HC708XL36TS/D
Rev. 1

HC08

MC68HC708XL36

TECHNICAL SUMMARY

MOTOROLA

MC68HC708XL36

HCMOS MICROCONTROLLER UNIT

PREFACE

All M68HC08 microcontroller units (MCUs) are modular, customer-specified designs. To meet customer requirements, Motorola is constantly designing new modules and creating new versions of existing modules. The following list shows the version levels of the modules in the MC68HC708XL36 MCU:

Module	Version
CPU8 (Central Processor Unit)[1]	A
SIM16 (System Integration Module)	C
CGM (Clock Generator Module)	A
DMA (Direct Memory Access Module)	A
Break	B
TIM4 (Timer Interface Module)[2]	B
SPI (Serial Peripheral Interface Module)	B
SCI (Serial Communications Interface Module)	C
COP (Computer Operating Properly Module)	B
IRQEPM (External Interrupt Module)	C
LVI27 (Low-Voltage Inhibit Module)	A
POR (Power-on Reset Module)	B
MOREPM (Mask Option Register)	A
EPROM	
RAM	
Firmware	**Version**
MON08 (Monitor ROM)	B

1. See *CPU08 REFERENCE MANUAL*, Motorola Document Number CPU08RM/AD.
2. See *TIM08 REFERENCE MANUAL*, Motorola Document Number TIM08RM/AD.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

SECTION 15
I/O PORTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES

SECTION 1
GENERAL DESCRIPTION

1.1 Introduction

The MC68HC708XL36 is the first member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). The M68HC08 Family is based on the customer-specified integrated circuit (CSIC) design strategy. All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

1.2 Features

Features of the MC68HC708XL36 include the following:

- High-Performance M68HC08 Architecture
- Fully Upward-Compatible Object Code with M6805, M146805, and M68HC05 Families
- 8-MHz Internal Bus Frequency
- 36 Kbytes of On-Chip Erasable Programmable Read-Only Memory (EPROM) or One-Time Programmable Read-Only Memory (OTPROM)
- On-Chip Programming Firmware for Use with Host Personal Computer
- EPROM/OTPROM Data Security
- 1 Kbyte of On-Chip RAM
- Serial Peripheral Interface Module (SPI)
- Serial Communications Interface Module (SCI)
- 16-Bit, 4-Channel Timer Interface Module (TIM)
- Three-Channel Direct Memory Access Module (DMA)
- Clock Generator Module (CGM)

- System Protection Features
  - Optional Computer Operating Properly (COP) Reset
  - Low-Voltage Detection with Optional Reset
  - Illegal Opcode Detection with Optional Reset
  - Illegal Address Detection with Optional Reset
- 56-Pin Plastic Shrink Dual-In-Line Package (SDIP) or 64-Pin Plastic Quad Flat Pack (QFP)
- Low-Power Design (Fully Static with Stop and Wait Modes)
- Master Reset Pin and Power-On Reset
- 8-Bit Key Wake-Up Port

Features of the CPU08 include the following:

- Enhanced HC05 Programming Model
- Extensive Loop Control Functions
- 16 Addressing Modes (Eight More Than the HC05)
- 16-Bit Index Register and Stack Pointer
- Memory-to-Memory Data Transfers
- Fast 8 × 8 Multiply Instruction
- Fast 16/8 Divide Instruction
- Binary-Coded Decimal (BCD) Instructions
- Optimization for Controller Applications
- C Language Support

1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC708XL36.

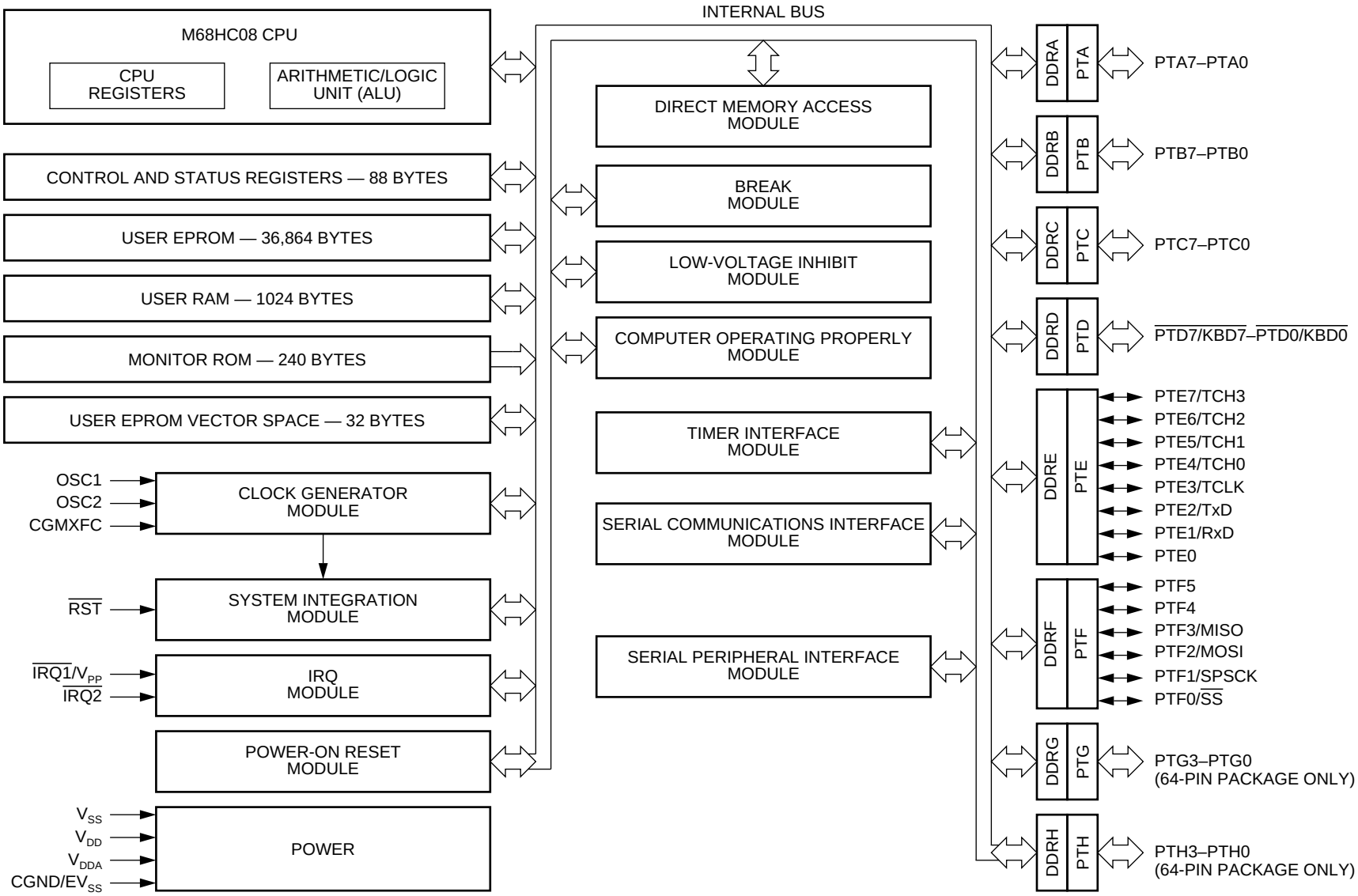


Figure 1-1. MCU Block Diagram

1.4 Pin Assignments

1

Figure 1-2 and Figure 1-3 show the SDIP and QFP pin assignments.

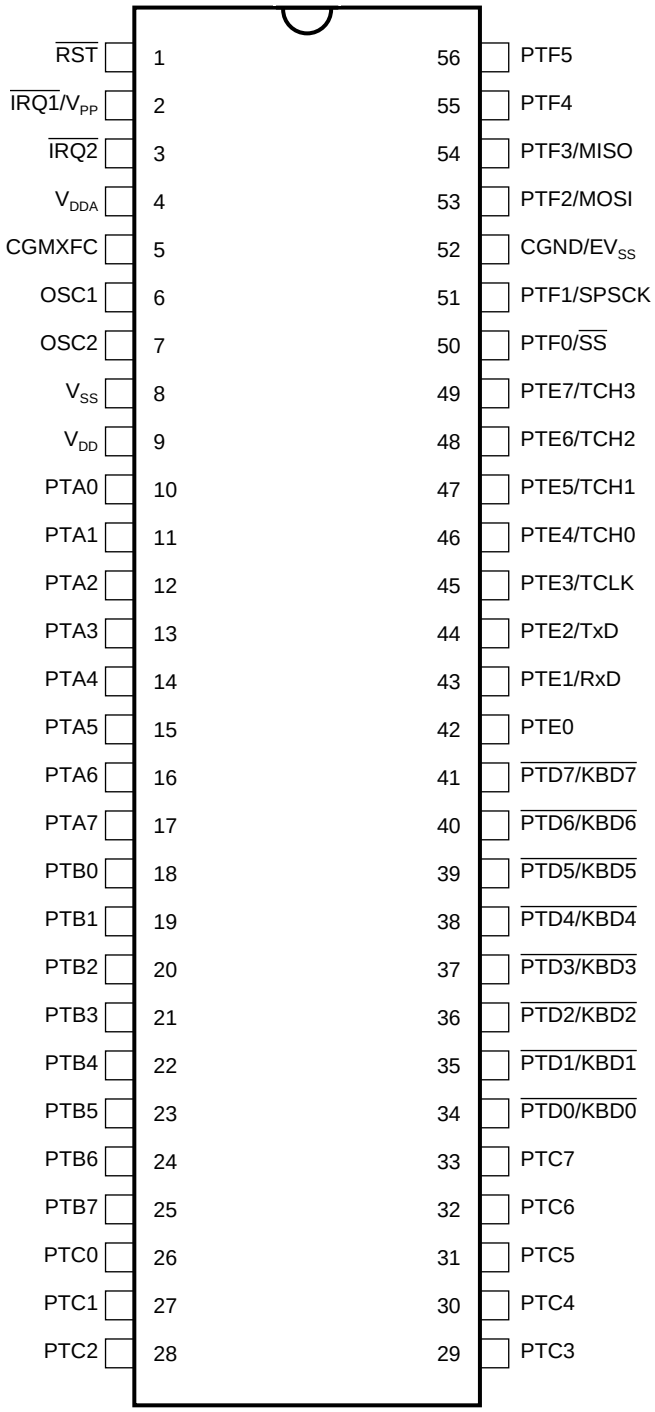
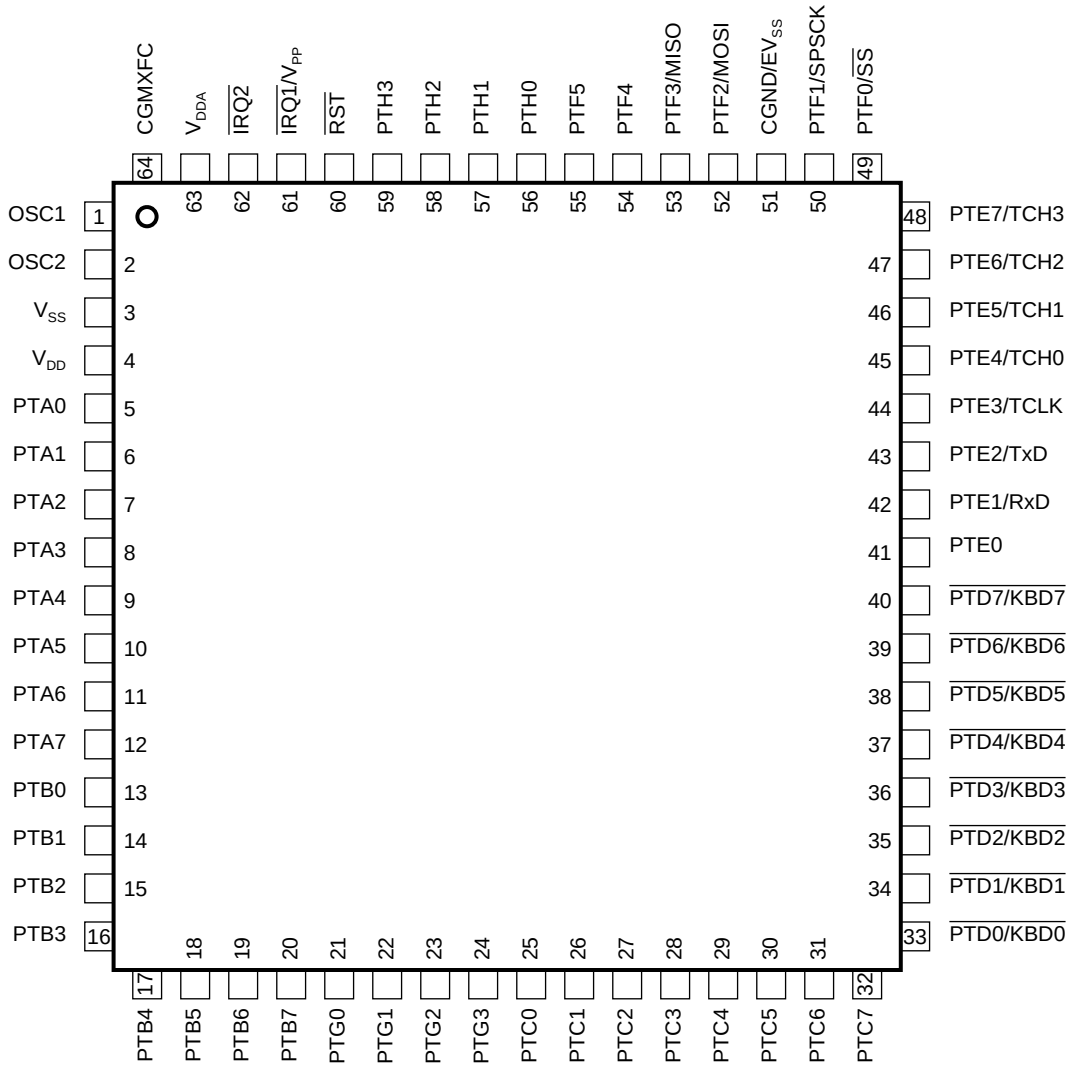


Figure 1-2. SDIP Pin Assignments



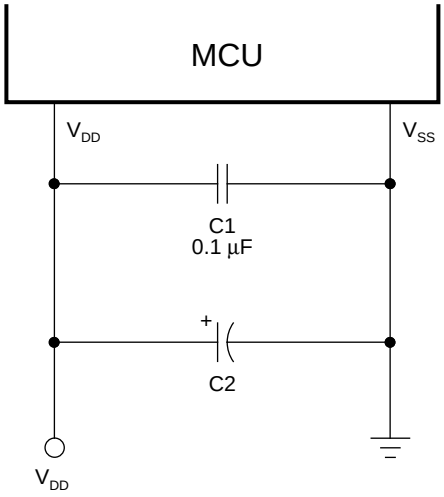
NOTE: Ports G and H are available only with the QFP.

Figure 1-3. QFP Pin Assignments

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

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

Fast signal transitions on MCU pins place high, short-duration current demands on the power supply. To prevent noise problems, take special care to provide power supply bypassing at the MCU as Figure 1-4 shows. Place the C1 bypass capacitor as close to the MCU as possible. Use a high-frequency-response ceramic capacitor for C1. C2 is an optional bulk current bypass capacitor for use in applications that require the port pins to source high current levels.



NOTE: Component values shown represent typical applications.

Figure 1-4. Power Supply Bypassing

1.4.2 Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. (See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**.)

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

A logic zero on the $\overline{\text{RST}}$ pin forces the MCU to a known start-up 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.)

1.4.4 External Interrupt Pins ($\overline{\text{IRQ1}}/V_{PP}$ and $\overline{\text{IRQ2}}$)

$\overline{\text{IRQ1}}/V_{PP}$ and $\overline{\text{IRQ2}}$ are asynchronous external interrupt pins. (See **SECTION 17 EXTERNAL INTERRUPT MODULE (IRQ)**.) $\overline{\text{IRQ1}}/V_{PP}$ is also the EPROM/OTPROM programming power pin. (See **SECTION 2 MEMORY MAP**.)

1.4.5 Clock Ground Pin ($CGND/EV_{SS}$)

$CGND/EV_{SS}$ is the ground for the port output buffers and the ground return for the serial clock in the serial peripheral interface module (SPI). (See **SECTION 13 SERIAL PERIPHERAL INTERFACE MODULE (SPI)**.)

NOTE

CGND/EV_{SS} must be grounded for proper MCU operation.

1.4.6 CGM Power Supply Pin (V_{DDA})

V_{DDA} is the power supply pin for the analog portion of the clock generator module (CGM). (See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**.)

1.4.7 External Filter Capacitor Pin (CGMXFC)

CGMXFC is an external filter capacitor connection for the CGM. (See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**.)

1.4.8 Port A Input/Output (I/O) Pins (PTA7–PTA0)

PTA7–PTA0 are general-purpose bidirectional I/O port pins. (See **SECTION 15 I/O PORTS**.)

1.4.9 Port B I/O Pins (PTB7–PTB0)

PTB7–PTB0 are general-purpose bidirectional I/O port pins. (See **SECTION 15 I/O PORTS**.)

1.4.10 Port C I/O Pins (PTC7–PTC0)

PTC7–PTC0 are general-purpose bidirectional I/O port pins. (See **SECTION 15 I/O PORTS**.)

1.4.11 Port D I/O Pins ($\overline{\text{PTD7/KBD7}}$–$\overline{\text{PTD0/KBD0}}$)

$\overline{\text{PTD7/KBD7}}$–$\overline{\text{PTD0/KBD0}}$ are general-purpose bidirectional I/O port pins. Any or all of the port D pins can be programmed to serve as external interrupt pins. (See **SECTION 15 I/O PORTS**.)

1.4.12 Port E I/O Pins (PTE7/TCH3–PTE0)

Port E is an 8-bit special function port that shares five of its pins with the timer interface module (TIM) and two of its pins with the serial communications interface (SCI) module. (See **SECTION 12 TIMER INTERFACE MODULE (TIM)**, **SECTION 14 SERIAL COMMUNICATIONS INTERFACE MODULE**, and **SECTION 15 I/O PORTS**.)

1.4.13 Port F I/O Pins (PTF5–PTF0/$\overline{SS}$)

Port F is a 6-bit special function port that shares four of its pins with the serial peripheral interface module (SPI). (See **SECTION 13 SERIAL PERIPHERAL INTERFACE MODULE (SPI)** and **SECTION 15 I/O PORTS**.)

1.4.14 Port G I/O Pins (PTG3–PTG0)

PTG3–PTG0 are general-purpose bidirectional I/O pins. (See **SECTION 15 I/O PORTS**.) Port G is available only with the 64-pin package.

1.4.15 Port H I/O Pins (PTH3–PTH0)

PTH3–PTH0 are general-purpose bidirectional I/O pins. (See **SECTION 15 I/O PORTS**.) Port H is available only with the 64-pin package.

SECTION 2
MEMORY MAP

2.1 Introduction

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

- 36 Kbytes of EPROM or OTPROM
- 1 Kbyte of RAM
- 32 bytes of user-defined vectors
- 240 bytes of monitor ROM

2.2 I/O Section

Addresses $0000–$004F, shown in Figure 2-2, contain most of the control, status, and data registers. Additional I/O registers have the following addresses:

- $FE00 (SIM break status register, SBSR)
- $FE01 (SIM reset status register, SRSR)
- $FE03 (SIM break flag control register, SBFCR)
- $FE07 (EPROM control register, EPMCR)
- $FE0C and $FE0D (break address registers, BRKH and BRKL)
- $FE0E (break status and control register, BRKSCR)
- $FE0F (LVI status register, LVISR)
- $FFFF (COP control register, COPCTL)

Table 2-1 is a list of vector locations.

$0000 ↓ $004F	I/O REGISTERS (80 BYTES)
$0050 ↓ $044F	RAM (1024 BYTES)
$0450 ↓ $6DFF	UNIMPLEMENTED (27,056 BYTES)
$6E00 ↓ $FDFF	EPROM (36,864 BYTES)
$FE00	SIM BREAK STATUS REGISTER (SBSR)
$FE01	SIM RESET STATUS REGISTER (SRSR)
$FE02	RESERVED
$FE03	SIM BREAK FLAG CONTROL REGISTER (SBFCR)
$FE04	RESERVED
$FE05	RESERVED
$FE06	UNIMPLEMENTED
$FE07	EPROM CONTROL REGISTER (EPMCR)
$FE08 ↓ $FE0B	UNIMPLEMENTED (4 BYTES)
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FEFF	MONITOR ROM (240 BYTES)
$FF00 ↓ $FFBF	UNIMPLEMENTED (192 BYTES)
$FFC0 ↓ $FFDF	RESERVED (32 BYTES)
$FFE0 ↓ $FFFF	VECTORS (32 BYTES)

Figure 2-1. Memory Map

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	R:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		W:								
$0001	Port B Data Register (PTB)	R:	PTB7	PTB6	PTB25	PTB4	PTB3	PTB2	PTB1	PTB0
		W:								
$0002	Port C Data Register	R:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		W:								
$0003	Port D Data Register (PTD)	R:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		W:								
$0004	Data Direction Register A (DDRA)	R:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		W:								
$0005	Data Direction Register B (DDRB)	R:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		W:								
$0006	Data Direction Register C (DDRC)	R:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		W:								
$0007	Data Direction Register D (DDRD)	R:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		W:								
$0008	Port E Data Register (PTE)	R:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		W:								
$0009	Port F Data Register (PTF)	R:	0	0	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
		W:								
$000A	Port G Data Register (PTG)	R:	0	0	0	0	PTG3	PTG2	PTG1	PTG0
		W:								
$000B	Port H Data Register (PTH)	R:	0	0	0	0	PTH3	PTH2	PTH1	PTH0
		W:								
$000C	Data Direction Register E (DDRE)	R:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		W:								
$000D	Data Direction Register F (DDRF)	R:	0	0	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
		W:								
$000E	Data Direction Register G (DDRG)	R:	0	0	0	0	DDRG3	DDRG2	DDRG1	DDRG0
		W:								
$000F	Data Direction Register H (DDRH)	R:	0	0	0	0	DDRH3	DDRH2	DDRH1	DDRH0
		W:								
$0010	SPI Control Register (SPCR)	R:	SPRIE	DMAS	SP-MSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		W:								
$0011	SPI Status and Control Register (SPSCR)	R:	SPRF	0	OVRF	MODF	SPTE	0	SPR1	SPR0
		W:								
$0012	SPI Data Register (SPDR)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0013	SCI Control Register 1 (SCC1)	R:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		W:								
$0014	SCI Control Register 2 (SCC2)	R:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		W:								
$0015	SCI Control Register 3 (SCC3)	R:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
		W:								
$0016	SCI Status Register 1 (SCS1)	R:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		W:								

= Unimplemented R = Reserved

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

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0017	SCI Status Register 2 (SCS2)	R:	0	0	0	0	0	0	BKF	RPF
		W:								
$0018	SCI Data Register (SCDR)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0019	SCI Baud Rate Register (SCBR)	R:	0	0	SCP1	SCP0	0	SCR2	SCR1	SCR0
		W:								
$001A	IRQ Status and Control Register (ISCR)	R:	PIN2	0	IMASK2	MODE2	IRQ2DIS	0	IMASK1	MODE1
		W:		ACK2				ACK1		
$001B	Keyboard Interrupt Control Register (KBICR)	R:	KB7IE	KB6IE	KB5IE	KB4IE	KB3IE	KB2IE	KB1IE	KB0IE
		W:								
$001C	PLL Control Register (PCTL)	R:	PLLIE	PLLF	PLLON	BCS	1	1	1	1
		W:								
$001D	PLL Bandwidth Control Register (PBWC)	R:	AUTO	LOCK	$\overline{\text{ACQ}}$	XLD	0	0	0	0
		W:								
$001E	PLL Programming Register (PPG)	R:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
		W:								
$001F	Mask Option Register (MOR)	R:	0	LVISTOP	LVIRST	LVIPWR	SSREC	SEC	STOP	COPD
		W:								
$0020	Timer Status and Control Register (TSC)	R:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		W:	0			TRST				
$0021	Timer DMA Select Register (TDMA)	R:	0	0	0	0	DMA3S	DMA2S	DMA1S	DMA0S
		W:								
$0022	Timer Counter Register High (TCNTH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0023	Timer Counter Register Low (TCNTL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0024	Timer Modulo Register High (TMODH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0025	Timer Modulo Register Low (TMODL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0026	Timer Channel 0 Status and Control Register (TSC0)	R:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		W:	0							
$0027	Timer Channel 0 Register High (TCH0H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0028	Timer Channel 0 Register Low (TCH0L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0029	Timer Channel 1 Status and Control Register (TSC1)	R:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		W:	0							
$002A	Timer Channel 1 Register High (TCH1H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$002B	Timer Channel 1 Register Low (TCH1L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$002C	Timer Channel 2 Status and Control Register (TSC2)	R:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		W:	0							
$002D	Timer Channel 2 Register High (TCH2H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								

= Unimplemented R = Reserved

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

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$002E	Timer Channel 2 Register Low (TCH2L)	R: W:	Bit 7	6	5	4	3	2	1	Bit 0
$002F	Timer Channel 3 Status and Control Register (TSC3)	R: W:	CH3F 0	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
$0030	Timer Channel 3 Register High (TCH3H)	R: W:	Bit 15	14	13	12	11	10	9	Bit 8
$0031	Timer Channel 3 Register Low (TCH3L)	R: W:	Bit 7	6	5	4	3	2	1	Bit 0
$0034	DMA Channel 0 Source Address Register High (D0SH)	R: W:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
$0035	DMA Channel 0 Source Address Register Low (D0SL)	R: W:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
$0036	DMA Channel 0 Destination Address Register High (D0DH)	R: W:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
$0037	DMA Channel 0 Destination Address Register Low (D0DL)	R: W:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
$0038	DMA Channel 0 Control Register (D0C)	R: W:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
$0039	DMA Channel 0 Block Length Register (D0BL)	R: W:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
$003B	DMA Channel 0 Byte Count Register (D0BC)	R: W:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
$003C	DMA Channel 1 Source Address Register High (D1SH)	R: W:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
$003D	DMA Channel 1 Source Address Register Low (D1SL)	R: W:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
$003E	DMA Channel 1 Destination Address Register High (D1DH)	R: W:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
$003F	DMA Channel 1 Destination Address Register Low (D1DL)	R: W:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
$0040	DMA Channel 1 Control Register (D1C)	R: W:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
$0041	DMA Channel 1 Block Length Register (D1BL)	R: W:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
$0043	DMA Channel 1 Byte Count Register (D1BC)	R: W:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
$0044	DMA Channel 2 Source Address Register High (D2SH)	R: W:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
$0045	DMA Channel 2 Source Address Register Low (D2SL)	R: W:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
$0046	DMA Channel 2 Destination Address Register High (D2DH)	R: W:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8

= Unimplemented R = Reserved

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

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0047	DMA Channel 2 Destination Address Register Low (D2DL)	R:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		W:								
$0048	DMA Channel 2 Control Register (D2C)	R:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
		W:								
$0049	DMA Channel 2 Block Length Register (D2BL)	R:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
		W:								
$004B	DMA Channel 2 Byte Count Register (D2BC)	R:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
		W:								
$004C	DMA Control Register 1 (DC1)	R:	BB1	BB0	TEC2	IEC2	TEC1	IEC1	TEC0	IEC0
		W:								
$004D	DMA Status and Control Register (DSC)	R:	DMAP	L2	L1	L0	DMAWE	IFC2	IFC1	IFC0
		W:								
$004E	DMA Control Register 2 (DC2)	R:	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
		W:								
$FE00	SIM Break Status Register (SBSR)	R:	R	R	R	R	R	R	SBSW	R
		W:								
$FE01	SIM Reset Status Register (SRSR)	R:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
		W:								
$FE03	SIM Break Flag Control Register (SBFCR)	R:	BCFE	R	R	R	R	R	R	R
		W:								
$FE07	EPROM Control Register (EPMCR)	R:	EPMCR	0	0	0	0	ELAT	0	EPGM
		W:								
$FE0C	Break Address Register High (BRKH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$FE0D	Break Address Register Low (BRKL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$FE0E	Break Status and Control Register (BRKSCR)	R:	BRKE	BRKA	0	0	0	0	0	0
		W:								
$FE0F	LVI Status Register (LVISR)	R:	LVIOUT	0	0	0	0	0	0	0
		W:								
$FFFF	COP Control Register (COPCTL)	R:	LOW BYTE OF RESET VECTOR							
		W:	WRITING TO $FFFF CLEARS COP COUNTER							

= Unimplemented R = Reserved

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

Table 2-1. Vector Addresses

Address	Vector
$FFE0	IRQ2/Keypad Vector (High)
$FFE1	IRQ2/Keypad Vector (Low)
$FFE2	SCI Transmit Vector (High)
$FFE3	SCI Transmit Vector (Low)
$FFE4	SCI Receive Vector (High)
$FFE5	SCI Receive Vector (Low)
$FFE6	SCI Error Vector (High)
$FFE7	SCI Error Vector (Low)
$FFE8	SPI Transmit Vector (High)
$FFE9	SPI Transmit Vector (Low)
$FFEA	SPI Receive Vector (High)
$FFEB	SPI Receive Vector (Low)
$FFEC	TIM Overflow Vector (High)
$FFED	TIM Overflow Vector (Low)
$FFEE	TIM Channel 3 Vector (High)
$FFEF	TIM Channel 3 Vector (Low)
$FFF0	TIM Channel 2 Vector (High)
$FFF1	TIM Channel 2 Vector (Low)
$FFF2	TIM Channel 1 Vector (High)
$FFF3	TIM Channel 1 Vector (Low)
$FFF4	TIM Channel 0 Vector (High)
$FFF5	TIM Channel 0 Vector (Low)
$FFF6	DMA Vector (High)
$FFF7	DMA Vector (Low)
$FFF8	PLL Vector (High)
$FFF9	PLL Vector (Low)
$FFFA	IRQ1 Vector (High)
$FFFB	IRQ1 Vector (Low)
$FFFC	SWI Vector (High)
$FFFD	SWI Vector (Low)
$FFFE	Reset Vector (High)
$FFFF	Reset Vector (Low)

Low ↑ Priority ↓ High

2.3 Monitor ROM

The 240 bytes at addresses $FE10–$FEFF are reserved ROM addresses that contain the instructions for the monitor functions. (See **SECTION 11 MONITOR ROM (MON)**.)

SECTION 3
RAM

3

3.1 Introduction

This section describes the 1024 bytes of RAM.

3.2 Functional Description

Addresses $0050 through $044F 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 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.

3

SECTION 4
EPROM/OTPROM

4.1 Introduction

This section describes the non-volatile memory (EPROM/OTPROM).

4.2 Functional Description

An MCU with a quartz window has 36 Kbytes of erasable, programmable ROM (EPROM). The quartz window allows EPROM erasure by using ultraviolet light. In an MCU without the quartz window, the EPROM cannot be erased and serves as 36 Kbytes of one-time programmable ROM (OTPROM). An unprogrammed or erased location reads as $00. The following addresses are user EPROM/OTPROM locations:

- $6E00–$FDFF
- $FFE0–$FFFF (These locations are reserved for user-defined interrupt and reset vectors.)

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

NOTE

A security feature prevents viewing of the EPROM/OTPROM contents.[1]

---

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

4.3 EPROM/OTPROM Control Register (EPMCR)

The EPROM control register controls EPROM/OTPROM programming.

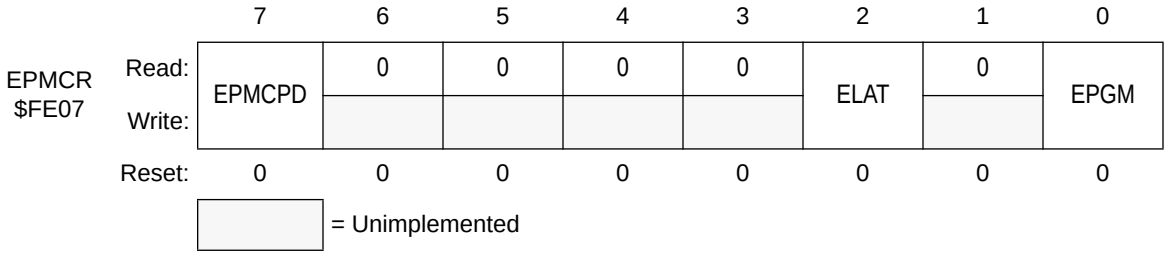


Figure 4-1. EPROM/OTPROM Control Register (EPMCR)

EPMCPD — EPROM/OTPROM Charge Pump Disable Bit

This read/write bit controls the charge pump used for reading the EPROM/OTPROM at low voltage. Write a logic one to the EPMCPD bit after reset to conserve power if the MCU is operating at 2.7 V or higher. Reset clears EPMCPD.

1 = Charge pump disabled
0 = Charge pump enabled

ELAT — EPROM/OTPROM Latch Control Bit

This read/write bit latches the address and data buses for programming the EPROM/OTPROM. Clearing ELAT also clears the EPGM bit. EPROM/OTPROM data cannot be read when ELAT is set.

1 = Buses configured for EPROM/OTPROM programming
0 = Buses configured for normal operation

EPGM — EPROM/OTPROM Program Control Bit

This read/write bit applies the programming voltage from the $\overline{IRQ1}/V_{PP}$ pin to the EPROM/OTPROM. To write to the EPGM bit, the ELAT bit must be set already. Reset clears the EPGM bit.

1 = EPROM/OTPROM programming power switched on
0 = EPROM/OTPROM programming power switched off

4.4 EPROM/OTPROM Programming Sequence

Use the following procedure to program a byte of EPROM/OTPROM:

1. Apply $V_{DD} + V_{HI}$ to the $\overline{IRQ1}/V_{PP}$ pin.
2. Set the ELAT bit.

NOTE

Writing logic ones to both the ELAT and EPGM bits with a single instruction sets only the ELAT bit. EPGM must be set by a separate instruction in the programming sequence.

---

3. Write to any user EPROM/OTPROM address.

NOTE

Writing to an invalid address prevents the programming voltage from being applied.

---

4. Set the EPGM bit.
5. Wait for a time, t_{EPGM}.
6. Clear the ELAT and EPGM bits.

4.5 Mask Option Register (MOR)

The mask option register is an EPROM/OTPROM byte that enables or disables MCU options. (See **SECTION 5 MASK OPTION REGISTER (MOR)**.)

SECTION 5
MASK OPTION REGISTER (MOR)

5.1 Introduction

This section describes the mask option register (MOR). The mask option register is an EPROM/OTPROM byte that enables or disables the following options:

- Operation of low-voltage inhibit module (LVI) during stop mode
- Resets caused by the LVI module
- Power to the LVI module
- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- EPROM/OTPROM security(1)
- STOP instruction
- Computer operating properly module (COP)

5.2 Functional Description

For reliable low-voltage operation and EPROM/OTPROM security, the MOR uses complementary bit sensing. Each MOR bit uses two EPROM cells that are in complementary states after programming.

NOTE

To correctly program each pair of complementary cells, program every MOR bit.

Reset has no effect on the MOR. Writing to the MOR has no effect. The MOR can be read at any time.

All MOR bits read as logic zeros after erasure with ultraviolet light.

---

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

NOTE

If the LVI module and the LVI reset signal are enabled, a reset occurs when V_{DD} falls to a voltage, LVI_{TRIPF}. Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises to a voltage, LVI_{TRIPR}, guaranteeing that when V_{DD} returns to normal, the MCU restarts with the correct MOR contents.

		Bit 7	6	5	4	3	2	1	Bit 0
MOR $001F	Read:	0	LVISTOP	LVIRST	LVIPWR	SSREC	SEC	STOP	COPD
	Write:								
	Reset:	Unaffected by reset							

= Unimplemented

Figure 5-1. Mask Option Register (MOR)

LVISTOP — LVI Enable in Stop Mode Bit

If the LVIPWR bit is at logic one, LVISTOP enables the LVI module to operate during stop mode.

1 = LVI not disabled by STOP instruction
0 = LVI disabled by STOP instruction

NOTE

If the LVIPWR bit is at logic one, the LVISTOP bit must be at logic zero to meet the minimum stop mode I_{DD} specification.

LVIPWR — LVI Power Enable Bit

LVIPWR enables the LVI module. (See **SECTION 18 LOW-VOLTAGE INHIBIT MODULE (LVI)**.)

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

LVIRST — LVI Reset Enable Bit

LVIRST enables the reset signal from the LVI module. (See **SECTION 18 LOW-VOLTAGE INHIBIT MODULE (LVI)**.)

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

SSREC — Short Stop Recovery Bit

SSREC enables the CPU to exit stop mode with a delay of 32 CGMXCLK cycles instead of a 4096-CGMXCLK cycle delay. (See **7.6.2 Stop Mode**.)

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

NOTE

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

SEC — EPROM/OTPROM Security Bit

SEC enables the EPROM/OTPROM security feature. Setting the SEC bit prevents dumping of the EPROM/OTPROM contents.

1 = EPROM/OTPROM security enabled
0 = EPROM/OTPROM security disabled

STOP — STOP Enable Bit

STOP enables the STOP instruction.

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

COPD — COP Disable Bit

COPD disables the COP module. (See **SECTION 16 COMPUTER OPERATING PROPERLY MODULE**.)

1 = COP module disabled
0 = COP module enabled

5.3 MOR Programming Sequence

Use the following procedure to program the mask option register:

1. Apply V_{DD} + V_{HI} to the $\overline{IRQ1}/V_{PP}$ pin.
2. Set the ELAT bit in the EPROM control register. (See **4.3 EPROM/OTPROM Control Register (EPMCR)**.)

NOTE

Writing logic ones to both the ELAT and EPGM bits with a single instruction sets only the ELAT bit. EPGM must be set by a separate instruction in the programming sequence.

---

3. Write to the mask option register.

NOTE

Writing to an invalid address prevents the programming voltage from being applied.

---

4. Set the EPGM bit.
5. Wait for a time, t_{EPGM}.
6. Clear the ELAT and EPGM bits.

SECTION 6
CENTRAL PROCESSOR UNIT (CPU)

6.1 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 number CPU08RM/AD) contains a description of the CPU instruction set, addressing modes, and architecture.

6.2 Features

Features of the CPU include the following:

- 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
- 8-MHz CPU Internal Bus Frequency
- 64-Kbyte Program/Data Memory Space
- 16 Addressing Modes
- Memory-to-Memory Data Moves Without Using Accumulator
- Fast 8-Bit by 8-Bit Multiply and 16-Bit by 8-Bit Divide Instructions
- Enhanced Binary-Coded Decimal (BCD) Data Handling
- Modular Architecture with Expandable Internal Bus Definition for Extension of Addressing Range beyond 64 Kbytes
- Low-Power Stop and Wait Modes

6.3 CPU Registers

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

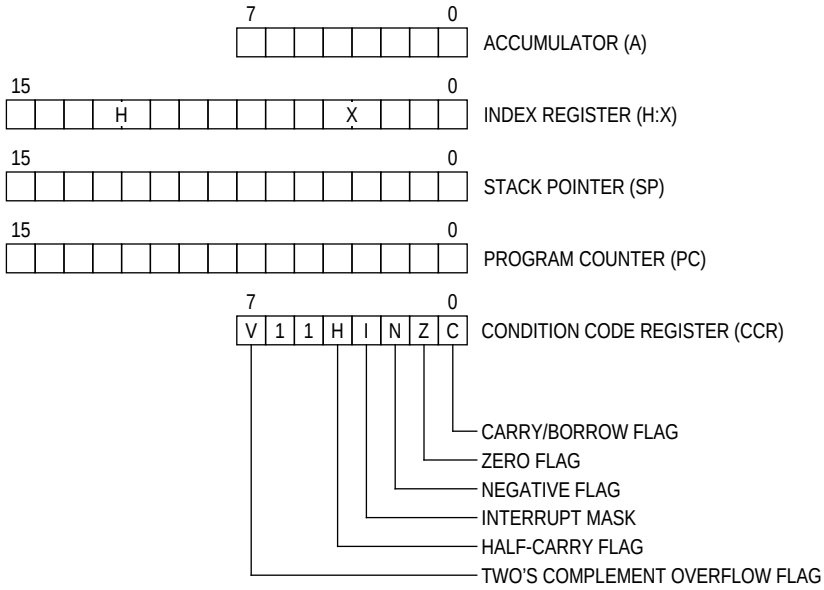


Figure 6-1. CPU Registers

6.3.1 Accumulator (A)

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

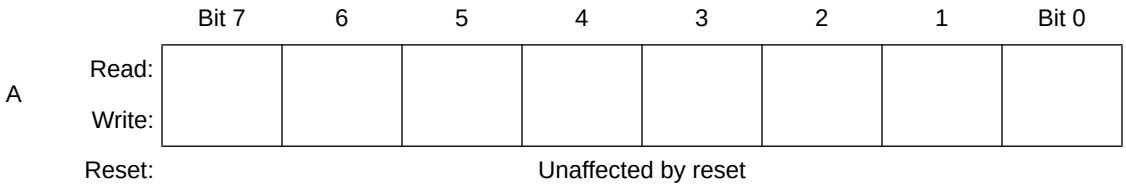


Figure 6-2. Accumulator (A)

6.3.2 Index Register (H:X)

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.

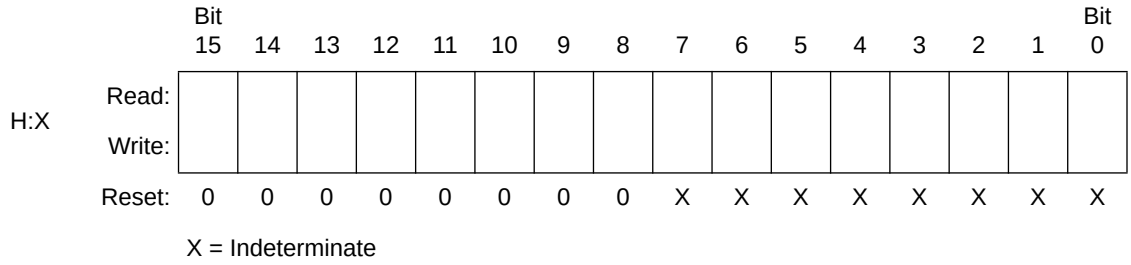


Figure 6-3. Index Register (H:X)

The index register can serve also as a temporary data storage location.

6.3.3 Stack Pointer (SP)

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

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

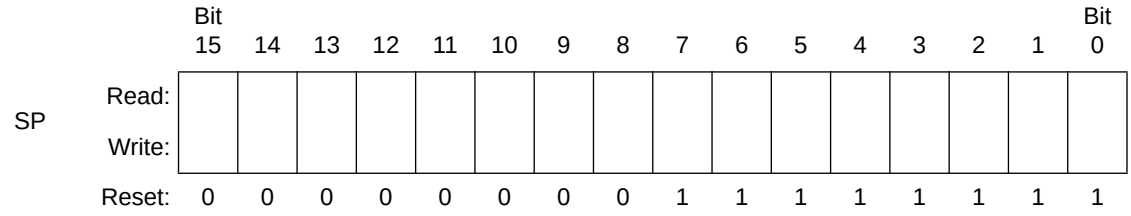


Figure 6-4. Stack Pointer (SP)

NOTE

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

6.3.4 Program Counter (PC)

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

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

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

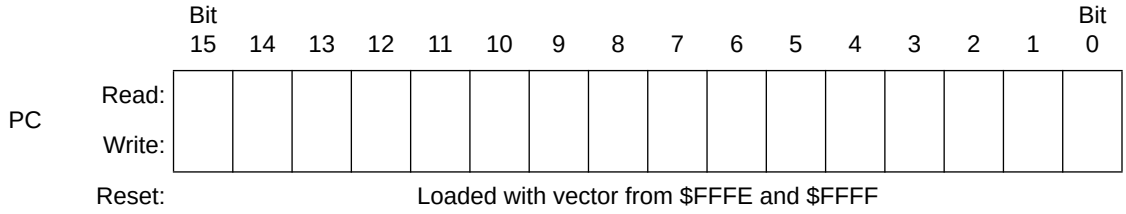


Figure 6-5. Program Counter (PC)

6.3.5 Condition Code Register (CCR)

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

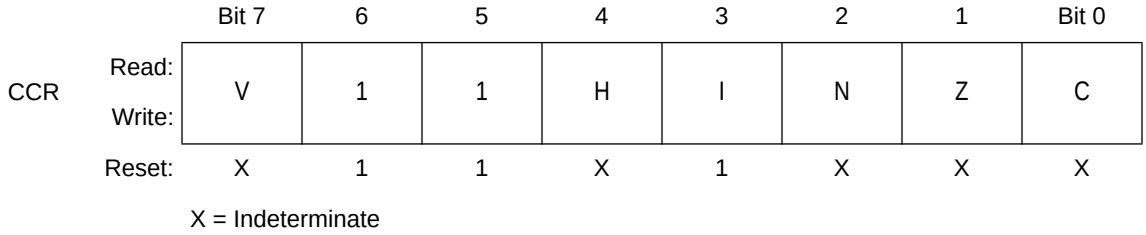


Figure 6-6. Condition Code Register (CCR)

V — Overflow Flag

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

1 = Overflow
0 = No overflow

H — Half-Carry Flag

The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an ADD or ADC operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C flags to determine the appropriate correction factor.

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

I — Interrupt Mask

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

1 = Interrupts disabled
0 = Interrupts enabled

NOTE

To maintain M6805 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 only be cleared by the clear interrupt mask software instruction (CLI).

N — Negative flag

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

1 = Negative result
0 = Non-negative result

Z — Zero flag

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

1 = Zero result
0 = Non-zero result

C — Carry/Borrow Flag

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

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

6.4 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 number CPU08RM/AD) for a description of the instructions and addressing modes and more detail about CPU architecture.

6.5 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 10 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.

SECTION 7
SYSTEM INTEGRATION MODULE (SIM)

7.1 Introduction

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

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

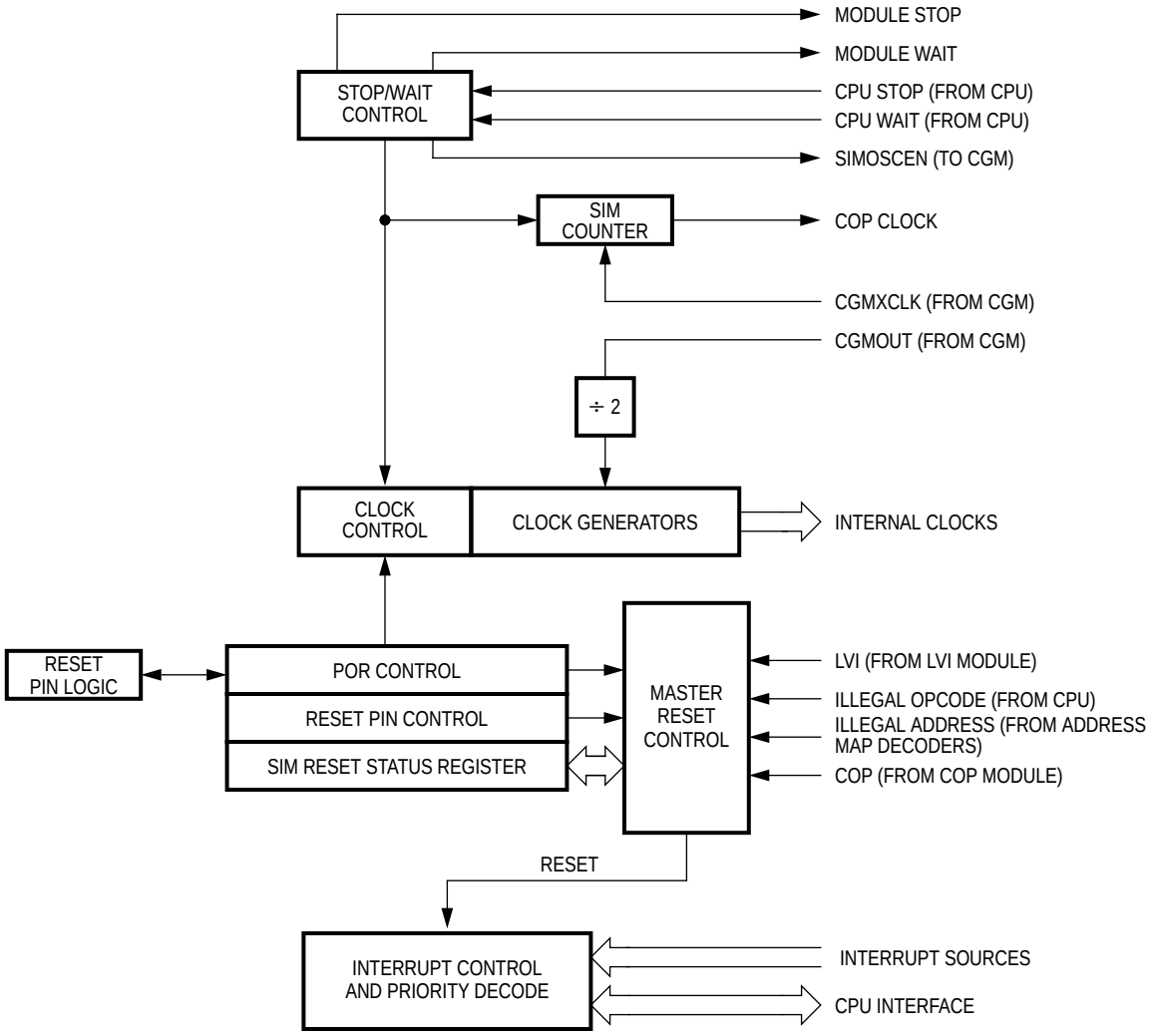


Figure 7-1. SIM Block Diagram

Table 7-1. SIM I/O Register Summary

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

R = Reserved for factory test

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

Table 7-2. Signal Name Conventions

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

7.2 SIM Bus Clock Control and Generation

The bus clock generator provides system clock signals for the CPU and peripherals on the MCU. The system clocks are generated from an incoming clock, CGMOUT, as shown in Figure 7-2. This clock can come from either an external oscillator or from the on-chip PLL. (See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**.)

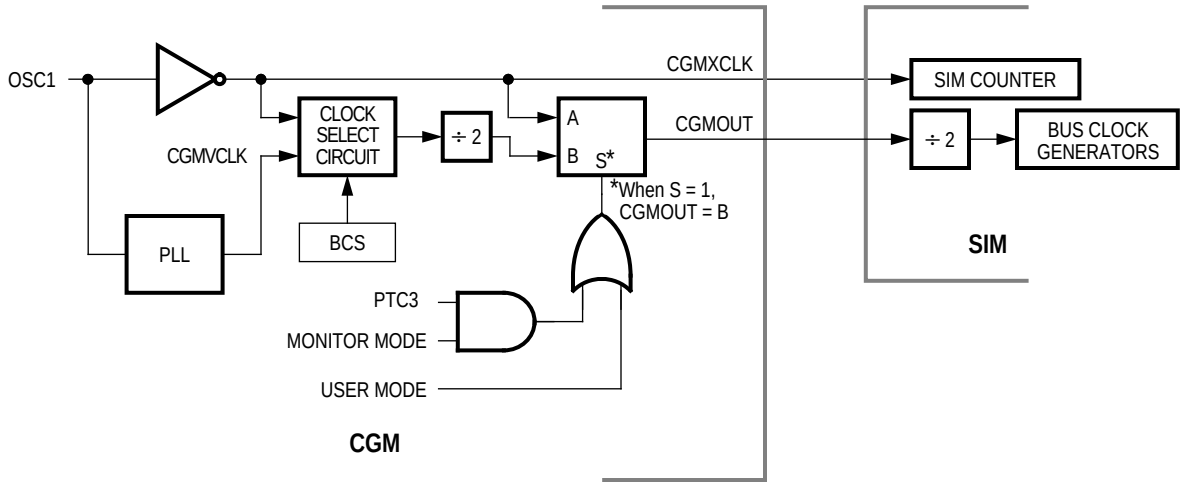


Figure 7-2. CGM Clock Signals

7.2.1 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. (See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**.)

7.2.2 Clock Start-Up from POR or LVI Reset

When the power-on reset module or the low-voltage inhibit 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.

7.2.3 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 **7.6.2 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.

7.3 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 **7.4 SIM Counter**), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See **7.7 SIM Registers**.)

7.3.1 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 7-3 for details. Figure 7-3 shows the relative timing.

Table 7-3. PIN Bit Set Timing

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

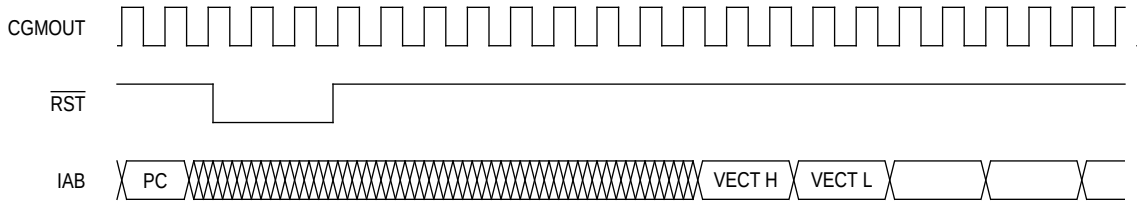


Figure 7-3. External Reset Timing

7.3.2 Active Resets from Internal Sources

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

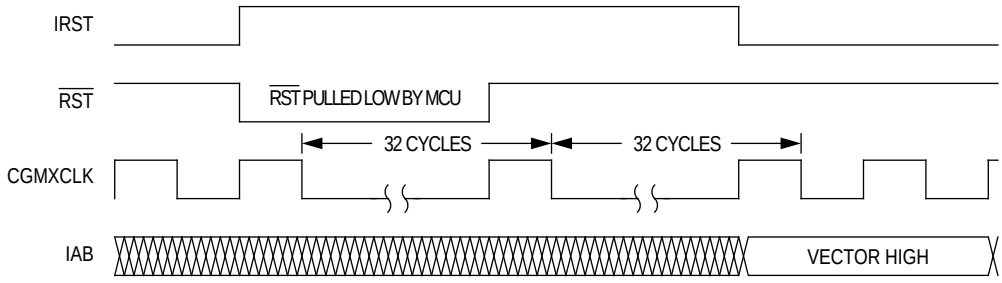


Figure 7-4. Internal Reset Timing

The COP reset is asynchronous to the bus clock.

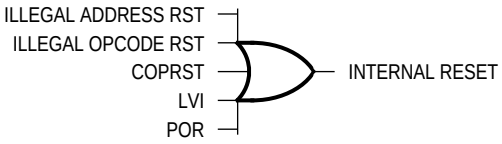


Figure 7-5. Sources of Internal Reset

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

7.3.2.1 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.

7

At power-on, the following events occur:

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

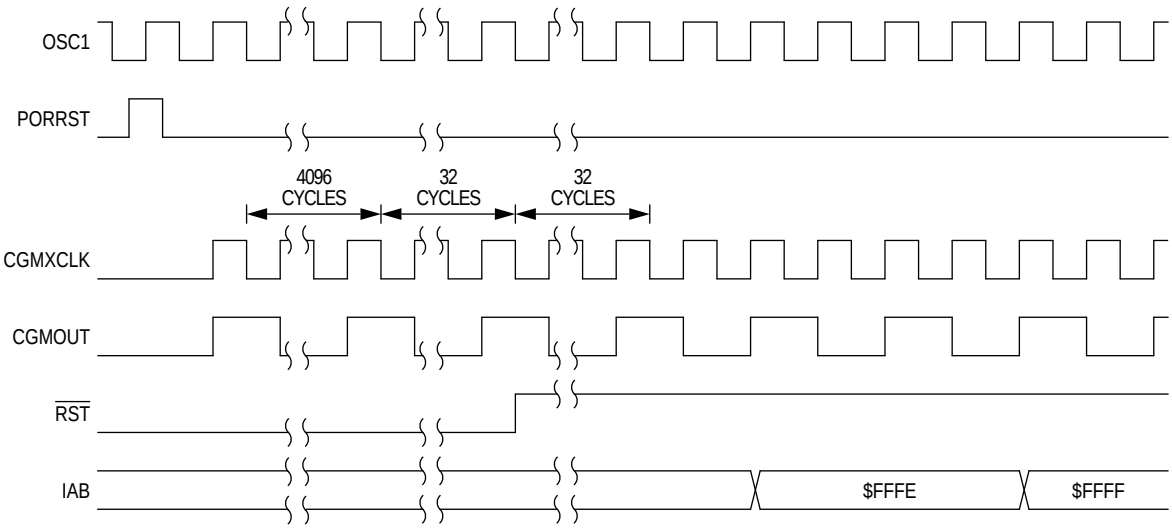


Figure 7-6. POR Recovery

7.3.2.2 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

7.3.2.3 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 zero, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset. The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

7.3.2.4 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.

7.3.2.5 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.

7.4 SIM Counter

The SIM counter is used by the power-on reset module (POR) and in stop mode recovery to allow the oscillator time to stabilize before enabling the internal bus (IBUS) clocks. The SIM counter also serves as a prescaler for the computer operating properly module (COP). The SIM counter overflow supplies the clock for the COP module. The SIM counter is 13 bits long and is clocked by the falling edge of CGMXCLK.

7.4.1 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.

7.4.2 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 start-up times from stop mode. External crystal applications should use the full stop recovery time, that is, with SSREC cleared.

7.4.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See **7.6.2 Stop Mode** for details.) The SIM counter is free-running after all reset states. (See **7.3.2 Active Resets from Internal Sources** for counter control and internal reset recovery sequences.)

7.5 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

7.5.1 Interrupts

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. **Figure 7-7. Interrupt Entry** shows interrupt entry timing. **Figure 7-9. Interrupt Recovery** shows interrupt recovery timing.

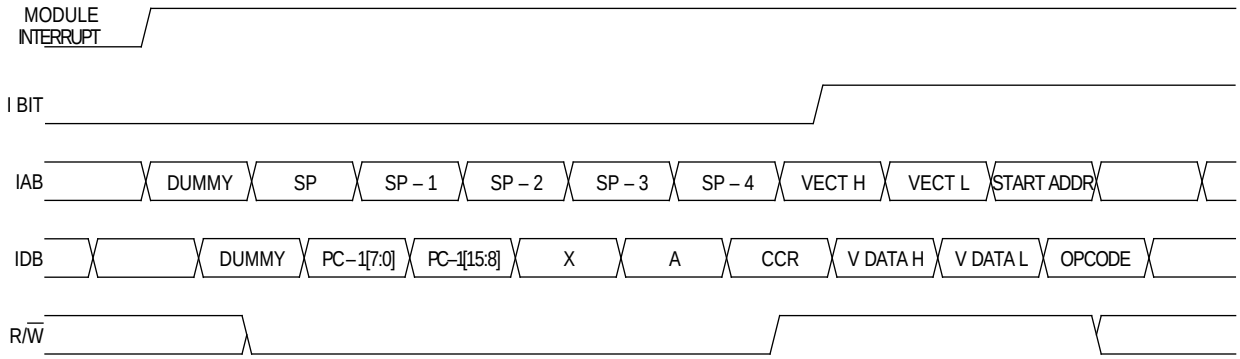


Figure 7-7. Interrupt Entry

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 7-8.)

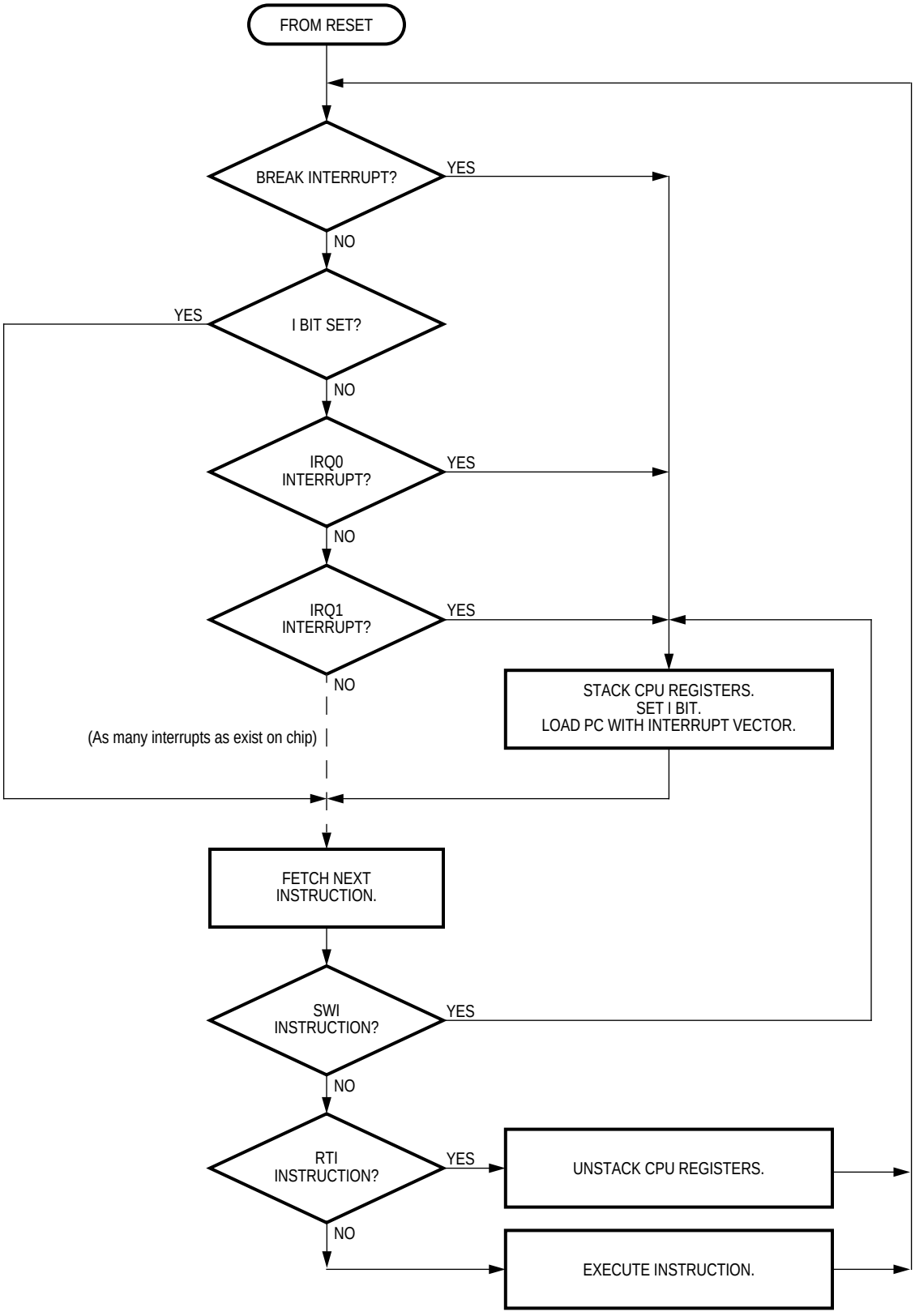


Figure 7-8. Interrupt Processing

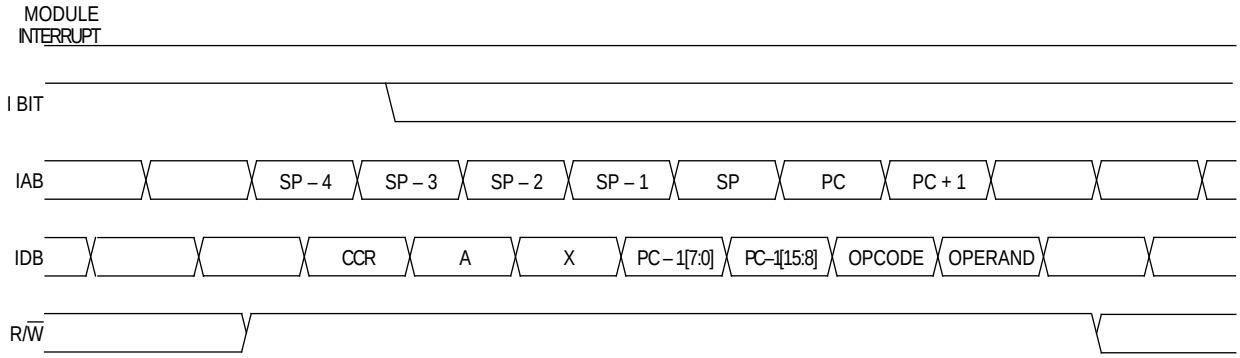


Figure 7-9. Interrupt Recovery

7.5.1.1 Hardware Interrupts

A hardware interrupt does not stop the current instruction. Processing of a hardware interrupt begins after completion of the current instruction. When the current instruction is complete, the SIM checks all pending hardware interrupts. If interrupts are not masked (I bit clear in the condition code register), and if the corresponding interrupt enable bit is set, the SIM proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. Figure 7-10 demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

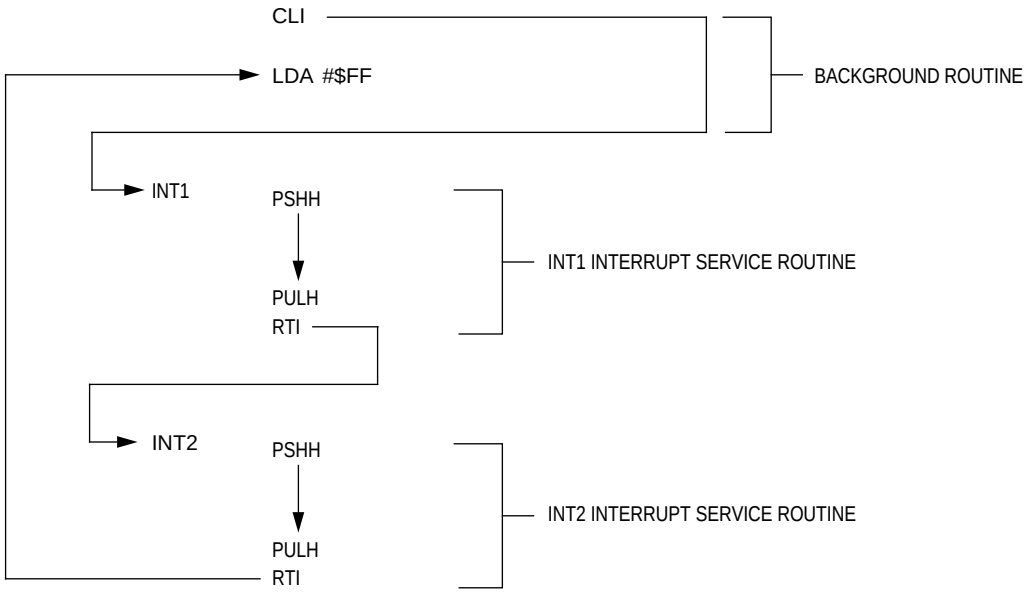


Figure 7-10. Interrupt Recognition Example

The LDA opcode is 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.

7.5.1.2 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.

7.5.2 Reset

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

7.5.3 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. (See **SECTION 10 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.

7.5.4 Status Flag Protection in Break Mode

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

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

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

7.6 Low-Power Modes

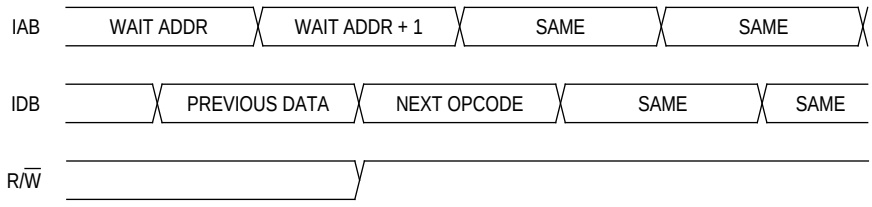
Executing the WAIT or STOP instruction puts the MCU in a low-power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

7.6.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. Figure 7-11 shows the timing for wait mode entry.

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

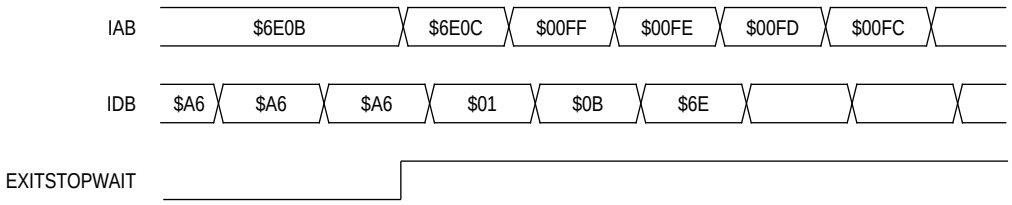
Wait mode can also be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the mask option register is logic zero, then the computer operating properly module (COP) is enabled and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



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

Figure 7-11. Wait Mode Entry Timing

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



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

Figure 7-12. Wait Recovery from Interrupt or Break

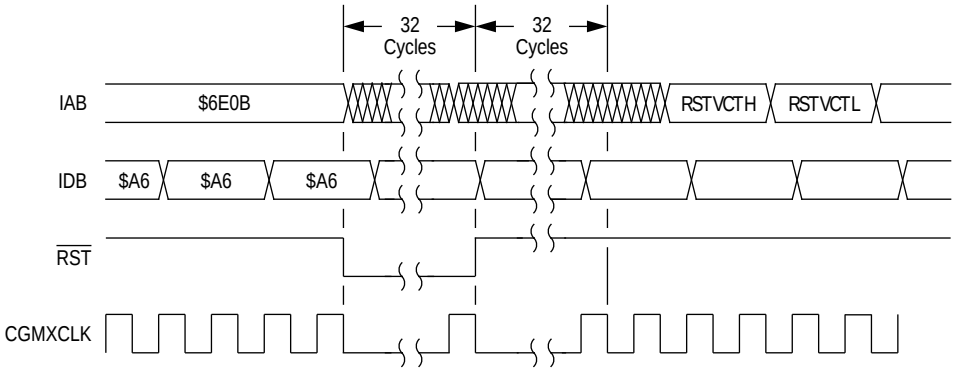


Figure 7-13. Wait Recovery from Internal Reset

7.6.2 Stop Mode

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

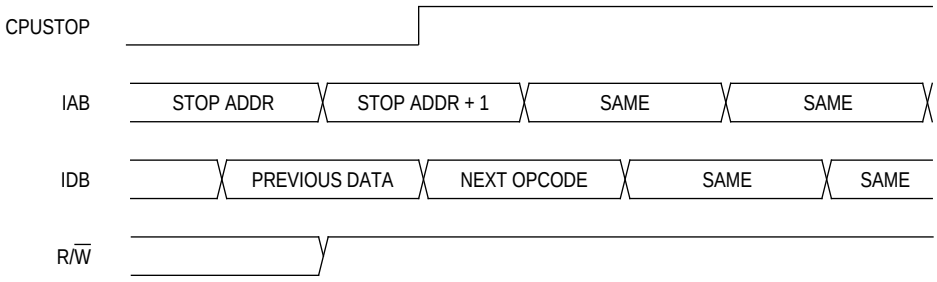
The SIM disables the clock generator module outputs (CGMOUT and CGMXCLK) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in the mask option register (MOR). If SSREC is set, stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down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

NOTE

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

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

The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. Figure 7-14 shows stop mode entry timing.



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

Figure 7-14. Stop Mode Entry Timing

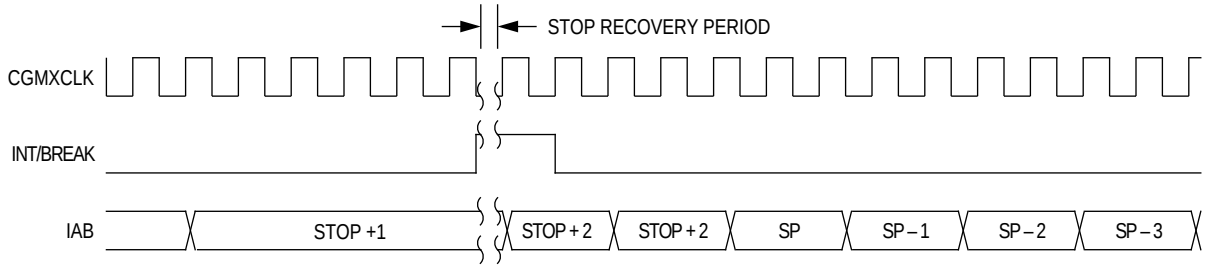


Figure 7-15. Stop Mode Recovery from Interrupt or Break

7.7 SIM Registers

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

Table 7-4. SIM Registers

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

7.7.1 SIM Break Status Register (SBSR)

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

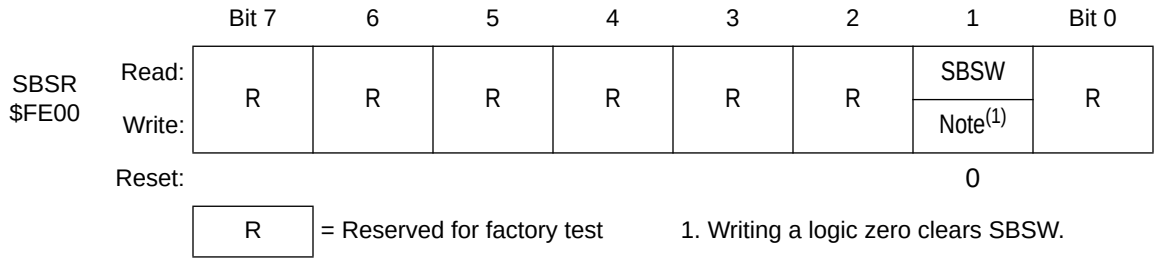


Figure 7-16. SIM Break Status Register (SBSR)

7

SBSW — SIM Break Stop/Wait

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

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

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it. The following code is an example of this. Writing zero to the SBSW bit clears it.

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

  HIBYTE   EQU    5

  LOBYTE   EQU    6

;          If not SBSW, do RTI

           BRCLR  SBSW,SBSR, 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
```

7.7.2 SIM Reset Status Register (SRSR)

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

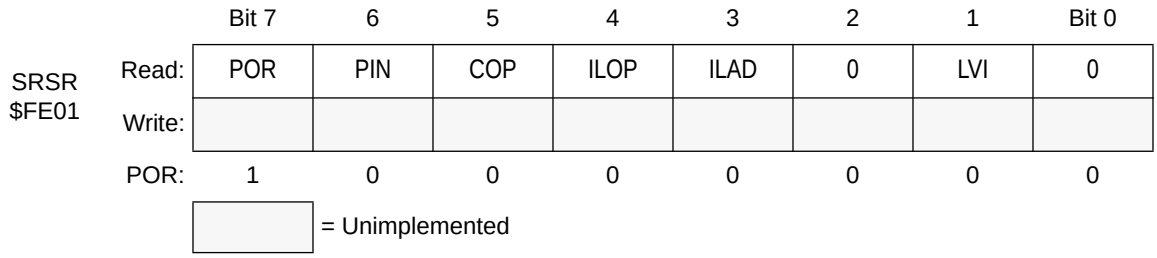


Figure 7-17. SIM Reset Status Register (SRSR)

7

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

LVI — Low-Voltage Inhibit Reset Bit
- 1 = Last reset was caused by the LVI circuit
- 0 = POR or read of SRSR

7.7.3 SIM Break Flag Control Register (SBFCR)

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

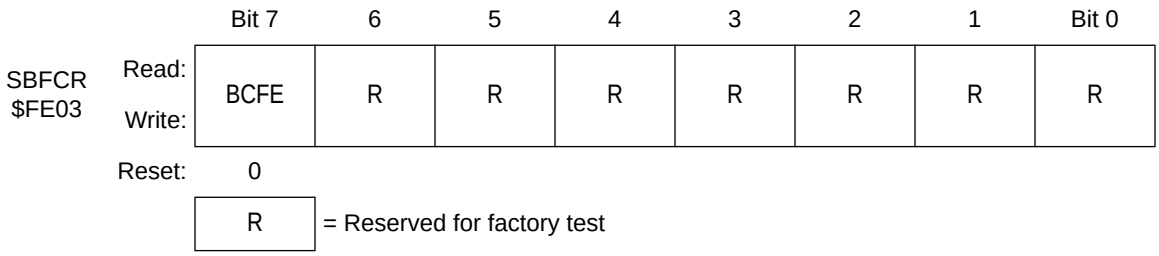


Figure 7-18. SIM Break Flag Control Register (SBFCR)

7

BCFE — Break Clear Flag Enable Bit

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

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

SECTION 8
CLOCK GENERATOR MODULE (CGM)

8.1 Introduction

This section describes the clock generator module (CGM, Version A). 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 frequency generator designed for use with 1-MHz to 16-MHz crystals or ceramic resonators. The PLL can generate an 8-MHz bus frequency without using a 32-MHz crystal.

8.2 Features

Features of the CGM include the following:

- Phase-Locked Loop with Output Frequency in Integer Multiples of the Crystal Reference
- Programmable Hardware Voltage-Controlled Oscillator (VCO) for Low-Jitter Operation
- Automatic Bandwidth Control Mode for Low-Jitter Operation
- Automatic Frequency Lock Detector
- CPU Interrupt on Entry or Exit from Locked Condition

8.3 Functional Description

The CGM consists of three major submodules:

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

Figure 8-1 shows the structure of the CGM.

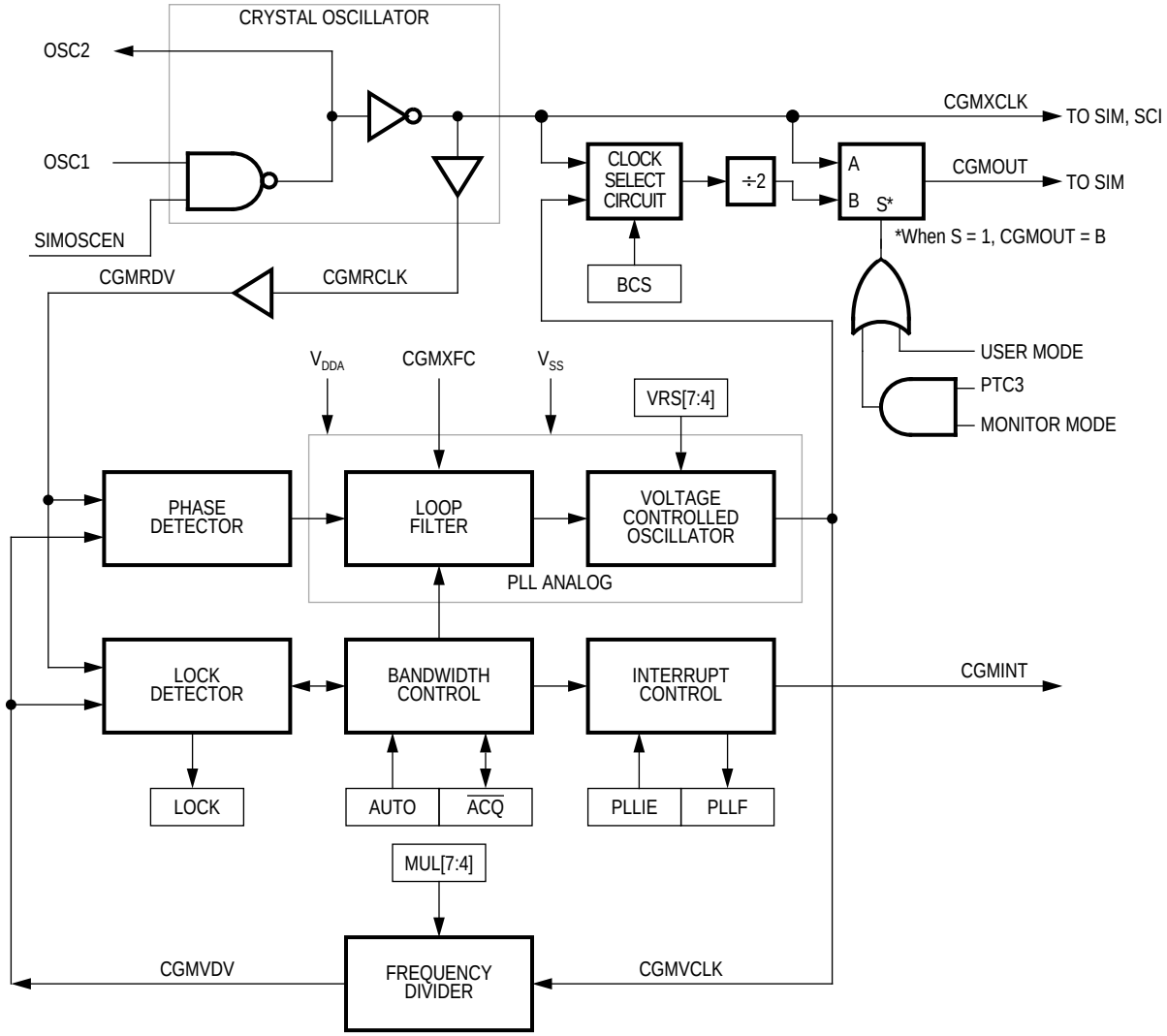


Figure 8-1. CGM Block Diagram

Table 8-1. CGM I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
PLL Control Register (PCTL)	PLLIE	PLLF	PLLON	BCS					$001C
PLL Bandwidth Control Register (PBWC)	AUTO	LOCK	$\overline{\text{ACQ}}$	XLD					$001D
PLL Programming Register (PPG)	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4	$001E

= Unimplemented

8.3.1 Crystal Oscillator Circuit

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

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

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

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

8.3.2 Phase-Locked Loop Circuit (PLL)

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

8.3.2.1 PLL Circuits

The PLL consists of the following circuits:

- Voltage-controlled oscillator (VCO)
- 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}, (4.9152 MHz) times a linear factor L, or (L)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 buffer. The buffer output is the final reference clock, CGMRDV, running at a frequency $f_{RDV} = f_{RCLK}$.

The VCO's output clock, CGMVCLK, running at a frequency f_{VCLK}, is fed back through a programmable modulo divider. The modulo divider reduces the VCO clock by a factor, N. The divider's output is the VCO feedback clock, CGMVDV, running at a frequency $f_{VDV} = f_{VCLK}/N$. (See **8.3.2.4 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 **8.3.2.2 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.

8.3.2.2 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 $\overline{ACQ}$ bit is clear in the PLL bandwidth control register. (See **8.5.2 PLL Bandwidth Control Register (PBWC)**.)
- Tracking mode — In tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source. (See **8.3.3 Base Clock Selector Circuit**.) The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{ACQ}$ bit is set.

8.3.2.3 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 **8.5.2 PLL Bandwidth Control Register (PBWC)**.) If PLL interrupts are enabled, the software can wait for a PLL interrupt request and then check the LOCK bit. If interrupts are disabled, software can poll the LOCK bit continuously (during PLL start-up, usually) or at periodic intervals. In either case, when the LOCK bit is set, the VCO clock is safe to use as the source for the base clock. (See **8.3.3 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 **8.6 Interrupts** for information and precautions on using interrupts.) The following conditions apply when the PLL is in automatic bandwidth control mode:

- The $\overline{\text{ACQ}}$ bit (see **8.5.2 PLL Bandwidth Control Register (PBWC)**) is a read-only indicator of the mode of the filter. (See **8.3.2.2 Acquisition and Tracking Modes**.)
- The $\overline{\text{ACQ}}$ bit is set when the VCO frequency is within a certain tolerance Δ_{TRK} and is cleared when the VCO frequency is out of a certain tolerance Δ_{UNT}. (See **8.9 Acquisition/Lock Time Specifications** 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 **8.9 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 **8.5.1 PLL Control Register (PCTL)**.)

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

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

8.3.2.4 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 a VCO frequency multiplier, N.

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

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

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

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

6. Select a VCO linear range multiplier, L.

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

where f_{NOM} = 4.9152 MHz

7. Calculate and verify the adequacy of the VCO programmed center-of-range frequency f_{VRS}.

$$f_{VRS} = (L)f_{NOM}$$

8. Verify the choice of N 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.

---

9. Program the PLL registers accordingly:
   a. In the upper 4 bits of the PLL programming register (PPG), program the binary equivalent of N.
   b. In the lower 4 bits of the PLL programming register (PPG), program the binary equivalent of L.

8.3.2.5 Special Programming Exceptions

The programming method described in **8.3.2.4 Programming the PLL** does not account for two possible exceptions. A value of zero for N or L is meaningless when used in the equations given. To account for these exceptions:

- A zero value for N is interpreted exactly the same as a value of one.
- A zero value for L disables the PLL and prevents its selection as the source for the base clock. (See **8.3.3 Base Clock Selector Circuit**.)

8.3.3 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 zero. This value would set up a condition inconsistent with the operation of the PLL, so that the PLL would be disabled and the crystal clock would be forced as the source of the base clock.

8.3.4 CGM External Connections

In its typical configuration, the CGM 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 8-2. Figure 8-2 shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

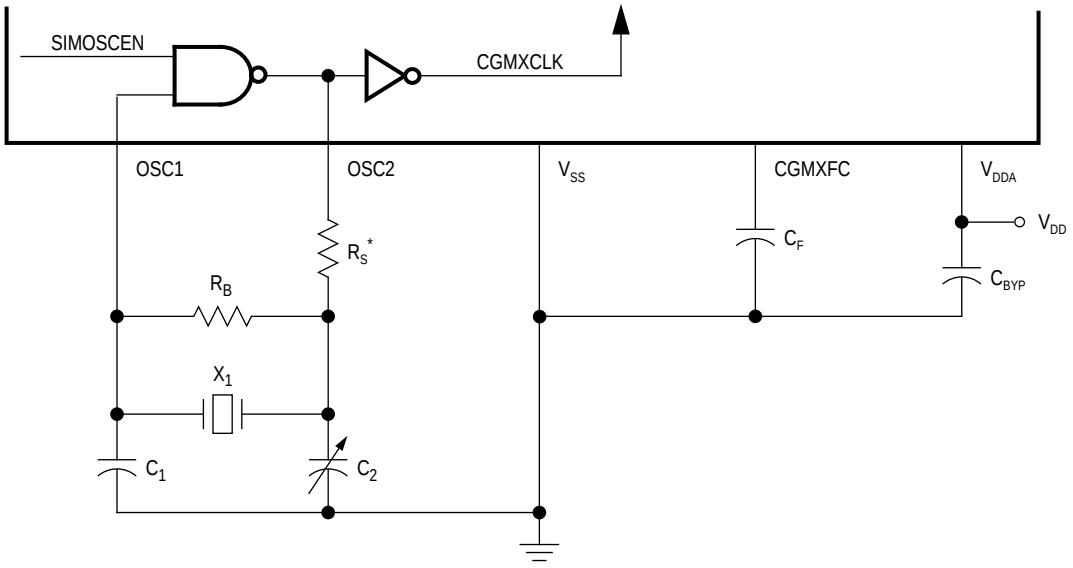
- Crystal, X_1
- Fixed capacitor, C_1
- Tuning capacitor, C_2 (can also be a fixed capacitor)
- Feedback resistor, R_B
- Series resistor, R_S (optional)

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 8-2 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. (See **8.9 Acquisition/Lock Time Specifications** for routing information and more information on the filter capacitor's value and its effects on PLL performance.)



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

Figure 8-2. CGM External Connections

8.4 I/O Signals

The following paragraphs describe the CGM I/O signals.

8.4.1 Crystal Amplifier Input Pin (OSC1)

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

8.4.2 Crystal Amplifier Output Pin (OSC2)

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

8.4.3 External Filter Capacitor Pin (CGMXFC)

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

NOTE

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

8.4.4 PLL Analog Power Pin (V_{DDA})

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

NOTE

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

8.4.5 Oscillator Enable Signal (SIMOSCEN)

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

8.4.6 Crystal Output Frequency Signal (CGMXCLK)

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

8.4.7 CGM Base Clock Output (CGMOUT)

CGMOUT is the clock output of the CGM. This signal goes to the SIM, which generates the MCU clocks. CGMOUT is a 50% duty cycle clock running at twice the bus frequency. CGMOUT is software programmable to be either the oscillator output, CGMXCLK, divided by two or the VCO clock, CGMVCLK, divided by two.

8.4.8 CGM CPU Interrupt (CGMINT)

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

8.5 CGM Registers

The following registers control and monitor operation of the CGM:

- PLL control register (PCTL) (See 8.5.1.)
- PLL bandwidth control register (PBWC) (See 8.5.2.)
- PLL programming register (PPG) (See 8.5.3.)

Figure 8-3 is a summary of the CGM registers.

		Bit 7	6	5	4	3	2	1	Bit 0
PCTL $001C	Read:	PLLIE	PLLF	PLLON	BCS	1	1	1	1
	Write:								
PBWC $001D	Read:	AUTO	LOCK	$\overline{ACQ}$	XLD	0	0	0	0
	Write:								
PPG $001E	Read:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
	Write:								

= Unimplemented

NOTES:

1. When AUTO = 0, PLLIE is forced to logic zero and is read-only.
2. When AUTO = 0, PLLF and LOCK read as logic zero.
3. When AUTO = 1, $\overline{ACQ}$ is read-only.
4. When PLLON = 0 or VRS[7:4] = $0, BCS is forced to logic zero 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 8-3. CGM I/O Register Summary

8.5.1 PLL Control Register (PCTL)

The PLL control register contains the interrupt enable and flag bits, the on/off switch, and the base clock selector bit.

		Bit 7	6	5	4	3	2	1	Bit 0
PCTL $001C	Read:	PLLIE	PLLF	PLLON	BCS	1	1	1	1
	Write:								
	Reset:	0	0	1	0	1	1	1	1

= Unimplemented

Figure 8-4. PLL Control Register (PCTL)

PLLIE — PLL Interrupt Enable Bit

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

1 = PLL interrupts enabled
0 = PLL interrupts disabled

PLLF — PLL Interrupt Flag Bit

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

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

NOTE

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

PLLON — PLL On Bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). (See **8.3.3 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 **8.3.3 Base Clock Selector Circuit**.) Reset and the STOP instruction 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 **8.3.3 Base Clock Selector Circuit**.)

PCTL[3:0] — Unimplemented bits

These bits provide no function and always read as logic ones.

8.5.2 PLL Bandwidth Control Register (PBWC)

The PLL bandwidth control register does the following:

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

		Bit 7	6	5	4	3	2	1	Bit 0
PBWC $001D	Read:	AUTO	LOCK	$\overline{ACQ}$	XLD	0	0	0	0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 8-6. 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 $\overline{ACQ}$ bit before turning on the PLL. Reset clears the AUTO bit.

1 = Automatic bandwidth control
0 = Manual bandwidth control

LOCK — Lock Indicator Bit

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock, CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as logic zero and has no meaning. 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

XLD — Crystal Loss Detect Bit

When the VCO output, CGMVCLK, is driving CGMOUT, this read/write bit can indicate whether the crystal reference frequency is active or not. To check the status of the crystal reference, do the following:

1. Write a logic one to XLD.
2. Wait N $\times$ 4 cycles. (N is the VCO frequency multiplier.)
3. Read XLD.

1 = Crystal reference is not active
0 = Crystal reference is active

The crystal loss detect function works only when the BCS bit is set, selecting CGMVCLK to drive CGMOUT. When BCS is clear, XLD always reads as logic zero.

PBWC[3:0] — Reserved for Test

These bits enable test functions not available in user mode. To ensure software portability from development systems to user applications, software should write zeros to PBWC[3:0] whenever writing to PBWC.

8.5.3 PLL Programming Register (PPG)

The PLL programming register contains the programming information for the modulo feedback divider and the programming information for the hardware configuration of the VCO.

PPG $001E	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
Reset:	0	1	1	0	0	1	1	0

Figure 8-7. PLL Programming Register (PPG)

MUL[7:4] — Multiplier Select Bits

These read/write bits control the modulo feedback divider that selects the VCO frequency multiplier, N. (See **8.3.2.1 PLL Circuits** and **8.3.2.4 Programming the PLL**.) A value of $0 in the multiplier select bits configures the modulo feedback divider the same as a value of $1. Reset initializes these bits to $6 to give a default multiply value of 6.

Table 8-4. VCO Frequency Multiplier (N) Selection

MUL7:MUL6:MUL5:MUL4	VCO Frequency Multiplier (N)
0000	1
0001	1
0010	2
0011	3
↓	↓
1101	13
1110	14
1111	15

8

NOTE

The multiplier select bits have built-in protection that prevents them from being written when the PLL is on (PLLON = 1).

VRS[7:4] — VCO Range Select Bits

These read/write bits control the hardware center-of-range linear multiplier L, which controls the hardware center-of-range frequency f_{VRS}. (See **8.3.2.1 PLL Circuits**, **8.3.2.4 Programming the PLL**, and **8.5.1 PLL Control Register (PCTL)**.) VRS[7:4] cannot be written when the PLLON bit in the PLL control register (PCTL) is set. (See **8.3.2.5 Special Programming Exceptions**.) A value of $0 in the VCO range select bits disables the PLL and clears the BCS bit in the PCTL. (See **8.3.3 Base Clock Selector Circuit** and **8.3.2.5 Special Programming Exceptions** for more information.) Reset initializes the bits to $6 to give a default range multiply value of 6.

NOTE

The VCO range select bits have built-in protection that prevents them from being written when the PLL is on (PLLON = 1) and prevents selection of the VCO clock as the source of the base clock (BCS = 1) if the VCO range select bits are all clear.

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

8.6 Interrupts

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

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

NOTE

Software can select the 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.

8.7 Special Modes

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

8.7.1 Wait Mode

The WAIT instruction does not affect the CGM. 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.

8.7.2 Stop Mode

When the STOP instruction executes, the SIM drives the SIMOSCEN signal low, disabling the CGM and holding low all CGM 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.

8.8 CGM 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 **7.7.3 SIM Break Flag Control Register (SBFCR)**.)

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

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

8.9 Acquisition/Lock Time Specifications

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

8.9.1 Acquisition/Lock Time Definitions

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

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

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

- Acquisition time, t_{ACQ}, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the tracking mode entry tolerance, Δ_{TRK}. Acquisition time is based on an initial frequency error, $(f_{DES} – f_{ORIG})/f_{DES}$, of not more than ±100%. In automatic bandwidth control mode (see **8.3.2.3 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%. In automatic bandwidth control mode, lock time expires when the LOCK bit becomes set in the PLL bandwidth control register (PBWC). (See **8.3.2.3 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.

8.9.2 Parametric Influences on Reaction Time

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

The most critical parameter which affects the reaction times of the PLL is the reference frequency, f_{RDV}. This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, the corrections must be small compared to the desired frequency, so several corrections are required to reduce the frequency error. Therefore, the slower the reference the longer it takes to make these corrections. This parameter is also under user control via the choice of crystal frequency f_{XCLK}.

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

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

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

8.9.3 Choosing a Filter Capacitor

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

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

For acceptable values of C_{FACT}, see **8.9 Acquisition/Lock Time Specifications**. For the value of V_{DDA}, choose the voltage potential at which the MCU is operating. If the power supply is variable, choose a value near the middle of the range of possible supply values.

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

8.9.4 Reaction Time Calculation

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

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

The K factor in the equations is derived from internal PLL parameters. K_{ACQ} is the K factor when the PLL is configured in acquisition mode, and K_{TRK} is the K factor when the PLL is configured in tracking mode. (See **8.3.2.2 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 the inverse proportionality between the lock time and the reference frequency.

In automatic bandwidth control mode the acquisition and lock times are quantized into units based on the reference frequency. (See **8.3.2.3 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 above.

In manual mode, it is usually necessary to wait considerably longer than t_{LOCK} before selecting the PLL clock (see **8.3.3 Base Clock Selector Circuit**), because the factors described in **8.9.2 Parametric Influences on Reaction Time** may slow the lock time considerably.

SECTION 9
DIRECT MEMORY ACCESS MODULE (DMA)

9.1 Introduction

This section describes the direct memory access module (DMA, Version A). The DMA can perform data transfers to and from any two CPU-addressable locations without CPU intervention.

9.2 Features

Features of the DMA include the following:

- Modular Architecture
- Service Request-Driven Operation Without CPU Intervention
- Three Independent Channels
- Byte or Word Transfer Capability
- Block Transfers and Loop Transfers
- CPU Interrupt Capability on Completion of Block Transfer or on Loop Restart
- Programmable DMA Bus Bandwidth (25%, 50%, 67%, or 100% of Total Bus Bandwidth)
- Programmable DMA Service Request/CPU Interrupt Request Priority
- Programmable DMA Enable during Wait Mode
- Block Transfers Up to 256 Bytes
- Expandable Architecture Up to Seven Channels and Eight Transfer Source Inputs

9.3 Functional Description

The DMA is a coprocessor for servicing peripheral devices that require data block transfers. For transmitting or receiving blocks of data to or from peripherals, DMA transfers are faster and more code-efficient than CPU interrupts. The following tasks that contribute to CPU interrupt overhead are not part of a DMA transfer:

- Stacking and unstacking CPU registers
- Loading interrupt vectors
- Loading address pointers
- Incrementing address pointers
- Storing address pointers
- Clearing interrupt flags
- Returning from interrupt

Once the DMA is initialized to transfer a block of data, a DMA service request usually requires only two bus cycles per 8-bit byte or four cycles per 16-bit word to transfer the source data to a destination.

Figure 9-1 shows the structure of the DMA. Each DMA channel can transfer data independently between any addresses in the memory map.

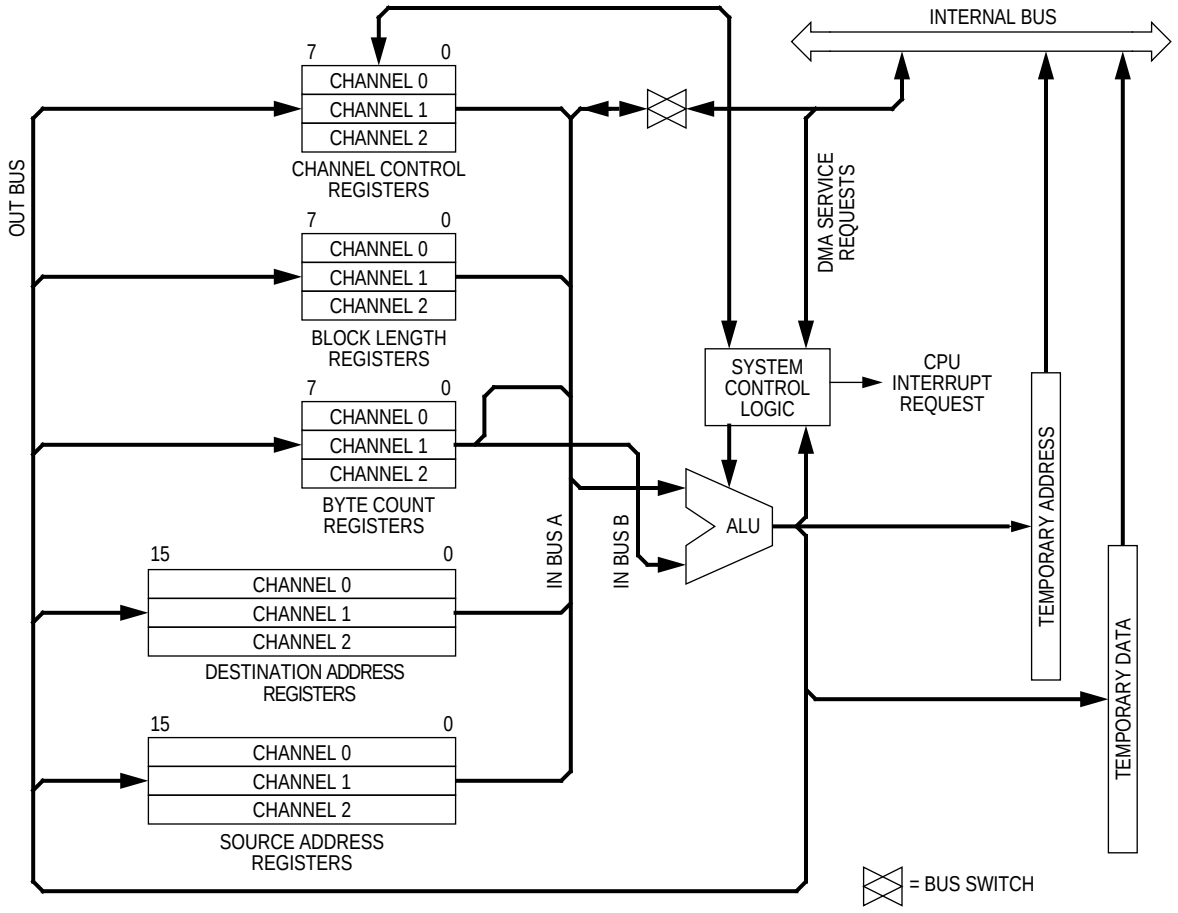


Figure 9-1. DMA Module Block Diagram

Table 9-1. DMA I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Ch. 0 Source Addr. Reg. High (D0SH)	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	$0034
Ch. 0 Source Addr. Reg. Low (D0SL)	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	$0035
Ch. 0 Destination Addr. Reg. High (D0DH)	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	$0036
Ch. 0 Destination Addr. Reg. Low (D0DL)	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	$0037
Ch. 0 Control Reg. (D0C)	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0	$0038
Ch. 0 Block Length Reg. (D0BL)	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0	$0039
Ch. 0 Byte Count Reg. (D0BC)	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0	$003B
Ch. 1 Source Addr. Reg. High (D1SH)	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	$003C
Ch. 1 Source Addr. Reg. Low (D1SL)	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	$003D
Ch. 1 Destination Addr. Reg. High (D1DH)	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	$003E
Ch. 1 Destination Addr. Reg. Low (D1DL)	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	$003F
Ch. 1 Control Reg. (D1C)	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0	$0040
Ch. 1 Block Length Reg. (D1BL)	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0	$0041
Ch. 1 Byte Count Reg. (D1BC)	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0	$0043
Ch. 2 Source Addr. Reg. High (D2SH)	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	$0044
Ch. 2 Source Addr. Reg. Low (D2SL)	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	$0045
Ch. 2 Destination Addr. Reg. High (D2DH)	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8	$0046
Ch. 2 Destination Addr. Reg. Low (D2DL)	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	$0047
Ch. 2 Control Reg. (D2C)	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0	$0048
Ch. 2 Block Length Reg. (D2BL)	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0	$0049
Ch. 2 Byte Count Reg. (D2BC)	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0	$004B
DMA Control Reg. 1 (DC1)	BB1	BB0	TEC2	IEC2	TEC1	IEC1	TEC0	IEC0	$004C
DMA Status and Control Reg. (DSC)	DMAP	L2	L1	L0	DMAWE	IFC2	IFC1	IFC0	$004D
DMA Control Reg. 2 (DC2)	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0	$004E

9.3.1 DMA/CPU Timing

When the DMA transfers data, it takes control of the address bus, data bus, and R/$\overline{W}$ line. During DMA transfers, the DMA signals the SIM to suspend the CPU clocks. The state of the CPU remains unchanged until the end of the DMA transfer when the DMA relinquishes control of the buses and R/$\overline{W}$ line. Then the CPU resumes operation as though nothing had happened.

Figure 9-2 and Figure 9-3 show the timing of DMA transfers.

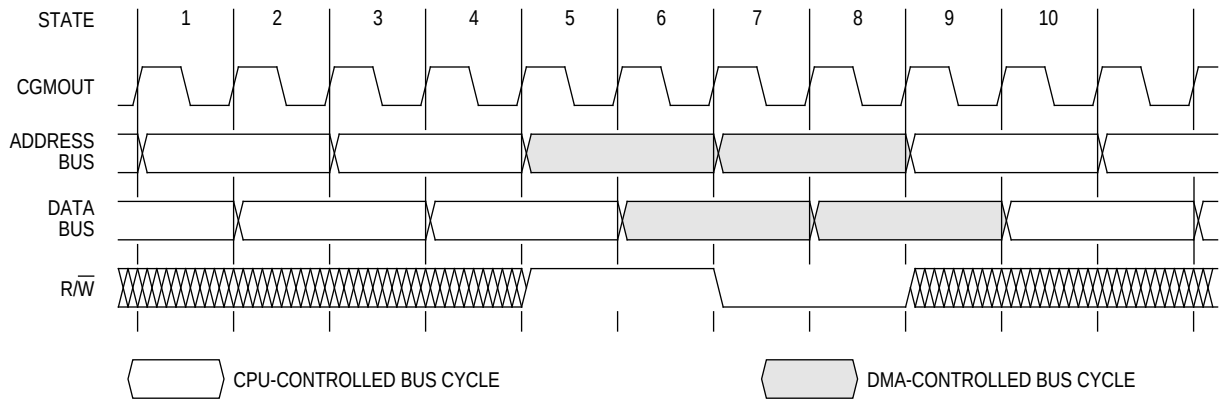


Figure 9-2. Single Byte Transfer Timing (Any DMA Bus Bandwidth)

Table 9-2. DMA Byte Transfer Activity

State 1	DMA service request occurs.
State 2	DMA arbitrates channel priority.
State 3	DMA generates internal control signals.
State 4	DMA calculates source address. DMA latches source address in temporary register.
State 5	DMA drives source address onto address bus. DMA drives R/$\overline{W}$ line high. DMA calculates destination address. DMA latches destination address into temporary register.
State 6	DMA latches source data into temporary register. DMA increments byte count register.
State 7	DMA drives destination address onto address bus. DMA drives R/$\overline{W}$ line low. DMA subtracts byte count register from block length register. If difference = 0, DMA disables channel by clearing TECx bit. (See 9.6.1.) If difference = 0 and IECx = 1, DMA generates CPU interrupt request. (See 9.6.1.)
State 8	DMA drives source data onto data bus.
State 9	DMA releases address bus and R/$\overline{W}$ line to CPU.
State 10	DMA releases data bus to CPU.

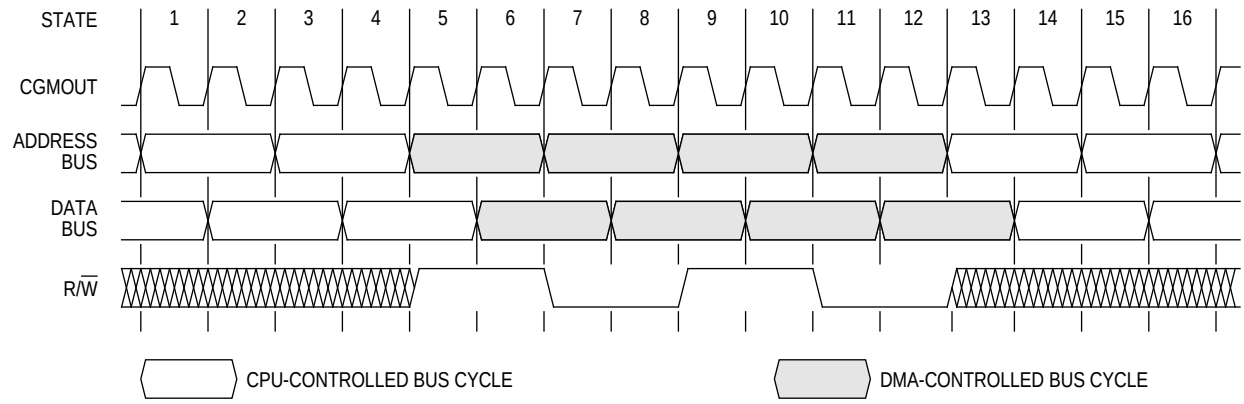


Figure 9-3. Single Word Transfer Timing (100% DMA Bus Bandwidth)

Table 9-3. DMA Word Transfer Activity

State 1	DMA service request occurs.
State 2	DMA arbitrates channel priority.
State 3	DMA generates internal control signals.
State 4	DMA calculates low byte of source address. DMA latches low byte of source address in temporary register.
State 5	DMA drives low byte of source address onto address bus. DMA drives R/$\overline{W}$ line high. DMA calculates low byte of destination address. DMA latches low byte of destination address into temporary register.
State 6	DMA latches low byte of source data into temporary register. DMA increments byte count register.
State 7	DMA drives low byte of destination address onto address bus. DMA drives R/$\overline{W}$ line low. DMA subtracts byte count register from block length register. If difference = 0, DMA deactivates channel. If difference = 0 and IECx = 1, DMA generates CPU interrupt request. (See 9.6.1.)
State 8	DMA drives low byte of source data onto data bus. DMA calculates high byte of source address. DMA latches high byte of source address into temporary register.
State 9	DMA drives the high byte of source address onto address bus. DMA drives R/$\overline{W}$ line high. DMA calculates high byte of destination address. DMA latches high byte of destination address in temporary register.
State 10	DMA latches high byte of source data into temporary register. DMA increments the byte count register.
State 11	DMA drives high byte of destination address onto address bus. DMA drives R/$\overline{W}$ line low. DMA subtracts byte count register from block length register. If difference = 0, DMA disables channel by clearing TECx bit. (See 9.6.1.) If difference = 0 and IECx bit set, CPU receives interrupt request. (See 9.6.1.)

Table 9-3. DMA Word Transfer Activity (Continued)

State 12	DMA drives high byte of destination address onto address bus.
State 13	DMA releases the address bus and R/$\overline{W}$ line to CPU.
State 14	DMA releases data bus to CPU.

The following procedure shows how to program a DMA transfer on a selected channel:

1. In DMA control register 1 (DC1), disable the channel by clearing the TECx bit (see 9.6.1).
2. In the source address registers (DxSH and DxSL), write the source base address (see 9.6.5).
3. In the destination address registers (DxDH and DxDL), write the destination base address (see 9.6.6).
4. In the DMA channel x control register (DxC), make the following selections (see 9.6.4):
   a. Select increment, decrement, or remain static for the source and destination addresses by writing to the source/destination address control bits, SDC[3:0].
   b. Select 8-bit or 16-bit data by writing to the byte/word control bit, BWC.
   c. Assign a DMA channel to the DMA transfer source input by writing to the DMA transfer source bits, DTS[2:0].
5. In the channel x DMA block length register (DxBL), write the number of bytes to transfer (see 9.6.7). For word transfers, the block length number is two times the number of words.
6. In the DMA status and control register (DSC), make the following selections (see 9.6.2):
   a. Enable or disable looping of the source and destination addresses by writing to the loop enable bit, Lx.
   b. Select DMA service request/CPU interrupt request priority by writing to the DMA priority bit, DMAP.
   c. Enable or disable DMA transfers during wait mode by writing to the DMA wait enable bit, DMAWE.

7. In DMA control register 1 (DC1), make the following selections (see 9.6.1):
   a. Enable the DMA channel x by writing to the transfer enable bit, TECx.
   b. Enable or disable DMA channel x to generate CPU interrupts on transfer completion by writing to the CPU interrupt enable bit, IECx.
   c. Select the DMA bus bandwidth by writing to the bus bandwidth control bits, BB0 and BB1.
8. To initiate the DMA transfer with software, set the software initiate bit, SWIx, in DMA control register 2 (DC2) (see 9.6.3).

9.3.2 Hardware-Initiated DMA Service Requests

The following sources can generate DMA service requests:

- Timer interface module (TIM) — The TIM can generate the following DMA service requests:
  – TIM channel 0 input capture/output compare
  – TIM channel 1 input capture/output compare
  – TIM channel 2 input capture/output compare
  – TIM channel 3 input capture/output compare
- Serial peripheral interface module (SPI) — The SPI can generate the following DMA service requests:
  – SPI receiver full
  – SPI transmitter empty
- Serial communications interface module (SCI) — The SCI can generate the following DMA service requests:
  – SCI receiver full
  – SCI transmitter empty

The DMA has eight inputs for transfer sources. Each DMA transfer source input corresponds to one of the above DMA service requests. To enable a transfer on one of the three DMA channels, software must first assign the channel to one of the transfer source inputs. The channel control register of each channel determines its transfer source assignment. (See **Table 9-8. DMA Transfer Source Selection**.)

9.3.3 Software-Initiated DMA Service Requests

Software can initiate a DMA service request by writing to DMA control register 2 (DC2). A software-initiated transfer begins when the following conditions are met:

- The channel is enabled by the channel x transfer enable bit, TECx, in DMA control register 1 (DC1).
- The channel is assigned to a DMA transfer source input by the DMA transfer source bits, DTS2–0, in the channel x control register (DxC).
- The corresponding software initiate bit, SWIx, in DMA control register 2 (DC2) is set, enabling a transfer on the transfer source input to which the channel is assigned.

During a DMA transfer on channel x, the channel x byte count register increments with every byte transferred. When the value in the channel x byte count register matches the value in the channel x block length register, the channel x CPU interrupt flag, IFCx, becomes set. If the channel x CPU interrupt enable bit, IECx, is also set, the DMA issues a CPU interrupt request.

9.3.4 DMA Latency

When one DMA channel is active, the normal DMA latency is two cycles. Writing to the destination/source address registers, the channel control registers, or the block length registers of another DMA channel during a transfer adds three cycles to DMA latency.

If more than one DMA channel is active, the latency of lower-priority channels increases.

If two or more DMA channels have pending service requests, at least one CPU cycle executes between each channel transfer.

9.3.5 DMA Source/Destination Address Calculation

Three 16-bit buses connect the 16-bit DMA arithmetic/logic unit (ALU) to the DMA channel registers. During a DMA transfer, the DMA ALU does the following:

- Calculates the transfer source and transfer destination addresses
- Increments the DMA byte count register for each byte transferred
- Determines when a block or loop transfer is complete by comparing the DMA byte count register with the value programmed in the DMA block length register

The DMA source address register and destination address register contain the base addresses for a DMA transfer. The DMA ALU uses these address registers as base pointers when it starts the transfer. The DMA byte count register contains the number of bytes transferred in the current DMA operation. The DMA ALU uses the source/destination address registers and the byte count register to calculate the actual source and destination addresses in the following manner:

- When an address is configured to increment, the DMA ALU adds the byte count register to the base address.
- When an address is configured to decrement, the DMA ALU subtracts the byte counter register from the base address.
- When an address is configured to remain static, the DMA ALU uses the base address as is.

The DMA can be programmed to stop after a number of bytes is transferred or to loop back to the base addresses and continue the transfer.

Figure 9-4 through Figure 9-12 show how the DMA calculates source and destination addresses.

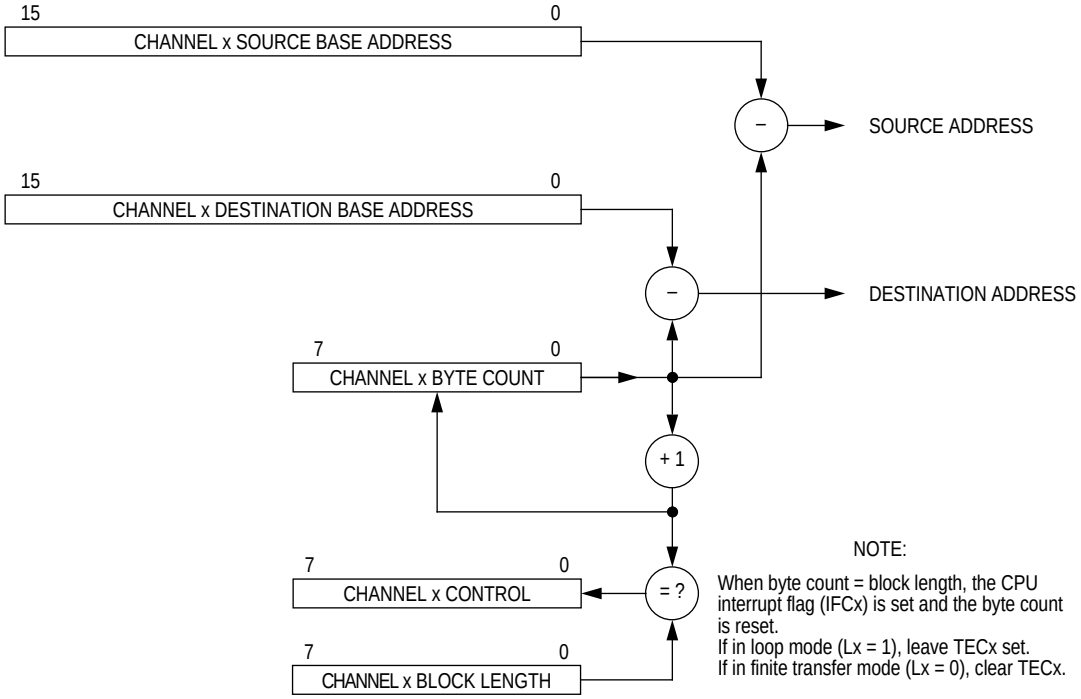


Figure 9-4. Decremented Source and Decremented Destination

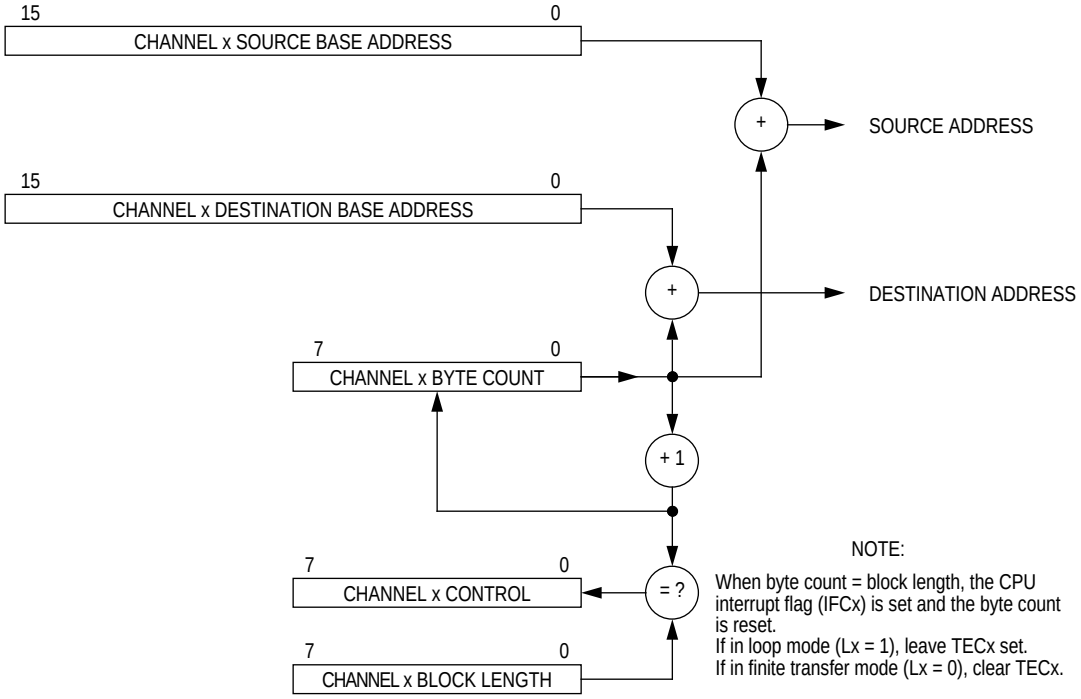


Figure 9-5. Incremented Source and Incremented Destination

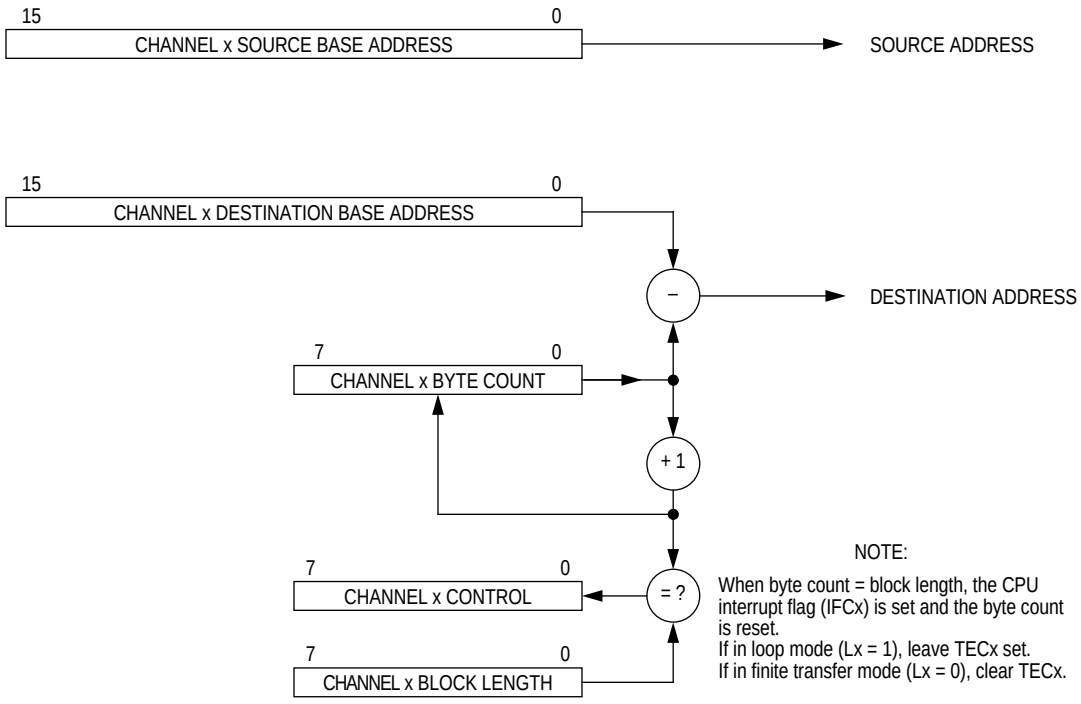


Figure 9-6. Static Source and Decremented Destination

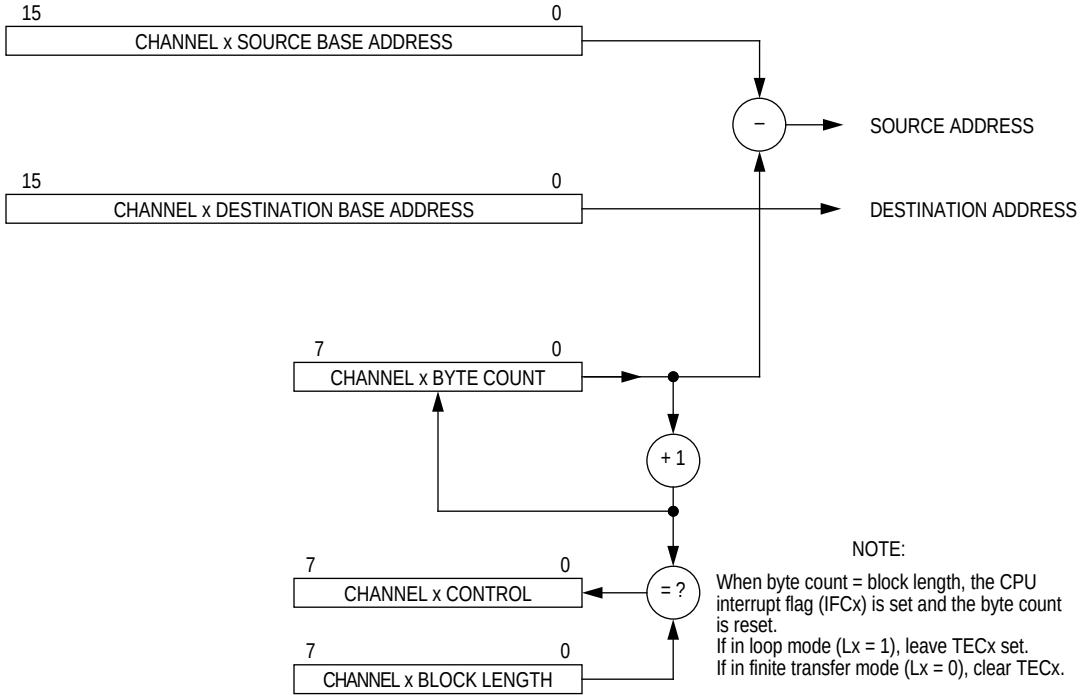


Figure 9-7. Decremented Source and Static Destination

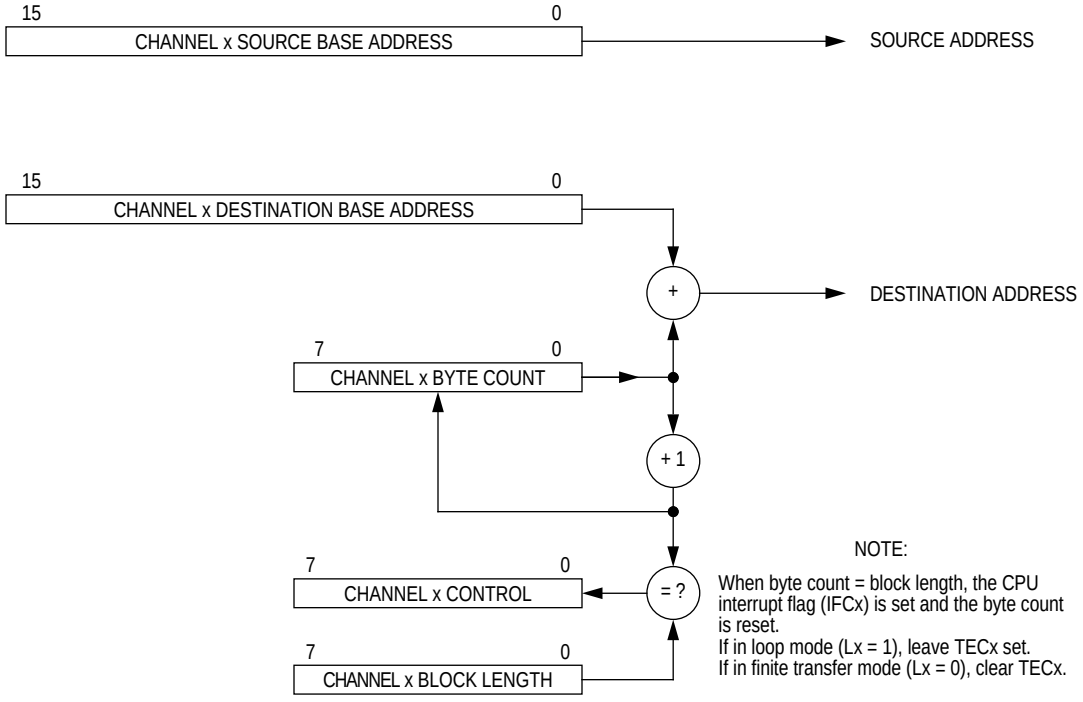


Figure 9-8. Static Source and Incremented Destination

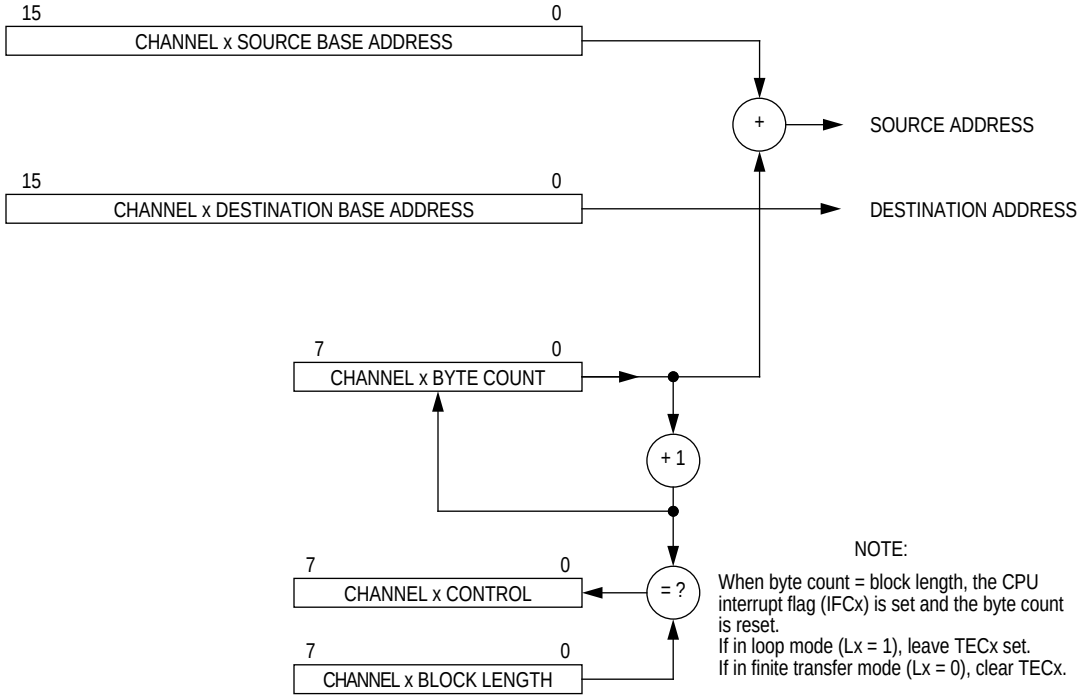


Figure 9-9. Incremented Source and Static Destination

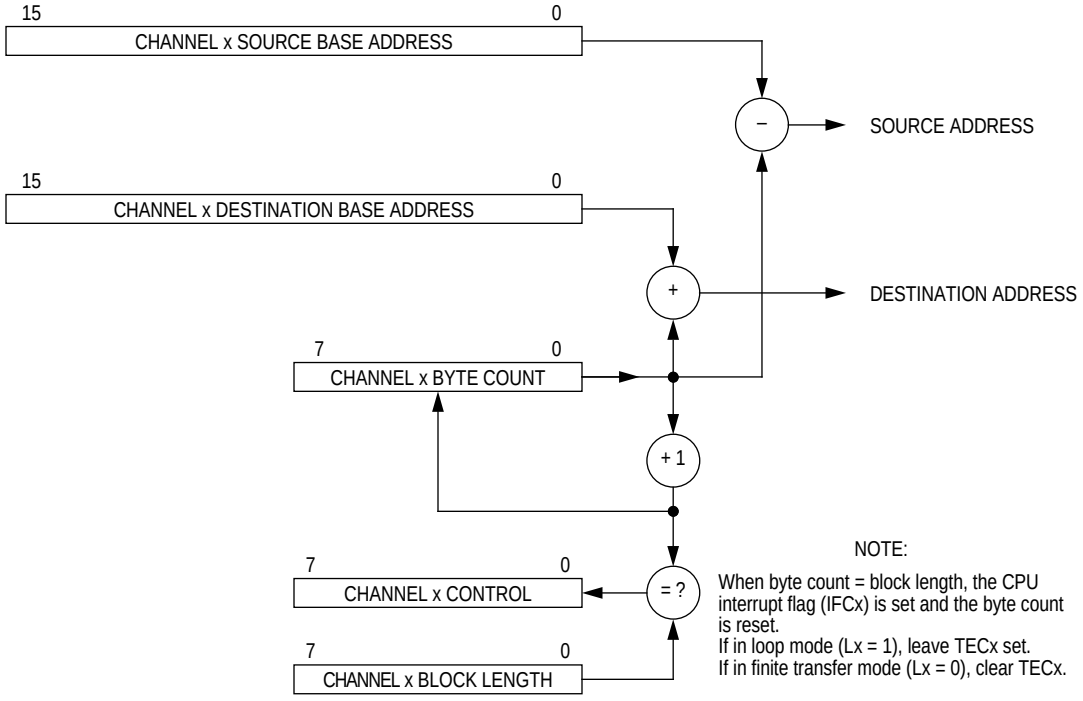


Figure 9-10. Decremented Source and Incremented Destination

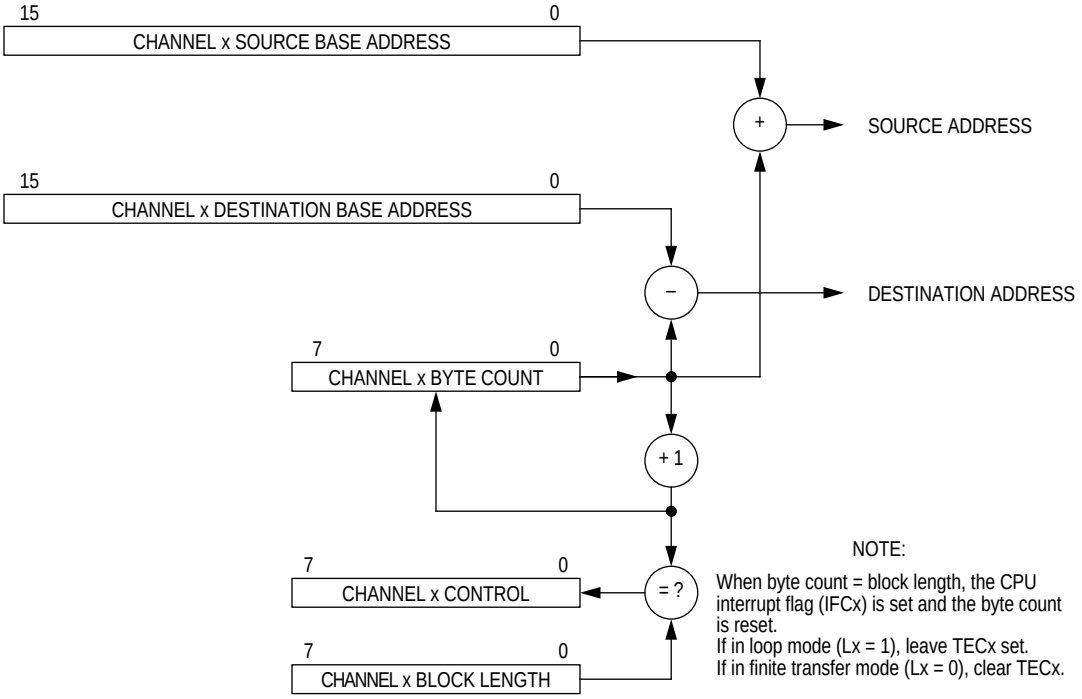


Figure 9-11. Incremented Source and Decremented Destination

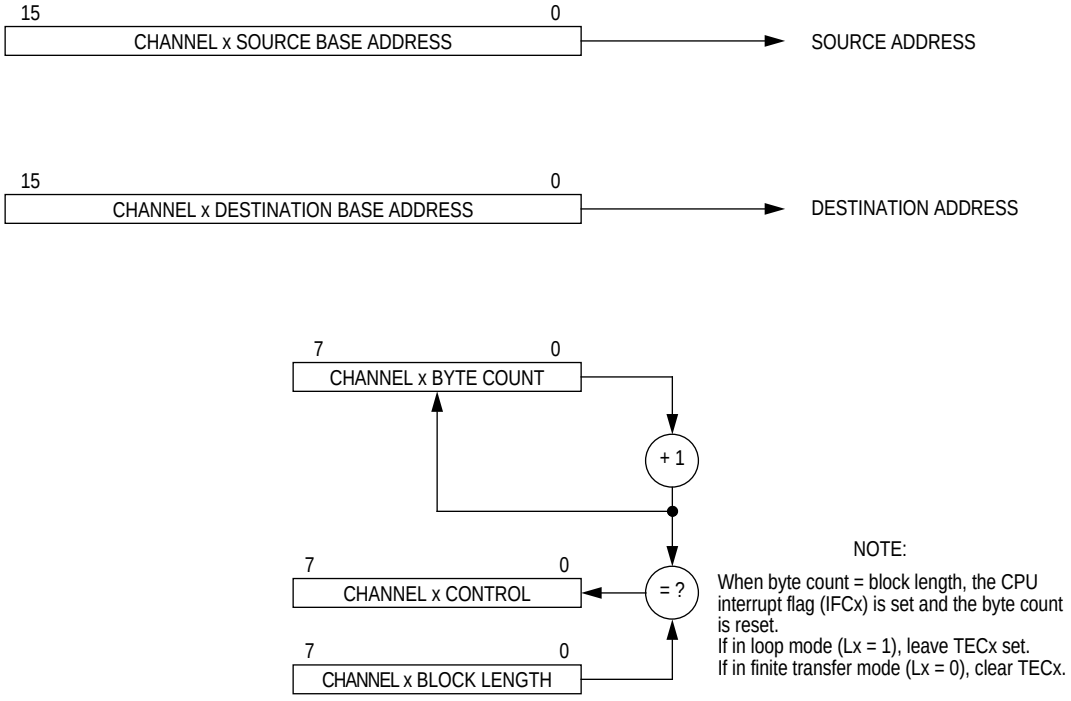


Figure 9-12. Static Source and Static Destination

9.4 Low-Power Modes

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

9.4.1 Wait Mode

If the DMA wait enable bit (DMAWE) is set, the DMA responds to DMA service requests during wait mode. The DMA can transfer data to and from peripherals while the MCU remains in the low-power wait mode.

If the WAIT instruction occurs during a DMA transfer while DMAWE is set, the DMA transfer continues to completion. If the DMAWE bit is clear, a WAIT instruction suspends the current DMA transfer. If the DMA priority bit (DMAP) is set, the transfer resumes when the MCU exits wait mode.

9.4.2 Stop Mode

The DMA is inactive during stop mode. A STOP instruction suspends any DMA transfer in progress. If an external interrupt brings the MCU out of stop mode and the DMA priority bit (DMAP) is set, the suspended DMA transfer resumes. If a reset brings the MCU out of stop mode, the transfer is aborted.

9.5 DMA During Break Interrupts

During a break interrupt, the DMA is inactive.

If a DMA-generated address matches the contents of the break address registers, a break interrupt begins at the end of the current CPU instruction.

If a break interrupt is asserted during the current address cycle and the DMA is active, the DMA releases the internal address and data buses at the next address boundary to preserve the current MCU state. During the break interrupt, the DMA continues to arbitrate DMA channel priorities. After the break interrupt, the DMA becomes active again and resumes transferring data according to its highest priority service request.

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 **7.7.3 SIM Break Flag Control Register (SBFCR)**.)

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

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

9.6 I/O Registers

The following registers control and monitor operation of the DMA:

- DMA control register 1 (DC1)
- DMA status and control register (DSC)
- DMA control register 2 (DC2)

DC1, DSC, and DC2 can be written during a DMA transfer without affecting DMA latency.

The following registers control operation of each of the DMA channels:

- DMA source address registers, high and low (D0SH:D0SL, D1SH:D1SL, and D2SH:D2SL)
- DMA destination address registers, high and low (D0DH:D0DL, D1DH:D1DL, and D2DH:D2DL)
- DMA channel x control registers (D0C–D2C)
- DMA channel x byte count registers (D0BC–D2BC)
- DMA channel x block length registers (D0BL–D2BL)

Writing to DxSH:DxSL, DxDH:DxDL, DxC, and DxBL during a transfer affects DMA latency. A write to a channel x control register during a transfer has a two-bus cycle latency if the transfer is first suspended by disabling the channel. Disable the channel by writing a zero to the TECx bit in DMA control register 1. Without first suspending the transfer, a write to a channel x control register during a transfer has a three-bus cycle latency.

9.6.1 DMA Control Register 1 (DC1)

DMA control register 1 does the following:

- Enables channels to transfer data when a DMA service request occurs
- Enables channels to generate CPU interrupt requests
- Controls how much of the bus bandwidth the DMA can use

		Bit 7	6	5	4	3	2	1	Bit 0
DC1 $004C	Read: Write:	BB1	BB0	TEC2	IEC2	TEC1	IEC1	TEC0	IEC0
	Reset:	0	0	0	0	0	0	0	0

Figure 9-13. DMA Control Register 1 (DC1)

BB1 and BB0 — Bus Bandwidth Control Bits

These read/write bits control the ratio of DMA/CPU bus activity during a DMA transfer. As Table 9-4 shows, the DMA can use 25%, 50%, 67%, or 100% of the bus bandwidth. Reset clears bits BB1 and BB0.

Table 9-4. DMA/CPU Bus Control Selection

	DMA Transfer	
BB1:BB0	**DMA Bus Cycles**	**CPU Bus Cycles**
00	2 (25%)	6 (75%)
01	2 (50%)	2 (50%)
10	2 (67%)	1 (33%)
11	All (100%)	0 (0%)

Figure 9-14, Figure 9-15, and Figure 9-16 show the timing of DMA transfers with DMA bus bandwidths of 25%, 50%, and 67%.

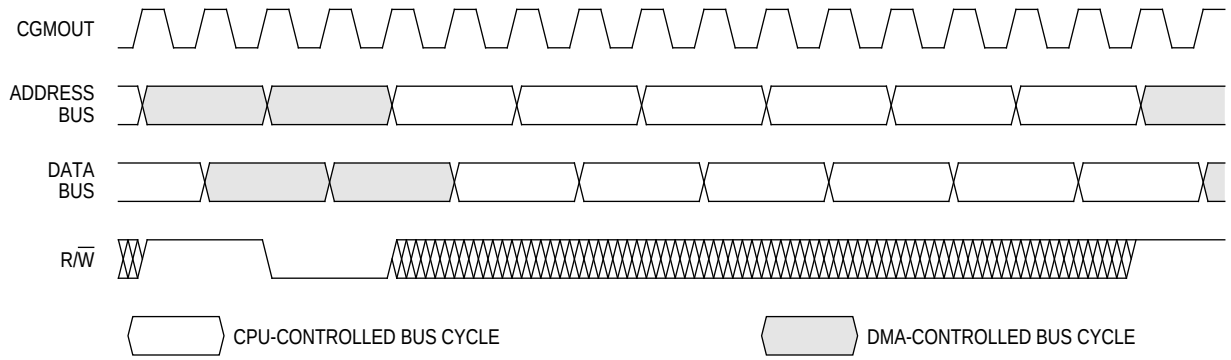


Figure 9-14. Multiple Byte/Word Transfer Timing: 25% DMA Bus Bandwidth

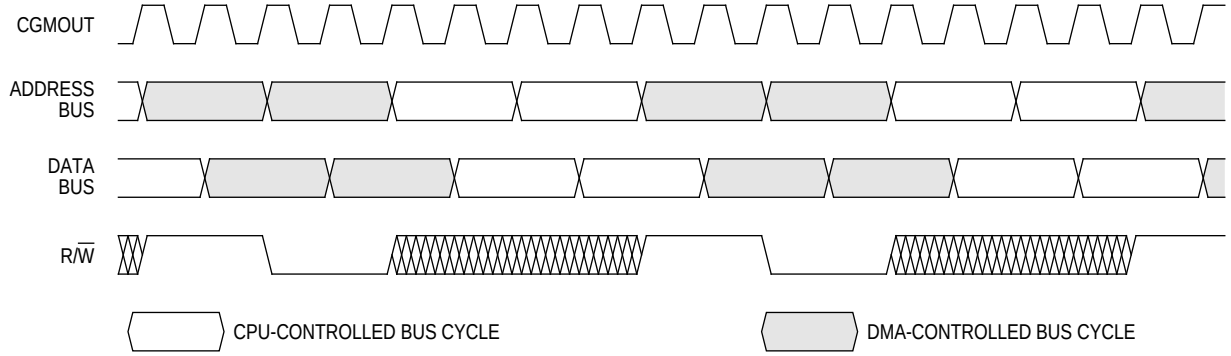


Figure 9-15. Multiple Byte/Word Transfer Timing: 50% DMA Bus Bandwidth

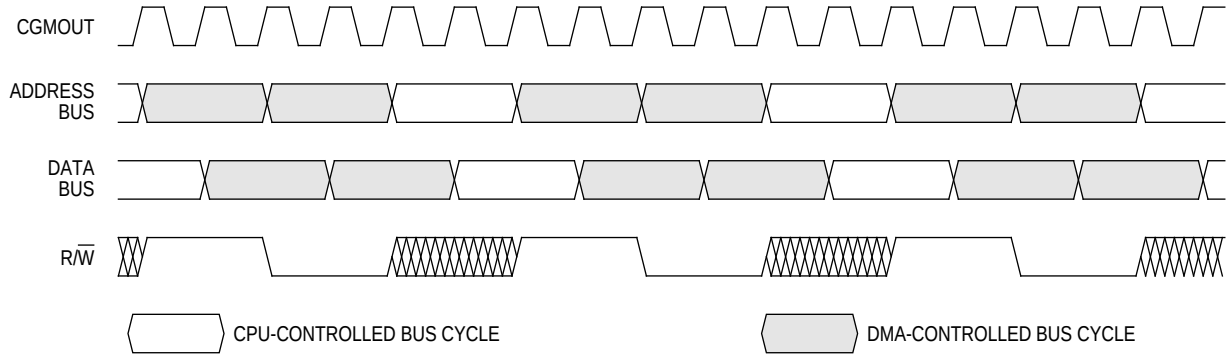


Figure 9-16. Multiple Byte/Word Transfer Timing: 67% DMA Bus Bandwidth

---

NOTE

When two or more DMA channels have transfers pending, the CPU executes at least one cycle between each DMA block length, even if the DMA channels have 100% of the bus bandwidth.

For DMA transfers of one byte or one word, giving the DMA 100% of the bus bandwidth is appropriate. However, for large, software-initiated transfers, limiting the bus bandwidth of the DMA may be useful to keep from slowing CPU activity.

TEC[2:0] — Transfer Enable Bits

These read/write bits enable the corresponding channels to perform transfers when DMA service requests occur. When two or more channels are enabled, a transfer on one channel cannot begin while another channel is transferring a byte or word. Reset clears the TEC[2:0] bits.

1 = Corresponding DMA channel enabled
0 = Corresponding DMA channel disabled

IEC[2:0] — CPU Interrupt Enable Bits

These read/write bits enable the corresponding channels to generate CPU interrupt requests upon completion of DMA block transfers or at the restart of DMA transfer loops. Reset clears the IEC[2:0] bits.

1 = CPU interrupts from corresponding channel enabled
0 = CPU interrupts from corresponding channel disabled

9.6.2 DMA Status and Control Register (DSC)

The DMA status and control register does the following:

- Flags completion of DMA transfers
- Controls looping of source and destination address counts
- Controls priority of DMA service requests and CPU interrupt requests

		Bit 7	6	5	4	3	2	1	Bit 0
DSC $004D	Read: / Write:	DMAP	L2	L1	L0	DMAWE	IFC2	IFC1	IFC0
	Reset:	0	0	0	0	0	0	0	0

Figure 9-17. DMA Status and Control Register (DSC)

DMAP — DMA Priority Bit

This read/write bit controls the priority of CPU interrupt requests during DMA transfers. Reset clears the DMAP bit.

1 = CPU interrupt requests inhibited during DMA transfers— When DMAP is set, a CPU interrupt request is not recognized until the end of the current DMA transfer. During a block transfer, the increase in CPU interrupt latency depends on the size of the block and the bus bandwidth bits, BB1:0].

0 = CPU interrupt requests recognized during DMA transfers — When DMAP is clear, a CPU interrupt request is recognized after the transfer of the current byte or word in the current DMA transfer. The CPU interrupt disables the DMA by clearing the transfer enable bits, TEC[2:0]. (See **9.6.1 DMA Control Register 1 (DC1)**.) Therefore, the DMA can increase CPU interrupt latency by up to three cycles in a byte transfer or five cycles in a word transfer.

NOTE

When DMAP = 0, a CPU interrupt clears the TECx bit if the channel has a pending DMA transfer. Software must re-enable channel x after each CPU interrupt by setting the TECx bit.

Table 9-5 shows the effect of the DMAP bit when the DMA has 100% of the bus bandwidth (BB[1:0] = 1:1).

Table 9-5. DMA Transfer/CPU Interrupt Request Priority Selection

	DMAP = 0	DMAP = 1
Highest Priority	CPU Interrupt Requests	DMA Channel 0 Transfer
	DMA Channel 0 Transfer	DMA Channel 1 Transfer
	DMA Channel 1 Transfer	DMA Channel 2 Transfer
Lowest Priority	DMA Channel 2 Transfer	CPU Interrupt Requests

L[2:0] — Loop Enable Bits

These read/write bits enable looping of the DMA back to the base addresses in the source address and destination address registers during block transfers. Reset clears the L[2:0] bits.

1 = Looping enabled — After transferring the number of bytes equal to the number programmed in the DMA block length register, the DMA does the following:

- Sets the CPU interrupt flag (IFCx) for that channel
- Generates a CPU interrupt request if enabled (IECx = 1)
- Clears the byte count register
- Continues the transfer from the base address

0 = Looping disabled — After transferring the number of bytes equal to the number programmed in the DMA block length register, the DMA does the following:

- Sets the CPU interrupt flag (IFCx) for that channel
- Generates a CPU interrupt request if enabled (IECx = 1)
- Clears the byte count register
- Disables the channel by clearing the TECx bit

NOTE

The CPU executes a minimum of one cycle before the next DMA loop begins, even if the DMA has 100% of the bus bandwidth.

DMAWE — DMA Wait Enable Bit

This read/write bit enables the DMA to operate while in wait mode. Reset clears the DMAWE bit.

1 = DMA transfer enabled after WAIT instruction
0 = DMA transfer suspended after WAIT instruction

IFC[2:0] — CPU Interrupt Flag Bits

These read/write bits become set when a DMA transfer is complete or at the end of each transfer loop. IFC2, IFC1, or IFC0 can generate a CPU interrupt request if the corresponding IECx bit is set in DMA control register 1. Clear IFC[2:0] by reading them and then writing zeros to them. Reset clears the IFC[2:0] bits.

1 = DMA transfer complete
0 = DMA transfer not complete

9.6.3 DMA Control Register 2 (DC2)

DMA control register 2 can perform two functions:

- Initiate DMA transfers through software
- Simulate DMA service requests for test purposes

		Bit 7	6	5	4	3	2	1	Bit 0
DC2 $004E	Read: Write:	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
	Reset:	0	0	0	0	0	0	0	0

Figure 9-18. DMA Control Register 2 (DC2)

SWI[7:0] — Software Initiate Bits

Each of these read/write bits corresponds to one of the eight DMA transfer sources. (See **Table 9-8. DMA Transfer Source Selection**.) Setting an SWIx bit can initiate a DMA service request from the selected transfer source.

1 = DMA software transfer initiated
0 = DMA software transfer halted

Use the following steps to perform a software-initiated DMA service request:

1. Enable a channel to perform a transfer by setting its TECx bit. (See **9.6.1 DMA Control Register 1 (DC1)**.)
2. Assign the channel to a DMA transfer source by writing a binary value from 000 to 111 to its DTS[2:0] bits. (See **9.6.4 DMA Channel Control Registers (D0C–D2C)**.)
3. Set the SWIx bit that corresponds to the selected transfer source. The bit positions (0–7) of the SWIx bits correspond to the binary values (000–111) that select the DMA transfer source. For example, after selecting transfer source 100 (binary), set bit SWI4 to initiate the DMA service request.

9.6.4 DMA Channel Control Registers (D0C–D2C)

Each DMA channel control register does the following:

- Controls calculation of source and destination addresses
- Selects transfer of 8-bit bytes or 16-bit words on the channel
- Assigns the channel to one of eight DMA transfer sources

The state of the DMA channel control registers after reset is indeterminate.

		Bit 7	6	5	4	3	2	1	Bit 0
D0C $0038	Read:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
	Write:								
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
D1C $0040	Read:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
	Write:								
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
D2C $0048	Read:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
	Write:								
	Reset:	Indeterminate after reset							

Figure 9-19. DMA Channel Control Registers (D0C–D2C)

SDC[3:0] — Source/Destination Address Control Bits

These read/write bits control calculation of the source and destination addresses as shown in Table 9-6.

Table 9-6. Source/Destination Address Register Control

SDC[3:0]	Source Address	Destination Address
1010	Increment	Increment
1001	Increment	Decrement
1000	Increment	Static
0110	Decrement	Increment
0101	Decrement	Decrement
0100	Decrement	Static
0010	Static	Increment
0001	Static	Decrement
0000	Static	Static

The DMA calculates an incremented address by adding the byte count register to the base address. To calculate a decremented address, the DMA subtracts the byte count register from the base address. To determine a static address, the DMA reads the base address.

BWC — Byte/Word Control Bit

This read/write bit determines whether the DMA channel transfers 8-bit bytes or 16-bit words. The BWC bit has no effect unless either the source or destination address is static or both are static. Table 9-7 shows how the DMA ALU calculates addresses in word transfers.

1 = 16-bit words
0 = 8-bit bytes

NOTE

To transfer a block of 16-bit words (BWC = 1), set the block length to the number of words times 2. (See **9.6.7 DMA Block Length Registers (D0BL–D2BL)**.)

When both the source and destination addresses are static, the first byte of the word transfers from the source base address to the destination base address. The second byte transfers from the source base address plus one to the destination address plus one. When either the source or destination address increments or decrements, the DMA transfers bytes from or to incrementing or decrementing addresses.

The CPU interrupt flag (IFCx) becomes set when the byte count register equals the block length register.

Table 9-7. DMA Word Transfer

		Static Source	Static Destination	Incremented Source	Static Destination	Static Source	Incremented Destination	Decremented Source	Static Destination	Static Source	Decremented Destination
Word	**Byte**	**From**	**To**	**From**	**To**	**From**	**To**	**From**	**To**	**From**	**To**
1	1	SBA(1)	DBA(2)	SBA	DBA	SBA	DBA	SBA	DBA	SBA	DBA
	2	SBA + 1	DBA + 1	SBA + 1	DBA + 1	SBA + 1	DBA + 1	SBA – 1	DBA + 1	SBA + 1	DBA – 1
2	3	SBA	DBA	SBA + 2	DBA	SBA	DBA + 2	SBA – 2	DBA	SBA	DBA – 2
	4	SBA + 1	DBA + 1	SBA + 3	DBA + 1	SBA + 1	DBA + 3	SBA – 3	DBA + 1	SBA + 1	DBA – 3
3	5	SBA	DBA	SBA + 4	DBA	SBA	DBA + 4	SBA – 4	DBA	SBA	DBA – 4
	6	SBA + 1	DBA + 1	SBA + 5	DBA + 1	SBA + 1	DBA + 5	SBA – 5	DBA + 1	SBA + 1	DBA – 5
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
n	2n – 1	SBA	DBA	SBA + 2n – 2	DBA	SBA	DBA + 2n – 2	SBA – (2n – 2)	DBA	SBA	DBA – (2n – 2)
	2n	SBA + 1	DBA + 1	SBA + 2n – 1	DBA +1	SBA + 1	DBA + 2n – 1	SBA – (2n – 1)	DBA + 1	SBA + 1	DBA – (2n – 1)

1. SBA = Source base address
2. DBA = Destination base address

DTS[2:0] — DMA Transfer Source Bits

These read/write bits assign the DMA channel to one of the eight transfer source inputs as shown in Table 9-8.

Table 9-8. DMA Transfer Source Selection

Transfer Source	DTS2:DTS1:DTS0
TIM Channel 0 Interrupt Request	000
TIM Channel 1 Interrupt Request	001
TIM Channel 2 Interrupt Request	010
TIM Channel 3 Interrupt Request	011
SPI Receive Interrupt Request	100
SPI Transmit Interrupt Request	101
SCI Receive Interrupt Request	110
SCI Transmit Interrupt Request	111

9.6.5 DMA Source Address Registers (D0SH/L–D2SH/L)

Each DMA channel takes its data from a source base address contained in a 16-bit source address register. During a block transfer, the DMA determines successive source addresses by adding to (to increment) or subtracting from (to decrement) the base address. In static address transfers, the DMA finds the source address by merely reading the source address registers. Figure 9-20 shows the DMA source address registers. The state of the source address registers after reset is indeterminate.

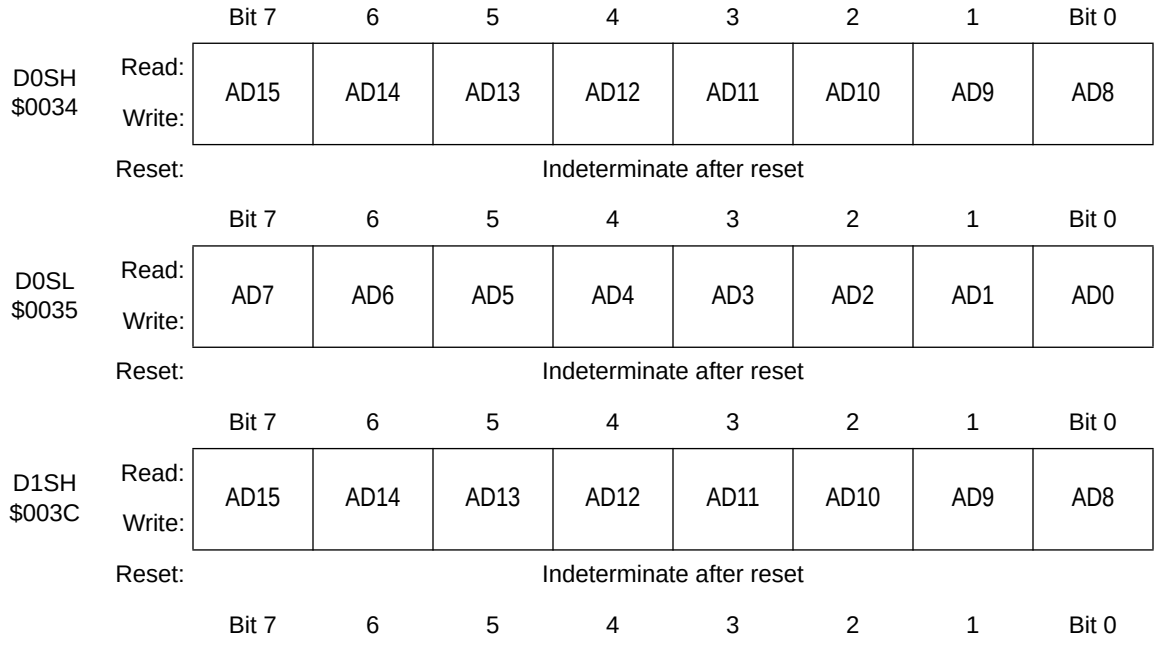


Figure 9-20. DMA Source Address Registers (D0SH/L–D2SH/L)

D1SL $003D	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Write:								
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
D2SH $0044	Read:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
	Write:								
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
D2SL $0045	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Write:								
	Reset:	Indeterminate after reset							

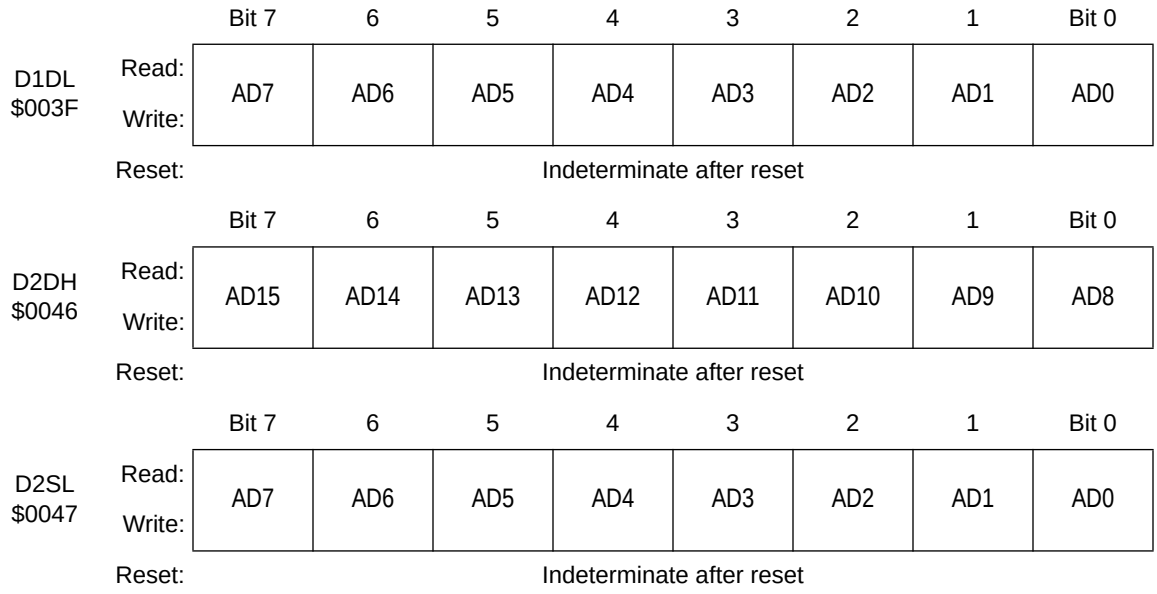
Figure 9-20. DMA Source Address Registers (D0SH/L–D2SH/L) (Continued)

9.6.6 DMA Destination Address Registers (D0DH/L–D2DH/L)

Each DMA channel transfers data to the destination base address contained in a 16-bit destination address register. During a block transfer, the DMA determines successive destination addresses by adding to (to increment) or subtracting from (to decrement) the base address. In static address transfers, the DMA finds the destination address by merely reading the destination address registers. Figure 9-21 shows the DMA destination address registers. The state of the destination address registers after reset is indeterminate.

		Bit 7	6	5	4	3	2	1	Bit 0
D0DH $0036	Read:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
	Write:								
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
D0DL $0037	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Write:								
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
D1DH $003E	Read:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
	Write:								
	Reset:	Indeterminate after reset							

Figure 9-21. DMA Destination Address Registers (D0DH/L–D2DH/L)



		Bit 7	6	5	4	3	2	1	Bit 0
D1DL $003F	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
D2DH $0046	Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
D2SL $0047	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Reset:	Indeterminate after reset							

Figure 9-21. DMA Destination Address Registers (D0DH/L–D2DH/L)

9.6.7 DMA Block Length Registers (D0BL–D2BL)

The read/write block length registers control the number of bytes transferred. During a block transfer, the DMA compares the number programmed into the channel's DMA block length register to the number in its DMA byte count register. When the byte count reaches the value in the block length register, the DMA does the following:

- Sets the CPU interrupt flag (IFCx) for that channel in the DMA status and control register
- Generates a CPU interrupt request if enabled
- Resets the byte count register

If looping is disabled (Lx = 0), the DMA then stops the transfer by clearing the TECx bit in DMA control register 1, disabling the channel. If looping is enabled (Lx = 1), the DMA continues the transfer from the base address.

The block length of a word transfer is twice the number of words.

The state of the DMA block length registers after reset is indeterminate.

		Bit 7	6	5	4	3	2	1	Bit 0
D0BL $0039	Read: Write:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
D1BL $0041	Read: Write:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
	Reset:	Indeterminate after reset							
		Bit 7	6	5	4	3	2	1	Bit 0
D2BL $0049	Read: Write:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
	Reset:	Indeterminate after reset							

Figure 9-22. DMA Block Length Registers (D0BL–D2BL)

9.6.8 DMA Byte Count Registers (D0BC–D2BC)

Each read/write DMA byte count register contains the number of bytes transferred on that channel in the current DMA transfer.

		Bit 7	6	5	4	3	2	1	Bit 0
D0BC $003B	Read: Write:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
	Reset:	0	0	0	0	0	0	0	0
		Bit 7	6	5	4	3	2	1	Bit 0
D1BC $0043	Read: Write:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
	Reset:	0	0	0	0	0	0	0	0
		Bit 7	6	5	4	3	2	1	Bit 0
D2BC $004B	Read: Write:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
	Reset:	0	0	0	0	0	0	0	0

Figure 9-23. DMA Byte Count Registers (D0BC–D2BC)

Writing to the channel x source address or destination address register clears the channel x byte count register. The channel x byte count register also is cleared when its count reaches the value in the channel x block length register. Reset clears the byte count registers.

SECTION 10
BREAK MODULE

10.1 Introduction

This section describes the break module (Break, Version B). The break module can generate a break interrupt that stops normal program flow at a defined address to enter a background program.

10.2 Features

Features of the break module include the following:

- Accessible I/O Registers during the Break Interrupt
- CPU-Generated and DMA-Generated Break Interrupts
- Software-Generated Break Interrupts
- COP Disabling during Break Interrupts

10.3 Functional Description

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

The following events can cause a break interrupt to occur:

- A CPU-generated address (the address in the program counter) matches the contents of the break address registers.
- During a DMA transfer, a DMA-generated address matches the contents of the break address registers.
- Software writes a logic one to the BRKA bit in the break status and control register.

When a CPU- or DMA-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 10-1 shows the structure of the break module.

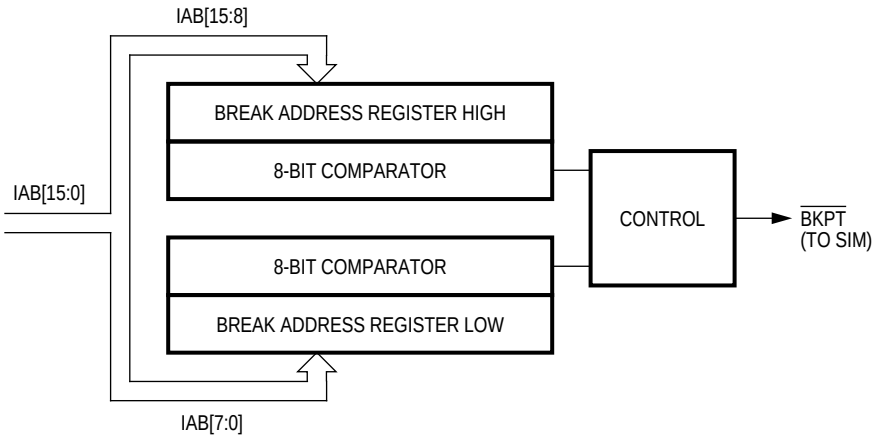


Figure 10-1. Break Module Block Diagram

10

Table 10-1. Break I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Break Address Register High (BRKH)	Bit 15	14	13	12	11	10	9	Bit 8	$FE0C
Break Address Register Low (BRKL)	Bit 7	6	5	4	3	2	1	Bit 0	$FE0D
Break Status/Control Register (BRKSCR)	BRKE	BRKA							$FE0E

= Unimplemented

10.3.1 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 **7.7.3 SIM Break Flag Control Register (SBFCR)** and see the **Break Interrupts** subsection for each module.)

10.3.2 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.

10.3.3 DMA During Break Interrupts

During a break interrupt, the DMA is inactive.

If a DMA-generated address matches the contents of the break address registers, a break interrupt begins at the end of the current CPU instruction.

If a break interrupt is asserted during the current address cycle and the DMA is active, the DMA releases the internal address and data buses at the next address boundary to preserve the current MCU state. During the break interrupt, the DMA continues to arbitrate DMA channel priorities. After the break interrupt, the DMA becomes active again and resumes transferring data according to its highest priority service request.

10.3.4 TIM During Break Interrupts

A break interrupt stops the timer counter.

10.3.5 COP During Break Interrupts

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

10.4 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)

10.4.1 Break Status and Control Register (BRKSCR)

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

		Bit 7	6	5	4	3	2	1	Bit 0
BRKSCR $FE0E	Read:	BRKE	BRKA	0	0	0	0	0	0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 10-2. Break Status and Control Register (BRKSCR)

10

BRKE — Break Enable Bit

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

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

BRKA — Break Active Bit

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

1 = Break address match
0 = No break address match

10.4.2 Break Address Registers (BRKH and BRKL)

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

		Bit 7	6	5	4	3	2	1	Bit 0
BRKH $FE0C	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	0	0	0	0	0	0	0	0
		Bit 7	6	5	4	3	2	1	Bit 0
BRKL $FE0D	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	0	0	0	0	0	0	0	0

Figure 10-3. Break Address Registers (BRKH and BRKL)

10.5 Low-Power Modes

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

10

10.5.1 Wait Mode

If enabled, the break module and the DMA module are active in wait mode. The SIM break stop/wait bit (SBSW) in the SIM break status register (see **7.7 SIM Registers**) becomes set if a DMA-generated address matches the break address registers in wait mode. The DMA can also use the break status and control register as its destination address in order to write to the BRKA and BRKE bits during wait mode. The SBSW bit is set if the DMA writes to the break status and control register. SBSW is for applications that require a return to wait mode after exiting wait mode for a DMA-generated break interrupt. In the break routine, the user can subtract one from the return address on the stack if SBSW is set. Clear the SBSW bit by writing logic zero to it.

10.5.2 Stop Mode

A break interrupt causes exit from stop mode and sets the SBSW bit in the SIM break status register. (See **7.7 SIM Registers**.)

SECTION 11
MONITOR ROM (MON)

11.1 Introduction

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

11.2 Features

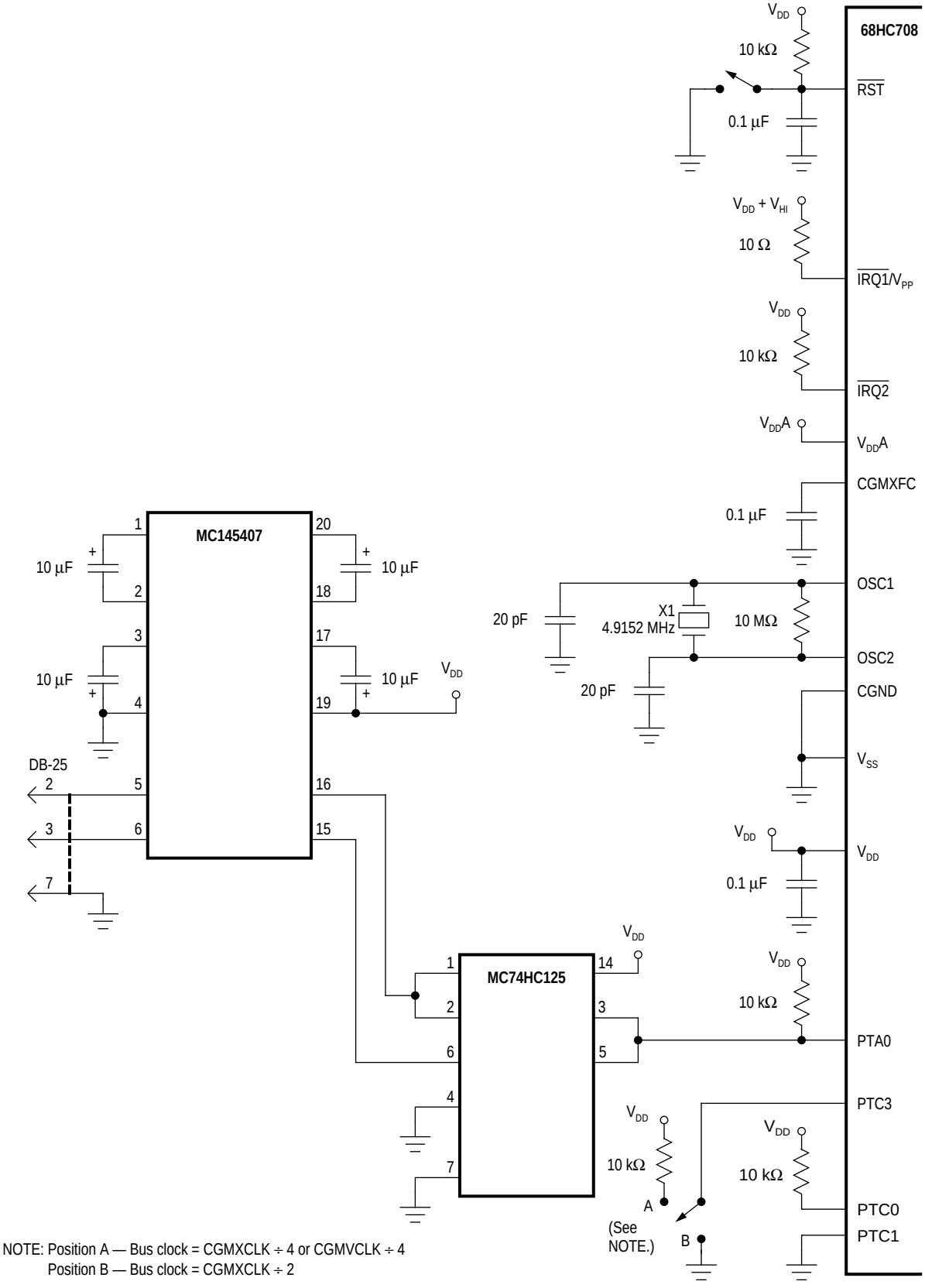
Features of the monitor ROM include the following:

- Normal User-Mode Pin Functionality
- One Pin Dedicated to Serial Communication between Monitor ROM and Host Computer
- Standard Mark/Space Non-Return-to-Zero (NRZ) Communication with Host Computer
- 4800 Baud–28.8 kBaud Communication with Host Computer
- Execution of Code in RAM or ROM
- (E)EPROM/OTPROM Programming

11.3 Functional Description

The monitor ROM receives and executes commands from a host computer. Figure 11-1 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 pull-up resistor.



NOTE: Position A — Bus clock = CGMXCLK ÷ 4 or CGMVCLK ÷ 4
Position B — Bus clock = CGMXCLK ÷ 2

Figure 11-1. Monitor Mode Circuit

NOTE

If the security bit (SEC) in the mask option register (MOR) is programmed to logic one, reading user OTPROM/EPROM is disabled in monitor mode.[1]

11.3.1 Entering Monitor Mode

Table 11-1 shows the pin conditions for entering monitor mode.

Table 11-1. Mode Selection

$\overline{IRQ1}/V_{PP}$ 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 zero and then a logic one to the $\overline{RST}$ pin.

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

Monitor mode uses alternate vectors for reset, SWI, and break interrupt. The alternate vectors are in the $FE page instead of the $FF page and allow code execution from the internal monitor firmware instead of user code. The COP module is disabled in monitor mode as long as $V_{DD} + V_{HI}$ is applied to either the $\overline{IRQ1}/V_{PP}$ pin or the $\overline{RST}$ pin. (See **SECTION 7 SYSTEM INTEGRATION MODULE (SIM)** for more information on modes of operation.)

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

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% duty cycle at maximum bus frequency.

Table 11-2 is a summary of the differences between user mode and monitor mode.

Table 11-2. 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

1. If the high voltage (V_{DD} + V_{HI}) is removed from the $\overline{IRQ1}/V_{PP}$ pin or the $\overline{RST}$ 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.

11

11.3.2 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. (See Figure 11-2 and Figure 11-3.)

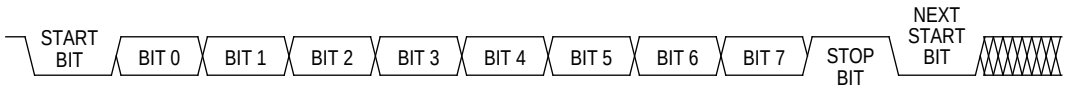


Figure 11-2. Monitor Data Format

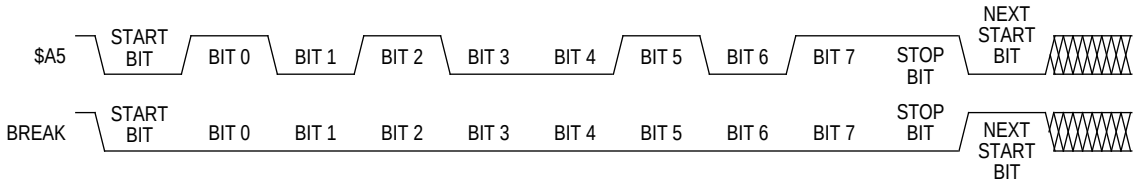


Figure 11-3. Sample Monitor Waveforms

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

11.3.3 Echoing

As shown in Figure 11-4, the monitor ROM immediately echoes each received byte back to the PTA0 pin for error checking.

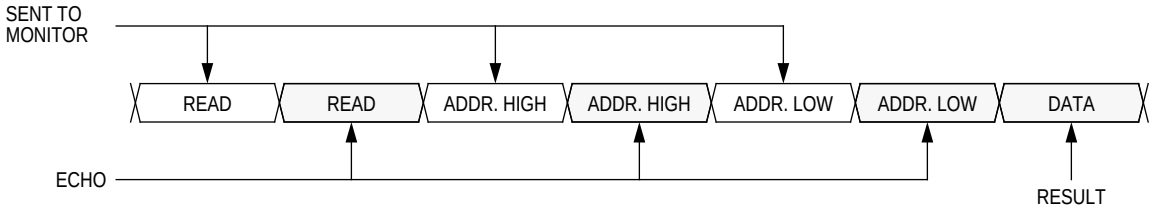


Figure 11-4. Read Transaction

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

11.3.4 Break Signal

A start bit followed by nine low bits is a break signal. (See Figure 11-5.) When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits before echoing the break signal.

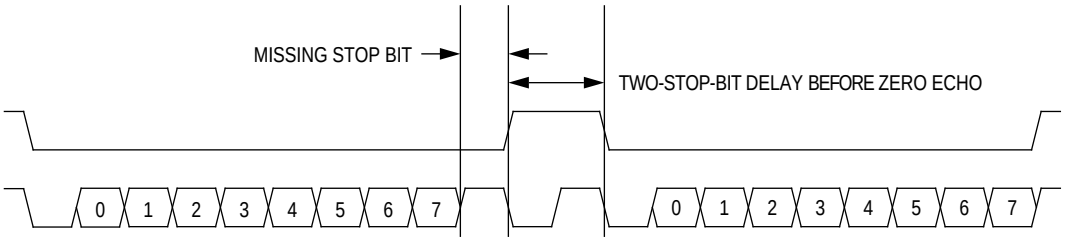


Figure 11-5. Break Transaction

11.3.4.1 Commands

The monitor ROM uses the following commands:

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

Table 11-3. 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	

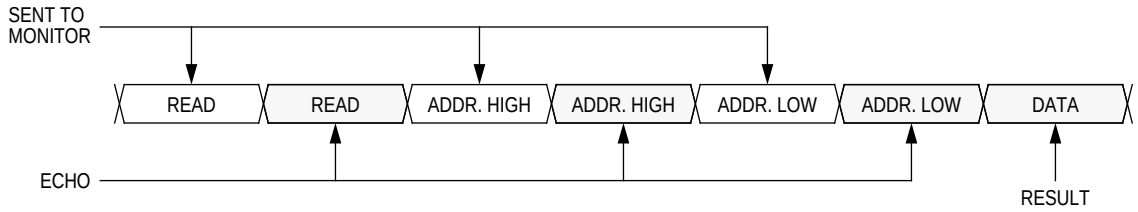


Table 11-4. WRITE (Write Memory) Command

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	

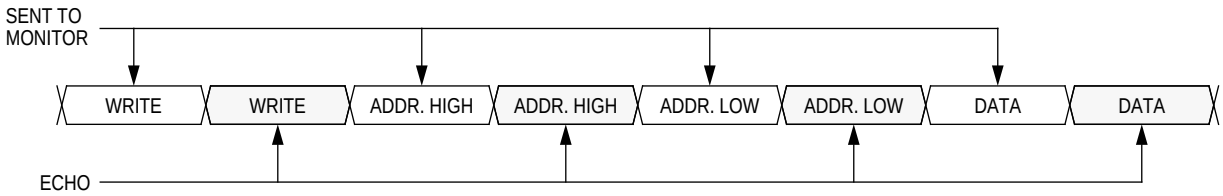


Table 11-5. IREAD (Indexed Read) Command

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

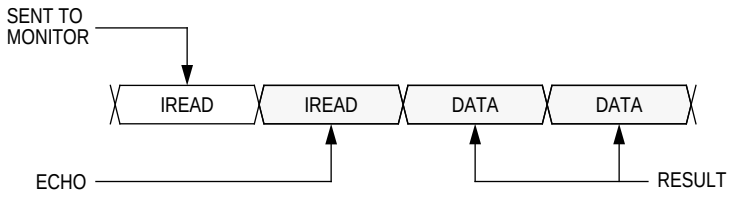
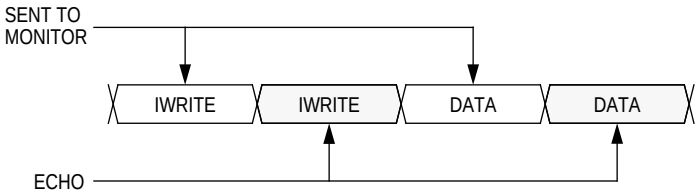


Table 11-6. IWRITE (Indexed Write) Command

Description	Write to last address accessed + 1
Operand	Specifies single data byte
Data Returned	None
Opcode	$19
Command Sequence	



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

Table 11-7. 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	

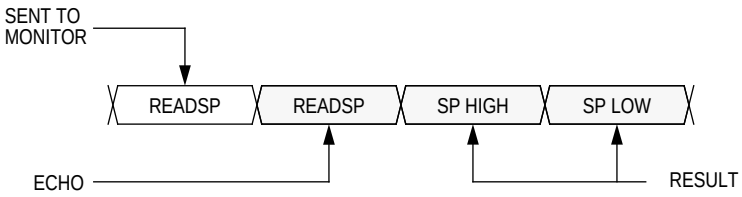
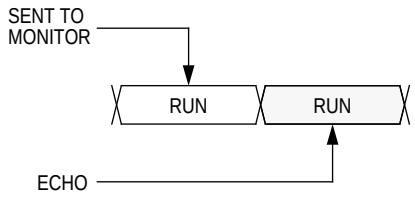


Table 11-8. RUN (Run User Program) Command

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

11.3.5 Baud Rate

With a 4.9152-MHz crystal and the PTC3 pin at logic one during reset, data is transferred between the monitor and host at 4800 baud. If the PTC3 pin is at logic zero during reset, the monitor baud rate is 9600. When the CGM output, CGMOUT, is driven by the PLL, the baud rate is determined by the MUL[7:4] bits in the PLL programming register (PPG). (See **SECTION 8 CLOCK GENERATOR MODULE (CGM)**.)

11

Table 11-9. 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 12
TIMER INTERFACE MODULE (TIM)

12.1 Introduction

This section describes the timer interface module (TIM4, Version B). The TIM is a four-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. Figure 12-1 is a block diagram of the TIM.

12.2 Features

Features of the TIM include the following:

- Four Input Capture/Output Compare Channels
  - Rising-Edge, Falling-Edge, or Any-Edge Input Capture Trigger
  - Set, Clear, or Toggle Output Compare Action
- Buffered and Unbuffered Pulse Width Modulation (PWM) Signal Generation
- Programmable TIM Clock Input
  - Seven-Frequency Internal Bus Clock Prescaler Selection
  - External TIM Clock Input (4-MHz Maximum Frequency)
- Free-Running or Modulo Up-Count Operation
- Toggle Any Channel Pin on Overflow
- TIM Counter Stop and Reset Bits
- DMA Service Request Generation
- Modular Architecture Expandable to 8 Channels

12.3 Functional Description

Figure 12-1 shows the structure of the TIM. The central component of the TIM is the 16-bit TIM counter that can operate as a free-running counter or a modulo up-counter. The TIM counter provides the timing reference for the input capture and output compare functions. The TIM counter modulo registers, TMODH:TMODL, control the modulo value of the TIM counter. Software can read the TIM counter value at any time without affecting the counting sequence.

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

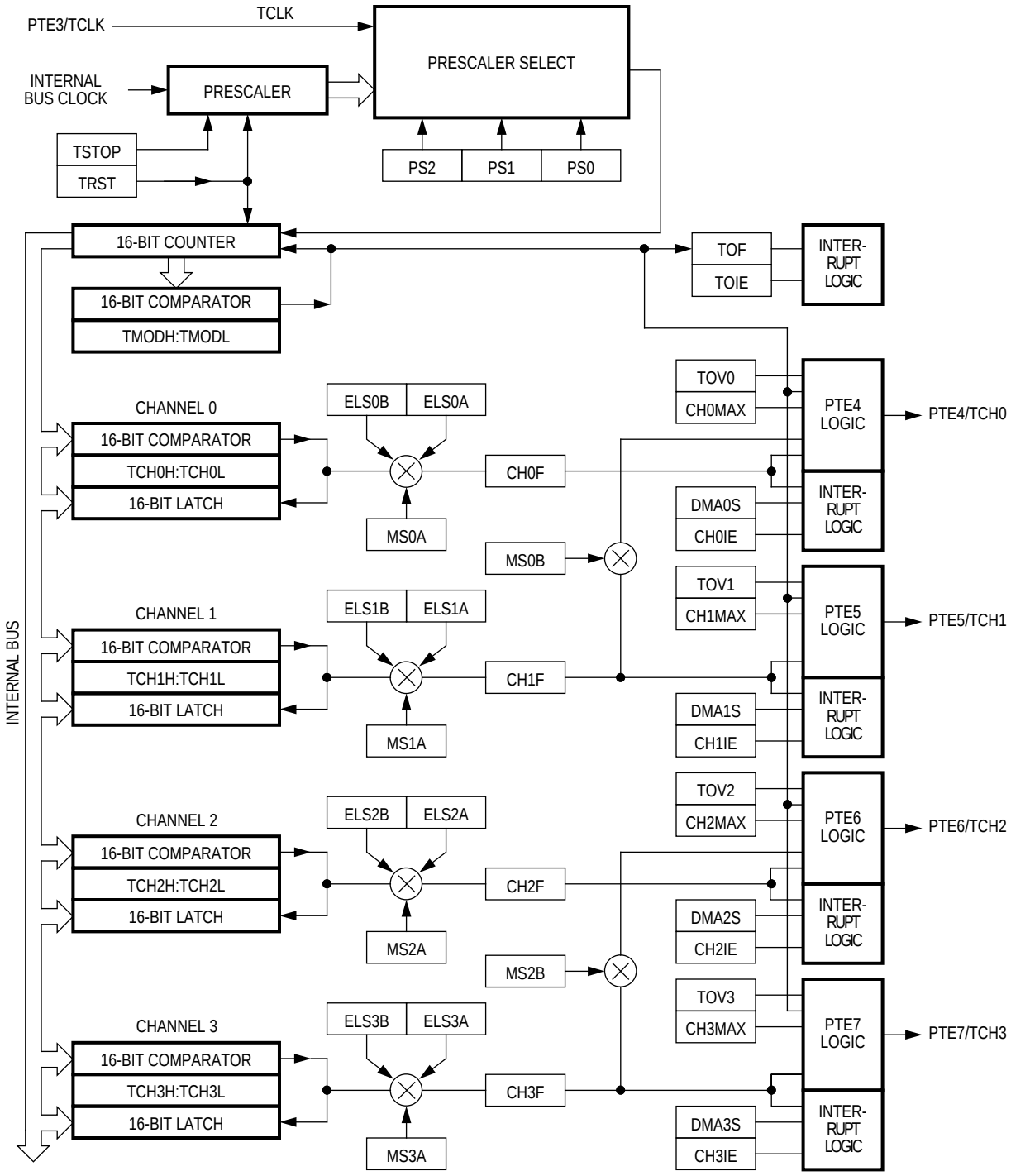


Figure 12-1. TIM Block Diagram

Table 12-1. TIM I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
TIM Status/Control Register (TSC)	TOF	TOIE	TSTOP	TRST	0	PS2	PS1	PS0	$0020
TIM DMA Select Register (TDMA)					DMAS3	DMAS2	DMAS1	DMAS0	$0021
TIM Counter Register High (TCNTH)	Bit 15	14	13	12	11	10	9	Bit 8	$0022
TIM Counter Register Low (TCNTL)	Bit 7	6	5	4	3	2	1	Bit 0	$0023
TIM Counter Modulo Reg. High (TMODH)	Bit 15	14	13	12	11	10	9	Bit 8	$0024
TIM Counter Modulo Reg. Low (TMODL)	Bit 7	6	5	4	3	2	1	Bit 0	$0025
TIM Ch. 0 Status/Control Register (TSC0)	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX	$0026
TIM Ch. 0 Register High (TCH0H)	Bit 15	14	13	12	11	10	9	Bit 8	$0027
TIM Ch. 0 Register Low (TCH0L)	Bit 7	6	5	4	3	2	1	Bit 0	$0028
TIM Ch. 1 Status/Control Register (TSC1)	CH1F	CH1IE		MS1A	ELS1B	ELS1A	TOV1	CH1MAX	$0029
TIM Ch. 1 Register High (TCH1H)	Bit 15	14	13	12	11	10	9	Bit 8	$002A
TIM Ch. 1 Register Low (TCH1L)	Bit 7	6	5	4	3	2	1	Bit 0	$002B
TIM Ch. 2 Status/Control Register (TSC2)	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX	$002C
TIM Ch. 2 Register High (TCH2H)	Bit 15	14	13	12	11	10	9	Bit 8	$002D
TIM Ch. 2 Register Low (TCH2L)	Bit 7	6	5	4	3	2	1	Bit 0	$002E
TIM Ch. 3 Status/Control Register (TSC3)	CH3F	CH3IE		MS3A	ELS3B	ELS3A	TOV3	CH3MAX	$002F
TIM Ch. 3 Register High (TCH3H)	Bit 15	14	13	12	11	10	9	Bit 8	$0030
TIM Ch. 3 Register Low (TCH3L)	Bit 7	6	5	4	3	2	1	Bit 0	$0031

= Unimplemented

12.3.1 TIM Counter Prescaler

The TIM clock source can be one of the seven prescaler outputs or the TIM clock pin, PTE3/TCLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIM status and control register select the TIM clock source.

12.3.2 Input Capture

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

12.3.3 Output Compare

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

12.3.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in **12.3.3 Output Compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM channel registers.

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

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

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

12.3.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTE4/TCH0 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The output compare value in the TIM channel 0 registers initially controls the output on the PTE4/TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE5/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 PTE6/TCH2 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIM channel 2 status and control register (TSC2) links channel 2 and channel 3. The output compare value in the TIM channel 2 registers initially controls the output on the PTE6/TCH2 pin. Writing to the TIM channel 3 registers enables the TIM channel 3 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM channel registers (2 or 3) that control the output are the ones written to last. TSC2 controls and monitors the buffered output compare function, and TIM channel 3 status and control register (TSC3) is unused. While the MS2B bit is set, the channel 3 pin, PTE7/TCH3, is available as a general-purpose I/O pin.

NOTE

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

12.3.4 Pulse Width Modulation (PWM)

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

As Figure 12-2 shows, the output compare value in the TIM channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIM to clear the channel pin on output compare if the state of the PWM pulse is logic one. Program the TIM to set the pin if the state of the PWM pulse is logic zero.

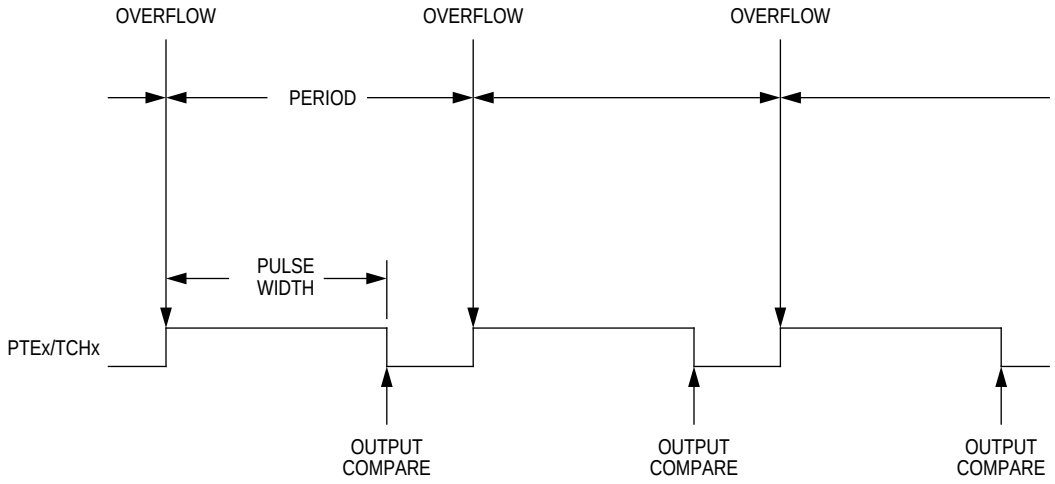


Figure 12-2. PWM Period and Pulse Width

The value in the TIM counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIM counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000 (see **12.8.1 TIM Status and Control Register (TSC)**).

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

12.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **12.3.4 Pulse Width Modulation (PWM)**. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

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

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

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

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

NOTE

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

12.3.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTE4/TCH0 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The TIM channel 0 registers initially control the pulse width on the PTE4/TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (0 or 1) that control the pulse width are the ones written to last. TSC0 controls and monitors the buffered PWM function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE5/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 PTE6/TCH2 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS2B bit in TIM channel 2 status and control register (TSC2) links channel 2 and channel 3. The TIM channel 2 registers initially control the pulse width on the PTE6/TCH2 pin. Writing to the TIM channel 3 registers enables the TIM channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (2 or 3) that control the pulse width are the ones written to last. TSC2 controls and monitors the buffered PWM function, and TIM channel 3 status and control register (TSC3) is unused. While the MS2B bit is set, the channel 3 pin, PTE7/TCH3, is available as a general-purpose I/O pin.

NOTE

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

12.3.4.3 PWM Initialization

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

1. In the TIM status and control register (TSC):
   a. Stop the TIM counter by setting the TIM stop bit, TSTOP.
   b. Reset the TIM counter by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH:TCHxL), write the value for the required pulse width.
4. In TIM channel x status and control register (TSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. (See **Table 12-3. Mode, Edge, and Level Selection**.)
   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 12-3.)

NOTE

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

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

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

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

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

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100% duty cycle output. (See **12.8.5 TIM Channel Status and Control Registers (TSC0–TSC3)**.)

12.4 Interrupts

The following TIM sources can generate interrupt requests:

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

CHxIE:DMAxS = 1:0. Channel x TIM DMA service requests are enabled when CHxIE:DMAxS = 1:1. CHxF and CHxIE are in the TIM channel x status and control register. DMAxS is in the TIM DMA select register.

12.5 Low-Power Modes

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

12.5.1 Wait Mode

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

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

The DMA can service the TIM without exiting wait mode.

12.5.2 Stop Mode

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

12.6 TIM During Break Interrupts

A break interrupt stops the TIM counter.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **7.7.3 SIM Break Flag Control Register (SBFCR)**.)

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

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

12.7 I/O Signals

Port E shares five of its pins with the TIM. PTE3/TCLK is an external clock input to the TIM prescaler. The four TIM channel I/O pins are PTE4/TCH0, PTE5/TCH1, PTE6/TCH2, and PTE7/TCH3.

12.7.1 TIM Clock Pin (PTE3/TCLK)

PTE3/TCLK is an external clock input that can be the clock source for the TIM counter instead of the prescaled internal bus clock. Select the PTE3/TCLK input by writing logic ones to the three prescaler select bits, PS[2:0]. (See **12.8.1 TIM Status and Control Register (TSC)**.) The minimum TCLK pulse width, $TCLK_{LMIN}$ or $TCLK_{HMIN}$, is:

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

The maximum TCLK frequency is:

$$\text{bus frequency} \div 2$$

PTE3/TCLK is available as a general-purpose I/O pin when not used as the TIM clock input. When the PTE3/TCLK pin is the TIM clock input, it is an input regardless of the state of the DDRE3 bit in data direction register E.

12.7.2 TIM Channel I/O Pins (PTE4/TCH0–PTE7/TCH3)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTE4/TCH0 and PTE6/TCH2 can be configured as buffered output compare or buffered PWM pins.

12.8 I/O Registers

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

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

12.8.1 TIM Status and Control Register (TSC)

The TIM status and control register does the following:

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

TSC $0020	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
Write:	0			TRST				
Reset:	0	0	1	0	0	0	0	0

= Unimplemented

Figure 12-3. TIM Status and Control Register (TSC)

TOF — TIM Overflow Flag Bit

This read/write flag is set when the TIM counter resets to $0000 after reaching the modulo value programmed in the TIM counter modulo registers. Clear TOF by reading the TIM status and control register when TOF is set and then writing a logic zero to TOF. If another TIM overflow occurs before the clearing sequence is complete, then writing logic zero to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic one to TOF has no effect.

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

TOIE — TIM Overflow Interrupt Enable Bit

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

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

TSTOP — TIM Stop Bit

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

1 = TIM counter stopped
0 = TIM counter active

NOTE

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

TRST — TIM Reset Bit

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

1 = Prescaler and TIM counter cleared
0 = No effect

NOTE

Setting the TSTOP and TRST bits simultaneously stops the TIM counter at a value of $0000.

PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTE3/TCLK pin or one of the seven prescaler outputs as the input to the TIM counter as Table 12-2 shows. Reset clears the PS[2:0] bits.

Table 12-2. Prescaler Selection

PS[2:0]	TIM Clock Source
000	Internal Bus Clock ÷1
001	Internal Bus Clock ÷ 2
010	Internal Bus Clock ÷ 4
011	Internal Bus Clock ÷ 8
100	Internal Bus Clock ÷ 16
101	Internal Bus Clock ÷ 32
110	Internal Bus Clock ÷ 64
111	PTE3/TCLK

12.8.2 TIM DMA Select Register (TDMA)

The TIM DMA select register enables either TIM CPU interrupt requests or TIM DMA service requests.

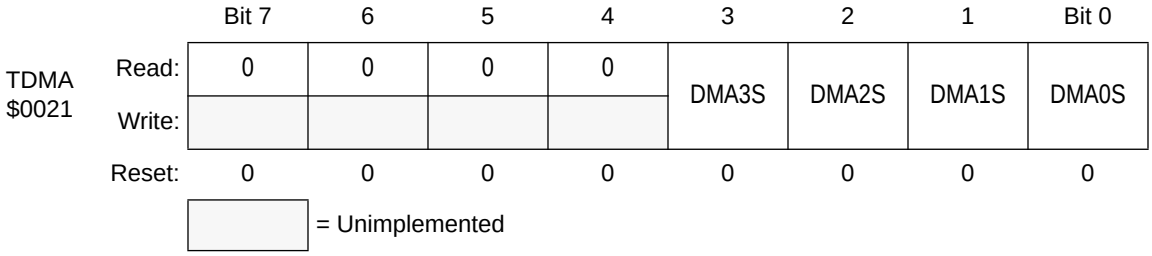


Figure 12-4. TIM DMA Select Register (TDMA)

DMA3S — DMA Channel 3 Select Bit

This read/write bit enables TIM DMA service requests on channel 3. Reset clears the DMA3S bit.

1 = TIM DMA service requests enabled on channel 3
(TIM CPU interrupt requests disabled on channel 3)
0 = TIM DMA service requests disabled on channel 3
(TIM CPU interrupt requests enabled on channel 3)

DMA2S — DMA Channel 2 Select Bit

This read/write bit enables TIM DMA service requests on channel 2. Reset clears the DMA2S bit.

1 = TIM DMA service requests enabled on channel 2
(TIM CPU interrupt requests disabled on channel 2)
0 = TIM DMA service requests disabled on channel 2
(TIM CPU interrupt requests enabled on channel 2)

DMA1S — DMA Channel 1 Select Bit

This read/write bit enables TIM DMA service requests on channel 1. Reset clears the DMA1S bit.

1 = TIM DMA service requests enabled on channel 1
(TIM CPU interrupt requests disabled on channel 1)
0 = TIM DMA service requests disabled on channel 1
(TIM CPU interrupt requests enabled on channel 1)

DMA0S — DMA Channel 0 Select Bit

This read/write bit enables TIM DMA service requests on channel 0. Reset clears the DMA0S bit.

1 = TIM DMA service requests enabled on channel 0
(TIM CPU interrupt requests disabled on channel 0)
0 = TIM DMA service requests disabled on channel 0
(TIM CPU interrupt requests enabled on channel 0)

12.8.3 TIM Counter Registers (TCNTH:TCNTL)

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

NOTE

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

		Bit 7	6	5	4	3	2	1	Bit 0
TCNTH $0022	Read:	Bit 15	14	13	12	11	10	9	Bit 8
	Write:								
	Reset:	0	0	0	0	0	0	0	0
		Bit 7	6	5	4	3	2	1	Bit 0
TCNTL $0023	Read:	Bit 7	6	5	4	3	2	1	Bit 0
	Write:								
		0	0	0	0	0	0	0	0
	Reset:		= Unimplemented						

Figure 12-5. TIM Counter Registers (TCNTH:TCNTL)

12.8.4 TIM Counter Modulo Registers (TMODH:TMODL)

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

		Bit 7	6	5	4	3	2	1	Bit 0
TMODH $0024	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	1	1	1	1	1	1	1	1
		Bit 7	6	5	4	3	2	1	Bit 0
TMODL $0025	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	1	1	1	1	1	1	1	1

Figure 12-6. TIM Counter Modulo Registers (TMODH:TMODL)

NOTE

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

12

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

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

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

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

= Unimplemented

Figure 12-7. TIM Channel Status and Control Registers (TSC0–TSC3)

CHxF— Channel x Flag Bit

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

When TIM CPU interrupt requests are enabled (CHxIE:DMAxS = 1:0), clear CHxF by reading TIM channel x status and control register with CHxF set and then writing a logic zero to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic zero to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

When TIM DMA service requests are enabled (CHxIE:DMAxS = 1:1), clear CHxF by reading or writing to the low byte of the TIM channel x registers (TCHxL).

Reset clears the CHxF bit. Writing a logic one to CHxF has no effect.

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

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIM CPU interrupts and TIM DMA service requests on channel x. The DMAxS bit in the TIM 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 TIM DMA select register.

Reset clears the CHxIE bit.

1 = Channel x CPU interrupt requests and DMA service requests enabled
0 = Channel x CPU interrupt requests and DMA service requests disabled

MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIM channel 0 and TIM channel 2 status and control registers.

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

Setting MS2B disables the channel 3 status and control register and reverts TCH3 to general-purpose I/O.

Reset clears the MSxB bit.

1 = Buffered output compare/PWM operation enabled
0 = Buffered output compare/PWM operation disabled

MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. (See **Table 12-3. Mode, Edge, and Level Selection**.)

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 12-3.). 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 TIM status and control register (TSC).

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port E, and pin PTEx/TCHx is available as a general-purpose I/O pin. Table 12-3 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 12-3. Mode, Edge, and Level Selection

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

NOTE

Before enabling a TIM channel register for input capture operation, make sure that the PTE/TCHx pin is stable for at least two bus clocks.

TOVx — Toggle-On-Overflow Bit

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

1 = Channel x pin toggles on TIM counter overflow.
0 = Channel x pin does not toggle on TIM counter overflow.

NOTE

When TOVx is set, a TIM counter overflow takes precedence over a channel x output compare if both occur at the same time.

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic zero, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 12-8 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.

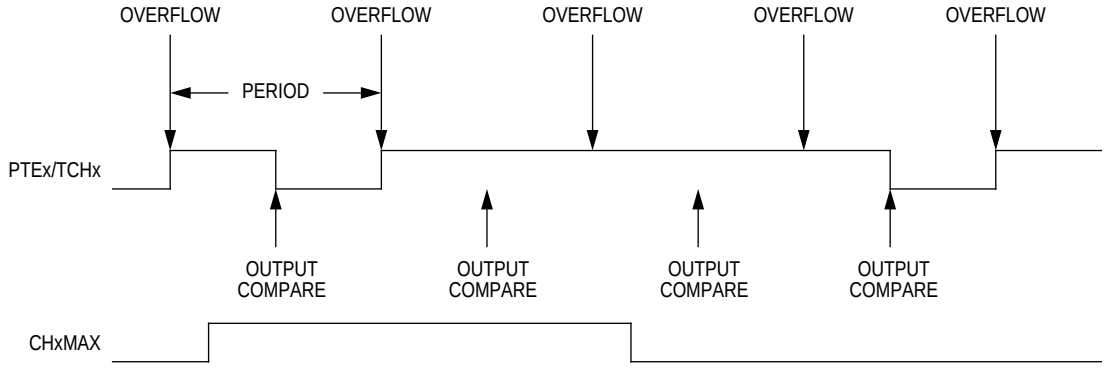


Figure 12-8. CHxMAX Latency

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

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

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

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

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

Figure 12-9. TIM Channel Registers (TCH0H/L–TCH3H/L)

SECTION 13
SERIAL PERIPHERAL INTERFACE MODULE (SPI)

13.1 Introduction

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

13.2 Features

Features of the SPI module include the following:

- Full-Duplex Operation
- Master and Slave Modes
- Separate Transmit and Receive Registers
- Four Master Mode Frequencies (Maximum = Bus Frequency ÷ 2)
- Maximum Slave Mode Frequency = Bus Frequency
- Separate Clock Ground for Reduced Radio Frequency (RF) Interference
- Serial Clock with Programmable Polarity and Phase
- Bus Contention Error Flag
- Overrun Error Flag
- Two Separately Enabled Interrupts with DMA or CPU Service:
  - SPRF (SPI Receiver Full)
  - SPTE (SPI Transmitter Empty)
- Programmable Wired-OR Mode
- I^2C (Inter-Integrated Circuit) Compatibility

13.3 Functional Description

Figure 13-1 shows the structure of the SPI module.

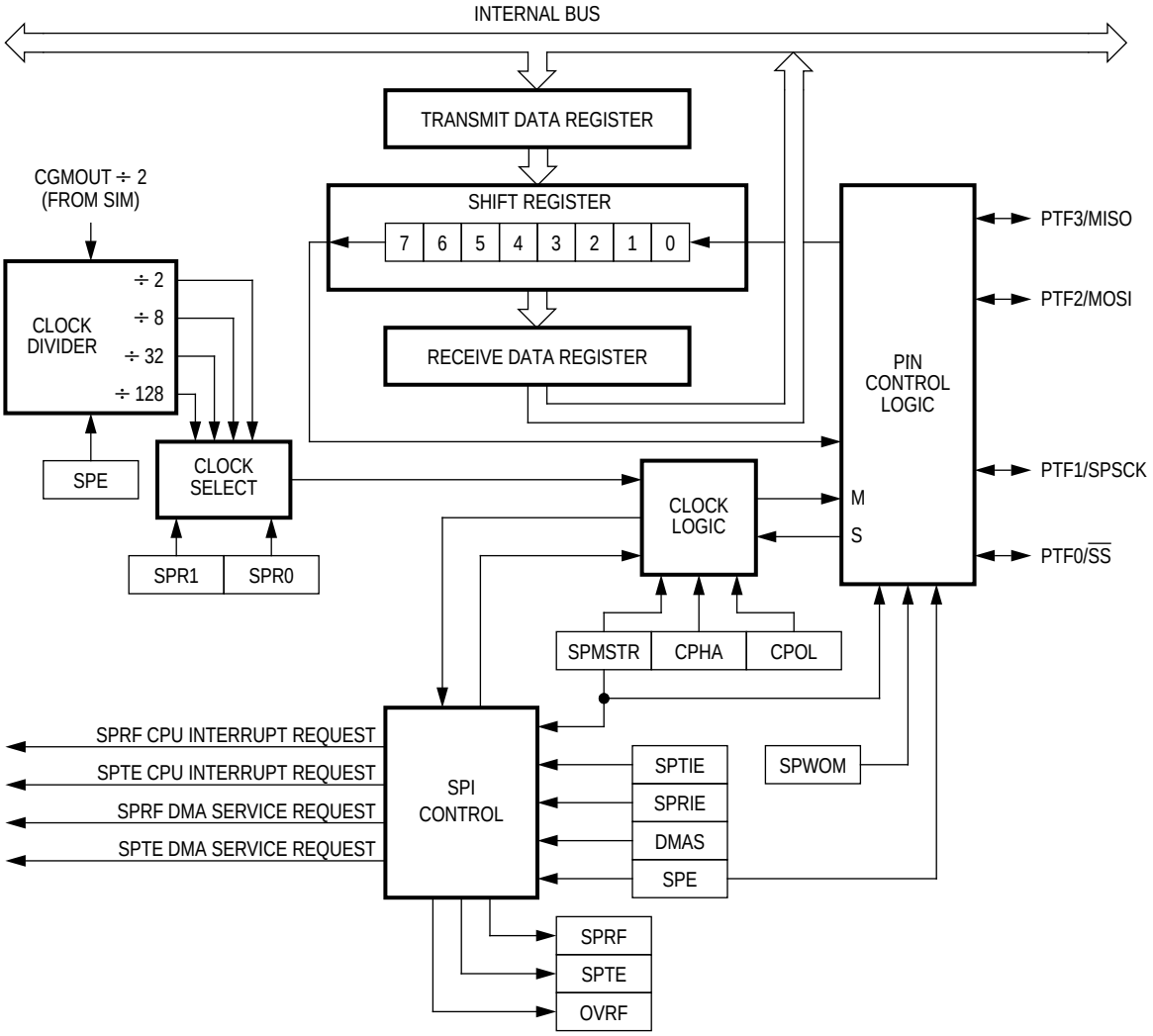


Figure 13-1. SPI Module Block Diagram

Table 13-1. SPI I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
SPI Control Register (SPCR)	SPRIE	DMAS	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE	$0010
SPI Status and Control Register (SPSCR)	SPRF		OVRF	MODF	SPTE		SPR1	SPR0	$0011
SPI Data Register (SPDR)									$0012

= Unimplemented

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. SPI interrupts can be serviced by either the DMA or the CPU.

During DMA transfers, the DMA fetches data from memory for the SPI to transmit and/or the DMA stores received data in memory.

The following paragraphs describe the operation of the SPI module.

13.3.1 Master Mode

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

NOTE

Configure the SPI modules as master and slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. (See **13.8.1 SPI Control Register (SPCR)**.)

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the SPI data register. The byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The byte begins shifting out on the PTF2/MOSI pin under the control of the serial clock. (See **Figure 13-2. Full-Duplex Master-Slave Connections**.)

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See **13.8.2 SPI Status and Control Register (SPSCR)**.) Through the PTF1/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 PTF2/MOSI pin of the master, another byte shifts in from the slave on the master's PTF3/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 and then reading the SPI data register.

In a DMA transfer, the DMA automatically clears SPRF by reading or writing the SPI data register.

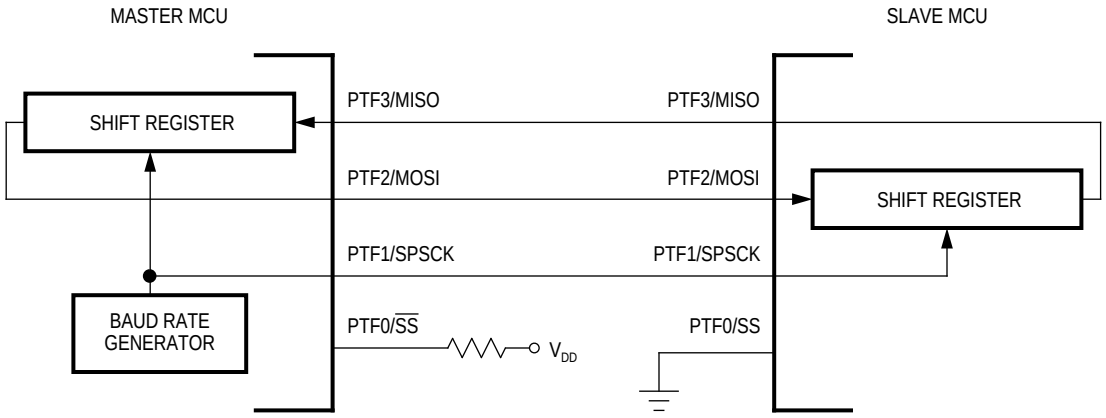


Figure 13-2. Full-Duplex Master-Slave Connections

13.3.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit is clear. In slave mode the PTF1/SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the PTF0/$\overline{SS}$ pin of the slave MCU must be at logic zero. PTF0/$\overline{SS}$ must remain low until the transmission is complete.

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

13.3.3 Serial Clock Polarity and Phase

13

To accommodate the different serial communication requirements of peripheral devices, software can change the phase and polarity of the SPI serial clock. The clock polarity bit, CPOL, and the clock phase bit, CPHA, both in the SPI control register, control the timing relationship between the serial clock and the transmitted data. Figure 13-3 shows how the CPOL and CPHA bits affect clock/data timing.

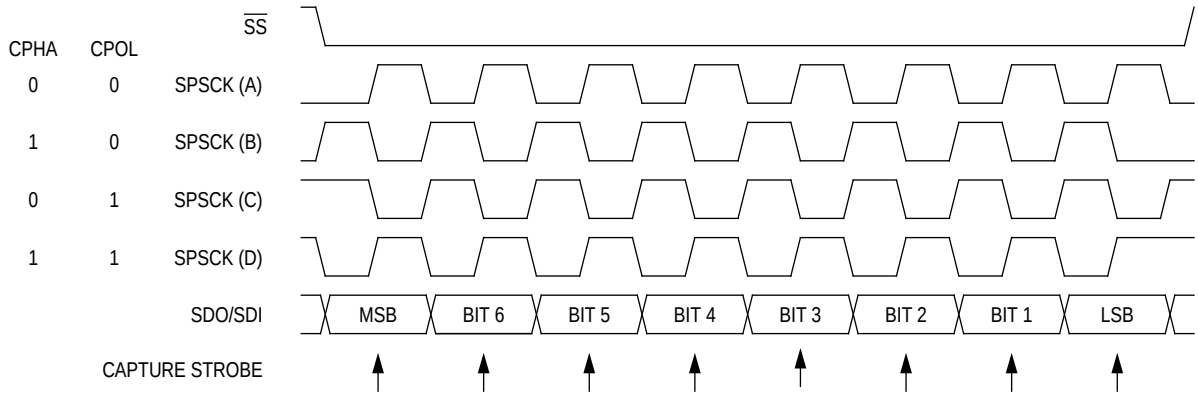


Figure 13-3. SPI Data/Clock Timing Diagram

13.3.4 Error Conditions

The following flags signal SPI error conditions:

- Mode fault error (MODF) — The MODF bit indicates that the slave select pin (PTF0/$\overline{SS}$) of a master SPI module is at logic zero. MODF is in the SPI status and control register.
- Overflow (OVRF) — Failing to read the SPI data register before the next byte enters the shift register sets the OVRF bit. The new byte does not transfer to the receive data register, and the unread byte still can be read by accessing the SPI data register. OVRF is in the SPI status and control register.

13.4 Interrupts

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

- SPI receiver full bit (SPRF) — The SPRF bit becomes set every time a byte transfers from the shift register to the receive data register. If the SPI receiver interrupt enable bit, SPRIE, is also set, SPRF can generate either an SPRF 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.

Figure 13-4. SPRF/SPTE Interrupt Timing

13.5 Low-Power Modes

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

13.5.1 Wait Mode

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

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

The DMA can service the SPI without exiting wait mode.

13.5.2 Stop Mode

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

13.6 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 **7.7.3 SIM Break Flag Control Register (SBFCR)**.)

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

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

13.7 I/O Signals

The SPI module has five I/O pins and shares four of them with port F:

- PTF3/MISO — Data received
- PTF2/MOSI — Data transmitted
- PTF1/SPSCK — Serial clock
- PTF0/$\overline{SS}$ — Slave select
- CGND/EV_{SS} — Clock ground

The SPI has limited inter-integrated circuit (I^2C) capability (requiring software support) as a master in a single-master environment. To communicate with I^2C peripherals, PTF2/MOSI becomes an open-drain output when the SPWOM bit in the SPI control register is set. In I^2C communication, the PTF2/MOSI and PTF3/MISO pins are connected to a bidirectional pin from the I^2C peripheral and through a pull-up resistor to V_{DD}.

13.7.1 PTF3/MISO (Master In/Slave Out)

PTF3/MISO is one of the two SPI module pins that transmits serial data. In full duplex operation, the PTF3/MISO pin of the master SPI module is connected to the PTF3/MISO pin of the slave SPI module. The master SPI simultaneously receives data on its PTF3/MISO pin and transmits data from its PTF2/MOSI pin.

Slave output data on the PTF3/MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is logic zero and its PTF0/$\overline{SS}$ pin is at logic zero.

When enabled, the SPI controls data direction of the PTF3/MISO pin regardless of the state of the DDRF3 bit in data direction register F.

13.7.2 PTF2/MOSI (Master Out/Slave In)

PTF2/MOSI is one of the two SPI module pins that transmits serial data. In full duplex operation, the PTF2/MOSI pin of the master SPI module is connected to the PTF2/MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its PTF2/MOSI pin and receives data on its PTF3/MISO pin. In a multiple-slave system, a logic one on the PTF0/$\overline{SS}$ pin puts the PTF3/MISO pin in a high-impedance state.

When enabled, the SPI controls data direction of the PTF2/MOSI pin regardless of the state of the DDRF2 bit in data direction register F.

13.7.3 PTF1/SPSCK (Serial Clock)

The serial clock synchronizes data transmission between master and slave devices. In a master MCU, the PTF1/SPSCK pin is the clock output. In a slave MCU, the PTF1/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 PTF1/SPSCK pin regardless of the state of the DDRF1 bit in data direction register F.

13.7.4 PTF0/$\overline{SS}$ (Slave Select)

Applying a logic one to the PTF0/$\overline{SS}$ pin of a master SPI module enables the module to transmit data from its PTF2/MOSI pin. Applying a logic zero to the PTF0/$\overline{SS}$ pin of a master SPI module sets the mode fault error bit, MODF.

Applying a logic zero to the PTF0/$\overline{SS}$ pin of a slave SPI module enables the module to receive data on its PTF2/MOSI pin. The PTF0/$\overline{SS}$ pin of a slave SPI module must be at logic zero. Applying a logic one to the PTF0/$\overline{SS}$ pin disables the slave SPI module.

The PTF0/$\overline{SS}$ pin of a slave SPI module must be at logic zero before a transmission begins and remain low throughout the transmission. If the PTF0/$\overline{SS}$ pin goes high during a transmission, the transmission stops.

When the CPHA bit in the SPI control register is at logic zero, the PTF0/$\overline{SS}$ pin must be deasserted and then reasserted between each transmitted byte. The falling edge of the PTF0/$\overline{SS}$ pin provides the first clock edge to the SPI shift register. When the CPHA bit is at logic one, the PTF0/$\overline{SS}$ pin can remain low between transmitted bytes. (See Figure 13-5.)

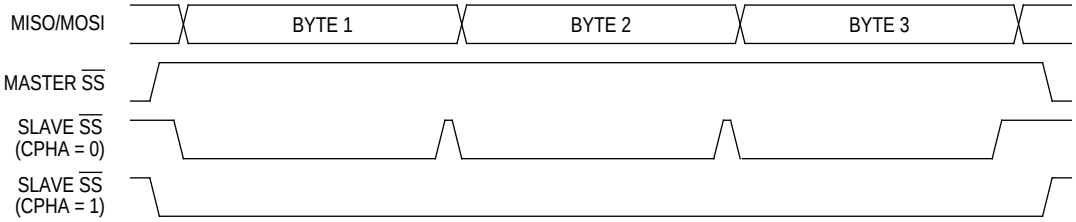


Figure 13-5. CPHA/$\overline{SS}$ Timing

The PTF0/$\overline{SS}$ pin of a master SPI is available as a general-purpose I/O pin. When using a master PTF0/$\overline{SS}$ pin for general-purpose I/O, the mode fault bit, MODF, can be ignored.

The PTF0/$\overline{SS}$ pin of a slave SPI is an input regardless of the state of the DDRF0 bit in data direction register F.

13.7.5 CGND/EV_{SS} (Clock Ground)

CGND/EV_{SS} is the ground for the port output buffers and the ground return for the serial clock pin, PTF1/SPSCK. To reduce the ground return path loop and minimize radio frequency (RF) emissions, connect the ground pin of the slave to the CGND/EV_{SS} pin.

13.8 I/O Registers

Three registers control and monitor SPI operation:

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

13.8.1 SPI Control Register (SPCR)

The SPI control register does the following:

- Enables SPI module interrupt requests
- Selects CPU interrupt requests or DMA service requests
- Configures the SPI module as master or slave
- Selects serial clock polarity and phase
- Configures the PTF1/SPSCK, PTF2/MOSI, and PTF3/MISO pins as open-drain outputs
- Enables the SPI module

		Bit 7	6	5	4	3	2	1	Bit 0
SPCR $0010	Read: / Write:	SPRIE	DMAS	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
	Reset:	0	0	1	0	1	0	0	0

Figure 13-6. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

This read/write bit enables CPU interrupt requests or DMA service requests generated by the SPRF bit. The SPRF bit is set when a byte transfers from the shift register to the receive data register. Reset clears the SPRIE bit. Disabling the SPI module by clearing the SPE bit also clears SPRIE.

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

NOTE

The SPRIE bit is writable only when the SPE bit is set.

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. Reset clears the DMAS bit.

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)

SPMSTR — SPI Master Bit

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

1 = Master mode
0 = Slave mode

CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the PTF1/SPSCK pin between transmissions. (See Figure 13-3.) To transmit data between SPI modules, the SPI modules must have identical CPOL bits. Reset clears the CPOL bit.

CPHA — Clock Phase Bit

13

This read/write bit controls the timing relationship between the serial clock and SPI data. To transmit data between SPI modules, the SPI modules must have identical CPHA bits. When CPHA = 0, the PTF0/$\overline{SS}$ pin of the slave SPI module must be set to logic one between bytes. Reset sets the CPHA bit.

SPWOM — SPI Wired-OR Mode Bit

This read/write bit disables the pull-up devices on pins PTF1/SPSCK, PTF2/MOSI, and PTF3/MISO so that those pins become open-drain outputs.

1 = Wired-OR PTF1/SPSCK, PTF2/MOSI, and PTF3/MISO pins
0 = Normal push-pull PTF1/SPSCK, PTF2/MOSI, and PTF3/MISO pins

SPE — SPI Enable

This read/write bit enables the SPI module. Clearing SPE returns the SPI status and control register to its reset state. (See 13.8.2.) Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE— SPI Transmit Interrupt Enable

This read/write bit enables CPU interrupt requests or DMA service requests generated by the SPTE bit. SPTE is set when a byte transfers from the transmit data register to the shift register. Reset clears the SPTIE bit. Disabling the SPI module by clearing the SPE bit also clears SPTIE.

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

---

NOTE

The SPTIE bit is writable only when the SPE bit is set.

13.8.2 SPI Status and Control Register (SPSCR)

The SPI status and control register contains flags to signal the following conditions:

- Receive data register full
- Failure to clear SPRF bit before next byte is received
- Logic zero on PTF0/$\overline{SS}$ pin of SPI module in master mode
- Transmit data register empty

The SPI status and control register also contains 2 bits that select the SPI baud rate.

SPSCR $0011	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SPRF	0	OVRF	MODF	SPTE	0	SPR1	SPR0
Write:								
Reset:	0	0	0	0	1	0	0	0

= Unimplemented

Figure 13-7. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

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

The DMA select bit (DMAS) in the SPI control register determines whether SPRF generates an SPRF CPU interrupt request or an SPRF DMA service request. During an SPRF CPU interrupt (DMAS = 0), the CPU clears SPRF by reading the SPI status and control register and then reading the SPI data register. During an SPRF DMA transfer (DMAS = 1), the DMA automatically clears SPRF.

Reset clears the SPRF bit.

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

OVRF — Overflow Bit

This clearable, read-only flag is set if software does not read the byte in the receive data register before the next byte enters the shift register. In an overflow condition, the byte already in the receive data register is unaffected, and the byte that shifted in last is lost. Clear the OVRF bit by reading the SPI status and control register and then reading the SPI data register. Reset clears the OVRF bit.

1 = Overflow
0 = No overflow

MODF — Mode Fault Bit

The MODF bit is meaningful only when the SPI module is in master mode. This clearable, read-only flag is set if the PTF0/$\overline{SS}$ pin goes low when the SPMSTR bit is at logic one. Clear the MODF bit by writing to the SPI status and control register. Reset clears the MODF bit.

SPTE — SPI Transmitter Empty Bit

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

The DMA select bit (DMAS) in the SPI control register determines whether SPTE generates an SPTE CPU interrupt request or an SPTE DMA service request. During an SPTE CPU interrupt (DMAS = 0), the CPU clears the SPTE bit by writing to the SPI data register. During an SPTE DMA transfer (DMAS = 1), the DMA automatically clears SPTE.

Reset sets the SPTE bit.

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

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 13-2. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 13-2. SPI Baud Rate Selection

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

Use the following formula to calculate the SPI baud rate:

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

where:

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

13

13.8.3 SPI Data Register (SPDR)

The SPI data register is the read/write buffer for the receive data register and the transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register.

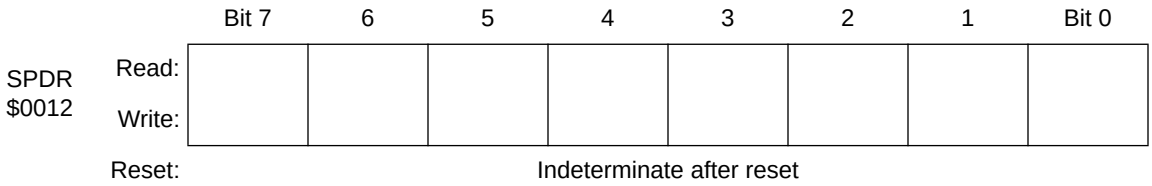


Figure 13-8. SPI Data Register (SPDR)

SECTION 14
SERIAL COMMUNICATIONS INTERFACE MODULE (SCI)

14.1 Introduction

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

14.2 Features

Features of the SCI module include the following:

- Full Duplex Operation
- Standard Mark/Space Non-Return-to-Zero (NRZ) Format
- 32 Programmable Baud Rates
- 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 Wake-Up Methods:
  - Idle Line Wake-Up
  - Address Mark Wake-Up
- Interrupt-Driven Operation with Eight Interrupt Flags:
  - Transmitter Empty
  - Transmission Complete
  - Receiver Full
  - Idle Receiver Input
  - Receiver Overrun
  - Noise Error
  - Framing Error
  - Parity Error
- Receiver Framing Error Detection
- Hardware Parity Checking
- 1/16 Bit-Time Noise Detection

14.3 Functional Description

Figure 14-1 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.

During DMA transfers, the DMA fetches data from memory for the SCI to transmit and/or the DMA stores received data in memory.

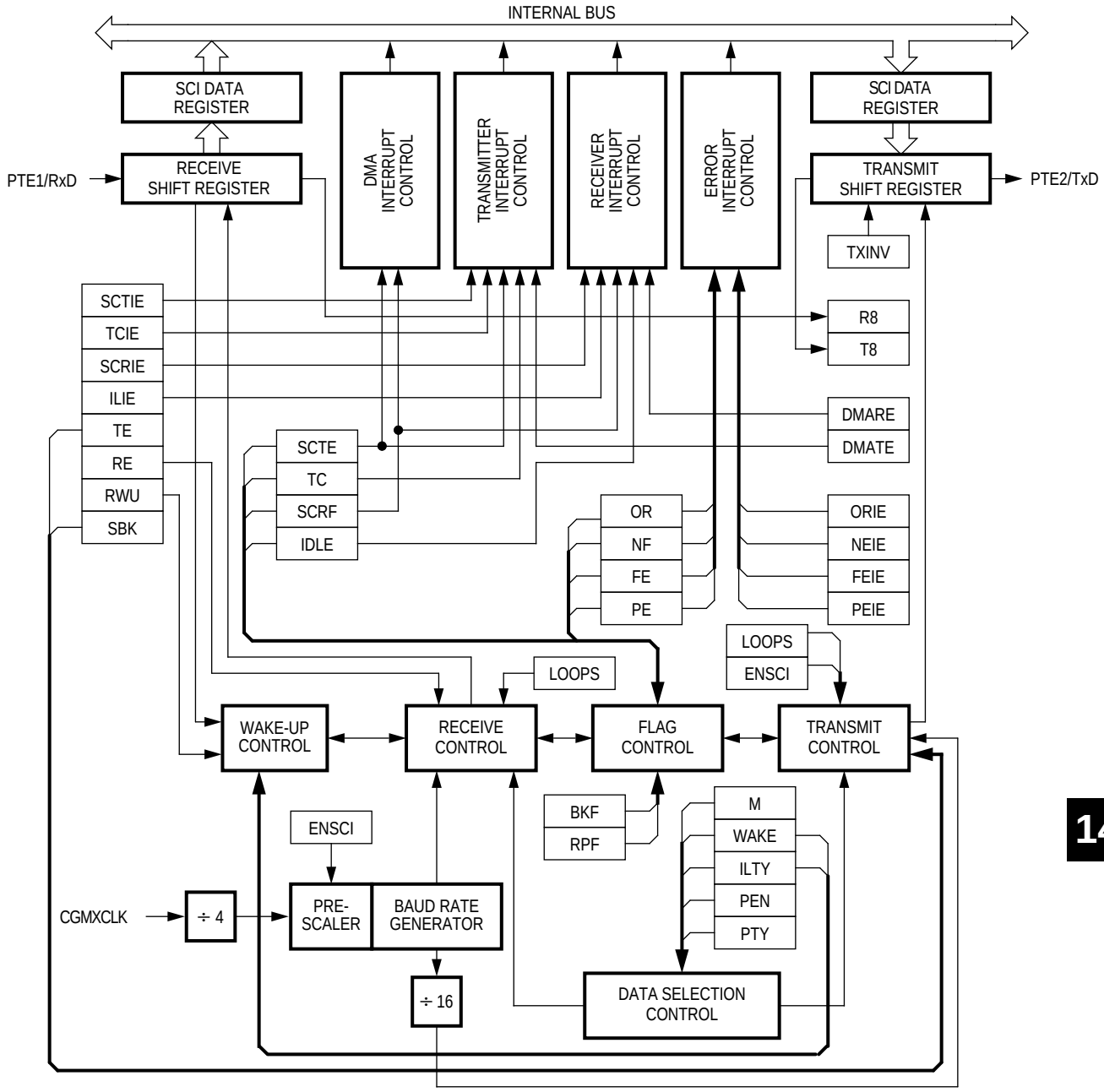


Figure 14-1. SCI Module Block Diagram

Table 14-1. SCI I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
SCI Control Register 1 (SCC1)	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY	$0013
SCI Control Register 2 (SCC2)	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK	$0014
SCI Control Register 3 (SCC3)	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE	$0015
SCI Status Register 1 (SCS1)	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE	$0016
SCI Status Register 2 (SCS2)							BKF	RPF	$0017
SCI Data Register (SCDR)									$0018
SCI Baud Rate Register (SCBR)			SCP1	SCP0		SCR2	SCR1	SCR0	$0019

= Unimplemented

14.3.1 Data Format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in Figure 14-2.

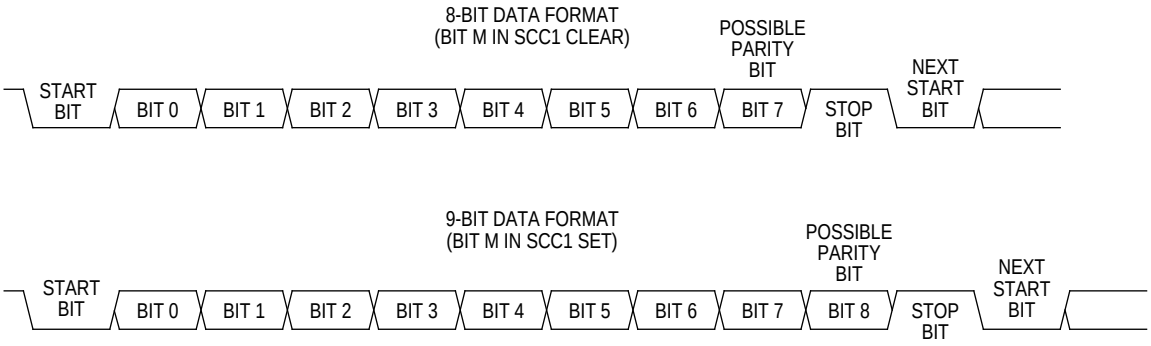


Figure 14-2. SCI Data Formats

14.3.2 Transmitter

Figure 14-3 shows the structure of the SCI transmitter.

14.3.2.1 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).

14.3.2.2 Character Transmission

During an SCI transmission, the transmit shift register shifts a character out to the PTE2/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 one to the enable SCI bit (ENSCI) in SCI control register 1 (SCC1).
2. Enable the transmitter by writing a logic one 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). In a DMA transfer, the DMA automatically clears the SCTE bit.
4. Write the data to transmit into the SCDR. In a DMA transfer, the DMA automatically writes to 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 ones. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A logic zero start bit automatically goes into the least significant bit position of the transmit shift register. A logic one stop bit goes into the most significant bit 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 or an SCTE DMA service request.

The SCTE bit generates an SCTE DMA service request if the DMA transfer enable bit, DMATE, in SCI control register 3 (SCC3) is set. Setting the DMATE bit enables SCTE DMA service requests and disables SCTE CPU interrupt requests.

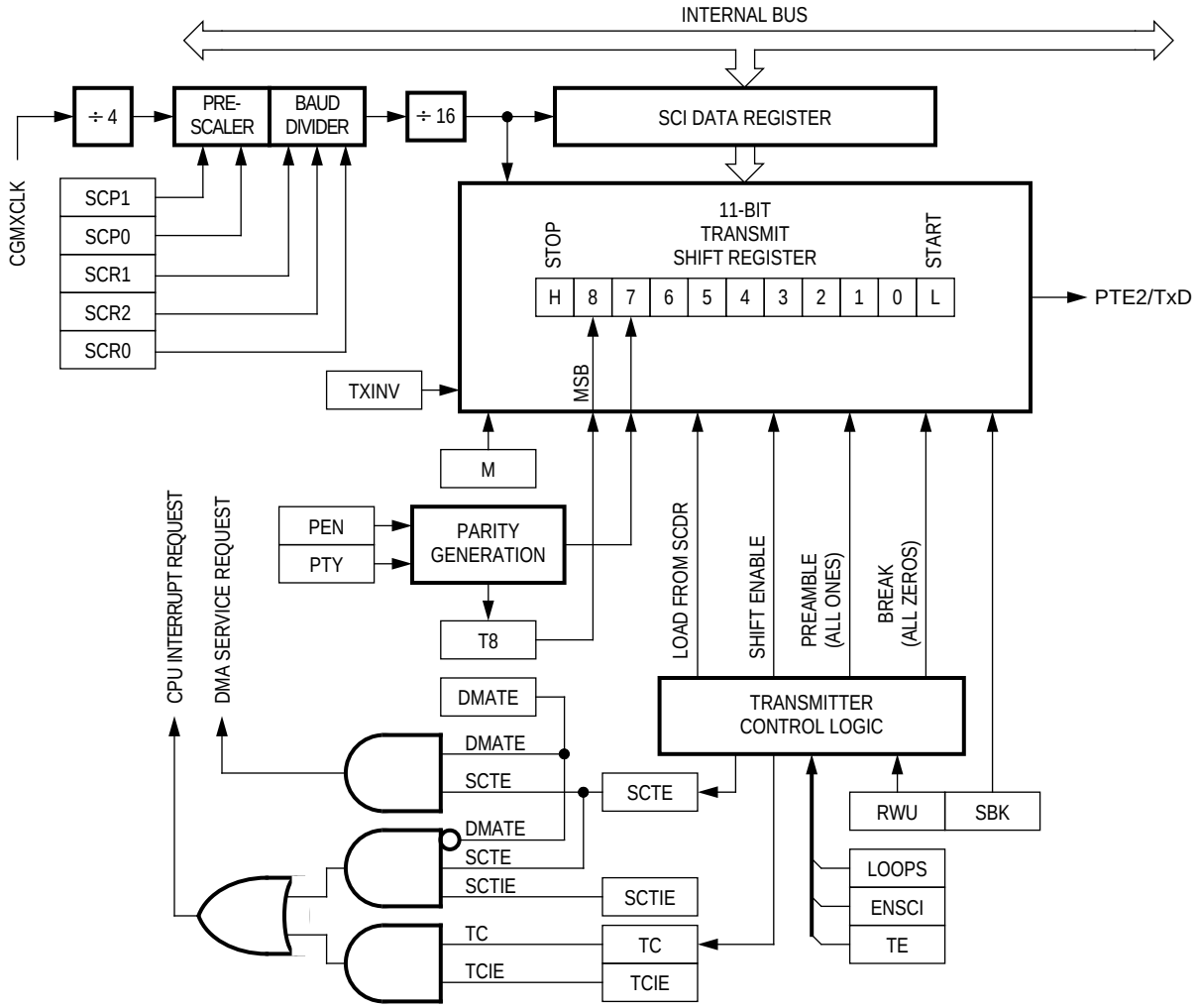


Figure 14-3. SCI Transmitter

14

Table 14-2. SCI Transmitter I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
SCI Control Register 1 (SCC1)	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY	$0013
SCI Control Register 2 (SCC2)	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK	$0014
SCI Control Register 3 (SCC3)	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE	$0015
SCI Status Register 1 (SCS1)	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE	$0016
SCI Data Register (SCDR)									$0018
SCI Baud Rate Register (SCBR)			SCP1	SCP0		SCR2	SCR1	SCR0	$0019

= Unimplemented

When the transmit shift register is not transmitting a character, the PTE2/TxD pin goes to the idle condition, logic one. 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.

14.3.2.3 Break Characters

Writing a logic one to the send break bit, SBK, in SCC2 loads the transmit shift register with a break character. A break character contains all logic zeros 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 one, 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 one. The automatic logic one at the end of a break character guarantees the recognition of the start bit of the next character.

Receiving a break character sets the framing error bit (FE) in SCI status register 1 (SCS1) and may also set the parity error bit (PE).

14.3.2.4 Idle Characters

An idle character contains all logic ones 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 PTE2/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 one before the stop bit of the current character shifts out to the PTE2/TxD pin. Setting TE after the stop bit appears on PTE2/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.

14.3.2.5 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 one. (See **14.7.1 SCI Control Register 1 (SCC1)**.)

14.3.2.6 Transmitter Interrupts

The following 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 or an SCTE DMA service request. Setting the SCI transmit interrupt enable bit, SCTIE, in SCC2 enables SCTE CPU interrupts. Setting both the SCTIE bit and the DMA transfer enable bit, DMATE, in SCC3 enables SCTE DMA service requests.
- Transmission complete (TC) — The TC bit in SCS1 indicates that the transmit shift register and the SCDR are empty and that no break or idle character has been generated. The SCI transmitter interrupt enable bit, SCTIE, in SCC2 enables TC CPU interrupt requests.

14.3.3 Receiver

Figure 14-4 shows the structure of the SCI receiver.

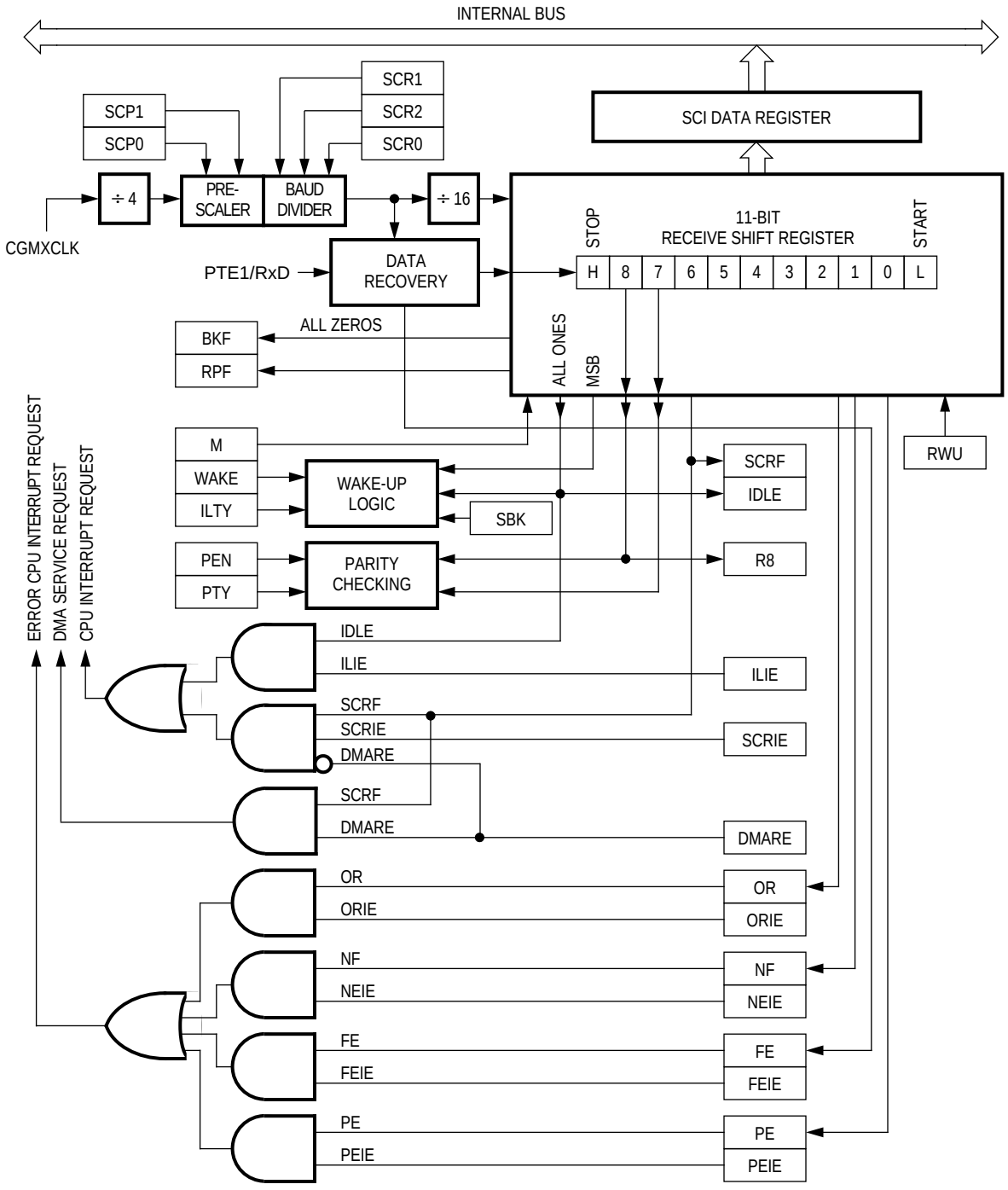


Figure 14-4. SCI Receiver Block Diagram

Table 14-3. SCI Receiver I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
SCI Control Register 1 (SCC1)	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY	$0013
SCI Control Register 2 (SCC2)	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK	$0014
SCI Control Register 3 (SCC3)	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE	$0015
SCI Status Register 1 (SCS1)	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE	$0016
SCI Status Register 2 (SCS2)							BKF	RPF	$0017
SCI Data Register (SCDR)									$0018
SCI Baud Rate Register (SCBR)			SCP1	SCP0		SCR2	SCR1	SCR0	$0019

= Unimplemented

14.3.3.1 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).

14.3.3.2 Character Reception

During an SCI reception, the receive shift register shifts characters in from the PTE1/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 or an SCRF DMA service request.

The SCRF bit generates an SCRF DMA service request if the DMA receive enable bit, DMARE, in SCI control register 3 (SCC3) is set. Setting the DMARE bit enables SCRF DMA service requests and disables SCRF CPU interrupt requests.

14.3.3.3 Data Sampling

The receiver samples the PTE1/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 14-5.)

- Start bit — To locate the start bit, recovery logic does an asynchronous search for a logic zero preceded by three logic ones. 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 ones, the RT clock is reset and the search for start bit begins again. If all three samples are logic zeros, start bit verification is successful. If only one of the three samples is logic one, 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.

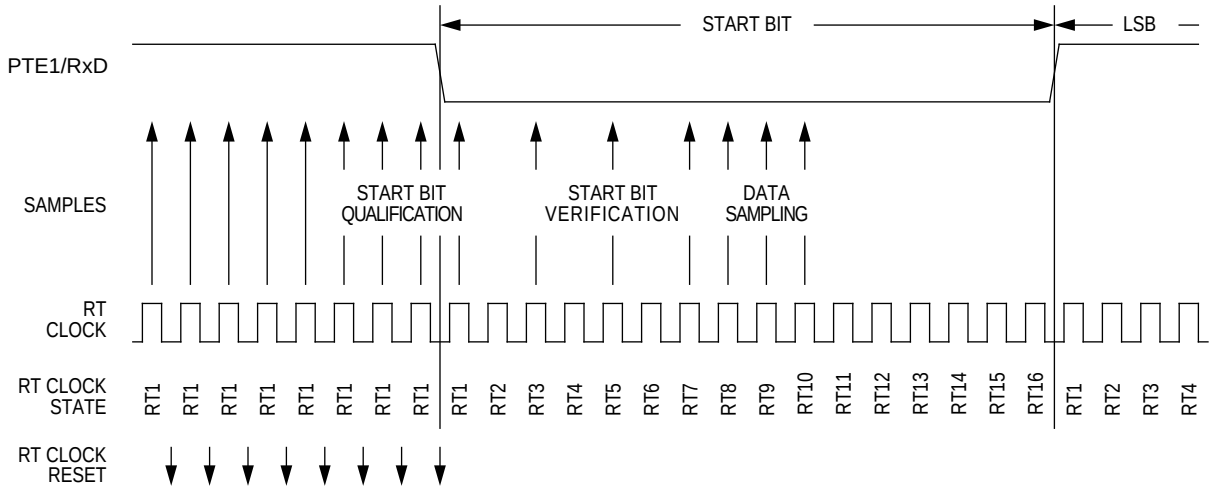


Figure 14-5. Receiver Data Sampling

14.3.3.4 Framing Errors

If the data recovery logic does not detect a logic one where the stop bit should be in an incoming character, it sets the framing error bit, FE, in SCS1. The FE flag is set at the same time that the SCRF bit is set. A break character also sets the FE bit.

14.3.3.5 Receiver Wake-Up

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 wake-up 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 PTE1/RxD pin can bring the MCU out of the standby state:

- Address mark — When the WAKE bit is set and a logic one occurs in the most significant bit position of a received character, receiver interrupts are enabled again.
- Idle input line condition — When the WAKE bit is clear and the PTE1/RxD pin is at logic one long enough for 10 or 11 logic ones to shift into the receive shift register, receiver interrupts are enabled again.

14.3.3.6 Receiver Interrupts

The following 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 ones shifted in from the PTE1/RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables IDLE CPU interrupts.

14.3.3.7 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 zero 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.

14.4 Low-Power Modes

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

14.4.1 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.

The DMA can service the SCI without exiting wait mode.

14.4.2 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.

14.5 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 **7.7.3 SIM Break Flag Control Register (SBFCR)**.)

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

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

14.6 I/O Signals

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

- PTE2/TxD — Transmit data
- PTE1/RxD — Receive data

14.6.1 PTE2/TxD (Transmit Data)

The PTE2/TxD pin is the serial data output from the SCI transmitter. The SCI shares the PTE2/TxD pin with port E. When the SCI is enabled, the PTE2/TxD pin is an output regardless of the state of the DDRE2 bit in data direction register E (DDRE).

14.6.2 PTE1/RxD (Receive Data)

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

14.7 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)

14.7.1 SCI Control Register 1 (SCC1)

SCI control register 1 does the following:

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

		Bit 7	6	5	4	3	2	1	Bit 0
SCC1 $0013	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
	Write:								
	Reset:	0	0	0	0	0	0	0	0

Figure 14-6. 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 8 or 9 bits long. (See Table 14-4.) The ninth bit can serve as an extra stop bit, as a receiver wake-up signal, or as a parity bit. Reset clears the M bit.

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

WAKE — Wake-Up Condition Bit

This read/write bit determines which condition wakes up the SCI: a logic one (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 wake-up
0 = Idle line wake-up

ILTY — Idle Line Type Bit

This read/write bit determines when the SCI starts counting logic ones 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 ones 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 14-4.) When enabled, the parity function inserts a parity bit in the most significant bit position. (See **Figure 14-2. SCI Data Formats**.) 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 14-4.) Reset clears the PTY bit.

1 = Odd parity
0 = Even parity

Table 14-4. 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

14.7.2 SCI Control Register 2 (SCC2)

SCI control register 2 does the following:

- Enables the following interrupts:
  - Transmitter interrupts
  - Transmission complete interrupts
  - Receiver interrupts
  - Idle line interrupts
- Enables the transmitter

- Enables the receiver
- Enables SCI wake-up
- Transmits SCI break characters

		Bit 7	6	5	4	3	2	1	Bit 0
SCC2 $0014	Read: Write:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
	Reset:	0	0	0	0	0	0	0	0

Figure 14-7. 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 ones 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 PTE2/TxD returns to the idle condition (logic one). 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 Wake-Up 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 one. The logic one after the break character guarantees recognition of a valid start bit. If SBK remains set, the transmitter continuously transmits break characters with no logic ones between them. Reset clears the SBK bit.

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

14.7.3 SCI Control Register 3 (SCC3)

SCI control register 3 does the following:

- 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

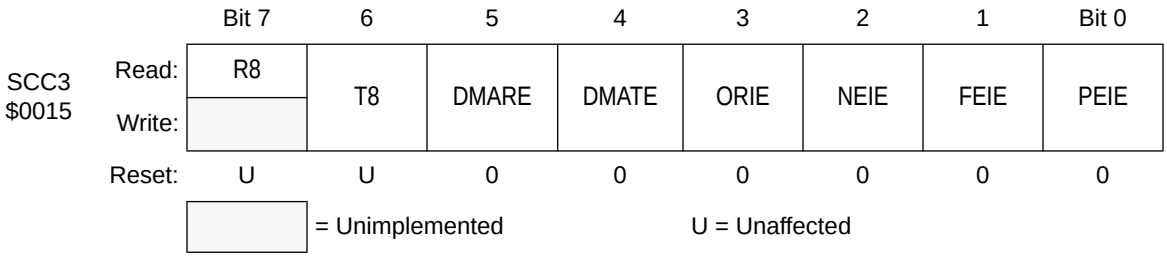


Figure 14-8. SCI Control Register 3 (SCC3)

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 8 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 14.7.4.) 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 14.7.4.) 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 14.7.4.) 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 14.7.4.) 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 14.7.4.) 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 14.7.4.) Reset clears PEIE.

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

14.7.4 SCI Status Register 1 (SCS1)

SCI status register 1 contains flags to signal the following 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

SCS1 $0016	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
Write:								
Reset:	1	1	0	0	0	0	0	0

= Unimplemented

Figure 14-9. 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 ones 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 PTE1/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 zero 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

14.7.5 SCI Status Register 2 (SCS2)

SCI status register 2 contains flags to signal the following conditions:

- Break character detected
- Incoming data

		Bit 7	6	5	4	3	2	1	Bit 0
SCS2 $0017	Read:							BKF	RPF
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-10. 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 PTE1/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 ones again appear on the PTE1/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

14.7.6 SCI Data Register (SCDR)

The SCI data register 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.

		Bit 7	6	5	4	3	2	1	Bit 0
SCDR $0018	Read:	R7	R6	R5	R4	R3	R2	R1	R0
	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	Reset:	Unaffected by reset							

Figure 14-11. SCI Data Register (SCDR)

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

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

14.7.7 SCI Baud Rate Register (SCBR)

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

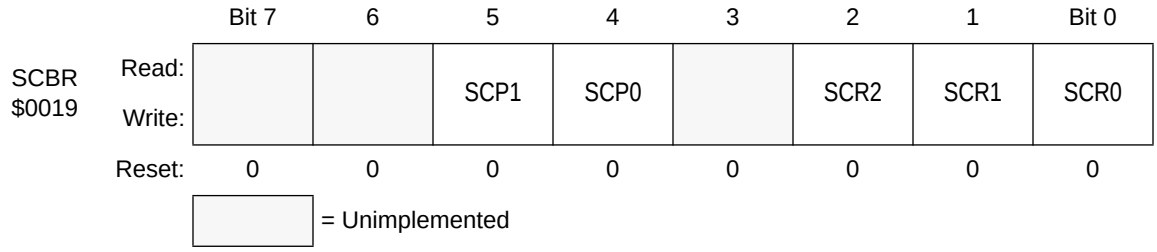


Figure 14-12. SCI Baud Rate Register (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in Table 14-5. Reset clears SCP1 and SCP0.

Table 14-5. SCI Baud Rate Prescaling

SCP1:0	Prescaler Divisor (PD)
00	1
01	3
10	4
11	13

SCR2–SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in Table 14-6. Reset clears SCR2–SCR0.

Table 14-6. SCI Baud Rate Selection

SCR2:1:0	Baud Rate Divisor (BD)
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Use the following formula to calculate the SCI baud rate:

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

where:

CGMXCLK = crystal oscillator frequency
PD = prescaler divisor
BD = baud rate divisor

Table 14-7 shows the SCI baud rates that can be generated with a 4.9152-MHz crystal.

Table 14-7. SCI Baud Rate Selection Examples

SCP1:0	Prescaler Divisor (PD)	SCR2:1:0	Baud Rate Divisor (BD)	Baud Rate (CGMXCLK = 4.9152 MHz)
00	1	000	1	76,800
00	1	001	2	38,400
00	1	010	4	19,200
00	1	011	8	9600
00	1	100	16	4800
00	1	101	32	2400
00	1	110	64	1200
00	1	111	128	600
01	3	000	1	25,600
01	3	001	2	12,800
01	3	010	4	6400
01	3	011	8	3200
01	3	100	16	1600
01	3	101	32	800
01	3	110	64	400
01	3	111	128	200
10	4	000	1	19,200
10	4	001	2	9600
10	4	010	4	4800
10	4	011	8	2400
10	4	100	16	1200
10	4	101	32	600
10	4	110	64	300
10	4	111	128	150
11	13	000	1	5908
11	13	001	2	2954
11	13	010	4	1477
11	13	011	8	739
11	13	100	16	369
11	13	101	32	185
11	13	110	64	92
11	13	111	128	46

SECTION 15
I/O PORTS

15.1 Introduction

Fifty-four bidirectional input-output (I/O) pins form eight 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.

Table 15-1. I/O Port Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Port A Data Register (PTA)	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0	$0000
Port B Data Register (PTB)	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0	$0001
Port C Data Register (PTC)	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0	$0002
Port D Data Register (PTD)	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0	$0003
Data Direction Register A (DDRA)	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0	$0004
Data Direction Register B (DDRB)	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0	$0005
Data Direction Register C (DDRC)	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0	$0006
Data Direction Register D (DDRD)	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0	$0007
Port E Data Register (PTE)	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0	$0008
Port F Data Register (PTF)	0	0	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0	$0009
Port G Data Register (PTG)	0	0	0	0	PTG3	PTG2	PTG1	PTG0	$000A

Table 15-1. I/O Port Register Summary (Continued)

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Port H Data Register (PTH)	0	0	0	0	PTH3	PTH2	PTH1	PTH0	$000B
Data Direction Register E (DDRE)	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0	$000C
Data Direction Register F (DDRF)	0	0	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0	$000D
Data Direction Register G (DDRG)	0	0	0	0	DDRG3	DDRG2	DDRG1	DDRG0	$000E
Data Direction Register H (DDRH)	0	0	0	0	DDRH3	DDRH2	DDRH1	DDRH0	$000F

15.2 Port A

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

15.2.1 Port A Data Register (PTA)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTA $0000	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
	Reset:	Unaffected by reset							

Figure 15-1. 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.

15

15.2.2 Data Direction Register A (DDRA)

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

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

Figure 15-2. Data Direction Register A (DDRA)

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

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

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

NOTE

Avoid glitches on port A pins by writing to the port A data register before changing data direction register A bits from 0 to 1.

Figure 15-3 shows the port A I/O logic.

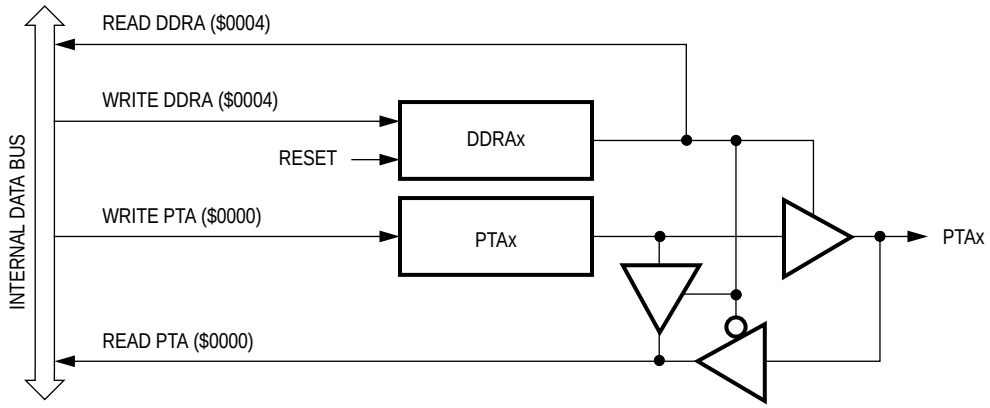


Figure 15-3. Port A I/O Circuit

When bit DDRAx is a logic one, reading address $0000 reads the PTAx data latch. When bit DDRAx is a logic zero, reading address $0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 15-2 summarizes the operation of the port A pins.

15

Table 15-2. Port A Pin Functions

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

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

15.3 Port B

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

15.3.1 Port B Data Register (PTB)

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

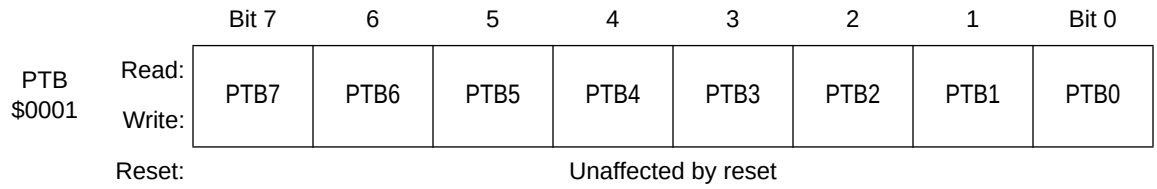


Figure 15-4. Port B Data Register (PTB)

PTB[7:0] — Port B Data Bits

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

15.3.2 Data Direction Register B (DDRB)

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

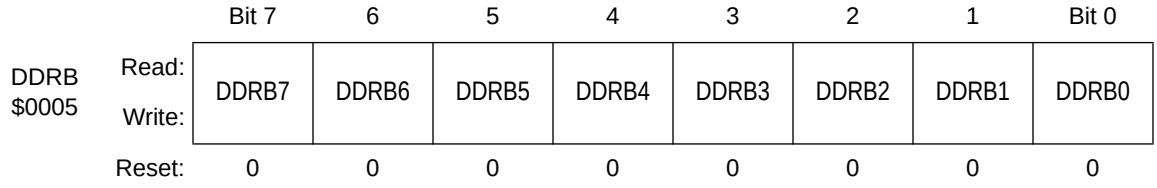


Figure 15-5. Data Direction Register B (DDRB)

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

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

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

NOTE

Avoid glitches on port B pins by writing to the port B data register before changing data direction register B bits from 0 to 1.

Figure 15-6 shows the port B I/O logic.

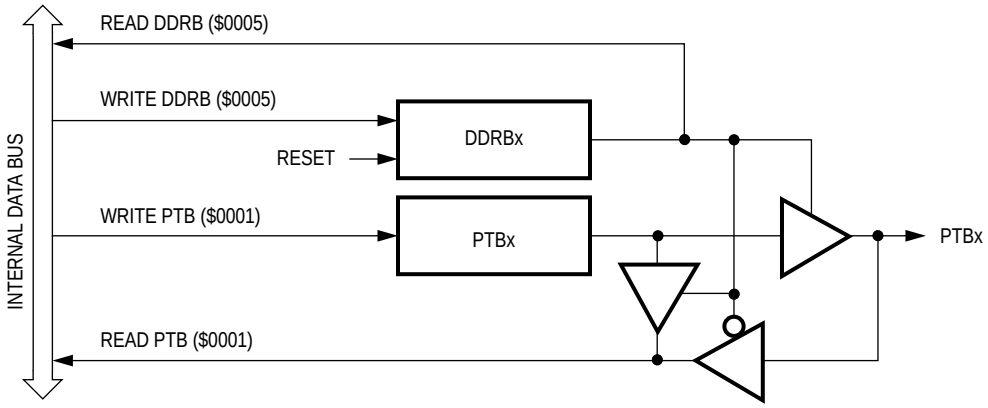


Figure 15-6. Port B I/O Circuit

When bit DDRBx is a logic one, reading address $0001 reads the PTBx data latch. When bit DDRBx is a logic zero, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 15-3 summarizes the operation of the port B pins.

Table 15-3. Port B Pin Functions

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

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

15.4 Port C

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

15.4.1 Port C Data Register (PTC)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTC $0002	Read: Write:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
	Reset:	Unaffected by reset							

Figure 15-7. Port C Data Register (PTC)

PTC[7:0] — Port C Data Bits

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

15.4.2 Data Direction Register C (DDRC)

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

		Bit 7	6	5	4	3	2	1	Bit 0
DDRC $0006	Read: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
	Reset:	0	0	0	0	0	0	0	0

Figure 15-8. Data Direction Register C (DDRC)

DDRC[7:0] — Data Direction Register C Bits

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

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

NOTE

Avoid glitches on port C pins by writing to the port C data register before changing data direction register C bits from 0 to 1.

Figure 15-9 shows the port C I/O logic.

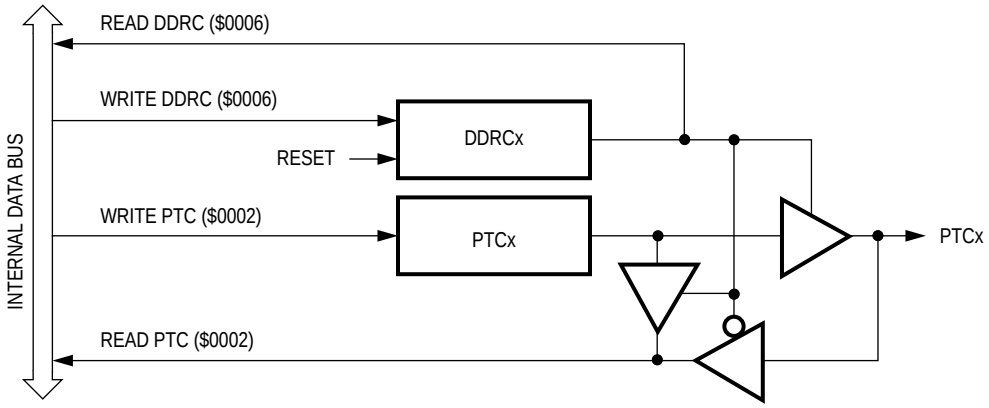


Figure 15-9. Port C I/O Circuit

When bit DDRCx is a logic one, reading address $0002 reads the PTCx data latch. When bit DDRCx is a logic zero, reading address $0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 15-4 summarizes the operation of the port C pins.

Table 15-4. Port C Pin Functions

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

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

15.5 Port D

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

15.5.1 Port D Data Register (PTD)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTD $0003	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
	Reset:	Unaffected by reset							

Figure 15-10. 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.

The keyboard interrupt enable bits, KBIE[7:0], in the keyboard interrupt control register (KBICR), enable the port D pins as external interrupt pins. (See **SECTION 17 EXTERNAL INTERRUPT MODULE (IRQ)**.)

15.5.2 Data Direction Register D (DDRD)

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

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

Figure 15-11. Data Direction Register D (DDRD)

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

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

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

NOTE

Avoid glitches on port D pins by writing to the port D data register before changing data direction register D bits from 0 to 1.

Figure 15-12 shows the port D I/O logic.

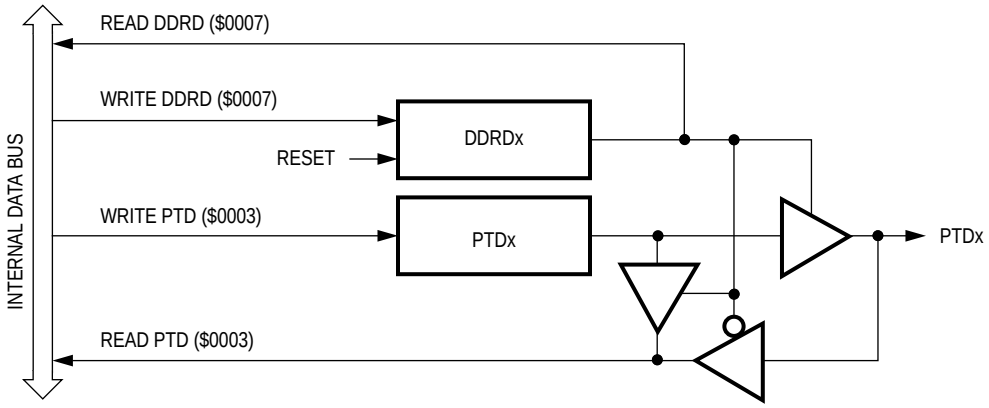


Figure 15-12. Port D I/O Circuit

When bit DDRDx is a logic one, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic zero, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 15-5 summarizes the operation of the port D pins.

Table 15-5. Port D Pin Functions

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

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

15.6 Port E

Port E is an 8-bit special function port that shares five of its pins with the timer interface module (TIM) and two of its pins with the serial communications interface module (SCI).

15.6.1 Port E Data Register (PTE)

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

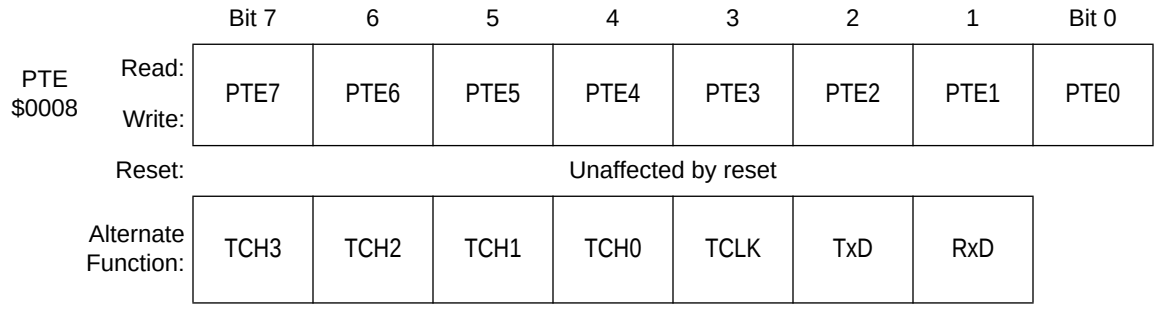


Figure 15-13. Port E Data Register (PTE)

PTE[7:0] — Port E Data Bits

PTE[7:0] are read/write, software programmable bits. Data direction of each port E pin is under the control of the corresponding bit in data direction register E.

TCH[3:0] — Timer Channel I/O Bits

The PTE7/TCH3–PTE4/TCH0 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTE7/TCH3–PTE4/TCH0 pins are timer channel I/O pins or general-purpose I/O pins. (See **12.8.5 TIM Channel Status and Control Registers (TSC0–TSC3)**.)

15

NOTE

Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the TIM. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See **Table 15-6. Port E Pin Functions**.)

TCLK — Timer Clock Input

The PTE3/TCLK pin is the external clock input for the TIM. The prescaler select bits, PS[2:0], select PTE3/TCLK as the TIM clock input. (See **12.8.1 TIM Status and Control Register (TSC)**.) When not selected as the TIM clock, PTE3/TCLK is available for general-purpose I/O.

TxD — SCI Transmit Data Output

The PTE2/TxD pin is the transmit data output for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE2/TxD pin is available for general-purpose I/O. (See **14.7.1 SCI Control Register 1 (SCC1)**.)

NOTE

Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SCI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See **Table 15-6. Port E Pin Functions**.)

RxD — SCI Receive Data Input

The PTE1/RxD pin is the receive data input for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE1/RxD pin is available for general-purpose I/O. (See **14.7.1 SCI Control Register 1 (SCC1)**.)

15.6.2 Data Direction Register E (DDRE)

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

		Bit 7	6	5	4	3	2	1	Bit 0
DDRE $000C	Read:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

Figure 15-14. Data Direction Register E (DDRE)

DDRE[7:0] — Data Direction Register E Bits

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

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

NOTE

Avoid glitches on port E pins by writing to the port E data register before changing data direction register E bits from 0 to 1.

Figure 15-15 shows the port E I/O logic.

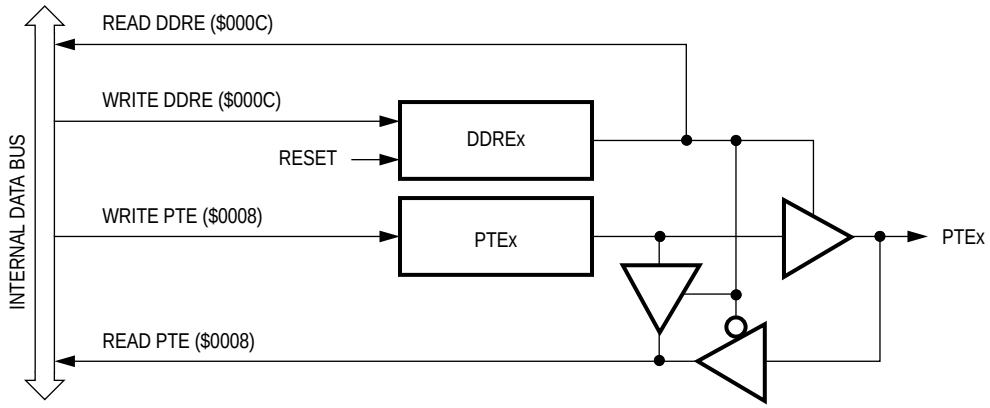


Figure 15-15. Port E I/O Circuit

When bit DDREx is a logic one, reading address $0008 reads the PTEx data latch. When bit DDREx is a logic zero, reading address $0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 15-6 summarizes the operation of the port E pins.

Table 15-6. Port E Pin Functions

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

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

15.7 Port F

Port F is a 6-bit special function port that shares four of its pins with the serial peripheral interface module (SPI).

15.7.1 Port F Data Register (PTF)

The port F data register contains a data latch for each of the six port F pins.

		Bit 7	6	5	4	3	2	1	Bit 0
PTF $0009	Read:	0	0	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
	Write:								
	Reset:	Unaffected by reset							
	Alternate Function:					MISO	MOSI	SPSCK	$\overline{SS}$

= Unimplemented

Figure 15-16. Port F Data Register (PTF)

PTF[5:0] — 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. Reset has no effect on PTF[5:0].

MISO — Master In/Slave Out

The PTF3/MISO pin is the master in/slave out terminal of the SPI module. When the SPI enable bit, SPE, is clear, the SPI module is disabled, and the PTF3/MISO pin is available for general-purpose I/O. (See **13.8.1 SPI Control Register (SPCR)**.)

NOTE

Data direction register F (DDRF) does not affect the data direction of port F pins that are being used by the SPI module. However, the DDRF bits always determine whether reading port F returns the states of the latches or the states of the pins. (See **Table 15-7. Port F Pin Functions**.)

MOSI — Master Out/Slave In

The PTF2/MOSI pin is the master out/slave in terminal of the SPI module. When the SPE bit is clear, the PTF2/MOSI pin is available for general-purpose I/O. (See **13.8.1 SPI Control Register (SPCR)**.)

SPSCK — SPI Serial Clock

The PTF1/SPSCK pin is the serial clock input of the SPI module. When the SPE bit is clear, the PTF1/SPSCK pin is available for general-purpose I/O.

$\overline{SS}$ — Slave Select

The PTF0/$\overline{SS}$ pin is the slave select input of the SPI module. When the SPE bit is clear, or when the SPI master bit, SPMSTR, is set, the PTF0/$\overline{SS}$ pin is available for general-purpose I/O. (See **13.8.1 SPI Control Register (SPCR)**.) When the SPI is enabled, the DDRF0 bit in data direction register F (DDRF) has no effect on the PTF0/$\overline{SS}$ pin.

15.7.2 Data Direction Register F (DDRF)

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

DDRF $000D	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
Write:								
Reset:			0	0	0	0	0	0

= Unimplemented

Figure 15-17. Data Direction Register F (DDRF)

DDRF[5:0] — Data Direction Register F Bits

These read/write bits control port F data direction. Reset clears DDRF[5:0], 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 15-18 shows the port F I/O logic.

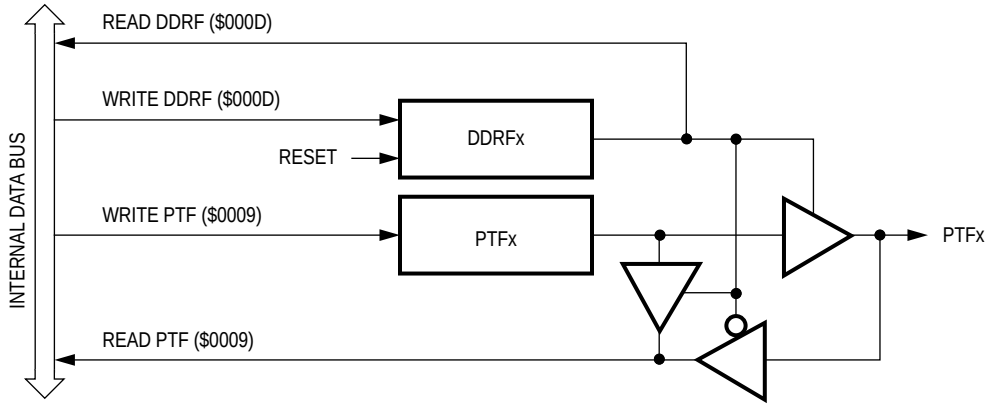


Figure 15-18. Port F I/O Circuit

When bit DDRFx is a logic one, reading address $0009 reads the PTFx data latch. When bit DDRFx is a logic zero, reading address $0009 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 15-7 summarizes the operation of the port F pins.

Table 15-7. 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)	DDRF[5:0]	Pin	PTF[5:0](3)
1	X	Output	DDRF[5:0]	PTF[5:0]	PTF[5:0]

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

15.8 Port G

Port G is a 4-bit general-purpose bidirectional I/O port.

NOTE

Port G is available only on the 64-pin QFP.

15.8.1 Port G Data Register (PTG)

The port G data register contains a data latch for each of the four port G pins.

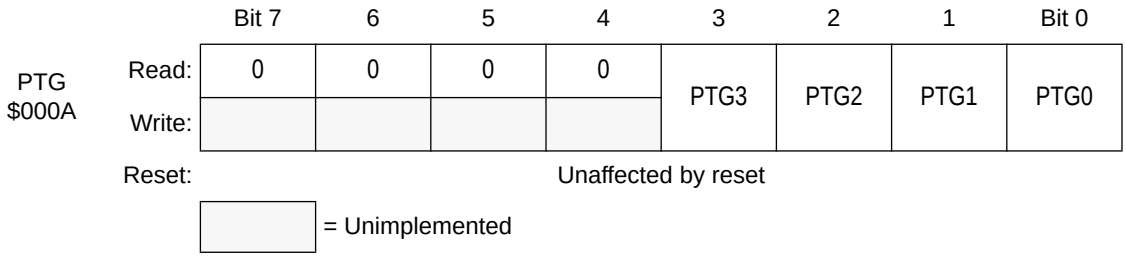


Figure 15-19. Port G Data Register (PTG)

PTG[3:0] — Port G Data Bits

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

15.8.2 Data Direction Register G (DDRG)

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

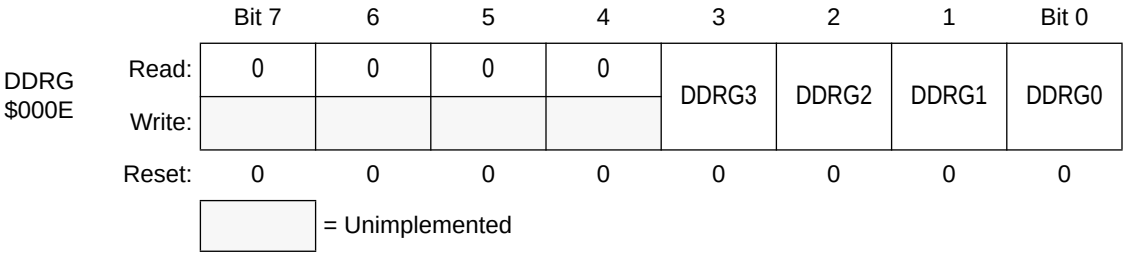


Figure 15-20. Data Direction Register G (DDRG)

DDRG[3:0] — Data Direction Register G Bits

These read/write bits control port G data direction. Reset clears DDRG[3:0], 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 15-21 shows the port G I/O logic.

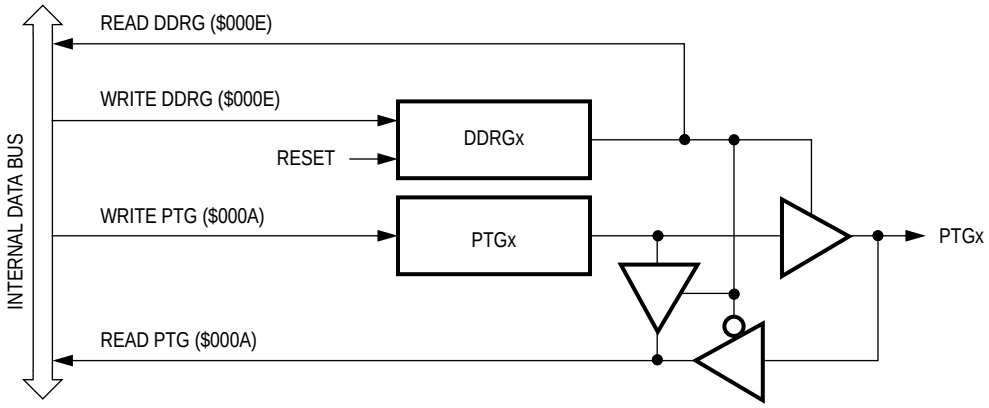


Figure 15-21. Port G I/O Circuit

When bit DDRGx is a logic one, reading address $000A reads the PTGx data latch. When bit DDRGx is a logic zero, reading address $000A reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 15-8 summarizes the operation of the port G pins.

Table 15-8. 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)	DDRG[3:0]	Pin	PTG[3:0](3)
1	X	Output	DDRG[3:0]	PTG[3:0]	PTG[3:0]

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

15.9 Port H

Port H is a 4-bit general-purpose bidirectional I/O port.

NOTE

Port H is available only on the 64-pin QFP.

15.9.1 Port H Data Register (PTH)

The port H data register contains a latch for each of the four port H pins.

		Bit 7	6	5	4	3	2	1	Bit 0
PTH $000B	Read:	0	0	0	0	PTH3	PTH2	PTH1	PTH0
	Write:								
	Reset:	Unaffected by reset							

= Unimplemented

Figure 15-22. Port H Data Register (PTH)

PTH[3:0] — Port H Data Bits

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

15.9.2 Data Direction Register H (DDRH)

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

		Bit 7	6	5	4	3	2	1	Bit 0
DDRH $000F	Read:	0	0	0	0	DDRH3	DDRH2	DDRH1	DDRH0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 15-23. Data Direction Register H (DDRH)

DDRH[3:0] — Data Direction Register H Bits

These read/write bits control port H data direction. Reset clears DDR[3:0], configuring all port H pins as inputs.

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

NOTE

Avoid glitches on port H pins by writing to the port H data register before changing the data direction register H bits from 0 to 1.

Figure 15-24 shows the port H I/O logic.

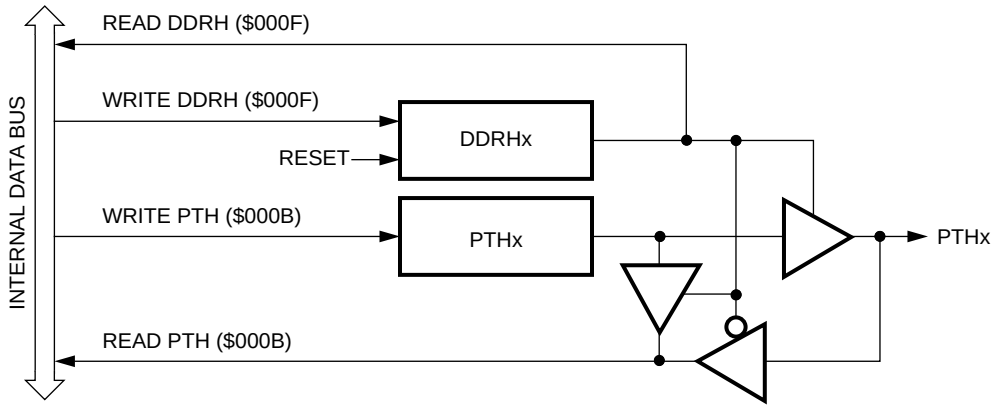


Figure 15-24. Port H I/O Circuit

When bit DDRHx is a logic one, reading address $000B reads the PTHx data latch. When bit DDRHx is a logic zero, reading address $000B reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 15-9 summarizes the operation of the port H pins.

Table 15-9. Port H Pin Functions

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

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

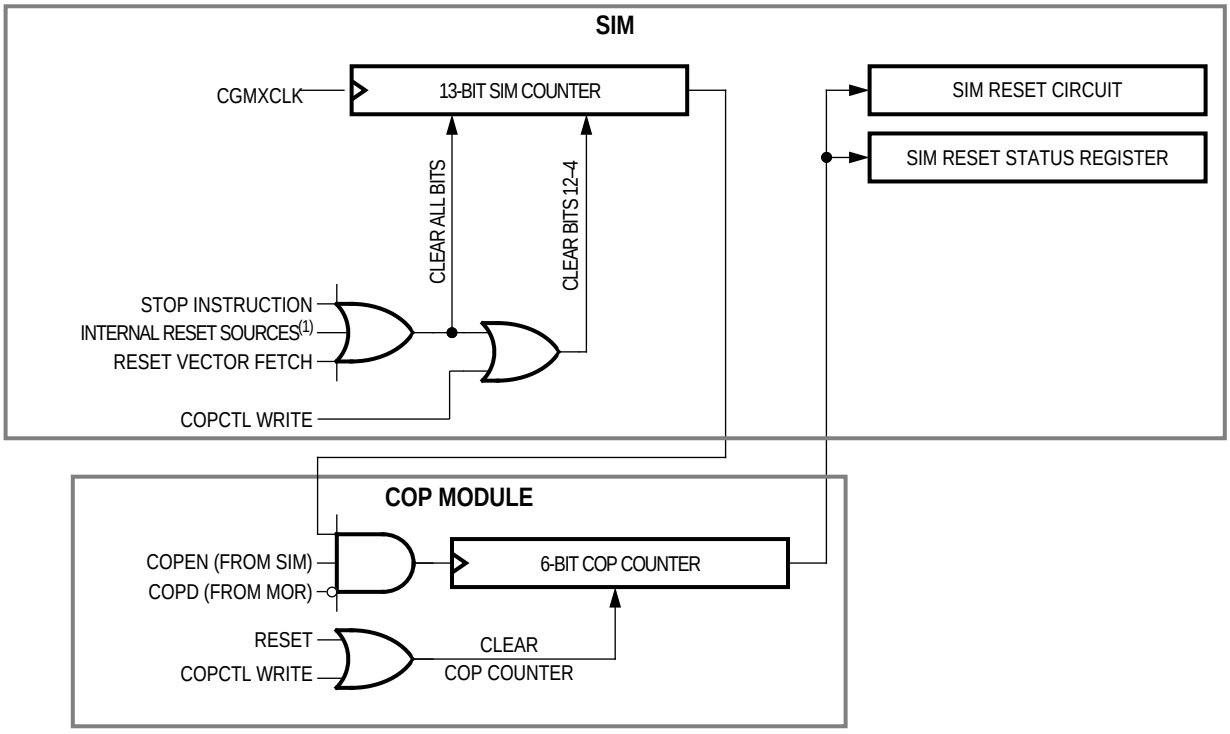
SECTION 16
COMPUTER OPERATING PROPERLY MODULE

16.1 Introduction

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

16.2 Functional Description

Figure 16-1 shows the structure of the COP module.



NOTE:
1. See **7.3.2 Active Resets from Internal Sources**.

Figure 16-1. COP Block Diagram

The COP counter is a free-running 6-bit counter preceded by the 13-bit system integration module (SIM) counter. If not cleared by software, the COP counter

Table 16-1. COP I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
COP Control Register (COPCTL)									$FFFF

overflows and generates an asynchronous reset after $2^{18} – 2^4$ CGMXCLK cycles. With a 4.9152-MHz crystal, the COP timeout period is 53.3 ms. 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) (see **7.7.2 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.

16.3 I/O Signals

The following paragraphs describe the signals shown in Figure 16-1.

16.3.1 CGMXCLK

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

16

16.3.2 STOP Instruction

The STOP instruction clears the SIM counter.

16.3.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see 16.4) clears the COP counter and clears bits 12 through 4 of the SIM counter. Reading the COP control register returns the reset vector.

16.3.4 Power-On Reset

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

16.3.5 Internal Reset

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

16.3.6 Reset Vector Fetch

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

16.3.7 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the mask option register (MOR). (See **SECTION 5 MASK OPTION REGISTER (MOR)**.)

16.4 COP Control Register (COPCTL)

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

		Bit 7	6	5	4	3	2	1	Bit 0
COPCTL $FFFF	Read:	Low byte of reset vector							
	Write:	Clear COP counter							
	Reset:	Unaffected by reset							

Figure 16-2. COP Control Register (COPCTL)

16.5 Interrupts

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

16.6 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.

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 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.

16.7.2 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 bit to logic zero.

16.8 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 17
EXTERNAL INTERRUPT MODULE (IRQ)

17.1 Introduction

This section describes the external interrupt module (IRQEPM, Version C), which supports external interrupt functions.

17.2 Features

Features of the IRQ module include the following:

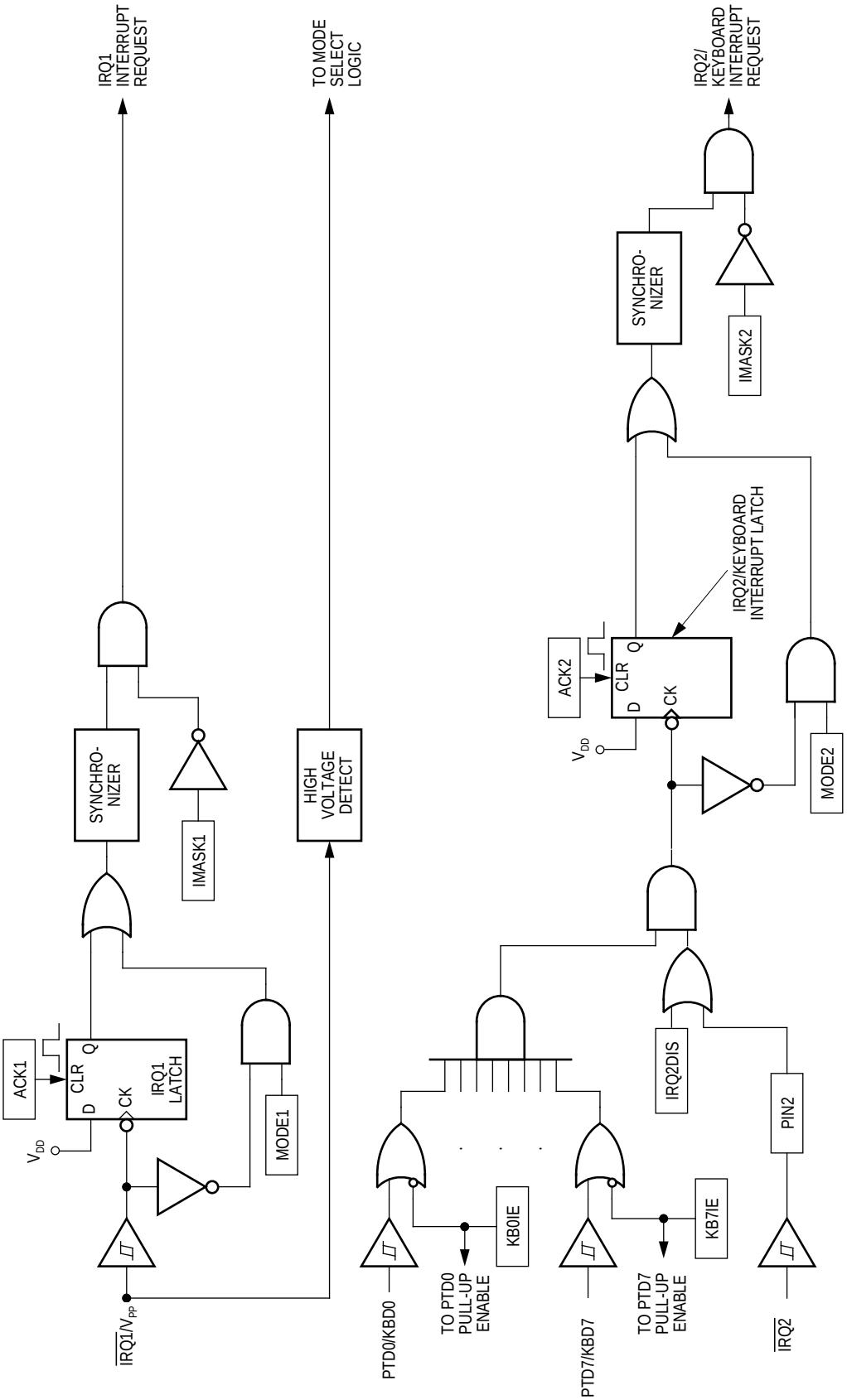
- Two Dedicated External Interrupt Pins ($\overline{IRQ1}/V_{PP}$ and $\overline{IRQ2}$)
- Separate IRQ1 and IRQ2/Keyboard Interrupt Masks
- Individually Enabled Keyboard Interrupt Pins
- IRQ2 Interrupt Disable
- Hysteresis Buffers

17.3 Functional Description

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

Interrupt signals on the $\overline{IRQ1}/V_{PP}$ pin are latched into the IRQ1 latch. Interrupt signals on the $\overline{IRQ2}$ pin and on the keyboard interrupt pins are latched into the IRQ2/keyboard interrupt latch. An interrupt latch remains set until one of the following actions occurs:

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the latch that caused the vector fetch.
- Software clear — Software can clear an interrupt latch by writing to the appropriate acknowledge bit in the interrupt status and control register (ISCR). Writing a logic one to the ACK1 bit clears the IRQ1 latch. Writing a logic one to the ACK2 bit clears the IRQ2/keyboard interrupt latch.
- Reset — A reset automatically clears both interrupt latches.



NOTE: IRQ2DIS does not prevent the reading of the state of the $\overline{IRQ2}$ pin in the IRQ status and control register (ISCR).

Figure 17-1. IRQ Module Block Diagram

Table 17-1. IRQ I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
IRQ Status/Control Register (ISCR)	PIN2	ACK2	IMASK2	MODE2	IRQ2DIS	ACK1	IMASK1	MODE1	$001A
Keyboard Interrupt Control Register (KBICR)	KB7IE	KB6IE	KB5IE	KB4IE	KB3IE	KB2IE	KB1IE	KB0IE	$001B

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 $\overline{IRQ1}/V_{PP}$ pin. The MODE2 bit controls the triggering sensitivity of the $\overline{IRQ2}$ pin and the keyboard interrupt pins.

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 the following occur:

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

The vector fetch or software clear may occur before or after the interrupt pin returns to logic one. As long as the pin is low, the interrupt request remains pending.

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.

The KBxIE bits in the keyboard interrupt control register (KBICR) individually enable interrupt requests from the keyboard interrupt pins. The IRQ2DIS bit in the interrupt status and control register (ISCR) disables interrupt requests from the $\overline{IRQ2}$ pin. The KBxIE and IRQ2DIS bits allow the user to control which pins can latch interrupt requests into the IRQ2/keyboard interrupt latch.

NOTE

The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. (See **Figure 17-2. IRQ Interrupt Flowchart**.)

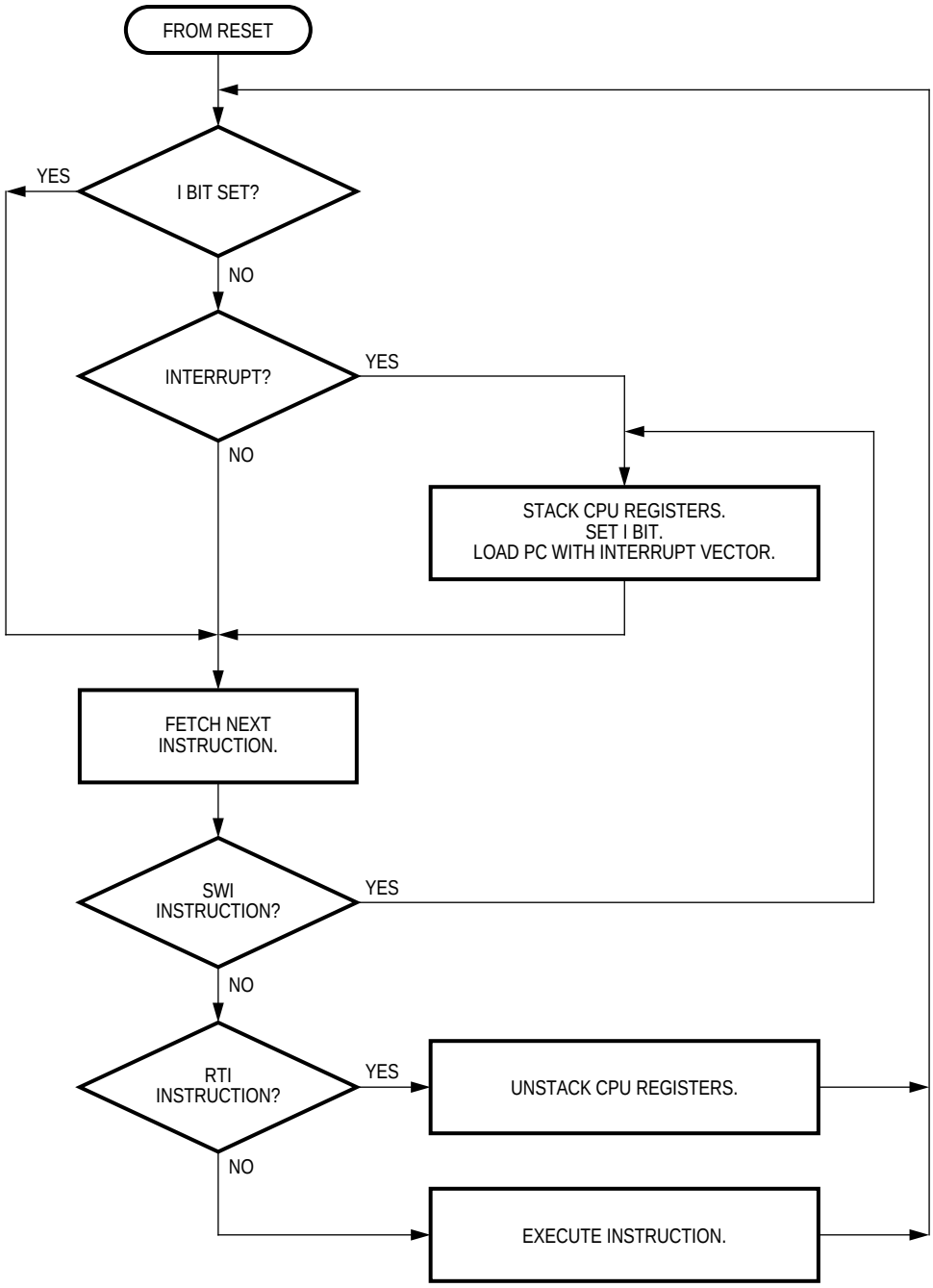


Figure 17-2. IRQ Interrupt Flowchart

17.3.1 $\overline{\text{IRQ1}}/V_{PP}$ Pin

A logic zero on the $\overline{\text{IRQ1}}/V_{PP}$ 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 $\overline{\text{IRQ1}}/V_{PP}$ pin is both falling-edge-sensitive and low-level-sensitive. With MODE1 set, both of the following 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 one to the ACK1 bit in the interrupt status and control register (ISCR). The ACK1 bit is useful in applications that poll the $\overline{\text{IRQ1}}/V_{PP}$ 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 $\overline{\text{IRQ1}}/V_{PP}$ 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 $\overline{\text{IRQ1}}/V_{PP}$ pin to logic one — As long as the $\overline{\text{IRQ1}}/V_{PP}$ pin is at logic zero, the IRQ1 latch remains set.

The vector fetch or software clear and the return of the $\overline{\text{IRQ1}}/V_{PP}$ pin to logic one may occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ1}}/V_{PP}$ pin is at logic zero.

If the MODE1 bit is clear, the $\overline{\text{IRQ1}}/V_{PP}$ 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 $\overline{\text{IRQ1}}/V_{PP}$ pin.

NOTE

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

17.3.2 $\overline{\text{IRQ2}}$ Pin

When the IRQ2DIS bit is clear, a logic zero on the $\overline{\text{IRQ2}}$ pin can latch an interrupt request into the IRQ2/keyboard interrupt latch. A vector fetch, software clear, or reset clears the IRQ2/keyboard interrupt latch.

If the MODE2 bit is set, the $\overline{\text{IRQ2}}$ pin is both falling-edge-sensitive and low-level-sensitive. With MODE2 set, both of the following actions must occur to clear the IRQ2/keyboard 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 one to the ACK2 bit in the interrupt status and control register (ISCR). The ACK2 bit is useful in applications that poll the $\overline{\text{IRQ2}}$ pin and require software to clear the IRQ2/keyboard 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 $FFE0 and $FFE1.
- Return of the $\overline{\text{IRQ2}}$ pin to logic one — As long as the $\overline{\text{IRQ2}}$ pin is at logic zero, the IRQ2/keyboard interrupt latch remains set.

The vector fetch or software clear and the return of the $\overline{\text{IRQ2}}$ pin to logic one may occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ2}}$ pin is at logic zero.

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/keyboard interrupt latch.

To determine the logic level on the $\overline{\text{IRQ2}}$ pin, read the $\overline{\text{IRQ2}}$ pin state bit, PIN2, in the ISCR. This bit reflects the value of the $\overline{\text{IRQ2}}$ pin, even when the $\overline{\text{IRQ2DIS}}$ bit is set.

NOTE

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

17.3.3 Keyboard Interrupt Pins

A logic zero on a keyboard interrupt pin latches an interrupt request into the IRQ2/keyboard interrupt latch. The keyboard interrupt control register (KBICR) contains an interrupt enable bit for each keyboard interrupt pin. Setting a KBxIE bit configures the corresponding port pin as an external interrupt pin and enables the pin's pull-up device.

If the MODE2 bit is set, the keyboard interrupt pin is both falling-edge-sensitive and low-level-sensitive. With MODE2 set, both of the following actions must occur to clear the IRQ2/keyboard 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 one to the ACK2 bit in the interrupt status and control register (ISCR). The ACK2 bit is useful in applications that poll the keyboard interrupt pin and require software to clear the IRQ2/keyboard interrupt latch. Writing to the ACK2 bit can also prevent spurious interrupts due to noise. Setting ACK2 does not affect subsequent transitions on the keyboard interrupt 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 $FFE0 and $FFE1.
- Return of the keyboard interrupt pin to logic one — As long as the keyboard interrupt pin is at logic zero, the IRQ2/keyboard interrupt latch remains set.

The vector fetch or software clear and the return of the keyboard interrupt pin to logic one may occur in any order. The interrupt request remains pending as long as the keyboard interrupt pin is at logic zero.

If the MODE2 bit is clear, the keyboard interrupt pin is falling-edge-sensitive only. With MODE2 clear, a vector fetch or software clear immediately clears the IRQ2/keyboard interrupt latch.

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 KBxIE bit forces the corresponding keyboard interrupt pin to be an input, overriding the data direction register. However, the data direction register bit must be a logic zero for software to read the pin.

NOTE

Holding the $\overline{IRQ2}$ pin low with $\overline{IRQ2}$ pin interrupts enabled (IRQ2DIS = 0) prevents falling edges on the keyboard interrupt pins from being detected.

Holding a keyboard interrupt pin low with $\overline{IRQ2}$ pin interrupts enabled prevents falling edges on the $\overline{IRQ2}$ pin from being detected. (See **Figure 17-1. IRQ Module Block Diagram**.)

17.4 IRQ Module During Break Interrupts

The system integration module (SIM) controls whether the IRQ1 and IRQ2/keyboard 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 **7.7.3 SIM Break Flag Control Register (SBFCR)**.)

To allow software to clear the IRQ1 latch and the IRQ2/keyboard interrupt latch during a break interrupt, write a logic one to the BCFE bit. If a latch is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect the latches during the break state, write a logic zero to the BCFE bit. With BCFE at logic zero (its default state), writing to the ACK1 and ACK2 bits in the IRQ status and control register during the break state has no effect on the IRQ latches. (See **17.5.1 IRQ Status and Control Register (ISCR)**.)

17.5 I/O Registers

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

- IRQ status and control register (ISCR)
- Keyboard interrupt control register (KBICR)

17.5.1 IRQ Status and Control Register (ISCR)

The IRQ status and control register has the following functions:

- Shows current state of $\overline{\text{IRQ2}}$ pin
- Clears the IRQ1 and IRQ2/keyboard interrupt latches
- Masks IRQ1 and IRQ2/keyboard interrupt requests
- Controls triggering sensitivity of the $\overline{\text{IRQ1}}/V_{PP}$, $\overline{\text{IRQ2}}$, and keyboard interrupt pins
- Disables IRQ2/keyboard interrupt request

		Bit 7	6	5	4	3	2	1	Bit 0
ISCR $001A	Read:	PIN2	0	IMASK2	MODE2	IRQ2DIS	0	IMASK1	MODE1
	Write:		ACK2				ACK1		
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-3. IRQ Status and Control Register (ISCR)

PIN2 — $\overline{\text{IRQ2}}$ Pin State Bit

This read-only bit reflects the current logic level of the $\overline{\text{IRQ2}}$ pin.

1 = $\overline{\text{IRQ2}}$ pin at logic one
0 = $\overline{\text{IRQ2}}$ pin at logic zero

ACK2 — IRQ2/Keyboard Interrupt Request Acknowledge Bit

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

IMASK2 — IRQ2/Keyboard Interrupt Mask Bit

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

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

MODE2 — IRQ2/Keyboard Interrupt Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ2}}$ and keyboard interrupt pins. Reset clears MODE2.

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

IRQ2DIS — $\overline{\text{IRQ2}}$ Pin Interrupt Latch Disable Bit

This read/write bit prevents the $\overline{\text{IRQ2}}$ pin from latching interrupt requests into the IRQ2/keyboard interrupt latch. Reset clears IRQ2DIS.

1 = $\overline{\text{IRQ2}}$ pin interrupt requests not latched
0 = $\overline{\text{IRQ2}}$ pin interrupt requests latched

ACK1 — IRQ1 Interrupt Request Acknowledge Bit

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

IMASK1 — IRQ1 Interrupt Mask Bit

Writing a logic one to this read/write bit disables IRQ1 interrupt requests. Reset clears IMASK1.

1 = IRQ1 interrupt requests disabled
0 = IRQ1 interrupt requests enabled

MODE1 — IRQ1 Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ1}}/V_{PP}$ pin. Reset clears MODE1.

1 = $\overline{\text{IRQ1}}/V_{PP}$ interrupt requests on falling edges and low levels
0 = $\overline{\text{IRQ1}}/V_{PP}$ interrupt requests on falling edges only

17.5.2 Keyboard Interrupt Control Register (KBICR)

The keyboard interrupt control register enables keyboard interrupts and enables the keyboard pin pull-up devices.

		Bit 7	6	5	4	3	2	1	Bit 0
KBICR $001B	Read: / Write:	KB7IE	KB6IE	KB5IE	KB4IE	KB3IE	KB2IE	KB1IE	KB0IE
	Reset:	0	0	0	0	0	0	0	0

Figure 17-4. Keyboard Interrupt Control Register (KBICR)

KB7IE–KB0IE — 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 control register.

1 = Corresponding keyboard interrupt pin enabled and pull-up device on
0 = Corresponding keyboard interrupt pin disabled and pull-up device off

17.6 Keypad Interrupt Code Example

The next three subsections contain keypad interrupt subroutines that perform the following functions:

- Time delay
- 4 × 4 keypad matrix decoding
- Keypad exit from stop mode and keypress decoding

17.6.1 Time Delay

This subroutine produces an accurate time delay for a whole number of milliseconds from 1 to 256. The number of milliseconds to delay is passed in the accumulator (A). The smallest delay is 8030 bus cycles when A = 1 (1 ms). The largest delay is 2,048,030 bus cycles when A = 0 (256 ms). These delay times are based on an 8-MHz bus speed.

NOTE

Passing 0 results in a 256 ms delay, not a zero delay.

The subroutine is formed by two loops. The inner loop (Delayms020) executes in 7995 cycles. The outer loop executes once for each millisecond and adds five bus cycles each time through the loop, using 8000 cycles for each millisecond of delay. The JSR, RTS, and stacking instructions add 30 bus cycles to the total time. The exact number of cycles for this subroutine to execute can be calculated with the following equation:

$$\text{Number of cycles} = 30 + (A \times 8000)$$

where:
A = the value in the accumulator

Upon exit from the routine, all registers return to their previous values. I/O registers are unaffected.

```
;Assembled under:            IASM08 Ver 3.02
;Entry Label:                Delayms_Body
;Module size:                20 bytes
;Stack space used:           4 bytes
;RAM used:                   0 bytes
;Worst case execution:       2,048,030 Cycles
;Entry conditions:           # of ms to delay in accumulator
;Part resources needed:      CPU08 running at 8 MHz bus frequency
;External variables used:    None
;Subroutines used:           None
;Number of exit points:      1
;Exit label:                 Delayms015
;Exit conditions:            Registers left as they were when routine was
;                            called


Delayms_Body
              psha             ;Stack A.
              phsx             ;Stack X.
              pshh             ;Stack H.
              tax              ;Save A in X.
              tpa              ;Put CCR into A.
              psha             ;Stack CCR.
              txa              ;Restore A.
Delayms010

Delayms020    ldhx  #$0378     ;Load delay into H:X.
Delayms030    aix   #-1        ;Decrement delay.
              nop              ;Burn 2 bus cycles.
              cphx  #0         ;Done yet?
              bne   Delayms030 ;Branch if not done.
              nop              ;Burn 1 bus cycle.
              nop              ;Burn 1 bus cycle.
              dbnza Delayms010 ;Decrement number of ms to delay.
                               ;Branch if not done.
              pula             ;Unstack CCR.
              tap              ;Restore CCR.
              pulh             ;Unstack H.
              pulx             ;Unstack X.
              pula             ;Unstack A.
Delayms015    rts              ;Done. Return to calling routine.
```

17.6.2 Keypad Matrix Decoding Subroutine

This subroutine decodes a 4 × 4 matrix keypad connected to port B. Features of this software include the following:

- Optimization for low power — When not decoding, outputs are high and drawing no pull-up current.
- Protection from short circuits due to simultaneous key presses — No two outputs are ever in opposite states.
- Single keypress decoding only — A multiple keypress gives an undefined but predictable response.

Pins PTD7/KBD7–PTD4/KBD4 are inputs and should have pull-up resistors. Pins PTD3/KBD3–PTD0/KBD0 are outputs.

On return, the accumulator holds the ASCII value of the decoded key. The value is user-defined by changing the value in the table at the end of the code. If no key is decoded, the accumulator holds $00.

```
;Flow
;              Point to top of table.
;       KeyPad010:
;              Precharge pullups.
;              Get key value from table.
;              Complement and mask for DDR.
;              Get key value again.
;              Write value to port.
;              Get value from port.
;              Compare to table.
;              Is this the right key?
;                      Yes:
;                            Precharge pullups.
;                            Get ASCII code for key.
;                            Return.
;                      No:
;                            Decrement table pointer.
;                            Is this the bottom of the table?
;                                                  Yes:
;                                                  Precharge pullups.
;                                                  Load null into A.
;                                                  Return.
;                                                  No:
;                                                  Branch to KeyPad010.
```

```
;Assembled under                 IASM08 Ver 3.02
;Entry Label:                    KeyPad_Body
;Module size:                    96 bytes
;Stack space used:               0 bytes
;RAM used:                       0 bytes
;Worst case execution:           787 Cycles
;Entry conditions:               None
;Part resources needed:          8-bit I/O port B
;External variables used:        None
;Subroutines used:               None
;Number of exit points:          2

;       Exit label:             Keypad020
;       Exit conditions:        Null character in A — no key decoded

;       Exit label:             Keypad040
;       Exit conditions:        ASCII value in A — key decoded


KeyPad_Body                             ;Load X with the offset of the last
                                        ;entry in the table.
            ldx  #{KeyPad_Table_Top — KeyPad_Table}
            lda  #$0f           ;Help the pullups by driving the lines
            sta  DDRD           ;high. This allows quick decoding with
            sta  PTD            ;weak pullups.
            lda  KeyPad_Table,x ;Get decode value from table.
            coma                ;Complement to get DDR correct in
                                ;positive logic.
            and  #$0f           ;Save only low nibble for DDR.
            sta  DDRD           ;Write to DDR to drive only one column.
            lda  KeyPad_Table,x ;Get key decode value again.
            sta  PTD            ;Write to port.
            lda  PTD            ;Read from port.
            and  #f0            ;Throw out columns to read only rows.
            cmp  KeyPad_Table + 1,x;See if high nibble bit was pulled low.
            beq  KeyPad030      ;If key found, branch.
            decx                ;Decrement X three times to point to
            decx                ;next value in table.
            decx
            bpl  KeyPad010      ;If not below bottom of table, try
                                ;again.
            lda  #$0f           ;Help pullups by driving lines high.
            sta  DDRD           ;This minimizes current draw.
            sta  PTD
            lda  #$00           ;Key was not decoded, so
```

```
KeyPad020       rts                     ;return with null character.
KeyPad030       lda  #$0f               ;Help pullups by driving lines high.
                sta  DDRD               ;This minimizes current draw.
                sta  PTD
                lda  KeyPad_Table + 2,x;Load ASCII into A.
KeyPad040       rts                     ;Return with ASCII value in A.
                    ;Table of keypad decode values and ASCII equivalents. Fill
                    ;in your own ASCII equivalents or other coding scheme.
                    ;Codes must be 1 byte each.
                                        ;Row    Column
KeyPad_Table    DB   $0E, $E0, '*'      ;PD4    PD0
                DB   $0D, $E0, '0'      ;PD4    PD1
                DB   $0B, $E0, '#'      ;PD4    PD2
                DB   $07, $E0, 'D'      ;PD4    PD3
                DB   $0E, $D0, '7'      ;PD5    PD0
                DB   $0D, $D0, '8'      ;PD5    PD1
                DB   $0B, $D0, '9'      ;PD5    PD2
                DB   $07, $D0, 'C'      ;PD5    PD3
                DB   $0E, $B0, '4'      ;PD6    PD0
                DB   $0D, $B0, '5'      ;PD6    PD1
                DB   $0B, $B0, '6'      ;PD6    PD2
                DB   $07, $B0, 'B'      ;PD6    PD3
                DB   $0E, $70, '1'      ;PD7    PD0
                DB   $0D, $70, '2'      ;PD7    PD1
                DB   $0B, $70, '3'      ;PD7    PD2
KeyPad_Table_Top DB  $07, $70, 'A'      ;PD7    PD3
```

17.6.3 Keypad Interrupt Subroutine

This subroutine, KeyPdInt, uses keypad interrupts to exit from stop mode, then decodes the keypress with the KeyPad subroutine. The KeyPad subroutine decodes one key at a time and protects against short circuits due to multiple keypresses.

KeyPdInt also uses Delayms as a debouncing and repeat delay subroutine. (See **17.6.1 Time Delay**.)

Features of KeyPdInt include the following:

- Low-Power Consumption
  - Executes STOP Instruction when not Decoding Keypress
  - Pull-Up Current = 0 during Debounce Delay
  - Protection from Short Circuits due to Simultaneous Keypresses
- Interrupt-Driven Operation
- Software Debouncing
- Key Repeat
- Internal Pull-Ups

```
; Flow
;     Reset Call init routine to set up port D DDR and Option Register.
;           Clear I bit to allow keypad interrupts.
;           STOP to remain in low power mode when key is not pressed.
;           Loop to STOP instruction after returning from interrupt.
;     ISR   Call KeyPad routine to see if a key is down. Just return if it
;           was a gho
;           st.
;           If key was there, debounce keypad with Delayms routine.
;           If no key was there, just return.
;           If key was there, perform action based on value returned by
;           KeyPad routine.
;           Delay 200 ms for repeat delay, and then branch to beginning of
;           ISR to see if the key is still bing pressed.
;           Delay to debounce the release of the key. RTI to return to main
;           loop.
```

```
     ;Assembled under:               IASM08 Ver 3.02
     ;Entry Label:                   KeyPdInt_Init
     ;Module size:                   274 bytes
     ;      KeyPdInt                        144 bytes
     ;      KeyPad                          102 bytes
     ;      Delayms                          28 bytes
     ;Stack space used:              0 bytes
     ;RAM used:                      0 bytes
     ;Global variables used:         None
     ;Worst case execution:          787 Cycles
     ;Subroutines used:              KeyPadPD, Delayms
     ;Part resources needed:         8-bit I/O port B


 Program Initialization
;This code sets up PTD as inputs on row lines and output low levels on column lines.
                 org    EPROM
 KeyPdInt_Init
                 mov    #$0f,DDRD         ;Set high nibble of PTD as input,
                                          ;low nibble as output. This plus
                                          ;values in PTDICR sets up high
                                          ;nibble as interrupt sources.
                 mov    #$f0, PTD         ;Set low nibble to output 0. This
                                          ;defeats pullups on high nibble if
                                          ;key is pressed.
                 mov    #$f0, PTDICR      ;Enable PTD pullups and interrupts
                 rts                      ;on high nibble.
;KeyPdInt Main Program Loop
;This section calls the setup routine and then enters stop mode. All other
;program execution is contained in the KeyPdInt_Isr, the external interrupt
;service routine for this code. The PTD keypad interrupts are enabled
 Start:
 KeyPdInt_Body   jsr    KeyPdInt_Init     ;Call the initialization routine.
                 cli                      ;Unmask interrupts.
 KeyPdInt_Body010
                 STOP                     ;Execute STOP instruction to put
                                          ;MCU in lowest power mode.
                                          ;The keypad can exit from STOP.
                 bra    KeyPdInt_Body010  ;Loop endlessly to remain in STOP.
 IRQ Interrupt Service Routine
;This is the external interrupt service routine. Both the external interrupt
;pin IRQ and the keypad interrupts use this routine. The real work of the
;program is done within this service routine.
```

```
KeyPdInt_Isr:    ;Any decoding of IRQ2 should be done here. There are no
                 ;provisions for latching ITQ2 separately from the PTD keypad
                 ;interrupts. Therefore, any IRQ2 use would be
                 ;level-sensitive-only using the PIN2 bit.
Keypdint_isr010  jsr    KeyPad_Body        ;See if a key is pressed.
                                           ;If no key pressed, return.
                 beq    KeyPdInt_Isr090
                 lda    #$4                ;Debounce key for 4 ms.
                 jsr    Delayms_Body       ;Jump to delay routine.
                 jsr    KeyPad_Body        ;Get the keypress.
                 beq    KeyPdInt_Isr090    ;If no key pressed, return.

                ;Place code here to act on each key.

KeyPdInt_Isr080  lda    #!200              ;Delay 200 ms.
                 jsr    Delayms_Body       ;Pause for key repeat.
                 bra    KeyPdInt_Isr010    ;Back to beginning to repeat.
KeyPdInt_Isr090  lda    #!10               ;Delay 10 ms.
                 jsr    Delayms_Body       ;Debounce the release.
                 jsr    KeyPd_Init         ;Put the keypad back in the
                                           ;correct state to interrupt the
                                            MCU.
                 bset   ack2,iscr          ;Clear any pending interrupts that
                                           ;resulted from decoding the
                                           ;keypad.
                 rti                       ;Return from interrupt.
;Subroutine Body Includes Section
;These include statements include the subroutines that are called by this
;program. KeyPadPD.SUB actually decodes the keypad. Delayms.SUB delays
;operation in increments of milliseconds.

 $INCLUDE ‘KeyPadPD.SUB’
 $INCLUDE ‘Delayms.SUB’

;Interrupt and reset vectors for main routine
 org             RESET
 fdb             Start
 org             IRQ2_INT
 fdb             KeyPdInt_Isr
```

SECTION 18
LOW-VOLTAGE INHIBIT MODULE (LVI)

18.1 Introduction

This section describes the low-voltage inhibit module (LVI27, Version A), which monitors the voltage on the V_{DD} pin and can force a reset when the V_{DD} voltage falls to the LVI trip voltage.

18.2 Features

Features of the LVI module include the following:

- Programmable LVI Reset
- Programmable Power Consumption

18.3 Functional Description

Figure 18-1 shows the structure of the LVI module. The LVI module contains a bandgap reference circuit and comparator. The LVI power bit, LVIPWR, enables the LVI to monitor V_{DD} voltage. The LVI reset bit, LVIRST, enables the LVI module to generate a reset when V_{DD} falls below a voltage, LVI_{TRIPF}. LVIPWR and LVIRST are in the mask option register (MOR). (See **SECTION 5 MASK OPTION REGISTER (MOR)**.) Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises above a voltage, LVI_{TRIPR}. The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

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

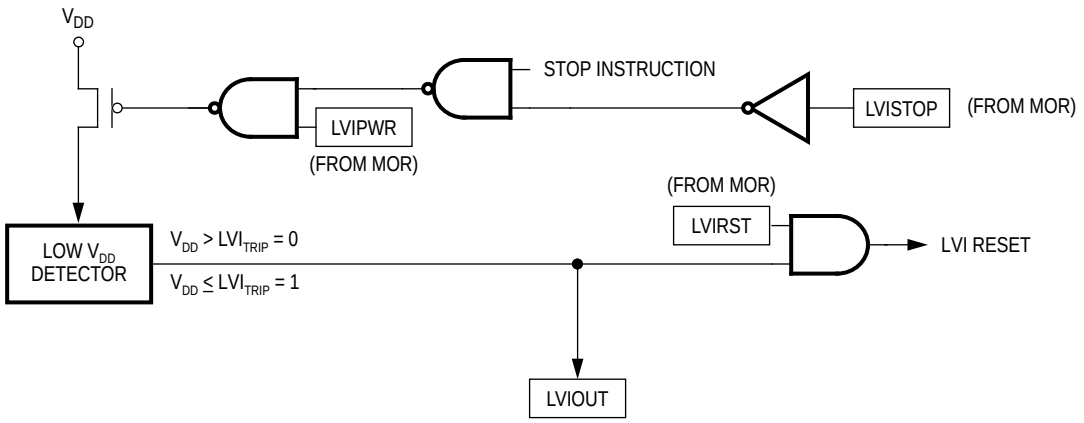


Figure 18-1. LVI Module Block Diagram

Table 18-1. LVI I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
LVI Status Register (LVISR)	LVIOUT								$FE0F

= Unimplemented

18.3.1 Polled LVI Operation

In applications that can operate at V_{DD} levels below the LVI_{TRIPF} level, software can monitor V_{DD} by polling the LVIOUT bit. In the mask option register, the LVIPWR bit must be at logic one to enable the LVI module, and the LVIRST bit must be at logic zero to disable LVI resets.

18.3.2 Forced Reset Operation

In applications that require V_{DD} to remain above the LVI_{TRIPF} level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls to the LVI_{TRIPF} level. In the mask option register, the LVIPWR and LVIRST bits must be at logic one to enable the LVI module and to enable LVI resets.

18.4 LVI Status Register (LVISR)

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

		Bit 7	6	5	4	3	2	1	Bit 0
LVISR $FE0F	Read:	LVIOUT	0	0	0	0	0	0	0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 18-2. LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

This read-only flag becomes set when the V_{DD} voltage falls below the LVI_{TRIPF} voltage. (See Table 18-2.) Reset clears the LVIOUT bit.

Table 18-2. LVIOUT Bit Indication

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

18.5 LVI Interrupts

The LVI module does not generate interrupt requests.

18.6 Low-Power Modes

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

18.6.1 Wait Mode

With the LVIPWR bit in the mask option register programmed to logic one, the LVI module is active after a WAIT instruction.

With the LVIRST bit in the mask option register programmed to logic one, the LVI module can generate a reset and bring the MCU out of wait mode.

18.6.2 Stop Mode

When the LVIPWR and LVISTOP bits in the mask option register are programmed to logic one, the LVI module remains active after a STOP instruction.

NOTE

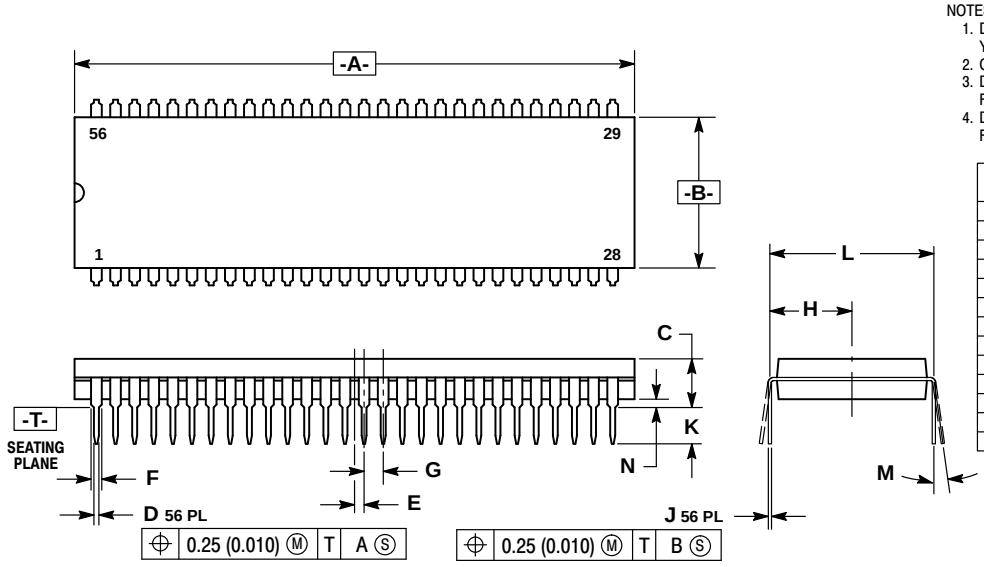
If the LVIPWR bit is at logic one, the LVISTOP bit must be at logic zero to meet the minimum stop mode I_{DD} specification.

SECTION 19
MECHANICAL SPECIFICATIONS

19.1 Introduction

This section gives the dimensions of the 56-pin plastic shrink dual-in-line package (SDIP) and the 64-lead plastic quad flat pack (QFP).

19.2 Plastic Shrink Dual-in-Line Package (SDIP)

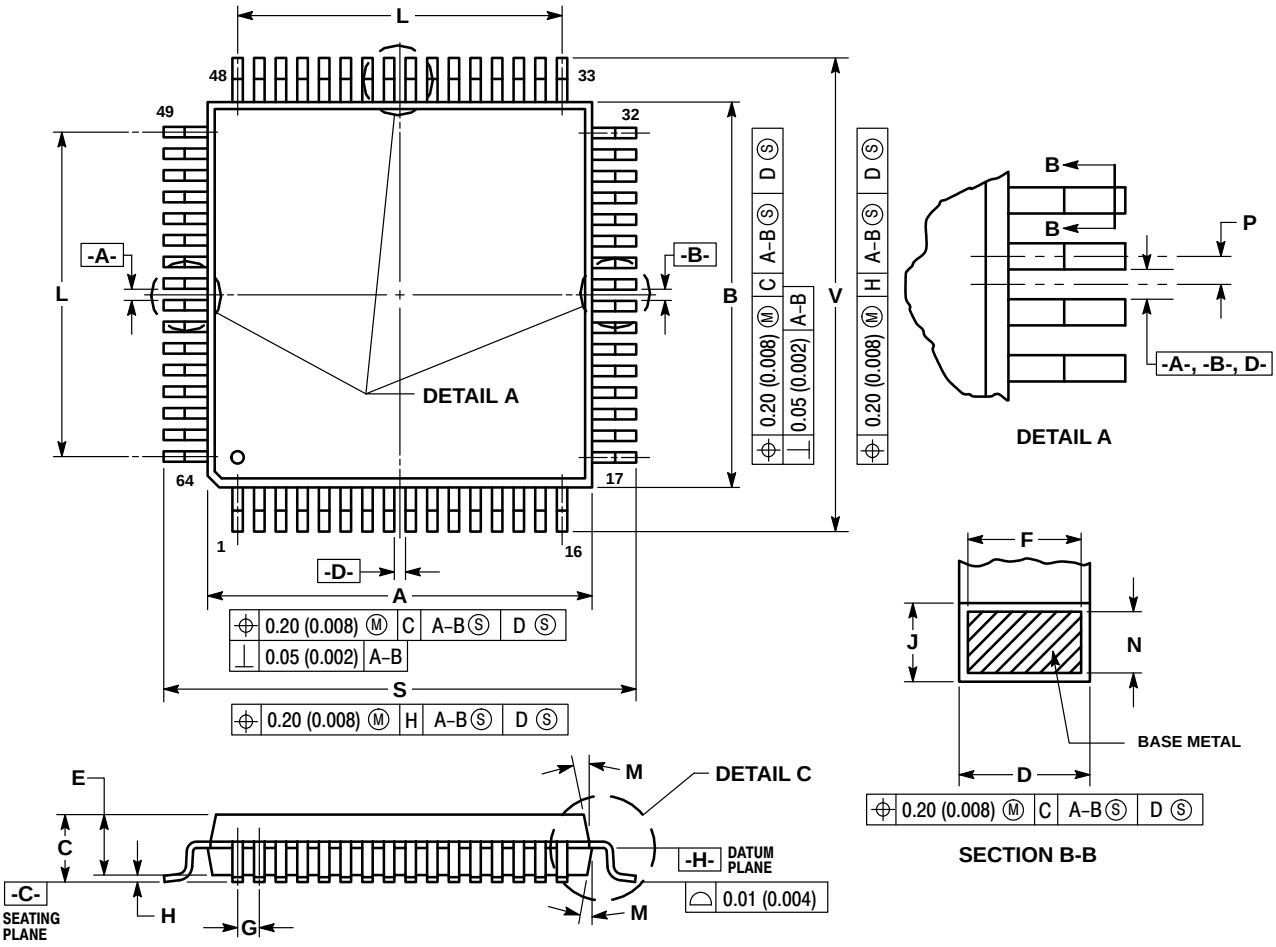


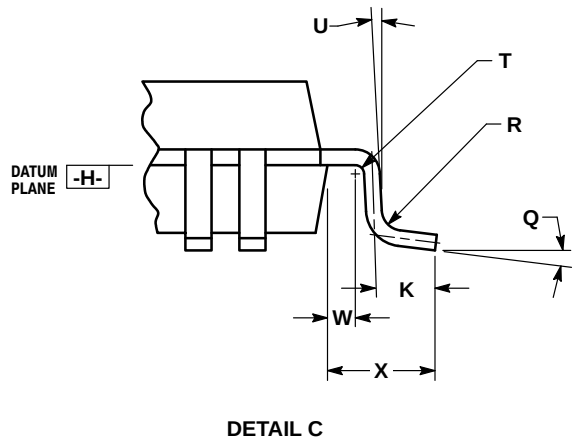
NOTES:
1. DIMENSIONS AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
4. DIMENSIONS A AND B DO NOT INCLUDE MOLD FLASH. MAXIMUM MOLD FLASH 0.25 (0.010).

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	2.035	2.065	51.69	52.45
B	0.540	0.560	13.72	14.22
C	0.155	0.200	3.94	5.08
D	0.014	0.022	0.36	0.56
E	0.035 BSC		0.89 BSC	
F	0.032	0.046	0.81	1.17
G	0.070 BSC		1.778 BSC	
H	0.300 BSC		7.62 BSC	
J	0.008	0.015	0.20	0.38
K	0.115	0.135	2.92	3.43
L	0.600 BSC		15.24 BSC	
M	0°	15°	0°	15°
N	0.020	0.040	0.51	1.02

Figure 19-1. MC68HC708XL36B (Case #859-01)

19.3 Plastic Quad Flat Pack (QFP)





DETAIL C

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE -H- IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS A–B AND -D- TO BE DETERMINED AT DATUM PLANE -H-.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE -C-.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 (0.010) PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE -H-.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 (0.003) TOTAL IN EXCESS OF THE D DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	13.90	14.10	0.547	0.555
B	13.90	14.10	0.547	0.555
C	2.15	2.45	0.085	0.096
D	0.30	0.45	0.012	0.018
E	2.00	2.40	0.079	0.094
F	0.30	0.40	0.012	0.016
G	0.80 BSC		0.031 BSC	
H	—	0.25	—	0.010
J	0.13	0.23	0.005	0.009
K	0.65	0.95	0.026	0.037
L	12.00 REF		0.472 REF	
M	5°	10°	5°	10°
N	0.13	0.17	0.005	0.007
P	0.40 BSC		0.016 BSC	
Q	0°	7°	0°	7°
R	0.13	0.30	0.005	0.012
S	16.95	17.45	0.667	0.687
T	0.13	—	0.005	—
U	0°	—	0°	—
V	16.95	17.45	0.667	0.687
W	0.35	0.45	0.014	0.018
X	1.6 REF		0.063 REF	

Figure 19-2. MC68HC708XL36FU (Case #840C-01)

GLOSSARY

A — See “accumulator (A).”

accumulator (A) — An 8-bit general-purpose register in the CPU08. The CPU08 uses the accumulator to hold operands and results of arithmetic and logic operations.

acquisition mode — A mode of PLL operation during startup before the PLL locks on a frequency. Also see "tracking mode."

address bus — The set of wires that the CPU or DMA uses to read and write memory locations.

addressing mode — The way that the CPU determines the operand address for an instruction. The M68HC08 CPU has 16 addressing modes.

ALU — See “arithmetic logic unit (ALU).”

arithmetic logic unit (ALU) — The portion of the CPU that contains the logic circuitry to perform arithmetic, logic, and manipulation operations on operands.

asynchronous — Refers to logic circuits and operations that are not synchronized by a common reference signal.

baud rate — The total number of bits transmitted per unit of time.

BCD — See “binary-coded decimal (BCD).”

binary — Relating to the base 2 number system.

binary number system — The base 2 number system, having two digits, 0 and 1. Binary arithmetic is convenient in digital circuit design because digital circuits have two permissible voltage levels, low and high. The binary digits 0 and 1 can be interpreted to correspond to the two digital voltage levels.

binary-coded decimal (BCD) — A notation that uses four-bit binary numbers to represent the 10 decimal digits and that retains the same positional structure of a decimal number. For example,

G

234 (decimal) = 0010 0011 0100 (BCD)

bit — A binary digit. A bit has a value of either logic zero or logic one.

branch instruction — An instruction that causes the CPU to continue processing at a memory location other than the next sequential address.

break module — A module in the M68HC08 Family. The break module allows software to halt program execution at a programmable point in order to enter a background routine.

breakpoint — A number written into the break address registers of the break module. When a number appears on the internal address bus that is the same as the number in the break address registers, the CPU executes the software interrupt instruction (SWI).

break interrupt — A software interrupt caused by the appearance on the internal address bus of the same value that is written in the break address registers.

bus — A set of wires that transfers logic signals.

bus clocks — There are two bus clocks, IT12 and IT23. These clocks are generated by the CGM and distributed throughout the MCU by the SIM. The frequency of the bus clocks, or operating frequency, is f_{OP}. While the frequency of these two clocks is the same, the phase is different.

byte — A set of eight bits.

C — The carry/borrow bit in the condition code register. The CPU08 sets the carry/borrow bit when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the carry/borrow bit (as in bit test and branch instructions and shifts and rotates).

CCR — See “condition code register.”

central processor unit (CPU) — The primary functioning unit of any computer system. The CPU controls the execution of instructions.

CGM — See “clock generator module (CGM).”

clear — To change a bit from logic one to logic zero; the opposite of set.

clock — A square wave signal used to synchronize events in a computer.

clock generator module (CGM) — A module in the M68HC08 Family. The CGM generates a base clock signal from which the system clocks are derived. The CGM may include a crystal oscillator circuit and or phase-locked loop (PLL) circuit.

comparator — A device that compares the magnitude of two inputs. A digital comparator defines the equality or relative differences between two binary numbers.

computer operating properly module (COP) — A counter module in the M68HC08 Family that resets the MCU if allowed to overflow.

condition code register (CCR) — An 8-bit register in the CPU08 that contains the interrupt mask bit and five bits that indicate the results of the instruction just executed.

control bit — One bit of a register manipulated by software to control the operation of the module.

control unit — One of two major units of the CPU. The control unit contains logic functions that synchronize the machine and direct various operations. The control unit decodes instructions and generates the internal control signals that perform the requested operations. The outputs of the control unit drive the execution unit, which contains the arithmetic logic unit (ALU), CPU registers, and bus interface.

COP — See "computer operating properly module (COP)."

counter clock — The input clock to the TIM counter. This clock is an output of the prescaler sub-module. The frequency of the counter clock is f_{TCNT}, and the period is t_{TCNT}.

CPU — See “central processor unit (CPU).”

CPU08 — The central processor unit of the M68HC08 Family.

CPU cycles — A CPU clock cycle is one period of the internal bus-rate clock, f_{OP}, normally derived by dividing a crystal oscillator source by two or more so the high and low times will be equal. The length of time required to execute an instruction is measured in CPU clock cycles.

CPU registers — Memory locations that are wired directly into the CPU logic instead of being part of the addressable memory map. The CPU always has direct access to the information in these registers. The CPU registers in an M68HC08 are:

- A (8-bit accumulator)
- H:X (16-bit index register)
- SP (16-bit stack pointer)
- PC (16-bit program counter)
- CCR (condition code register containing the V, H, I, N, Z, and C bits)

CSIC — customer-specified integrated circuit

cycle time — The period of the operating frequency: $t_{CYC} = 1/f_{OP}$.

decimal number system — Base 10 numbering system that uses the digits zero through nine.

direct memory access module (DMA) — A M68HC08 Family module that can perform data transfers between any two CPU-addressable locations without CPU intervention. For transmitting or receiving blocks of data to or from peripherals, DMA transfers are faster and more code-efficient than CPU interrupts.

DMA — See "direct memory access module (DMA)."

DMA service request — A signal from a peripheral to the DMA module that enables the DMA module to transfer data.

duty cycle — A ratio of the amount of time the signal is on versus the time it is off. Duty cycle is usually represented by a percentage.

EEPROM — Electrically erasable, programmable, read-only memory. A nonvolatile type of memory that can be electrically reprogrammed.

EPROM — Erasable, programmable, read-only memory. A non-volatile type of memory that can be erased by exposure to an ultraviolet light source and then reprogrammed.

exception — An event such as an interrupt or a reset that stops the sequential execution of the instructions in the main program.

external interrupt module (IRQ) — A module in the M68HC08 Family with both dedicated external interrupt pins and port pins that can be enabled as interrupt pins.

fetch — To copy data from a memory location into the accumulator.

firmware — Instructions and data programmed into nonvolatile memory.

free-running counter — A device that counts from zero to a predetermined number, then rolls over to zero and begins counting again.

full-duplex transmission — Communication on a channel in which data can be sent and received simultaneously.

H — The upper byte of the 16-bit index register (H:X) in the CPU08.

H — The half-carry bit in the condition code register of the CPU08. This bit indicates a carry from the low-order four bits of the accumulator value to the high-order four bits. The half-carry bit is required for binary-coded decimal arithmetic operations. The decimal adjust accumulator (DAA) instruction uses the state of the H and C bits to determine the appropriate correction factor.

hexadecimal — Base 16 numbering system that uses the digits 0 through 9 and the letters A through F.

high byte — The most significant eight bits of a word.

illegal address — An address not within the memory mapl

illegal opcode — A nonexistent opcode.

I — The interrupt mask bit in the condition code register of the CPU08. When I is set, all interrupts are disabled.

index register (H:X) — A 16-bit register in the CPU08. The upper byte of H:X is called H. The lower byte is called X. In the indexed addressing modes, the CPU uses the contents of H:X to determine the effective address of the operand. H:X can also serve as a temporary data storage location.

input/output (I/O) — Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal and writes to an output to change the level on an external signal.

instructions — Operations that a CPU can perform. Instructions are expressed by programmers as assembly language mnemonics. A CPU interprets an opcode and its associated operand(s) and instruction.

interrupt — A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine.

interrupt request — A signal from a peripheral to the CPU intended to cause the CPU to execute a subroutine.

I/O — See “input/output (I/0).”

IRQ — See "external interrupt module (IRQ)."

jitter — Short-term signal instability.

latch — A circuit that retains the voltage level (logic one or logic zero) written to it for as long as power is applied to the circuit.

latency — The time lag between instruction completion and data movement.

least significant bit (LSB) — The rightmost digit of a binary number.

logic one — A voltage level approximately equal to the input power voltage (V_{DD}).

logic zero — A voltage level approximately equal to the ground voltage (V_{SS}).

low byte — The least significant eight bits of a word.

low voltage inhibit module (LVI) — A module in the M68HC08 Family that monitors power supply voltage.

LVI — See "low voltage inhibit module (LVI)."

M68HC08 — A Motorola family of 8-bit MCUs.

mark/space — The logic one/logic zero convention used in formatting data in serial communication.

mask — 1. A logic circuit that forces a bit or group of bits to a desired state. 2. A photomask used in integrated circuit fabrication to transfer an image onto silicon.

mask option — A optional microcontroller feature that that the customer chooses to enable or disable.

mask option register (MOR) — An EPROM location containing bits that enable or disable certain MCU features.

MCU — Microcontroller unit. See “microcontroller.”

memory location — Each M68HC08 memory location holds one byte of data and has a unique address. To store information in a memory location, the CPU places the address of the location on the address bus, the data information on the data bus, and asserts the write signal. To read information from a memory location, the CPU places the address of the location on the address bus and asserts the read signal. In response to the read signal, the selected memory location places its data onto the data bus.

memory map — A pictorial representation of all memory locations in a computer system.

microcontroller — Microcontroller unit (MCU). A complete computer system, including a CPU, memory, a clock oscillator, and input/output (I/O) on a single integrated circuit.

modulo counter — A counter that can be programmed to count to any number from zero to its maximum possible modulus.

monitor ROM — A section of ROM that can execute commands from a host computer for testing purposes.

MOR — See "mask option register (MOR)."

most significant bit (MSB) — The leftmost digit of a binary number.

multiplexer — A device that can select one of a number of inputs and pass the logic level of that input on to the output.

N — The negative bit in the condition code register of the CPU08. The CPU sets the negative bit when an arithmetic operation, logical operation, or data manipulation produces a negative result.

nibble — A set of four bits (half of a byte).

object code — The output from an assembler or compiler that is itself executable machine code, or is suitable for processing to produce executable machine code.

opcode — A binary code that instructs the CPU to perform an operation.

open-drain — An output that has no pullup transistor. An external pullup device can be connected to the power supply to provide the logic one output voltage.

operand — Data on which an operation is performed. Usually a statement consists of an operator and an operand. For example, the operator may be an add instruction, and the operand may be the quantity to be added.

oscillator — A circuit that produces a constant frequency square wave that is used by the computer as a timing and sequencing reference.

OTPROM — One-time programmable read-only memory. A nonvolatile type of memory that cannot be reprogrammed.

overflow — A quantity that is too large to be contained in one byte or one word.

page zero — The first 256 bytes of memory (addresses $0000–$00FF).

parity — An error-checking scheme that counts the number of logic ones in each byte transmitted. In a system that uses odd parity, every byte is expected to have an odd number of logic ones. In an even parity system, every byte should have an even number of logic ones. In the transmitter, a parity generator appends an extra bit to each byte to make the number of logic ones odd for odd parity or even for even parity. A parity checker in the receiver counts the number of logic ones in each byte. The parity checker generates an error signal if it finds a byte with an incorrect number of logic ones.

PC — See “program counter (PC).”

peripheral — A circuit not under direct CPU control.

phase-locked loop (PLL) — A oscillator circuit in which the frequency of the oscillator is synchronized to a reference signal.

PLL — See "phase-locked loop (PLL)."

pointer — Pointer register. An index register is sometimes called a pointer register because its contents are used in the calculation of the address of an operand, and therefore points to the operand.

polarity — The two opposite logic levels, logic one and logic zero, which correspond to two different voltage levels, V_{DD} and V_{SS}.

polling — Periodically reading a status bit to monitor the condition of a peripheral device.

port — A set of wires for communicating with off-chip devices.

prescaler — A circuit that generates an output signal related to the input signal by a fractional scale factor such as 1/2, 1/8, 1/10 etc.

program — A set of computer instructions that cause a computer to perform a desired operation or operations.

program counter (PC) — A 16-bit register in the CPU08. The PC register holds the address of the next instruction or operand that the CPU will use.

pull — An instruction that copies into the accumulator the contents of a stack RAM location. The stack RAM address is in the stack pointer.

pullup — A transistor in the output of a logic gate that connects the output to the logic one voltage of the power supply.

pulse-width — The amount of time a signal is on as opposed to being in its off state.

pulse-width modulation (PWM) — Controlled variation (modulation) of the pulse width of a signal with a constant frequency.

push — An instruction that copies the contents of the accumulator to the stack RAM. The stack RAM address is in the stack pointer.

PWM period — The time required for one complete cycle of a PWM waveform.

RAM — Random access memory. All RAM locations can be read or written by the CPU. The contents of a RAM memory location remain valid until the CPU writes a different value or until power is turned off.

RC circuit — A circuit consisting of capacitors and resistors having a defined time constant.

read — To copy the contents of a memory location to the accumulator.

register — A circuit that stores a group of bits.

reserved memory location — A memory location that is used only in special factory-test modes. Writing to a reserved location has no effect. Reading a reserved location returns an unpredictable value.

reset — To force a device to a known condition.

ROM — Read-only memory. A type of memory that can be read but cannot be changed (written). The contents of ROM must be specified before manufacturing the MCU.

SCI — See "serial communication interface module (SCI)."

serial — Pertaining to sequential transmission over a single line.

serial communication interface module (SCI) — A module in the M68HC08 Family that supports asynchronous communication.

serial peripheral interface module (SPI) — A module in the M68HC08 Family that supports synchronous communicaton.

set — To change a bit from logic zero to logic one; opposite of clear.

shift register — A chain of circuits that can retain the logic levels (logic one or logic zero) written to them and that can shift the logic levels to the right or left through adjacent circuits in the chain.

signed — A binary number notation that accommodates both positive and negative numbers. The most significant bit is used to indicate whether the number is positive or negative, normally logic zero for positive and logic one for negative. The other seven bits indicate the magnitude of the number.

SIM — See "system integration module (SIM)."

software — Instructions and data that control the operation of a microcontroller.

software interrupt (SWI) — An instruction that causes an interrupt and its associated vector fetch.

SPI — See "serial peripheral interface module (SPI)."

stack — A portion of RAM reserved for storage of CPU register contents and subroutine return addresses.

stack pointer (SP) — A 16-bit register in the CPU08 containing the address of the next available storage location on the stack.

start bit — A bit that signals the beginning of an asynchronous serial transmission.

status bit — A register bit that indicates the condition of a device.

stop bit — A bit that signals the end of an asynchronous serial transmission.

subroutine — A sequence of instructions to be used more than once in the course of a program. The last instruction in a subroutine is a return from subroutine (RTS) instruction. At each place in the main program where the subroutine instructions are needed, a jump or branch to subroutine (JSR or BSR) instruction is used to call the subroutine. The CPU leaves the flow of the main program to execute the instructions in the subroutine. When the RTS instruction is executed, the CPU returns to the main program where it left off.

synchronous — Refers to logic circuits and operations that are synchronized by a common reference signal.

system integration module (SIM) — One of a number of modules that handle a variety of control functions in the modular M68HC08 Family. The SIM controls mode of operation, resets and interrupts, and system clock distribution.

TIM — See "timer interface module (TIM)."

timer interface module (TIM) — A module used to relate events in a system to a point in time.

timer — A module used to relate events in a system to a point in time.

toggle — To change the state of an output from a logic zero to a logic one or from a logic one to a logic zero.

tracking mode — Mode of low-jitter PLL operation during which the PLL is locked on a frequency. Also see "acquisition mode."

two's complement — A means of performing binary subtraction using addition techniques. The most significant bit of a two's complement number indicates the sign of the number (1 indicates negative). The two's complement negative of a number is obtained by inverting each bit in the number and then adding 1 to the result.

unbuffered — Utilizes only one register for data; new data overwrites current data.

unimplemented memory location — A memory location that is not used. Writing to an unimplemented location has no effect. Reading an unimplemented location returns an unpredictable value. Executing an opcode at an unimplemented location causes an illegal address reset.

V —The overflow bit in the condition code register of the CPU08. The CPU08 sets the V bit when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow bit.

variable — A value that changes during the course of program execution.

VCO — See "voltage-controlled oscillator."

vector — A memory location that contains the address of the beginning of a subroutine written to service an interrupt or reset.

voltage-controlled oscillator (VCO) — A circuit that produces an oscillating output signal of a frequency that is controlled by a dc voltage applied to a control input.

waveform — A graphical representation in which the amplitude of a wave is plotted against time.

wired-OR — Connection of circuit outputs so that if any output is high, the connection point is high.

word — A set of two bytes (16 bits).

write — The transfer of a byte of data from the CPU to a memory location.

X — The lower byte of the index register (H:X) in the CPU08.

Z — The zero bit in the condition code register of the CPU08. The CPU08 sets the zero bit when an arithmetic operation, logical operation, or data manipulation produces a result of $00.

INDEX

I

D

MOTOROLA

MC68HC708XL36 TECHNICAL SUMMARY
CUSTOMER RESPONSE SURVEY

To make M68HC08 documentation as clear, complete, and easy to use as possible, we need your comments. Please complete this form and return it by mail, or FAX it to 512-891-3236.

1. How do you rate the quality of this document?

	High				Low
Organization	☐	☐	☐	☐	☐
Readability	☐	☐	☐	☐	☐
Accuracy	☐	☐	☐	☐	☐
Figures	☐	☐	☐	☐	☐

	High				Low
Tables	☐	☐	☐	☐	☐
Table of contents	☐	☐	☐	☐	☐
Page size/binding	☐	☐	☐	☐	☐
Overall impression	☐	☐	☐	☐	☐

Comments: _____

_____

_____

2. What is your intended use for this document?

Device selection for new application ☐
System design ☐
Training ☐
Other ☐ Please specify: _____

3. Does this document help you to perform your job?

Yes				No
☐	☐	☐	☐	☐

Comments: _____

_____

_____

4. Are you able to easily find the information you need?

Yes				No
☐	☐	☐	☐	☐

Comments: _____

_____

_____

5. Does each section of the document provide you with enough information?

	Yes				No
SEC. 1: GEN. DESCRIPTION	☐	☐	☐	☐	☐
SEC. 2: MEMORY MAP	☐	☐	☐	☐	☐
SEC. 3: RAM	☐	☐	☐	☐	☐
SEC. 4: EPROM/OTPROM	☐	☐	☐	☐	☐
SEC. 5: MOR	☐	☐	☐	☐	☐
SEC. 6: CPU	☐	☐	☐	☐	☐
SEC. 7: SIM	☐	☐	☐	☐	☐
SEC. 8: CGM	☐	☐	☐	☐	☐
SEC. 9: DMA	☐	☐	☐	☐	☐
SEC. 10: BRK	☐	☐	☐	☐	☐

	Yes				No
SEC. 11: MON	☐	☐	☐	☐	☐
SEC. 12: TIM	☐	☐	☐	☐	☐
SEC. 13: SPI	☐	☐	☐	☐	☐
SEC. 14: SCI	☐	☐	☐	☐	☐
SEC. 15: I/O	☐	☐	☐	☐	☐
SEC. 16: COP	☐	☐	☐	☐	☐
SEC. 17: IRQ	☐	☐	☐	☐	☐
SEC. 18: LVI	☐	☐	☐	☐	☐
SEC. 19: MECHANICAL	☐	☐	☐	☐	☐
GLOSSARY	☐	☐	☐	☐	☐
INDEX	☐	☐	☐	☐	☐

6. What would you like us to do to improve this document? _____

_____

_____

_____

_____

_____

_____

Motorola
6501 William Cannon Drive West
Mail Stop OE17
Austin, Texas 78735-8598

Attn: CSIC Publications Department

MOTOROLA

CSIC Microcontroller Division

First: cut along this line to remove

Second: fold back along this line

Please supply the following information (optional).

Name: _____

Company Name: _____

Title: _____

Address: _____

City: _____State: _____Zip:_____

Phone Number: _____