Order this document by
MC68HC08PT48/D
Rev. 2.0

HC08

68HC08PT48
68HC908PT48

Advance Information

This document contains information on a new product.
Specifications and information herein are subject to change without notice.

freescale™
semiconductor

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 Map

Section 3. Central Processing Unit (CPU)

Section 4. Clock Generator Module (CGMB)

Section 5. Computer Operating Properly (COP) Module

Section 6. Keyboard Interrupt (KBI) Module

Section 7. System Integration Module (SIM)

Section 8. Random-Access Memory (RAM)

Section 9. 2-Kbyte FLASH Memory

Section 10. 48-Kbyte FLASH Memory

Section 11. Serial Peripheral Interface (SPI) Module

Section 12. Serial Communications Interface (SCI) Module

Section 13. Analog-to-Digital Converter (ADC) Module

Section 14. Configuration Register (CONFIG)

Section 15. Timer Interface Module (TIM)

Section 16. Timebase Module (TIMTBX)

Section 17. Input/Output (I/O) Ports

Section 18. Monitor ROM (MON)

Section 19. Break Module

Section 20. External Interrupt Module (IRQ)

Section 21. Alert Output Generator (ALR)

Section 22. Electrical Specifications

Section 23. Mechanical Data

Section 24. Ordering Information

List of Figures

List of Figures

Figure	Title	Page

List of Figures

Figure	Title	Page

List of Figures

List of Tables

List of Tables

Section 1. General Description

1.1 Contents

1.2 Introduction

The MC68HC(9)08PT48 is a member of the low-cost, low-power, 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 MC68HC(9)08PT48 include:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- On-chip FLASH/ROM of 48 Kbytes
- On-chip FLASH of 2 Kbytes separate from program ROM/FLASH
- ROM data security[1] option (no security option for MC68HC908PT48)
- Configuration register (CONFIG)
- 2.5 Kbytes of on-chip MCU random-access memory (RAM)
- 56 general-purpose input/output (I/O) pins
- Programmable phase locked loop (PLL) for bus clock generation
- Serial peripheral interface module (SPI)
- Serial communications interface module (SCI)
- Timebase module (TBM) with software selection of crystal clock source
- Two external interrupt request pins
- ALERT generator module (ALR)

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

- 16-bit, 4-channel timer interface module (TIM)
- Computer operating properly (COP) reset
- 8-bit, 4-channel analog-to-digital converter (ADC)
- System protection features:
  - Illegal opcode detect reset
  - Illegal address detect reset
- Packaged in an 80-pin quad flat pack (LQFP)
- Low-power design (fully static with stop and wait modes)
- Master reset pin and power-on reset (POR)

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
- Fast 8 x 8 multiply instruction
- Fast 16 ÷ 8 divide instruction
- Binary coded decimal (BCD) instructions
- Optimization for controller applications
- High-level language (C language) support

1.4 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC(9)08PT48.

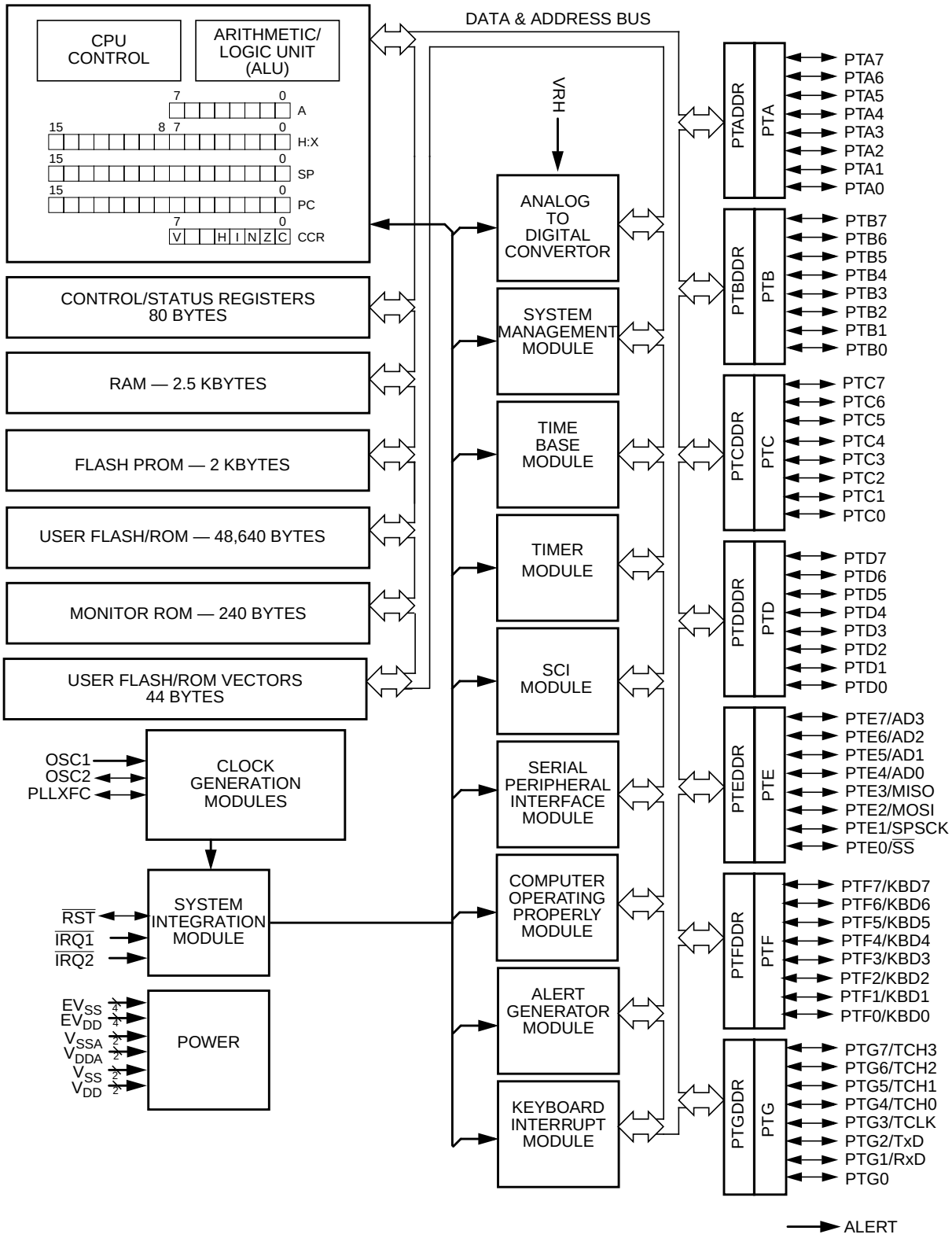


Figure 1-1. MC68HC(9)08PT48 Block Diagram

1.5 Pin Assignments

Pin assignments for the MC68HC(9)08PT48 are shown in **Figure 1-2**.

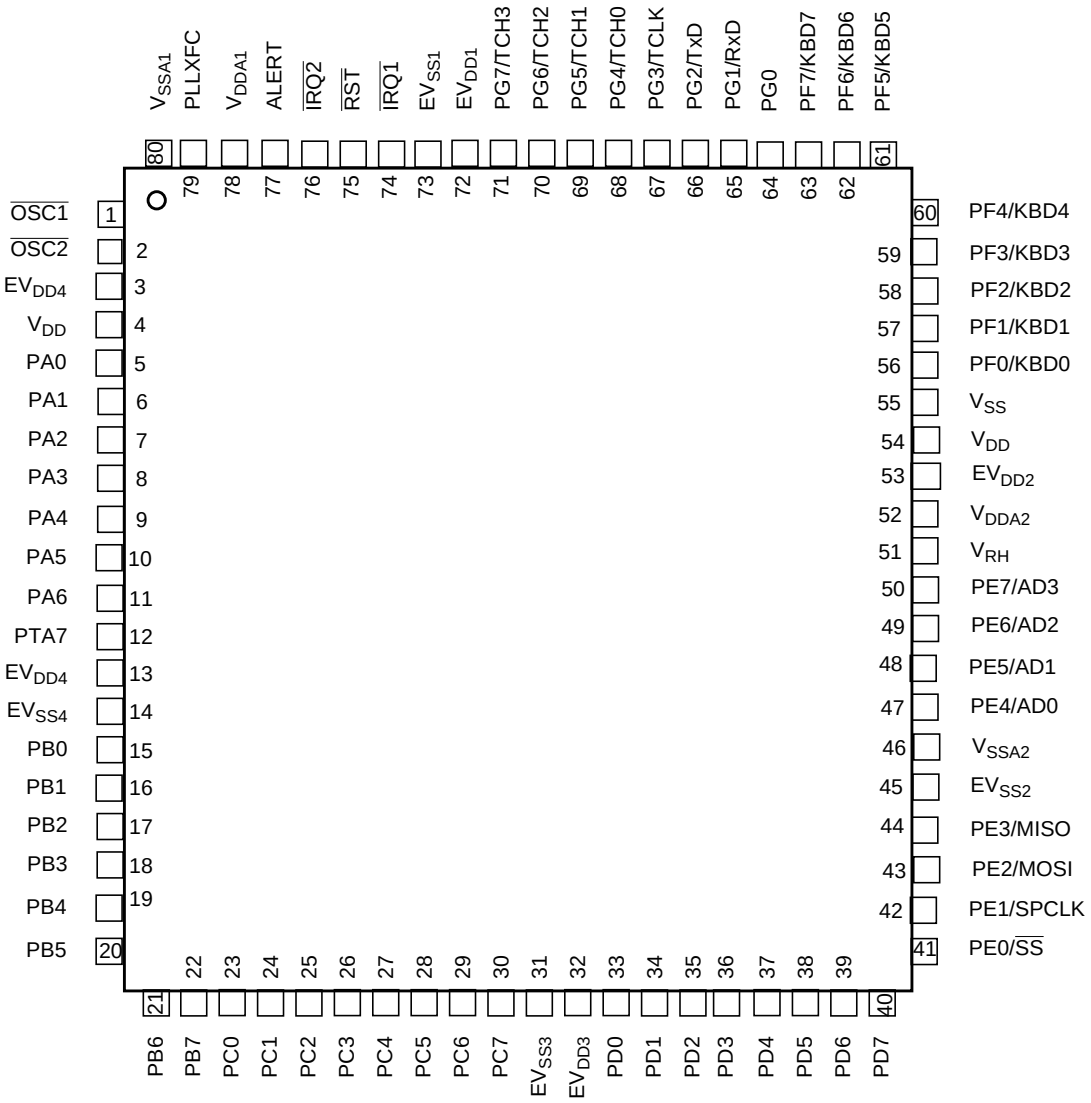


Figure 1-2. MC68HC(9)08PT48 Pinout (Top View)

1.6 Pin Functions

Descriptions of the pin functions are provided here.

1.6.1 Power Supply Pins (V_{DD}, V_{SS}, EV_{DD1-4}, EV_{SS1-4}, $V_{DDA1,2}$, $V_{SSA1,2}$)

V_{DD} and V_{SS} are power supply and ground pins for the digital sections of the MCU. The MCU operates from a single power supply ranging from 2 V ± 10% to 3 V ± 10%.

The EV_{DD} and EV_{SS} pins are power supply and ground pins for the I/O section of the MCU. V_{DDA1} and V_{SSA1} are used for the analog portion of the bus clock PLL to reduce noise injected to the clocks.

Very 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 good power supply bypassing at the MCU as shown in **Figure 1-3**. 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.

EV_{SS2} pin is also the ground return pin for the serial clock in the serial peripheral interface module (SPI). It enables the user to implement a coplanar transmission line for the SPI clock on printed circuit boards (PCBs) with no ground plane. V_{SS} can help reduce radiated radio frequency (RF) emissions by controlling trace impedance and minimizing radiating loop area.

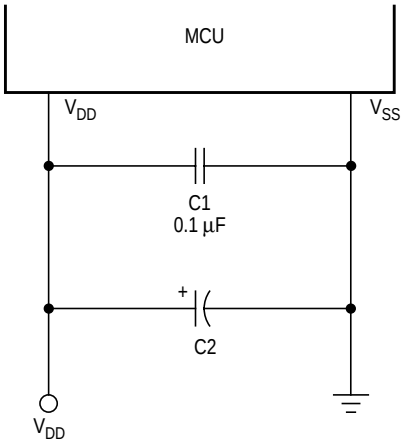


Figure 1-3. Power Supply Bypassing

NOTE: *Component values shown represent typical applications.*

1.6.2 Oscillator Pins ($\overline{\text{OSC1}}$ and $\overline{\text{OSC2}}$)

The $\overline{\text{OSC1}}$ and $\overline{\text{OSC2}}$ pins are the crystal connections for the on-chip oscillator. **Figure 1-4** shows a typical crystal oscillator circuit for a parallel resonant crystal. Follow the crystal supplier's recommendations, as the crystal parameters determine the external component values required to provide reliable start up and maximum stability.

NOTE: *The load capacitance values used in the oscillator circuit design should include all stray layout capacitances.*

To minimize output distortion and RF emissions, mount the crystal and capacitors as close as possible to the pins.

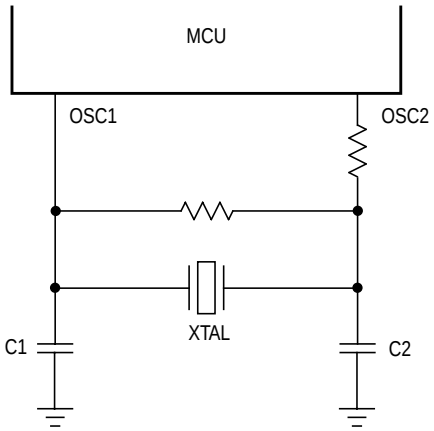


Figure 1-4. Crystal Connections

NOTE: *Follow the crystal manufacturer's recommendations for component sizes.*

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

A logic 0 on the $\overline{\text{RST}}$ pin forces the MCU to a known startup state. $\overline{\text{RST}}$ is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted. See **Section 7. System Integration Module (SIM)** for more information.

1.6.4 External Interrupt Pin ($\overline{\text{IRQ1}}$)

$\overline{\text{IRQ1}}$ is an asynchronous external interrupt pin. See **Section 20. External Interrupt Module (IRQ)** for more information.

1.6.5 External Interrupt Pin ($\overline{\text{IRQ2}}$)

$\overline{\text{IRQ2}}$ is an asynchronous external interrupt pin. See **Section 20. External Interrupt Module (IRQ)** for more information.

1.6.6 External Filter Capacitor Pins (PLLXFC)

PLLXFC is the external filter capacitor connections for the PLL. See **Section 17. Input/Output (I/O) Ports** for more information.

1.6.7 Port A I/O Pins (PTA7–PTA0)

PTA7–PTA0 are general-purpose bidirectional I/O port pins. See **Section 17. Input/Output (I/O) Ports** for more information.

1.6.8 Port B I/O Pins (PTB7–PTB0)

PTB7–PTB0 are general-purpose bidirectional I/O port pins.See **Section 17. Input/Output (I/O) Ports** for more information.

1.6.9 Port C I/O Pins (PTC7–PTC0)

PTC7–PTC0 are general-purpose bidirectional I/O port pins. See **Section 17. Input/Output (I/O) Ports** for more information.

1.6.10 Port D I/O Pins (PTD7–PTD0)

PTD7–PTD0 are general-purpose bidirectional I/O port pins. See **Section 17. Input/Output (I/O) Ports** for more information.

1.6.11 Port E I/O Pins (PTE7/AD3–PTE4/AD0 and PTE3/MISO–PTE0/$\overline{SS}$)

Port E is an 8-bit special function port that shares four of its pins with the ADC and the other four pins with SPI.

1.6.12 Port F I/O Pins (PTF7/KBD7–PTF0/KBD0)

Port F is an 8-bit special function port that shares with the keyboard interrupts.

1.6.13 Port G I/O Pins (PTG7/TCH3–PTG3/TCLK, PTG2/TxD, PTG1/RxD, and PTG0)

Port G is an 8-bit special function port that shares five of its pins with the timer and two of its pins with SCI.

1.6.14 Alert Generator Output (ALERT)

ALERT is the output from the ALR. See **21.1 Contents** for more information.

1.6.15 ADC Voltage Reference Pin (V_{RH})

V_{RH} is the power supply for setting the reference voltage V_{RH}. Connect the V_{RH} pin to a voltage potential ← V_{DDA2}, not less than 1.5 V. It supplies the resistor legs. Ideally, route this to its own pad. It can be routed to V_{DDA}.

General Description

Section 2. Memory Map

2.1 Contents

2.2 Introduction

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

- 48 Kbytes of low-voltage FLASH/ROM
- 2.5 Kbytes of RAM
- 2 Kbytes of low-voltage FLASH separate from ROM/emulation FLASH
- 44 bytes of user-defined vectors
- 240 bytes of monitor ROM

2.3 I/O Section

Addresses $0000–$004F, shown in **Figure 2-1**, contain most of the control, status, and data registers. Additional input/output (I/O) registers located in upper page memory are shown in **Figure 2-2**.

2.4 Random-Access Memory (RAM)

The 2.5 Kbyte addresses from $0050–$0A4F are RAM locations. The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in RAM, allowing all page zero locations to be used for I/O control and user data or code. Within page zero there are 176 bytes of RAM. When the stack pointer is moved from its reset location at $00FF, 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. For M6805 compatibility, the H register is not stacked. During a subroutine call, the CPU uses two bytes of the stack to store the return address. The stack pointer decrements during pushes and increments during pulls.

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

2.5 Memory Map

See **Figure 2-1**, **Figure 2-2**, **Figure 2-3**, and **Figure 2-4**.

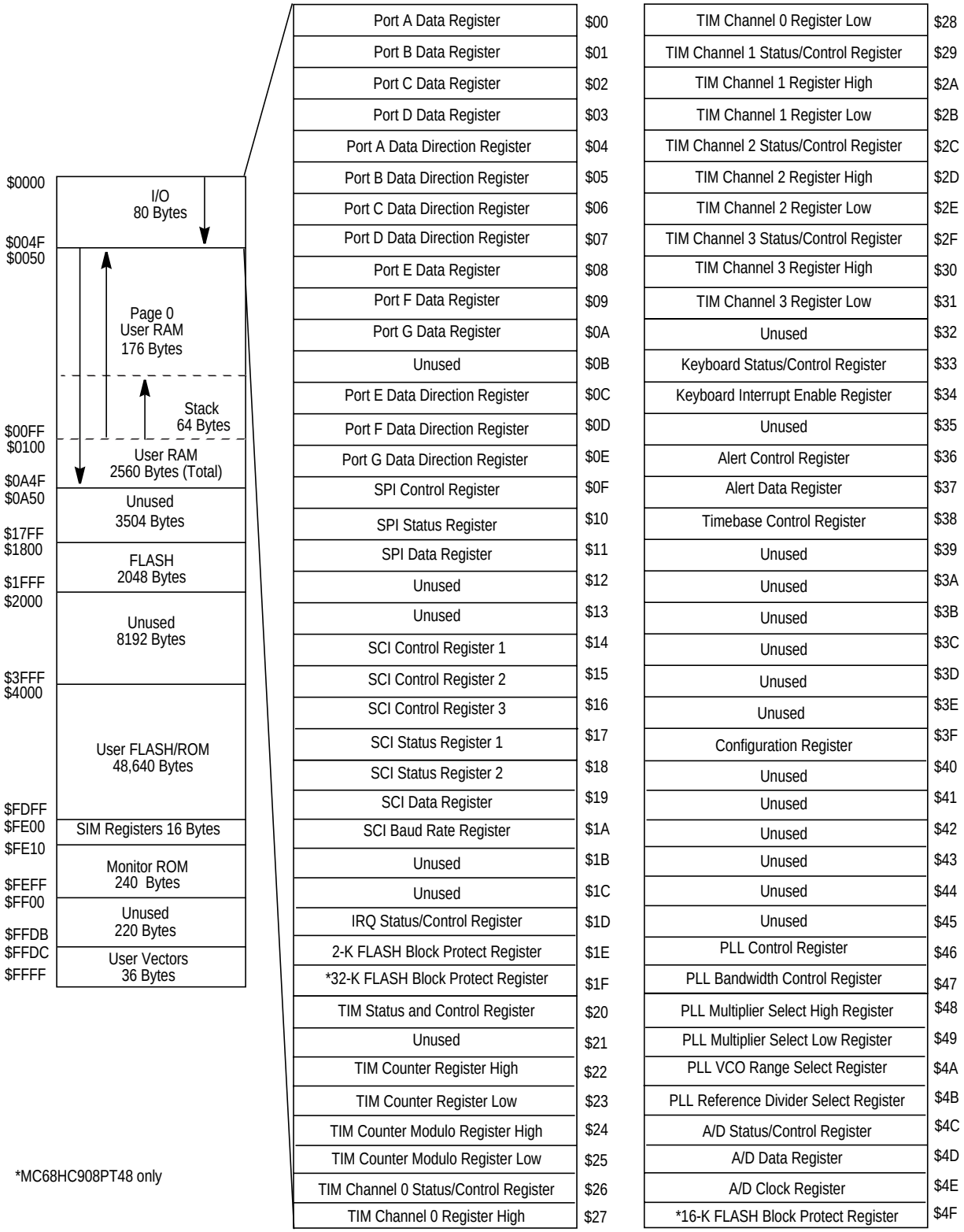


Figure 2-1. Memory Map

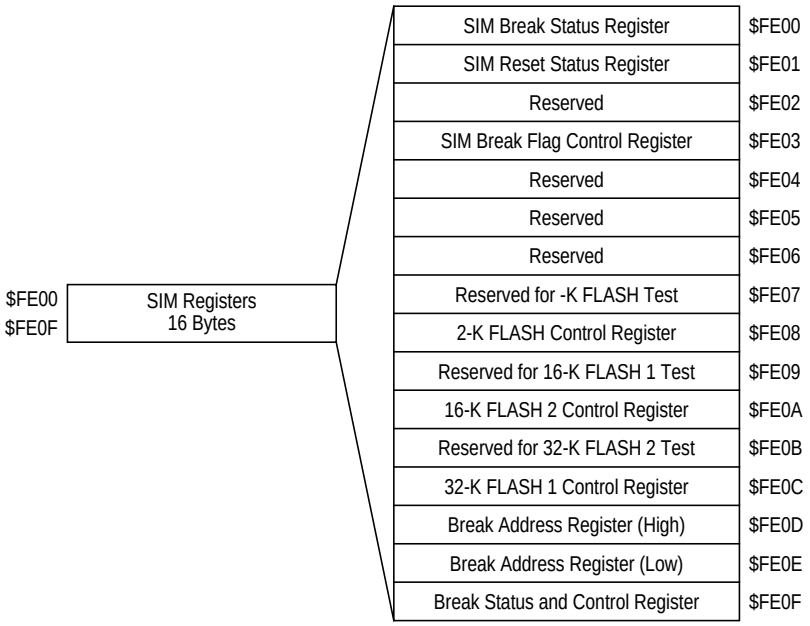


Figure 2-2. MC68HC(9)08PT48 Register Map

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read: Write:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Reset:	Unaffected by reset							

= Unimplemented U = Undetermined X = Indeterminate

Figure 2-3. Control, Status, and Data Registers (Sheet 1 of 10)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0004	Port A Data Direction Register (DDRA)	Read: Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Reset:	0	0	0	0	0	0	0	0
$0005	Port B Data Direction Register (DDRB)	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Reset:	0	0	0	0	0	0	0	0
$0006	Port C Data Direction Register (DDRC)	Read: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Reset:	0	0	0	0	0	0	0	0
$0007	Port D Data Direction Register (DDRD)	Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE)	Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Reset:	Unaffected by reset							
$0009	Port F Data Register (PTF)	Read: Write:	PTF7	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
		Reset:	Unaffected by reset							
$000A	Port G Data Register (PTG)	Read: Write:	PTG7	PTG6	PTG5	PTG4	PTG3	PTG2	PTG1	PTG0
		Reset:	Unaffected by reset							
$000B			Unimplemented							
$000C	Port E Data Direction Register (DDRE)	Read: Write:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		Reset:	0	0	0	0	0	0	0	0
$000D	Port F Data Direction Register (DDRF)	Read: Write:	DDRF7	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented U = Undetermined X = Indeterminate

Figure 2-3. Control, Status, and Data Registers (Sheet 2 of 10)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$000E	Port G Data Direction Register (DDRG)	Read: Write:	DDRG7	DDRG6	DDRG5	DDRG4	DDRG3	DDRG2	DDRG1	DDRG0
		Reset:	0	0	0	0	0	0	0	0
$000F	SPI Control Register (SPCR)	Read: Write:	SPRIE	DMAS	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Reset:	0	0	1	0	1	0	0	0
$0010	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
$0011	SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Indeterminate after reset							
$0012			Unimplemented							
$0013			Unimplemented							
$0014	SCI Control Register 1 (SCC1)	Read: Write:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Reset:	0	0	0	0	0	0	0	0
$0015	SCI Control Register 2 (SCC2)	Read: Write:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Reset:	0	0	0	0	0	0	0	0
$0016	SCI Control Register 3 (SCC3)	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
		Write:								
		Reset:	U	U	0	0	0	0	0	0
$0017	SCI Status Register 1 (SCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0

= Unimplemented U = Undetermined X = Indeterminate

Figure 2-3. Control, Status, and Data Registers (Sheet 3 of 10)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0018	SCI Status Register 2 (SCS2)	Read:							BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0019	SCI Data Register (SCDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$001A	SCI Baud Rate Register (SCBR)	Read:								
		Write:			SCP1	SCP0		SCR2	SCR1	SCR0
		Reset:	0	0	0	0	0	0	0	0
$001B			Unimplemented							
$001C			Unimplemented							
$001D	IRQ Status and Control Register (ISCR)	Read:	IRQF2	0	IMASK2	MODE2	IRQF1	0	IMASK1	MODE1
		Write:		ACK2				ACK1		
		Reset:	0	0	0	0	0	0	0	0
$001E	FLASH 3 Block Protect Register (FL3BPR)	Read:								
		Write:					BPR3	BPR2	BPR1	BPR0
		Reset:	X	X	X	X	1	1	1	1
$001F	FLASH 1 Block Protect Register (FL1BPR)	Read:								
		Write:					F1BPR3	F1BPR2	F1BPR1	F1BPR0
		Reset:	X	X	X	X	1	1	1	1
$0020	Timer Status and Control Register (TSC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0021			Unimplemented							
$0022	Timer Counter Register High (TCNTH)	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented U = Undetermined X = Indeterminate

Figure 2-3. Control, Status, and Data Registers (Sheet 4 of 10)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0023	Timer Counter Register Low (TCNTL)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0024	Timer Modulo Register High (TMODH)	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	Timer Modulo Register Low (TMODL)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0026	Timer Channel 0 Status and Control Register (TSC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0027	Timer Channel 0 Register High (TCH0H)	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0028	Timer Channel 0 Register Low (TCH0L)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0029	Timer Channel 1 Status and Control Register (TSC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$002A	Timer Channel 1 Register High (TCH1H)	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002B	Timer Channel 1 Register Low (TCH1L)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Unaffected by reset							
$002C	Timer Channel 2 Status and Control Register (TSC2)	Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented U = Undetermined X = Indeterminate

Figure 2-3. Control, Status, and Data Registers (Sheet 5 of 10)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002D	Timer Channel 2 Register High (TCH2H)	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	Unaffected by reset							
$002E	Timer Channel 2 Register Low (TCH2L)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	Unaffected by reset							
$002F	Timer Channel 3 Status and Control Register (TSC3)	Read: Write:	CH3F 0	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Reset:	0	0	0	0	0	0	0	0
$0030	Timer Channel 3 Register High (TCH3H)	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	Unaffected by reset							
$0031	Timer Channel 3 Register Low (TCH3L)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	Unaffected by reset							
$0032			Unimplemented							
$0033	Keyboard Status and Control Register (KBSCR)	Read: Write:	0	0	0	0	KEYF	0 ACKK	IMASKK	MODEK
		Reset:	0	0	0	0	0	0	0	0
$0034	Keyboard Interrupt Enable Register (KBIER)	Read: Write:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Reset:	0	0	0	0	0	0	0	0
$0035			Unimplemented							
$0036	Alert Control Register (ALCR)	Read: Write:	0	0	0	0	AL3	AL2	AL1	AL0
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented U = Undetermined X = Indeterminate

Figure 2-3. Control, Status, and Data Registers (Sheet 6 of 10)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0037	Alert Data Register (ALDR)	Read: Write:	SPL7	SPL6	SPL5	SPL4	SPL3	SPL2	SPL1	SPL0
		Reset:	0	0	0	0	0	0	0	0
$0038	Timebase Control Register (TBXCR)	Read:	TBXIF	TBXIE	TBXR1	TBXR0	0	TBXON	XTALR1	XTALR0
		Write:					TACK			
		Reset:	0	0	0	0	0	0	0	0
$0039	Unimplemented									
$003A	Unimplemented									
$003B	Unimplemented									
$003C	Unimplemented									
$003D	Unimplemented									
$003E	Unimplemented									
$003F	MC68HC908PT48 Configuration Register (CONFIG)	Read: Write:	0	0	0	SSREC	SCIBDSRC	0	STOP	COPD
		Reset:	0	0	0	0	1	0	0	0
$003F	MC68HC08PT48 Mask Option Register (MOR)	Read: Write:	0	0	0	SSREC	SCIBDSRC	SEC	STOP	COPD
		Reset:	0	0	0	X	X	X	X	X
$0040	Unimplemented									
$0041	Unimplemented									

= Unimplemented U = Undetermined X = Indeterminate

Figure 2-3. Control, Status, and Data Registers (Sheet 7 of 10)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0042			Unimplemented							
$0043			Unimplemented							
$0044			Unimplemented							
$0045			Unimplemented							
$0046	PLL Control Register (PCTL)	Read:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
		Write:								
		Reset:	0	0	1	0	1	1	1	1
$0047	PLL Bandwidth Control Register (PBWC)	Read:	AUTO	LOCK	$\overline{\text{ACQ}}$	0	0	0	0	$\overline{\text{COE}}$
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0048	PLL Multiplier Select Register High (PMSH)	Read:	0	0	0	0	MUL11	MUL10	MUL9	MUL8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0049	PLL Multiplier Select Register Low (PMSL)	Read:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004A	PLL VCO Range Select Register (PVRS)	Read:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
		Write:								
		Reset:	0	1	0	0	0	0	0	0
$004B	PLL Reference Divider Select Register (PRDS)	Read:	0	0	0	0	RDS3	RDS2	RDS1	RDS0
		Write:								
		Reset:	0	0	0	0	0	0	0	1
$004C	Analog-to-Digital Status and Control Register (ADSCR)	Read:	COCO/ IDMAS	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:								
		Reset:	0	0	0	1	1	1	1	1

= Unimplemented U = Undetermined X = Indeterminate

Figure 2-3. Control, Status, and Data Registers (Sheet 8 of 10)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$004D	Analog-to-Digital Data Register (ADR)	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004E	Analog-to-Digital Clock Register (ADCLKR)	Read:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004F	MC68HC908PT48 16 K-FLASH 2 Block Protect Register (FL2BPR)	Read:								
		Write:					F2BPR3	F2BPR2	F2BPR1	F2BPR0
		Reset:	X	X	X	X	1	1	1	1
↓										
$FE00	SIM Break Status Register (SBSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							Note 1	
		Reset:							0	
$FE01	SIM Reset Status Register (SRSR)	Read:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
		Write:								
		Reset:	1	0	0	0	0	0	0	0
$FE02			Reserved							
$FE03	SIM Break Flag Control Register (SBFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE04			Reserved							
$FE05			Reserved							
$FE06			Reserved							

Note 1. Writing a logic 0 clears SBSW.

= Unimplemented U = Undetermined X = Indeterminate

Figure 2-3. Control, Status, and Data Registers (Sheet 9 of 10)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE07			Reserved							
$FE08	FLASH 3 Control Register (FL3CR)	Read: Write:	F3DIV1	F3DIV0	F3BLK1	F3BLK0	HVEN	MARG	ERASE	PGM
		Reset:	0	0	0	0	0	0	0	0
$FE09			Reserved							
$FE0A	FLASH2 Control Register (FL2CR)	Read: Write:	F2DIV1	F2DIV0	F2BLK1	F2BLK0	HVEN	MARG	ERASE	PGM
		Reset:	0	0	0	0	0	0	0	0
$FE0B			Reserved							
$FE0C	FLASH 1 Control Register (FL1CR)	Read: Write:	F1DIV1	F1DIV0	F1BLK1	F1BLK0	HVEN	MARG	ERASE	PGM
		Reset:	0	0	0	0	0	0	0	0
$FE0D	Break Address Register High (BRKH)	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$FE0E	Break Address Register Low (BRKL)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$FE0F	Break Status/Control Register (BRKSCR)	Read: Write:	BRKE	BRKA						
		Reset:	0	0	0	0	0	0	0	0
↓										
$FFFF	COP Control Register (COPCTL)	Read:	Low Byte of Reset Vector							
		Write:	Clear COP Counter							
		Reset:	Unaffected by reset							

= Unimplemented U = Undetermined X = Indeterminate

Figure 2-3. Control, Status, and Data Registers (Sheet 10 of 10)

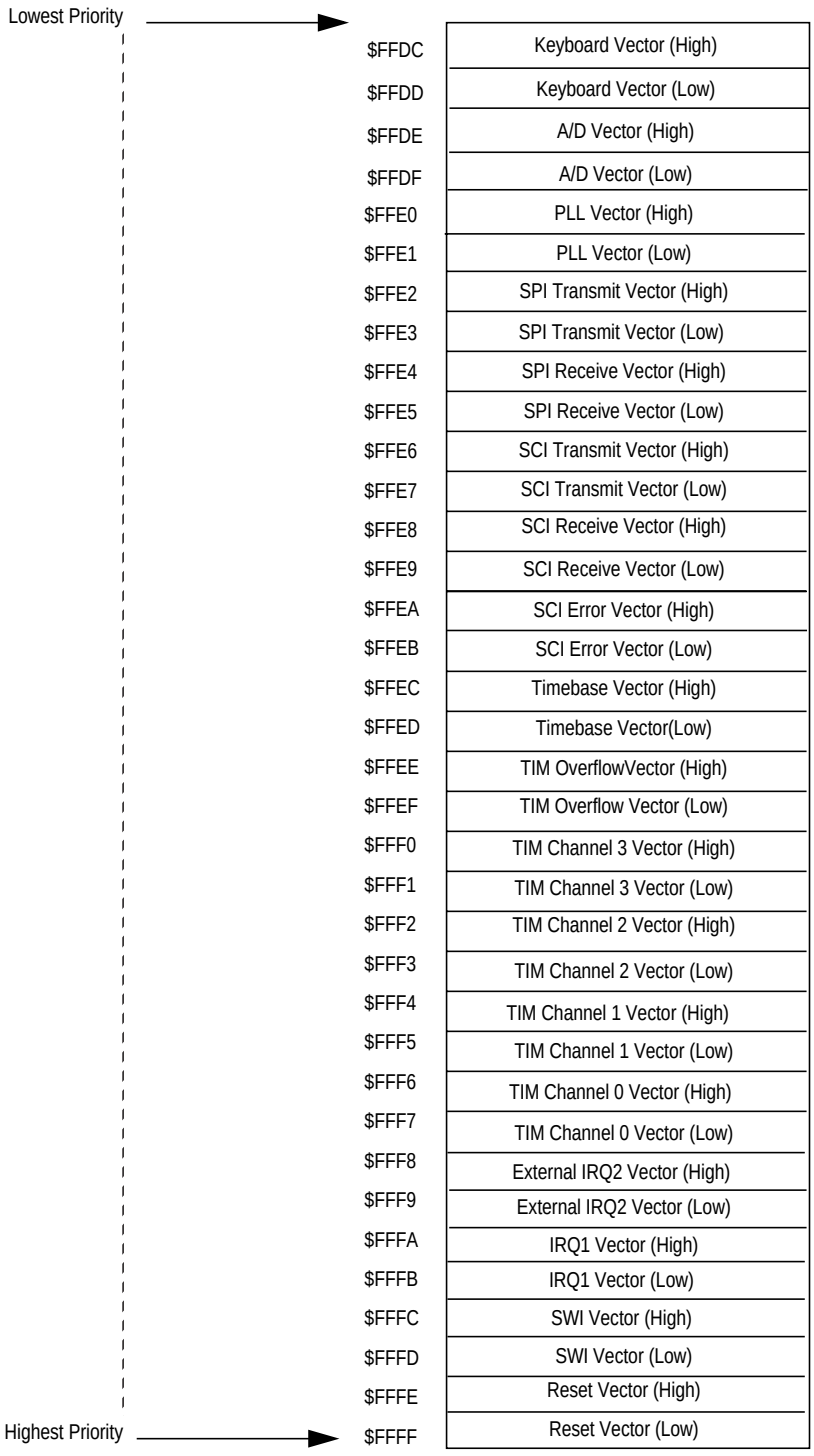


Figure 2-4. Vector Addresses in FLASH/ROM

2.6 User FLASH/ROM

The MCU has 48 Kbytes of FLASH (MC68HC908PT48) or mask programmable ROM (MC68HC08PT48). These addresses are user FLASH/ROM locations:

- $4000–$FDFF
- $FFDC–$FFFF; reserved for user-defined interrupt and reset vectors)

2.7 FLASH PROM

The MCU has 2 Kbytes of FLASH separate from the main array. These addresses are FLASH locations:

- $1800–$1FFF

2.8 Monitor ROM

The 240 bytes at addresses $FE10–$FEFF are reserved ROM addresses that contain the instructions for the monitor functions. For more information, see **Section 18. Monitor ROM (MON)**.

Memory Map

Section 3. Central Processor Unit (CPU)

3.1 Contents

3.2 Introduction

This section describes the central processor unit (CPU8, Version A). The M68HC08 CPU 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.

3.3 Features

Features of the CPU include:

- Full upward, object-code compatibility with M68HC05 Family
- 16-bit stack pointer with stack manipulation instructions
- 16-bit index register with X-register manipulation instructions
- 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

3.4 CPU Registers

Figure 3-1 shows the five CPU registers. CPU registers are not part of the memory map.

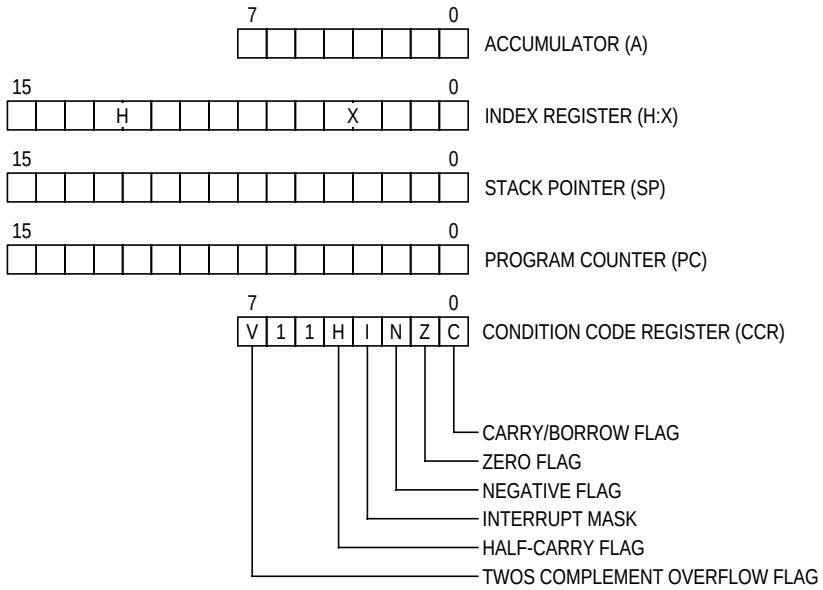


Figure 3-1. CPU Registers

3.4.1 Accumulator

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

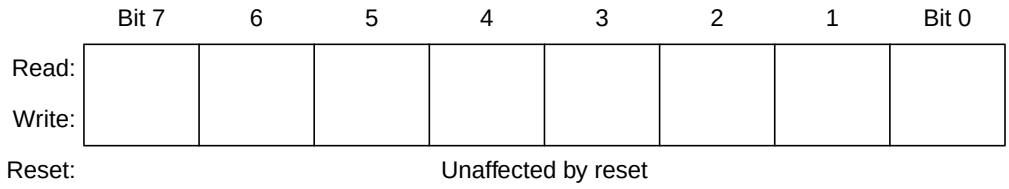


Figure 3-2. Accumulator (A)

3.4.2 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.

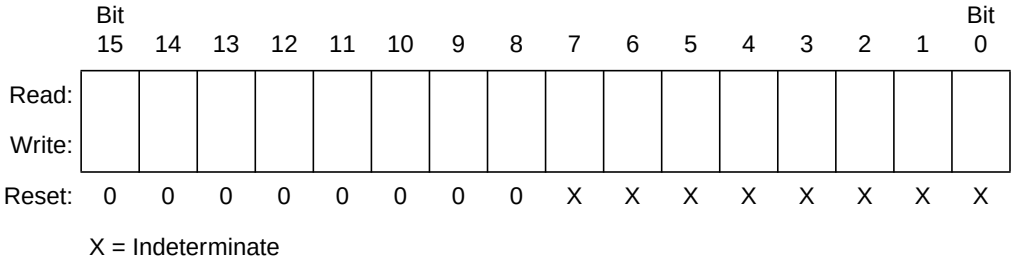


Figure 3-3. Index Register (H:X)

3.4.3 Stack Pointer

The stack pointer (SP) is a 16-bit register that contains the address of the next location on the stack. During a reset, the stack pointer is preset to $00FF. The reset stack pointer (RSP) instruction sets the least significant byte (LSB) to $FF and does not affect the most significant byte (MSB). The stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

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

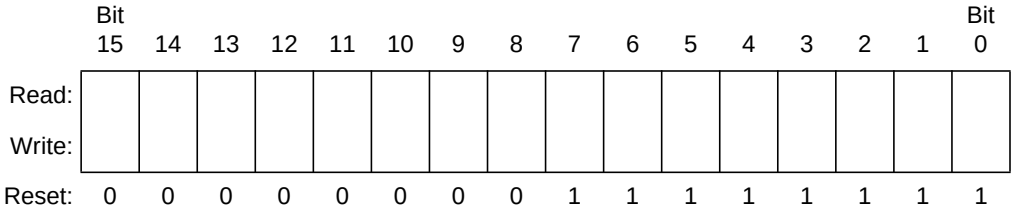


Figure 3-4. Stack Pointer (SP)

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

3.4.4 Program Counter

The program counter (PC) 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.

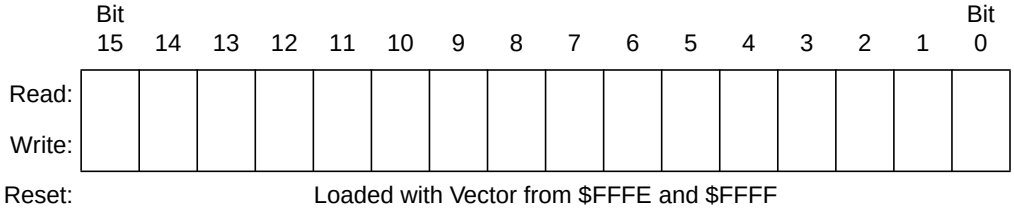


Figure 3-5. Program Counter (PC)

3.4.5 Condition Code Register

The 8-bit condition code register (CCR) contains the interrupt mask and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to logic 1. The 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 3-6. Condition Code Register (CCR)

V — Overflow Flag

The CPU sets the overflow flag when a twos 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 ADC (add with carry) or ADD (add without carry) operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA (decimal adjust A) instruction uses the states of the H and C flags to determine the appropriate correction factor.

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

I — Interrupt Mask

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

1 = Interrupts disabled
0 = Interrupts enabled

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

3.5 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 CPU architecture.

3.6 CPU During Break Interrupts

If the break module is enabled, a break interrupt causes the CPU to execute the software interrupt instruction (SWI) at the completion of the current CPU instruction. (See **Section 19. Break Module**.) The program counter vectors to $FFFC–$FFFD ($FEFC–$FEFD in monitor mode).

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

3.7 Instruction Set Summary

Table 3-1. Instruction Set Summary

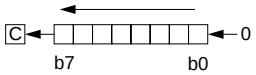
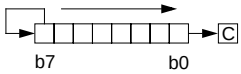
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X ADC *opr*,SP ADC *opr*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9	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
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

Table 3-1. Instruction Set Summary (Continued)

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

Table 3-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
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 ← $00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	3 1 1 1 3 2 4
CMP #*opr* CMP *opr* CMP *opr* CMP *opr*,X CMP *opr*,X CMP ,X CMP *opr*,SP CMP *opr*,SP	Compare A with M	(A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A1 B1 C1 D1 E1 F1 9EE1 9ED1	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
COM *opr* COMA COMX COM *opr*,X COM ,X COM *opr*,SP	Complement (One's Complement)	M ← ($\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

Table 3-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
DBNZ *opr,rel* DBNZA *rel* DBNZX *rel* DBNZ *opr*,X,*rel* DBNZ X,*rel* DBNZ *opr*,SP,*rel*	Decrement and Branch if Not Zero	A ← (A) – 1 or M ← (M) – 1 or X ← (X) – 1 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 4 + *rel* ? (result) ≠ 0	–	–	–	–	–	–	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr rr ff rr	5 3 3 5 4 6
DEC *opr* DECA DECX DEC *opr*,X DEC ,X DEC *opr*,SP	Decrement	M ← (M) – 1 A ← (A) – 1 X ← (X) – 1 M ← (M) – 1 M ← (M) – 1 M ← (M) – 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff	4 1 1 4 3 5
DIV	Divide	A ← (H:A)/(X) H ← Remainder	–	–	–	–	↕	↕	INH	52		7
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X EOR *opr*,SP EOR *opr*,SP	Exclusive OR M with A	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9EE8 9ED8	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
INC *opr* INCA INCX INC *opr*,X INC ,X INC *opr*,SP	Increment	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1 M ← (M) + 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	4 1 1 4 3 5
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

Table 3-1. Instruction Set Summary (Continued)

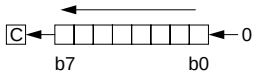
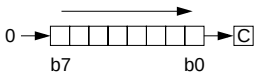
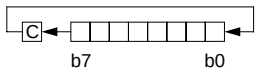
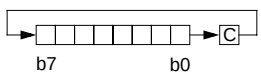
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LSL *opr* LSLA LSLX LSL *opr*,X LSL ,X LSL *opr*,SP	Logical Shift Left (Same as ASL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
LSR *opr* LSRA LSR*X* LSR *opr*,X LSR ,X LSR *opr*,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	4 1 1 4 3 5
MOV *opr,opr* MOV *opr*,X+ MOV #*opr,opr* MOV X+,*opr*	Move	$(M)_{Destination} \leftarrow (M)_{Source}$ H:X ← (H:X) + 1 (IX+D, DIX+)	0	–	–	↕	↕	–	DD DIX+ IMD IX+D	4E 5E 6E 7E	dd dd dd ii dd dd	5 4 4 4
MUL	Unsigned multiply	X:A ← (X) × (A)	–	0	–	–	–	0	INH	42		5
NEG *opr* NEGA NEGX NEG *opr*,X NEG ,X NEG *opr*,SP	Negate (Two's Complement)	M ← –(M) = $00 – (M) A ← –(A) = $00 – (A) X ← –(X) = $00 – (X) M ← –(M) = $00 – (M) M ← –(M) = $00 – (M)	↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	4 1 1 4 3 5
NOP	No Operation	None	–	–	–	–	–	–	INH	9D		1
NSA	Nibble Swap A	A ← (A[3:0]:A[7:4])	–	–	–	–	–	–	INH	62		3
ORA #*opr* ORA *opr* ORA *opr* ORA *opr*,X ORA *opr*,X ORA ,X ORA *opr*,SP ORA *opr*,SP	Inclusive OR A and M	A ← (A) | (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AA BA CA DA EA FA 9EEA 9EDA	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
PSHA	Push A onto Stack	Push (A); SP ← (SP) – 1	–	–	–	–	–	–	INH	87		2
PSHH	Push H onto Stack	Push (H); SP ← (SP) – 1	–	–	–	–	–	–	INH	8B		2
PSHX	Push X onto Stack	Push (X); SP ← (SP) – 1	–	–	–	–	–	–	INH	89		2
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–	–	INH	88		2
ROL *opr* ROLA ROLX ROL *opr*,X ROL ,X ROL *opr*,SP	Rotate Left through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	4 1 1 4 3 5
ROR *opr* RORA RORX ROR *opr*,X ROR ,X ROR *opr*,SP	Rotate Right through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	4 1 1 4 3 5

Table 3-1. Instruction Set Summary (Continued)

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

Table 3-1. Instruction Set Summary (Continued)

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

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

3.8 Opcode Map

See **Table 3-2**.

Table 3-2. Opcode Map

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

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

*Pre-byte for stack pointer indexed instructions

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

Cycles
Opcode Mnemonic
Number of Bytes / Addressing Mode

Section 4. Clock Generator Module (CGMB)

4.1 Contents

4.2 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 the base clock signal, CGMOUT, from which the system integration module (SIM) derives the system clocks. CGMOUT is based on either the crystal clock divided by two or the phase-locked loop (PLL) clock, CGMVCLK, divided by two. The PLL is a fully functional frequency generator designed for use with crystals or ceramic resonators. The PLL can generate a 4-MHz bus frequency without using a 16-MHz crystal.

4.3 Features

Features of the CGMB include:

- Phase-locked loop with output frequency in integer multiples of an integer dividend of the crystal reference
- Low -requency crystal operation with low-power operation and high-output frequency resolution
- Programmable reference divider for even 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
- Fast stop recovery mode for exiting stop mode even without a stable crystal

4.4 Functional Description

The CGMB consists of three major submodules:

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

Figure 4-1 shows the structure of the CGM.

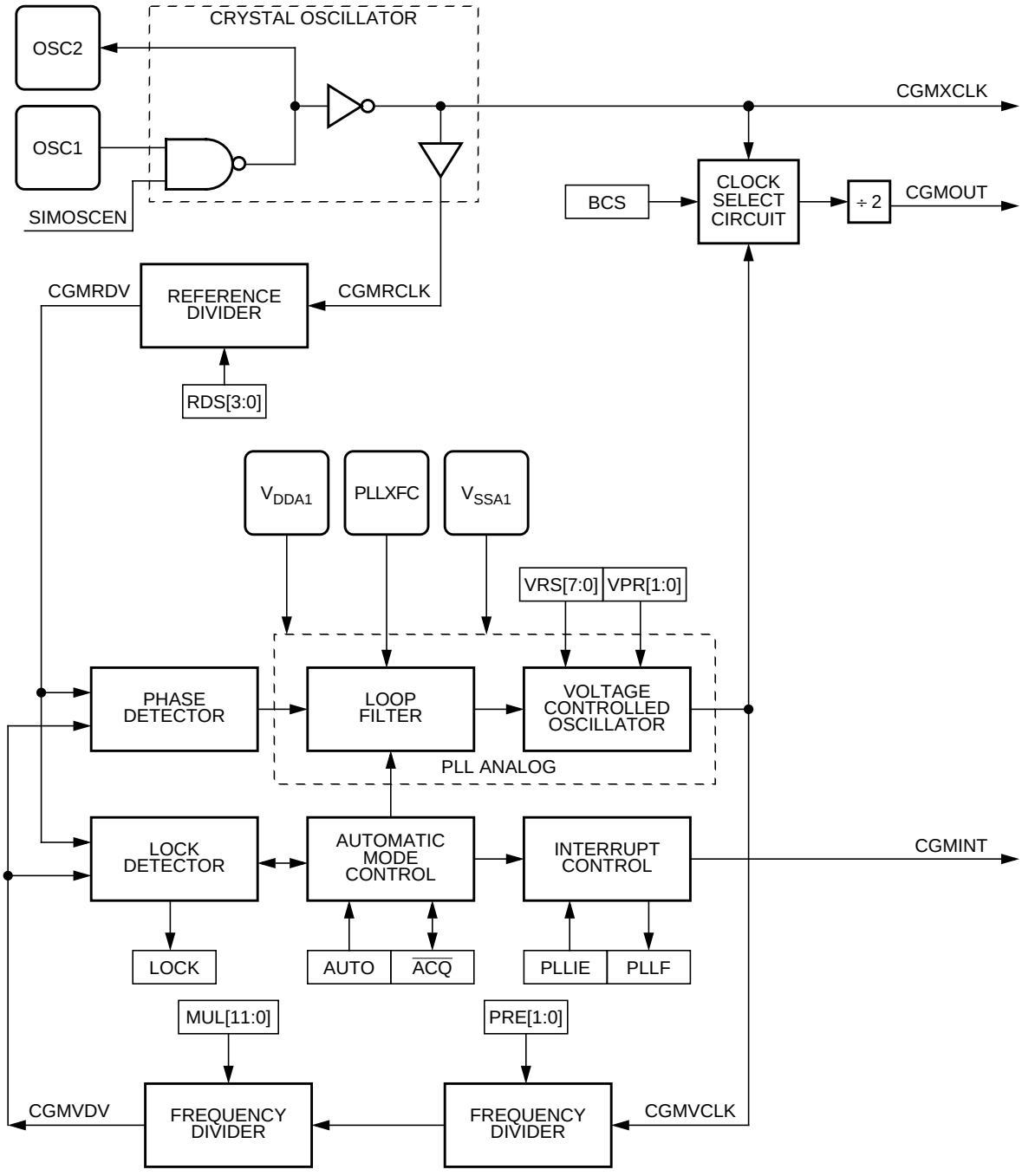


Figure 4-1. CGMB Block Diagram

4.4.1 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.

4.4.2 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.

4.4.2.1 PLL Circuits

The PLL consists of these 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 CGMXFC 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 CGMXFC 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 prescale 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 **4.4.2.4 Programming the PLL** for more information.)

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 CGMXFC based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, described in **4.4.2.2 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.

4.4.2.2 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 $\overline{\text{ACQ}}$ bit is clear in the PLL bandwidth control register. (See **4.6.2 PLL Bandwidth Control Register**.)
- 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 **4.4.3 Base Clock Selector Circuit**.) The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{\text{ACQ}}$ bit is set.

4.4.2.3 Manual and Automatic PLL Bandwidth Modes

The PLL can change the bandwidth or operational mode of the loop filter manually or automatically.

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 **4.6.2 PLL Bandwidth Control Register**.) 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 startup, 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 **4.4.3 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 **4.7 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 **4.6.2 PLL Bandwidth Control Register**) is a read-only indicator of the mode of the filter. (See **4.4.2.2 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 **4.10 Acquisition/Lock Time Specifications** for more information.)
- 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 **4.10 Acquisition/Lock Time Specifications** for more information.)
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. (See **4.6.1 PLL Control Register**.)

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} and require fast startup. 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 **4.10 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.

4.4.2.4 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 reference frequency, f_{RCLK}.
4. Select the prescaler power-of-two multiplier, P.

$$P = \text{integer}\left[\frac{\log\left(\frac{2^{PMAX} \times f_{VCLKDES}}{f_{VCLKMAX}}\right)}{\log(2)}\right] + 1$$

5. Select the reference divider based on the resolution desired. For maximum resolution, use this formula. However, higher degrees of resolution slow down the final reference frequency, which may cause acquisition time to increase and may affect the value of the external capacitor. For more information, see **4.10 Acquisition/Lock Time Specifications**.

$$R = \text{round}\left[R_{MAX} \times \left\{\left(\frac{f_{VCLKDES}}{2^P \times f_{RCLK}}\right) - \text{integer}\left(\frac{f_{VCLKDES}}{2^P \times f_{RCLK}}\right)\right\}\right]$$

Select a VCO frequency multiplier, N.

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

6. For fastest acquisition time, reduce N/R until R is the smallest value possible. For example, if N = 6 and R = 4, N reduces to 3 and R reduces to 2.
7. Calculate and verify the adequacy of the VCO and bus frequencies f_{VCLK} and f_{BUS}.

$$f_{VCLK} = N \times f_{RCLK}$$

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

$$f_{BUS} = (f_{VCLK})/4$$

8. Select the VCO's power-of-two range multiplier E. Higher values of E should be used at higher frequencies.

$$E = \text{integer}\left[\frac{\log\left(\frac{2^{EMAX} \times f_{VCLK}}{f_{VRSMAX}}\right)}{\log(2)}\right] + 1$$

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}.

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

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**, program the binary equivalent of P.
    b. In the VPR bits of the **PLL Control Register**, program the binary equivalent of E.
    c. In the **PLL Multiplier Select Register Low** and the **PLL Multiplier Select Register High**, program the binary equivalent of N.
    d. In the **PLL VCO Range Select Register**, program the binary coded equivalent of L.
    e. In the **PLL Reference Divider Select Register**, program the binary coded equivalent of R.

Table 4-1 provides a numeric example with numbers in hexadecimal notation.

Table 4-1. Numeric Example

Bus Frequency	E	P	N	L	R
307,200 Hz	1	1	10	10	1
614,400 Hz	1	1	20	20	1
652,800 Hz	2	2	11	11	1
691,200 Hz	2	2	12	12	1
729,600 Hz	2	2	13	13	1
768,000 Hz	2	2	14	14	1
806,400 Hz	2	2	15	15	1
998,400 Hz	2	2	1a	1a	1

4.4.2.5 Special Programming Exceptions

The programming method described in **4.4.2.4 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 0 value for R or N is interpreted exactly the same as a value of 1. at the minimum frequency and the VCO range power-of-two bits.This mode is currently disabled in MT2.
- A 0 value for L disables the PLL and prevents its selection as the source for the base clock. (See **4.4.3 Base Clock Selector Circuit**.)

4.4.3 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 0. 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.

4.4.4 CGMB External Connections

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

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in **Figure 4-2**. **Figure 4-2** shows only the logical representation of the internal components and may not represent actual circuitry.

The oscillator configuration uses five components:

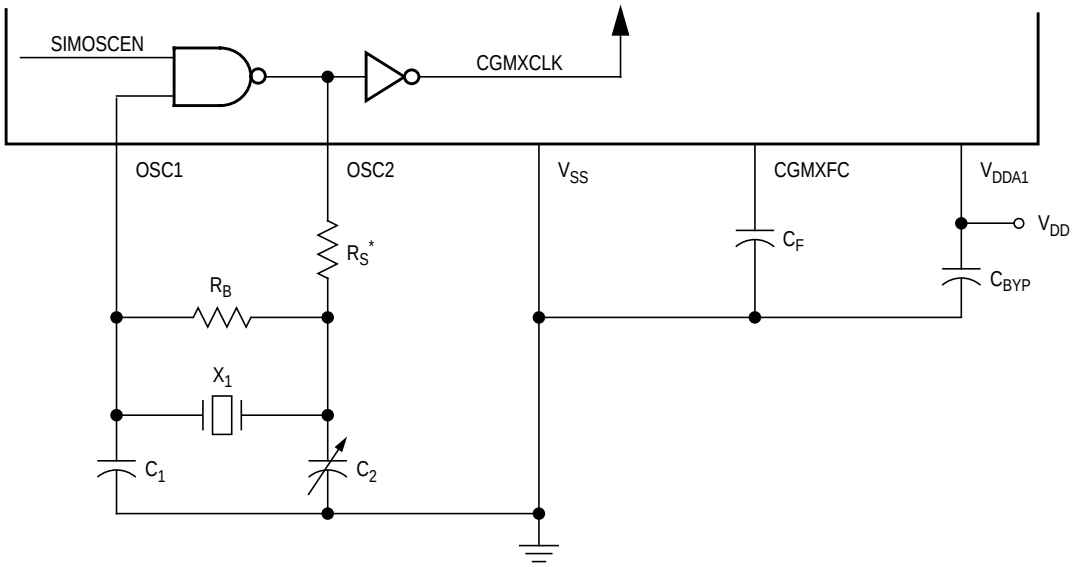
- Crystal, X_1
- Fixed capacitor, C_1
- Tuning capacitor, C_2; can also be a fixed capacitor
- Feedback resistor, R_B
- Series resistor, R_S; optional

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 4-2 also shows the external components for the PLL:

- Bypass capacitor, C_{BYP}
- Filter capacitor, C_F

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



*Rs can be 0 (shorted) when used with higher-frequency crystals. Refer to manufacturer's data.

Figure 4-2. CGMB External Connections

4.5 I/O Signals

The following paragraphs describe the CGMB I/O (input/output) signals. The CGM may also have up to four additional inputs, if enabled in MT2.

4.5.1 Crystal Amplifier Input Pin (OSC1)

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

4.5.2 Crystal Amplifier Output Pin (OSC2)

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

4.5.3 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.*

4.5.4 PLL Analog Power Pin (V_{DD1})

V_{DDA1} is a power pin used by the analog portions of the PLL. Connect the V_{DDA1} pin to the same voltage potential as the V_{DD} pin.

NOTE: *Route V_{DDA1} carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.*

4.5.5 PLL Analog Ground Pin (V_{SSA1})

V_{SSA1} is a ground pin used by the analog portions of the PLL. Connect the V_{SSA1} pin to the same voltage potential as the V_{SS} pin.

NOTE: *Route V_{SSA1} carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.*

4.5.6 Buffered Crystal Clock Output (CGMVOUT)

CGMVOUT buffers the OSC1 clock for external use.

4.5.7 CGMVSEL

CGMVSEL must be tied low or floated.

4.5.8 Oscillator Enable Signal (SIMOSCEN)

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

4.5.9 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 4-2** 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 startup.

4.5.10 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 percent 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 two.

4.5.11 CGMB CPU Interrupt (CGMINT)

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

4.6 CGMB Registers

The following registers control and monitor operation of the CGMB:

- PLL control register (PCTL)
  See **4.6.1 PLL Control Register**.
- PLL bandwidth control register (PBWC)
  See **4.6.2 PLL Bandwidth Control Register**.
- PLL multiplier select register high (PMSH)
  See **4.6.3 PLL Multiplier Select Register High**.
- PLL multiplier select register low (PMSL)
  See **4.6.4 PLL Multiplier Select Register Low**.
- PLL VCO range select register
  See **4.6.5 PLL VCO Range Select Register**.
- PLL reference divider select register (PRDS)
  See **4.6.6 PLL Reference Divider Select Register**.

Figure 4-3 is a summary of the CGMB registers.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0046	PLL Control Register (PCTL)	Read:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
		Write:								
		Reset:	0	0	1	0	1	1	1	1
$0047	PLL Bandwidth Control Register (PBWC)	Read:	AUTO	LOCK	$\overline{\text{ACQ}}$	0	0	0	0	$\overline{\text{COE}}$
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0048	PLL Multiplier Select Register High (PMSH)	Read:	0	0	0	0	MUL11	MUL10	MUL9	MUL8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0049	PLL Multiplier Select Register Low (PMSL)	Read:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004A	PLL VCO Range Select Register (PVRS)	Read:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
		Write:								
		Reset:	0	1	0	0	0	0	0	0
$004B	PLL Reference Divider Select Register (PRDS)	Read:	0	0	0	0	RDS3	RDS2	RDS1	RDS0
		Write:								
		Reset:	0	0	0	0	0	0	0	1

= Unimplemented

Notes:

1. When AUTO = 0, PLLIE is forced clear and is read only.
2. When AUTO = 0, PLLF and LOCK read as clear.
3. When AUTO = 1, $\overline{\text{ACQ}}$ is read-only.
4. When PLLON = 0 or VRS7:VRS0 = $0, BCS is forced clear and is read only.
5. When PLLON = 1, the PLL programming register is read only.
6. When BCS = 1, PLLON is forced set and is read only.

Figure 4-3. CGMB I/O Register Summary

4.6.1 PLL Control Register

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.

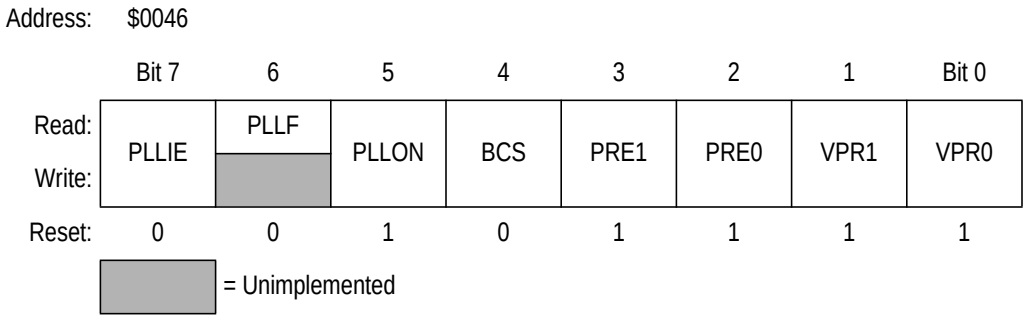


Figure 4-4. PLL Control Register (PCTL)

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 0. 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 0 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 **4.4.3 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, 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 **4.4.3 Base Clock Selector Circuit**.) Reset clear the BCS bit.

1 = CGMVCLK 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 **4.4.3 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 **4.4.2.1 PLL Circuits** and **4.4.2.4 Programming the PLL**.) PRE1 and PRE0 cannot be written when the PLLON bit is set. Reset clears these bits. (See **Table 4-2**.)

Table 4-2. PRE1 and PRE0 Programming

PRE1 and PRE0	P	Prescaler Multiplier
00	0	1
01	1	2
10	2	4
11	3	8

VPR1 and VPR0 — 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 **4.4.2.1 PLL Circuits**, **4.4.2.4 Programming the PLL**, and **4.6.5 PLL VCO Range Select Register**.) controls the hardware center-of-range frequency f_{VRS}. VPR1:VPR0 cannot be written when the PLLON bit is set. Reset clears these bits. (See **Table 4-3**.)

Table 4-3. VPR1 and VPR0 Programming

VPR1 and VPR0	E	VCO Power-of-Two Range Multiplier
00	0	1
01	1	2
10	2	4
11	3	8

4.6.2 PLL Bandwidth Control Register

The PLL bandwidth control register:

- 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

Address: $0047

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

= Unimplemented

Figure 4-5. PLL Bandwidth 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 $\overline{\text{ACQ}}$ 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 0 and has no meaning. The write function of this bit is reserved for test, so this bit must always be written a 0. 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

$\overline{COE}$— Clock Out Enable

When the $\overline{COE}$ bit is cleared, a buffered version of OSC1 is present on the CGMVOUT pin. When the $\overline{COE}$ bit is set, the CGMVOUT pin will be driven low.

1 = CGMVOUT driven low
0 = CGMVOUT is buffered OSC1

4.6.3 PLL Multiplier Select Register High

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

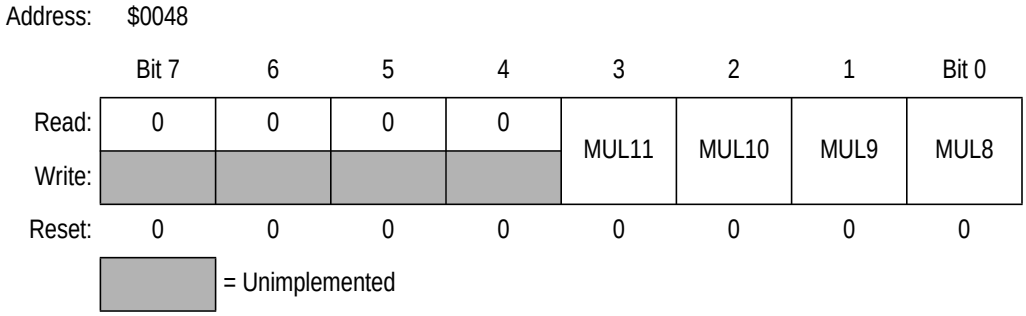


Figure 4-6. PLL Multiplier Select Register High (PMSH)

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 **4.4.2.1 PLL Circuits** and **4.4.2.4 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).*

4.6.4 PLL Multiplier Select Register Low

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

Address: $0049

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

Figure 4-7. 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 **4.4.2.1 PLL Circuits** and **4.4.2.4 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).*

4.6.5 PLL VCO Range Select Register

The PLL VCO range select register contains the programming information required for the hardware configuration of the VCO.

Address: $004A

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

Figure 4-8. PLL VCO Range Select Register (PVRS)

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 **4.4.2.1 PLL Circuits**, **4.4.2.4 Programming the PLL**, and **4.6.1 PLL Control Register**), 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 **4.4.2.5 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**. (See **4.4.3 Base Clock Selector Circuit** and **4.4.2.5 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.*

The PLL VCO range select register must be programmed correctly. Incorrect programming may result in failure of the PLL to achieve lock.

4.6.6 PLL Reference Divider Select Register

The PLL reference divider select register contains the programming information for the modulo reference divider.

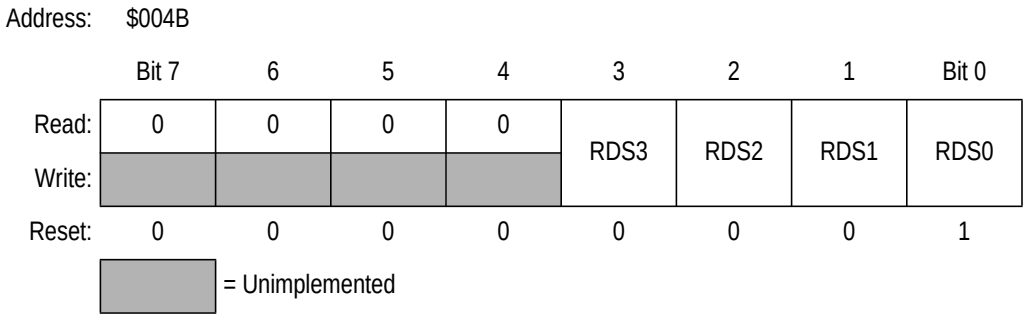


Figure 4-9. PLL Reference Divider Select Register (PRDS)

RDS[3:0] — Reference Divider Select Bits

These read/write bits control the modulo reference divider that selects the reference division factor R. (See **4.4.2.1 PLL Circuits** and **4.4.2.4 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 **4.4.2.5 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).*

PRDS[7:4] — Unimplemented Bits

These bits have no function and always read as logic 0s.

4.7 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 0.

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 CGMVCLK 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.*

4.8 Special Modes

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

4.8.1 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). Less power-sensitive applications can disengage the PLL without turning it off. Applications that require the PLL to wake the MCU from wait mode also can deselect the PLL output without turning off the PLL.

4.8.2 Stop Mode

The STOP instruction disables the CGMB and holds low all CGMB outputs (CGMXCLK, CGMOUT, and CGMINT).

If the STOP instruction is executed with the VCO clock, CGMVCLK, divided by two driving CGMOUT, the PLL automatically clears the BCS bit in the PLL control register (PCTL), thereby selecting the crystal clock, CGMXCLK, divided by two as the source of CGMOUT. When the MCU recovers from STOP, the crystal clock divided by two drives CGMOUT and BCS remains clear.

4.9 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**.)

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

To protect the PLLF bit during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write the PLL control register during the break state without affecting the PLLF bit.

If this mode is desired during reset, the reset conditions of BCS and PLLON must be set. If this mode is desired for use in applications where no crystal is used, the BCS and PLLON bits must not be clearable. During a large frequency change, the software must allow a stabilization time. The CGMXCLK signal will always reflect the crystal clock, so the value of CGMXCLK upon removing the crystal will reflect the value of the OSC1 pin. If OSC1 is floating, the module could consume significant power and the output of the CGMXCLK signal would be indeterminate.

4.10 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.

4.10.1 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 percent 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:

- 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 percent. In automatic bandwidth control mode (see **4.4.2.3 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 percent. In automatic bandwidth control mode, lock time expires when the LOCK bit becomes set in the PLL bandwidth control register (PBWC). (See **4.4.2.3 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.

4.10.2 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 also under user control via the choice of crystal frequency f_{XCLK} and the R value programmed in the reference divider. (See **4.4.2.1 PLL Circuits**, **4.4.2.4 Programming the PLL**, and **4.6.6 PLL Reference Divider Select Register**).

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 **4.10.3 Choosing a Filter Capacitor**.)

Also important is the operating voltage potential applied to V_{DDA1}. 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.

4.10.3 Choosing a Filter Capacitor

As described in **4.10.2 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 this equation:

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

For acceptable values of C_{FACT}, see **4.10 Acquisition/Lock Time Specifications**. For the value of V_{DDA1}, 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 percent or better) and low dissipation.

4.10.4 Reaction Time Calculation

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

- Correct selection of filter capacitor, C_F (See **4.10.3 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 **4.4.2.2 Acquisition and Tracking Modes**.)

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

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

$$t_{LOCK} = t_{ACQ} + t_{AL}$$

NOTE:T *he 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 **4.4.2.3 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}. Also, since the average frequency over the entire measurement period must be within the specified tolerance, the total time usually is longer than t_{LOCK} as calculated earlier.

In manual mode, it is usually necessary to wait considerably longer than t_{LOCK} before selecting the PLL clock (see **4.4.3 Base Clock Selector Circuit**) because the factors described in **4.10.2 Parametric Influences on Reaction Time** may slow the lock time considerably.

Section 5. Computer Operating Properly (COP) Module

5.1 Contents

5.2 Introduction

This section describes the computer operating properly module (COP, version B10), 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.

5.3 Functional Description

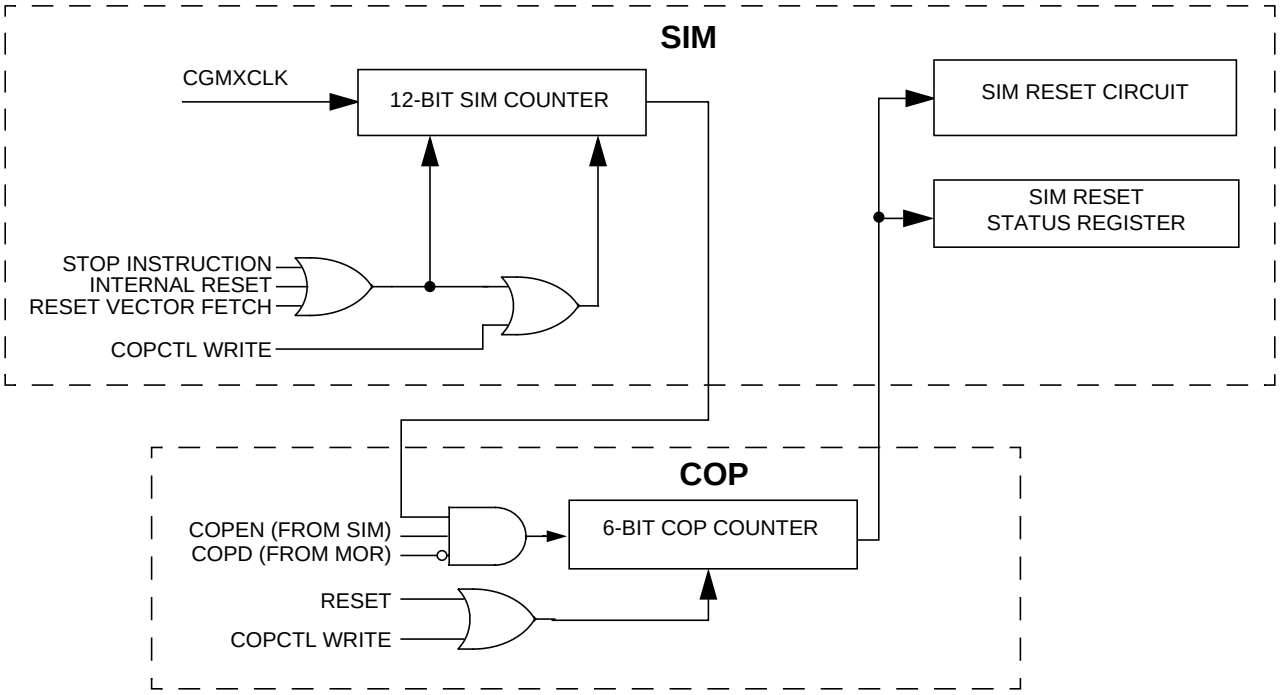


Figure 5-1. COP Block Diagram

The COP counter uses a free-running 6-bit counter preceded by the 13-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 38.4-kHz crystal, the COP timeout period is 6.83 seconds. 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 and sets the COP bit in the SIM reset status register (SRSR). 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 or a DMA service 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.*

5.4 I/O Signals

This section describes the signals shown in **Figure 5-1**.

5.4.1 CGMXCLK

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

5.4.2 STOP Instruction

The STOP instruction clears the SIM counter.

5.4.3 COPCTL Write

Writing any value to the 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.

5.4.4 Power-On Reset

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

5.4.5 Internal Reset

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

5.4.6 Reset Vetor Fetch

A reset vector fetch occurs when the vector address appears on the data bus. A reset vector fetch clears the SIM counter.

5.4.7 COPD

The COPD (COP disable) signal refects the state of the COP disable bit (COPD) in the mask option register (MOR).

5.5 COP Control Register

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

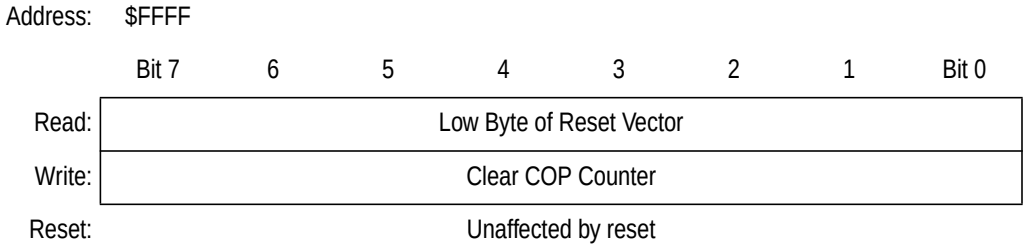


Figure 5-2. COP Control Register (COPCTL)

5.6 Interrupts

The COP does not generate CPU interrupt requests or DMA service requests.

5.7 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 $\overline{RST}$ pin.

5.8 Low-Power Modes

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

5.8.1 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.

5.8.2 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 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 bits to logic 0.

5.9 COP Module During Break Interrupts

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

Section 6. Keyboard Interrupt (KBI) Module

6.1 Contents

6.2 Introduction

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

6.3 Features

The KBI features include:

- Eight keyboard interrupt pins with separate keyboard interrupt enable bits and one keyboard interrupt mask
- Hysteresis buffers
- Programmable edge-only or edge- and level-interrupt sensitivity
- Automatic interrupt acknowledge
- Exit from low-power modes

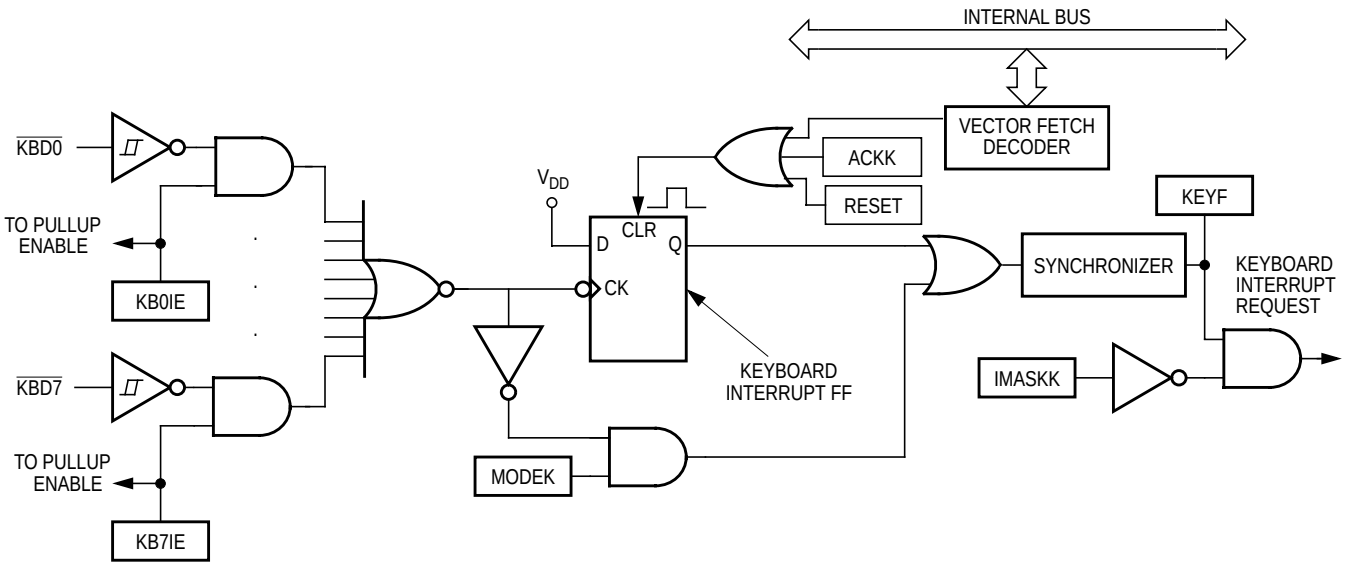


Figure 6-1. KBI Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0033	Keyboard Status and Control Register (KBSCR)	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
		Write:						ACKK		
		Reset:	0	0	0	0	0	0	0	0
$0034	Keyboard Interrupt Enable Register (KBICR)	Read:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented U = Undetermined X = Indeterminate

Figure 6-2. KBI Register Summary

6.4 Functional Description

Writing to the KBIE7–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port F pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its internal pullup device. A logic 0 applied to an enabled keyboard interrupt pin latches a keyboard interrupt request.

A keyboard interrupt request is latched when one or more keyboard pins goes low after all were high. The MODEK bit in the keyboard status and control register controls the triggering sensitivity of the keyboard interrupt pins.

- If the keyboard interrupt pins are 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 pins are falling edge- and low level-sensitive, an interrupt request is present as long as any keyboard pin is low.

If the MODEK bit is set, the keyboard interrupt pins are both falling edge- and low level-sensitive, and both of the following actions must occur to clear a keyboard interrupt request:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software can generate the interrupt acknowledge signal by writing a logic 1 to the ACKK bit in the keyboard status and control register. The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit in an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the vector address at locations $FFDC and $FFDD.

- Return of all enabled keyboard interrupt pins to logic 1 — As long as any enabled keyboard interrupt pin is at logic 0, the keyboard interrupt request remains set.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to logic 1 may occur in any order.

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

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt pin stays at logic 0.

The keyboard flag bit (KEYF) in the keyboard status and control register can be used to see if a pending interrupt exists. The KEYF bit is not affected by the keyboard interrupt mask bit (IMASKK) which makes it useful in applications where polling is preferred.

To determine the logic level on a keyboard interrupt pin, use the data direction register to configure the pin as an input and read the data register.

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

6.5 Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup to reach a logic 1. Therefore, a false interrupt can occur as soon as the pin is enabled.

To prevent a false interrupt on keyboard initialization:

1. Mask keyboard interrupts by setting the IMASKK bit in the keyboard status and control register.
2. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.

3. Write to the ACKK bit in the keyboard status and control register to clear any false interrupts.
4. Clear the IMASKK bit.

An interrupt signal on an edge-triggered pin can be acknowledged immediately after enabling the pin. An interrupt signal on an edge- and level-triggered interrupt pin must be acknowledged after a delay that depends on the external load.

Another way to avoid a false interrupt:

1. Configure the keyboard pins as outputs by setting the appropriate DDRF bits.
2. Write logic 1s to the appropriate port F data register bits.
3. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.

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 keyboard interrupt module remains active in wait mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of wait mode.

6.6.2 Stop Mode

The keyboard module remains active in stop mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of stop mode.

6.7 KBI During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See **7.8.3 SIM Break Flag Control Register**.)

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

To protect the KEYF bit during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0, writing to the keyboard acknowledge bit (ACKK) in the keyboard status and control register during the break state has no effect. (See **6.8.1 Keyboard Status and Control Register**.)

6.8 I/O Registers

These control and monitor operation of the keyboard interrupt module:

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

6.8.1 Keyboard Status and Control Register

The keyboard status and control register:

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

Address: $0033

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

= Unimplemented

Figure 6-3. Keyboard Status and Control Register (KBSCR)

Bits 7–4 — Not used

These read-only bits always read as logic 0s.

KEYF — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending. Reset clears the KEYF bit.

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

ACKK — Keyboard Acknowledge Bit

Writing a logic 1 to this write-only bit clears the keyboard interrupt request. ACKK always reads as logic 0. Reset clears ACKK.

IMASKK — Keyboard Interrupt Mask Bit

Writing a logic 1 to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests. Reset clears the IMASKK bit.

1 = Keyboard interrupt requests masked
0 = Keyboard interrupt requests not masked

MODEK — Keyboard Triggering Sensitivity Bit

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

1 = Keyboard interrupt requests on falling edges and low levels
0 = Keyboard interrupt requests on falling edges only

6.8.2 Keyboard Interrupt Enable Register

The keyboard interrupt enable register enables or disables each port F pin to operate as a keyboard interrupt pin.

Address: $0034

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
Reset:	0	0	0	0	0	0	0	0

Figure 6-4. Keyboard Interrupt Enable Register (KBIER)

KBIE7–KBIE0 — Keyboard Interrupt Enable Bits

Each of these read/write bits enables the corresponding keyboard interrupt pin to latch interrupt requests. Reset clears the keyboard interrupt enable register.

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

Section 7. System Integration Module (SIM)

7.1 Contents

7.2 Introduction

This section describes the system integration module (SIM24, version E), which supports up to 24 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**. **Figure 7-2** is a summary of the SIM input/output (I/O) registers. The SIM is a system state controller that coordinates CPU and exception timing.

The SIM is responsible for:

- Controlling mode selection
- 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 clock control with DMA transfer and low-power modes
- Slow memory read/write timing
- CPU enable/disable timing
- Modular architecture expandable to 128 interrupt sources

NOTE: *All references to LVI and DMA operation in this section should be ignored.*

Table 7-1 shows the internal signal names used in this section.

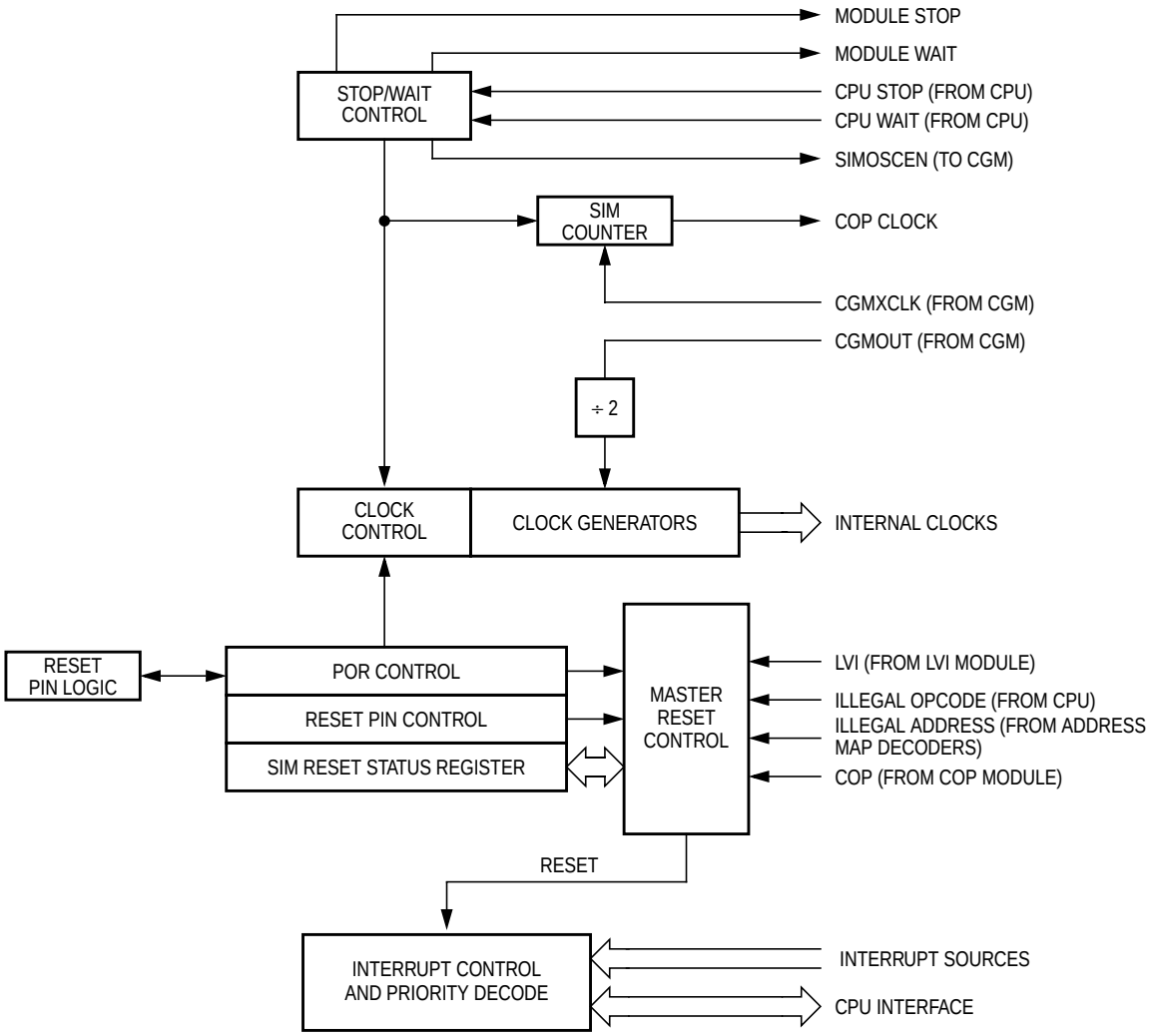


Figure 7-1. SIM Block Diagram

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

Note 1. Writing a logic 0 clears SBSW.

R = Reserved = Unimplemented

Figure 7-2. SIM I/O Register Summary

Table 7-1. 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 7-3**. This clock can come from either an external oscillator or from the on-chip PLL. For additional information, refer to **Section 4. Clock Generator Module (CGMB)**.

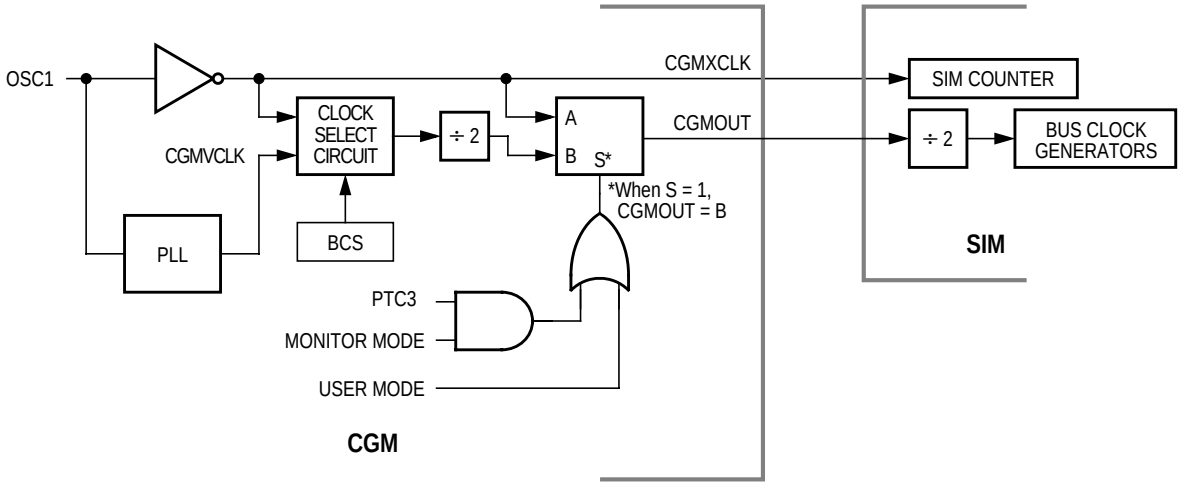


Figure 7-3. CGM Clock Signals

7.3.1 Bus Timing

In user mode**,** the internal bus frequency is either the crystal oscillator output (CGMXCLK) divided by four or the PLL output (CGMVCLK) divided by four. For additional information, refer to **Section 4. Clock Generator Module (CGMB)**.

7.3.2 Clock Startup from POR or LVI Reset

When the power-on reset (POR) module or the low-voltage inhibit (LVI) module generates a reset, the clocks to the CPU and peripherals are inactive and held in an inactive phase until after the 4096 CGMXCLK cycle POR timeout has completed. The $\overline{\text{RST}}$ pin is driven low by the SIM during this entire period. 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 inhibit module (LVI)
- Illegal opcode
- Illegal address

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

An internal reset clears the SIM counter (see **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 $\overline{RST}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{RST}$ is held low for a minimum of 67 CGMXCLK cycles, assuming that neither the POR nor the LVI was the source of the reset. See **Table 7-2** for details. **Figure 7-4** shows the relative timing.

Table 7-2. PIN Bit Set Timing

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

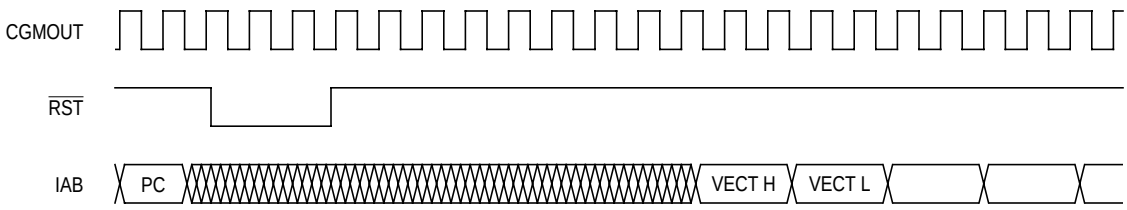


Figure 7-4. External Reset Timing

7.4.2 Active Resets from Internal Sources

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 7-5**.) An internal reset can be caused by an illegal address, illegal opcode, COP timeout, module reset, LVI, or POR. (See **Figure 7-6**.)

NOTE: *For LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM forces the $\overline{RST}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{RST}$ shown in **Figure 7-5**.*

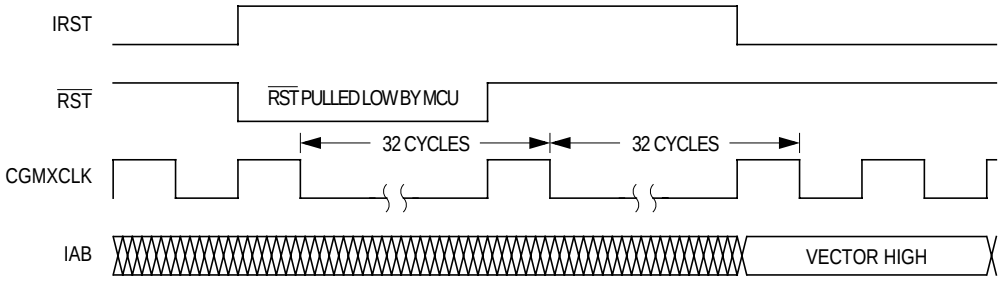


Figure 7-5. Internal Reset Timing

The COP reset is asynchronous to the bus clock.

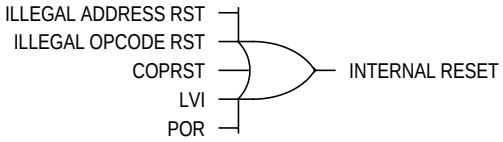


Figure 7-6. Sources of Internal Reset

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.

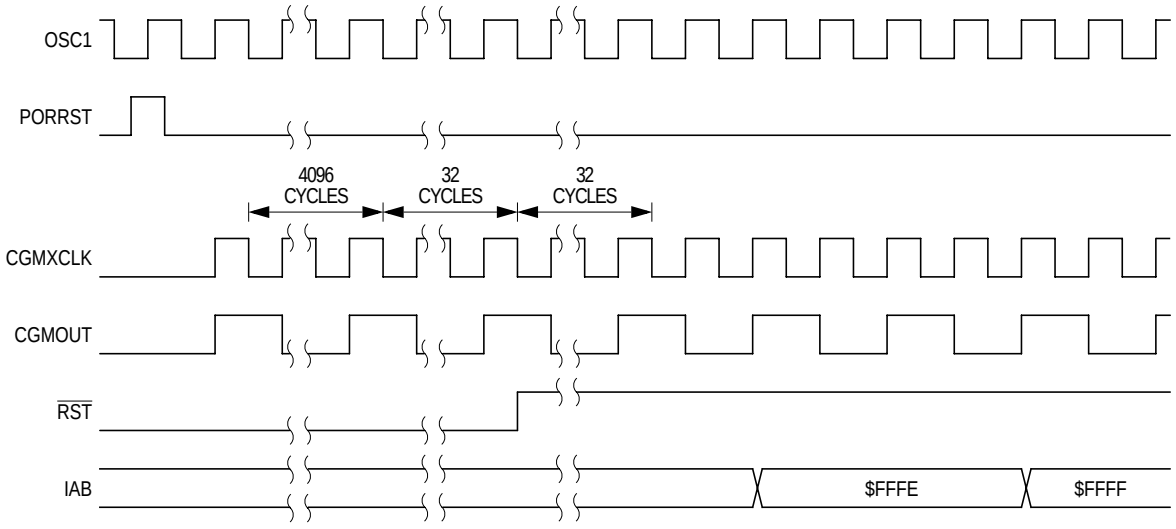


Figure 7-7. POR Recovery

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 $\overline{RST}$ 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 $\overline{RST}$ 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 $\overline{RST}$ 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 mask option register is logic 0, 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 Inhibit (LVI) Reset

The low-voltage inhibit module (LVI) asserts its output to the SIM when the V_{DD} voltage falls to the LVI_{TRIPF} 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. Reset recovery control logic and real time interrupt models also use taps from the SIM counter. 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 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 startup times from stop mode. External crystal applications should use the full stop recovery time, that is, with SSREC cleared (grounded).

7.5.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See **7.7.2 Stop Mode** for details.) 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

Normal, 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 7-8** shows interrupt entry timing. **Figure 7-9** shows interrupt recovery timing.

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 7-10**.)

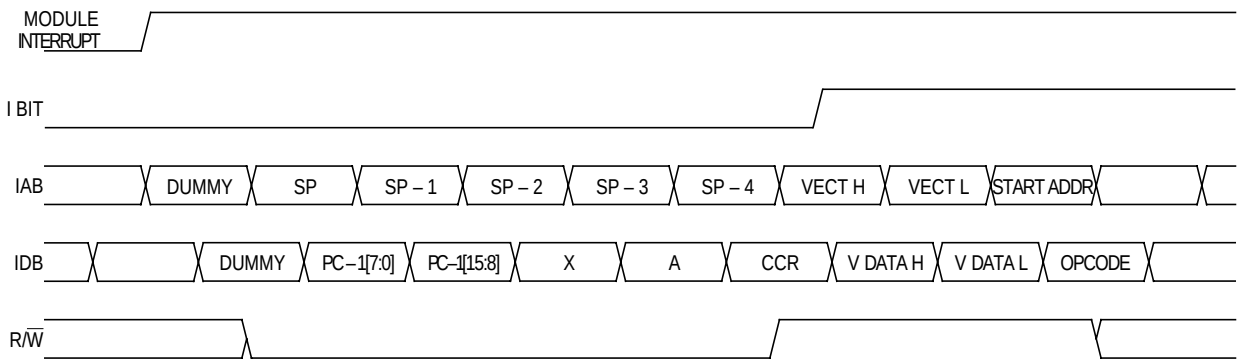


Figure 7-8. Interrupt Entry

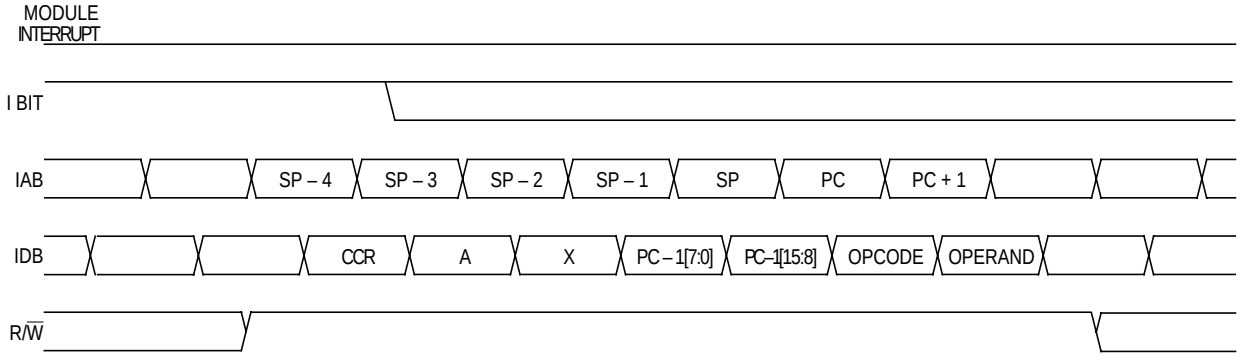


Figure 7-9. Interrupt Recovery

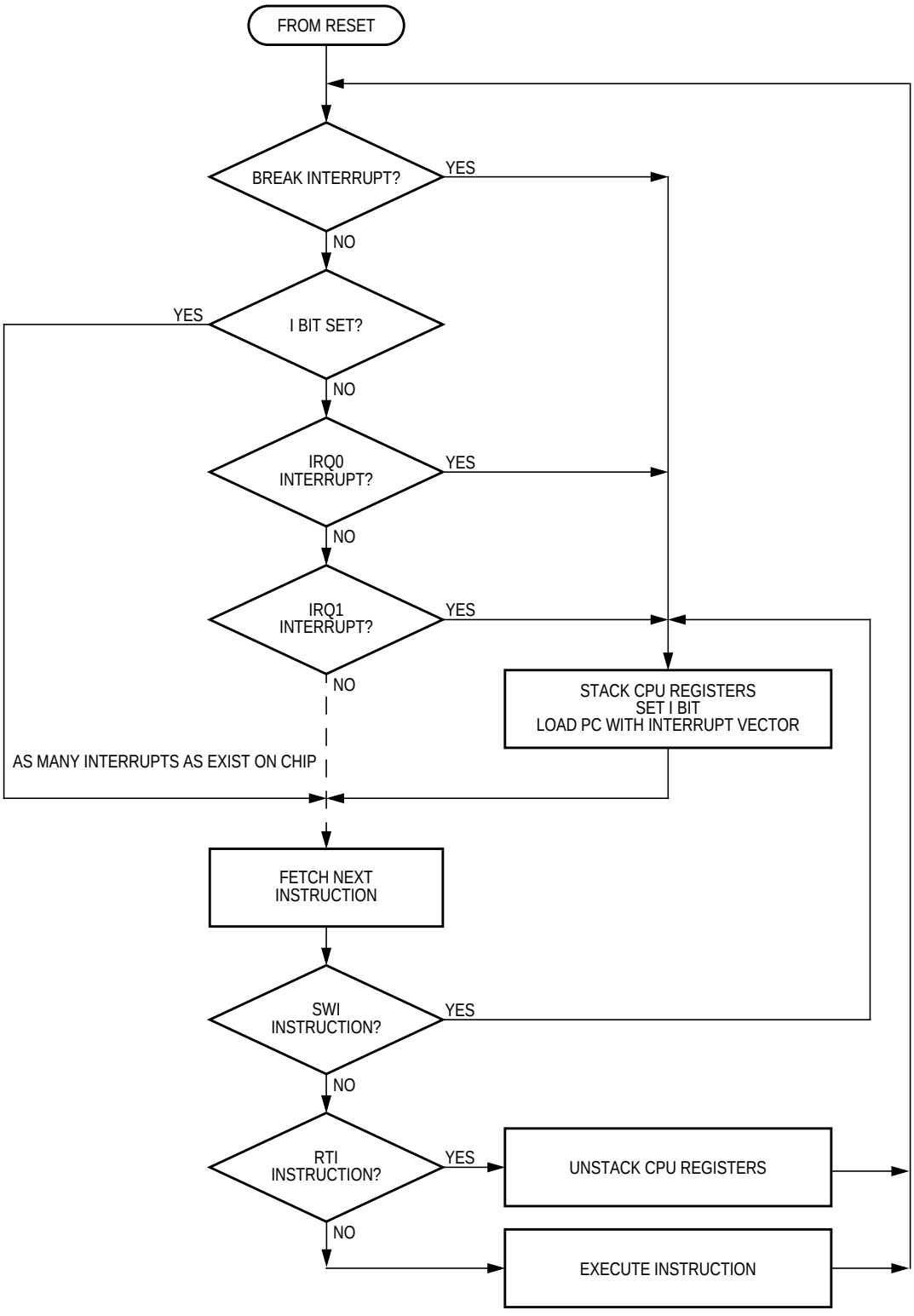


Figure 7-10. Interrupt Processing

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 7-11** 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.

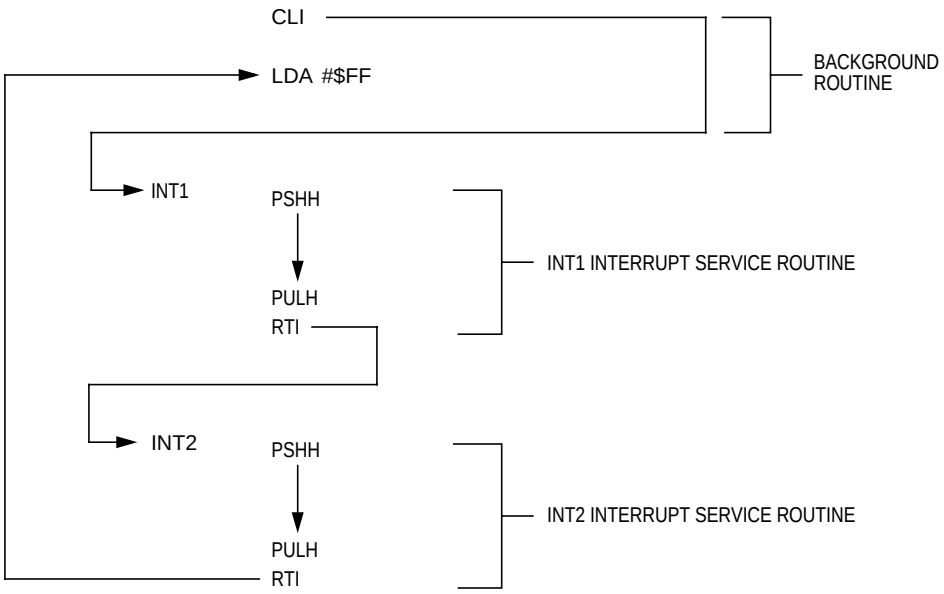


Figure 7-11. Interrupt Recognition Example

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

NOTE: *To maintain compatibility with the M6805 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 19. Break Module**.) 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.

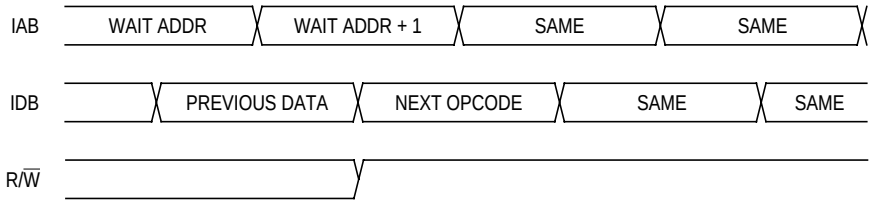
Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a 2-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step 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 in the subsections that follow. 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 7-12** shows the timing for wait mode entry.



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

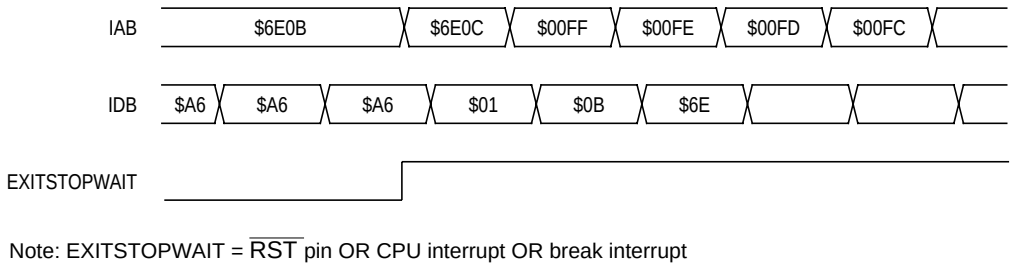
Figure 7-12. Wait Mode Entry Timing

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

wait mode. Some modules can be programmed to be active in wait mode.

Wait mode can also be exited by a reset or break. 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 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.

Figure 7-13 and **Figure 7-14** show the timing for WAIT recovery.



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

Figure 7-13. Wait Recovery from Interrupt or Break

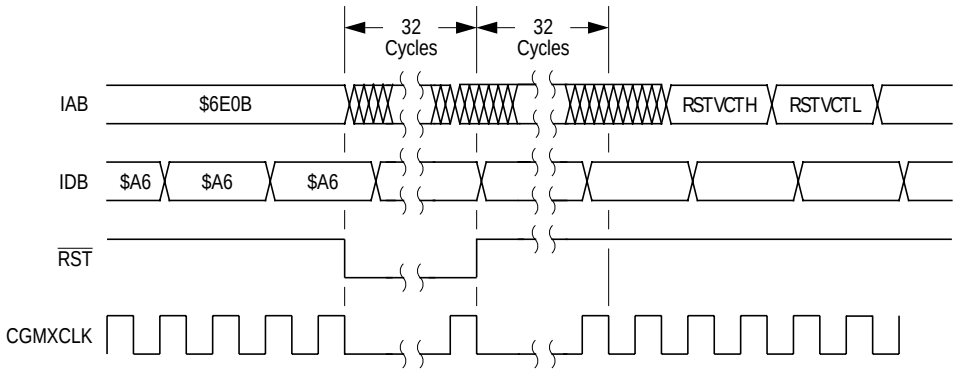


Figure 7-14. Wait Recovery from Internal Reset

7.7.2 Stop Mode

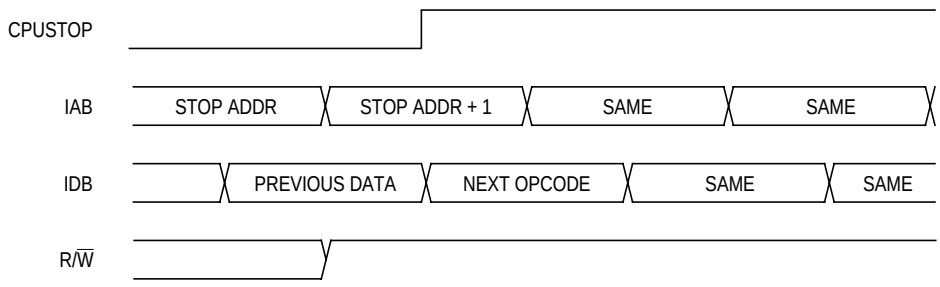
In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset or break also causes an exit from stop mode.

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 7-15** shows stop mode entry timing and **Figure 7-16** shows stop mode recovery timing from interrupt or break.



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

Figure 7-15. Stop Mode Entry Timing

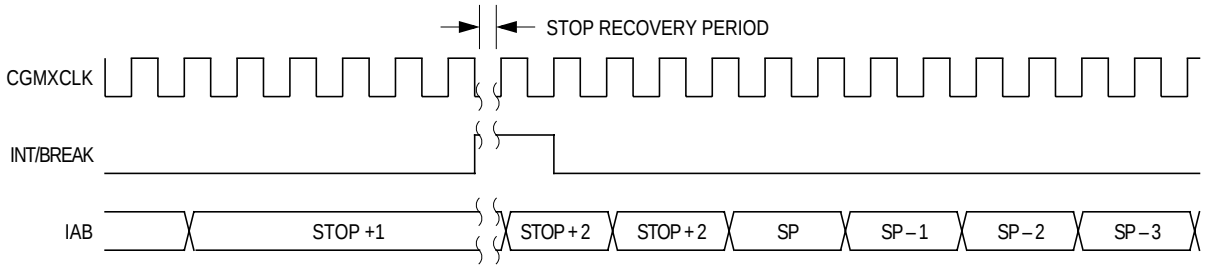


Figure 7-16. Stop Mode Recovery from Interrupt or Break

7.8 SIM Registers

The SIM has three memory mapped registers. **Table 7-3** shows the mapping of these registers.

Table 7-3. SIM Registers

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

7.8.1 SIM Break Status Register

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

Address: $FE00

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

R = Reserved Note 1. Writing a logic 0 clears SBSW.

Figure 7-17. 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 0 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 0 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, RETURN ; See if wait mode or stop mode 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

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.

Address: $FE01

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

= Unimplemented

Figure 7-18. SIM Reset Status Register (SRSR)

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

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

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

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

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

MOD — Module Reset Bit
- 1 = Last reset caused by one of the internal modules
- 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

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

Address: $FE03

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

R = Reserved

Figure 7-19. 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. Random-Access Memory (RAM)

8.1 Contents

8.2 Introduction

This section describes the 2.5 Kbytes of RAM.

8.3 Functional Description

Addresses $0050 through $0A4F 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 are 176 bytes of RAM. Because the location of the stack RAM is programmable, all page zero RAM locations can be used for I/O (input/output) control and user data or code. When the stack pointer is moved from its reset location at $00FF, direct addressing mode instructions can access efficiently all page zero RAM locations. Page zero RAM, therefore, provides ideal locations for frequently accessed global variables.

Before processing an interrupt, the CPU uses 5 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 9. 2-Kbyte FLASH Memory

9.1 Contents

9.2 Introduction

This section describes the operation of the embedded 2-Kbyte FLASH memory. This is non-volatile memory which can be read, programmed, and erased from a single external supply.

9.3 Functional Description

This FLASH memory array contains 2,048 bytes. An erased bit reads as a logic 0 and a programmed bit reads as a logic 1. Program and erase operations are facilitated through control bits in a memory mapped register. Details for these operations appear later in this section. Memory in the FLASH array is organized into pages and rows. There are eight pages of memory per row and for this array, 1 byte per page. The minimum erase block size is a single row, eight bytes. Programming is performed on a per page basis, or for this array, one byte at a time.

The address ranges for the 2-Kbyte FLASH memory are:

- $1800–$1FFF; general-purpose FLASH array

When programming the FLASH, just enough program time must be utilized via an iterative programming algorithm. Too much program time can result in a disturb condition in which an erased bit becomes programmed. This can be prevented as long as no more than eight program operations are performed per row before again performing an erase operation. Each programmed page is read in margin mode to ensure that the bits are programmed enough for data retention over device lifetime.

The row architecture for this array is:

- $1800–$1807; row 0
- $1808–$180F; row 1
- $1810–$1817; row 2
- ----------
- $1FF8–$1FFF; row 255

9.4 FLASH 3 Control Register

The FLASH control register (FL3CR) controls FLASH program, erase, and margin operations.

Address: $FE08

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	F3DIV1	F3DIV0	F3BLK1	F3BLK0	HVEN	MARG	ERASE	PGM
Reset:	0	0	0	0	0	0	0	0

Figure 9-1. FLASH 3 Control Register (FL3CR)

NOTE: *Devices with more than one FLASH have multiple control registers (FLCRs). Only one FLASH control register should be accessed at a time, so while accessing one control register, ensure that any others are cleared.*

F3DIV0 — Frequency Divide Control Bit

This bit selects the factor by which CGMVCLK is divided to derive the charge pump frequency. See **Table 9-2**. Note that F3DIV1 has no effect.

F3BLK1 and F3BLK0 — Block Erase Control Bits

These bits control erasing of blocks of varying size. **Table 9-1** shows the various block sizes which can be erased in one erase operation.

Table 9-1. Erase Block Sizes

BLK1	BLK0	Block Size	Row Boundaries
0	0	Full array: 2 Kbytes	0–255($1800–$1FFF)
0	1	One-half array: 1 Kbyte	0–127($1800–$1BFF) 128–255($1C00–$1FFF
1	0	Eight rows: 64 bytes	0–7($1800–$183F) 8–15($1840–$187F) 16–23($1880–$18BF) --- 248–255($1FC0–$1FFF)
1	1	Single row: 8 bytes	0($1800–$1807) 1($1808–$180F) ---- 255($1FF8–$1FFF)

In step 4 of the erase operation in section **9.8 FLASH Erase Operation**, the upper addresses are latched and used to determine the location of the block to be erased. For the full array, the only requirement is that the target address points to any byte in this array. Writing to any address in the array will enable the erase.

HVEN — High-Voltage Enable Bit

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

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

MARG — Program Margin Control Bit

This read/write bit configures the memory for a program margin operation. It cannot be set if the HVEN bit is set, and if it is set when HVEN is set, it will automatically return to 0.

1 = Margin operation selected
0 = Margin operation unselected

ERASE — Erase Control Bit

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

1 = Erase operation selected
0 = Erase operation unselected

PGM — Program Control Bit

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

1 = Program operation selected
0 = Program operation unselected

9.5 FLASH 3 Block Protect Register

The block protect register (FL3BPR) is implemented as an input/output (I/O) register. Each bit, when programmed, protects a range of addresses in the FLASH.

Address: $001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:								
Write:					BPR3	BPR2	BPR1	BPR0
Reset:	X	X	X	X	1	1	1	1

= Unimplemented X = Indeterminate

Figure 9-2. FLASH 3 Block Protect Register (FL3BPR)

BPR3 — Block Protect Register Bit 3

This bit protects the memory contents in the address range $1C00 to $1FFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

BPR2 — Block Protect Register Bit 2

This bit protects the memory contents in the address range $1A00 to $1FFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

BPR1 — Block Protect Register Bit 1

This bit protects the memory contents in the address range $1900 to $1FFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

BPR0 — Block Protect Register Bit 0

This bit protects the memory contents in the address range $1800 to $1FFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

By programming the block protect bits, a portion of the memory will be locked so that no further erase or program operations may be performed. Programming more than one bit at a time is redundant. If both BPR3 and BPR2 are set, for instance, the address range $1A00 through $1FFF is locked. If all bits are cleared, then all of the memory is available for erase and program.

9.6 Block Protection

Because of 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. This protection is done by reserving a location in the I/O space for block protect information. If the address range for an erase or program operation includes a protected block, the PGM or ERASE bit is cleared which prevents the HVEN bit in the FLASH control register from being set so that no high voltage is allowed in the array.

When the block protect register is cleared, the entire memory is accessible for program and erase. When bits within the register are programmed, they lock blocks of memory address ranges as shown in the **9.5 FLASH 3 Block Protect Register**.

9.7 Charge Pump Frequency Control

The internal charge pump for this array is to be operated over the specified frequency range (refer to **22.16 2-K FLASH Memory Electrical Characteristics**). The PLL output clock, CGMVCLK, is used to derive the two quadrature clocks, VCLK12 and VCLK23 which are one-half CGMVCLK. Additional pump frequency control is provided using the FDIV0 bit to keep the VCLKs within the specified range. The PLL must be ON and locked (but not necessarily engaged) before program/erase operations can be performed. See **Table 9-2**.

Table 9-2. Charge Pump Clock Frequency

FDIV0	Pump Clock Frequency
0	CGMVCLK ÷ 2
1	CGMVCLK ÷ 4

9.8 FLASH Erase Operation

NOTE: *After a total of eight program operations have been applied to a row, the row must be erased before further use to avoid the disturb condition. An erased byte will read $00.*

22.16 2-K FLASH Memory Electrical Characteristics has a detailed description of the times used in this algorithm. Use this step-by-step procedure to erase a block of FLASH memory:

1. Establish pump frequency by configuring PLL.
2. Unprotect target portion of the array (BPR0–BPR3.)
3. Set the ERASE bit, the BLK0, BLK1, and FDIV0 bit in the FLASH control register.
4. Write to any FLASH address with any data within the block address range desired.
5. Set the HVEN bit.
6. Wait for a time, t_{ERASE}.
7. Clear the HVEN bit.
8. Wait for a time, t_{Kill}, for the high voltages to dissipate.
9. Clear the ERASE bit.
10. After time, t_{HVD}, the memory can be accessed again in read mode.

NOTE: *These operations must be performed in the order as shown, but other unrelated operations may occur between the steps. Do not exceed t_{Erase} maximum.*

9.9 FLASH Program and Margin Read Operation

Programming of this FLASH array is done on a page basis where one page equals one byte. The purpose of the margin read mode is to ensure that data has been programmed with sufficient margin for long-term data retention. During a margin read, the control gates of the selected memory bits are held at a slightly negative voltage by an internal charge pump. Reading the data in margin mode is the same as for ordinary read mode except that a built-in counter stretches the data access for an additional eight cycles to allow sensing of the lower cell current. In short, a margin read applies a more stringent condition on the bitcell during read which ensures the data will be valid throughout the life of the product. A margin read can only follow a program operation. All times listed here are specified in **Section 22. Electrical Specifications**.

The procedure for programming the FLASH memory is:

1. Establish pump frequency by configuring the PLL.
2. Set the PGM bit and program FDIV0 to the appropriate value. This configures the memory for program operation and enables the latching of address and data for programming.
3. Write data to the page (1 byte) being programmed.
4. Set the HVEN bit.
5. Wait for a time, t_{STEP}.
6. Clear the HVEN bit.
7. Wait for a time, t_{HVTV}.
8. Set the MARG bit.
9. Wait for a time, t_{VTP}.
10. Clear the PGM bit.
11. Wait for a time, t_{HVD}.
12. Read the page of data. (This is in margin mode.)
13. Clear the MARG bit.
14. If any programmed bits do not read correctly, repeat the process from step 2 through 13 up to maximum program pulses. (See **22.16 2-K FLASH Memory Electrical Characteristics**.)

Note: This page program algorithm assumes the PLL is on and locked and the page in question has been erased before entry.

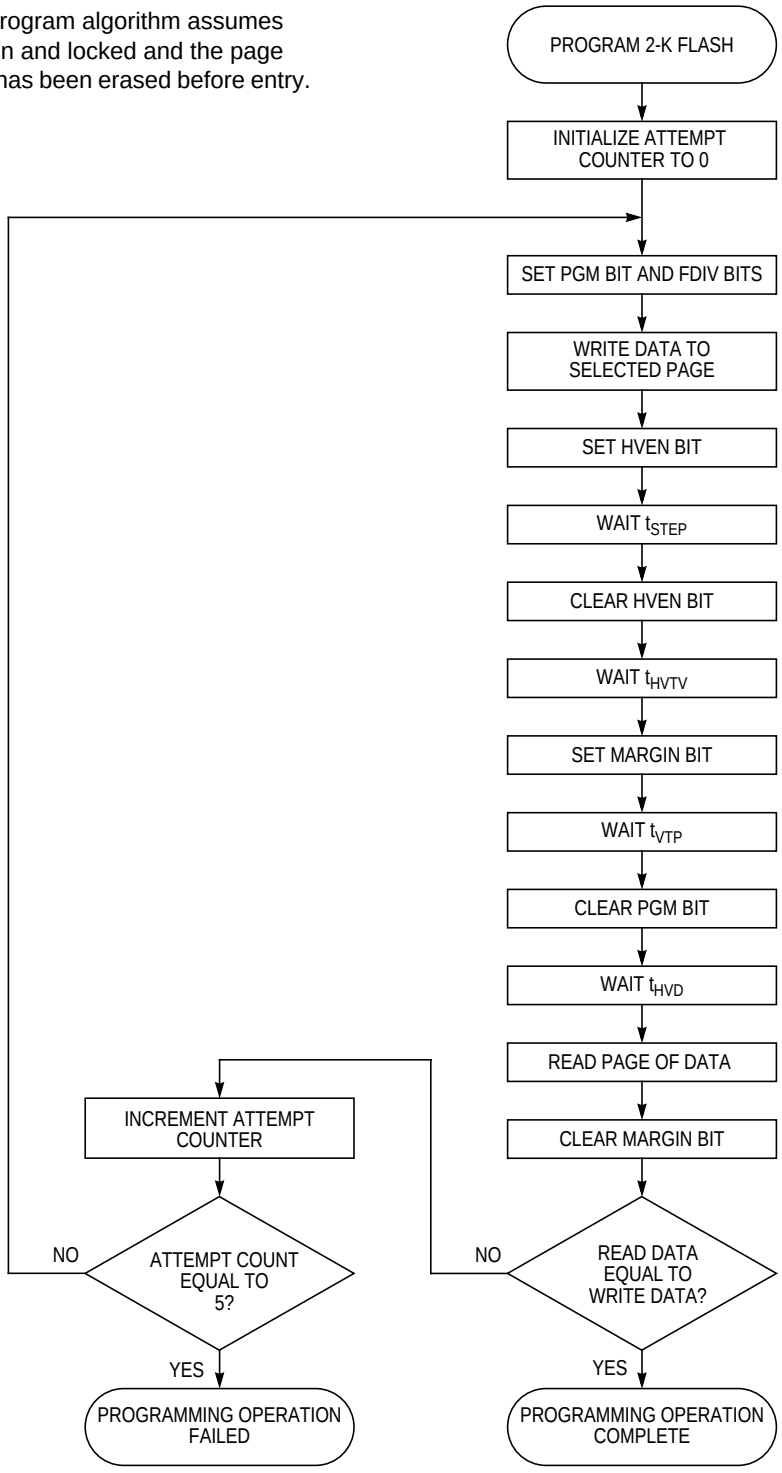


Figure 9-3. Page Program Algorithm

2-Kbyte FLASH Memory

Section 10. 48-Kbyte FLASH Memory

10.1 Contents

10.2 Introduction

This section describes the operation of the embedded 48-Kbyte FLASH memory. This is non-volatile memory which can be read, programmed, and erased from a single external supply.

10.3 Functional Description

The FLASH memory contains 48,676 bytes divided between two FLASH arrays. An erased bit reads as a logic 0 and a programmed bit reads as a logic 1. Program and erase operations are facilitated through control bits in a memory mapped register. Details for these operations appear later in this section. Memory in the FLASH array is organized into pages and rows. There are eight pages of memory per row and for this array, 8 bytes per page. The minimum erase block size is a single row, 64 bytes. Programming is performed on a per page basis, or for this array, eight bytes at a time. The address ranges for the 48-Kbyte FLASH memory are:

- $4000–$7FFF; FLASH 2 array
- $8000–$FDFF; FLASH 1 array
- $FFD4–$FFFF; user vector space and part of FLASH 1 array

When programming the FLASH, just enough program time must be utilized via an iterative programming algorithm. Too much program time can result in a disturb condition in which an erased bit becomes programmed. This can be prevented as long as no more than eight program operations are performed per row before again performing an erase operation. Each programmed page is read in margin mode to ensure that the bits are programmed enough for data retention over device lifetime.

The row architecture for this array is:

- $8000–$803F; row 0 of FLASH 1
- $8040–$807F; row 1 of FLASH 1
- $8080–$80CF; row 2 of FLASH 1
- ----------
- $FFC0–$FFFF; row 511 of FLASH 1

- $4000–$403F; row 0 of FLASH 2
- $4040–$407F; row 1 of FLASH 2
- $4080–$40BF; row 2 of FLASH 2
- ----------
- $7FC0–$7FFF; row 255 of FLASH 2

10.4 FLASH 1 Control Register

The FLASH 1 control register (FL1CR) controls FLASH program, erase, and margin operations.

Address: $FE0C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	F1DIV1	F1DIV0	F1BLK1	F1BLK0	HVEN	MARG	ERASE	PGM
Reset:	0	0	0	0	0	0	0	0

Figure 10-1. FLASH 1 Control Register (FL1CR)

10.5 FLASH 2 Control Register

The FLASH 2 control register (FL2CR) controls FLASH program, erase, and margin operations.

Address: $FE0A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	F2DIV1	F2DIV0	F2BLK1	F2BLK0	HVEN	MARG	ERASE	PGM
Reset:	0	0	0	0	0	0	0	0

Figure 10-2. FLASH 2 Control Register (FL2CR)

NOTE: *Devices with more than one FLASH have multiple control registers (FLCRs). Only one FLASH control register should be accessed at a time, so while accessing one control register, ensure that any others are cleared.*

F1DIV0, F2DIV0 — Frequency Divide Control Bit

These bits are logically ORed together and the output selects the factor by which CGMVCLK is divided to derive the charge pump frequency. See **Table 10-3**. Note that F1DIV1 and F2DIV1 have no effect.

F1BLK1, F1BLK0, F2BLK1, F2BLK0— Block Erase Control Bits

These bits control erasing of blocks of varying size. **Table 10-1** and **Table 10-2** show the various block sizes which can be erased in one erase operation.

Table 10-1. 32-Kbyte Erase Block Sizes

F1BLK1	F1BLK0	Block Size	Row Boundaries
0	0	Full array: 32 Kbytes	0–511 ($8000–$FFFF)
0	1	One-half array: 16 Kbytes	0–255 ($8000–$BFFF) 256–511 ($C000–$FFFF)
1	0	Eight rows: 512 bytes	0–7 ($8000–$81FF) 8–15 ($8200–$83FF) 16–23 ($8400–$86FF) --- 504–511 ($FE00–$FFFF)
1	1	Single row: 64 bytes	0 ($8000–$803F) 1 ($8040–$807F) ---- 511 ($FFC0–$FFFF)

Table 10-2. 16-Kbyte Erase Block Sizes

F2BLK1	F2BLK0	Block Size	Row Boundaries
0	0	Full array: 16 Kbytes	0–255 ($4000–$7FFF)
0	1	One half array: 8 Kbytes	0–127 ($4000–$5FFF) 128–255 ($6000–$7FFF)
1	0	Eight rows: 512 bytes	0–7 ($4000–$41FF) 8–15 ($4200–$43FF) 16–23 ($4400–$45FF) --- 248–255 ($7E00–$7FFF)
1	1	Single row: 64 bytes	0 ($4000–$403F) 1 ($4040–$407F) ---- 255 ($7FC0–$7FFF)

In step 4 of the erase operation in **10.10 FLASH Erase Operation**, the upper addresses are latched and used to determine the location of the block to be erased. For the full array, the only requirement is that the target address points to any byte in this array. Writing to any address in the array will enable the erase.

HVEN — High-Voltage Enable Bit

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

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

MARG — Program Margin Control Bit

This read/write bit configures the memory for a program margin operation. It cannot be set if the HVEN bit is set, and if it is set when HVEN is set, it will return to 0 automatically.

1 = Margin operation selected
0 = Margin operation unselected

ERASE — Erase Control Bit

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

1 = Erase operation selected
0 = Erase operation unselected

PGM — Program Control Bit

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

1 = Program operation selected
0 = Program operation unselected

10.6 FLASH 1 Block Protect Register

The FLASH 1 block protect register (FL1BPR) is implemented as an input/output (I/O) register. Each bit, when programmed, protects a range of addresses in the FLASH.

Address: $001F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:								
Write:					F1BPR3	F1BPR2	F1BPR1	F1BPR0
Reset:	X	X	X	X	1	1	1	1

= Unimplemented X = Indeterminate

Figure 10-3. FLASH 1 Block Protect Register (FL1BPR)

F1BPR3 — Block Protect Register Bit 3

This bit protects the memory contents in the address range $C000 to $FFFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

F1BPR2 — Block Protect Register Bit 2

This bit protects the memory contents in the address range $A000 to $FFFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

F1BPR1 — Block Protect Register Bit 1

This bit protects the memory contents in the address range $9000 to $FFFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

F1BPR0 — Block Protect Register Bit 0

This bit protects the memory contents in the address range $8000 to $FFFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

By programming the block protect bits, a portion of the memory will be locked so that no further erase or program operations may be performed. Programming more than one bit at a time is redundant. If both F1BPR3 and F1BPR2 are set, for instance, the address range $A000 through $FFFF is locked. If all bits are cleared, then all of the memory is available for erase and program.

10.7 FLASH 2 Block Protect Register

The FLASH 2 block protect register (FL2BPR) is implemented as an I/O register. Each bit, when programmed, protects a range of addresses in the FLASH.

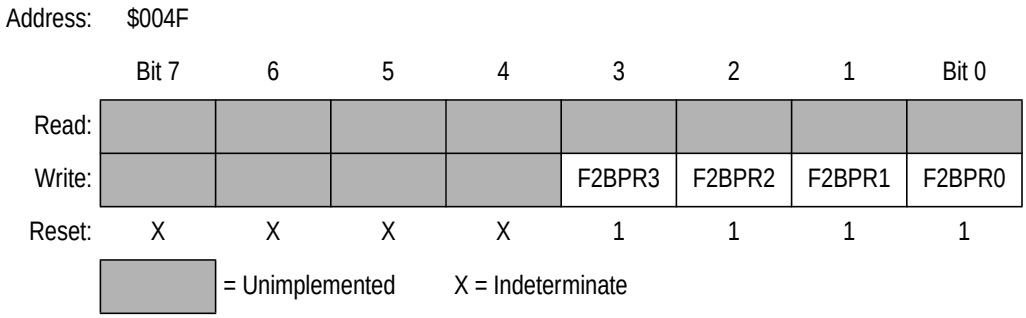


Figure 10-4. FLASH 2 Block Protect Register (FL2BPR)

F2BPR3 — Block Protect Register Bit 3

This bit protects the memory contents in the address range $6000 to $7FFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

F2BPR2 — Block Protect Register Bit 2

This bit protects the memory contents in the address range $5000 to $7FFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

F2BPR1 — Block Protect Register Bit 1

This bit protects the memory contents in the address range $4800 to $7FFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

F2BPR0 — Block Protect Register Bit 0

This bit protects the memory contents in the address range $4000 to $7FFF.

1 = Address range protected from erase or program
0 = Address range open to erase or program

By programming the block protect bits, a portion of the memory will be locked so that no further erase or program operations may be performed. Programming more than one bit at a time is redundant. If both F2BPR3 and F2BPR2 are set, for instance, the address range $5000 through $7FFF is locked. If all bits are cleared, then all of the memory is available for erase and program.

10.8 Block Protection

Because of 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. This protection is done by reserving a location in the I/O space for block protect information. If the address range for an erase or program operation includes a protected block, the PGM or ERASE bit is cleared which prevents the HVEN bit in the FLASH control register from being set so that no high voltage is allowed in the array.

When the block protect register is cleared, the entire memory is accessible for program and erase. When bits within the register are programmed, they lock blocks of memory address ranges as shown in **10.7 FLASH 2 Block Protect Register**.

10.9 Charge Pump Frequency Control

The internal charge pump for this array is to be operated over the specified frequency range (refer to **22.16 2-K FLASH Memory Electrical Characteristics**). The PLL output clock, CGMVCLK, is used to derive the two quadrature clocks, VCLK12 and VCLK23 which are one-half CGMVCLK. Additional pump frequency control is provided using the FxDIV0 bit in order to keep the VCLKs within the specified range. The PLL must be ON and Locked (but not necessarily engaged) before program/erase operations can be performed.

Table 10-3. Charge Pump Clock Frequency

FDIV0	Pump Clock Frequency
0	CGMVCLK ÷ 2
1	CGMVCLK ÷ 4

10.10 FLASH Erase Operation

NOTE: *After a total of eight program operations have been applied to a row, the row must be erased before further use to avoid the disturb condition. An erased byte will read $00.*

Section 22. Electrical Specifications has a detailed description of the times used in this algorithm.

Use this step-by-step procedure to erase a block of FLASH memory:

1. Establish pump frequency by configuring PLL.
2. Unprotect target portion of the array (FxBPR0–FxBPR3).
3. Set the ERASE bit, the FxBLK0, FxBLK1, and FxDIV0 bits in the FLASH control register.
4. Write to any FLASH address with any data within the block address range desired.
5. Set the HVEN bit.
6. Wait for a time, t_{Erase}.

7. Clear the HVEN bit.
8. Wait for a time, t_{Kill}, for the high voltages to dissipate.
9. Clear the ERASE bit.
10. After a time, t_{HVD}, the memory can be accessed in read mode again.

NOTE: *These operations must be performed in the order shown, but other unrelated operations may occur between the steps. Do not exceed t_{Erase} maximum.*

10.11 FLASH Program and Margin Read Operation

Programming of this FLASH array is done on a page basis where one page equals eight bytes. The purpose of the margin read mode is to ensure that data has been programmed with sufficient margin for long-term data retention. During a margin read, the control gates of the selected memory bits are held at a slightly negative voltage by an internal charge pump. Reading the data in margin mode is the same as for ordinary read mode except that a built-in counter stretches the data access for an additional eight cycles to allow sensing of the lower cell current. In short, a margin read applies a more stringent condition on the bitcell during read which ensures the data will be valid throughout the life of the product. A margin read can only follow a program operation. All times listed here are specified in **Section 22. Electrical Specifications**.

The step-by-step procedure for programming the FLASH memory is:

1. Establish pump frequency by configuring the PLL.
2. Set the PGM bit and program FxDIV0 appropriately. This configures the memory for program operation and enables the latching of address and data for programming.
3. Write data to the page (8 bytes) being programmed.
4. Set the HVEN bit.
5. Wait for a time, $t_{Step.}$
6. Clear the HVEN bit.

7. Wait for a time, t_{HVTV}.
8. Set the MARG bit.
9. Wait for a time, t_{VTP}.
10. Clear the PGM bit.
11. Wait for a time, t_{HVD}.
12. Read the page of data. (This is in margin mode.)
13. Clear the MARG bit.
14. If any programmed bits do not read correctly, repeat the process from step 2 through 13 up to maximum program pulses. (See **22.16 2-K FLASH Memory Electrical Characteristics**.)

Note: This page program algorithm assumes the PLL is on and locked, and the page in question has been erased before entry.

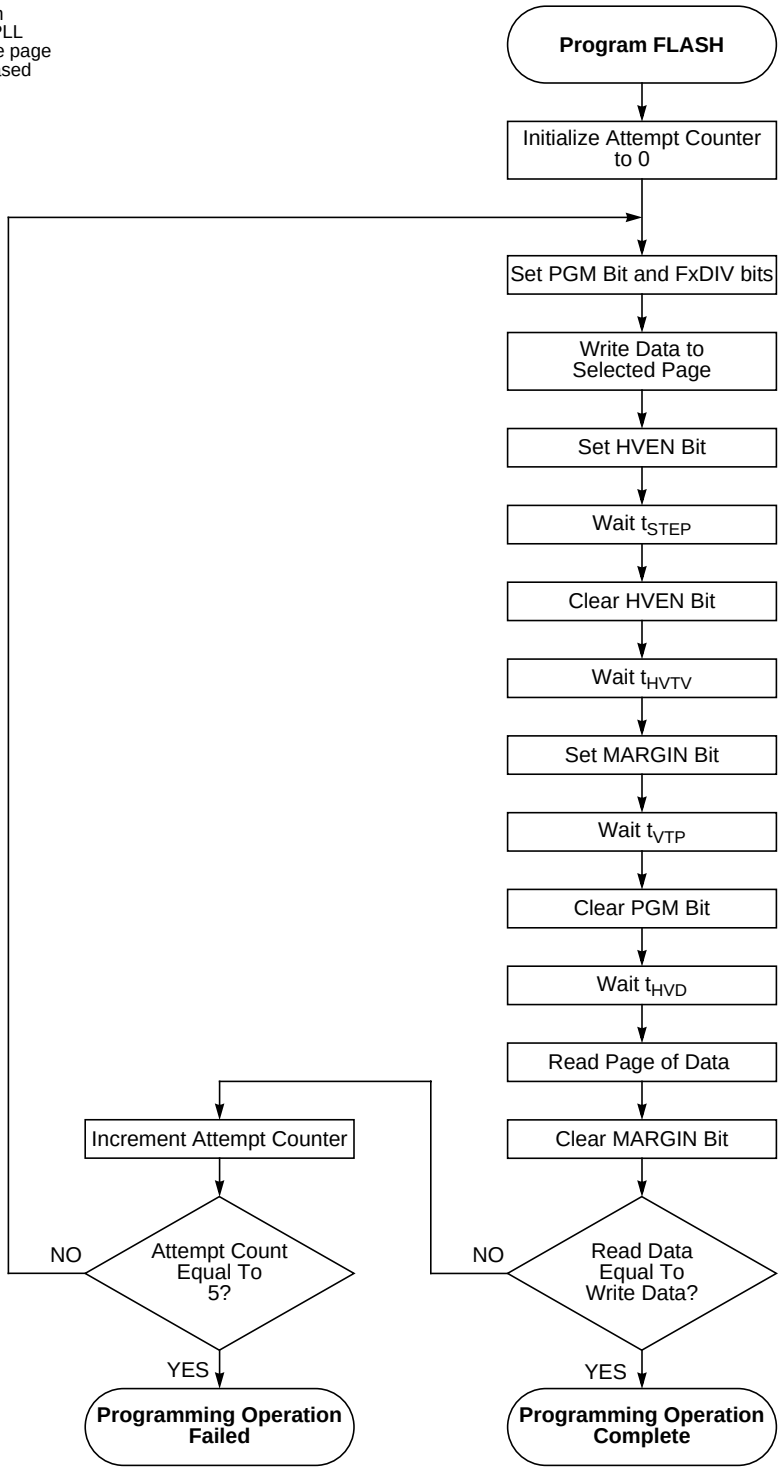


Figure 10-5. Page Program Algorithm

Section 11. Serial Peripheral Interface (SPI) Module

11.1 Contents

11.2 Introduction

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

11.3 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
- Clock ground for reduced radio frequency (RF) interference
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts:
  - 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

11.4 Functional Description

Figure 11-1 shows the structure of the SPI module and **Figure 11-2** shows the locations and contents of the SPI input/output (I/O) registers.

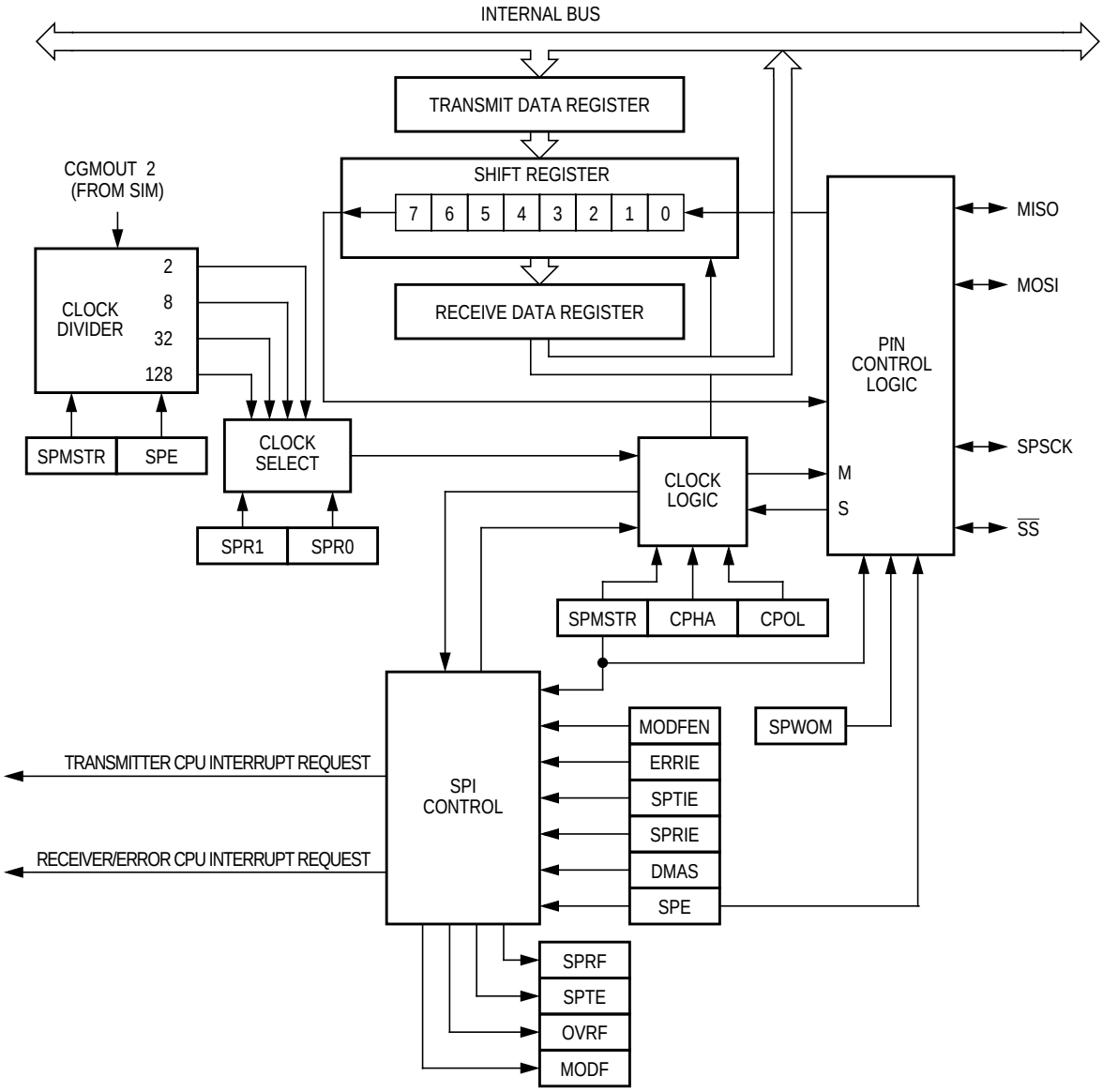


Figure 11-1. SPI Module Block Diagram

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

The following subsections describe the operation of the SPI module.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$000F	SPI Control Register (SPCR)	Read: Write:	SPRIE	DMAS	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Reset:	0	0	1	0	1	0	0	0
$0010	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
$0011	SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Indeterminate after reset							

= Unimplemented

Figure 11-2. SPI I/O Register Summary

11.4.1 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 **11.14.1 SPI Control Register**.)*

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the transmit data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The byte begins shifting out on the MOSI pin under the control of the serial clock. (See **Figure 11-3**.)

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See **11.14.2 SPI Status and Control Register**.) Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

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.

When the DMAS bit is set, the SPI status and control register does not have to be read to clear the SPRF bit. A read of the SPI data register by either the CPU or the DMA clears the SPRF bit. A write to the SPI data register clears the SPTE bit.

NOTE: *This device has no DMA. DMAS should be cleared.*

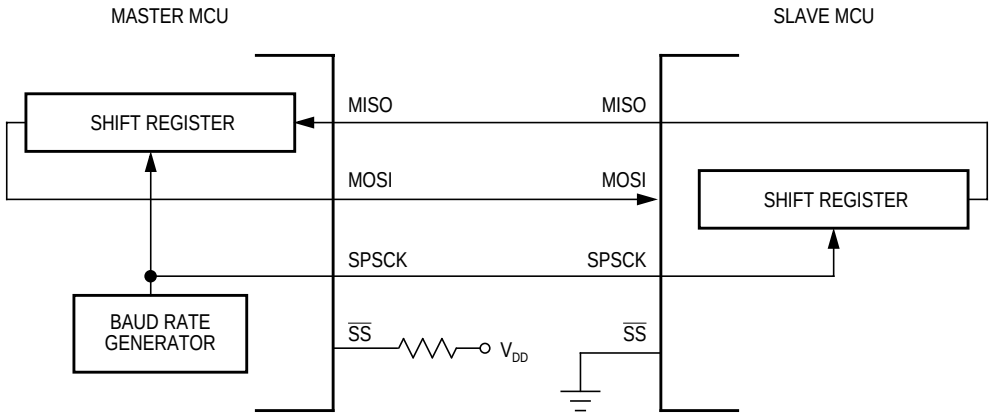


Figure 11-3. Full-Duplex Master-Slave Connections

11.5 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 0. $\overline{SS}$ must remain low until the transmission is complete. (See **11.8.2 Mode Fault Error**.)

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

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

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

When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transmission. (See **11.6 Transmission Formats**.)

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

11.6 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 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.

11.6.1 Clock Phase and Polarity Controls

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

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

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

11.6.2 Transmission Format When CPHA = 0

Figure 11-4 shows an SPI transmission in which CPHA is logic 0. The figure should not be used as a replacement for data sheet parametric information.Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line

is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See **11.8.2 Mode Fault Error**.) When CPHA = 0, the first SPSCK edge is the MSB (most significant bit) capture strobe. Therefore the slave must begin driving its data before the first SPSCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the slave data transmission. The slave's $\overline{SS}$ pin must be toggled back to high and then low again between each byte transmitted as shown in **Figure 11-5**.

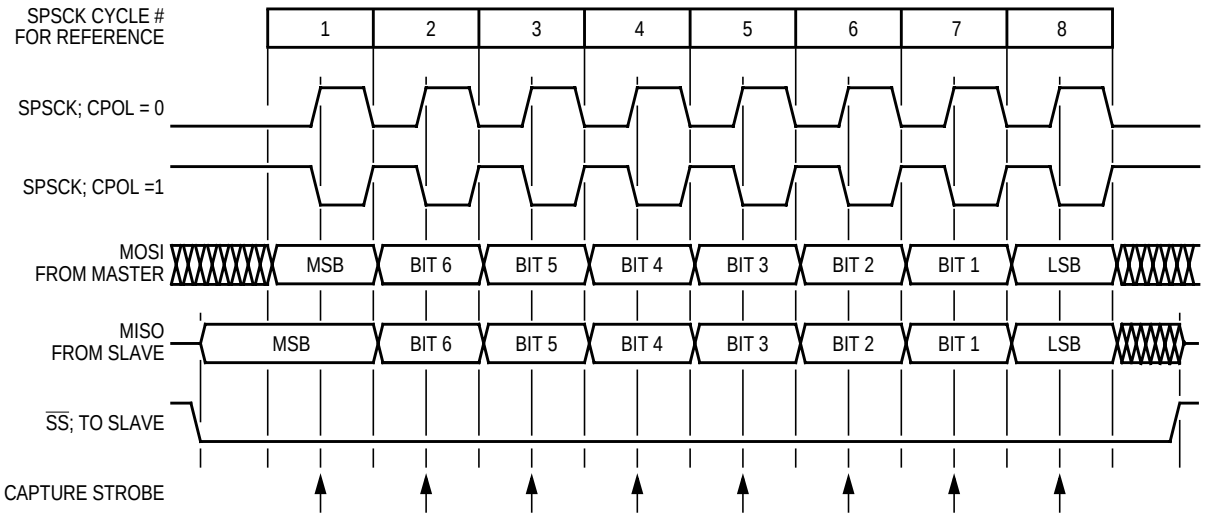


Figure 11-4. Transmission Format (CPHA = 0)

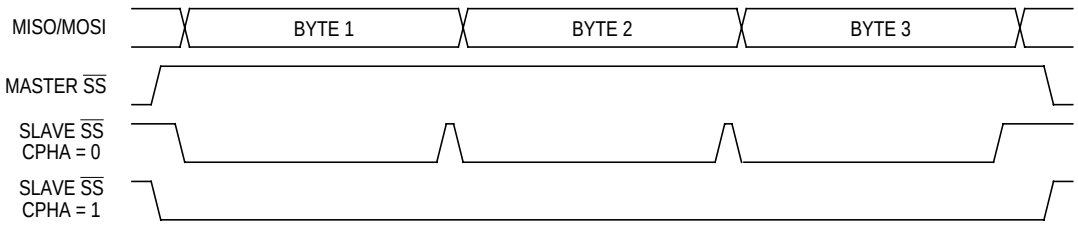


Figure 11-5. CPHA/$\overline{SS}$ Timing

When CPHA = 0 for a slave, the falling edge of $\overline{SS}$ indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. Once the transmission begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the transmit data register and transferred to the shift register after the current transmission.

11.6.3 Transmission Format When CPHA = 1

Figure 11-6 shows an SPI transmission in which CPHA is logic 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See **11.8.2 Mode Fault Error**.) When CPHA = 1, the master begins driving its MOSI pin on the first SPSCK edge. Therefore, the slave uses the first SPSCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

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

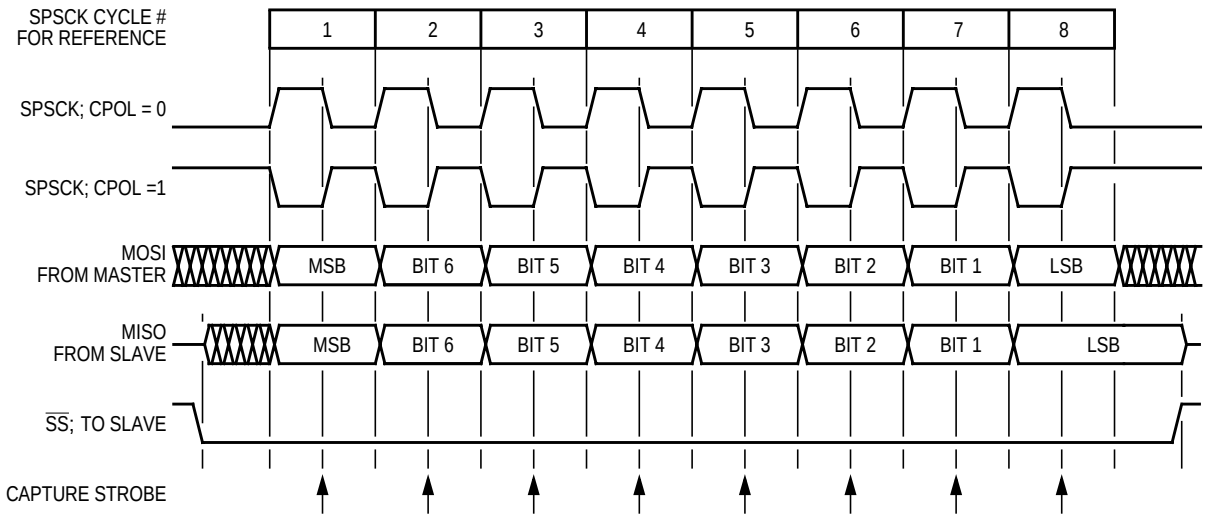


Figure 11-6. Transmission Format (CPHA = 1)

11.6.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), writing to the SPDR starts a transmission. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SPSCK signal. When CPHA = 0, the SPSCK signal remains inactive for the first half of the first SPSCK cycle. When CPHA = 1, the first SPSCK cycle begins with an edge on the SPSCK line from its inactive to its active level. The SPI clock rate (selected by SPR1:SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See **Figure 11-7**.) The internal SPI clock in the master is a free-running derivative of the internal MCU clock. To conserve power, it is enabled only when both the SPE and SPMSTR bits are set. SPSCK 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 occurs relative to the slower SPSCK. This uncertainty causes the variation in the initiation delay shown in **Figure 11-7**. This delay is no longer than a single SPI bit time. That is, the maximum delay is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

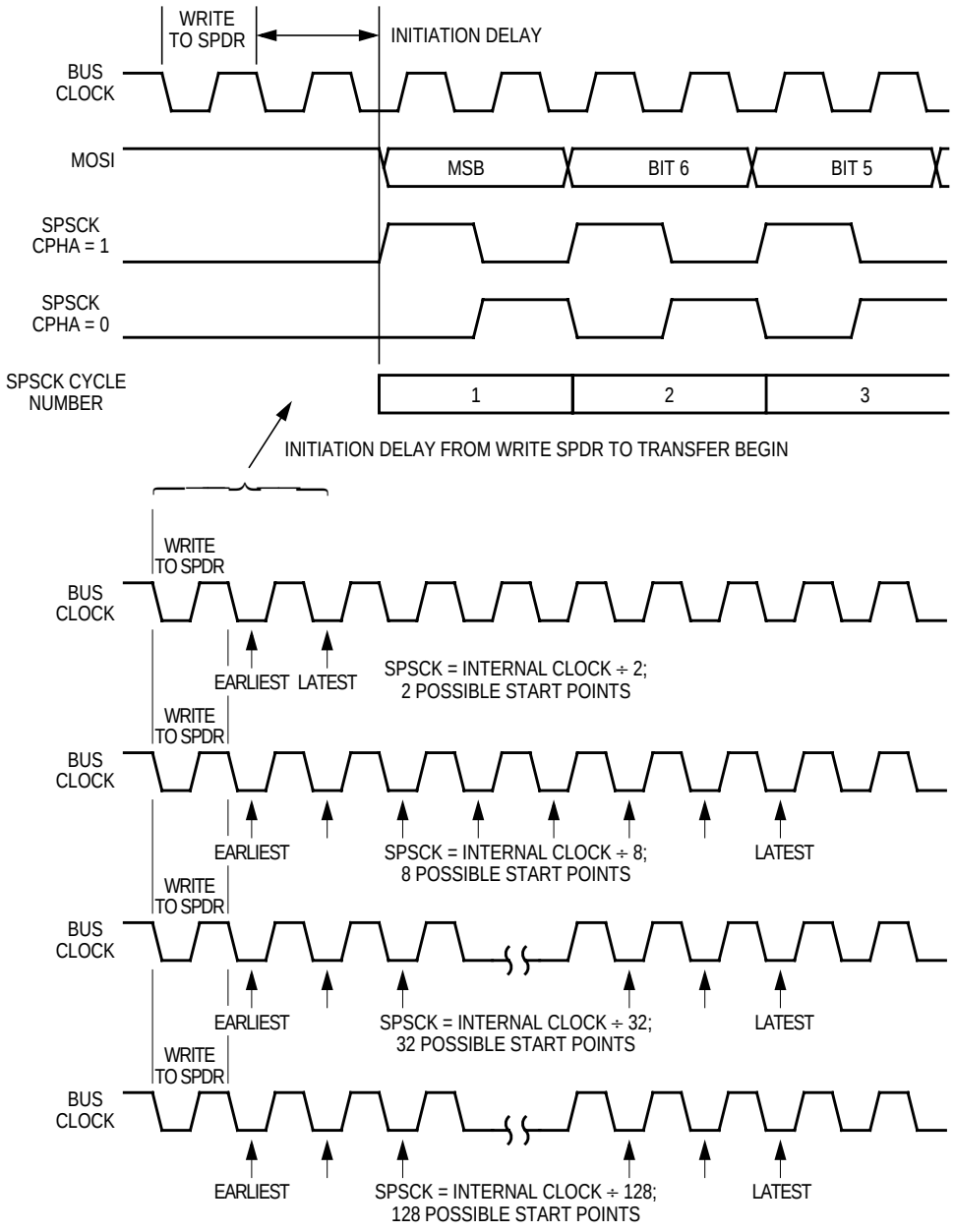


Figure 11-7. Transmission Start Delay (Master)

11.7 Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the transmit data register only when the SPTE bit is high. **Figure 11-8** shows the timing associated with doing back-to-back transmissions with the SPI. (SPSCK has CPHA:CPOL = 1:0.)

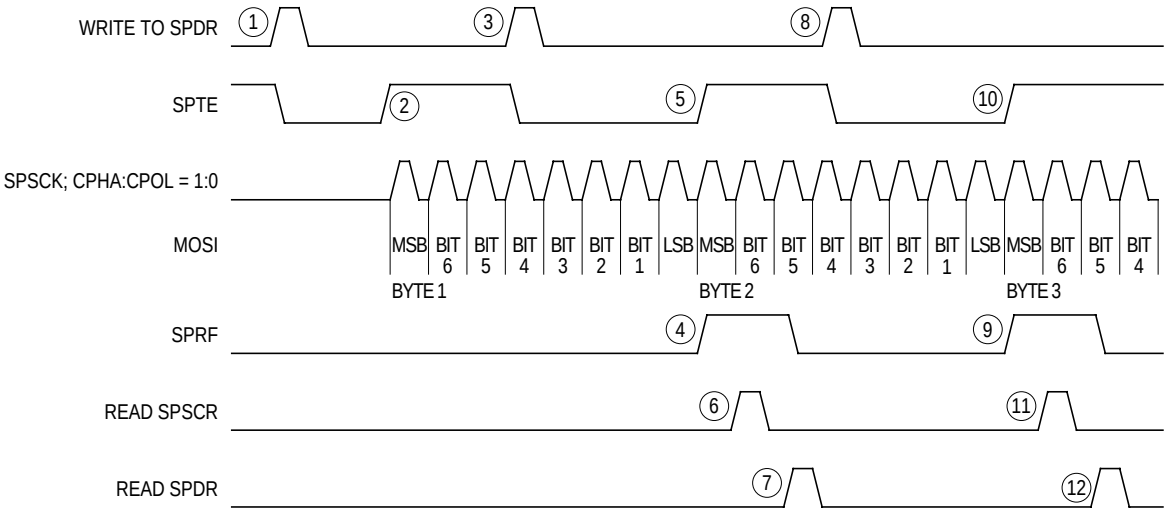


Figure 11-8. SPRF/SPTE CPU Interrupt Timing

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

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

11.8 Error Conditions

These flags signal SPI error conditions:

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

11.8.1 Overflow Error

The overflow flag (OVRF) becomes set if the receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. The bit 1 capture strobe occurs in the middle of SPSCK cycle 7. (See **Figure 11-4** and **Figure 11-6**.) If an overflow occurs, all data received after the overflow and before the OVRF bit is cleared does not transfer to the receive data register and does not set the SPI receiver full bit (SPRF). The unread data that transferred to the receive data register before the overflow occurred can

still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPI status and control register and then reading the SPI data register.

NOTE: *This device has no DMA. DMAS should be cleared.*

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. When the DMAS bit is low, the SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. When the DMAS bit is high, SPRF generates a receiver DMA service request, and MODF and OVRF can generate a receiver/error CPU interrupt request. (See **Figure 11-11**.) 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 11-9** shows how it is possible to miss an overflow. The first part of **Figure 11-9** shows how it is possible to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF bit can be set in between the time that SPSCR and SPDR are read.

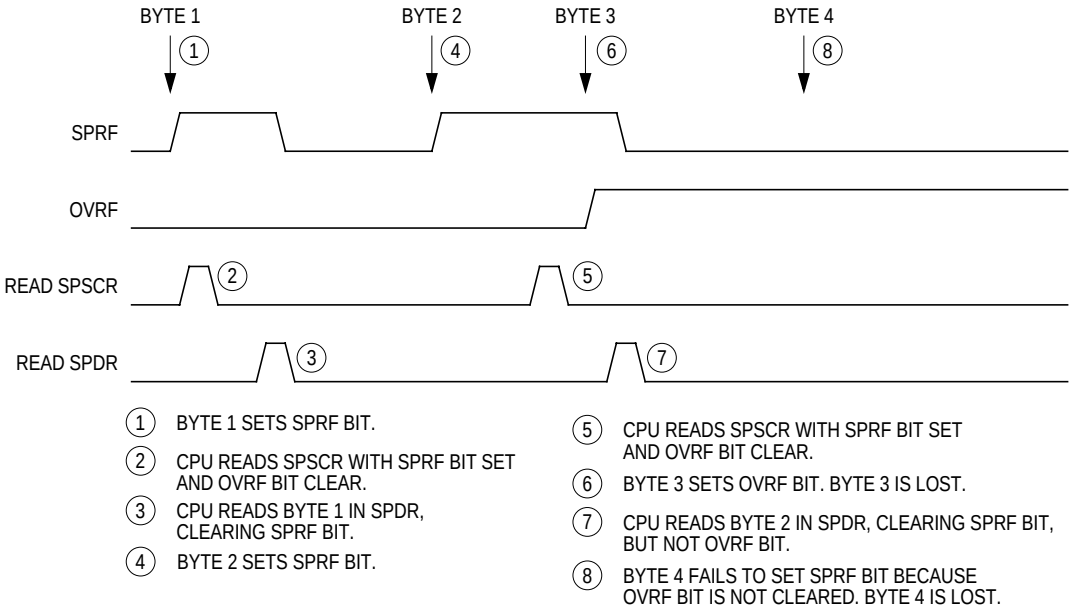


Figure 11-9. Missed Read of Overflow Condition

In this case, an overflow can easily be missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it is not obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR following the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions can set the SPRF bit. **Figure 11-10** illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit.

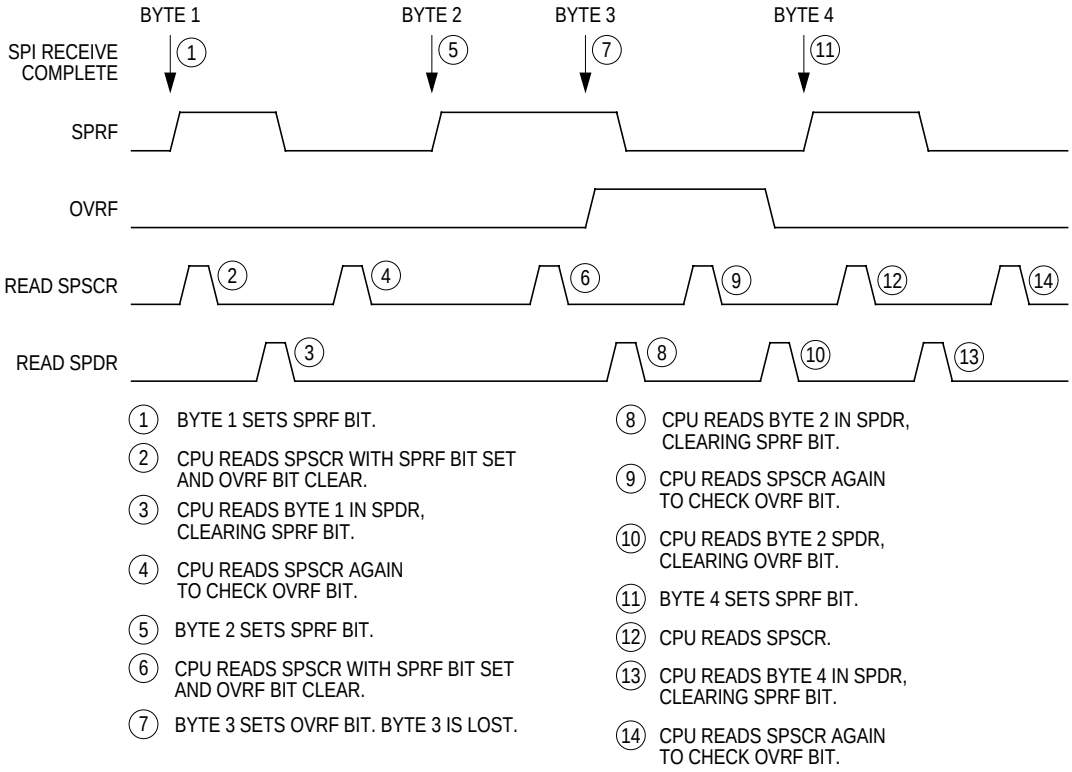


Figure 11-10. Clearing SPRF When OVRF Interrupt Is Not Enabled

11.8.2 Mode Fault Error

Setting the SPMSTR bit selects master mode and configures the SPSCK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SPSCK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, becomes set any time the state of the slave select pin, $\overline{SS}$, is inconsistent with the mode selected by SPMSTR. To prevent SPI pin contention and damage to the MCU, a mode fault error occurs if:

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

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

NOTE: *This device has no DMA. DMAS should be cleared.*

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. When the DMAS bit is low, the SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. When the DMAS bit is high, SPRF generates a receiver DMA service request instead of a CPU interrupt request, but MODF and OVRF can generate a receiver/error CPU interrupt request. (See **Figure 11-11**.) 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 0. A mode fault in a master SPI causes these 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 **11.6 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 1 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.*

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

11.9 Interrupts

Four SPI status flags can be enabled to generate CPU interrupt requests, see **Table 11-1**.

Table 11-1. SPI Interrupts

Flag	Request
SPTE Transmitter empty	SPI transmitter CPU interrupt request DMAS = 0, SPTIE = 1, SPE = 1 SPI transmitter DMA service request DMAS = 1, SPTIE = 1, SPE = 1
SPRF Receiver full	SPI receiver CPU interrupt request DMAS = 0, SPRIE = 1 SPI receiver DMA service request DMAS = 1, SPRIE = 1
OVRF Overflow	SPI receiver/error interrupt request ERRIE = 1
MODF Mode fault	SPI receiver/error interrupt request ERRIE = 1

NOTE: *This device has no DMA. DMAS should be cleared.*

The DMA select bit (DMAS) controls whether SPTE and SPRF generate CPU interrupt requests or DMA service requests. When DMAS = 0, reading the SPI status and control register with SPRF set and then reading the receive data register clears SPRF. When DMAS = 1, any read of the receive data register clears the SPRF flag. The clearing mechanism for the SPTE flag is always just a write to the transmit data register.

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

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

The error interrupt enable bit (ERRIE) enables both the MODF and OVRF bits 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 bit is enabled by the ERRIE bit to generate receiver/error CPU interrupt requests.

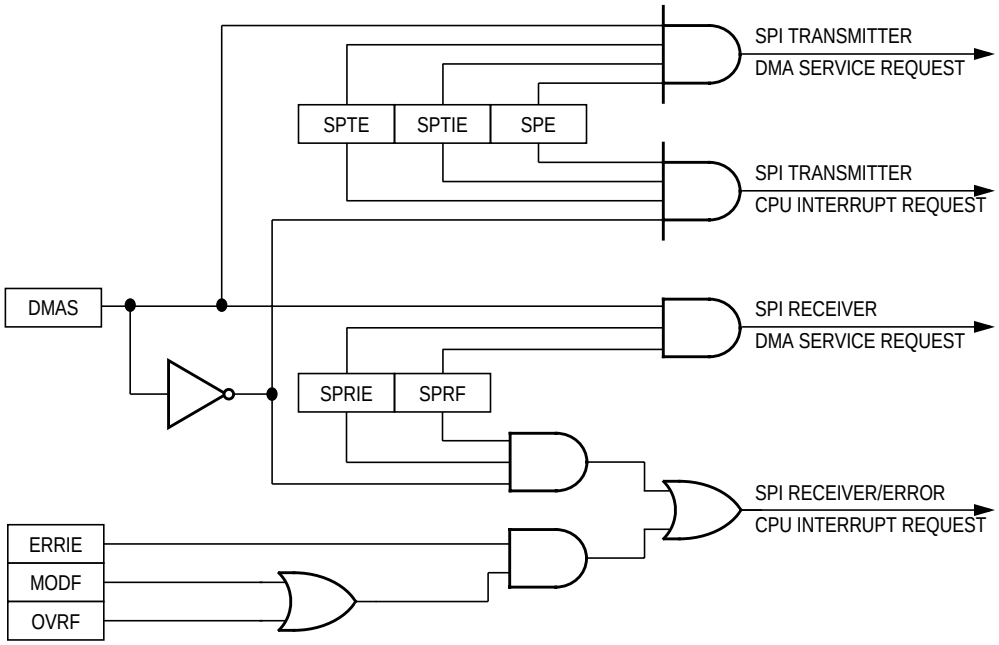


Figure 11-11. SPI Interrupt Request Generation

NOTE: *This device has no DMA. DMAS should be cleared.*

The following two sources in the SPI status and control register can generate interrupt 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 can generate either an SPI receiver/error CPU interrupt request or an SPRF DMA service request.

  If the DMA select bit, DMAS, is clear, SPRF generates an SPRF CPU interrupt request. If DMAS is set, SPRF generates an SPRF DMA service 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 can generate either an SPTE CPU interrupt request or an SPTE DMA service request.

  If the DMAS bit is clear, SPTE generates an SPTE CPU interrupt request. If DMAS is set, SPTE generates an SPTE DMA service request.

11.10 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 disabled.

These items are reset only by a system reset:

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

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

11.11 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.

11.12 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 **7.8.3 SIM Break Flag Control Register**.)

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

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

Since the SPTE bit cannot be cleared during a break with the BCFE bit cleared, a write to the transmit data register in break mode does not initiate a transmission nor is this data transferred into the shift register. Therefore, a write to the SPDR in break mode with the BCFE bit cleared has no effect.

11.13 I/O Signals

The SPI module has five I/O (input/output) pins:

- MISO — Data received
- MOSI — Data transmitted
- SPSCK — Serial clock
- $\overline{SS}$ — Slave select
- CGND — Clock ground

The SPI has limited inter-integrated circuit (I^2C) capability (requiring software support) as a master in a single-master environment. To communicate with I^2C peripherals, MOSI becomes an open-drain output when the SPWOM bit in the SPI control register is set. In I^2C communication, the MOSI and MISO pins are connected to a bidirectional pin from the I^2C peripheral and through a pullup resistor to V_{DD}.

11.13.1 MISO (Master In/Slave Out)

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

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is logic 0 and its $\overline{SS}$ pin is at logic 0. To support a multiple-slave system, a logic 1 on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

11.13.2 MOSI (Master Out/Slave In)

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

11.13.3 SPSCK (Serial Clock)

The serial clock synchronizes data transmission between master and slave devices. In a master MCU, the SPSCK pin is the clock output. In a slave MCU, the SPSCK pin is the clock input. In full duplex operation, the master and slave MCUs exchange a byte of data in eight serial clock cycles.

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

11.13.4 $\overline{SS}$ (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 **11.6 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 between transmissions for the CPHA = 1 format. See **Figure 11-12**.

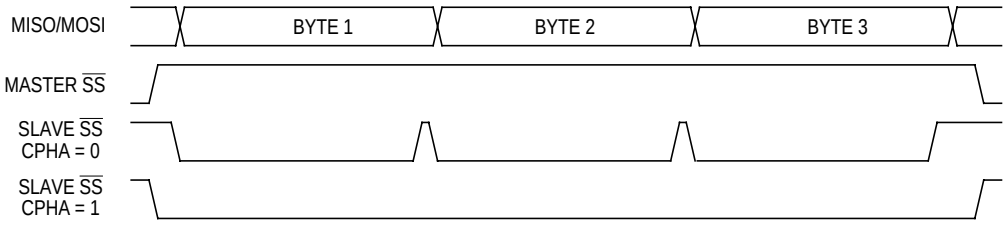


Figure 11-12. CPHA/$\overline{SS}$ Timing

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 **11.14.2 SPI Status and Control Register**.)

NOTE: *A logic 1 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 SPSCK 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 SPSCK. (See **11.8.2 Mode Fault Error**.) For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is low for an SPI master, the $\overline{SS}$ 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 port data register. (See **Table 11-2**.)

Table 11-2. 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

1. X = don't care

11.13.5 CGND (Clock Ground)

CGND is the ground return for the serial clock pin, SPSCK, and the ground for the port output buffers. It is connected to the EV_{SS1} pad.

11.14 I/O Registers

Three I/O registers control and monitor SPI operation:

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

11.14.1 SPI Control Register

The SPI control register (SPCR):

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

Address: $000F

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

Figure 11-13. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

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

1 = SPRF CPU interrupt requests or SPRF DMA service requests enabled
0 = SPRF CPU interrupt requests or SPRF DMA service requests disabled

DMAS —DMA Select Bit

This read/write bit selects DMA service requests when the SPI receiver full bit, SPRF, or the SPI transmitter empty bit, SPTE, becomes set. Setting the DMAS bit disables SPRF CPU interrupt requests and SPTE CPU interrupt requests.

1 = SPRF DMA and SPTE DMA service requests enabled (SPRF CPU and SPTE CPU interrupt requests disabled)
0 = SPRF DMA and SPTE DMA service requests disabled (SPRF CPU and SPTE CPU interrupt requests enabled)

NOTE: *This device has no DMA. DMAS should be cleared.*

SPMSTR — SPI Master Bit

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

1 = Master mode
0 = Slave mode

CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the SPSCK pin between transmissions. (See **Figure 11-4** and **Figure 11-6**.) To transmit data between SPI modules, the SPI modules must have identical CPOL values.

CPHA — Clock Phase Bit

This read/write bit controls the timing relationship between the serial clock and SPI data. (See **Figure 11-4** and **Figure 11-6**.) 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 1 between bytes. (See **Figure 11-12**.)

SPWOM — SPI Wired-OR Mode Bit

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

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

SPE — SPI Enable

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. (See **11.10 Resetting the SPI**.)

1 = SPI module enabled
0 = SPI module disabled

SPTIE — SPI Transmit Interrupt Enable

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

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

11.14.2 SPI Status and Control Register

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

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

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

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

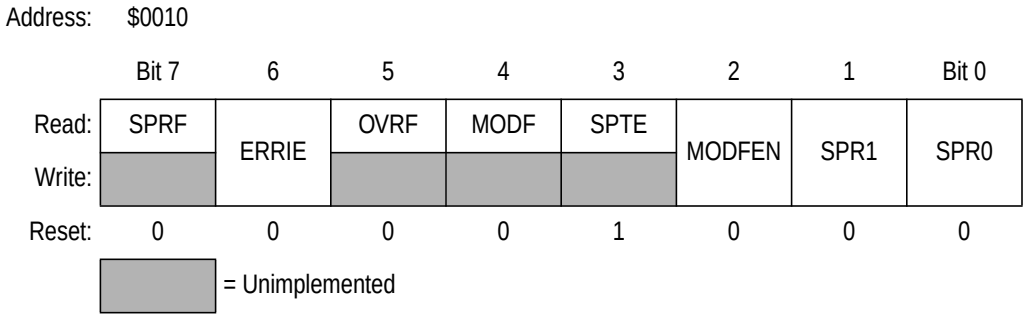


Figure 11-14. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

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

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

ERRIE — Error Interrupt Enable Bit

This read/write bit enables the MODF and OVRF bits to generate 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 full byte enters the shift register. In an overflow condition, the byte already in the receive data register is unaffected, and the byte that shifted in last is lost. Clear the OVRF bit by reading the SPI status and control register with OVRF set and then reading the receive data register.

1 = Overflow
0 = No overflow

MODF — Mode Fault Bit

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

1 = $\overline{SS}$ pin at inappropriate logic level
0 = $\overline{SS}$ pin at appropriate logic level

SPTE — SPI Transmitter Empty Bit

This clearable, read-only flag is set each time the transmit data register transfers a byte into the shift register. SPTE generates an interrupt request if the SPTIE bit in the SPI control register is set also.

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

NOTE: *Do not write to the SPI data register unless the SPTE bit is high.*

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 **11.13.4 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 **11.8.2 Mode Fault Error**.)

SPR1 and SPR0 — SPI Baud Rate Select Bits

In master mode, these read/write bits select one of four baud rates as shown in **Table 11-3**. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 11-3. SPI Master Baud Rate Selection

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

Use this formula to calculate the SPI baud rate:

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

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

11.14.3 SPI Data Register

The SPI data register (SPDR) consists of the read-only receive data register and the write-only transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate registers that can contain different values. See **Figure 11-1.**

Address: $0011

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

Figure 11-15. SPI Data Register (SPDR)

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

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

Section 12. Serial Communications Interface (SCI) Module

12.1 Contents

12.2 Introduction

This section describes the serial communications interface module (SCI, Version D), which allows high-speed asynchronous communications with peripheral devices and other MCUs.

12.3 Features

Features of the SCI module include:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- 32 programmable baud rates
- Selectable clock source for baud rate (see **Section 14. Configuration Register (CONFIG)**)
- Programmable 8-bit or 9-bit character length
- Separately enabled transmitter and receiver
- Separate receiver and transmitter CPU interrupt requests
- Separate receiver and transmitter DMA service requests
- Programmable transmitter output polarity
- Two receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup

- Interrupt-driven operation with eight interrupt flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

12.4 Functional Description

Figure 12-1 shows the structure of the SCI module. The SCI allows full-duplex, asynchronous, NRZ serial communication between the MCU and remote devices, including other MCUs. The transmitter and receiver of the SCI operate independently, although they use the same baud rate generator.

During normal operation, the CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

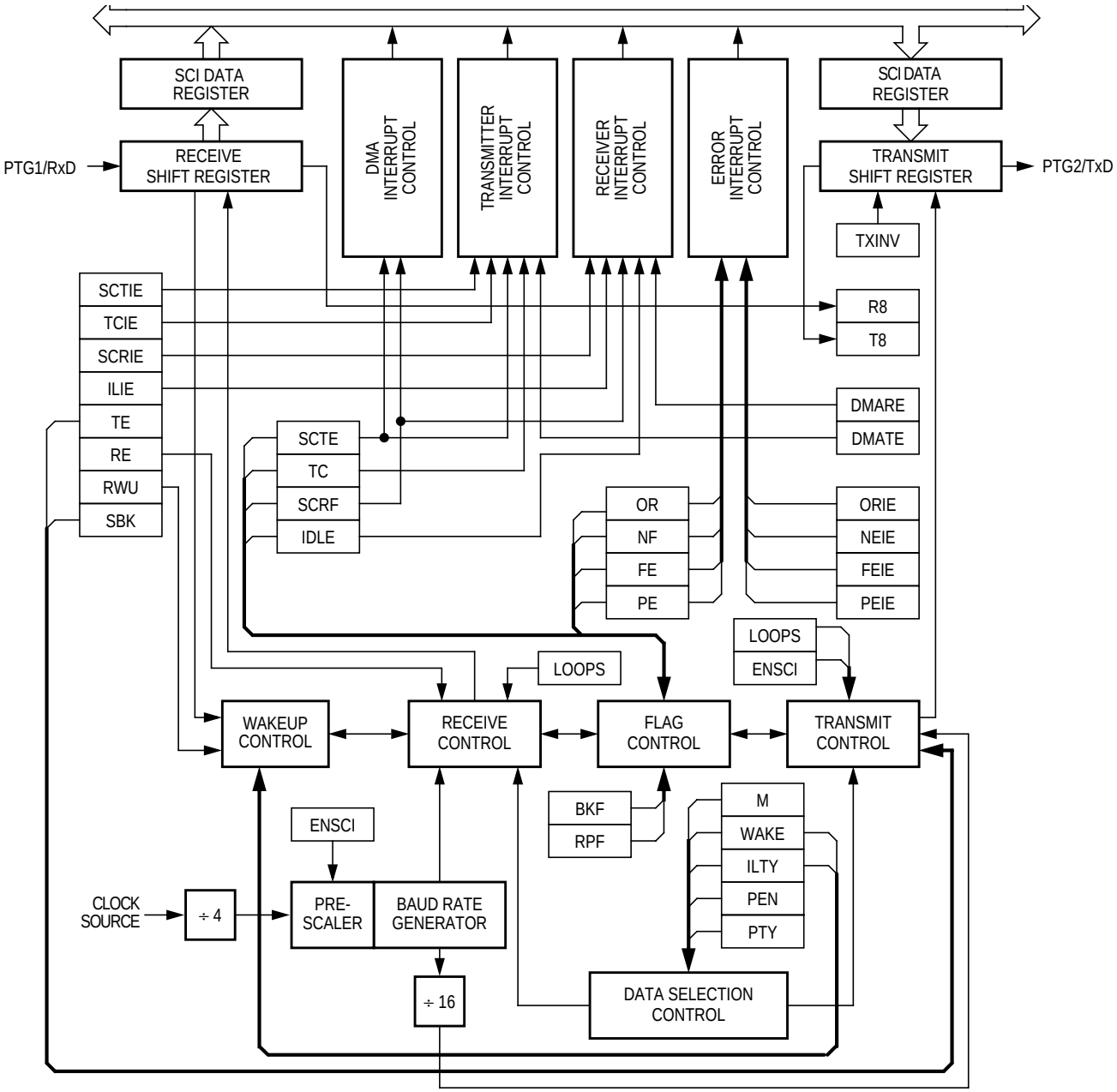


Figure 12-1. SCI Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0014	SCI Control Register 1 (SCC1)	Read: Write:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Reset:	0	0	0	0	0	0	0	0
$0015	SCI Control Register 2 (SCC2)	TCIE Write:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Reset:	0	0	0	0	0	0	0	0
$0016	SCI Control Register 3 (SCC3)	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
		Write:								
		Reset:	U	U	0	0	0	0	0	0
$0017	SCI Status Register 1 (SCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$0018	SCI Status Register 2 (SCS2)	Read:							BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0019	SCI Data Register (SCDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$001A	SCI Baud Rate Register (SCBR)	Read:			SCP1	SCP0		SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented U = Undetermined

Figure 12-2. SCI I/O Register Summary

12.4.1 Data Format

The SCI uses the standard non-return-to-zero (NRZ) mark/space data format illustrated in **Figure 12-3**.

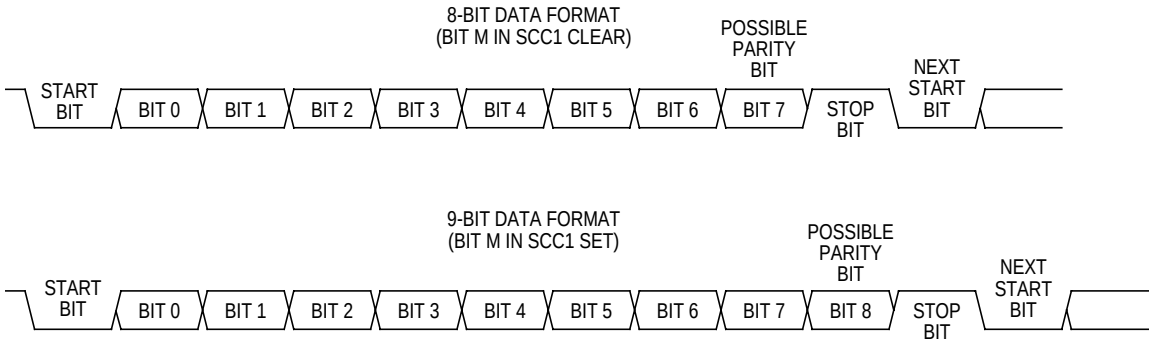


Figure 12-3. SCI Data Formats

12.4.2 Transmitter

Figure 12-4 shows the structure of the SCI transmitter.

NOTE: *The transmission output pin is enabled by TE bit of SCC2 instead of ENSCI bit of SCC1.*

12.4.2.1 Character Length

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

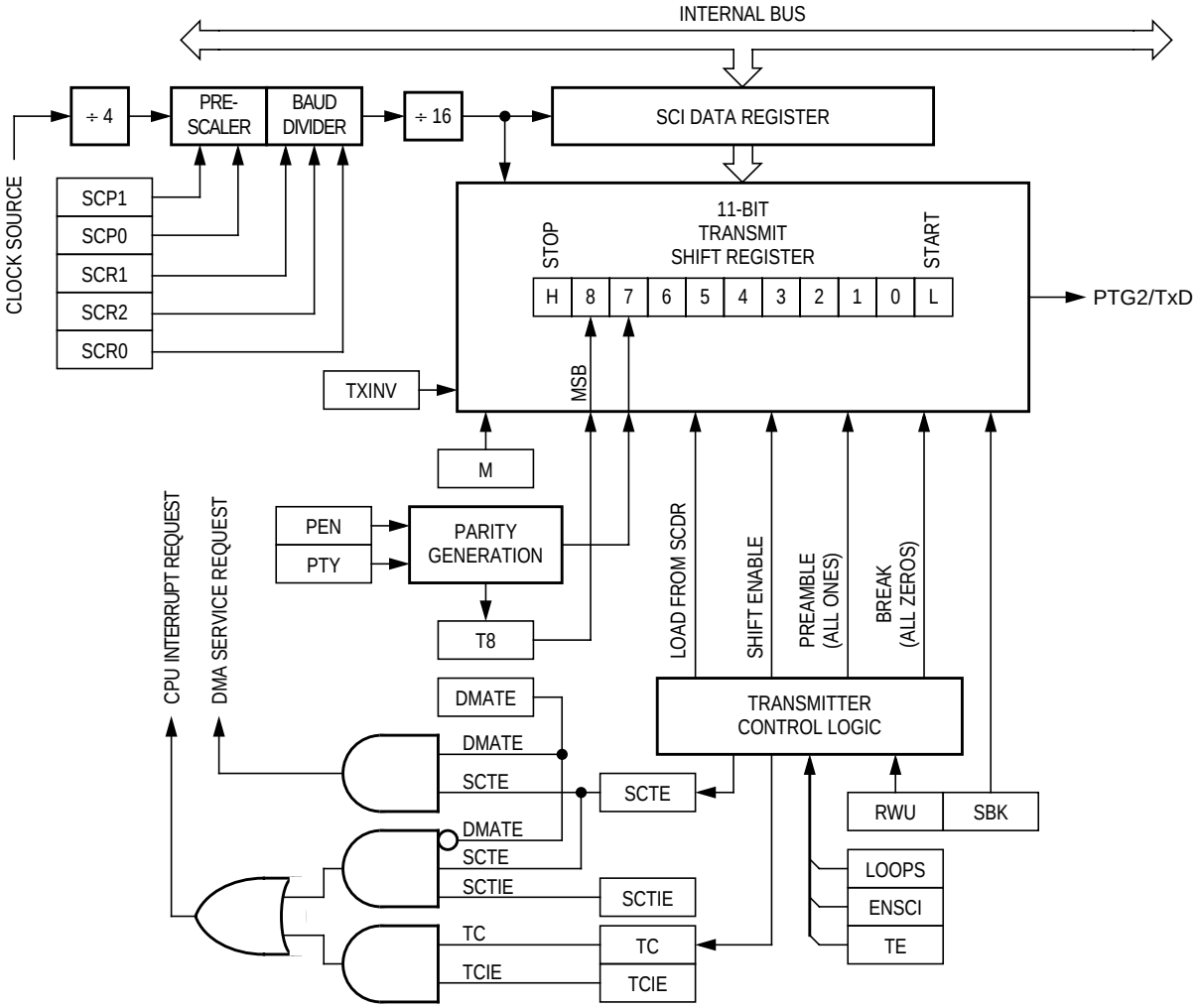


Figure 12-4. SCI Transmitter

12.4.2.2 Character Transmission

During an SCI transmission, the transmit shift register shifts a character out to the PTG2/TxD pin. The SCI data register (SCDR) is the write-only buffer between the internal data bus and the transmit shift register. To initiate an SCI transmission:

1. Enable the SCI by writing a logic 1 to the enable SCI bit (ENSCI) in SCI control register 1 (SCC1).
2. Enable the transmitter by writing a logic 1 to the transmitter enable bit (TE) in SCI control register 2 (SCC2).

3. Clear the SCI transmitter empty bit (SCTE) by reading SCI status register 1 (SCS1).
4. Write the data to transmit into the SCDR.
5. Repeat steps 3 and 4 for each subsequent transmission.

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

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

When the transmit shift register is not transmitting a character, the PTG2/TxD pin goes to the idle condition, logic 1. If at any time software clears the ENSCI bit in SCI control register 1 (SCC1), the transmitter and receiver relinquish control of the port E pins.

12.4.2.3 Break Characters

Writing a logic 1 to the send break bit, SBK, in SCC2 loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in SCC1. As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next character.

The SCI recognizes a break character when a start bit is followed by eight or nine logic 0 data bits and a logic 0 where the stop bit should be. Receiving a break character has these effects on SCI registers:

- Sets the framing error flag, FE
- Sets the SCI receiver full flag, SCRF
- Clears the SCI data register
- Clears the received bit 8, R8
- Sets the break flag, BKF
- May set the overrun flag, OR, noise flag, NF, parity error flag, PE, or the reception in progress flag, RPF

12.4.2.4 Idle Characters

An idle character contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in SCC1. The preamble is a synchronizing idle character that begins every transmission.

If the TE bit is cleared during a transmission, the PTG2/TxD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the character currently being transmitted.

NOTE: *When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current character shifts out to the PTG2/TxD pin. Setting TE after the stop bit appears on PTG2/TxD causes data previously written to the SCDR to be lost.*

A good time to toggle the TE bit is when the SCTE bit becomes set and just before writing the next byte to the SCDR.

12.4.2.5 Inversion of Transmitted Output

The transmit inversion bit (TXINV) in SCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values, including idle, break, start, and stop bits, are inverted when TXINV is at logic 1. (See **12.8.1 SCI Control Register 1**.)

12.4.2.6 Transmitter Interrupts

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

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

12.4.3 Receiver

Figure 12-5 shows the structure of the SCI receiver.

12.4.3.1 Character Length

The receiver can accommodate either 8-bit or 9-bit data. The state of the M bit in SCI control register 1 (SCC1) determines character length. When receiving 9-bit data, bit R8 in SCI control register 2 (SCC2) is the ninth bit (bit 8).

12.4.3.2 Character Reception

During an SCI reception, the receive shift register shifts characters in from the PTG1/RxD pin. The SCI data register (SCDR) is the read-only buffer between the internal data bus and the receive shift register.

After a complete character shifts into the receive shift register, the data portion of the character transfers to the SCDR. The SCI receiver full bit, SCRF, in SCI status register 1 (SCS1) becomes set, indicating that the received byte can be read. If the SCI receive interrupt enable bit, SCRIE, in SCC2 is also set, the SCRF bit generates an SCRF CPU interrupt request.

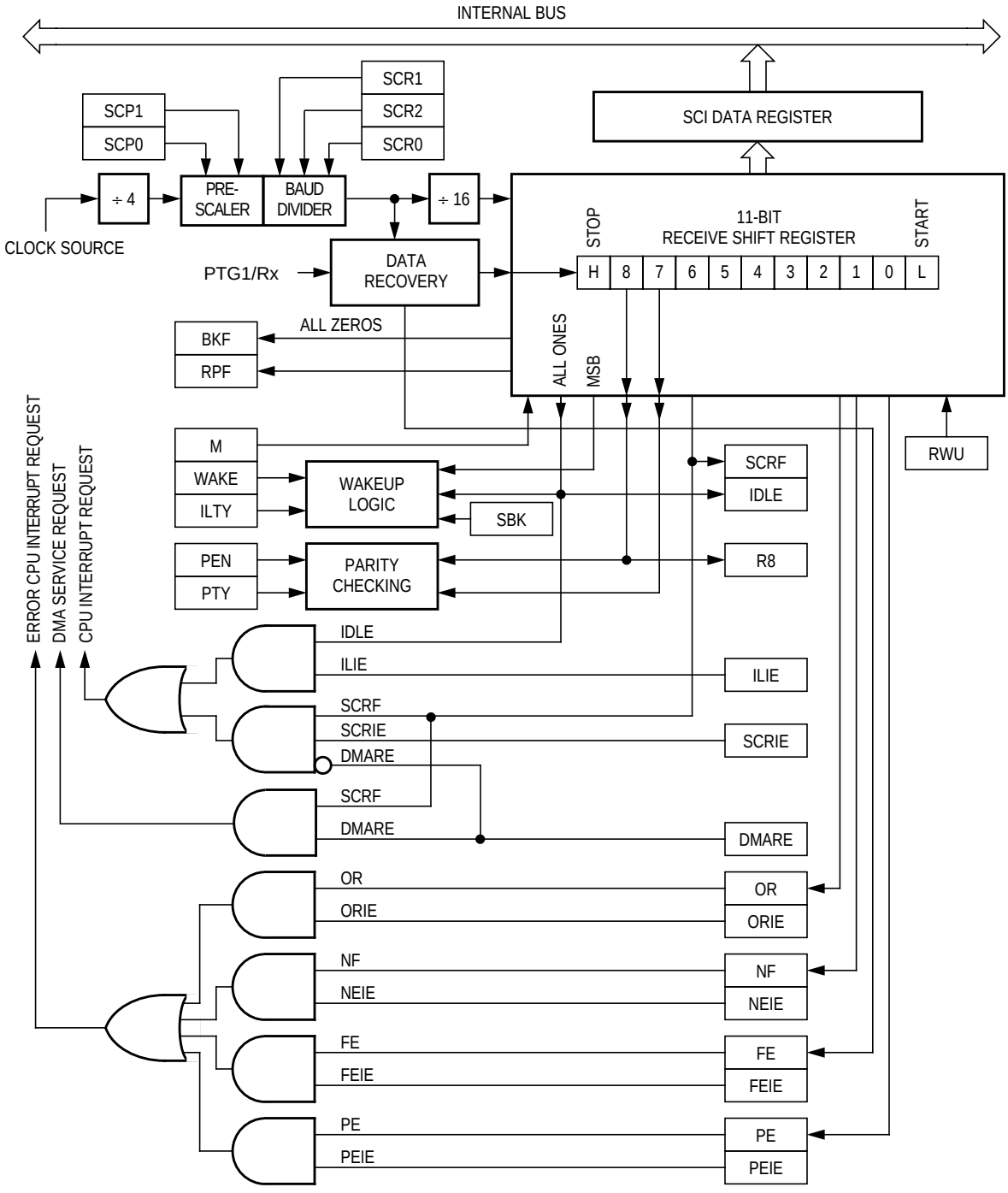


Figure 12-5. SCI Receiver Block Diagram

12.4.3.3 Data Sampling

The receiver samples the PTG1/RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud-rate frequency. (See **Figure 12-6**.)

- Start bit — To locate the start bit, recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.
- To verify a valid start bit, data recovery logic takes samples at RT3, RT5, and RT7. If any two of these three samples are logic 1s, the RT clock is reset and the search for start bit begins again. If all three samples are logic 0s, start bit verification is successful. If only one of the three samples is logic 1, start bit verification is successful, but the noise flag (NF) becomes set.
- Data bit — To detect noise in data bits, recovery logic takes samples at RT8, RT9, and RT10 of every data bit time. If all three samples are not unanimous, the noise flag becomes set.
- Stop bit — To detect noise in stop bits, recovery logic takes samples at RT8, RT9, and RT10. If all three samples are not unanimous, the noise flag becomes set.

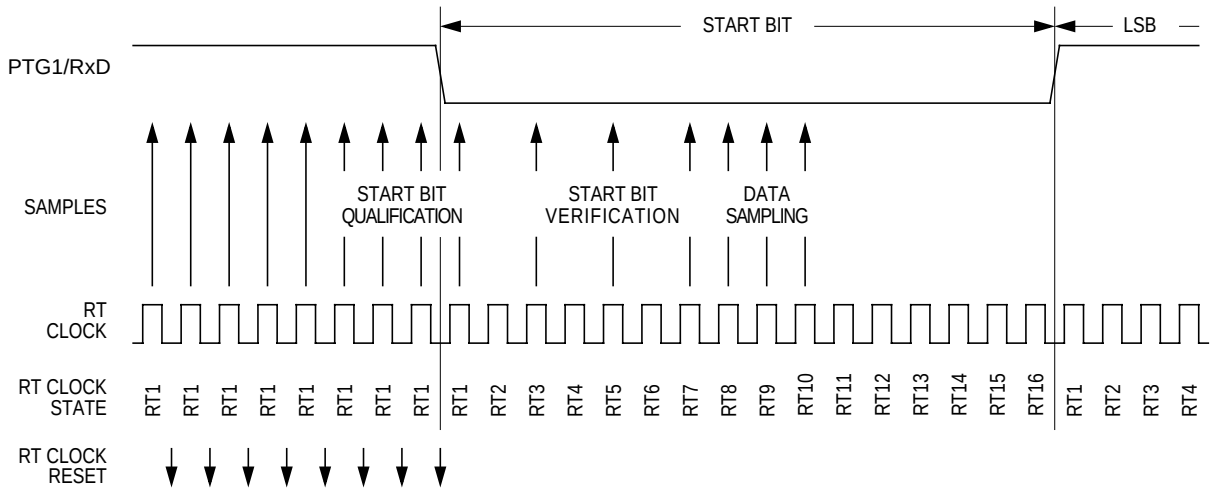


Figure 12-6. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. **Table 12-1** summarizes the results of the start bit verification samples.

Table 12-1. Start Bit Verification

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

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

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. **Table 12-2** summarizes the results of the data bit samples.

Table 12-2. Data Bit Recovery

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

NOTE: *The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s*

following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit.

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. **Table 12-3** summarizes the results of the stop bit samples.

Table 12-3. Stop Bit Recovery

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

12.4.3.4 Framing Errors

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

12.4.3.5 Baud Rate Tolerance

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

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

Slow Data Tolerance

Figure 12-7 shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

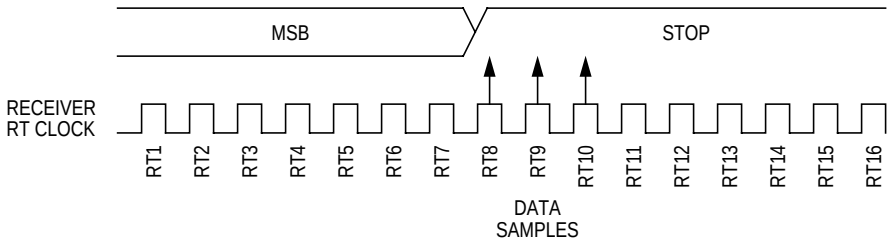


Figure 12-7. Slow Data

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

With the misaligned character shown in **Figure 12-7**, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit times × 16 RT cycles + 3 RT cycles = 147 RT cycles.

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

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

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

With the misaligned character shown in **Figure 12-7**, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles.

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

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

Fast Data Tolerance

Figure 12-8 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

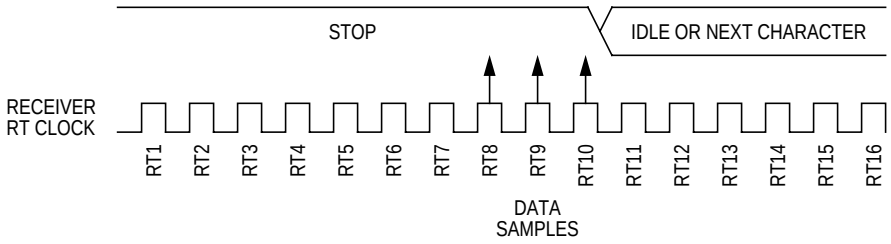


Figure 12-8. Fast Data

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

With the misaligned character shown in **Figure 12-8**, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles = 160 RT cycles.

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

$$\left|\frac{154 - 160}{154}\right| \times 100 = 3.\dot{9}0\%$$

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

With the misaligned character shown in **Figure 12-8**, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times × 16 RT cycles = 176 RT cycles.

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

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

12.4.3.6 Receiver Wakeup

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

Depending on the state of the WAKE bit in SCC1, either of two conditions on the PTG1/RxD pin can bring the MCU out of the standby state:

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

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

12.4.3.7 Receiver Interrupts

These two sources can generate CPU interrupt requests from the SCI receiver:

- SCI receiver full (SCRF) — The SCRF bit in SCS1 indicates that the receive shift register has transferred a character to the SCDR. SCRF can generate an SCRF CPU interrupt request or an SCRF DMA service request. Setting the SCI receive interrupt enable bit, SCRIE, in SCC2 enables SCRF CPU interrupts. Setting both the SCRIE bit and the DMA receive enable bit, DMARE, in SCC3 enables SCRF DMA service requests.
- Idle input (IDLE) — The IDLE bit in SCS1 indicates that 10 or 11 consecutive logic 1s shifted in from the PTG1/RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables IDLE CPU interrupts.

12.4.3.8 Error Interrupts

The following receiver error conditions can generate CPU interrupt requests:

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

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

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

12.5.2 Stop Mode

The SCI module is inactive after the execution of a STOP instruction. The STOP instruction does not affect SCI register states. SCI module operation resumes after an external interrupt.

12.6 SCI 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**.)

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

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does

the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

12.7 I/O Signals

Port E shares two of its pins with the SCI module. The two SCI I/O (input/output) pins are:

- PTG2/TxD — Transmit data
- PTG1/RxD — Receive data

12.7.1 PTG2/TxD (Transmit Data)

The PTG2/TxD pin is the serial data output from the SCI transmitter.

12.7.2 PTG1/RxD (Receive Data)

The PTG1/RxD pin is the serial data input to the SCI receiver.

12.8 I/O Registers

The following I/O registers control and monitor SCI operation:

- SCI control register 1 (SCC1)
- SCI control register 2 (SCC2)
- SCI control register 3 (SCC3)
- SCI status register 1 (SCS1)
- SCI status register 2 (SCS2)
- SCI data register (SCDR)
- SCI baud rate register (SCBR)

12.8.1 SCI Control Register 1

SCI control register 1 (SCC1):

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

Address: $0014

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

Figure 12-9. SCI Control Register 1 (SCC1)

LOOPS — Loop Mode Select Bit

This read/write bit enables loop mode operation. In loop mode, the PTE1/RxD pin is disconnected from the SCI, and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use loop mode. Reset clears the LOOPS bit.

1 = Loop mode enabled
0 = Normal operation enabled

ENSCI — Enable SCI Bit

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

1 = SCI enabled
0 = SCI disabled

TXINV — Transmit Inversion Bit

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

1 = Transmitter output inverted
0 = Transmitter output not inverted

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

M — Mode (Character Length) Bit

This read/write bit determines whether SCI characters are eight or nine bits long. (See **Table 12-4**.) The ninth bit can serve as an extra stop bit, as a receiver wakeup signal, or as a parity bit. Reset clears the M bit.

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

WAKE — Wakeup Condition Bit

This read/write bit determines which condition wakes up the SCI: a logic 1 (address mark) in the most significant bit position of a received character or an idle condition on the PTE1/RxD pin. Reset clears the WAKE bit.

1 = Address mark wakeup
0 = Idle line wakeup

ILTY — Idle Line Type Bit

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

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

PEN — Parity Enable Bit

This read/write bit enables the SCI parity function. (See **Table 12-4**.) When enabled, the parity function inserts a parity bit in the most significant bit position. (See **Figure 12-3**.) Reset clears the PEN bit.

1 = Parity function enabled
0 = Parity function disabled

PTY — Parity Bit

This read/write bit determines whether the SCI generates and checks for odd parity or even parity. (See **Table 12-4**.) Reset clears the PTY bit.

1 = Odd parity
0 = Even parity

Table 12-4. Character Format Selection

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

12.8.2 SCI Control Register 2

SCI control register 2 (SCC2):

- Enables the following interrupts:
  - Transmitter interrupts
  - Transmission complete interrupts
  - Receiver interrupts
  - Idle line interrupts
- Enables the transmitter
- Enables the receiver
- Enables SCI wakeup
- Transmits SCI break characters

Address: $0015

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

Figure 12-10. SCI Control Register 2 (SCC2)

SCTIE — SCI Transmit Interrupt Enable Bit

This read/write bit enables SCTE CPU interrupt requests or SCTE DMA service requests. Setting the SCTIE bit and clearing the DMA transfer enable bit, DMATE, in SCC3 enables SCTE CPU interrupt requests. Setting both the SCTIE and DMATE bits enables SCTE DMA service requests. Reset clears the SCTIE bit.

1 = SCTE CPU interrupt requests or SCTE DMA service requests enabled

0 = SCTE CPU interrupt requests or SCTE DMA service requests disabled

TCIE — Transmission Complete Interrupt Enable Bit

This read/write bit enables TC CPU interrupt requests. Reset clears the TCIE bit.

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

SCRIE — SCI Receive Interrupt Enable Bit

This read/write bit enables SCRF CPU interrupt requests or SCRF DMA service requests. Setting the SCRIE bit and clearing the DMA receive enable bit, DMARE, in SCC3 enables SCRF CPU interrupt requests. Setting both the SCRIE and DMARE bits enables SCRF DMA service requests. Reset clears the SCRIE bit.

1 = SCRF CPU interrupt requests or SCRF DMA service requests enabled
0 = SCRF CPU interrupt requests or SCRF DMA service requests disabled

ILIE — Idle Line Interrupt Enable Bit

This read/write bit enables IDLE CPU interrupt requests when the IDLE bit becomes set. Reset clears the ILIE bit.

1 = IDLE CPU interrupts enabled
0 = IDLE CPU interrupts disabled

TE — Transmitter Enable Bit

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 logic 1s from the transmit shift register to the PTE2/TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the PTG2/TxD returns to the idle condition (three-state). Reset clears the TE bit.

1 = Transmission enabled
0 = Transmission disabled

RE — Receiver Enable Bit

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

1 = Receiver enabled
0 = Receiver disabled

RWU — Receiver Wakeup Bit

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

1 = Standby state
0 = Normal operation

SBK — Send Break Bit

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

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

12.8.3 SCI Control Register 3

SCI control register 3 (SCC3):

- Stores the ninth SCI data bit received and the ninth SCI data bit to be transmitted
- Enables SCI receiver full (SCRF) DMA service requests
- Enables SCI transmitter empty (SCTE) DMA service requests
- Enables the following interrupts:
  - Receiver overrun interrupts
  - Noise error interrupts
  - Framing error interrupts
  - Parity error interrupts

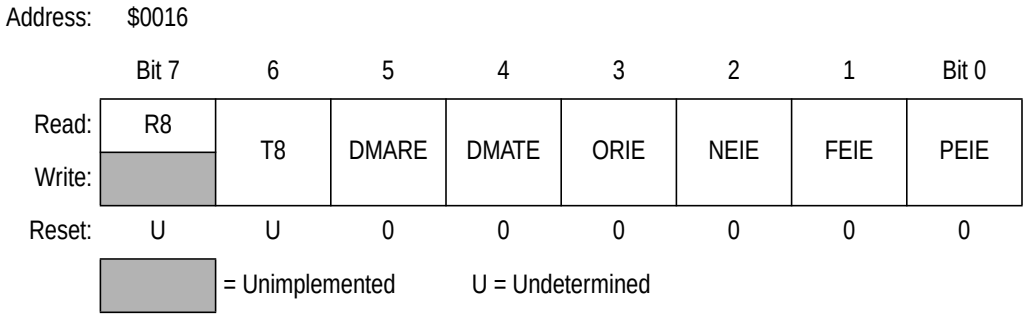


Figure 12-11. SCI Control Register 3 (SCC3)

R8 — Received Bit 8

When the SCI is receiving 9-bit characters, R8 is the read-only bit 8 of the received character. R8 is received at the same time that the SCDR receives the other eight bits. Reset has no effect on the R8 bit.

T8 — Transmitted Bit 8

When the SCI is transmitting 9-bit characters, T8 is the read/write bit 8 of the transmitted character. T8 is loaded into the transmit shift register at the same time that the SCDR is loaded into the transmit shift register. Reset has no effect on the T8 bit.

DMARE — DMA Receive Enable Bit

This read/write bit enables SCI receiver full (SCRF) DMA service requests. (See **12.8.4 SCI Status Register 1**.) Setting the DMARE bit disables SCRF CPU interrupt requests. Reset clears the DMARE bit.

1 = SCRF DMA service requests enabled (SCRF CPU interrupt requests disabled)

0 = SCRF DMA service requests disabled (SCRF CPU interrupt requests enabled)

DMATE — DMA Transfer Enable Bit

This read/write bit enables SCI transmitter empty (SCTE) DMA service requests. (See **12.8.4 SCI Status Register 1**.) Setting the DMATE bit disables SCTE CPU interrupt requests. Reset clears DMATE.

1 = SCTE DMA service requests enabled (SCTE CPU interrupt requests disabled)
0 = SCTE DMA service requests disabled (SCTE CPU interrupt requests enabled)

ORIE — Receiver Overrun Interrupt Enable Bit

This read/write bit enables receiver overrun (OR) CPU interrupt requests. (See **12.8.4 SCI Status Register 1**.) Reset clears ORIE.

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

NEIE — Receiver Noise Error Interrupt Enable Bit

This read/write bit enables receiver noise error (NE) CPU interrupt requests. (See **12.8.4 SCI Status Register 1**.) Reset clears NEIE.

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

FEIE — Receiver Framing Error Interrupt Enable Bit

This read/write bit enables receiver framing error (FE) CPU interrupt requests. (See **12.8.4 SCI Status Register 1**.) Reset clears FEIE.

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

PEIE — Receiver Parity Error Interrupt Enable Bit

This read/write bit enables receiver parity error (PE) CPU interrupt requests. (See **12.8.4 SCI Status Register 1**.) Reset clears PEIE.

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

12.8.4 SCI Status Register 1

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

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

Address: $0017

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

= Unimplemented

Figure 12-12. SCI Status Register 1 (SCS1)

SCTE — SCI Transmitter Empty Bit

This clearable, read-only bit is set when the SCDR transfers a character to the transmit shift register. SCTE can generate an SCTE CPU interrupt request or an SCTE DMA service request. When the SCTIE bit in SCC2 is set and the DMATE bit in SCC3 is clear, SCTE generates an SCTE CPU interrupt request. With both the SCTIE and DMATE bits set, SCTE generates an SCTE DMA service request. In normal operation, clear the SCTE bit by reading SCS1 with SCTE set and then writing to SCDR. In DMA transfers, the DMA automatically clears the SCTE bit when it writes to the SCDR. Reset sets the SCTE bit.

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

TC — Transmission Complete Bit

This clearable, read-only bit is set when the SCTE bit is set, and no data, preamble, or break character is being transmitted. TC generates a TC CPU interrupt request if the TCIE bit in SCC2 is also set. Clear the TC bit by reading SCS1 with TC set and then writing to the SCDR. When the DMA services an SCTE DMA service request, the DMA clears the TC bit by writing to the SCDR. Reset sets the TC bit.

1 = No transmission in progress
0 = Transmission in progress

SCRF — SCI Receiver Full Bit

This clearable, read-only bit is set when the data in the receive shift register transfers to the SCI data register. SCRF can generate an SCRF CPU interrupt request or an SCRF DMA service request. When the SCRIE bit in SCC2 is set and the DMARE bit in SCC3 is clear, SCRF generates an SCRF CPU interrupt request. With both the SCRIE and DMARE bits set, SCRF generates an SCRF DMA service request. In normal operation, clear the SCRF bit by reading SCS1 with SCRF set and then reading the SCDR. In DMA transfers, the DMA clears the SCRF bit when it reads the SCDR. Reset clears SCRF.

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

IDLE — Receiver Idle Bit

This clearable, read-only bit is set when 10 or 11 consecutive logic 1s appear on the receiver input. IDLE generates an IDLE CPU interrupt request if the ILIE bit in SCC2 is also set. Clear the IDLE bit by reading SCS1 with IDLE set and then reading the SCDR. Once cleared, the IDLE bit can become set again only after the SCRF bit becomes set and another idle character appears on the receiver input. Reset clears the IDLE bit.

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

OR — Receiver Overrun Bit

This clearable, read-only bit is set when software fails to read the SCDR before the receive shift register receives the next character. The OR bit generates an OR CPU interrupt request if the ORIE bit in SCC3 is also set. The data in the shift register is lost, but the data already in the SCDR is not affected. Clear the OR bit by reading SCS1 with OR set and then reading the SCDR. Reset clears the OR bit.

1 = Receive shift register full and SCRF = 1
0 = No receiver overrun

NF — Receiver Noise Flag Bit

This clearable, read-only bit is set when the SCI detects noise on the PTG1/RxD pin. NF generates an NF CPU interrupt request if the NEIE bit in SCC3 is also set. Clear the NF bit by reading SCS1 and then reading the SCDR. Reset clears the NF bit.

1 = Noise detected
0 = No noise detected

FE — Receiver Framing Error Bit

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

1 = Framing error detected
0 = No framing error detected

PE — Receiver Parity Error Bit

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

1 = Parity error detected
0 = No parity error detected

12.8.5 SCI Status Register 2

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

- Break character detected
- Incoming data

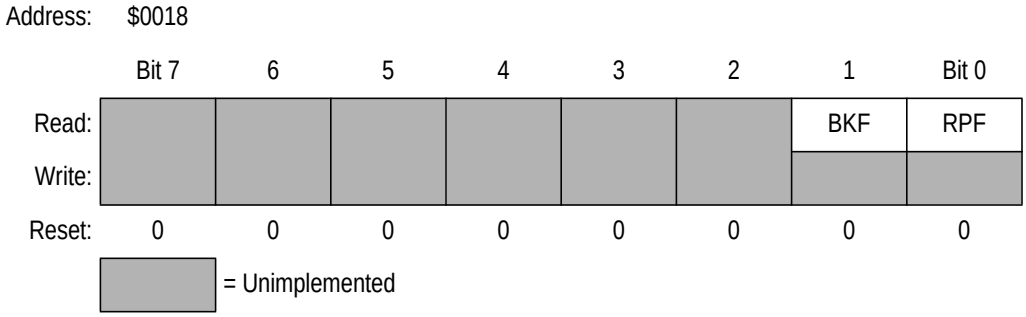


Figure 12-13. SCI Status Register 2 (SCS2)

BKF — Break Flag Bit

This clearable, read-only bit is set when the SCI detects a break character on the PTG1/RxD pin. BKF does not generate an interrupt request. Clear BKF by reading SCS2 with BKF set and then reading the SCDR. Once cleared, BKF can become set again only after logic 1s again appear on the PTG1/RxD pin followed by another break character. Reset clears the BKF bit.

1 = Break character detected
0 = No break character detected

RPF — Reception in Progress Flag Bit

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

1 = Reception in progress
0 = No reception in progress

12.8.6 SCI Data Register

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

Address: $0019

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

Figure 12-14. SCI Data Register (SCDR)

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

Reading SCDR accesses the read-only received data bits, R7–R0. Writing to SCDR writes the data to be transmitted, T7–T0. Reset has no effect on the SCI data register.

12.8.7 SCI Baud Rate Register

The baud rate register (SCBR) selects the baud rate for both the receiver and the transmitter.

Address: $001A

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

= Unimplemented

Figure 12-15. SCI Baud Rate Register (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in **Table 12-5**. Reset clears SCP1 and SCP0.

Table 12-5. SCI Baud Rate Prescaling

SCP1 and SCP0	Prescaler Divisor (PD)
00	1
01	3
10	4
11	13

RCHK — SCI Rate Check

This bit is only available in Test Mode. Setting this bit enables the transmitter clock to be visible on the transmit data pin instead of the transmit data.

SCR2–SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in **Table 12-6**. Reset clears SCR2–SCR0.

Table 12-6. SCI Baud Rate Selection

SCR2, SCR1, and SCR0	Baud Rate Divisor (BD)
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Use this formula to calculate the SCI baud rate:

$$\text{Baud rate} = \frac{\text{CGMOUT} \div 2}{64 \times \text{PD} \times \text{BD}}$$

where:

CGMOUT÷2 = bus frequency
PD = prescaler divisor
BD = baud rate divisor

SCI_BDSRC is an input to the SCI. Normally, it will be tied off low at the top level to select CGMXLCK as the clock source. If it is tied off high, it will select IT12 as the clock source. This makes the formula:

$$\text{Baud rate} = \frac{\text{IT12}}{64 \times \text{PD} \times \text{BD}}$$

Table 12-7 shows the SCI baud rates that can be generated with a 4.9152-MHz crystal.

Table 12-7. SCI Baud Rate Selection Examples

SCP1 and SCP0	Prescaler Divisor (PD)	SCR2, SCR1, and SCR0	Baud Rate Divisor (BD)	Baud Rate (CGMXCLK ÷ 2 = 0.8192 MHz)
00	1	000	1	12,800
00	1	001	2	6400
00	1	010	4	3200
00	1	011	8	1600
00	1	100	16	800
00	1	101	32	400
00	1	110	64	200
00	1	111	128	100
01	3	000	1	4267
01	3	001	2	2133
01	3	010	4	1067
01	3	011	8	533
01	3	100	16	267
01	3	101	32	133
01	3	110	64	67
01	3	111	128	33
10	4	000	1	3200
10	4	001	2	1600
10	4	010	4	800
10	4	011	8	400
10	4	100	16	200
10	4	101	32	100
10	4	110	64	50
10	4	111	128	25
11	13	000	1	984
11	13	001	2	492
11	13	010	4	246
11	13	011	8	123
11	13	100	16	62
11	13	101	32	31
11	13	110	64	15
11	13	111	128	8

Section 13. Analog-to-Digital Converter (ADC) Module

13.1 Contents

13.2 Introduction

This section describes the 8-bit analog-to-digital converter (ADC).

13.3 Features

Features include:

- Four channels with multiplexed input
- Linear successive approximation
- 8-bit resolution
- Single or continous conversion
- Conversion complete flag or conversion complete interrupt
- Selectable ADC clock

13.4 Functional Description

Four pins for sampling external sources are located at pins PTE7/AD3–PTE4/AD0. An analog multiplexer allows the single ADC to select one of four ADC channels as ADC voltage input (ADVIN). ADVIN is converted by the successive approximation register based ADC. When the conversion is completed, ADC places the result in the ADC data register and sets a flag or generates an interrupt.
(See **Figure 13-1**.)

NOTE: *References to DMA and associated functions are only valid if the MCU has a DMA module. If the MCU has no DMA, any DMA-related register bits should be left in their reset state for expected MCU operation.*

13.4.1 ADC Port I/O Pins

PTE7/AD3–PTE4/AD0 are general-purpose input/output (I/O) pins that share with the ADC channels. The channel select bits define which ADC channel/port pin will be used as the input signal. The ADC overrides the port I/O logic by forcing that pin as input to the ADC. The remaining ADC channels/port pins are controlled by the port I/O logic and can be used as general-purpose I/O. Writes to the port register or data direction register (DDR) will not have any effect on the port pin that is selected by the ADC. A read of a port pin which is in use by the ADC will return a logic 0.

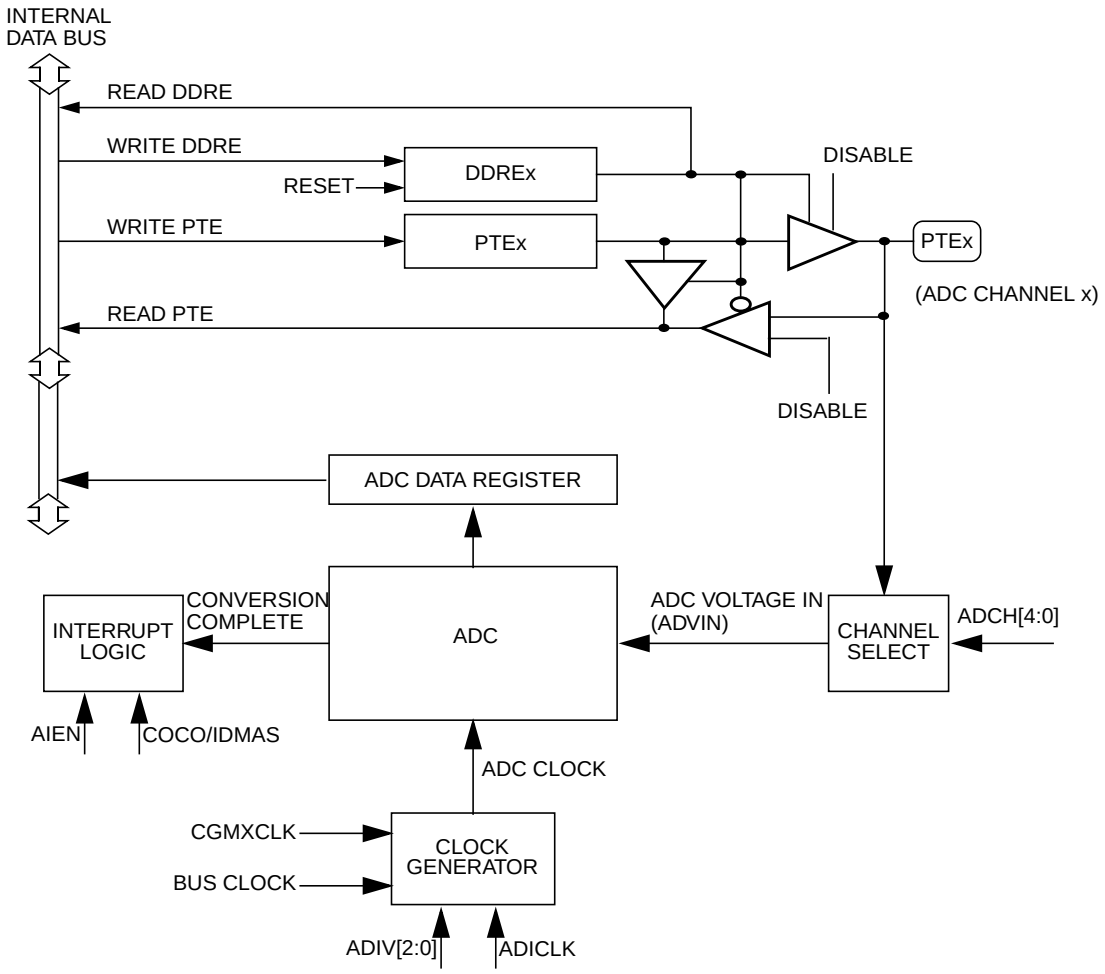


Figure 13-1. ADC Block Diagram

13.4.2 Voltage Conversion

When the input voltage to the ADC equals V_{RH}, the ADC converts the signal to $FF (full scale). If the input voltage equals AV_{SS}, the ADC converts it to $00. Input voltages between V_{RH} and AV_{SS} are a straight-line linear conversion. All other input voltages will result in $FF, if greater than V_{RH}.

NOTE: *Input voltage should not exceed the analog supply voltages.*

13.4.3 Conversion Time

Conversion starts after a write to the ADC status and control register (ADSCR). Conversion time in terms of the number of bus cycles is a function of oscillator frequency, bus frequency, and ADIV prescaler bits. For example, with bus frequency of 4 MHz and ADC clock frequency of 1 MHz, one conversion will take between 16 ADC and 17 ADC clock cycles or between 16 µs and 17 µs. There will be 128 bus cycles between each conversion. Sample rate is approximately 30 kHz.

$$\text{Conversion time} = \frac{\text{16–17 ADC cycles}}{\text{ADC frequency}}$$

Bus cycles = conversion time x bus frequency

13.4.4 Conversion

In the continuous conversion mode, the ADC data register (ADR) will be filled with new data after each conversion. Data from the previous conversion will be overwritten whether that data has been read or not. Conversions will continue until the ADCO bit is cleared. The COCO/IDMAS bit is set after the first conversion and will stay set until the next write of the ADSCR or the next read of the ADR.

In the single conversion mode, conversion begins with a write to the ADSCR. Only one conversion occurs between writes to the ADSCR.

13.4.5 Accuracy and Precision

The conversion process is monotonic and has no missing codes.

13.5 Interrupts

When the AIEN bit is set, the ADC module is capable of generating either CPU or DMA interrupts after each ADC conversion. A CPU interrupt is generated if the COCO/IDMAS bit is at logic 0. If COCO/IDMAS bit is set, a DMA interrupt is generated. The

COCO/IDMAS bit is not used as a conversion complete flag when interrupts are enabled.

13.6 Low-Power Modes

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

13.6.1 Wait Mode

The ADC continues normal operation during wait mode. Any enabled CPU interrupt request from the ADC can bring the MCU out of wait mode. If the ADC is not required to bring the MCU out of wait mode, power down the ADC by setting ADCH4–ADCH0 bits in the ADSCR before executing the WAIT instruction.

13.6.2 Stop Mode

The ADC module is inactive after the execution of a STOP instruction. Any pending conversion is aborted. ADC conversions resume when the MCU exits stop mode after an external interrupt. Allow one conversion cycle to stabilize the analog circuitry.

13.7 I/O Signals

The ADC module has four pins shared with port E.

13.7.1 ADC Analog Power Pin (V_{DDA2})

The ADC analog portion uses AV_{DD} as its power pin. Connect the V_{DDA2} pin to the same voltage potential as V_{DD}. External filtering may be necessary to ensure clean V_{DDA2} for good results.

NOTE: *Route AV_{DD} carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.*

13.7.2 ADC Analog Ground Pin (V_{SSA2})

The ADC analog portion uses AV_{SS} as its ground pin. Connect the V_{SSA2} pin to the same voltage potential as V_{SS}.

13.7.3 ADC Voltage Reference Pin (V_{RH})

V_{RH} is the power supply for setting the reference voltage V_{RH}. Connect the V_{RH} pin to a voltage potential <= V_{DDA2}, not less than 1.5 V.

13.7.4 ADC Voltage In (ADVIN)

ADVIN is the input voltage signal from one of the four ADC channels to the ADC module.

13.8 Input/Output Registers

These I/O registers control and monitor operation of the ADC:

- ADC status and control register (ADSCR)
- ADC data register (ADR)
- ADC clock register (ADCLK)

13.8.1 ADC Status and Control Register

The function of the ADC status and control register (ADSCR) is described here.

Address $004C

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

Figure 13-2. ADC Status and Control Register (ADSCR)

COCO/IDMAS — Conversions Complete/Interrupt DMA Select

When AIEN bit is a logic 0, the COCO/IDMAS is a read-only bit which is set each time a conversion is completed except in the continous conversion mode where it is set after the first conversion. This bit is cleared whenever the ADSCR is written or whenever the ADR is read.

If AIEN bit is a logic 1, the COCO/IDMAS is a read/write bit which selects either CPU or DMA to service the ADC interrupt request. Reset clears this bit.

1 = Conversion completed (AIEN = 0)/DMA interrrupt (AIEN = 1)
0 = Conversion not completed (AIEN = 0)/CPU interrupt (AIEN = 1)

AIEN — ADC Interrupt Enable

When this bit is set, an interrupt is generated at the end of an ADC conversion. The interrupt signal is cleared when the ADR is read or the ADSCR is written. Reset clears AIEN bit.

1 = ADC interrupt enabled
0 = ADC interrupt disabled

ADCO — ADC Continuous Conversion

When set, the ADC will continuously convert samples and update the ADR at the end of each conversion. Only one conversion is completed between writes to the ADSCR when this bit is cleared. Reset clears the ADCO bit.

1 = Continuous ADC conversion
0 = One ADC conversion

ADCH4–ADCH0 — ADC Channel Select Bits

ADCH4–ADCH0 form a 5-bit field which is used to select one of 16 ADC channels. Only four channels, ADCH3–ADCH0, are avaliable on this MCU. The channels are detailed in **Table 13-1**. Care should be taken when using a port pin as both an analog and digital input simultaneously to prevent switching noise from corrupting the analog signal.

The ADC subsystem is turned off when the channel select bits are all set to 1. This feature allows for reduced power consumption for the MCU when the ADC is not being used.

NOTE: *Recovery from the disabled state requires one conversion cycle to stabilize.*

The voltage levels supplied from internal reference nodes as specified in **Table 13-1** are used to verify the operation of the ADC converter both in production test and for user applications.

Table 13-1. Mux Channel Select

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	Input Select
0	0	0	0	0	PTE4/AD0
0	0	0	0	1	PTE5/AD1
0	0	0	1	0	PTE6/AD2
0	0	0	1	1	PTE7/AD3
0	0	1	0	0	Reserved
↓	↓	↓	↓	↓	Reserved
1	1	0	1	1	Reserved
1	1	1	0	0	V_{RH}
1	1	1	0	1	V_{RH}
1	1	1	1	0	AV_{SS}
1	1	1	1	1	ADC power off

Note: If any unused channels are selected, the resulting ADC conversion will be unknown.

13.8.2 ADC Data Register

One 8-bit result register, ADC data register (ADR), is provided. This register is updated each time an ADC conversion completes.

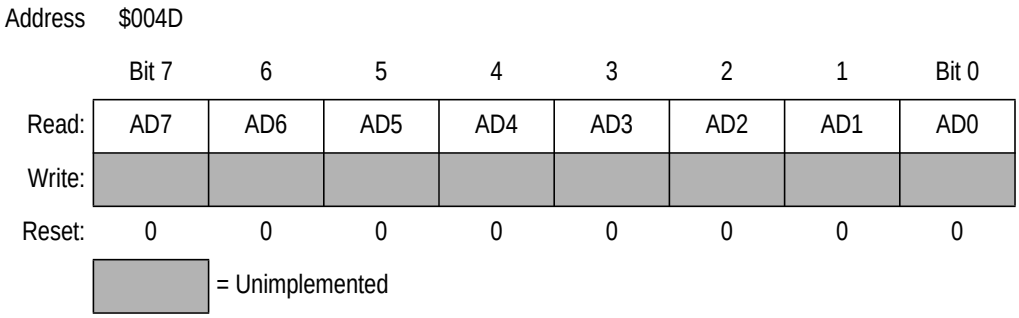


Figure 13-3. ADC Data Register (ADR)

13.8.3 ADC Clock Register

The ADC clock register (ADCLKR) selects the clock frequency for the ADC.

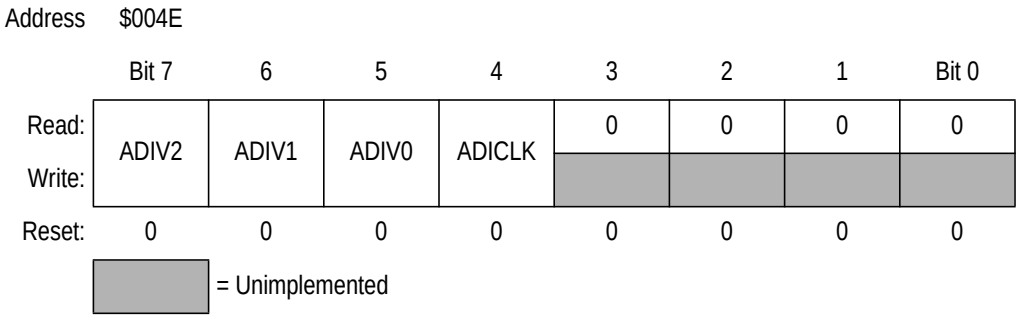


Figure 13-4. ADC Clock Register (ADCLKR)

ADIV2–ADIV0 — ADC Clock Prescaler Bits

ADIV2, ADIV1, and ADIV0 form a 3-bit field which selects the divide ratio used by the ADC to generate the internal ADC clock. **Table 13-2** shows the available clock configurations. The ADC clock should be set to approximately 1 MHz.

Table 13-2. ADC Clock Divide Ratio

ADIV2	ADIV1	ADIV0	ADC Clock Rate
0	0	0	ADC input clock ÷ 1
0	0	1	ADC input clock ÷ 2
0	1	0	ADC input clock ÷ 4
0	1	1	ADC input clock ÷ 8
1	X	X	ADC input clock ÷ 16

X = Don’t care

ADICLK — ADC Input Clock Select

ADICLK selects either bus clock or CGMXCLK as the input clock source to generate the internal ADC clock. Reset selects CGMXCLK as the ADC clock source.

If the external clock (CGMXCLK) is equal or greater than 1 MHz, CGMXCLK can be used as the clock source for the ADC. If CGMXCLK is less than 1 MHz, use the PLL-generated bus clock as the clock source. As long as the internal ADC clock is at approximately 1 MHz, correct operation can be guaranteed.

1 = Internal bus clock

0 = External clock (CGMXCLK)

$$1\ \text{MHz} = \frac{\text{ADC input clock frequency}}{\text{ADIV[2:0]}}$$

Section 14. Configuration Register (CONFIG)

14.1 Contents

14.2 Introduction

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

- Stop mode recovery time (32 OSCXCLK cycles or 4096 OSCXCLK cycles)
- STOP instruction
- Computer operating properly module (COP)
- SCI clock source select (SCIBDSRC)

14.3 Functional Description

The configuration register is used in the initialization of various options. The configuration register can be written once after each 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 $003F. For compatibility, a write to a ROM version of the MCU at this location will have no effect. The configuration register may be read at any time.

Figure 14-1. Configuration Register (CONFIG)

NOTE: *The CONFIG module is known as an MOR (mask option register) on a ROM device. (See **Figure 14-2**.) For references in the documentation which refer to the MOR, the CONFIG would be applicable for the FLASH version of the device.*

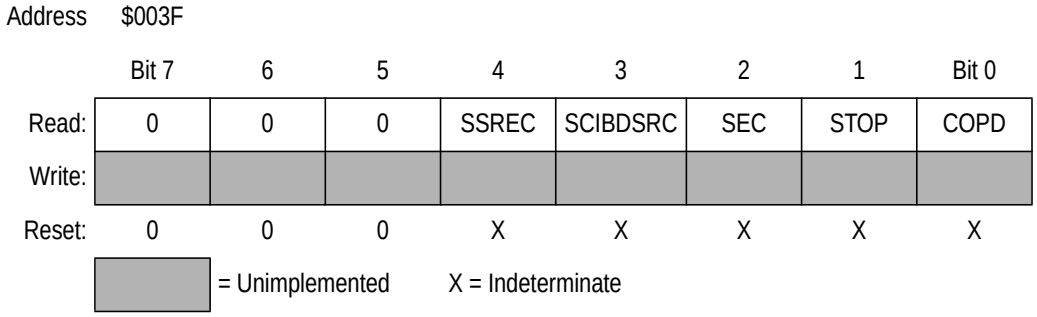


Figure 14-2. Mask Option Register (MOR)

SSREC — Short Stop Recovery Bit

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

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

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

If using an external crystal, do not set the SSREC bit.

SCIBDSRC — SCI Clock Select

1 = SCI Clock selects IT12
0 = SCI Clock selects CGMXCLK

SEC — ROM Security Bit

1 = ROM security enabled
0 = ROM security disabled

NOTE: *This bit is not valid on the MC68HC908PT48.*

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 5. Computer Operating Properly (COP) Module**.)

1 = COP module disabled
0 = COP module enabled

Configuration Register (CONFIG)

Section 15. Timer Interface Module (TIM)

15.1 Contents

15.2 Introduction

This section describes the timer interface module (TIM, version B). The TIM is a 4-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. **Figure 15-1** is a block diagram of the TIM.

15.3 Features

Features of the TIM include:

- Modular architecture
- 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
  - 7-frequency internal bus clock prescaler selection
  - External TIM clock input (2-MHz maximum frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- Timer counter stop and reset bit

15.4 Functional Description

Figure 15-1 shows the structure of the TIM. The central component of the TIM is the 16-bit timer counter that can operate as a free-running counter or a modulo up-counter. The timer counter provides the timing reference for the input capture and output compare functions. The timer counter modulo registers, TMODH:TMODL, control the modulo value of the timer counter. Software can read the timer counter value at any time without affecting the counting sequence.

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

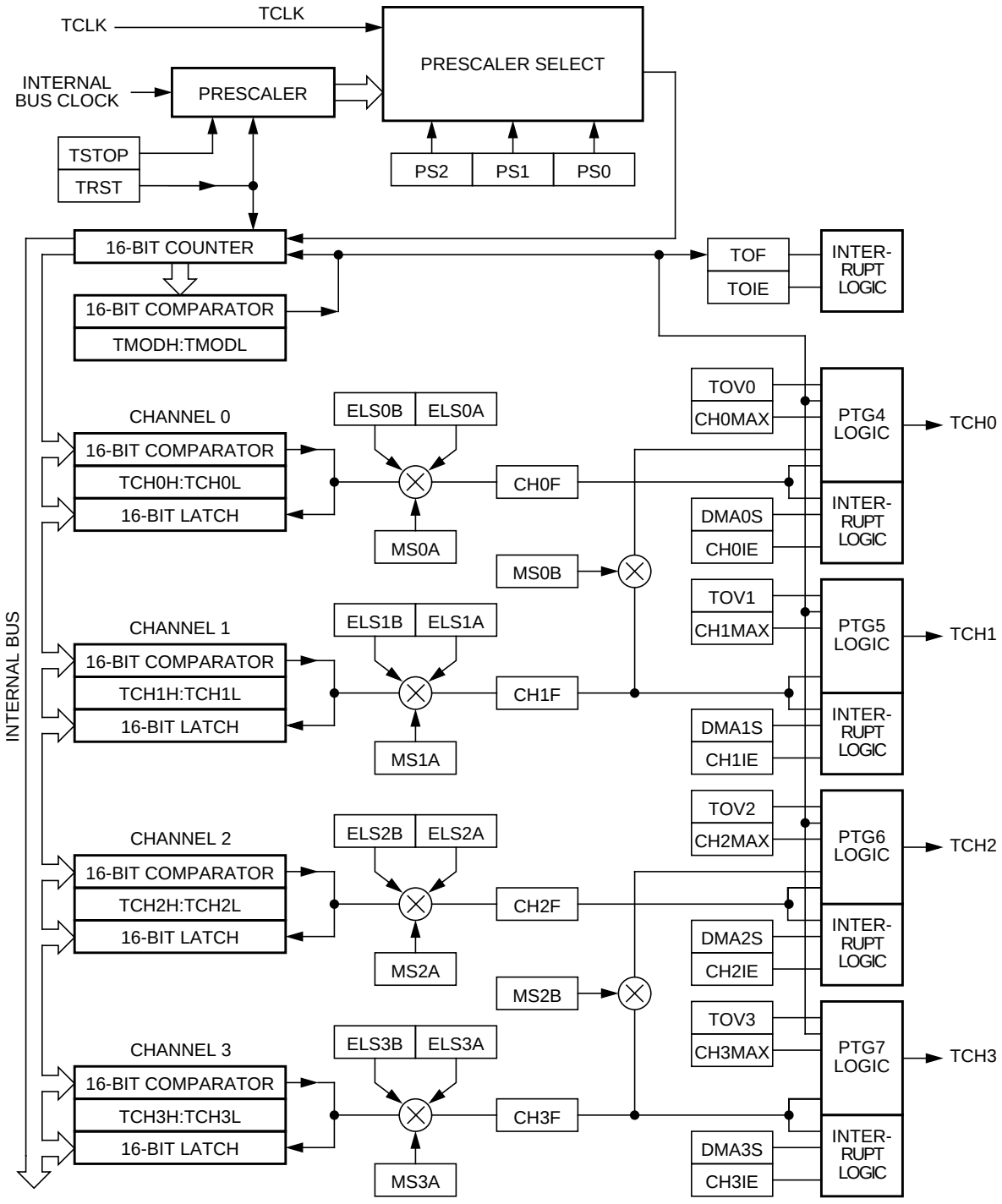


Figure 15-1. TIM Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0020	Timer Status and Control Register (TSC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0022	Timer Counter Register High (TCNTH)	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0023	Timer Counter Register Low (TCNTL)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0024	Timer Modulo Register High (TMODH)	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	Timer Modulo Register Low (TMODL)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0026	Timer Channel 0 Status and Control Register (TSC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0027	Timer Channel 0 Register High (TCH0H)	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0028	Timer Channel 0 Register Low (TCH0L)	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0029	Timer Channel 1 Status and Control Register (TSC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 15-2. Timer I/O Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002A	Timer Channel 1 Register High (TCH1H)	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	Indeterminate after reset							
$002B	Timer Channel 1 Register Low (TCH1L)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	Unaffected by reset							
$002C	Timer Channel 2 Status and Control Register (TSC2)	Read: Write:	CH2F 0	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		Reset:	0	0	0	0	0	0	0	0
$002D	Timer Channel 2 Register High (TCH2H)	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	Unaffected by reset							
$002E	Timer Channel 2 Register Low (TCH2L)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	Unaffected by reset							
$002F	Timer Channel 3 Status and Control Register (TSC3)	Read: Write:	CH3F 0	CH3IE	0 = Unimplemented	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Reset:	0	0	0	0	0	0	0	0
$0030	Timer Channel 3 Register High (TCH3H)	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	Unaffected by reset							
$0031	Timer Channel 3 Register Low (TCH3L)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	Unaffected by reset							

Figure 15-2. Timer I/O Register Summary (Continued)

15.4.1 Timer Counter Prescaler

The TIM clock source can be one of the seven prescaler outputs or the TIM clock pin, PTG3/TCLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS2–PS0, in the timer status and control register select the TIM clock source.

15.4.2 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 timer counter into the timer channel registers, TCHxH:TCHxL. The polarity of the active edge is programmable. Input capture latency can be up to three bus clock cycles. Input captures can generate TIM CPU interrupt requests.

15.4.3 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 or TIM DMA service requests.

15.4.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in **15.4.3 Output Compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the timer channel registers.

An unsynchronized write to the timer 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 timer overflow interrupt routine to write a new, smaller output

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

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

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

15.4.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTG4/TCH0 pin. The timer channel registers of the linked pair alternately control the output.

Setting the MS0B bit in timer channel 0 status and control register (TSC0) links channel 0 and channel 1. The output compare value in the timer channel 0 registers initially controls the output on the PTG4/TCH0 pin. Writing to the timer channel 1 registers enables the timer channel 1 registers to synchronously control the output after the timer overflows. At each subsequent overflow, the timer 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 timer channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTG5/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 PTG6/TCH2 pin. The timer channel registers of the linked pair alternately control the output.

Setting the MS2B bit in timer channel 2 status and control register (TSC2) links channel 2 and channel 3. The output compare value in the timer channel 2 registers initially controls the output on the PTG6/TCH2 pin. Writing to the timer channel 3 registers enables the timer channel 3 registers to synchronously control the output after the timer overflows. At each subsequent overflow, the timer 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 timer channel 3 status and control register (TSC3) is unused. 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.

15.4.4 Pulse Width Modulation (PWM)

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

As **Figure 15-3** shows, the output compare value in the timer 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 1. Program the TIM to set the pin if the state of the PWM pulse is logic 0.

The value in the timer counter modulo registers and the selected prescaler output determine the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the timer counter modulo registers produces a PWM period of 256 times the internal bus clock period.

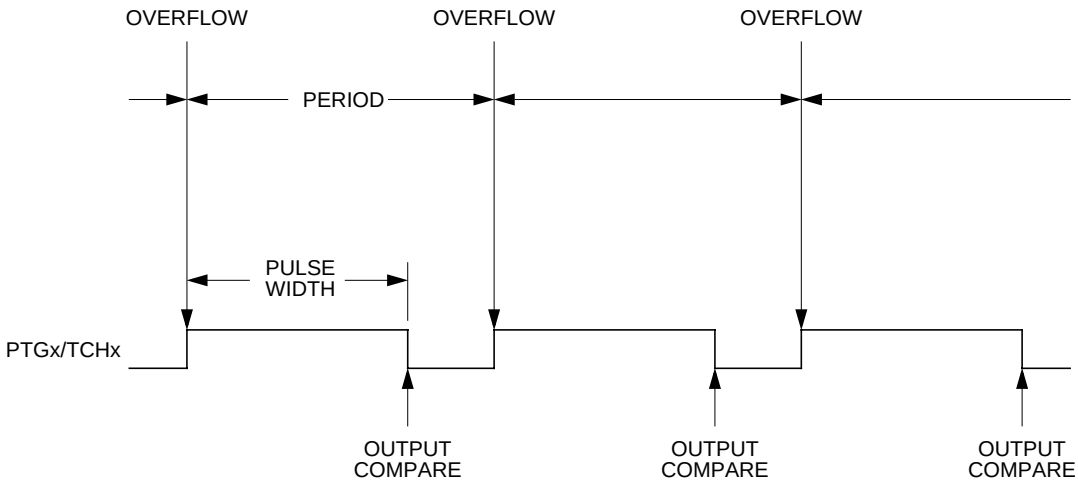


Figure 15-3. PWM Period and Pulse Width

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

15.4.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **15.4.4 Pulse Width Modulation (PWM)**. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the timer channel registers.

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

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

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable channel x timer overflow interrupts and write the new value in the timer overflow interrupt routine. The timer 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 percent duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare also can cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

15.4.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTG4/TCH0 pin. The timer channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in timer channel 0 status and control register (TSC0) links channel 0 and channel 1. The timer channel 0 registers initially control the pulse width on the PTG4/TCH0 pin. Writing to the timer channel 1 registers enables the timer channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the timer channel registers (0 or 1) that control the pulse width are written to last. TSC0 controls and monitors the buffered PWM function, and timer channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTG5/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 PTG6/TCH2 pin. The timer channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS2B bit in timer channel 2 status and control register (TSC2) links channel 2 and channel 3. The timer channel 2 registers initially control the pulse width on the PTG6/TCH2 pin. Writing to the timer channel 3 registers enables the timer channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the timer 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 timer channel 3 status and control register (TSC3) is unused.

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.*

15.4.4.3 PWM Initialization

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

1. In the timer status and control register (TSC):
   a. Stop the timer counter by setting the timer stop bit, TSTOP.
   b. Reset the timer counter by setting the timer reset bit, TRST.
2. In the timer counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the timer channel x registers (TCHxH:TCHxL), write the value for the required pulse width.
4. In timer channel x status and control register (TSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. (See **Table 15-2**.)
   b. Write 1 to the toggle-on-overflow bit, TOVx.

c. Write 1:0 (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 15-2**.)

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0 percent 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 timer status control register (TSC), clear the timer stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The timer channel 0 registers (TCH0H:TCH0L) initially control the buffered PWM output. Timer 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 timer channel 2 registers (TCH2H:TCH2L) initially control the PWM output. Timer 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 timer 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 percent duty cycle output.

Setting the CHxMAX bit generates a 100 percent duty cycle output. (See **15.9.4 Timer Channel Status and Control Registers**.)

15.5 Interrupts

These TIM sources can generate interrupt requests:

- TIM overflow flag (TOF) — TOF is set when the TIM counter value matches the value in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables the TOF flag to generate TIM overflow CPU interrupt requests. TOF and TOIE are in the TIM status and control register.
- TIM channel flags (CH3F–CH0F) — CHxF is set when an input capture or output compare occurs on channel x. The channel x interrupt enable bit, CHxIE, enables the CHxF flag to generate TIM channel x CPU interrupt requests. CHxF and CHxIE are in the TIM channel x status and control register.

15.6 Low-Power Modes

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

15.6.1 Wait Mode

The TIM remains active after the execution of a WAIT instruction. Any enabled CPU interrupt request from the TIM can bring the MCU out of wait mode. If the TIM is not required to bring the MCU out of wait mode, reduce power consumption by stopping the TIM before executing the WAIT instruction.

15.6.2 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 timer counter. Timer operation resumes when the MCU exits stop mode after an external interrupt.

15.7 TIM During Break Interrupts

A break interrupt stops the timer 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 **19.5.2 Break Address Registers**.)

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

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

15.8 I/O Signals

Port E shares five of its pins with the TIM. PTG3/TCLK is an external clock input to the timer prescaler. The four timer channel I/O pins are PTG4/TCH0, PTG5/TCH1, PTG6/TCH2, and PTG7/TCH3.

15.8.1 TIM Clock Pin (PTG3/TCLK)

PTG3/TCLK is an external clock input that can be the clock source for the timer counter instead of the prescaled internal bus clock. Select the PTG3/TCLK input by writing logic 1s to the three prescaler select bits, PS[2:0]. (See **15.9.1 Timer Status and Control Register**.) 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

15.8.2 Timer Channel I/O Pins (PTG4/TCH0–PTG7/TCH3)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTG4/TCH0 and PTG6/TCH2 can be configured as buffered output compare or buffered PWM pins.

15.9 I/O Registers

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

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

15.9.1 Timer Status and Control Register

The timer status and control register (TSC):

- Enables timer overflow interrupts
- Flags timer overflows
- Stops the timer counter
- Resets the timer counter and prescaler
- Prescales the timer counter clock

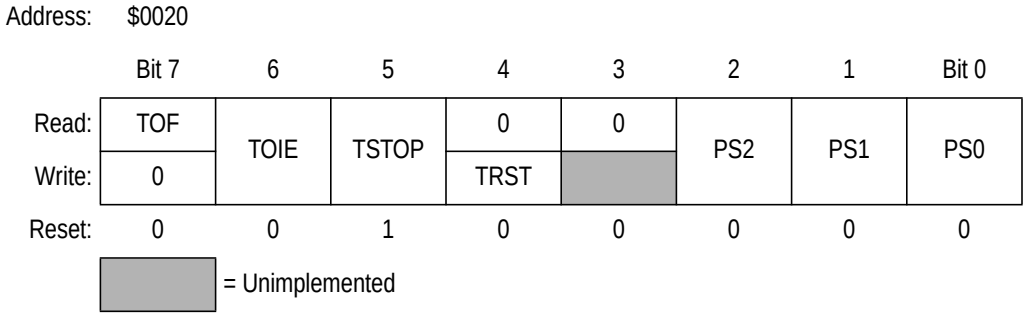


Figure 15-4. Timer Status and Control Register (TSC)

TOF — Timer Overflow Flag Bit

This read/write flag is set when the timer counter reaches the modulo value programmed in the timer counter modulo registers. Clear TOF by reading the timer status and control register when TOF is set and then writing a logic 0 to TOF. If another timer overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

1 = Timer counter has reached modulo value.
0 = Timer counter has not reached modulo value.

TOIE — Timer Overflow Interrupt Enable Bit

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

1 = Timer overflow interrupts enabled
0 = Timer overflow interrupts disabled

TSTOP — Timer Stop Bit

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

1 = Timer counter stopped
0 = Timer counter active

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

TRST — Timer Reset Bit

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

1 = Prescaler and timer counter cleared
0 = No effect

NOTE: *Setting the TSTOP and TRST bits simultaneously stops the timer counter at a value of $0000.*

PS2–PS0 — Prescaler Select Bits

These read/write bits select either the PTG3/TCLK pin or one of the seven prescaler outputs as the input to the timer counter as **Table 15-1** shows. Reset clears the PS2–PS0 bits.

Table 15-1. Prescaler Selection

PS2–PS0	TIM Clock Source
000	Internal bus clock
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	PTG3/TCLK

NOTE: *TCLK is a floating input pin. Do not select PS2–PS0 = 111.*

15.9.2 Timer Counter Registers

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

Setting the timer reset bit (TRST) also clears the timer counter registers.

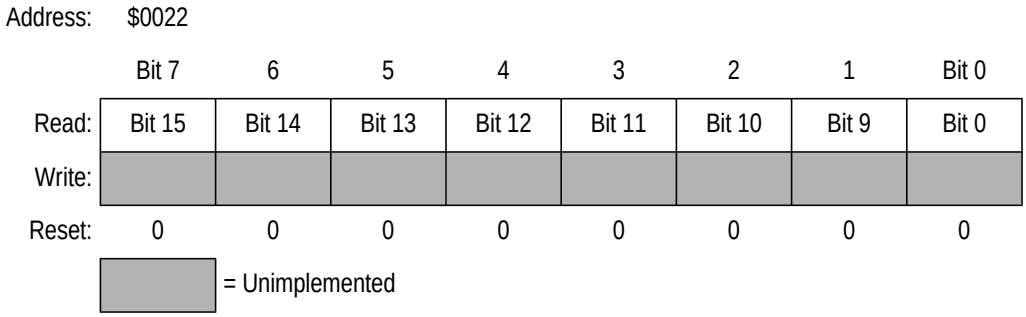


Figure 15-5. Timer Counter Register High (TCNTH)

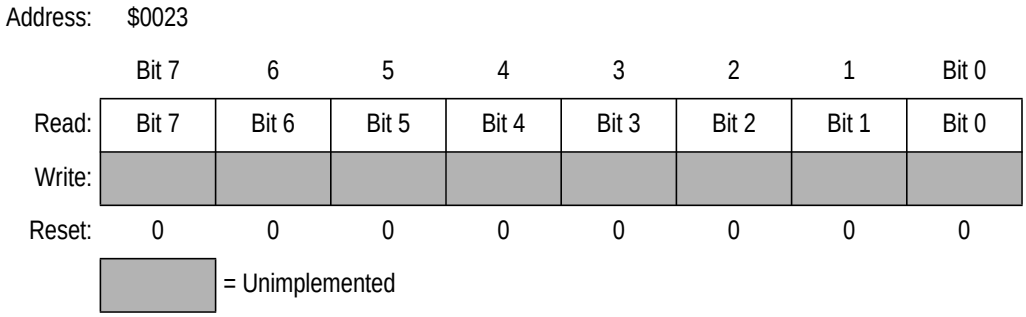


Figure 15-6. Timer Counter Register Low (TCNTL)

15.9.3 Timer Modulo Registers

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

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

Address: $0024

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

Figure 15-7. Timer Modulo Register High (TMODH)

Address: $0025

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

Figure 15-8. Timer Modulo Register Low (TMODL)

15.9.4 Timer Channel Status and Control Registers

Each of the timer channel status and control registers (TSC0–TSC3):

- 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 timer overflow
- Selects 100 percent PWM duty cycle

Address: $0026

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 15-9. Timer Channel 0 Status and Control Register (TSC0)

Address: $0029

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

= Unimplemented

Figure 15-10. Timer Channel 1 Status and Control Register (TSC1)

Address: $002C

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

Figure 15-11. Timer Channel 2 Status and Control Register (TSC2)

Address: $002F

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

= Unimplemented

Figure 15-12. Timer Channel 3 Status and Control Register (TSC3)

CHxF— Channel x Flag Bit

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

When TIM CPU interrupt requests are enabled (CHxIE:DMAxS = 1:0), clear CHxF by reading timer channel x status and control register with CHxF set and then writing a logic 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

When TIM DMA service requests are enabled (CHxIE:DMAxS = 1:1), clear CHxF by reading or writing to the low byte of the timer channel x registers (TCHxL).

Reset clears the CHxF bit. Writing a logic 1 to CHxF has no effect.

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

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIM CPU interrupts service requests on channel x. The DMAxS bit in the timer DMA select register selects channel x TIM DMA service requests or TIM CPU interrupt requests.

NOTE: *TIM DMA service requests cannot be used in buffered PWM mode. In buffered PWM mode, disable TIM DMA service requests by clearing the DMAxS bit in the timer DMA select register.*

Reset clears the CHxE bit.

1 = Channel x CPU interrupt requests and DMA service requests enabled

0 = Channel x CPU interrupt requests and DMA service requests disabled

NOTE: *Reading the high byte of the timer channel x registers (TCHxH) inhibits the CHxF flag until the low byte (TCHxL) is read.*

MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the timer channel 0 and timer channel 2 status and control registers.

Setting MS0B disables the channel 1 status and control register.

Setting MS2B disables the channel 3 status and control register.

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 15-2**.)

1 = Unbuffered output compare/PWM operation

0 = Input capture operation

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin. (See **Table 15-2**.) Reset clears the MSxA bit.

1 = Initial output level low

0 = Initial output level high

NOTE: *Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the timer 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 PTGx/TCHx is available as a general-purpose I/O pin. **Table 15-2** shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

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

MSxB and MSxA	ELSxB and ELSxA	Mode	Configuration
X0	00	Output preset	Set initial output level high
X1	00		Set initial output level low
XX	00	—	TCHx pin under port control
00	01	Input capture	Capture on rising edge only
00	10		Capture on falling edge only
00	11		Capture on rising or falling edge
XX	00	—	TCHx pin under port control; initial output low
01	01	Output compare or PWM	Toggle output on compare
01	10		Clear output on compare
01	11		Set output on compare
XX	00	—	TCHx pin under port control(1)
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:
1. Initial output high if MSxA = 0. Initial output low if MSxA = 1.

NOTE: *Before enabling a timer channel register for input capture operation, make sure that the PTG/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 timer 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 timer counter overflow.

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

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

NOTE: *Reading the high byte of the timer channel x registers prevents the channel x pin from toggling until the low byte is read.*

CHxMAX — Channel x Maximum (100 percent) PWM Duty Cycle Bit

This read/write bit forces the duty cycle of buffered and unbuffered PWM signals to 100 percent. As **Figure 15-13** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100 percent duty cycle level until the cycle after CHxMAX is cleared.

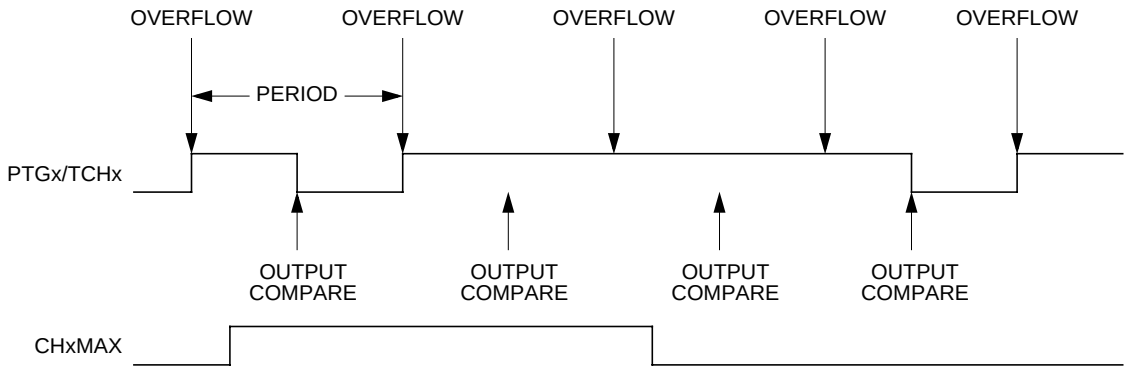


Figure 15-13. CHxMAX Latency

15.9.5 Timer Channel Registers

The timer channel registers (TCH0H/L–TCH3H/L) are read/write registers containing the captured timer counter value of the input capture function or the output compare value of the output compare function. The state of the timer channel registers after reset is unknown.

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the timer 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 timer channel x registers (TCHxH) inhibits output compares until the low byte (TCHxL) is written.

Address: $0027

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

Figure 15-14. Timer Channel 0 Register High (TCH0H)

Address: $0028

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

Figure 15-15. Timer Channel 0 Register Low (TCH0L)

Address: $002A

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

Figure 15-16. Timer Channel 1 Register High (TCH1H)

Address: $002B

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

Figure 15-17. Timer Channel 1 Register Low (TCH1L)

Address: $002D

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

Figure 15-18. Timer Channel 2 Register High (TCH2H)

Address: $002E

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

Figure 15-19. Timer Channel 2 Register Low (TCH2L)

Address: $0030

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

Figure 15-20. Timer Channel 3 Register High (TCH3H)

Address: $0031

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

Figure 15-21. Timer Channel 3 Register Low (TCH3L)

Section 16. Timebase Module (TIMTBX)

16.1 Contents

16.2 Introduction

The timebase module (TIMTBX) consists of a counter/divider clocked by the crystal clock which will generate periodic interrupts at user selectable rates.

16.3 Features

Timebase features include:

- Software programmable 1-Hz, 2-Hz, 4-Hz, and 8-Hz periodic interrupt
- Uses 32.0-kHz, 32.748-kHz or 38.4-kHz crystal

16.4 Functional Description

NOTE: *This module is designed to support 32-kHz, 32.768-kHz, and 38.4-kHz oscillators. Input crystal frequency is selected by the first write of the XTALR1:XTALR0 bits of the control register TBXCR, immediately after reset.*

This module can generate a periodic interrupt by dividing the crystal frequency, CGMXCLK. The counter is initialized to 0 when TBXON bit is cleared. The counter, shown in **Figure 16-1**, starts counting when the TBXON bit is set. When the counter overflows at the tap selected by TBXR1:TBXR0, the TBXIF bit gets set. If the TBXIE bit is set, an interrupt request is sent to the CPU. The TBXIF flag is cleared by writing a 1 to the TACK bit. The first time the TBXIF flag is set after enabling the timebase module, the interrupt is generated at approximately half of the overflow period. Subsequent events occur at the exact period.

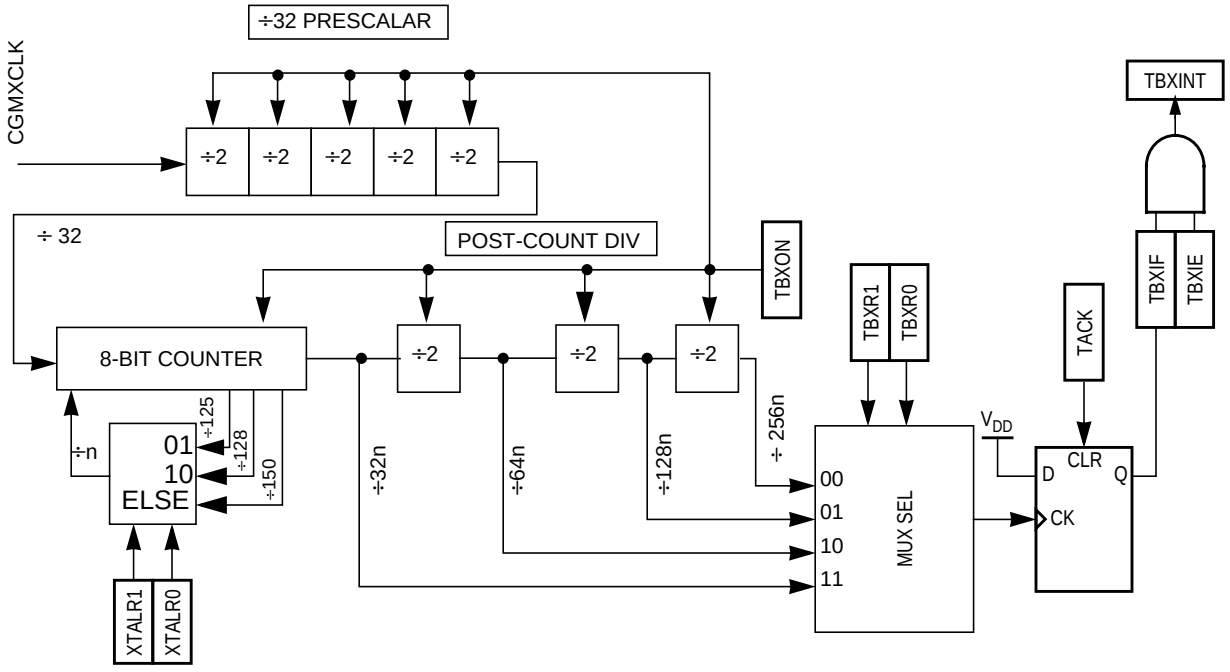


Figure 16-1. Timebase Block Diagram

16.5 Timebase Control Register Description

The timebase control register (TBXCR) is used to enable the timebase interrupts and set the rate.

Address: $0038

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TBXIF	TBXIE	TBXR1	TBXR0	0	TBXON	XTALR1	XTALR0
Write:					TACK			
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-2. Timebase Control Register (TBXCR)

TBXIF — Timebase Interrupt Flag

The read-only flag bit is set when the timebase counter has rolled over.

1 = Timebase interrupt pending
0 = Timebase interrupt not pending

TBXIE — Timebase Interrupt Enabled

The read/write bit enables the timebase interrupt when the TBIF bit becomes set. Reset clears the TBIE bit

1 = Timebase interrupt enabled
0 = Timebase interrupt disabled

TBXR1 and TBXR0 — Timebase Rate Selection

These read/write bits are used to select the rate of timebase interrupts as shown in **Table 16-1**.

Table 16-1. Timebase Rate Selection

TBXR1 and TBXR0	Divider	Timebase Interrupt Rate (Hz) vs (ms)	
00	1/256n	1	1000
01	1/128n	2	500
10	1/64n	4	250
11	1/32n	8	125

NOTE: *Do not change TBXR1:TBXR0 bits while the timebase is enabled (TBXON=1). Divider ratio, n, is defined by the XTALR[1:0] selection.*

TACK — Timebase ACKnowledge

The write-only TACK bit always reads 0. Writing a logic 1 to this bit clears TBIF, the timebase interrupt flag. Writing a logic 0 to this bit has no effect.

1 = Clear timebase interrupt flag
0 = No effect

TBXON — Timebase Enabled

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

1 = Timebase enabled
0 = Timebase disabled; counter initialized to 0

XTALR1 and XTALR0 — Input Crystal Frequency Selection

These bits are used to select the input crystal frequencies as shown in **Table 16-1**. They can be written **once only** following a reset. Any subsequent writes to these bits after the first valid write will be ignored. Default value upon reset is 00, so a crystal frequency of 38.4 kHz is assumed.

Table 16-2. Input Crystal Frequency Selection

XTALR1 and XTALR0	Crystal Frequencies (kHz)	Divider Ratio, n
00 default	38.4	150
01	32.0	125
10	32.768	128
11	38.4	150

NOTE: *It is recommended that these bits be written into at the beginning of the RESET sequence. To minimize the effect of timing discrepancies resulting from the choice of different input crystal frequencies. Although these bits are defaulted to 0, the user should write to these bits to prevent subsequent writes from unintentionally changing the crystal frequency selection.*

16.6 Interrupt

The timebase module can interrupt the CPU on a regular basis with a rate defined by TBXR1 and TBXR0. When the timebase counter chain rolls over, the counter chain overflow will generate a CPU interrupt request.

Interrupt must be acknowledged by writing a logic 1 to TACK bit.

16.7 Low-Power Modes

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

16.7.1 Wait Mode

The timebase module remains active after execution of the WAIT instruction. In wait mode, the timebase register is not accessible by the CPU.

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

16.7.2 Stop Mode

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

Timebase Module (TIMTBX)

Section 17. Input/Output (I/O) Ports

17.1 Contents

17.2 Introduction

56 bidirectional input-output (I/O) pins form seven parallel ports. All I/O pins are programmable as inputs or outputs.

NOTE: *Connect any unused I/O pins to an appropriate logic level, either V_{DD} or V_{SS}. Although the I/O ports do not require termination for proper operation, termination reduces excess current consumption and the possibility of electrostatic damage.*

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read: Write:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Reset:	Unaffected by reset							
$0004	Port A Data Direction Register (DDRA)	Read: Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Reset:	0	0	0	0	0	0	0	0
$0005	Port B Data Direction Register (DDRB)	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Reset:	0	0	0	0	0	0	0	0

Figure 17-1. I/O Port Registers

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0006	Port C Data Direction Register (DDRC)	Read: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Reset:	0	0	0	0	0	0	0	0
$0007	Port D Data Direction Register (DDRD)	Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE)	Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Reset:	Unaffected by reset							
$0009	Port F Data Register (PTF)	Read: Write:	PTF7	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
		Reset:	Unaffected by reset							
$000A	Port G Data Register (PTG)	Read: Write:	PTG7	PTG6	PTG5	PTG4	PTG3	PTG2	PTG1	PTG0
		Reset:	Unaffected by reset							
$000C	Port E Data Direction Register (DDRE)	Read: Write:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		Reset:	0	0	0	0	0	0	0	0
$000D	Port F Data Direction Register (DDRF)	Read: Write:	DDRF7	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
		Reset:	0	0	0	0	0	0	0	0
$000E	Port G Data Direction Register (DDRG)	Read: Write:	DDRG7	DDRG6	DDRG5	DDRG4	DDRG3	DDRG2	DDRG1	DDRG0
		Reset:	0	0	0	0	0	0	0	0

Figure 17-1. I/O Port Registers

17.3 Port A

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

17.3.1 Port A Data Register

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

Address: $0000

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
Reset:	Unaffected by reset							

Figure 17-2. 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.

17.3.2 Data Direction Register A

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

Address: $0004

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

Figure 17-3. Data Direction Register A (DDRA)

DDRA7–DDRA0 — Data Direction Register A Bits

These read/write bits control port A data direction. Reset clears DDRA7–DDRA0, 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 17-4 shows the port A I/O logic.

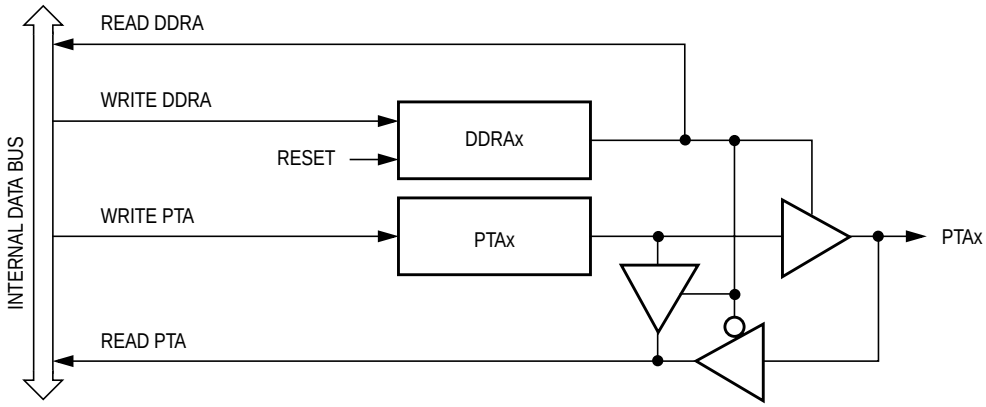


Figure 17-4. Port A I/O Circuit

When bit DDRAx is a logic 1, reading address $0000 reads the PTAx data latch. When bit DDRAx is a logic 0, reading address $0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 17-1** summarizes the operation of the port A pins.

Table 17-1. Port A Pin Functions

DDRA Bit	PTA Bit	I/O Pin Mode	Accesses to DDRA	Accesses to PTA	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRA7–DDRA0	Pin	PTA7–PTA0(3)
1	X	Output	DDRA7–DDRA0	PTA7 – PTA0	PTA7–PTA0

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

17.4 Port B

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

17.4.1 Port B Data Register

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

Address: $0001

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

Figure 17-5. Port B Data Register (PTB)

PTB7–PTB0 — 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.

17.4.2 Data Direction Register B

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

Address: $0005

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

Figure 17-6. Data Direction Register B (DDRB)

DDRB7–DDRB0 — 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 17-7 shows the port B I/O logic.

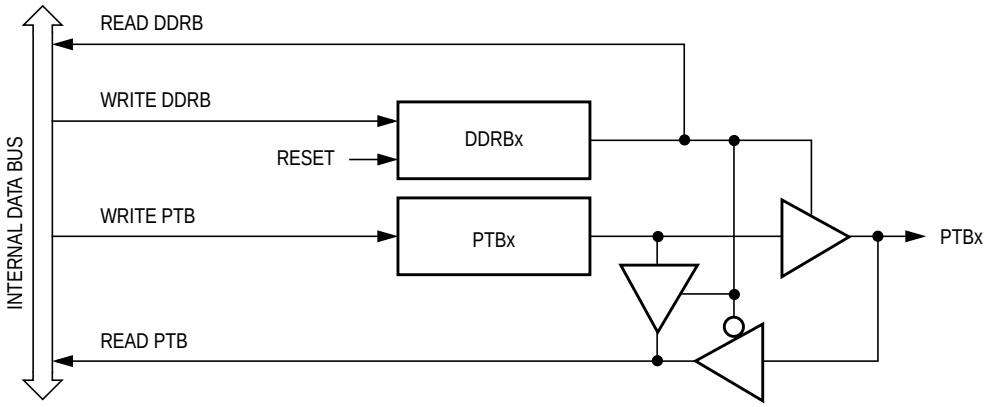


Figure 17-7. Port B I/O Circuit

When bit DDRBx is a logic 1, reading address $0001 reads the PTBx data latch. When bit DDRBx is a logic 0, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 17-2** summarizes the operation of the port B pins.

Table 17-2. Port B Pin Functions

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

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

17.5 Port C

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

17.5.1 Port C Data Register

The port C data register (PTC) contains a data latch for each of the port C pins.

Address: $0002

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
Reset:	Unaffected by reset							

Figure 17-8. Port C Data Register (PTC)

PTC7–PTC0 — 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.

17.5.2 Data Direction Register C

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

Address: $0006

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
Reset:	0	0	0	0	0	0	0	0

Figure 17-9. Data Direction Register C (DDRC)

DDRC7–DDRC0 — Data Direction Register C Bits

These read/write bits control port C data direction. Reset clears DDRC7–DDRC0, 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 17-10 shows the port C I/O logic.

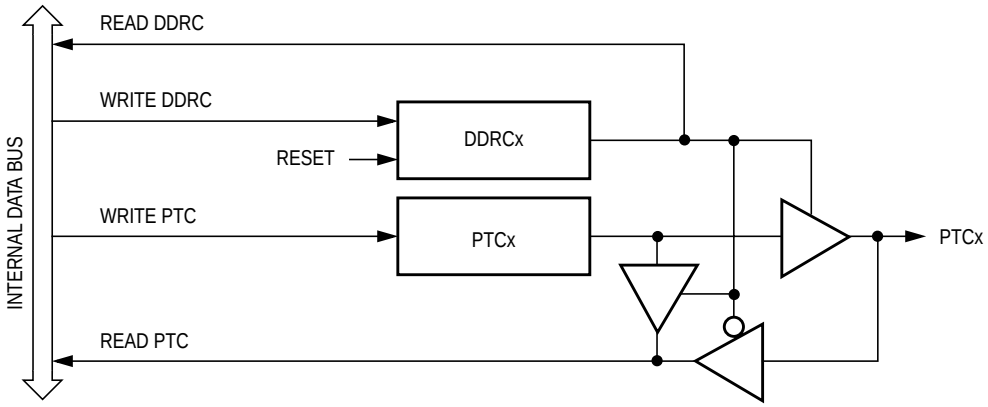


Figure 17-10. Port C I/O Circuit

When bit DDRCx is a logic 1, reading address $0002 reads the PTCx data latch. When bit DDRCx is a logic 0, reading address $0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 17-3** summarizes the operation of the port C pins.

Table 17-3. 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)	DDRC7–DDRC0	Pin	PTC7–PTC0(3)
1	X	Output	DDRC7–DDRC0	PTC7–PTC0	PTC7–PTC0

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

17.6 Port D

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

17.6.1 Port D Data Register

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

Address: $0003

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
Reset:	Unaffected by reset							

Figure 17-11. 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.

17.6.2 Data Direction Register D

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

Address: $0007

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

Figure 17-12. Data Direction Register D (DDRD)

DDRD7–DDRD0 — 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 17-13 shows the port D I/O logic.

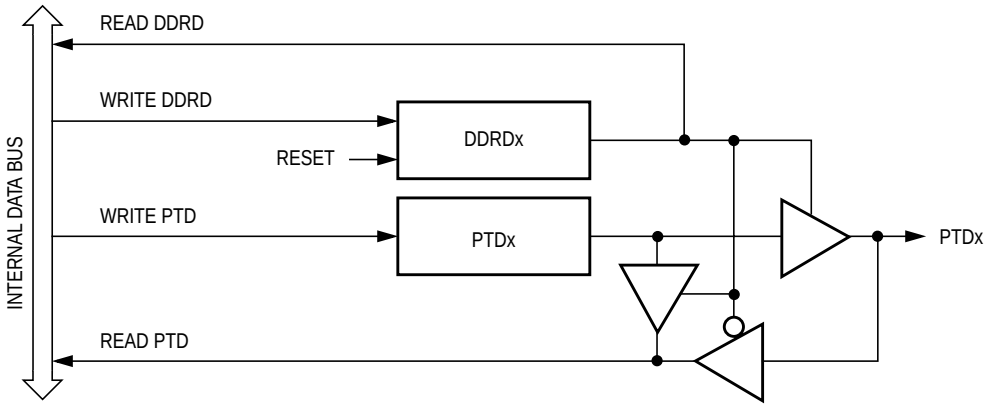


Figure 17-13. Port D I/O Circuit

When bit DDRDx is a logic 1, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic 0, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 17-4** summarizes the operation of the port D pins.

Table 17-4. Port D Pin Functions

DDRD Bit	PTD Bit	I/O Pin Mode	Accesses to DDRD	Accesses to PTD	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRD7–DDRD0	Pin	PTD7–PTD0(3)
1	X	Output	DDRD7–DDRD0	PTD7–PTD0	PTD7–PTD0

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

17.7 Port E

Port E is an 8-bit special function port that shares four of its pins with the A/D and the other four pins with SPI.

17.7.1 Port E Data Register

The port E data register (PTE) contains a data latch for each of the eight port pins.

Address: $0008

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
Reset:	Unaffected by reset							

Figure 17-14. Port E Data Register (PTE)

PTE7–PTE0 — Port E Data Bits

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

17.7.2 Data Direction Register E

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

Address: $000C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
Reset:	0	0	0	0	0	0	0	0

Figure 17-15. Data Direction Register E (DDRE)

DDRE7–DDRE0 — Data Direction Register E Bits

These read/write bits control port E data direction. Reset clears DDRE7–DDRE0, 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 17-16 shows the port E I/O logic.

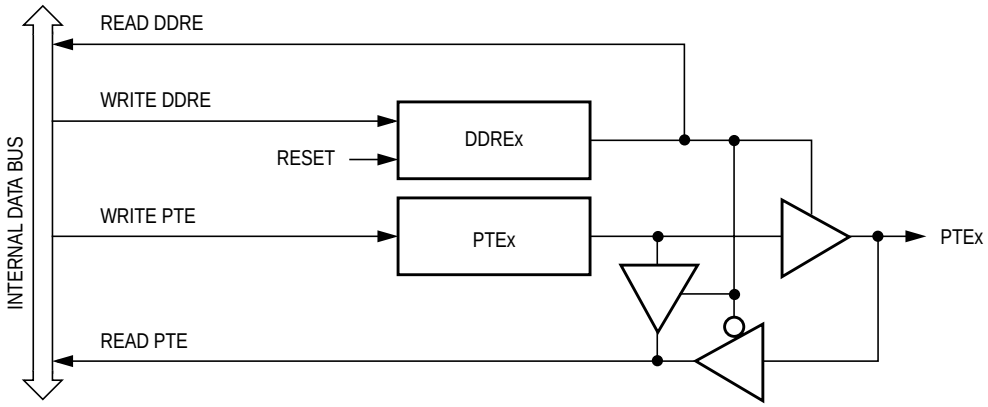


Figure 17-16. Port E I/O Circuit

When bit DDREx is a logic 1, reading address $0046 reads the PTEx data latch. When bit DDREx is a logic 0, reading address $0046 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 17-5** summarizes the operation of the port E pins.

Table 17-5. Port E Pin Functions

DDRE Bit	PTE Bit	I/O Pin Mode	Accesses to DDRE	Accesses to PTE	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRE7–DDRE0	Pin	PTE7–PTE0(3)
1	X	Output	DDRE7–DDRE0	PTE7–PTE0	PTE7–PTE0

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

17.8 Port F

Port F is an 8-bit special function port that shares with the keyboard interrupts.

17.8.1 Port F Data Register

The port F data register (PTF) contains a data latch for each of the port F pins.

Address: $0009

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTF7	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
Reset:	Unaffected by reset							

Figure 17-17. Port F Data Register (PTF)

PTF7–PTF0 — Port F Data Bits

These read/write bits are software-programmable. Data direction of each port F pin is under the control of the corresponding bit in data direction register F.

17.8.2 Data Direction Register F

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

Address: $000D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRF7	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
Reset:	0	0	0	0	0	0	0	0

Figure 17-18. Data Direction Register F (DDRF)

DDRF7–DDRF0 — Data Direction Register F Bits

These read/write bits control port F data direction. Reset clears DDRF7–DDRF0, configuring all port F pins as inputs.

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

NOTE: *Avoid glitches on port F pins by writing to the port F data register before changing data direction register F bits from 0 to 1.*

Figure 17-19 shows the port F I/O logic.

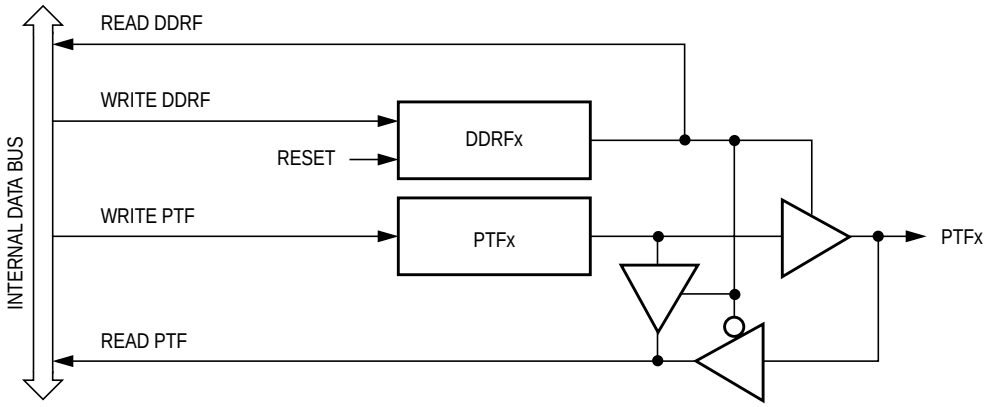


Figure 17-19. Port F I/O Circuit

When bit DDRFx is a logic 1, reading address $0047 reads the PTFx data latch. When bit DDRFx is a logic 0, reading address $0047 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 17-6** summarizes the operation of the port F pins.

Table 17-6. Port F Pin Functions

DDRF Bit	PTF Bit	I/O Pin Mode	Accesses to DDRF	Accesses to PTF	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRF7–DDRF0	Pin	PTF7–PTF0(3)
1	X	Output	DDRF7–DDRF0	PTF7–PTF0	PTF7–PTF0

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

17.9 Port G

Port G is am 8-bit special function port that shares five of its pins with the timer and two of its pins with SCI.

17.9.1 Port G Data Register

The port G data register (PTG) contains a data latch for each of the port G pins.

Address: $000A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTG7	PTG6	PTG5	PTG4	PTG3	PTG2	PTG1	PTG0
Reset:	Unaffected by reset							

Figure 17-20. Port G Data Register (PTG)

PTG7–PTG0 — Port G Data Bits

These read/write bits are software-programmable. Data direction of each port G pin is under the control of the corresponding bit in data direction register G.

17.9.2 Data Direction Register G

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

Address: $000E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRG7	DDRG6	DDRG5	DDRG4	DDRG3	DDRG2	DDRG1	DDRG0
Reset:	0	0	0	0	0	0	0	0

Figure 17-21. Data Direction Register G (DDRG)

DDRG7–DDRG0 — Data Direction Register G Bits

These read/write bits control port G data direction. Reset clears DDRG7–DDRG0, configuring all port G pins as inputs.

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

NOTE: *Avoid glitches on port G pins by writing to the port G data register before changing data direction register G bits from 0 to 1.*

Figure 17-22 shows the port G I/O logic.

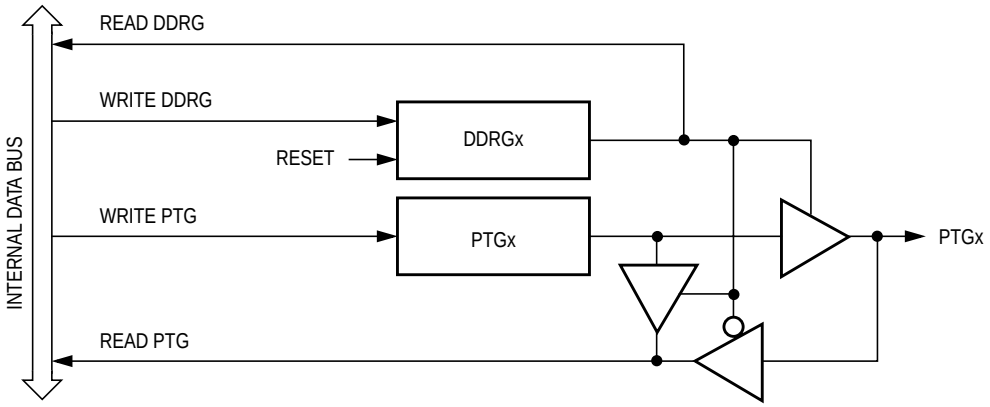


Figure 17-22. Port G I/O Circuit

When bit DDRGx is a logic 1, reading address $0048 reads the PTGx data latch. When bit DDRGx is a logic 0, reading address $0048 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 17-7** summarizes the operation of the port G pins.

Table 17-7. Port G Pin Functions

DDRG Bit	PTG Bit	I/O Pin Mode	Accesses to DDRG	Accesses to PTG	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRG7–DDRG0	Pin	PTG7–PTG0(3)
1	X	Output	DDRG7–DDRG0	PTG7–PTG0	PTG7–PTG0

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

Input/Output (I/O) Ports

Section 18. Monitor ROM (MON)

18.1 Contents

18.2 Introduction

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

18.3 Features

Features of the monitor ROM include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between monitor ROM and host computer
- Standard mark/space non-return-to-zero (NRZ) communication with host computer
- 4800 baud–28.8 kbaud communication with host computer
- Execution of code in RAM or ROM

18.4 Functional Description

The monitor ROM receives and executes commands from a host computer. **Figure 18-1** 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 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.

18.4.1 Entering Monitor Mode

Table 18-1 shows the pin conditions for entering monitor mode.

Table 18-1. Mode Selection

$\overline{IRQ1}$ Pin	PTC0 Pin	PTC1 Pin	PTA0 Pin	PTC3 Pin	Mode	CGMOUT	Bus Frequency
$V_{DD} + V_{HI}$	1	0	1	1	Monitor	$\frac{CGMXCLK}{2}$ or $\frac{CGMVCLK}{2}$	$\frac{CGMOUT}{2}$
$V_{DD} + V_{HI}$	1	0	1	0	Monitor	CGMXCLK	$\frac{CGMOUT}{2}$

Enter monitor mode by either

- Executing a software interrupt instruction (SWI) or
- Applying a logic 0 and then a logic 1 to the $\overline{RST}$ pin

The MCU sends a break signal (10 consecutive logic 0s) 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.

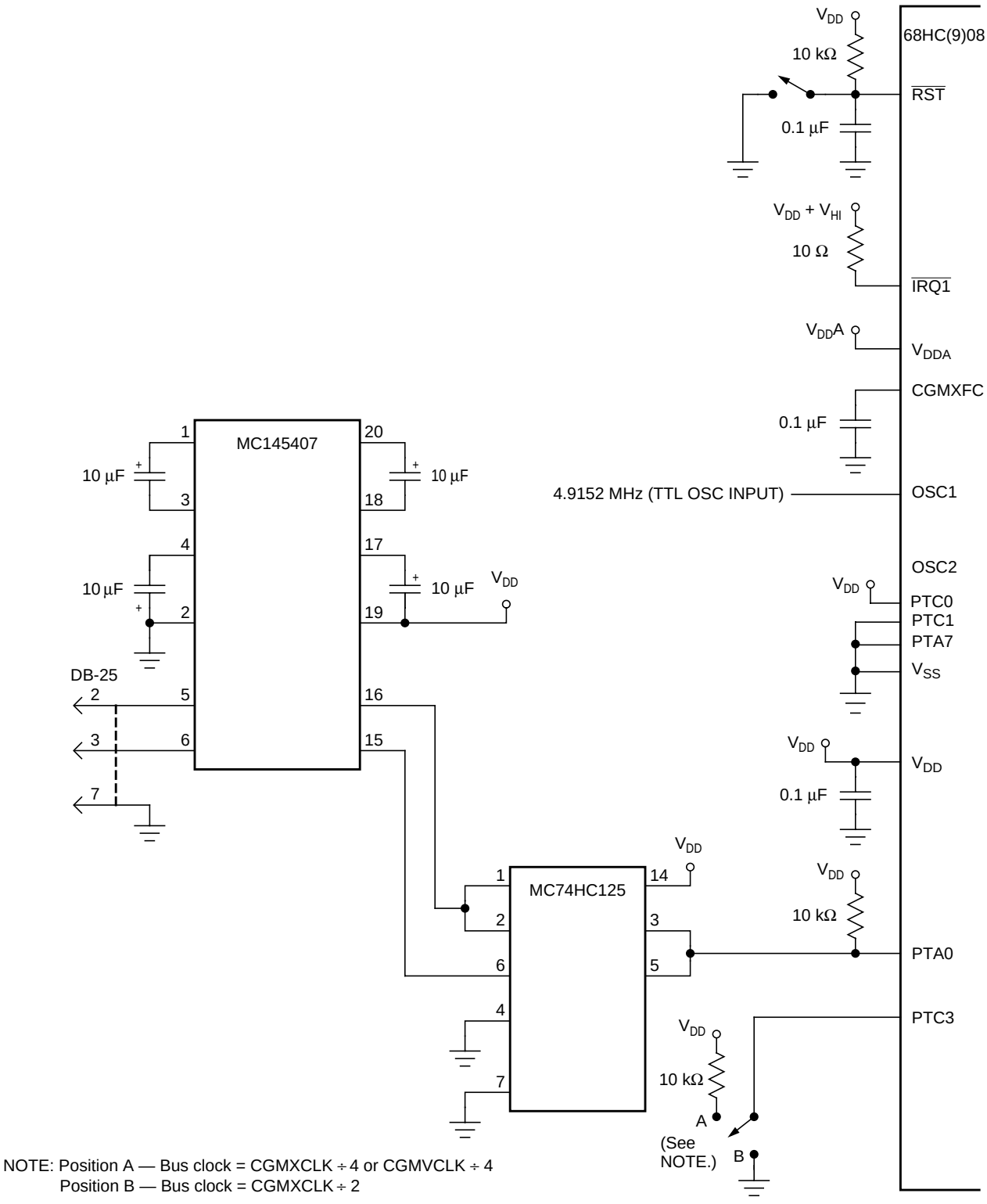


Figure 18-1Monitor Mode Circuit

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 $\overline{\text{IRQ1}}$ pin or the $\overline{\text{RST}}$ pin. Burn-in can be accomplished by uploading code to RAM in monitor mode, then pulling PTA0 low and executing a reset. This causes a jump to RAM.(See **Section 7. System Integration Module (SIM)** for more information on modes of operation.)

NOTE: *Holding the PTC3 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 percent duty cycle at maximum bus frequency.*

Table 18-2 is a summary of the differences between user mode and monitor mode.

Table 18-2. 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

Note:
1. If the high voltage (V_{DD} + V_{HI}) is removed from the $\overline{\text{IRQ1}}$ pin or the $\overline{\text{RST}}$ 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.

18.4.2 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. (See **Figure 18-2** and **Figure 18-3**.)

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

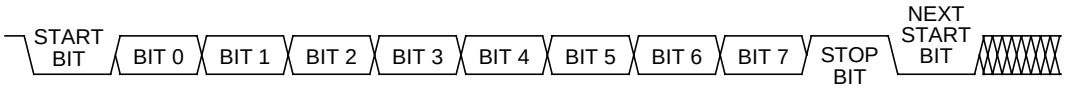


Figure 18-2. Monitor Data Format

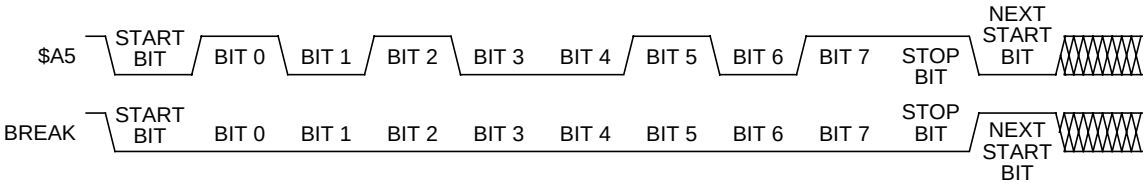


Figure 18-3. Sample Monitor Waveforms

18.4.3 Echoing

As shown in **Figure 18-4**, the monitor ROM immediately echoes each received byte back to the PTA0 pin for error checking.

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

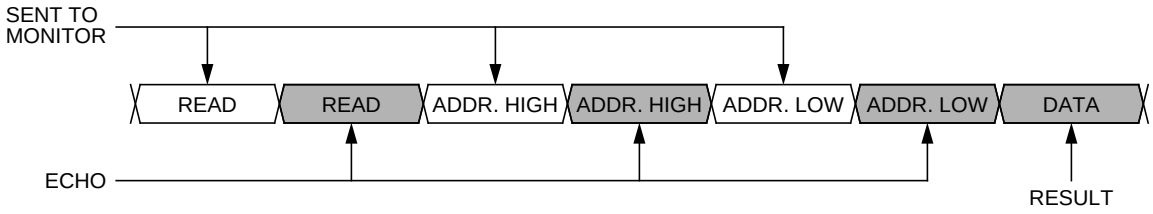


Figure 18-4. Read Transaction

18.4.4 Break Signal

A start bit followed by nine low bits is a break signal. (See **Figure 18-5**.) When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits before echoing the break signal.

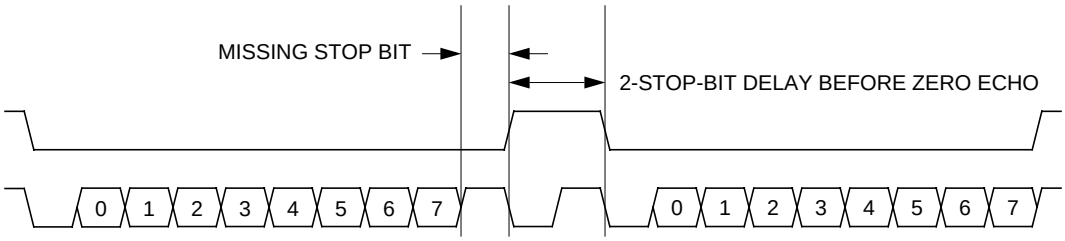


Figure 18-5. Break Transaction

18.4.5 Commands

The monitor ROM uses these commands:

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

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

Table 18-3. 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	

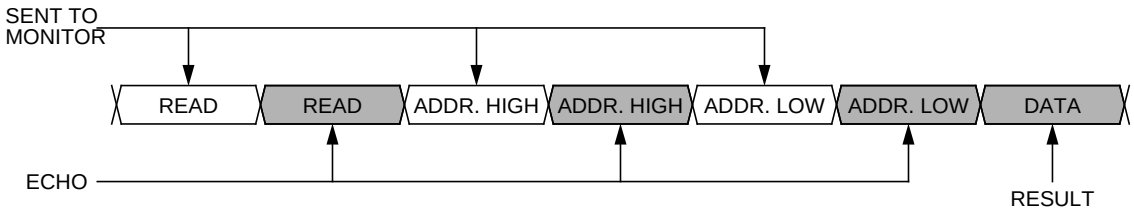


Table 18-4. WRITE (Write Memory) Command

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

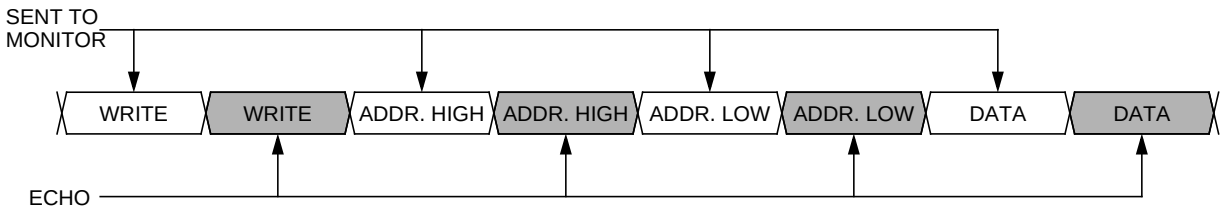


Table 18-5. READ (Indexed Read) Command

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	

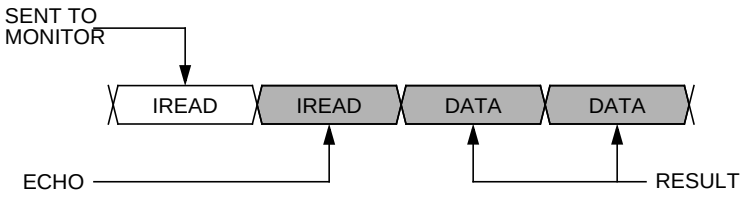


Table 18-6. IWRITE (Indexed Write) Command

Description	Write to last address accessed + 1
Operand	Specifies single data byte
Data Returned	None
Opcode	$19
Command Sequence	

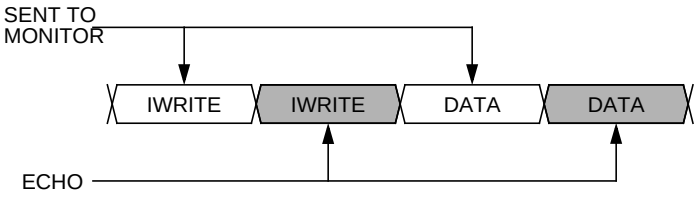


Table 18-7. READSP (Read Stack Pointer) Command

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

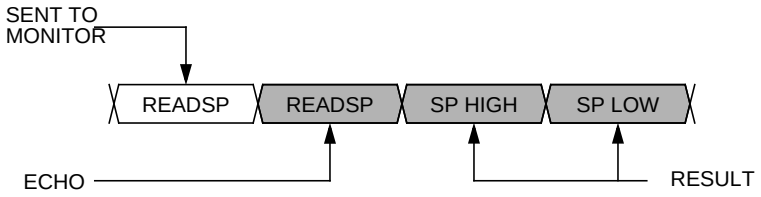
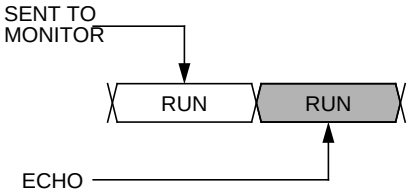


Table 18-8. RUN (Run User Program) Command

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



18.4.6 Baud Rate

With a 4.9152-MHz crystal and the PTC3 pin at logic 1 during reset, data is transferred between the monitor and host at 4800 baud. If the PTC3 pin is at logic 0 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 MUL7–MUL4 bits in the PLL programming register (PPG). (See **Section 4. Clock Generator Module (CGMB)**.)

Table 18-9. 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 19. Break Module

19.1 Contents

19.2 Introduction

This section describes the break module. The break module can generate a break interrupt that stops normal program flow at a defined address to enter a background program.

19.3 Features

Features of the break module include:

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

19.4 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).

These events can cause a break interrupt to occur:

- A CPU-generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a logic 1 to the BRKA bit in the break status and control register.
- Software writes a logic 1 to the break active bit (BRKA) 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 19-1** shows the structure of the break module.

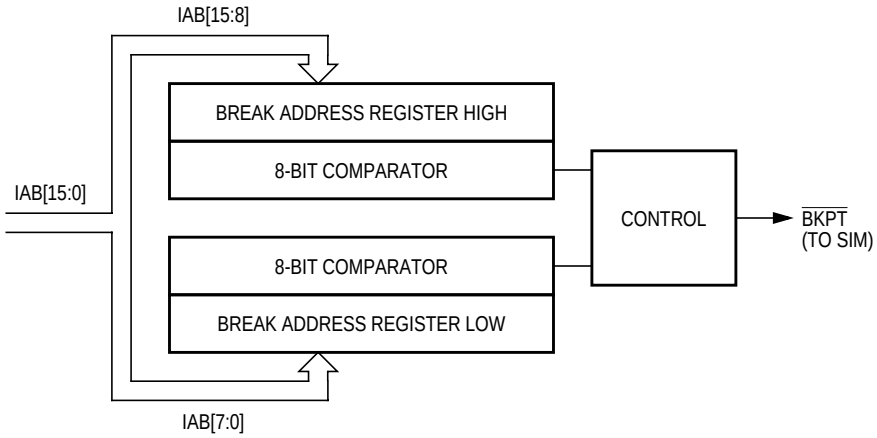


Figure 19-1. Break Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE0D	Break Address Register High (BRKH)	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$FE0E	Break Address Register Low (BRKL)	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$FE0F	Break Status/Control Register (BRKSCR)	Read: Write:	BRKE	BRKA						
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 19-2. Break I/O Register Summary

19.4.1 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** and see the **Break Interrupts** subsection for each module.)

19.4.2 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.

19.4.3 TIM During Break Interrupts

A break interrupt stops the timer counter.

19.4.4 COP During Break Interrupts

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

19.4.5 COP During Break

If the $\overline{RST}$ pin is at $2 \times V_{DD}$ during a break, the COP counter stops. If the $\overline{RST}$ pin falls to logic 1 during break, the COP resumes operation.

19.5 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)

19.5.1 Break Status and Control Register

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

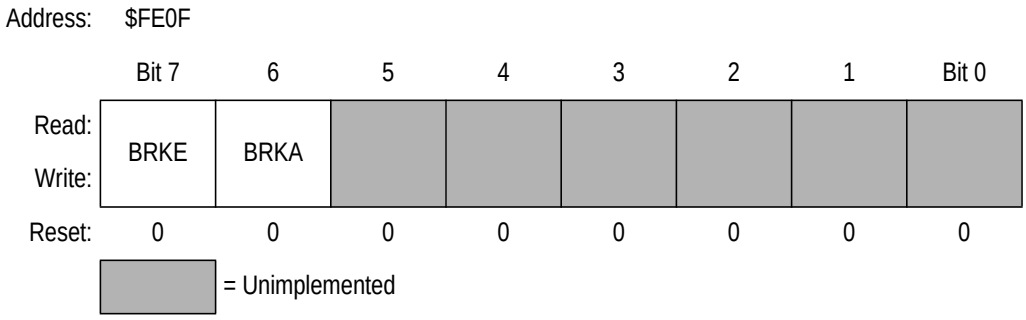


Figure 19-3. Break Status and Control Register (BRKSCR)

BRKE — Break Enable Bit

This bit enables breaks on break address register matches. Clear BRKE by writing a logic 0 to bit 7.

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

BRKA — Break Active Bit

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

1 = Break address match
0 = No break address match

19.5.2 Break Address Registers

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

Address: $FE0D

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

Figure 19-4. Break Address Register High (BRKH)

Address: $FE0E

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

Figure 19-5. Break Address Register Low (BRKL)

19.6 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 **7.8.1 SIM Break Status Register**) Clear the SBSW bit by writing logic 0 to it.

Section 20. External Interrupt Module (IRQ)

20.1 Contents

20.2 Introduction

This section describes the external interrupt module which supports external interrupt functions.

20.3 Features

Features of the IRQ module include:

- Two dedicated external interrupt pins ($\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$)
- Separate IRQ1 and IRQ2 interrupt masks
- Hysteresis buffers

20.4 Functional Description

A logic 0 applied to any of the external interrupt pins can latch a CPU interrupt request. **Figure 20-1** 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 these actions occurs:

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the latch that caused the vector fetch.
- Software clear — Software can clear an interrupt latch by writing to the appropriate acknowledge bit in the interrupt status and control register (ISCR). Writing a logic 1 to the ACK1 bit clears the IRQ1 latch. Writing a logic 1 to the ACK2 bit clears the IRQ2 interrupt latch.
- Reset — A reset automatically clears both interrupt latches.

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 $\overline{\text{IRQ2}}$ interrupt pin.

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

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

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

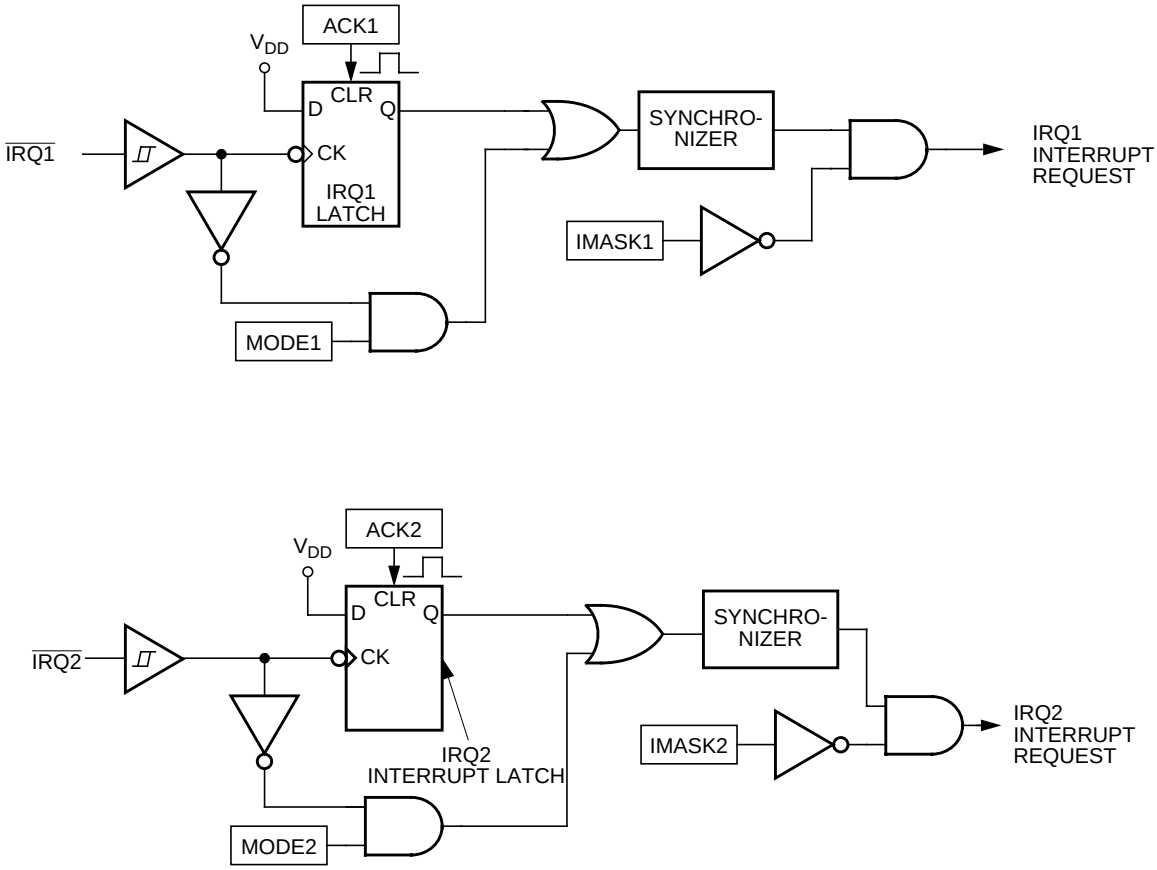


Figure 20-1. IRQ Module Block Diagram

The vector fetch or software clear may occur before or after the interrupt pin returns to logic 1. As long as the pin is low, the interrupt request remains pending.

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 20-2**.)*

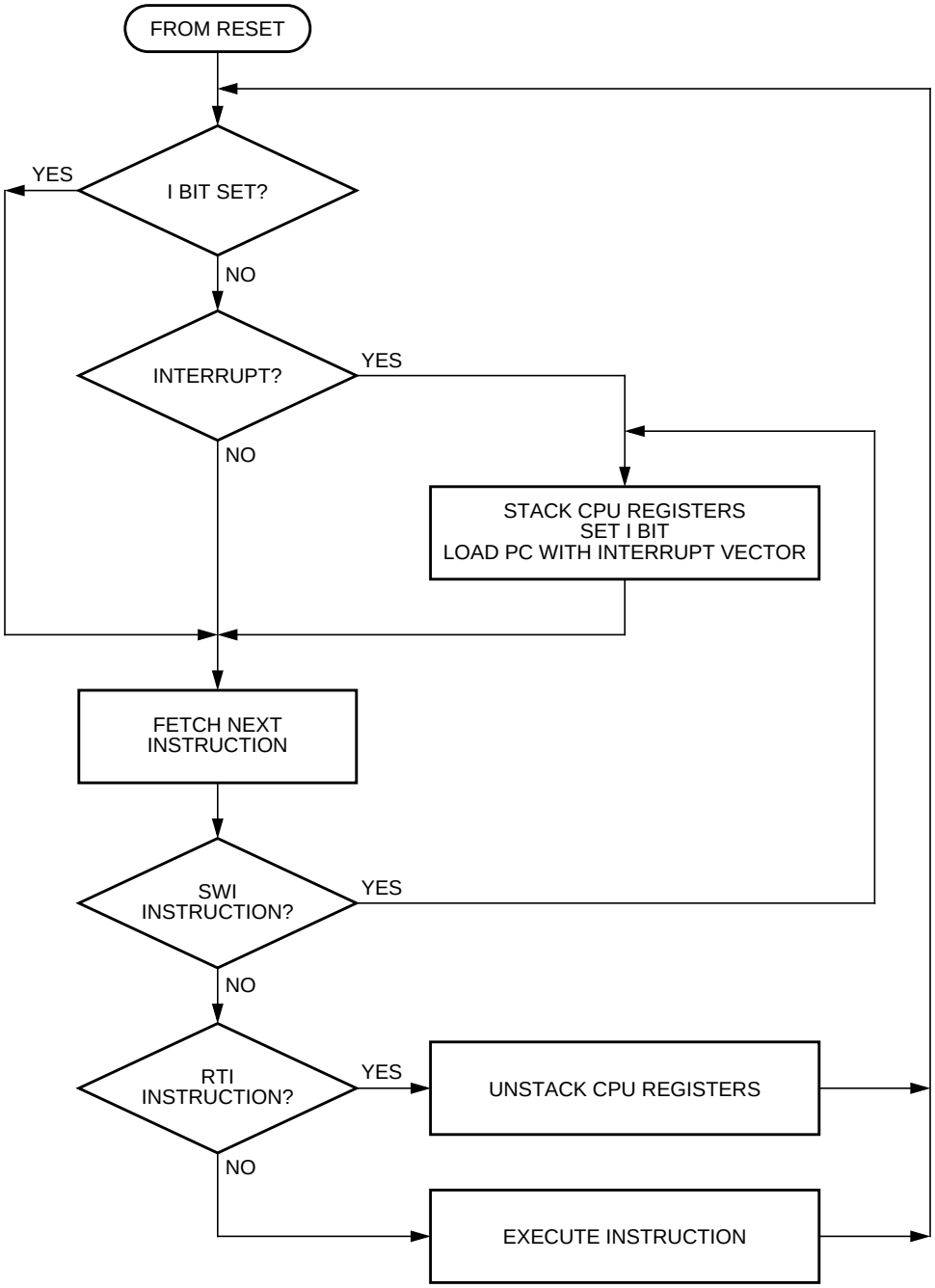


Figure 20-2. IRQ Interrupt Flowchart

20.4.1 $\overline{\text{IRQ1}}$ Pin

A logic 0 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 these actions must occur to clear the IRQ1 latch:

- Vector fetch, 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 1 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 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 $FFFA and $FFFB.
- Return of the $\overline{\text{IRQ1}}$ pin to logic 1 — As long as the $\overline{\text{IRQ1}}$ pin is at logic 0, the IRQ1 latch remains set.

The vector fetch or software clear and the return of the $\overline{\text{IRQ1}}$ pin to logic 1 may occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ1}}$ pin is at logic 0.

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.

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.*

20.4.2 $\overline{\text{IRQ2}}$ Pin

A logic 0 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 these actions must occur to clear the IRQ2 interrupt latch:

- Vector fetch, 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 1 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 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 $FFF8 and $FFF9.
- Return of the $\overline{\text{IRQ2}}$ pin to logic 1 — As long as the $\overline{\text{IRQ2}}$ pin is at logic 0, the IRQ2 interrupt latch remains set.

The vector fetch or software clear and the return of the $\overline{\text{IRQ2}}$ pin to logic 1 may occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ2}}$ pin is at logic 0.

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.

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

20.5 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 **7.8.3 SIM Break Flag Control Register**.)

To allow software to clear the IRQ1 latch and the IRQ2 interrupt latch during a break interrupt, write a logic 1 to the BCFE bit. If a latch is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect the latches during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (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. (See **20.6 IRQ Status and Control Register**.)

20.6 IRQ Status and Control Register

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

- Shows current 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

Address: $001D

	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 20-3. IRQ Status and Control Register (ISCR)

IRQ2F — IRQ2 Flag Bit

This read-only bit is high when an IRQ2 CPU interrupt is pending.

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

ACK2 — IRQ2 Interrupt Request Acknowledge Bit

Writing a logic 1 to this write-only bit clears the IRQ2 interrupt latch. ACK2 always reads as logic 0. Reset clears ACK2.

IMASK2 — IRQ2 Interrupt Mask Bit

Writing a logic 1 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 request disabled
0 = $\overline{\text{IRQ2}}$ pin interrupt request enabled

MODE2 — IRQ2 Interrupt Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ2}}$ interrupt pin. Reset clears MODE2.

1 = IRQ2 interrupt request on falling edges and low levels
0 = IRQ2 interrupt request on falling edges only

IRQ1F — IRQ1 Flag Bit

This read-only bit is high when an IRQ1 CPU interrupt is pending.

1 = $\overline{\text{IRQ1}}$ pin interrupt pending
0 = $\overline{\text{IRQ1}}$ pin interrupt not pending

ACK1 — IRQ1 Interrupt Request Acknowledge Bit

Writing a logic 1 to this write-only bit clears the IRQ1 latch. ACK1 always reads as logic 0. Reset clears ACK1.

IMASK1 — IRQ1 Interrupt Mask Bit

Writing a logic 1 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 request on falling edges and low levels
0 = $\overline{\text{IRQ1}}$ interrupt request on falling edges only

Section 21. Alert Output Generator (ALR)

21.1 Contents

21.2 Introduction

This section describes the alert output generator (ALR), which provides 14 software selectable square-wave output frequencies.

21.3 Features

Features of the ALR module include:

- 14 software selectable audio alert tone outputs
- 4-bit software-selectable, sound pressure level control

21.4 Functional Description

This system will be used to generate alert tones as the output signal ALERT. The audio alert tone generator is controlled by the four control bits shown in **21.4.1 Alert Control Register**. This allows 14 possible frequencies to drive the alert output. The zero state acts as an off mode and places the output in a high-impedance mode and an ON mode places the output in ground state. (See **Table 21-1**.).

NOTE: *The alert module is enabled only when the on-chip phase-locked loop (PLL) is engaged. During wait mode, the alert control register (ALCR) register should be programmed with value $00 to ensure a three-state output and eliminate any residual output from the audio frequency generator.*

Table 21-1. Audio Alert Tone Generator divider Ratios

AL3–AL0	Audio Alert Generator Frequencies at Given f_{OSC}			
	f_{OSC}	32.768 kHz	32.000 kHz	38.4 kHz
0000	Off	Hi-Z	Hi-Z	Hi-Z
0001	$f_{OSC} \div 32$	1024	1000	1200
0010	$f_{OSC} \div 8$	4096	4000	4800
0011	$f_{OSC} \div 16$	2048	2000	2400
0100	$f_{OSC} \div 6$	5461	5333	6400
0101	$f_{OSC} \div 12$	2730	2666	3200
0110	$f_{OSC} \div 24$	1365	1333	1600
0111	$f_{OSC} \div 48$	683	667	800
1000	$f_{OSC} \div 10$	3276	3200	3840
1001	$f_{OSC} \div 20$	1638	1600	1920
1010	$f_{OSC} \div 40$	819	800	960
1011	$f_{OSC} \div 80$	410	400	480
1100	$f_{OSC} \div 14$	2341	2285	2743
1101	$f_{OSC} \div 28$	1170	1143	1371
1110	$f_{OSC} \div 56$	585	571	686
1111	Off	V_{SS}	V_{SS}	V_{SS}

21.4.1 Alert Control Register

The alert control register (ALCR) bits are defined here.

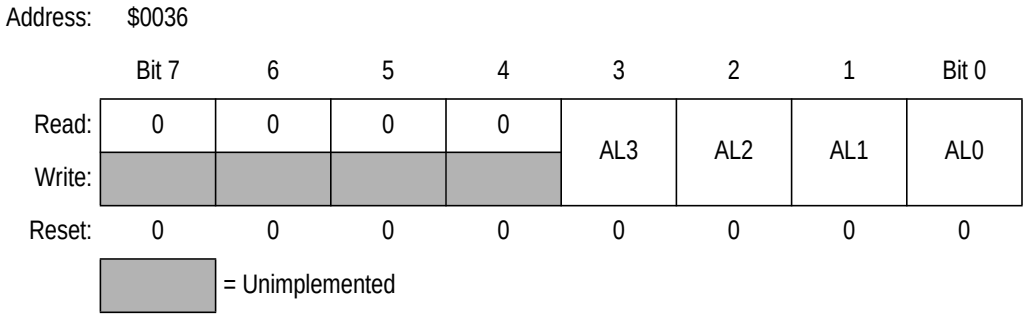


Figure 21-1. Alert Control Register (ALCR)

AL3–AL0 — Alert Frequency Select

The values of these bits determine the frequency of the alert output (see **Table 21-1**). If these bits are set to all 0s, the ALERT output will be high impedance (off). If set to all 1s, the ALERT output will be at V_{SS} (logic 0).

Reset clears these bits, turning this output off to the high-impedance state.

21.4.2 Sound Pressure Level Circuit

The sound pressure level (SPL) control register is used to control the volume of the alert transducer. A high frequency clock signal is used to modulate the normal alert output frequency, this modulation reduces the amplitude of the frequency components in the audible range while adding new frequencies outside the audible range.

This feature causes a significant reduction in the SPL of the transducer. A broad range of volume control is obtained by altering the duty cycle of the high frequency modulation signal. The SPL control register is software programmable to allow user to select different frequencies with different duty cycles. (See **Figure 21-2**.)

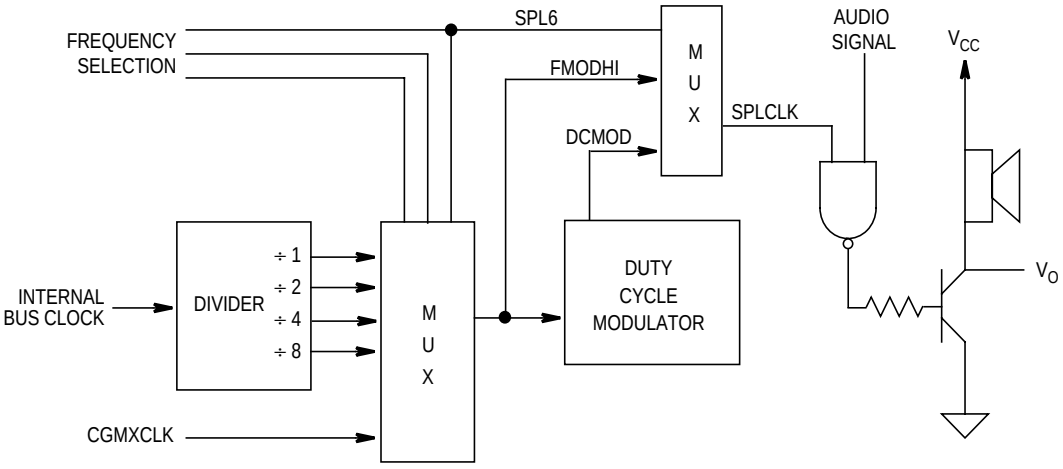


Figure 21-2. Block Diagram of SPL Reduction Circuit

21.4.3 Alert Data Register

The alert data register (ALDR) bits are defined here.

Address: $0037

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SPL7	SPL6	SPL5	SPL4	SPL3	SPL2	SPL1	SPL0
Reset:	0	0	0	0	0	0	0	0

Figure 21-3. Alert Data Register (ALDR)

SPL7

This bit selects between 16-phase duty cycle modulation if bit SPL7 = 0 and 8-phase duty cycle modulation if bit SPL7 = 1.

SPL6

This bit selects between the output (DCMOD) of the duty cycle modulator when SPL6 = 0 and the clock FMODHI when SPL6 = 1 to send to the output SPLCLK to modulate the alert output. (Refer to **Figure 21-2**.)

SPL5 and SPL4

These bits are used to control the frequency divider selections (divided by 1, 2, and 4), and, along with bit SPL6 in one case where both SPL4 and SPL5 are high, to select between a CPU clock divided by eight and the crystal clock. (See **Table 21-2**.)

Table 21-2. Clock Divider and Modulator Selections

SPL6	SPL5	SPL4	FMODHI	SPLCLK
0	0	0	Bus clock/1	DCMOD
0	0	1	Bus clock/2	DCMOD
0	1	0	Bus clock/4	DCMOD
0	1	1	Bus clock/8	DCMOD
1	0	0	Bus clock/1	FMODHI

Table 21-2. Clock Divider and Modulator Selections (Continued)

SPL6	SPL5	SPL4	FMODHI	SPLCLK
1	0	1	Bus clock/2	FMODHI
1	1	0	Bus clock/4	FMODHI
1	1	1	CGMXCLK	FMODHI

SPL3–SPL0

These bits control the duty cycle of the modulation signal, as shown in **Table 21-3**.

Table 21-3. Duty Cycle Selection

SPL3	SPL2	SPL1	SPL0	Duty Cycle 8-Phase Count (SPL7 = 1)	Duty Cycle 16-Phase Count (SPL7 = 0)
0	0	0	0	SPL disabled	SPL disabled
0	0	0	1	1/8 high 7/8 low	1/16 high 15/16 low
0	0	1	0	2/8 high 6/8 low	2/16 high 14/16 low
0	0	1	1	3/8 high 5/8 low	3/16 high 13/16 low
0	1	0	0	4/8 high 4/8 low	4/16 high 12/16 low
0	1	0	1	5/8 high 3/8 low	5/16 high 11/16 low
0	1	1	0	6/8 high 2/8 low	6/16 high 10/16 low
0	1	1	1	7/8 high 1/8 low	7/16 high 9/16 low
1	0	0	0	SPI disabled	8/16 high 8/16 low
1	0	0	1	1/8 high 7/8 low	9/16 high 7/16 low
1	0	1	0	2/8 high 6/8 low	10/16 high 6/16 low
1	0	1	1	3/8 high 5/8 low	11/16 high 5/16 low
1	1	0	0	4/8 high 4/8 low	12/16 high 4/16 low
1	1	0	1	5/8 high 3/8 low	13/16 high 3/16 low
1	1	1	0	6/8 high 2/8 low	14/16 high 2/16 low
1	1	1	1	7/8 high 1/8 low	15/16 high 1/16 low

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.

The MCU contains circuitry to protect the inputs against damage from high static voltages; however, do not apply voltages higher than those shown in this table. Keep V_{In} and V_{Out} within the range $V_{SS} \le (V_{In}$ or $V_{Out}) \le V_{DD}$. Connect unused inputs to the appropriate voltage level, either V_{SS} or V_{DD}.

Rating	Symbol	Value	Unit
Supply voltage	V_{DD}	–0.3 to +3.6	V
Input voltage	V_{In}	V_{SS} –0.3 to V_{DD} +0.3	V
Maximum current per pin excluding V_{DD} and V_{SS}	I	± 25	mA
Storage temperature	T_{STG}	–55 to +150	°C
Maximum current out of V_{SS}	I_{MVSS}	100	mA
Maximum current into V_{DD}	I_{MVDD}	100	mA

Note: Voltages are referenced to V_{SS}.

NOTE: *This device is not guaranteed to operate properly at the maximum ratings. Refer to* ***22.6 3.0-Volt DC Electrical Characteristics*** *and* ***22.7 2.0-Volt DC Electrical Characteristics*** *for guaranteed operating conditions.*

22.4 Functional Operating Range

Rating	Symbol	Value	Unit
Operating temperature range	T_A	–20 to +65	°C
Operating voltage range(1)	V_{DD}	2.0 ± 10% 3.0 ± 10%	V

Note:
1. LCD charge pump optimized for given ranges

22.5 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance, 80 pin LQFP	θ_{JA}	57	°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

Note:
1. Power dissipation is a function of temperature
2. K is a constant unique to the device. K can be determined from a known T_A and measured P_D. With this value of K, P_D and T_J can be determined for any value of T_A.

22.6 3.0-Volt DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Output high voltage (I_{Load} = –0.4 mA) all ports	V_{OH}	0.7 x V_{DD}	—	—	V
Output low voltage (I_{Load} = 0.8 mA) all ports	V_{OL}	—	—	0.3 x V_{DD}	V
Input high voltage All ports, IRQs, $\overline{\text{RESET}}$, OSC1	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input low voltage All ports, IRQs, $\overline{\text{RESET}}$, OSC1	V_{IL}	V_{SS}	—	0.3 x V_{DD}	V
V_{DD} supply current Run [3] Wait [4] Stop[5]	I_{DD}	— — —	— — —	4 2 10	mA mA µA
I/O ports Hi-Z leakage current	I_{IL}	—	—	± 10	µA
Input current	I_{IN}	—	—	1	µA
Capacitance Ports (as input or output)	C_{OUT} C_{IN}	— —	— —	12 8	pF
POR re-arm voltage[6]	V_{POR}	0	—	200	mV
POR reset voltage[7]	V_{PORRST}	0	700	800	mV
POR rise time ramp rate[8]	R_{POR}	0.02	—	—	V/ms

Notes:

1. V_{DD} = 3.0 Vdc ± 10%, 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. Run (operating) I_{DD} measured using external square wave clock source (f_{op} = 4.0 MHz). Measurements represent total current drain on $EV_{DD}/V_{DD}/V_{DDA}$ power supply pins. $EV_{DD} = V_{DD} = V_{DDA}$. All inputs at 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects Run I_{DD}. Measured with CGM (PLL off) disabled.
4. Wait I_{DD} measured using external square wave clock source (f_{op} = 4.0 MHz). Measurements represent total current drain on $EV_{DD}/V_{DD}/V_{DDA}$ power supply pins. $EV_{DD} = V_{DD} = V_{DDA}$. All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects Wait I_{DD}. Measured with CGM (PLL off) module disabled.
5. Stop I_{DD} measured with OSC1 = V_{SS}.
6. Maximum is highest voltage that POR is guaranteed.
7. Maximum is highest voltage that POR is possible.
8. If minimum V_{DD} is not reached before the internal POR reset is released, $\overline{\text{RST}}$ must be driven low externally until minimum V_{DD} is reached.

22.7 2.0-Volt DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Output high voltage (I_{Load} = -0.4 mA) all ports	V_{OH}	0.7 x V_{DD}	—	—	V
Output low voltage (I_{Load} = 0.8 mA) all ports	V_{OL}	—	—	0.3 x V_{DD}	V
Input high voltage All ports, IRQs, $\overline{\text{RESET}}$, OSC1	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input low voltage All ports, IRQs, $\overline{\text{RESET}}$, OSC1	V_{IL}	V_{SS}	—	0.3 x V_{DD}	V
V_{DD} supply current Run [3] Wait [4] Stop[5]	I_{DD}	— — —	— — —	2 1 8	mA mA µA
I/O ports Hi-Z leakage current	I_{IL}	—	—	± 10	µA
Input current	I_{IN}	—	—	1	µA
Capacitance Ports (as input or output)	C_{OUT} C_{IN}	— —	— —	12 8	pF
POR re-arm voltage[6]	V_{POR}	0	—	200	mV
POR reset voltage[7]	V_{PORRST}	0	700	800	mV
POR rise time ramp rate[8]	R_{POR}	0.02	—	—	V/ms

Notes:

1. V_{DD} = 2.0 Vdc ± 10%, 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. Run (operating) I_{DD} measured using external square wave clock source (f_{op} = 2.0 MHz). Measurements represent total current drain on $EV_{DD}/V_{DD}/V_{DDA}$ power supply pins. $EV_{DD} = V_{DD} = V_{DDA}$. All inputs at 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects Run I_{DD}. Measured with CGM (PLL off) disabled.
4. Wait I_{DD} measured using external square wave clock source (f_{op} = 2.0 MHz). Measurements represent total current drain on $EV_{DD}/V_{DD}/V_{DDA}$ power supply pins. $EV_{DD} = V_{DD} = V_{DDA}$. All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects Wait I_{DD}. Measured with CGM (PLL off) module disabled.
5. Stop I_{DD} measured with OSC1 = V_{SS}.
6. Maximum is highest voltage that POR is guaranteed.
7. Maximum is highest voltage that POR is possible.
8. If minimum V_{DD} is not reached before the internal POR reset is released, $\overline{\text{RST}}$ must be driven low externally until minimum V_{DD} is reached.

22.8 RAM Retention

Characteristic	Symbol	Min	Typ	Max	Unit
RAM data retention voltage	V_{RDR}	0.7	—	—	V

22.9 3.0-Volt Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Frequency of operation crystal option[2]	f_{OSC}	32	100	kHz
Internal operating frequency	f_{OP}	—	4.0	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_{SS} = 0 Vdc; timing shown with respect to 20% V_{DD} and 70% V_{DD} unless noted
2. See **22.14 PLL2P12M Component Specifications** 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

22.10 2.0-Volt Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Frequency of operation crystal option[2]	f_{OSC}	32	100	kHz
Internal operating frequency	f_{OP}	—	2.0	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_{SS} = 0 Vdc; timing shown with respect to 20% V_{DD} and 70% V_{DD} unless noted
2. See **22.14 PLL2P12M Component Specifications** 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.

22.11 3.0-Volt SPI Characteristics

Diagram Number[1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	$f_{OP(M)}$ $f_{OP(S)}$	$f_{OP}/128$ DC	$f_{OP}/2$ f_{OP}	MHz
1	Cycle time Master Slave	$t_{CYC(M)}$ $t_{CYC(S)}$	2 1	128 —	t_{CYC}
2	Enable lead time	$t_{Lead(S)}$	30	—	ns
3	Enable lag time	$t_{Lag(S)}$	30	—	ns
4	Clock (SCK) high time Master Slave	$t_{SCKH(M)}$ $t_{SCKH(S)}$	200 100	— —	ns
5	Clock (SCK) low time Master Slave	$t_{SCKL(M)}$ $t_{SCKL(S)}$	200 100	— —	ns
6	Data setup time (inputs) Master Slave	$t_{SU(M)}$ $t_{SU(S)}$	90 10	— —	ns
7	Data hold time (inputs) Master Slave	$t_{H(M)}$ $t_{H(S)}$	0 30	— —	ns
8	Access time, slave[3] CPHA = 0 CHPA = 1	$t_{A(CP0)}$ $t_{A(CP1)}$	0 0	80 40	ns
9	Disable time, slave[4]	$t_{DIS(S)}$	—	50	ns
10	Data valid time (after enable edge) Master Slave[5]	$t_{V(M)}$ $t_{V(S)}$	— —	20 80	ns
11	Data hold time (outputs, after enable edge) Master Slave	$t_{HO(M)}$ $t_{HO(S)}$	0 10	— —	ns

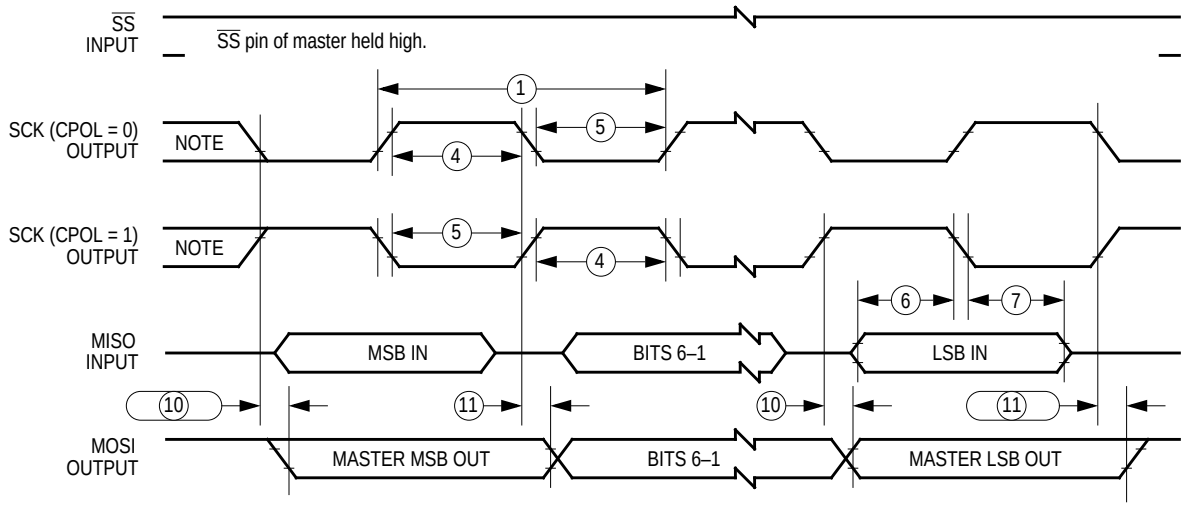
Notes:
1. Numbers refer to dimensions in **Figure 22-1** and **Figure 22-2.**
2. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; assumes 100 pF load on all SPI pins.
3. Time to data active from high-impedance state
4. Hold time to high-impedance state
5. With 100 pF on all SPI pins

22.12 2.0-Volt SPI Characteristics

Diagram Number[1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	$f_{OP(M)}$ $f_{OP(S)}$	$f_{OP}/128$ DC	$f_{OP}/2$ f_{OP}	MHz
1	Cycle time Master Slave	$t_{CYC(M)}$ $t_{CYC(S)}$	2 1	128 —	t_{CYC}
2	Enable lead time	$t_{Lead(S)}$	60	—	ns
3	Enable lag time	$t_{Lag(S)}$	60	—	ns
4	Clock (SCK) high time Master Slave	$t_{SCKH(M)}$ $t_{SCKH(S)}$	400 200	— —	ns
5	Clock (SCK) low time Master Slave	$t_{SCKL(M)}$ $t_{SCKL(S)}$	400 200	— —	ns
6	Data setup time (inputs) Master Slave	$t_{SU(M)}$ $t_{SU(S)}$	180 20	— —	ns
7	Data hold time (inputs) Master Slave	$t_{H(M)}$ $t_{H(S)}$	0 60	— —	ns
8	Access time, slave[3] CPHA = 0 CHPA = 1	$t_{A(CP0)}$ $t_{A(CP1)}$	0 0	160 80	ns
9	Disable time, slave[4]	$t_{DIS(S)}$	—	100	ns
10	Data valid time (after enable edge) Master Slave[5]	$t_{V(M)}$ $t_{V(S)}$	— —	40 160	ns
11	Data hold time (outputs, after enable edge) Master Slave	$t_{HO(M)}$ $t_{HO(S)}$	0 2	— —	ns

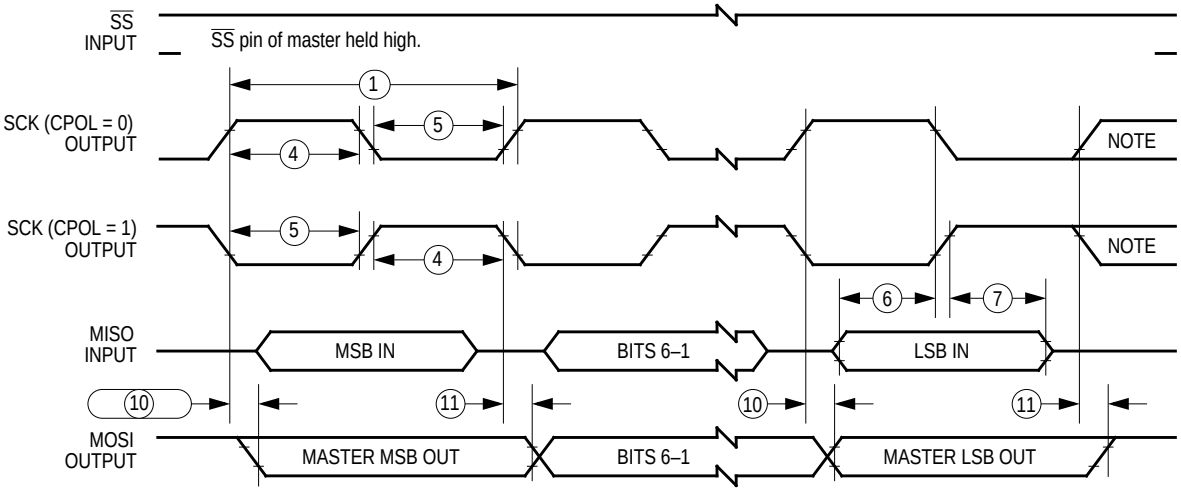
Notes:

1. Numbers refer to dimensions in **Figure 22-1** and **Figure 22-2.**
2. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; assumes 100 pF load on all SPI pins
3. Time to data active from high-impedance state.
4. Hold time to high-impedance state.
5. With 100 pF on all SPI pins.



NOTE: This first clock edge is generated internally, but is not seen at the SCK pin.

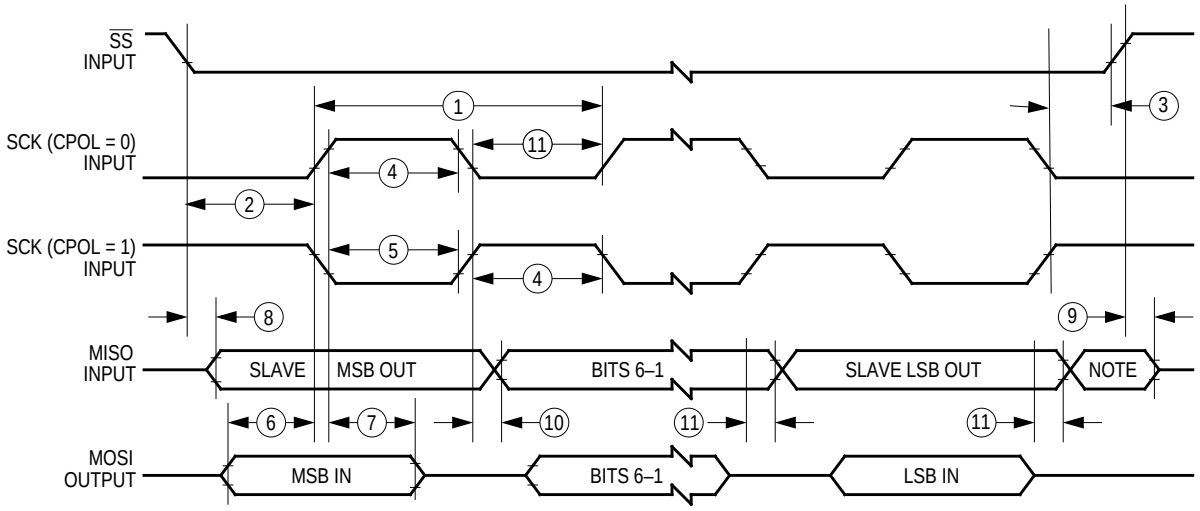
a) SPI Master Timing (CPHA = 0)



NOTE: This last clock edge is generated internally, but is not seen at the SCK pin.

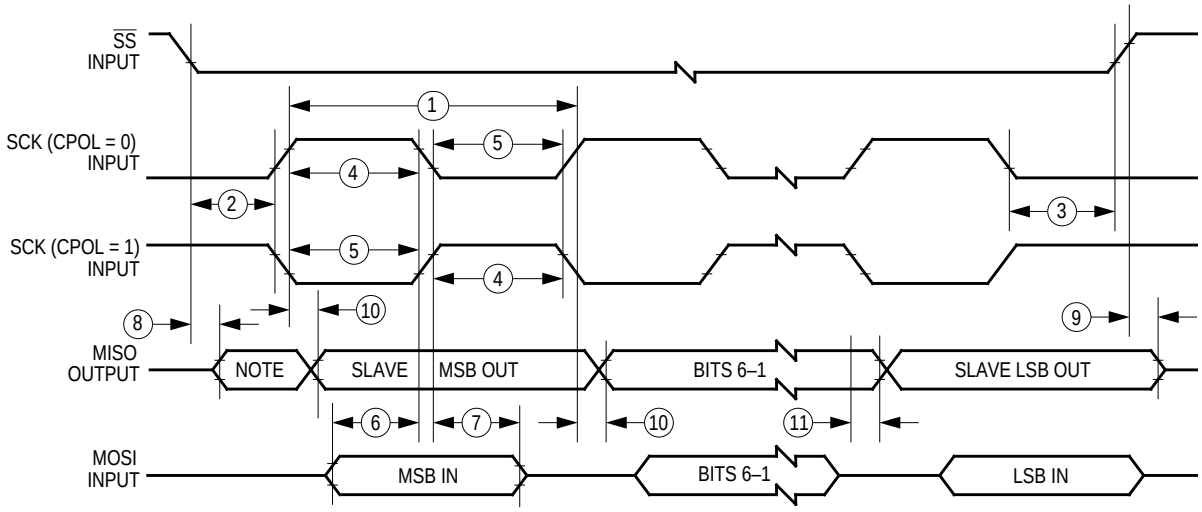
b) SPI Master Timing (CPHA = 1)

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

22.13 PLL2P12M Electrical Specifications

Description	Symbol	Min	Typ	Max	Notes
CGMXCLK reference frequency (Hz)	f_{RCLK}	—	38.4 k	—	
Range nominal multiplier (Hz)	f_{NOM}	—	38.4 k	—	
VCO center-of-range frequency (Hz)	f_{VRS}	38.4 k 38.4 k	—	20.0 M 10.0 M	2.7–3.3 V, V_{DD} only 1.8–2.7 V, V_{DD} only
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}	—	—	4 M 2 M	2.7–3.3 V, V_{DD} only 1.8–2.7 V, V_{DD} only

22.14 PLL2P12M Component Specifications

Characteristic	Symbol	Min	Typ	Max	Notes
Crystal load capacitance	C_L	—	—	—	Consult crystal manufacturer's data
Crystal fixed capacitance	C_1	—	$2*C_L$	—	Consult crystal manufacturer's data
Crystal tuning capacitance	C_2	—	$2*C_L$	—	Consult crystal manufacturer's data
Feedback bias resistor	R_B	—	22 MΩ	—	
Series resistor	R_S	0	330 kΩ	1MΩ	Not required
Filter capacitor	C_F	—	$C_{FACT}*$ (V_{DDA}/f_{XCLK})	—	
Bypass capacitor	C_{BYP}	—	0.1 µF	—	C_{BYP} must provide low AC impedance from f = $f_{XCLK}/100$ to $100*f_{VCLK}$, so series resistance must be considered.

22.15 Bus Clock PLL Acquisition/Lock Time Specifications

Specifications for the entry and exit of acquisition and tracking modes, as well as required manual mode delay times are provided here.

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}	—	10 ms	—	C_F = 0.1 µF(1)
Manual stable to lock time	t_{AL}	—	20 ms	—	C_F = 0.1 µF(1)
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 frequency tolerance	Δ_{Lock}	0	—	± 0.9%	
LOCK exit frequency 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}	—	10 ms	—	C_F = 0.1 µF(1)
Automatic stable to lock time	t_{AL}	—	15 ms	—	C_F = 0.1 µF(1)
Automatic lock time	t_{Lock}	—	$t_{ACQ} + t_{AL}$	—	

Note:

1. Refer to **22.14 PLL2P12M Component Specifications** for information on calculating C_F.

22.16 2-K FLASH Memory Electrical Characteristics

Parameter	Description	Min	Recommended	Max	Units
t_{Erase}	Erase time	100	—	110	ms
t_{kill}	High voltage kill time	200	200	—	µs
t_{HVD}	Return to read mode time	50	50	—	µs
t_{Step}(1)	Program step size	0.8	1.0	1.2	ms
Pulses	Number of program pulses/page	1	—	5	ms
Endurance(2)	Erase/program cycles	10,000	—	—	Cycles
f_{CP2}	Charge pump clock frequency for 2-K FLASH	1.0	2.0	2.5	MHz
t_{HVTV}	HVEN low to VERF high time	50	—	—	µs
t_{VTP}	VERF high to PGM low time	150	—	—	µs

Notes:
1. t_{Step} is defined as the amount of time during a program cycle in which the HVEN bit is set.
2. Minimum endurance means the part is guaranteed to work up to this many erase/program cycles.

22.17 48-K FLASH Memory Electrical Characteristics

Parameter	Description	Min	Recommended	Max	Units
t_{Erase}	Erase time	100	—	110	ms
t_{kill}	High voltage kill time	200	200	—	µs
t_{HVD}	Return to read mode time	50	50	—	µs
t_{Step}(1)	Program step size	0.8	1.0	1.2	ms
Pulses	Number of program pulses/page	1	—	5	ms
Endurance(2)	Erase/program cycles	100	—	—	Cycles
f_{CP2}	Charge pump clock frequency for 48-K FLASH	1.8	2.0	2.5	MHz
t_{HVTV}	HVEN low to VERF high time	50	—	—	µs
t_{VTP}	VERF high to PGM low time	150	—	—	µs

Notes:
1. t_{Step} is defined as the amount of time during a program cycle in which the HVEN bit is set.
2. Minimum endurance means the part is guaranteed to work up to this many erase/program cycles.

22.18 ADC Characteristics

Characteristic	Symbol	Min	Typ	Max	Notes
Supply voltage	V_{DDA2}	1.8	3.3	V	V_{DDA2} should be tied to the same potential as V_{DD} via separate traces.
Input voltages	V_{ADIN} V_{REFH}	0 1.5	V_{REFH} V_{DDA2}	V	
Resolution	B_{AD}	7	7	Bits	
ADC internal clock	f_{ADIC}	500 k	1.048 M	Hz	$t_{AIC} = 1/f_{ADIC}$
Conversion range	R_{AD}	V_{SSA2}	V_{REFH}	V	
Power-up time	t_{ADPU}	16	—	t_{AIC} cycles	
Conversion time	t_{ADC}	16	17	t_{AIC} cycles	
Sample time	t_{ADS}	5	—	t_{AIC} cycles	
Monotocity	M_{AD}	Guaranteed			
Zero input reading	Z_{ADI}	00	—	Hex	$V_{In} = V_{SSA2}$
Full-scale reading	F_{ADI}		FF	Hex	$V_{In} = V_{REFH}$
Input capacitance	C_{ADI}	—	20	pF	Not tested

Section 23. Mechanical Data

23.1 Contents

23.2 Introduction

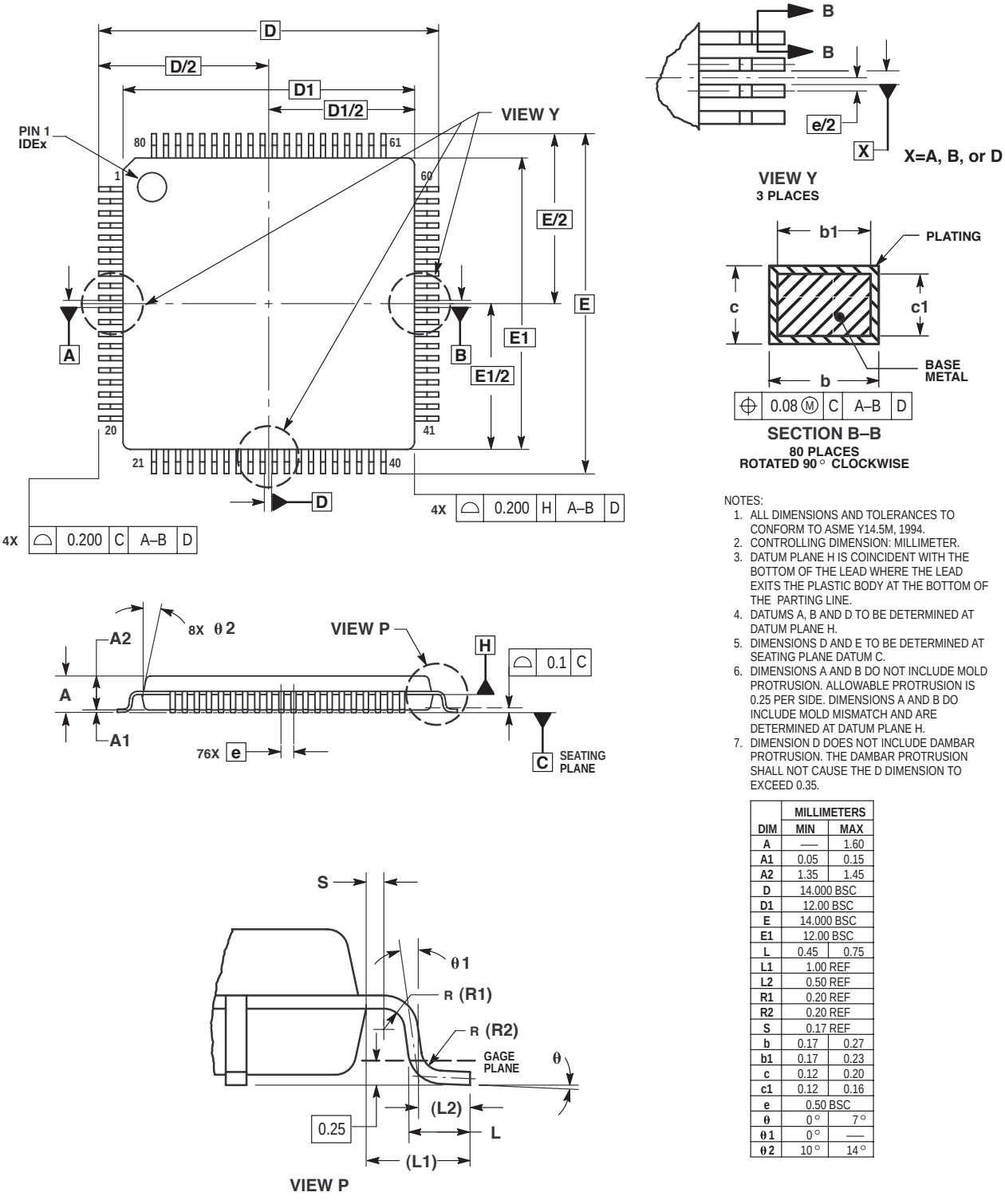
This section describes the dimensions of the 80-pin quad flat pack (LQFP).

23.3 80-Pin LQFP (Case 917-01) shows the latest package 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 Fax Back System (Mfax™)
  - Phone 1-602-244-6609
  - EMAIL RMFAX0@email.sps.mot.com; http://sps.motorola.com/mfax/
- Worldwide Web (wwweb) home page at http://www.mot-sps.com/cgi-bin-cases

Follow Mfax or wwweb on-line instructions to retrieve the current mechanical specifications.

23.3 80-Pin LQFP (Case 917-01)



NOTES:
1. ALL DIMENSIONS AND TOLERANCES TO CONFORM TO ASME Y14.5M, 1994.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE H IS COINCIDENT WITH THE BOTTOM OF THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS A, B AND D TO BE DETERMINED AT DATUM PLANE H.
5. DIMENSIONS D AND E TO BE DETERMINED AT SEATING PLANE DATUM C.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 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. THE DAMBAR PROTRUSION SHALL NOT CAUSE THE D DIMENSION TO EXCEED 0.35.

	MILLIMETERS	
DIM	MIN	MAX
A	—	1.60
A1	0.05	0.15
A2	1.35	1.45
D	14.000 BSC	
D1	12.00 BSC	
E	14.000 BSC	
E1	12.00 BSC	
L	0.45	0.75
L1	1.00 REF	
L2	0.50 REF	
R1	0.20 REF	
R2	0.20 REF	
S	0.17 REF	
b	0.17	0.27
b1	0.17	0.23
c	0.12	0.20
c1	0.12	0.16
e	0.50 BSC	
θ	0°	7°
θ1	0°	—
θ2	10°	14°

Section 24. Ordering Information

24.1 Contents

24.2 MCU Ordering Forms

To initiate an order for a ROM-based MCU, first obtain the current ordering form for the MCU from a Motorola representative. Submit the following items when ordering MCUs:

- A current MCU ordering form that is **completely filled out** (Contact your Motorola sales office for assistance.)
- A copy of the customer specification if it deviates from the Motorola specification for the MCU
- Customer's application program on one of the media listed in **24.3 Application Program Media**

The current MCU ordering form is also available through the Motorola Freeware Bulletin Board Service (BBS). The telephone number is (512) 891-FREE. After making the connection, type bbs in lowercase letters. Then press the return key to start the BBS software.

24.3 Application Program Media

Deliver the application program to Motorola in one of these media:

- Macintosh®[1] 3-1/2-inch diskette (double-sided 800 K or double-sided high-density 1.4 M)
- MS-DOS®[2] or PC-DOS™[3] 3 1/2-inch diskette (double-sided 720 K or double-sided high-density 1.44 M)
- MS-DOS® or PC-DOS 5 1/4-inch diskette (double-sided double-density 360 K or double-sided high-density 1.2 M)

Use positive logic for data and addresses.

When submitting the application program on a diskette, clearly label the diskette with the following information:

- Customer name
- Customer part number
- Project or product name
- File name of object code
- Date
- Name of operating system that formatted diskette
- Formatted capacity of diskette

On diskettes, the application program must be in Motorola's S-record format (S1 and S9 records), a character-based object file format generated by M6805 cross assemblers and linkers.

Begin the application program at the first user ROM location. Program addresses must correspond exactly to the available on-chip user ROM addresses as shown in the memory map. **Write $00 in all non-user ROM locations or leave all non-user ROM locations blank**. Refer to the current MCU ordering form for additional requirements. Motorola may request pattern resubmission if non-user areas contain any non-zero code.

1. Macintosh is a registered trademark of Apple Computer, Inc.
2. MS-DOS is a registered trademark of Microsoft Corporation.
3. PC-DOS is a trademark of International Business Machines Corporation.

If the memory map has two user ROM areas with the same addresses, then write the two areas in separate files on the diskette. Label the diskette with both filenames.

In addition to the object code, a file containing the source code can be included. Motorola keeps this code confidential and uses it only to expedite ROM pattern generation in case of any difficulty with the object code. Label the diskette with the filename of the source code.

24.4 ROM Program Verification

The primary use for the on-chip ROM is to hold the customer's application program. The customer develops and debugs the application program and then submits the MCU order along with the application program.

Motorola enters the customer's application program code into a computer program that generates a listing verify file. The listing verify file represents the memory map of the MCU. The listing verify file contains the user ROM code and may also contain non-user ROM code, such as selfcheck code. Motorola sends the customer a computer printout of the listing verify file along with a listing verify form.

To aid the customer in checking the listing verify file, Motorola programs the listing verify file into customer-supplied, blank, preformatted Macintosh or DOS disks. All original pattern media are filed for contractual purposes and are not returned.

Check the listing verify file thoroughly, then complete and sign the listing verify form and return the listing verify form to Motorola. The signed listing verify form constitutes the contractual agreement for the creation of the custom mask.

24.5 ROM Verification Units (RVUs)

After receiving the signed listing verify form, Motorola manufactures a custom photographic mask. The mask contains the customer's application program and is used to process silicon wafers. The application program cannot be changed after the manufacture of the mask begins. Motorola then produces 10 MCUs, called RVUs, and sends the RVUs to the customer. RVUs are usually packaged in unmarked ceramic and tested to 5 Vdc at room temperature. RVUs are not tested to environmental extremes because their sole purpose is to demonstrate that the customer's user ROM pattern was properly implemented. The 10 RVUs are free of charge with the minimum order quantity. These units are not to be used for qualification or production. RVUs are not guaranteed by Motorola Quality Assurance.

24.6 MC Order Numbers

Table 24-1. MC Order Numbers

MC Order Number	Operating Temperature Range
MC68HC(9)08PT48CFU(1)	–40 °C to + 85 °C

1. FU = plastic quad flat pack package

Home Page:
www.freescale.com
email:
support@freescale.com
USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
(800) 521-6274
480-768-2130
support@freescale.com
Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com
Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 2666 8080
support.japan@freescale.com
Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com
For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
(800) 441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

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

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