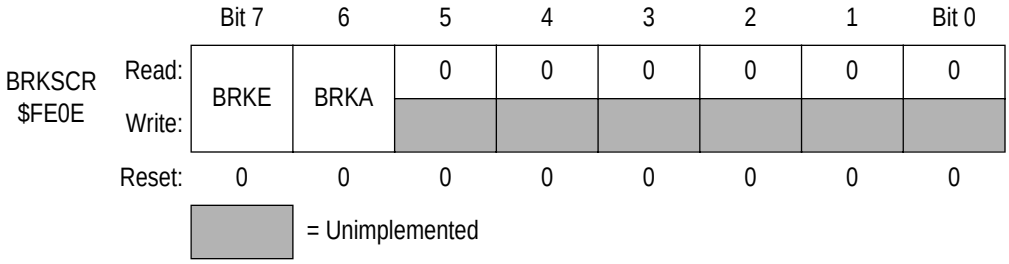


Figure 21-2. Break Status and Control Register (BRKSCR)

BRKE — Break Enable Bit

This read/write bit enables breaks on break address register matches. Clear BRKE by writing a logic zero 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 one to BRKA generates a break interrupt. Clear BRKA by writing a logic zero to it before exiting the break routine. Reset clears the BRKA bit.

1 = Break address match
0 = No break address match

21.5.2 Break Address Registers (BRKH and BRKL)

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

		Bit 7	6	5	4	3	2	1	Bit 0
BRKH $FE0C	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	0	0	0	0	0	0	0	0
BRKL $FE0D	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	0	0	0	0	0	0	0	0

Figure 21-3. Break Address Registers (BRKH and BRKL)

21.6 Low-Power Modes

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

21.6.1 Wait Mode

If enabled, the break module is active in wait mode. In the break routine, the user can subtract one from the return address on the stack if SBSW is set. (see Section 5.6) Clear the SBSW bit by writing logic zero to it.

21.6.2 Stop Mode

A break interrupt causes exit from stop mode and sets the SBSW bit in the SIM break status register. (See **7.8 SIM Registers**.)

Section 22. Electrical Specifications

22.1 Contents

22.2 Introduction

This section contains electrical and timing specifications.

22.3 Absolute Maximum Ratings

Maximum ratings are the extreme limits to which the MCU can be exposed without permanently damaging it.

NOTE: *This device is not guaranteed to operate properly at the maximum ratings. Refer to for guaranteed operating conditions*

Table 22-1. Absolute Maximum Ratings(1)

Characteristic	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
Programming Voltage	V_{PP}	V_{SS}–0.3 to 14.0	V
Maximum Current Per Pin Excluding V_{DD} and V_{SS}	I	±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

Notes:
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}).*

22.4 Functional Operating Range

Table 22-2. Operating Range

Characteristic	Symbol	Value	Unit
Operating Temperature Range	T_A	0 to +70	°C
Operating Voltage Range	V_{DD}	2.7 to 3.6	V

22.5 Thermal Characteristics

Table 22-3 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal Resistance 44-pin QFP 28-pin SOIC	θ_{JA}	 95 70	 °C/W °C/W
I/O Pin Power Dissipation	$P_{I/O}$	User Determined	W
Power Dissipation(1)	P_D	$P_D = (I_{DD} \times V_{DD}) + P_{I/O} = K/(T_J + 273\ °C)$	W
Constant(2)	K	$P_D \times (T_A + 273\ °C) + P_D^2 \times \theta_{JA}$	W/°C
Average Junction Temperature	T_J	$T_A + (P_D \times \theta_{JA})$	°C
Maximum Junction Temperature	T_{JM}	100	°C

Notes:
1. Power dissipation is a function of temperature.
2. K is a constant unique to the device. K can be determined for 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.

22.6 DC Electrical Characteristics

Table 22-4. DC Electrical Characteristics[1]

Characteristic	Symbol	Min	Typ[2]	Max	Unit
Output High Voltage (I_{LOAD} = –2.0 mA) all ports	V_{OH}	V_{DD} – 0.8	—	—	V
Output Low Voltage (I_{LOAD} = 1.6 mA) all ports	V_{OL}	—	—	0.4	V
Input High Voltage All ports, IRQs, RESET, OSC1	V_{IH}	$0.7 \times V_{DD}$	—	V_{DD}	V
Input Low Voltage All ports, IRQs, RESET, OSC1	V_{IL}	V_{SS}	—	$0.3 \times V_{DD}$	V
V_{DD} Supply Current Run Wait Stop	I_{DD}	 — — —	 3.5 0.7 4	 4 1 10	 mA mA µA
Low Voltage Interrupt	V_{LVI1} V_{LVI2} V_{LVI3} V_{LVI4}		2.8 2.5 2.3 2.2		V V V V
Low Voltage Interrupt Hysteresis	H_{LVI}	60	80	100	mV
Low Voltage Reset	V_{LVR}		2.45		V
Low Voltage Interrupt Hysteresis	H_{LVR}	60	80	100	mV
POR ReArm Voltage[3]	V_{POR}	0	—	200	mV
POR Rise Time Ramp Rate[4]	R_{POR}	0.02	—	—	V/ms
Capacitance Ports (as Input or Output)	C_{OUT} C_{IN}	— —	— —	12 8	pF pF

Notes:
1. V_{DD} = 2.7 to 3.6Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted.
2. Typical values reflect average measurements at midpoint of voltage range, 25°C only.
3. Maximum is highest voltage that POR is guaranteed.
4. If minimum V_{DD} is not reached before the internal POR reset is released, RST must be driven low externally until minimum V_{DD} is reached.

22.7 Control Timing

Table 22-5. Control Timing(1)

Characteristic	Symbol	Min	Typ	Max	Unit
Frequency of External Crystal(2)	f_{OSC}		32.768		KHz
Internal Operating Frequency	f_{OP}	—	2.49		MHz
$\overline{\text{RESET}}$ input Pulse Width Low (3)	t_{IRL}	125		—	ns
$\overline{\text{IRQ}}$ Interrupt Pulse Width Low (4) (Edge-Triggered)	t_{ILIH}	125		—	ns

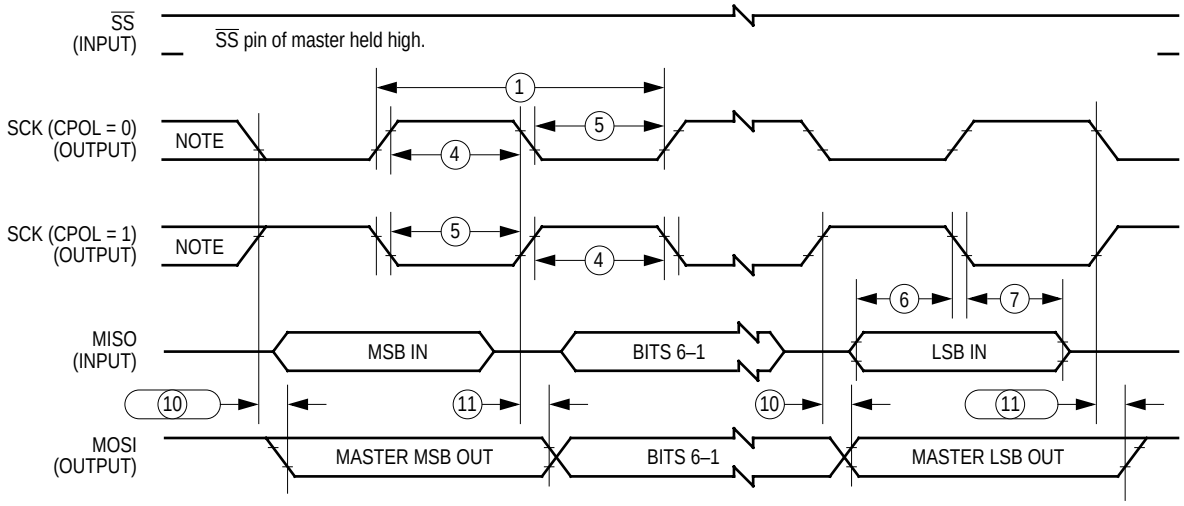
Notes:
1. V_{DD} = 2.7 to 3.6Vdc, V_{SS}=0 Vdc; timing shown with respect to 20% V_{DD} and 70% V_{DD} unless noted.
2. See **Table 22-8 . CGM Component Specifications** and **Table 22-9 . CGM Operating Conditions** for more information
3. Minimum pulse width reset is guaranteed to be recognized; it is possible for a smaller pulse width to cause a reset.
4. Minimum pulse width is for guaranteed interrupt; it is possible for a smaller pulse width to be recognized

22.8 Serial Peripheral Interface Characteristics

Table 22-6. Serial Peripheral Interface (SPI) Timing[1]

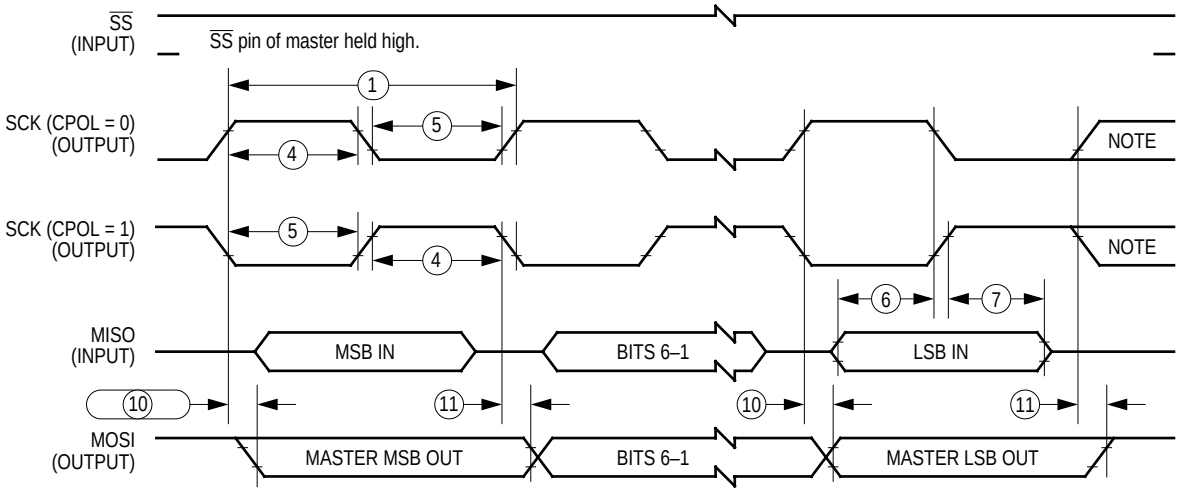
Diagram Number[2]	Characteristic[3]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	 $f_{OP(M)}$ $f_{OP(S)}$	 $f_{OP}/128$ DC	 $f_{OP}/2$ f_{OP}	 MHz MHz
1	Cycle time Master Slave	 $t_{CYC(M)}$ $t_{CYC(S)}$	 2 1	 128 —	 t_{cyc} t_{cyc}
2	Enable lead time	$t_{Lead(s)}$	1	—	t_{cyc}
3	Enable lag time	$t_{Lag(s)}$	1	—	t_{cyc}
4	Clock (SPSCK) high time Master Slave	 $t_{SCKH(M)}$ $t_{SCKH(S)}$	 t_{cyc} –35 1/2 t_{cyc} –35	 64 t_{cyc} —	 ns ns
5	Clock (SPSCK) low time Master Slave	 $t_{SCKL(M)}$ $t_{SCKL(S)}$	 t_{cyc} –35 1/2 t_{cyc} –35	 ± 64 t_{cyc} —	 ns ns
6	Data setup time (inputs) Master Slave	 $t_{SU(M)}$ $t_{SU(S)}$	 40 40	 — —	 ns ns
7	Data hold time (inputs) Master Slave	 $t_{H(M)}$ $t_{H(S)}$	 40 40	 — —	 ns ns
8	Access time, slave[4] CPHA = 0 CPHA = 1	 $t_{A(CP0)}$ $t_{A(CP1)}$	 0 0	 50 50	 ns ns
9	Disable time, slave[5]	$t_{DIS(S)}$	—	50	ns
10	Data valid time, after enable edge Master Slave[6]	 $t_{V(M)}$ $t_{V(S)}$	 — —	 60 60	 ns ns
11	Data hold time, outputs, after enable edge Master Slave	 $t_{HO(M)}$ $t_{HO(S)}$	 0 0	 — —	 ns ns

Notes:
1. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; assumes 100pF load on all SPI pins
2. Numbers refer to dimensions in **Figure 22-1** and **Figure 22-2.**
3. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; 100 pF load on all SPI pins.
4. Time to data active from high-impedance state
5. Hold time to high-impedance state
6. With 100 pF on all SPI pins



NOTE: This first clock edge is generated internally, but is not seen at the SCK pin.

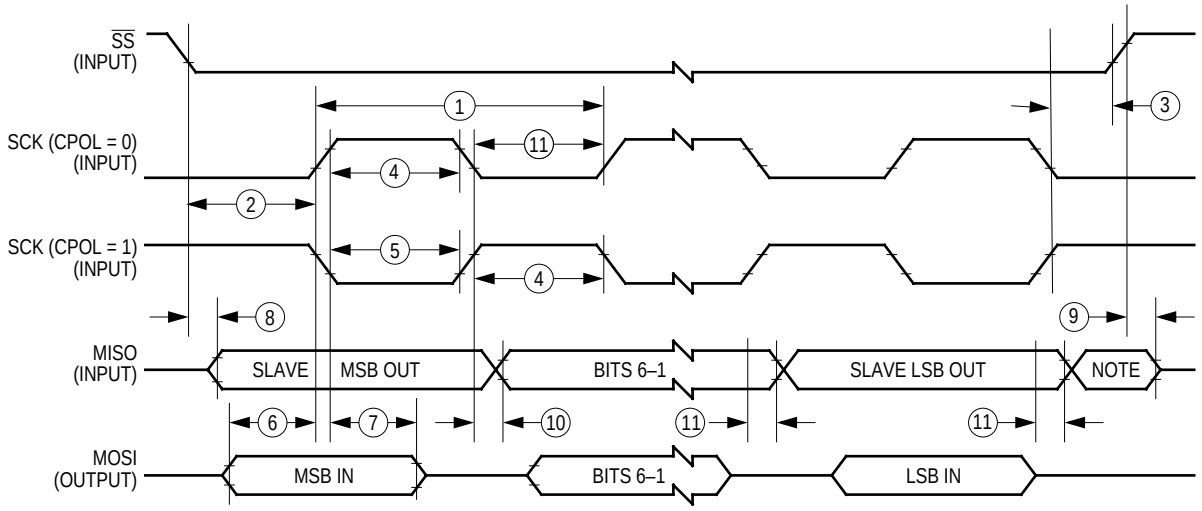
(a) SPI Master Timing (CPHA = 0)



NOTE: This last clock edge is generated internally, but is not seen at the SCK pin.

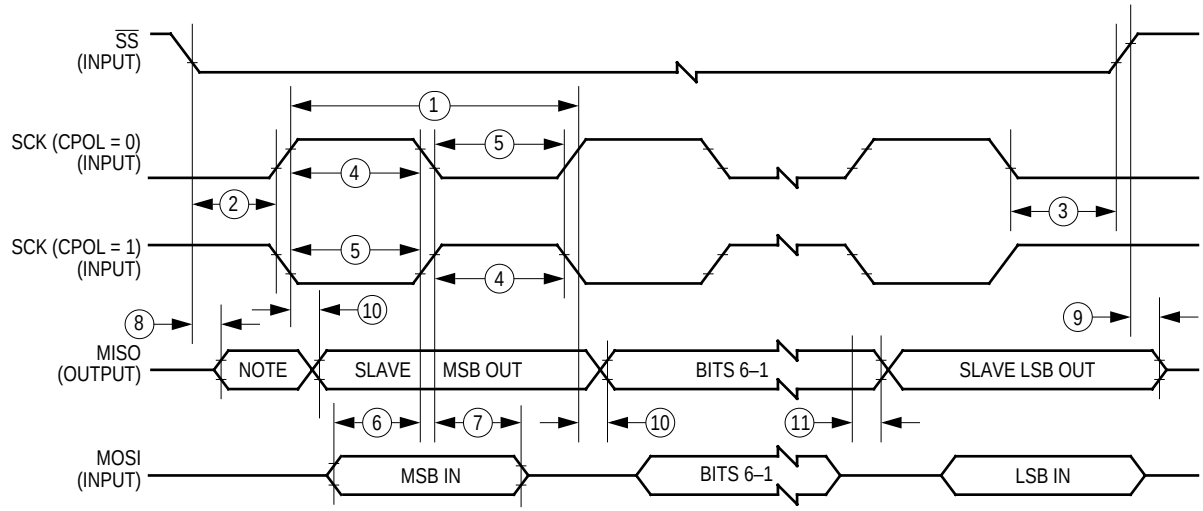
(b) SPI Master Timing (CPHA = 1)

Figure 22-1. 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 22-2. SPI Slave Timing

22.9 TImer Interface Module Characteristics

Table 22-7. TIM Timing

Characteristic	Symbol	Min	Max	Unit
Input Capture Pulse Width	t_{TIH},t_{TIL}	125	—	ns
Input Clock Pulse Width	t_{TCH},t_{TCL}	$(1/f_{OP}) + 5$	—	ns

22.10 Clock Generation Module Electrical Characteristics

Table 22-8. CGM Component Specifications

Characteristic	Symbol	Min	Typ	Max	Notes
Crystal Load Capacitance	C_L	—	—	—	Consult Crystal Mfg. Data
Crystal Fixed Capacitance	C_1	—	$2\times C_L$	—	Consult Crystal Mfg. Data
Crystal Tuning Capacitance	C_2	—	$2\times C_L$	—	Consult Crystal Mfg. Data
Filter Capacitor	C_F	—	$C_{FACT}\times (V_{DDA}/f_{XCLK})$	—	
Bypass Capacitor	C_{BYP}	—	0.1µF	—	C_{BYP} must provide low AC impedance from f = $f_{XCLK}/100$ to $100\times f_{VCLK}$, so series resistance must be considered.
Current Reference Resistor	R_{REF}	—	10.0kΩ	—	R_{REF} must be ±5% tolerance or better.

Table 22-9. CGM Operating Conditions

Characteristic	Symbol	Min	Typ	Max	Notes
Crystal Reference Frequency	f_{XCLK}		32.768kHz		
Range Nominal Multiplier	f_{NOM}	—	32.768kHz	—	
VCO center-of-range Frequency	f_{VRS}	32.768kHz	—	20.0MHz	2.7 to 3.6V V_{DD} only
VCO Power-of-Two Range Multiplier	2^E	1	1	8	
VCO Prescale Multiplier	2^P	1	1	8	
Reference Divider Factor	R	1	1	15	
VCO Operating Frequency	f_{VCLK}	f_{VRSMIN}	—	f_{VRSMAX}	

Table 22-10. CGM Acquisition / Lock Time Specifications

Description	Sym-bol	Min	Typ	Max	Notes
Manual Mode Time to Stable	t_{ACQ}	—	$(8\times V_{DDA}) \div (f_{X\ CLK}\times K_{ACQ})$	—	If C_F chosen correctly.
Manual Stable to Lock Time	t_{AL}	—	$(4\times V_{DDA}) \div (f_{X\ CLK}\times K_{TRK})$	—	If C_F chosen correctly.
Manual Acquisition Time	t_{LOCK}	—	$t_{ACQ} + t_{AL}$	—	
Tracking Mode Entry Frequency Tolerance	Δ_{TRK}	0	—	±3.6%	
Acquisition Mode Entry Frequency Tolerance	Δ_{ACQ}	±6.3%	—	±7.2%	
LOCK Entry Freq. Tolerance	Δ_{LOCK}	0	—	±0.9%	
LOCK Exit Freq. Tolerance	Δ_{UNL}	±0.9%		±1.8%	
Reference cycles per Acquisition Mode Measurement	n_{ACQ}	—	32	—	
Reference cycles per Tracking Mode Measurement	n_{TRK}	—	128	—	
Automatic Mode Time to Stable	t_{ACQ}	n_{ACQ}/f_{XCLK}	$(8\times V_{DDA}) \div (f_{X\ CLK}\times K_{ACQ})$	—	If C_F chosen correctly.
Automatic Stable to Lock Time	t_{AL}	n_{TRK}/f_{XCLK}	$(4\times V_{DDA}) \div (f_{X\ CLK}\times K_{TRK})$	—	If C_F chosen correctly.
Automatic Lock Time	t_{LOCK}	–	$t_{ACQ} + t_{AL}$	—	
PLL Jitter (deviation of average bus frequency over 2 ms)	f_J	0	—	$\pm(f_{CRYS}) \times(0.025\%) \times((2^P N/R)/4)$	N = VCO freq. mult. (GBNT)

22.11 Memory Characteristics

Table 22-11. Memory Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
FLASH Data Retention Time	t_{FDR}	—	10	—	yrs
FLASH Programming Time	t_{PROG}	—	30	—	µs/byte
FLASH Mass Erase Time	t_{ERASE}	—	4	—	ms
FLASH High Voltage Hold Time	t_{NVH}	—	5	—	µs
FLASH High Voltage Hold Time (Mass Erase)	t_{NVHL}	—	100	—	µs
RAM Data Retention Voltage	V_{RDR}	2.0	—	—	V

22.12 DTMF/Melody Generator Characteristics

Table 22-12. DTMF/Melody Generator Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
Tone Output Level Low Group (Row) High Group (Column)		 0.136 0.174	 0.160 0.205	 0.184 0.236	V_{rms}
Frequency Deviation (DTMF)		–0.5		+0.5	%
Frequency Deviation (Melody)		–0.8		+0.8	%
Tone Output DC Level		0.45	0.5	0.55	V
High Group Pre-emphasis		1	2.15	3	dB
Total Harmonic Distortion			–25		dB
TNX Output Level (Square Wave)			V_{DD}		V_{p-p}

22.13 Ring Detector Characteristics

Table 22-13. Ring Detector Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
Ring Detect Input Threshold Voltage	V_{RD_TH}		1.2		V
Ring Detect Input Hysteresis	V_{RD_HY}		200		mV
RT Input Threshold Voltage			1.2		V
RT Input Hysteresis			200		mV
RT Discharge Resistance			1000		Ω

Electrical Specifications

Section 23. Mechanical Specifications

23.1 Contents

23.2 Introduction

This section gives the dimensions for:

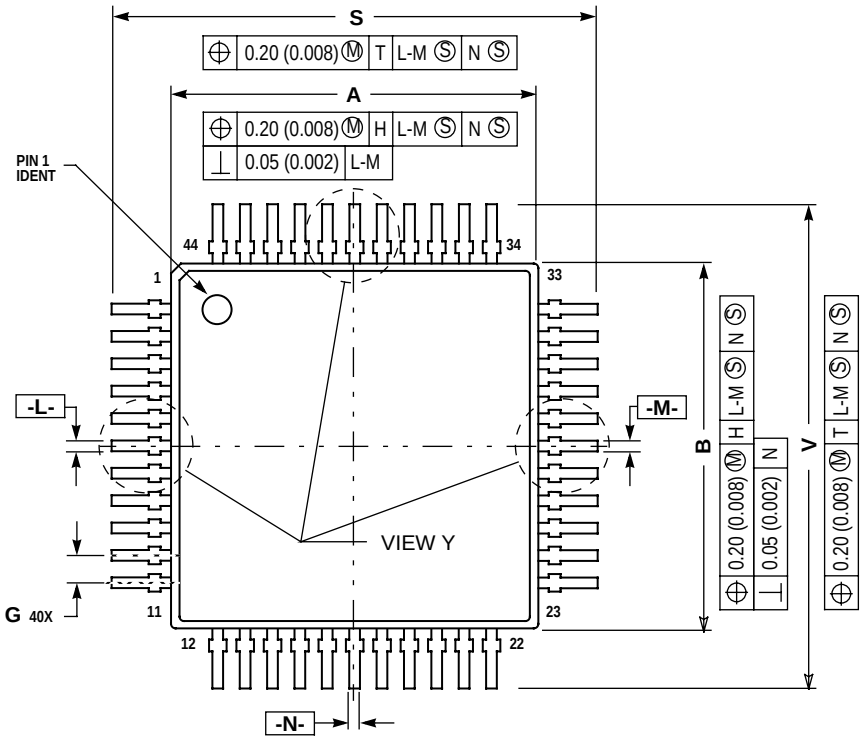
- 44-pin plastic quad flat pack (case 824E-02)
- 28-pin small outline integrated circuit package (case 751F-04)

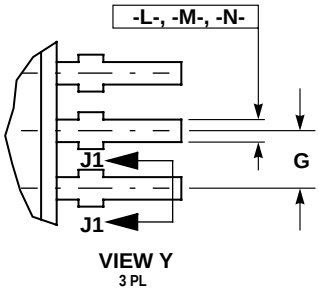
The following figures show the latest package drawings at the time of this publication. To make sure that you have the latest package specifications, contact one of the following:

- Local Motorola Sales Office
- Motorola Mfax
  - Phone 602-244-6609
  - EMAIL rmfax0@email.sps.mot.com
- Worldwide Web (wwweb) at http://design-net.com

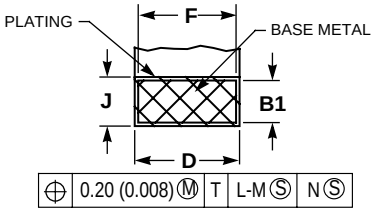
Follow Mfax or Worldwide Web on-line instructions to retrieve the current mechanical specifications.

23.3 44-Pin Plastic Quad Flat Pack (QFP)

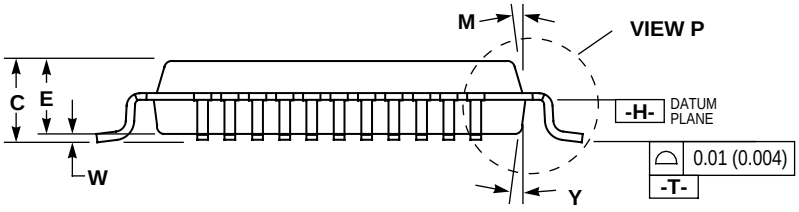


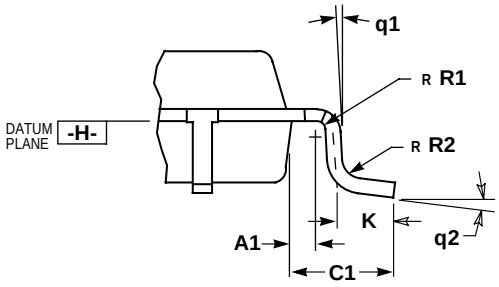


VIEW Y
3 PL



SECTION J1-J1
44 PL





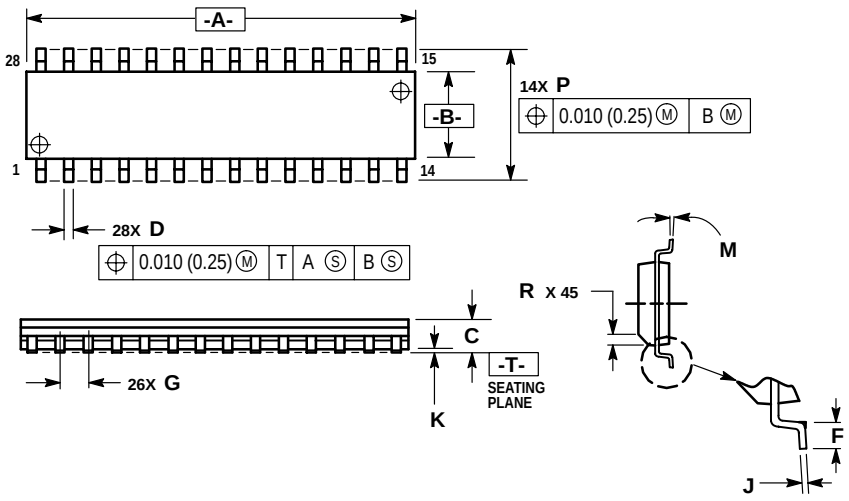
VIEW P

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 -L-, -M- AND -N- TO BE DETERMINED AT DATUM PLANE -H-.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE -T-.
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. DAMBAR PROTRUSION SHALL NOT CAUSE THE D DIMENSION TO EXCEED 0.530 (0.021).

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	9.90	10.10	0.390	0.398
B	9.90	10.10	0.390	0.398
C	2.00	2.21	0.079	0.087
D	0.30	0.45	0.0118	0.0177
E	2.00	2.10	0.079	0.083
F	0.30	0.40	0.012	0.016
G	0.80 BSC		0.031 BSC	
J	0.13	0.23	0.005	0.009
K	0.65	0.95	0.026	0.037
M	5°	10°	5°	10°
S	12.95	13.45	0.510	0.530
V	12.95	13.45	0.510	0.530
W	0.000	0.210	0.000	0.008
Y	5°	10°	5°	10°
A1	0.450 REF		0.018 REF	
B1	0.130	0.170	0.005	0.007
C1	1.600 REF		0.063 REF	
R1	0.130	0.300	0.005	0.012
R2	0.130	0.300	0.005	0.012
q1	5°	10°	5°	10°
q2	0°	7°	0°	7°

23.4 28-Pin Small Outline Integrated Circuit (SOIC)



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DIMENSION A AND B DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 (0.006) PER SIDE.
5. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.13 (0.005) TOTAL IN EXCESS OF D DIMENSION AT MAXIMUM MATERIAL CONDITION.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	17.80	18.05	0.701	0.711
B	7.40	7.60	0.292	0.299
C	2.35	2.65	0.093	0.104
D	0.35	0.49	0.014	0.019
F	0.41	0.90	0.016	0.035
G	1.27 BSC		0.050 BSC	
J	0.23	0.32	0.009	0.013
K	0.13	0.29	0.005	0.011
M	0°	8°	0°	8°
P	10.01	10.55	0.395	0.415
R	0.25	0.75	0.010	0.029

Mechanical Specifications

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1-800-441-2447 or 1-303-675-2140

JAPAN: Nippon Motorola Ltd. SPD, Strategic Planning Office 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan. 03-5487-8488

Mfax™, Motorola Fax Back System: RMFAX0@email.sps.mot.com; http://sps.motorola.com/mfax/; TOUCHTONE 1-602-244-6609; US and Canada ONLY 1-800-774-1848

HOME PAGE: http://motorola.com/sps/