MC68HC908AB32/D
REV. 1.0

HC08

MC68HC908AB32

HCMOS Microcontroller Unit

TECHNICAL DATA

MOTOROLA

List of Sections

List of Sections

Table of Contents

Section 1. General Description

Section 2. Memory Map

Section 3. Random-Access Memory (RAM)

Section 4. FLASH Memory

Section 5. EEPROM

Section 6. Configuration Register (CONFIG)

Section 7. Central Processor Unit (CPU)

Section 8. System Integration Module (SIM)

Section 9. Clock Generator Module (CGM)

Section 10. Monitor ROM (MON)

Section 11. Timer Interface Module A (TIMA)

Section 12. Timer Interface Module B (TIMB)

Section 13. Programmable Interrupt Timer (PIT)

Section 14. Analog-to-Digital Converter (ADC)

Section 15. Serial Communications Interface Module (SCI)

Section 16. Serial Peripheral Interface Module (SPI)

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

Section 18. External Interrupt (IRQ)

Section 19. Keyboard Interrupt Module (KBI)

Section 20. Computer Operating Properly (COP)

Section 21. Low-Voltage Inhibit (LVI)

Section 22. Break Module (BRK)

Section 23. Electrical Specifications

Section 24. Mechanical Specifications

Section 25. Ordering Information

List of Figures

List of Figures

Figure	Title	Page

Figure	Title	Page

Figure	Title	Page

List of Figures

List of Tables

List of Tables

Section 1. General Description

1.1 Contents

1.2 Introduction

The MC68HC908AB32 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs) with embedded EEPROM for user data storage. All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

1.3 Features

Features of the MC68HC908AB32 include the following:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- Memory map and pin functions compatible with MC68HC08AB32 and MC68HC08AB16
- 8-MHz internal bus frequency
- 32K-bytes user program FLASH memory with security[1] feature
- 512 bytes of on-chip EEPROM with security feature
- 1K-byte of on-chip RAM
- Clock generator module (CGM)
- Two 16-bit, 4-channel timer interface modules (TIMA and TIMB) with selectable input capture, output compare, and PWM capability on each channel
- Programmable interrupt timer (PIT)
- Serial peripheral interface module (SPI)
- Serial communications interface module (SCI)
- 8-channel. 8-bit analog-to-digital converter (ADC)
- Low-power design (fully static with STOP and WAIT modes)
- Master reset pin and power-on reset

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

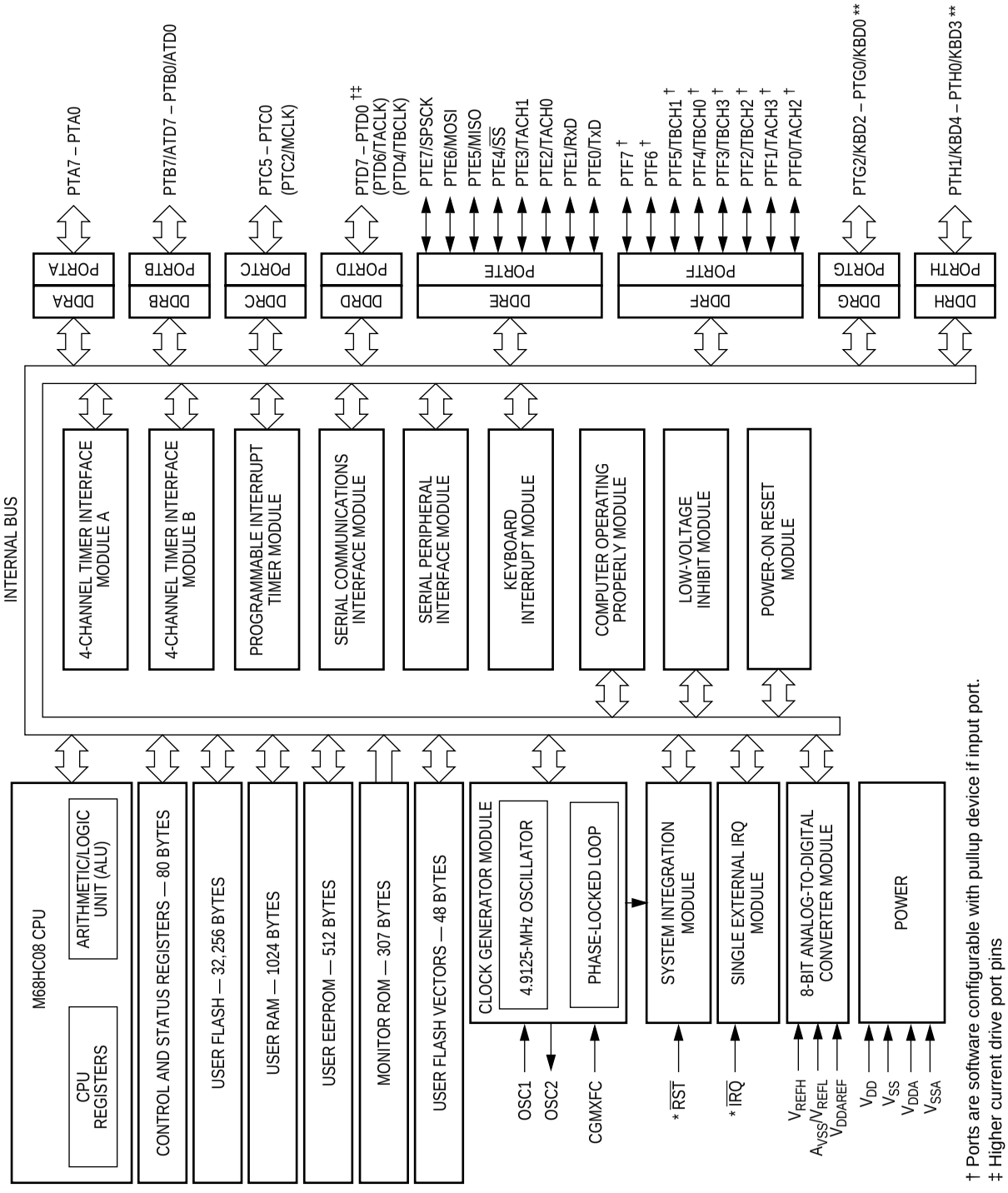
- 51 general-purpose input/output (I/O) pins:
  - 30 shared-function I/O pins
  - 5-bit keyboard wakeup port
  - Selectable pullups on inputs on port D and port F
- System protection features
  - Optional computer operating properly (COP) reset
  - Low-voltage detection with optional reset
  - Illegal opcode detection with optional reset
  - Illegal address detection with optional reset
- 64-pin quad flat pack (QFP)

Features of the CPU08 include the following:

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

1.4 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908AB32.



† Ports are software configurable with pullup device if input port.
‡ Higher current drive port pins
* Pin contains integrated pullup device
** Pullup enabled when configured as keyboard interrupt pin

Figure 1-1. MC68HC908AB32 Block Diagram

1.5 Pin Assignments

Figure 1-2 shows the pin assignment for the MC68HC908AB32.

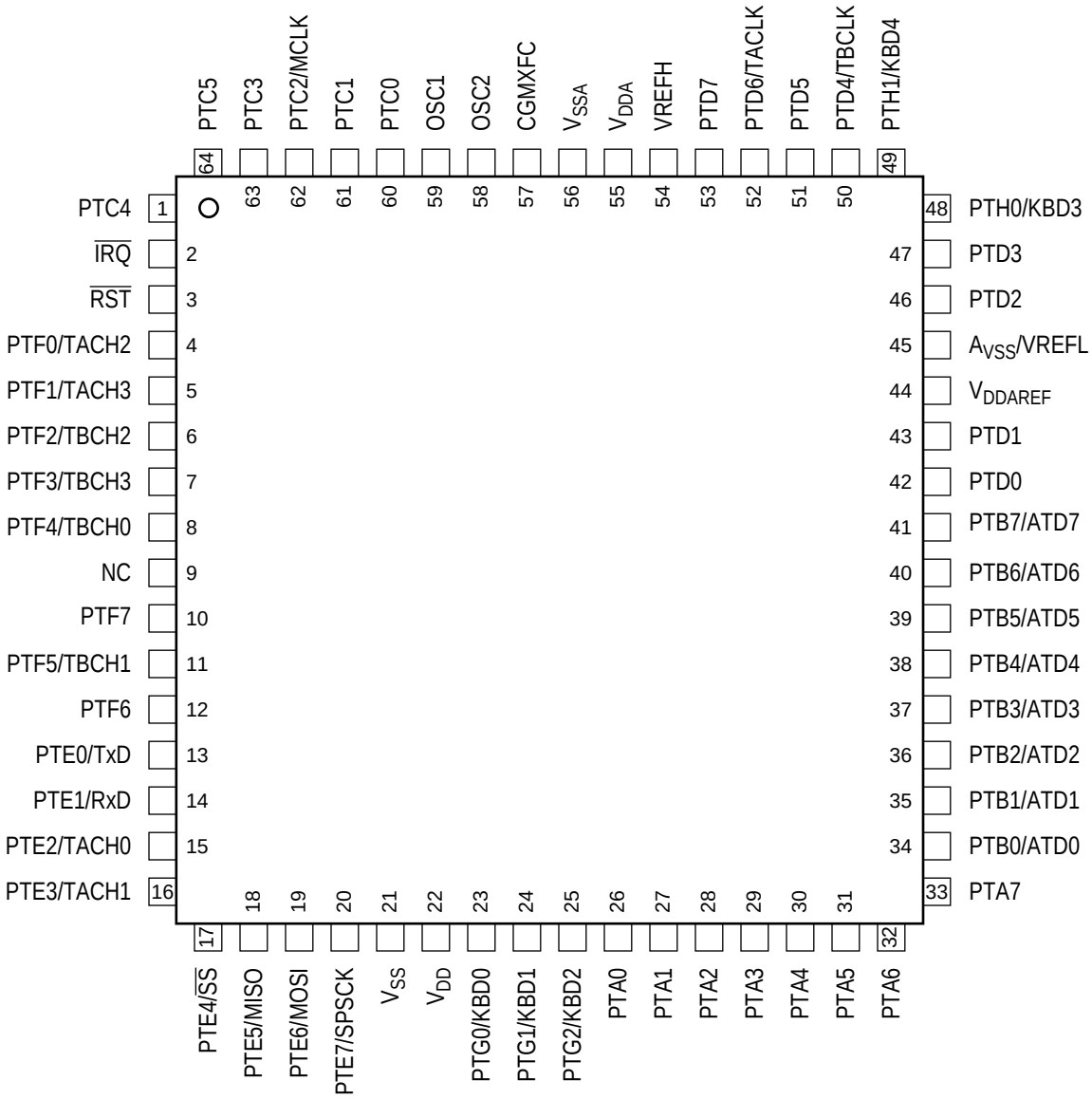


Figure 1-2. 64-Pin QFP Pin Assignment

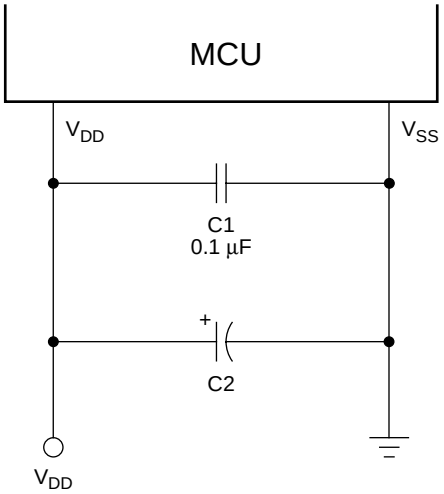
1.6 Pin Functions

Description of pin functions are provided here.

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

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

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



NOTE: Component values shown represent typical applications.

Figure 1-3. Power Supply Bypassing

V_{SS} is also the ground for the port output buffers and the ground return for the serial clock in the serial peripheral interface module (SPI). See **Section 16. Serial Peripheral Interface Module (SPI)**.

V_{SS} must be grounded for proper MCU operation.

1.6.2 Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. See **Section 9. Clock Generator Module (CGM)**.

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

A logic 0 on the $\overline{RST}$ pin forces the MCU to a known start-up state. $\overline{RST}$ is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted. This pin contains an internal pullup resistor. See **Section 8. System Integration Module (SIM)**.

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

$\overline{IRQ}$ is an asynchronous external interrupt pin. This pin contains an internal pullup resistor. See **Section 18. External Interrupt (IRQ)**.

1.6.5 Analog Power Supply Pin (V_{DDA})

V_{DDA} is the power supply pin for the clock generator module (CGM).

1.6.6 Analog Ground Pin (V_{SSA})

The V_{SSA} analog ground pin is used only for the ground connections for the clock generator module (CGM) section of the circuit and should be decoupled as per the V_{SS} digital ground pin. See **Section 9. Clock Generator Module (CGM)**.

1.6.7 Analog Ground Pin (A_{VSS}/VREFL)

The A_{VSS} analog ground pin is used only for the ground connections for the analog to digital convertor (ADC) and should be decoupled as per the V_{SS} digital ground pin.

1.6.8 ADC Voltage Reference Pin (VREFH)

VREFH is the power supply for setting the reference voltage VREFH. Connect this pin to a voltage such that 1.5V < VREFH ≤ V_{DDAREF}.

1.6.9 Analog Supply Pin (V_{DDAREF})

The V_{DDAREF} analog supply pin is used only for the supply connections for the analog-to-digital convertor (ADC).

1.6.10 External Filter Capacitor Pin (CGMXFC)

CGMXFC is an external filter capacitor connection for the CGM. See **Section 9. Clock Generator Module (CGM)**.

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

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

1.6.12 Port B I/O Pins (PTB7/ATD7–PTB0/ATD0)

PTB7–PTB0 are special function, bidirectional port pins. PTB7–PTB0 are shared with the analog to digital convertor (ADC) input pins ATD7–ATD0. See **Section 14. Analog-to-Digital Converter (ADC)** and **Section 17. Input/Output (I/O) Ports**.

1.6.13 Port C I/O Pins (PTC5–PTC0)

PTC5–PTC0 are general-purpose bidirectional I/O port pins. PTC2 is a special function port pin that is shared with the system clock output pin, MCLK. See **Section 17. Input/Output (I/O) Ports**.

1.6.14 Port D I/O Pins (PTD7–PTD0)

PTD7–PTD0 are general-purpose bidirectional I/O port pins. PTD6 and PTD4 are special function port pins that are shared with the timer interface modules (TIMA and TIMB). See **Section 11. Timer Interface Module A (TIMA)** and **Section 12. Timer Interface Module B (TIMB)**.

1.6.15 Port E I/O Pins (PTE7/SPSCK–PTE0/TxD)

PTE7–PTE0 are special function, bidirectional port pins. PTE7–PTE4 are shared with the serial peripheral interface mode (SPI), PTE3–PTE2 are shared with timer A (TIMA), and PTE1–PTE0 are shared with the serial communications interface (SCI). See **Section 15. Serial Communications Interface Module (SCI)**, **Section 16. Serial Peripheral Interface Module (SPI)**, **Section 11. Timer Interface Module A (TIMA)**, and **Section 17. Input/Output (I/O) Ports**.

1.6.16 Port F I/O Pins (PTF7–PTF0/TACH2)

PTF7–PTF6 are general-purpose bidirectional I/O port pins. PTF5–PTF0 are special function, bidirectional port pins. PTF5–PTF2 are shared with timer B (TIMB), and PTF1–PTF0 are shared with timer A (TIMA). See **Section 11. Timer Interface Module A (TIMA)**, **Section 12. Timer Interface Module B (TIMB)**, and **Section 17. Input/Output (I/O) Ports**.

1.6.17 Port G I/O Pins (PTG2/KBD2–PTG0/KBD0)

PTG2–PTG0 are general-purpose bidirectional I/O pins with keyboard wakeup function. See **Section 19. Keyboard Interrupt Module (KBI)** and **Section 17. Input/Output (I/O) Ports**.

1.6.18 Port H I/O Pins (PTH1/KBD4–PTH0/KBD3)

PTH1–PTH0 are general-purpose bidirectional I/O pins with Keyboard wakeup function. See **Section 19. Keyboard Interrupt Module (KBI)** and **Section 17. Input/Output (I/O) Ports**.

1.7 I/O Pin Summary

Table 1-1. I/O Pins Summary

Pin Name	Function	Driver Type	Hysteresis	Reset State
PTA7–PTA0	General purpose I/O	Dual State	No	Input (Hi-Z)
PTB7/ATD7–PTB0/ATD0	General purpose I/O / ADC channel	Dual State	No	Input (Hi-Z)
PTC5–PTC3	General purpose I/O	Dual State	No	Input (Hi-Z)
PTC2/MCLK	General purpose I/O / System clock	Dual State	No	Input (Hi-Z)
PTC1–PTC0	General purpose I/O	Dual State	No	Input (Hi-Z)
PTD7	General purpose I/O	Dual State	No	Input (Hi-Z)
PTD6/TACLK	General purpose I/O / Timer external input clock	Dual State	No	Input (Hi-Z)
PTD5	General purpose I/O	Dual State	No	Input (Hi-Z)
PTD4/TBCLK	General purpose I/O / Timer external input clock	Dual State	No	Input (Hi-Z)
PTD3–PTD0	General purpose I/O	Dual State	No	Input (Hi-Z)
PTE7/SPSCK	General purpose I/O / SPI clock	Dual State (open drain)	Yes	Input (Hi-Z)
PTE6/MOSI	General purpose I/O / SPI data path	Dual State (open drain)	Yes	Input (Hi-Z)
PTE5/MISO	General purpose I/O / SPI data path	Dual State (open drain)	Yes	Input (Hi-Z)
PTE4/$\overline{SS}$	General purpose I/O / SPI slave select	Dual State	Yes	Input (Hi-Z)
PTE3/TACH1	General purpose I/O / Timer A channel 1	Dual State	Yes	Input (Hi-Z)
PTE2/TACH0	General purpose I/O / Timer A channel 0	Dual State	Yes	Input (Hi-Z)
PTE1/RxD	General purpose I/O / SCI receive data	Dual State	Yes	Input (Hi-Z)
PTE0/TxD	General purpose I/O / SCI transmit data	Dual State	Yes	Input (Hi-Z)
PTF7–PTF6	General purpose I/O	Dual State	Yes	Input (Hi-Z)
PTF5/TBCH1	General purpose I/O / Timer B channel 1	Dual State	Yes	Input (Hi-Z)

Table 1-1. I/O Pins Summary

Pin Name	Function	Driver Type	Hysteresis	Reset State
PTF4/TBCH0	General purpose I/O / Timer B channel 0	Dual State	Yes	Input (Hi-Z)
PTF3/TBCH3	General purpose I/O / Timer B channel 3	Dual State	Yes	Input (Hi-Z)
PTF2/TBCH2	General purpose I/O / Timer B channel 2	Dual State	Yes	Input (Hi-Z)
PTF1/TACH3	General purpose I/O / Timer A channel 3	Dual State	Yes	Input (Hi-Z)
PTF0/TACH2	General purpose I/O / Timer A channel 2	Dual State	Yes	Input (Hi-Z)
PTG2/KBD2–PTG0/KBD0	General purpose I/O with key wakeup feature	Dual State	Yes	Input (Hi-Z)
PTH1/KBD4–PTH0/KBD3	General purpose I/O with key wakeup feature	Dual State	Yes	Input (Hi-Z)
V_{DD}	Logical chip power supply	NA	NA	NA
V_{SS}	Logical chip ground	NA	NA	NA
V_{DDA}	Analog power supply (CGM)	NA	NA	NA
V_{SSA}	Analog ground (CGM)	NA	NA	NA
V_{REFH}	ADC reference voltage	NA	NA	NA
A_{VSS}/V_{REFL}	ADC ground and reference voltage	NA	NA	NA
V_{DDAREF}	ADC power supply	NA	NA	NA
OSC1	External clock in	NA	NA	Input (Hi-Z)
OSC2	External clock out	NA	NA	Output
CGMXFC	PLL loop filter cap	NA	NA	NA
$\overline{IRQ}$	External interrupt request	NA	NA	Input (pullup)
$\overline{RST}$	Reset	NA	NA	Input (pullup)

Details of the clock connections to each of the modules on the MC68HC908AB32 are shown in **Table 1-2**. A short description of each clock source is also given in **Table 1-3**.

1.8 Signal Name Conventions

Table 1-2. Signal Name Conventions

Signal name	Description
CGMXCLK	Buffered version of OSC1 from clock generator module (CGM)
CGMOUT	PLL-based or OSC1-based clock output from CGM module)
Bus clock	CGMOUT divided by two
SPSCK	SPI serial clock (see **16.13.3 SPSCK (Serial Clock)**)
TACLK	External clock input for TIMA (see **11.9.1 TIMA Clock Pin**)
TBCLK	External clock input for TIMB (see **12.9.1 TIMB Clock Pin**)

1.9 Clock Source Summary

Table 1-3. Clock Source Summary

Module	Clock Source
ADC	CGMXCLK or bus clock
COP	CGMXCLK
CPU	Bus clock
EEPROM	CGMXCLK or bus clock
ROM	Bus clock
RAM	Bus clock
SPI	SPSCK
SCI	CGMXCLK
TIMA	Bus clock or PTD6/TACLK
TIMB	Bus clock or PTD4/TBCLK
PIT	Bus clock
KBI	Bus clock

Section 2. Memory Map

2.1 Contents

2.2 Introduction

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

- 32, 256 bytes of user FLASH memory
- 512 bytes of EEPROM
- 1024 bytes of random-access memory (RAM)
- 48 bytes of user-defined vectors
- 307 bytes of monitor ROM

2.3 Unimplemented Memory Locations

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

2.4 Reserved Memory Locations

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

2.5 Input/Output (I/O) Section

Most of the control, status, and data registers are in the zero page $0000–$004F. 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
- $FE08; FLASH control register, FLCR
- $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; EEPROM divider non-volatile register high, EEDIVHNVR
- $FE11; EEPROM divider non-volatile register low, EEDIVLNVR
- $FE1A; EEPROM timebase divider register high, EEDIVH
- $FE1B; EEPROM timebase divider register low, EEDIVL
- $FE1C; EEPROM non-volatile register, EENVR
- $FE1D; EEPROM control register, EECR
- $FE1F; EEPROM array configuration register, EEACR
- $FF7E; FLASH block protect register, FLBPR
- $FFFF; COP control register, COPCTL

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

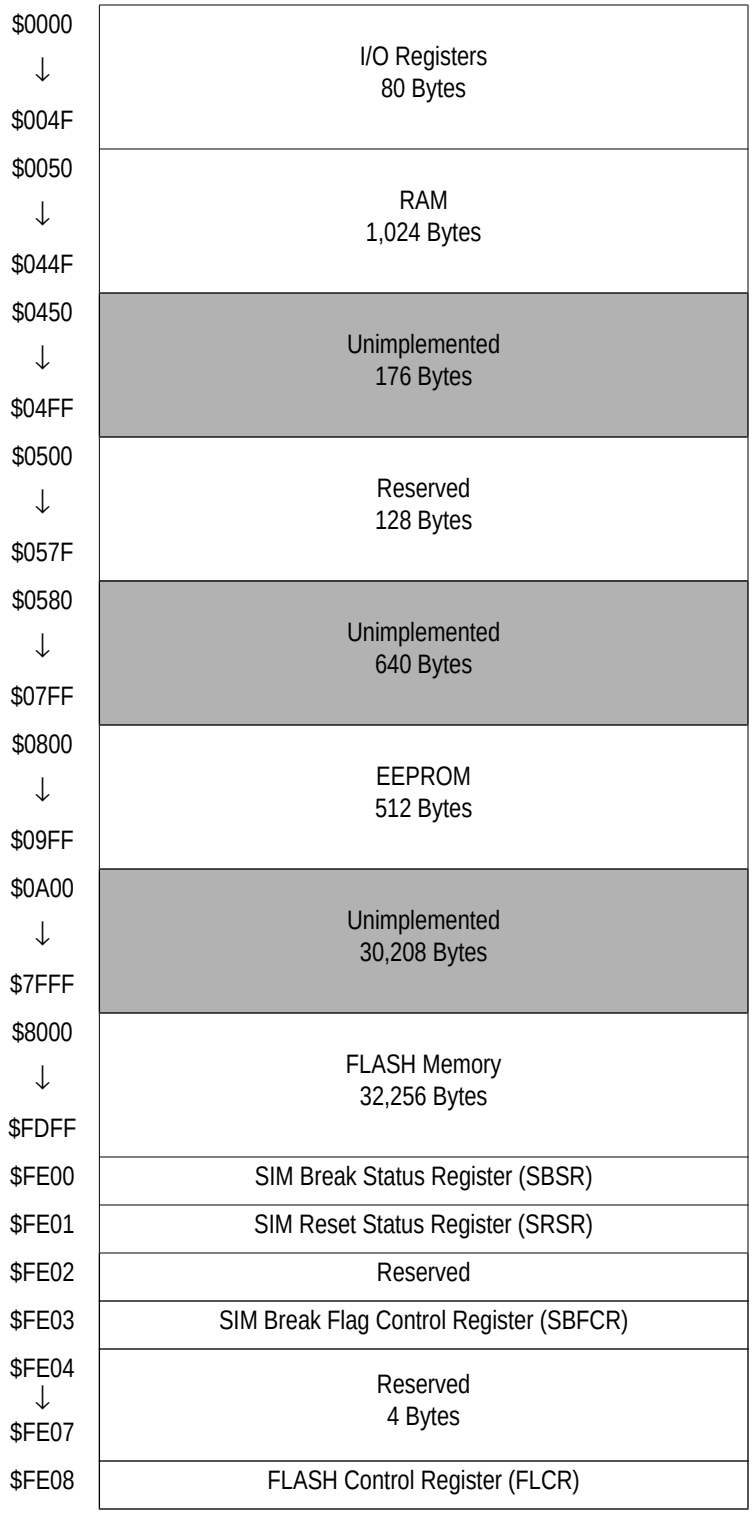


Figure 2-1. Memory Map

$FE09 ↓ $FE0B	Reserved 3 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	EEPROM Divider Non-volatile Register High (EEDIVHNVR)
$FE11	EEPROM Divider Non-volatile Register Low (EEDIVLNVR)
$FE12 ↓ $FE19	Reserved 8 Bytes
$FE1A	EEPROM Timebase Divider Register High (EEDIVH)
$FE1B	EEPROM Timebase Divider Register Low (EEDIVL)
$FE1C	EEPROM Non-volatile Register (EENVR)
$FE1D	EEPROM Control Register (EECR)
$FE1E	Reserved
$FE1F	EEPROM Array Configuration Register (EEACR)
$FE20 ↓ $FF52	Monitor ROM 307 Bytes
$FF53 ↓ $FF7D	Unimplemented 43 Bytes
$FF7E	FLASH Block Protect Register (FLBPR)
$FF7F ↓ $FFBF	Unimplemented 65 Bytes
$FFC0 ↓ $FFCF	Reserved FLASH Memory 16 Bytes Reserved for Compatibility with HC08AB16/24/32
$FFD0 ↓ $FFFF	FLASH Vectors 48 Bytes

Figure 2-1. Memory Map (Continued)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read:	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Write:								
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA)	Read: Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB)	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Reset:	0	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC)	Read:	MCLKEN	0	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD)	Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE)	Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Reset:	Unaffected by reset							
$0009	Port F Data Register (PTF)	Read: Write:	PTF7	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 1 of 11)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$000A	Port G Data Register (PTG)	Read:	0	0	0	0	0	PTG2	PTG1	PTG0
		Write:								
		Reset:	Unaffected by reset							
$000B	Port H Data Register (PTH)	Read:	0	0	0	0	0	0	PTH1	PTH0
		Write:								
		Reset:	Unaffected by reset							
$000C	Data Direction Register E (DDRE)	Read:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000D	Data Direction Register F (DDRF)	Read:	DDRF7	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000E	Data Direction Register G (DDRG)	Read:	0	0	0	0	0	DDRG2	DDRG1	DDRG0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000F	Data Direction Register H (DDRH)	Read:	0	0	0	0	0	0	DDRH1	DDRH0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0010	SPI Control Register (SPCR)	Read:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Write:								
		Reset:	0	0	1	0	1	0	0	0
$0011	SPI Status and Control Register (SPSCR)	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	1	0	0	0
$0012	SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0013	SCI Control Register 1 (SCC1)	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 2 of 11)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0014	SCI Control Register 2 (SCC2)	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0015	SCI Control Register 3 (SCC3)	Read:	R8	T8	R	R	ORIE	NEIE	FEIE	PEIE
		Write:								
		Reset:	Unaffected	Unaffected	0	0	0	0	0	0
$0016	SCI Status Register 1 (SCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$0017	SCI Status Register 2 (SCS2)	Read:	0	0	0	0	0	0	BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	SCI Data Register (SCDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0019	SCI Baud Rate Register (SCBR)	Read:	0	0	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001A	IRQ Status and Control Register (ISCR)	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0
$001B	Keyboard Status and Control Register (KBSCR)	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
		Write:						ACKK		
		Reset:	0	0	0	0	0	0	0	0
$001C	PLL Control Register (PCTL)	Read:	PLLIE	PLLF	PLLON	BCS	1	1	1	1
		Write:								
		Reset:	0	0	1	0	1	1	1	1
$001D	PLL Bandwidth Control Register (PBWC)	Read:	AUTO	LOCK	$\overline{ACQ}$	XLD	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 11)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001E	PLL Programming Register (PPG)	Read: Write:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
		Reset:	0	1	1	0	0	1	1	0
$001F	Configuration Register 1 (CONFIG1)†	Read: Write:	LVISTOP	R	LVIRSTD	LVIPWRD	SSREC	COPRS	STOP	COPD
		Reset:	0	0	0	0	0	0	0	0
$0020	Timer A Status and Control Register (TASC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0021	Keyboard Interrupt Enable Register (KBIER)	Read:	0	0	0	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0022	Timer A Counter Register High (TACNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0023	Timer A Counter Register Low (TACNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0024	Timer A Counter Modulo Register High (TAMODH)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	1	1	1	1	1	1	1	1
$0025	Timer A Counter Modulo Register Low (TAMODL)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	1	1	1	1	1	1	1	1
$0026	Timer A Channel 0 Status and Control Register (TASC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0027	Timer A Channel 0 Register High (TACH0H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							

† One-time writable register after each reset.

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 11)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0028	Timer A Channel 0 Register Low (TACH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0029	Timer A Channel 1 Status and Control Register (TASC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$002A	Timer A Channel 1 Register High (TACH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002B	Timer A Channel 1 Register Low (TACH1L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$002C	Timer A Channel 2 Status and Control Register (TASC2)	Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$002D	Timer A Channel 2 Register High (TACH2H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002E	Timer A Channel 2 Register Low (TACH2L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$002F	Timer A Channel 3 Status and Control Register (TASC3)	Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0030	Timer A Channel 3 Register High (TACH3H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0031	Timer A Channel 3 Register Low (TACH3L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 11)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0032	Timer B Channel 2 Status and Control Register (TBSC2)	Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0033	Timer B Channel 2 Register High (TBCH2H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0034	Timer B Channel 2 Register Low (TBCH2L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0035	Timer B Channel 3 Status and Control Register (TBSC3)	Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0036	Timer B Channel 3 Register High (TACH3H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0037	Timer B Channel 3 Register Low (TBCH3L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0038	Analog-to-Digital Status and Control Register (ADSCR)	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:								
		Reset:	0	0	0	1	1	1	1	1
$0039	Analog-to-Digital Data Register (ADR)	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$003A	Analog-to-Digital Clock Register (ADCLK)	Read:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$003B	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 11)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$003C	Reserved	Read: Write:	R	R	R	R	R	R	R	R
		Reset:								
$003D	Port D Input Pullup Enable Register (PTDPUE)	Read: Write:	PTDPUE7	PTDPUE6	PTDPUE5	PTDPUE4	PTDPUE3	PTDPUE2	PTDPUE1	PTDPUE0
		Reset:	0	0	0	0	0	0	0	0
$003E	Port F Input Pullup Enable Register (PTFPUE)	Read: Write:	PTFPUE7	PTFPUE6	PTFPUE5	PTFPUE4	PTFPUE3	PTFPUE2	PTFPUE1	PTFPUE0
		Reset:	0	0	0	0	0	0	0	0
$003F	Configuration Register 2 (CONFIG2)†	Read: Write:	R	EEDIVCLK	R	R	R	R	R	R
		Reset:		0						

† One-time writable register after each reset.

$0040	Timer B Status and Control Register (TBSC)	Read: Write:	TOF 0	TOIE	TSTOP	0 TRST	0	PS2	PS1	PS0
		Reset:	0	0	1	0	0	0	0	0
$0041	Timer B Counter Register High (TBCNTH)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$0042	Timer B Counter Register Low (TBCNTL)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0043	Timer B Counter Modulo Register High (TBMODH)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	1	1	1	1	1	1	1	1
$0044	Timer B Counter Modulo Register Low (TBMODL)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	1	1	1	1	1	1	1	1
$0045	Timer B Channel 0 Status and Control Register (TBSC0)	Read: Write:	CH0F 0	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 7 of 11)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0046	Timer B Channel 0 Register High (TBCH0H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0047	Timer B Channel 0 Register Low (TBCH0L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0048	Timer B Channel 1 Status and Control Register (TBSC1)	Read: Write:	CH1F 0	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Reset:	0	0	0	0	0	0	0	0
$0049	Timer B Channel 1 Register High (TBCH1H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$004A	Timer B Channel 1 Register Low (TBCH1L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$004B	PIT Status and Control Register (PSC)	Read: Write:	POF 0	POIE	PSTOP	0 PRST	0	PPS2	PPS1	PPS0
		Reset:	0	0	1	0	0	0	0	0
$004C	PIT Counter Register High (PCNTH)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$004D	PIT Counter Register Low (PCNTL)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$004E	PIT Counter Modulo Register High (PMODH)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	1	1	1	1	1	1	1	1
$004F	PIT Counter Modulo Register Low (PMODL)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	1	1	1	1	1	1	1	1

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 8 of 11)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	SIM Break Status Register (SBSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							Note	
		Reset:	0	0	0	0	0	0	0	0
$FE01	SIM Reset Status Register (SRSR)	Read:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
		Write:								
		POR:	1	0	0	0	0	0	0	0
$FE02	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE03	SIM Break Flag Control Register (SBFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE04	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								
$FE05	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								
$FE06	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								
$FE07	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								
$FE08	FLASH Control Register (FLCR)	Read:	0	0	0	0	HVEN	MASS	ERASE	PGM
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE09	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								

Note: Writing a logic 0 clears SBSW.

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 9 of 11)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE0A	Reserved	Read: Write:	R	R	R	R	R	R	R	R
		Reset:								
$FE0B	Reserved	Read: Write:	R	R	R	R	R	R	R	R
		Reset:								
$FE0C	Break Address Register High (BRKH)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$FE0D	Break Address Register Low (BRKL)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$FE0E	Break Status and Control Register (BRKSCR)	Read: Write:	BRKE	BRKA	0	0	0	0	0	0
		Reset:	0	0	0	0	0	0	0	0
$FE0F	Low-Voltage Inhibit Status Register (LVISR)	Read: Write:	LVIOUT	0	0	0	0	0	0	0
		Reset:	0	0	0	0	0	0	0	0
$FE10	EEDIV Non-volatile Register High (EEDIVHNVR)*	Read: Write:	EEDIVSECD	R	R	R	R	EEDIV10	EEDIV9	EEDIV8
		Reset:	Unaffected by reset; $FF when blank							
$FE11	EEDIV Non-volatile Register Low (EEDIVLNVR)*	Read: Write:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0
		Reset:	Unaffected by reset; $FF when blank							

* Non-volatile FLASH register; write by programming.

$FE1A	EE Divider Register High (EEDIVH)	Read: Write:	EEDIVSECD	R	R	R	R	EEDIV10	EEDIV9	EEDIV8
		Reset:	Contents of EEDIVHNVR ($FE10)							
$FE1B	EE Divider Register Low (EEDIVL)	Read: Write:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0
		Reset:	Contents of EEDIVLNVR ($FE11)							

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 10 of 11)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE1C	EEPROM Non-volatile Register (EENVR)*	Read: Write:	CON3	CON2	CON1	EEPRTCT	EEBP3	EEBP2	EEBP1	EEBP0
		Reset:	Unaffected by reset; $FF when blank							
$FE1D	EEPROM Control Register (EECR)	Read: Write:	EEDUM	0	EEOFF	EERAS1	EERAS0	EELAT	AUTO	EEPGM
		Reset:	0	0	0	0	0	0	0	0
$FE1E	Reserved	Read: Write:	R	R	R	R	R	R	R	R
		Reset:								
$FE1F	EEPROM Array Configuration Register (EEACR)	Read:	CON3	CON2	CON1	EEPRTCT	EEBP3	EEBP2	EEBP1	EEBP0
		Write:								
		Reset:	Contents of EENVR ($FE1C)							
$FF7E	FLASH Block Protect Register (FLBPR)*	Read: Write:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
		Reset:	Unaffected by reset							

* Non-volatile FLASH register; write by programming.

$FFFF	COP Control Register (COPCTL)	Read:	Low byte of reset vector
		Write:	Writing clears COP counter (any value)
		Reset:	Unaffected by reset

= Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 11 of 11)

Table 2-1. Vector Addresses

Vector Priority	Address	Vector
Lowest	$FFD0	ADC Conversion Complete Vector (High)
	$FFD1	ADC Conversion Complete Vector (Low)
	$FFD2	Keyboard Vector (High)
	$FFD3	Keyboard Vector (Low)
	$FFD4	SCI Transmit Vector (High)
	$FFD5	SCI Transmit Vector (Low)
	$FFD6	SCI Receive Vector (High)
	$FFD7	SCI Receive Vector (Low)
	$FFD8	SCI Error Vector (High)
	$FFD9	SCI Error Vector (Low)
	$FFDA	Reserved
	$FFDB	Reserved
	$FFDC	Reserved
	$FFDD	Reserved
	$FFDE	Timer B Channel 3 Vector (High)
	$FFDF	Timer B Channel 3 Vector (Low)
	$FFE0	Timer B Channel 2 Vector (High)
	$FFE1	Timer B Channel 2 Vector (Low)
	$FFE2	SPI Transmit Vector (High)
	$FFE3	SPI Transmit Vector (Low)
	$FFE4	SPI Receive Vector (High)
	$FFE5	SPI Receive Vector (Low)
	$FFE6	Timer B Overflow Vector (High)
	$FFE7	Timer B Overflow Vector (Low)
	$FFE8	Timer B Channel 1 Vector (High)
	$FFE9	Timer B Channel 1 Vector (Low)
	$FFEA	Timer B Channel 0 Vector (High)
	$FFEB	Timer B Channel 0 Vector (Low)
	$FFEC	Timer A Overflow Vector (High)
	$FFED	Timer A Overflow Vector (Low)
	$FFEE	Timer A Channel 3 Vector (High)
	$FFEF	Timer A Channel 3 Vector (Low)
	$FFF0	Timer A Channel 2 Vector (High)
	$FFF1	Timer A Channel 2 Vector (Low)
	$FFF2	Timer A Channel 1 Vector (High)
	$FFF3	Timer A Channel 1 Vector (Low)
	$FFF4	Timer A Channel 0 Vector (High)
	$FFF5	Timer A Channel 0 Vector (Low)
	$FFF6	Programmable Interrupt Timer (High)
	$FFF7	Programmable Interrupt Timer (Low)
	$FFF8	PLL Vector (High)
	$FFF9	PLL Vector (Low)
	$FFFA	IRQ Vector (High)
	$FFFB	IRQ Vector (Low)
	$FFFC	SWI Vector (High)
	$FFFD	SWI Vector (Low)
	$FFFE	Reset Vector (High)
Highest	$FFFF	Reset Vector (Low)

Section 3. Random-Access Memory (RAM)

3.1 Contents

3.2 Introduction

This section describes the 1024 bytes of RAM (random-access memory).

3.3 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 64K-byte 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 out of page zero, direct addressing mode instructions can efficiently access all page zero RAM locations. Page zero RAM, therefore, provides ideal locations for frequently accessed global variables.

Before processing an interrupt, the CPU uses five bytes of the stack to save the contents of the CPU registers.

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

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

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

Section 4. FLASH Memory

4.1 Contents

4.2 Introduction

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

4.3 Functional Description

The FLASH memory is an array of 32,256 bytes with an additional 48 bytes of user vectors and one byte of block protection. *An erased bit reads as logic 1 and a programmed bit reads as a logic 0*. Memory in the FLASH array is organized into two rows per page basis. For the 32K word by 8-Bit Embedded FLASH Memory, the page size is 128 bytes per

page. Hence the minimum erase page size is 128 bytes. Program and erase operations are facilitated through control bits in the FLASH Control Register (FLCR). Details for these operations appear later in this section. The address ranges for the user memory and vectors are:

- $8000–$FDFF; user memory.
- $FF7E; FLASH block protect register.
- $FE08; FLASH control register.
- $FFDC–$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 FLASH contents.*[1]

4.4 FLASH Control Register

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

Address: $FE08

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

Figure 4-1. FLASH Control Register (FLCR)

HVEN — High-Voltage Enable Bit

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

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

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

MASS — Mass Erase Control Bit

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

1 = MASS erase operation selected
0 = MASS erase operation unselected

ERASE — Erase Control Bit

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

1 = Erase operation selected
0 = Erase operation unselected

PGM — Program Control Bit

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

1 = Program operation selected
0 = Program operation unselected

4.5 FLASH Page Erase Operation

Use this step-by-step procedure to erase a page (128 bytes) of FLASH memory to read as logic 1:

1. Set the ERASE bit, and clear the MASS bit in the FLASH control register.
2. Read the FLASH block protect register.
3. Write any data to any FLASH address within the page address range desired.
4. Wait for a time, t_{nvs} (min. 10µs)
5. Set the HVEN bit.
6. Wait for a time, t_{Erase} (min. 1ms)
7. Clear the ERASE bit.
8. Wait for a time, t_{nvh} (min. 5µs)

9. Clear the HVEN bit.
10. After a time, t_{rcv} (typ. 1µs), the memory can be accessed again in read mode.

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

4.6 FLASH Mass Erase Operation

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

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

* When in Monitor mode, with security sequence failed (see **10.5 Security**), write to the FLASH block protect register instead of any FLASH address.

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

4.7 FLASH Program/Read Operation

Programming of the FLASH memory is done on a row basis. A row consists of 64 consecutive bytes starting from addresses $XX00, $XX40, $0080 and $XXC0. Use this step-by-step procedure to program a row of FLASH memory (**Figure 4-2** is a flowchart representation):

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

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

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

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

NOTE: *Programming and erasing of FLASH locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Do not exceed t_{PROG} maximum. See **23.13 FLASH Memory Characteristics**.*

4.8 FLASH Block Protection

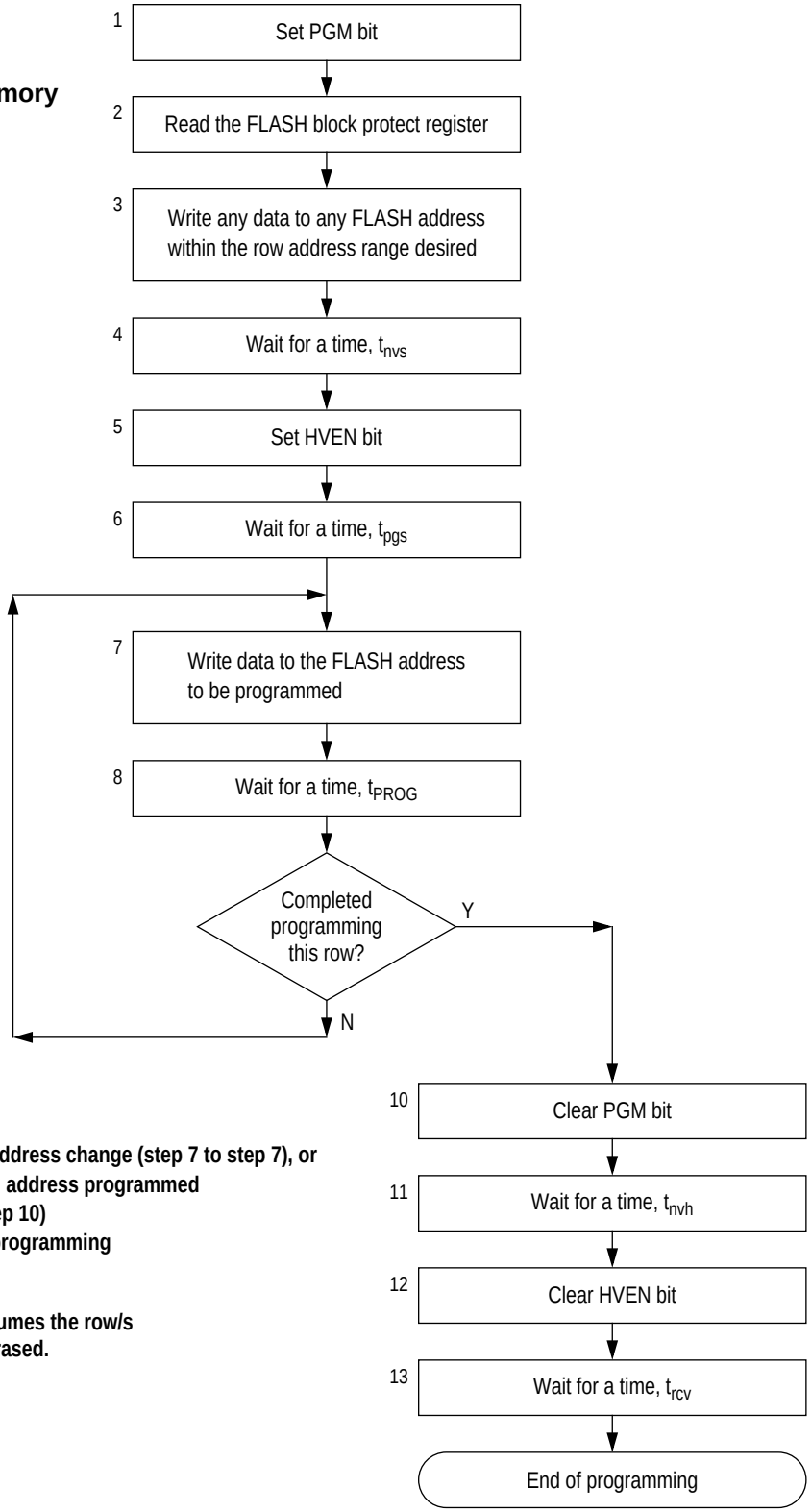
Due to the ability of the on-board charge pump to erase and program the FLASH memory in the target application, provision is made for protecting a block of memory from unintentional erase or program operations due to system malfunction. This protection is done by using of a FLASH Block Protect Register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

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

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

When bits within the FLBPR are programmed, they lock a block of memory, address ranges as shown in **4.8.1 FLASH Block Protect Register**. Once the FLBPR is programmed with a value other than $FF, any erase or program of the FLBPR or the protected block of FLASH memory is prohibited. The FLBPR itself can be erased or programmed only with an external voltage, V_{TST}, present on the $\overline{IRQ}$ pin. This voltage also allows entry from reset into the monitor mode.

Algorithm for programming a row (64 bytes) of FLASH memory



NOTE:

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

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

Figure 4-2. FLASH Programming Flowchart

4.8.1 FLASH Block Protect Register

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

Address: $FF7E

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

U = Unaffected by reset. Initial value from factory is 1.
Write to this register is by a programming sequence to the FLASH memory.

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

BPR[7:0] — FLASH Block Protect Bits

These eight bits represent bits [14:7] of a 16-bit memory address. Bit-15 is logic 1 and bits [6:0] are logic 0s.

The resultant 16-bit address is used for specifying the start address of the FLASH memory for block protection. The FLASH is protected from this start address to the end of FLASH memory, at $FFFF. With this mechanism, the protect start address can be XX00 and XX80 (128 bytes page boundaries) within the FLASH memory.

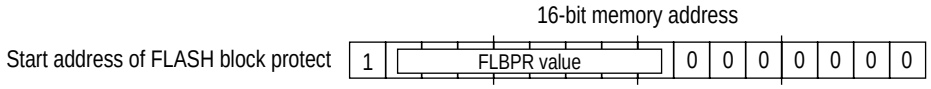


Figure 4-4. FLASH Block Protect Start Address

Examples of protect start address:

BPR[7:0]	Start of Address of Protect Range
$00	The entire FLASH memory is protected.
$01 (**0000 0001**)	$8080 (1**000 0000 1**000 0000)
$02 (**0000 0010**)	$8100 (1**000 0001 0**000 0000)
and so on...	
$FE (**1111 1110**)	$FF00 (1**111 1111 0**000 0000)
$FF	The entire FLASH memory is not protected.

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

4.9 Wait Mode

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

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

4.10 Stop Mode

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

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

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

Section 5. EEPROM

5.1 Contents

5.2 Introduction

This section describes the 512 bytes electrically erasable programmable read-only-memory (EEPROM).

5.3 Features

Features of the EEPROM include the following:

- 512 bytes non-volatile memory
- Byte, block or bulk erasable operations
- Non-volatile EEPROM configuration and block protection options
- On-chip charge pump for programming/erasing
- Security option

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE10	EEDIV Non-volatile Register High (EEDIVHNVR)*	Read:	EEDIVSECD	R	R	R	R	EEDIV10	EEDIV9	EEDIV8
		Write:								
		Reset:	Unaffected by reset; $FF when blank							
$FE11	EEDIV Non-volatile Register Low (EEDIVLNVR)*	Read:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0
		Write:								
		Reset:	Unaffected by reset; $FF when blank							
$FE1A	EE Divider Register High (EEDIVH)	Read:	EEDIVSECD	R	R	R	R	EEDIV10	EEDIV9	EEDIV8
		Write:								
		Reset:	Contents of EEDIVHNVR ($FE10)							
$FE1B	EE Divider Register Low (EEDIVL)	Read:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0
		Write:								
		Reset:	Contents of EEDIVLNVR ($FE11)							
$FE1C	EEPROM Non-volatile Register (EENVR)*	Read:	CON3	CON2	CON1	EEPRTCT	EEBP3	EEBP2	EEBP1	EEBP0
		Write:								
		Reset:	Unaffected by reset; $FF when blank; factory programmed $10							
$FE1D	EEPROM Control Register (EECR)	Read:	EEDUM	0	EEOFF	EERAS1	EERAS0	EELAT	AUTO	EEPGM
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE1F	EEPROM Array Configuration Register (EEACR)	Read:	CON3	CON2	CON1	EEPRTCT	EEBP3	EEBP2	EEBP1	EEBP0
		Write:								
		Reset:	Contents of EENVR ($FE1C)							

* Non-volatile EEPROM register; write by programming.

= Unimplemented R = Reserved

Figure 5-1. EEPROM I/O Register Summary

5.4 Functional Description

The 512 bytes of EEPROM is located at $0800–$09FF, and can be programmed or erased without an additional external high voltage supply. The program and erase operations are enabled through the use of an internal charge pump. For each byte of EEPROM, the write/erase endurance is 10,000 cycles.

5.5 EEPROM Configuration

The 8-bit EEPROM non-volatile register (EENVR) and the 16-bit EEPROM timebase divider non-volatile register (EEDIVNVR) contain the default settings for the following EEPROM configurations:

- Security option
- Block protection
- EEPROM timebase reference

EENVR and EEDIVNVR are non-volatile, EEPROM registers. They are programmed and erased in the same way as EEPROM bytes. The contents of these registers are loaded into their respective volatile registers during a MCU reset. The values in these read/write, volatile registers define the EEPROM configurations.

For EENVR, the corresponding volatile register is the EEPROM array configuration register (EEACR).

For the EEDIVNVR (two 8-bit registers: EEDIVHNVR and EEDIVLNVR), the corresponding volatile register is the EEPROM timebase divider register (EEDIV: EEDIVH and EEDIVL)

5.6 EEPROM Timebase Requirements

A 35µs timebase is required by the EEPROM control circuit for program and erase of EEPROM content. This timebase is derived from dividing the CGMXCLK or bus clock (selected by EEDIVCLK bit in CONFIG2 register) using a timebase divider circuit, controlled by the 16-bit EEPROM timebase divider register (EEDIVH and EEDIVL).

As the CGMXCLK or bus clock is user selected, the EEPROM timebase divider register must be configured with the appropriate value to obtain the 35µs. The timebase divider is calculated using the following formula:

$$\text{EEDIV} = \text{INT}\left[\text{Reference frequency (Hz)} \times 35 \times 10^{-6} + 0.5\right]$$

This value is written to the EEPROM timebase divider register (EEDIVH and EEDIVL) or programmed into the EEPROM timebase divider non-volatile register prior to any EEPROM program or erase operations (see **5.5 EEPROM Configuration** and **5.11.3.1 EEPROM Timebase Divider Non-Volatile Register**).

5.7 EEPROM Security Options

The EEPROM has a special security option, enabled by programming the EEPRTCT bit to 0 in the EEPROM non-volatile register (EENVR). Once security is enabled, the following limitations apply to the EEPROM:

- The 16-byte EEPROM locations from $08F0 to $08FF are protected from erase and program operations.
- The block erase and bulk erase modes are disabled. Byte erase can be used for all EEPROM locations except $08F0 to $08FF.
- The EENVR is protected from further erase or program operations.

5.8 EEPROM Block Protection

The 512 bytes of EEPROM is divided into four 128-byte blocks. Each of these blocks can be protected from erase/program operations by setting the EEBPx bit in the EENVR. **Table 5-1** shows the address ranges for the blocks.

Table 5-1. EEPROM Array Address Blocks

Block Number (EEBPx)	Address Range
EEBP0	$0800–$087F
EEBP1	$0880–$08FF
EEBP2	$0900–$097F
EEBP3	$0980–$09FF

These bits are effective after a reset or a read to EENVR register. The block protect configuration can be modified by erasing/programming the corresponding bits in the EENVR register and then reading the EENVR register.

5.9 EEPROM Programming and Erasing

The unprogrammed or erased state of an EEPROM bit is a logic 1. The factory default for the EEPROM array is $FF for all bytes.

The programming operation changes an EEPROM bit from logic 1 to logic 0 (programming cannot change a bit from logic 0 to a logic 1). In a single programming operation, the minimum EEPROM programming size is zero bits; the maximum is eight bits (one byte).

The erase operation changes an EEPROM bit from logic 0 to logic 1. In a single erase operation, the minimum EEPROM erase size is one byte; the maximum is the entire EEPROM array.

For each EEPROM byte, the write/erase endurance is 10,000 cycles. One write/erase cycle is defined as: *a maximum of eight programming operations on the same byte followed by an erase operation of the that byte*. Therefore, it is possible to program a byte, bit by bit to logic 0 before requiring an erase on that byte.

NOTE: *Although programming a bit (from 0 or 1) with a logic 1 does not change the state of that bit, it is still regarded as a programming operation. That is, if the same byte is programmed eight times (with any value), that byte must be erased before it can be successfully programmed again.*

5.9.1 EEPROM Programming

The unprogrammed or erased state of an EEPROM bit is a logic 1. Programming changes the state to a logic 0. Only EEPROM bytes in the non-protected blocks and the EENVR register can be programmed.

Use the following procedure to program a byte of EEPROM:

1. Clear EERAS1 and EERAS0, and set EELAT in the EECR.(A)
2. Write the desired data to the desired EEPROM address.(B)
3. Set the EEPGM bit.(C)
   Go to step 7 if AUTO is set.
4. Wait for a time, t_{EEPGM}, to program the byte.
5. Clear EEPGM bit.
6. Wait for a time, t_{EEFPV}, for the programming voltage to fall.
   Go to step 8.
7. Poll the EEPGM bit until it is cleared by the internal timer.(D)
8. Clear EELAT bit.(E)

NOTE: ***A.*** *EERAS1 and EERAS0 must be cleared for programming. Setting the EELAT bit configures the address and data buses to latch data for programming the array. Only data with a valid EEPROM address will be latched. If EELAT is set, other writes to the EECR will be allowed after a valid EEPROM write.*

B. *If more than one valid EEPROM writes occur, the last address and data will be latched, overriding the previous address and data. Once written data to the desired address, do not read EEPROM locations other than the written location. (Reading an EEPROM location returns the latched data, and causes the read address to be latched.)*

C. *The EEPGM bit cannot be set if the EELAT bit is cleared or a non-valid EEPROM address is latched. This is to ensure proper programming sequence. Once EEPGM is set, do not read any EEPROM locations, otherwise the current program cycle will be unsuccessful. When EEPGM is set, the on-board programming sequence will be activated.*

D. *The delay time for the EEPGM bit to be cleared in AUTO mode is less than t_{EEPGM}. However, on other MCUs, this delay time may be different. For forward compatibility, software should not make any dependency on this delay time.*

E. *Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM array.*

5.9.2 EEPROM Erasing

The programmed state of an EEPROM bit is logic 0. Erasing changes the state to a logic 1. Only EEPROM bytes in the non-protected blocks and EENVR register can be erased.

Use the following procedure to erase a byte, block, or the entire EEPROM:

1. Configure EERAS1 and EERAS0 for byte, block, or bulk erase; set EELAT in EECR.(A)
2. Byte erase: write any data to the desired address.(B)
   Block erase: write any data to an address within the desired block.(B)
   Bulk erase: write any data to an address within the array.(B)
3. Set the EEPGM bit.(C)
   Go to step 7 if AUTO is set.
4. Wait for a time: t_{EBYTE} for byte erase; t_{EBLOCK} for block erase; t_{EBULK} for bulk erase.
5. Clear EEPGM bit.
6. Wait for a time, t_{EEFPV}, for the erasing voltage to fall.
   Go to step 8.
7. Poll the EEPGM bit until it is cleared by the internal timer.(D)
8. Clear EELAT bits.(E)

NOTE: ***A.*** *Setting the EELAT bit configures the address and data buses to latch data for erasing the array. Only valid EEPROM addresses will be latched. If EELAT is set, other writes to the EECR will be allowed after a valid EEPROM write.*

B. *If more than one valid EEPROM writes occur, the last address and data will be latched, overriding the previous address and data. Once written data to the desired address, do not read EEPROM locations other than the written location. (Reading an EEPROM location returns the latched data, and causes the read address to be latched.) EENVR is not affected by block or bulk erase.*

C. *The EEPGM bit cannot be set if the EELAT bit is cleared or a non-valid EEPROM address is latched. This is to ensure proper programming sequence. Once EEPGM is set, do not read any EEPROM locations, otherwise the current erase cycle will be unsuccessful. When EEPGM is set, the erase mode cannot be changed, and the on-board erasing sequence will be activated.*

D. *The delay time for the EEPGM bit to be cleared in AUTO mode is less than t_{EBYTE} / t_{EBLOCK} / t_{EBULK}. However, on other MCUs, this delay time may be different. For forward compatibility, software should not make any dependency on this delay time.*

E. *Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM array.*

5.10 Low Power Modes

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

5.10.1 Wait Mode

The WAIT instruction does not affect the EEPROM. It is possible to start the program or erase sequence on the EEPROM and put the MCU in wait mode.

5.10.2 Stop Mode

The STOP instruction reduces the EEPROM power consumption to a minimum. The STOP instruction should not be executed while the programming and erasing sequence is in progress.

If stop mode is entered while EELAT and EEPGM is set, the programming sequence will be stopped and the programming voltage to the EEPROM array removed. The programming sequence will be restarted after leaving stop mode; access to the EEPROM is only possible after the programming sequence has completed.

If stop mode is entered while EELAT and EEPGM is cleared, the programming sequence will be terminated abruptly.

In either case, the data integrity of the EEPROM is not guaranteed.

5.11 EEPROM Registers

Four I/O registers and three non-volatile registers control program, erase, and options of the EEPROM array.

5.11.1 EEPROM Control Register

This read/write register controls programming/erasing of the EEPROM array.

Address: $FE1D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EEDUM	0	EEOFF	EERAS1	EERAS0	EELAT	AUTO	EEPGM
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 5-2. EEPROM Control Register (EECR)

EEDUM — Dummy Bit

This read/write bit has no function.

EEOFF — EEPROM Power-Off

This read/write bit disables the EEPROM module for lower power consumption. Any attempts to access the array will give unpredictable results. Reset clears this bit.

1 = Disable EEPROM array
0 = Enable EEPROM array

EERAS[1:0] — Erase/Program Mode Select Bits

These read/write bits set the erase modes. Reset clears these bits.

Table 5-2. EEPROM Program/Erase Mode Select

EEBPx	EERAS1	EERAS0	Mode
0	0	0	Byte Program
0	0	1	Byte Erase
0	1	0	Block Erase
0	1	1	Bulk Erase
1	X	X	No Erase/Program

X = don’t care

EELAT — EEPROM Latch Control

This read/write bit latches the address and data buses for programming the EEPROM array. EELAT can not be cleared if EEPGM is still set. Reset clears this bit.

1 = Buses configured for EEPROM program or erase operation
0 = Buses configured for normal operation

AUTO — Automatic termination of program/erase cycle

When AUTO is set, EEPGM is cleared automatically after the program/erase cycle is terminated by the internal timer.
(See note D for **5.9.1 EEPROM Programming** and **5.9.2 EEPROM Erasing**.)

0 = Automatic clear of EEPGM is disabled
1 = Automatic clear of EEPGM is enabled

EEPGM — EEPROM Program/Erase Enable

This read/write bit enables the internal charge pump and applies the programming/erasing voltage to the EEPROM array if the EELAT bit is set and a write to a valid EEPROM location has occurred. Reset clears the EEPGM bit.

1 = EEPROM programming/erasing power switched on
0 = EEPROM programming/erasing power switched off

NOTE: *Writing 0s to both the EELAT and EEPGM bits with a single instruction will only clear EEPGM. This is to allow time for the removal of high voltage.*

5.11.2 EEPROM Array Configuration Register

The EEPROM array configuration register configures EEPROM security and EEPROM block protection.

This read-only register is loaded with the contents of the EEPROM non-volatile register (EENVR) after a reset.

Address: $FE1F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CON3	CON2	CON1	EEPRTCT	EEBP3	EEBP2	EEBP1	EEBP0
Write:								
Reset:	Contents of EENVR ($FE1C)							

Figure 5-3. EEPROM Array Configuration Register (EEACR)

CON[3:1] — Unused

EEPRTCT — EEPROM Protection Bit

The EEPRTCT bit is used to enable the security feature in the EEPROM (see **5.7 EEPROM Security Options**).

1 = EEPROM security disabled
0 = EEPROM security enabled

EEBP[3:0] — EEPROM Block Protection Bits

These bits prevent blocks of EEPROM array from being programmed or erased.

1 = EEPROM array block is protected
0 = EEPROM array block is unprotected

Block Number (EEBPx)	Address Range
EEBP0	$0800–$087F
EEBP1	$0880–$08FF
EEBP2	$0900–$097F
EEBP3	$0980–$09FF

5.11.2.1 EEPROM Non-Volatile Register

The contents of this register is loaded into the EEPROM array configuration register (EEACR) after a reset.

This register is erased and programmed in the same way as an EEPROM byte.

Address: $FE1C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CON3	CON2	CON1	EEPRTCT	EEBP3	EEBP2	EEBP1	EEBP0
Write:								
Reset:	Unaffected by reset; $FF when blank; factory programmed $10							

Note: Non-volatile EEPROM register; write by programming.

Figure 5-4. EEPROM Non-Volatile Register (EENVR)

NOTE: *The EENVR is factory programmed with $10.*

5.11.3 EEPROM Timebase Divider Register

The 16-bit EEPROM timebase divider register consists of two 8-bit registers: EEDIVH and EEDIVL. The 11-bit value in this register is used to configure the timebase divider circuit to obtain the 35µs timebase for EEPROM control.

These two read/write registers are respectively loaded with the contents of the EEPROM timebase divider non-volatile registers (EEDIVHNVR and EEDIVLNVR) after a reset.

Address: $FE1A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EEDIVSECD	R	R	R	R	EEDIV10	EEDIV9	EEDIV8
Write:								
Reset:	Contents of EEDIVHNVR ($FE10)							

Figure 5-5. EEPROM Divider Register High (EEDIVH)

Address: $FE1B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0
Write:								
Reset:	Contents of EEDIVLNVR ($FE11)							

Figure 5-6. EEPROM Divider Register Low (EEDIVL)

EEDIVSECD — EEPROM Divider Security Disable

This bit enables/disables the security feature of the EEDIV registers. When EEDIV security feature is enabled, the state of the registers EEDIVH and EEDIVL are locked (including this EEDIVSECD bit). The EEDIVHNVR and EEDIVLNVR non-volatile memory registers are also protected from being erased/programmed.

1 = EEDIV security feature disabled
0 = EEDIV security feature enabled

EEDIV[10:0] — EEPROM Timebase Prescaler

These prescaler bits store the value of EEDIV which is used as the divisor to derive a timebase of 35µs from the selected reference clock source (CGMXCLK or bus clock, see **6.5 Configuration Register 2**) for the EEPROM related internal timer and circuits. EEDIV[10:0] bits are readable at any time. They are writable when EELAT=0 and EEDIVSECD=1.

The EEDIV value is calculated by the following formula:

$$EEDIV = INT\left[\text{Reference frequency (Hz)} \times 35 \times 10^{-6} + 0.5\right]$$

Where the result inside the bracket is rounded down to the nearest integer value.

For example, if the reference frequency is 4.9152MHz, the EEDIV value is 172.

NOTE: *Programming/erasing the EEPROM with an improper EEDIV value may result in data lost and reduce endurance of the EEPROM device.*

5.11.3.1 EEPROM Timebase Divider Non-Volatile Register

The 16-bit EEPROM timebase divider non-volatile register consists of two 8-bit registers: EEDIVHNVR and EEDIVLNVR. The contents of these two registers are respectively loaded into the EEPROM timebase divider registers, EEDIVH and EEDIVL, after a reset.

These two registers are erased and programmed in the same way as an EEPROM byte.

Address: $FE10

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EEDIVSECD	R	R	R	R	EEDIV10	EEDIV9	EEDIV8
Write:								
Reset:	Unaffected by reset; $FF when blank							

Figure 5-7. EEPROM Divider Non-volatile Register High (EEDIVHNVR)

Address: $FE11

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0
Write:								
Reset:	Unaffected by reset; $FF when blank							

Figure 5-8. EEPROM Divider Non-volatile Register Low (EEDIVLNVR)

These two registers are protected from erase and program operations if the EEDIVSECD is set to logic 1 in the EEDIVH (see **5.11.3 EEPROM Timebase Divider Register**), or programmed to a logic 1 in the EEDIVHNVR.

NOTE: *Once EEDIVSECD in the EEDIVHNVR is programmed to 0 and after a system reset, the EEDIV security feature is permanently enabled because the EEDIVSECD bit in the EEDIVH is always loaded with a 0 thereafter. Once this security feature is armed, erase and program operations are disabled for EEDIVHNVR and EEDIVLNVR. Modifications to the EEDIVH and EEDIVL registers are also disabled. Therefore, care should be taken before programming a value into the EEDIVHNVR.*

Section 6. Configuration Register (CONFIG)

6.1 Contents

6.2 Introduction

This section describes the configuration registers, CONFIG1 and CONFIG2. The configuration registers enable or disable these options:

- Low-voltage inhibit (LVI) in stop mode
- LVI reset
- LVI module power
- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- COP timeout period ($2^{18} – 2^4$ or $2^{13} – 2^4$ CGMXCLK cycles)
- STOP instruction
- Computer operating properly module (COP)
- EEPROM reference clock source (CPU bus clock or CGMXCLK)

6.3 Functional description

The configuration registers are used in the initialization of various options. The configuration registers can be written once after each reset. All of the configuration register bits are cleared during reset. Since the various options affect the operation of the MCU, it is recommended that these registers be written immediately after reset. The configuration registers are located at $001F and $003F. The configuration register may be read at anytime.

6.4 Configuration Register 1

Address: $001F

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

R = Reserved

Figure 6-1. Configuration Register 1 (CONFIG1)

LVISTOP — LVI Enable in Stop Mode Bit

When the LVIPWRD bit is clear, setting the LVISTOP bit enables the LVI to operate in stop mode. Reset clears LVISTOP. (See **Section 21. Low-Voltage Inhibit (LVI)**.)

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

LVIRSTD — LVI Reset Disable Bit

LVIRSTD disables the reset signal from the LVI module. (See **Section 21. Low-Voltage Inhibit (LVI)**.)

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

LVIPWRD — LVI Power Disable Bit

LVIPWRD disables the LVI module. (See **Section 21. Low-Voltage Inhibit (LVI)**.)

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

SSREC — Short Stop Recovery Bit

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

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

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

COPRS — COP Rate Select Bit

COPRS selects the COP timeout period. Reset clears COPRS. (See **Section 20. Computer Operating Properly (COP)**.)

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

STOP — STOP Instruction Enable Bit

STOP enables the STOP instruction.

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

COPD — COP Disable Bit

COPD disables the COP module. (See **Section 20. Computer Operating Properly (COP)**.)

1 = COP module disabled
0 = COP module enabled

Extra care should be exercised when using this emulation part for development of code to be run in ROM AB, AS or AZ parts that the options selected by setting the CONFIG1 register match exactly the options selected on any ROM code request submitted. The enable/disable logic is not necessarily identical in all parts of the AB, AS, and AZ families. If in doubt, check with your local field applications representative.

6.5 Configuration Register 2

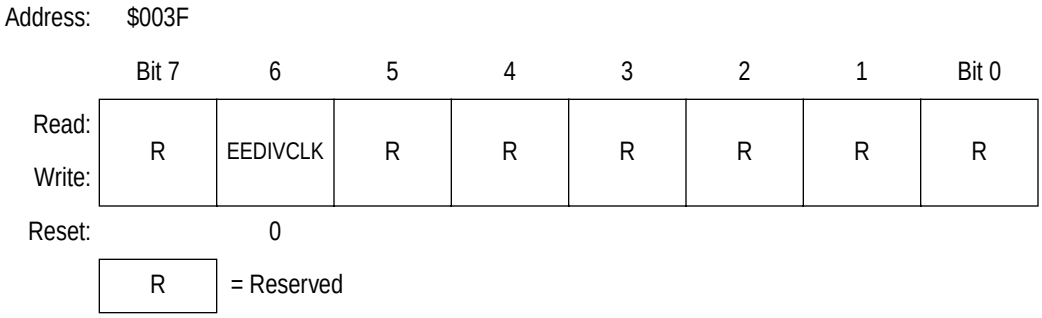


Figure 6-2. Configuration Register 2 (CONFIG2)

EEDIVCLK — EEPROM Timebase Divider Clock Select Bit

EEDIVCLK selects the reference clock source for the EEPROM timebase divider. (See **Section 5. EEPROM**.)

1 = CPU bus clock drives the EEPROM timebase divider
0 = CGMXCLK drives the EEPROM timebase divider

Extra care should be exercised when using this emulation part for development of code to be run in ROM AB, AS or AZ parts that the options selected by setting the CONFIG2 register match exactly the options selected on any ROM code request submitted. The enable/disable logic is not necessarily identical in all parts of the AB, AS, and AZ families. If in doubt, check with your local field applications representative.

Section 7. Central Processor Unit (CPU)

7.1 Contents

7.2 Introduction

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

7.3 Features

- Object code fully upward-compatible with M68HC05 Family
- 16-bit stack pointer with stack manipulation instructions
- 16-bit index register with x-register manipulation instructions
- 8-MHz CPU internal bus frequency
- 64K-byte 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 64K-bytes
- Low-power stop and wait modes

7.4 CPU Registers

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

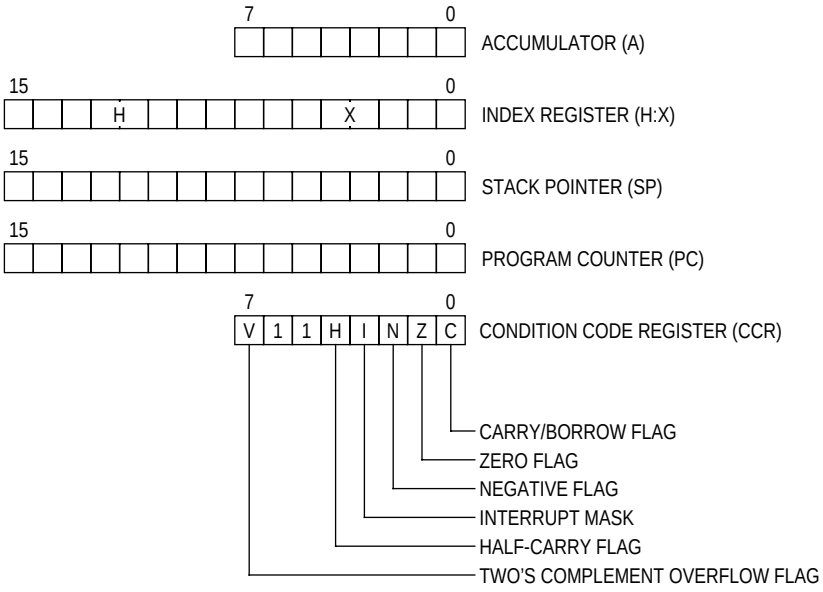


Figure 7-1. CPU Registers

7.4.1 Accumulator

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

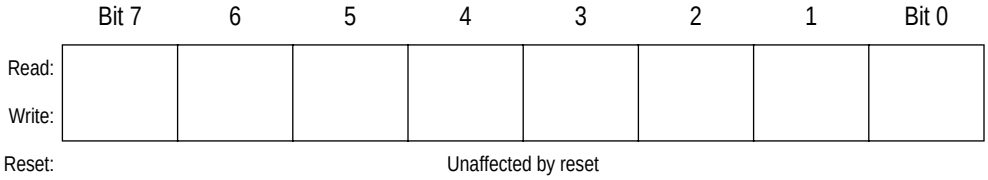


Figure 7-2. Accumulator (A)

7.4.2 Index Register

The 16-bit index register allows indexed addressing of a 64K-byte memory space. H is the upper byte of the index register, and X is the lower byte. H:X is the concatenated 16-bit index register.

In the indexed addressing modes, the CPU uses the contents of the index register to determine the conditional address of the operand.

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

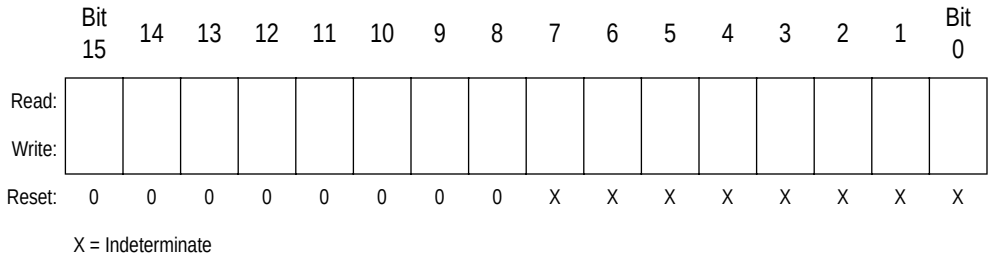


Figure 7-3. Index Register (H:X)

7.4.3 Stack Pointer

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

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

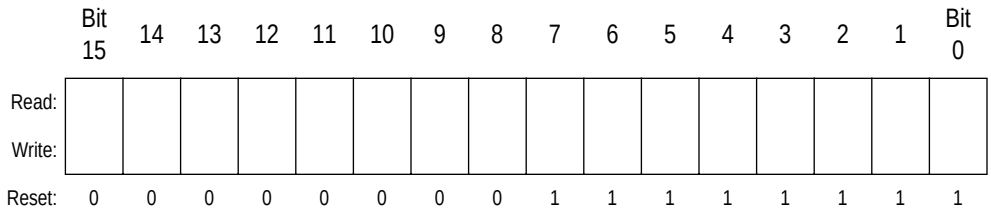


Figure 7-4. Stack Pointer (SP)

NOTE: *The location of the stack is arbitrary and may be relocated anywhere in RAM. Moving the SP out of page 0 ($0000 to $00FF) frees direct address (page 0) space. For correct operation, the stack pointer must point only to RAM locations.*

7.4.4 Program Counter

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

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

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

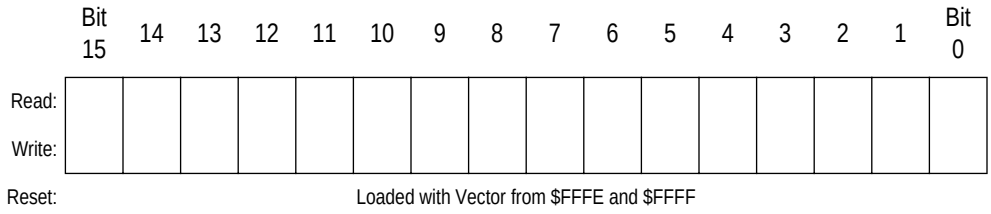


Figure 7-5. Program Counter (PC)

7.4.5 Condition Code Register

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

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

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

X = Indeterminate

Figure 7-6. Condition Code Register (CCR)

V — Overflow Flag

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

1 = Overflow
0 = No overflow

H — Half-Carry Flag

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

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

I — Interrupt Mask

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

1 = Interrupts disabled
0 = Interrupts enabled

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

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

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

N — Negative flag

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

1 = Negative result
0 = Non-negative result

Z — Zero flag

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

1 = Zero result
0 = Non-zero result

C — Carry/Borrow Flag

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

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

7.5 Arithmetic/Logic Unit (ALU)

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

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

7.6 Low-Power Modes

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

7.6.1 Wait Mode

The WAIT instruction:

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

7.6.2 Stop Mode

The STOP instruction:

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

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

7.7 CPU During Break Interrupts

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

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

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

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

7.8 Instruction Set Summary

7.9 Opcode Map

See **Table 7-2**.

Table 7-1. Instruction Set Summary

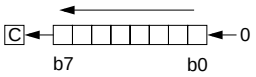
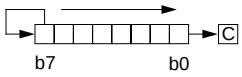
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X ADC *opr*,SP ADC *opr*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9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

Table 7-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BEQ *rel*	Branch if Equal	PC ← (PC) + 2 + *rel* ? (Z) = 1	–	–	–	–	–	–	REL	27	rr	3
BGE *opr*	Branch if Greater Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) = 0	–	–	–	–	–	–	REL	90	rr	3
BGT *opr*	Branch if Greater Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) | (N ⊕ V) = 0	–	–	–	–	–	–	REL	92	rr	3
BHCC *rel*	Branch if Half Carry Bit Clear	PC ← (PC) + 2 + *rel* ? (H) = 0	–	–	–	–	–	–	REL	28	rr	3
BHCS *rel*	Branch if Half Carry Bit Set	PC ← (PC) + 2 + *rel* ? (H) = 1	–	–	–	–	–	–	REL	29	rr	3
BHI *rel*	Branch if Higher	PC ← (PC) + 2 + *rel* ? (C) | (Z) = 0	–	–	–	–	–	–	REL	22	rr	3
BHS *rel*	Branch if Higher or Same (Same as BCC)	PC ← (PC) + 2 + *rel* ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if $\overline{\text{IRQ}}$ Pin High	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if $\overline{\text{IRQ}}$ Pin Low	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #*opr* BIT *opr* BIT *opr* BIT *opr*,X BIT *opr*,X BIT ,X BIT *opr*,SP BIT *opr*,SP	Bit Test	(A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A5 B5 C5 D5 E5 F5 9EE5 9ED5	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
BLE *opr*	Branch if Less Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) | (N ⊕ V) = 1	–	–	–	–	–	–	REL	93	rr	3
BLO *rel*	Branch if Lower (Same as BCS)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BLS *rel*	Branch if Lower or Same	PC ← (PC) + 2 + *rel* ? (C) | (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT *opr*	Branch if Less Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) =1	–	–	–	–	–	–	REL	91	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	PC ← (PC) + 2 + *rel* ? (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI *rel*	Branch if Minus	PC ← (PC) + 2 + *rel* ? (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	PC ← (PC) + 2 + *rel* ? (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + 2 + *rel* ? (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL *rel*	Branch if Plus	PC ← (PC) + 2 + *rel* ? (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA *rel*	Branch Always	PC ← (PC) + 2 + *rel*	–	–	–	–	–	–	REL	20	rr	3

Table 7-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BRCLR *n*,*opr*,*rel*	Branch if Bit *n* in M Clear	PC ← (PC) + 3 + *rel* ? (Mn) = 0	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	PC ← (PC) + 2	–	–	–	–	–	–	REL	21	rr	3
BRSET *n*,*opr*,*rel*	Branch if Bit *n* in M Set	PC ← (PC) + 3 + *rel* ? (Mn) = 1	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n*,*opr*	Set Bit *n* in M	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	4
CBEQ *opr*,*rel* CBEQA #*opr*,*rel* CBEQX #*opr*,*rel* CBEQ *opr*,X+,*rel* CBEQ X+,*rel* CBEQ *opr*,SP,*rel*	Compare and Branch if Equal	PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (X) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 2 + rel ? (A) – (M) = $00 PC ← (PC) + 4 + rel ? (A) – (M) = $00	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 4 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	I ← 0	–	–	0	–	–	–	INH	9A		2
CLR *opr* CLRA CLRX CLRH CLR *opr*,X CLR ,X CLR *opr*,SP	Clear	M ← $00 A ← $00 X ← $00 H ← $00 M ← $00 M ← $00 M ← $00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	3 1 1 1 3 2 4

Table 7-1. Instruction Set Summary (Continued)

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

Table 7-1. Instruction Set Summary (Continued)

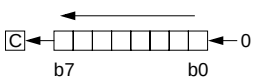
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X EOR *opr*,SP EOR *opr*,SP	Exclusive OR M with A	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9EE8 9ED8	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
INC *opr* INCA INCX INC *opr*,X INC ,X INC *opr*,SP	Increment	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1 M ← (M) + 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	4 1 1 4 3 5
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff 	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR ,X	Jump to Subroutine	PC ← (PC) + *n* (*n* = 1, 2, or 3) Push (PCL); SP ← (SP) – 1 Push (PCH); SP ← (SP) – 1 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff 	4 5 6 5 4
LDA #*opr* LDA *opr* LDA *opr* LDA *opr*,X LDA *opr*,X LDA ,X LDA *opr*,SP LDA *opr*,SP	Load A from M	A ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A6 B6 C6 D6 E6 F6 9EE6 9ED6	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LDHX #*opr* LDHX *opr*	Load H:X from M	H:X ← (M:M + 1)	0	–	–	↕	↕	–	IMM DIR	45 55	ii jj dd	3 4
LDX #*opr* LDX *opr* LDX *opr* LDX *opr*,X LDX *opr*,X LDX ,X LDX *opr*,SP LDX *opr*,SP	Load X from M	X ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AE BE CE DE EE FE 9EEE 9EDE	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LSL *opr* LSLA LSLX LSL *opr*,X (Same as ASL) LSL ,X LSL *opr*,SP	Logical Shift Left  		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5

Table 7-1. Instruction Set Summary (Continued)

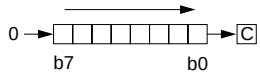
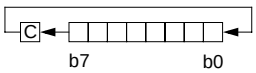
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LSR *opr* LSRA LSR*X* LSR *opr*,X LSR ,X LSR *opr*,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	4 1 1 4 3 5
MOV *opr,opr* MOV *opr*,X+ MOV #*opr,opr* MOV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

Table 7-1. Instruction Set Summary (Continued)

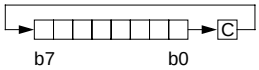
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ROR *opr* RORA RORX ROR *opr*,X ROR ,X ROR *opr*,SP	Rotate Right through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	4 1 1 4 3 5
RSP	Reset Stack Pointer	SP ← $FF	–	–	–	–	–	–	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 1; Pull (CCR) SP ← (SP) + 1; Pull (A) SP ← (SP) + 1; Pull (X) SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	↕	↕	↕	↕	↕	↕	INH	80		7
RTS	Return from Subroutine	SP ← SP + 1; Pull (PCH) SP ← SP + 1; Pull (PCL)	–	–	–	–	–	–	INH	81		4
SBC #*opr* SBC *opr* SBC *opr* SBC *opr*,X SBC *opr*,X SBC ,X SBC *opr*,SP SBC *opr*,SP	Subtract with Carry	A ← (A) – (M) – (C)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A2 B2 C2 D2 E2 F2 9EE2 9ED2	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
SEC	Set Carry Bit	C ← 1	–	–	–	–	–	1	INH	99		1
SEI	Set Interrupt Mask	I ← 1	–	–	1	–	–	–	INH	9B		2
STA *opr* STA *opr* STA *opr*,X STA *opr*,X STA ,X STA *opr*,SP STA *opr*,SP	Store A in M	M ← (A)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	B7 C7 D7 E7 F7 9EE7 9ED7	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 5
STHX *opr*	Store H:X in M	(M:M + 1) ← (H:X)	0	–	–	↕	↕	–	DIR	35	dd	4
STOP	Enable $\overline{\text{IRQ}}$ Pin; Stop Oscillator	I ← 0; Stop Oscillator	–	–	0	–	–	–	INH	8E		1
STX *opr* STX *opr* STX *opr*,X STX *opr*,X STX ,X STX *opr*,SP STX *opr*,SP	Store X in M	M ← (X)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	BF CF DF EF FF 9EEF 9EDF	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 5

Table 7-1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
SUB #*opr* SUB *opr* SUB *opr* SUB *opr*,X SUB *opr*,X SUB ,X SUB *opr*,SP SUB *opr*,SP	Subtract	A ← (A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A0 B0 C0 D0 E0 F0 9EE0 9ED0	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
SWI	Software Interrupt	PC ← (PC) + 1; Push (PCL) SP ← (SP) – 1; Push (PCH) SP ← (SP) – 1; Push (X) SP ← (SP) – 1; Push (A) SP ← (SP) – 1; Push (CCR) SP ← (SP) – 1; I ← 1 PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	–	–	1	–	–	–	INH	83		9
TAP	Transfer A to CCR	CCR ← (A)	↕	↕	↕	↕	↕	↕	INH	84		2
TAX	Transfer A to X	X ← (A)	–	–	–	–	–	–	INH	97		1
TPA	Transfer CCR to A	A ← (CCR)	–	–	–	–	–	–	INH	85		1
TST *opr* TSTA TSTX TST *opr*,X TST ,X TST *opr*,SP	Test for Negative or Zero	(A) – $00 or (X) – $00 or (M) – $00	0	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	3 1 1 3 2 4
TSX	Transfer SP to H:X	H:X ← (SP) + 1	–	–	–	–	–	–	INH	95		2
TXA	Transfer X to A	A ← (X)	–	–	–	–	–	–	INH	9F		1
TXS	Transfer H:X to SP	(SP) ← (H:X) – 1	–	–	–	–	–	–	INH	94		2

Table 7-1. Instruction Set Summary (Continued)

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

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

Table 7-2. Opcode Map

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

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

*Pre-byte for stack pointer indexed instructions

Low Byte of Opcode in Hexadecimal

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

Section 8. System Integration Module (SIM)

8.1 Contents

8.2 Introduction

This section describes the system integration module (SIM), which supports up to 24 external and/or internal interrupts. Together with the CPU, the SIM controls all MCU activities. A block diagram of the SIM is shown in **Figure 8-1**. **Figure 8-2** 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

Table 8-1 shows the internal signal names used in this section.

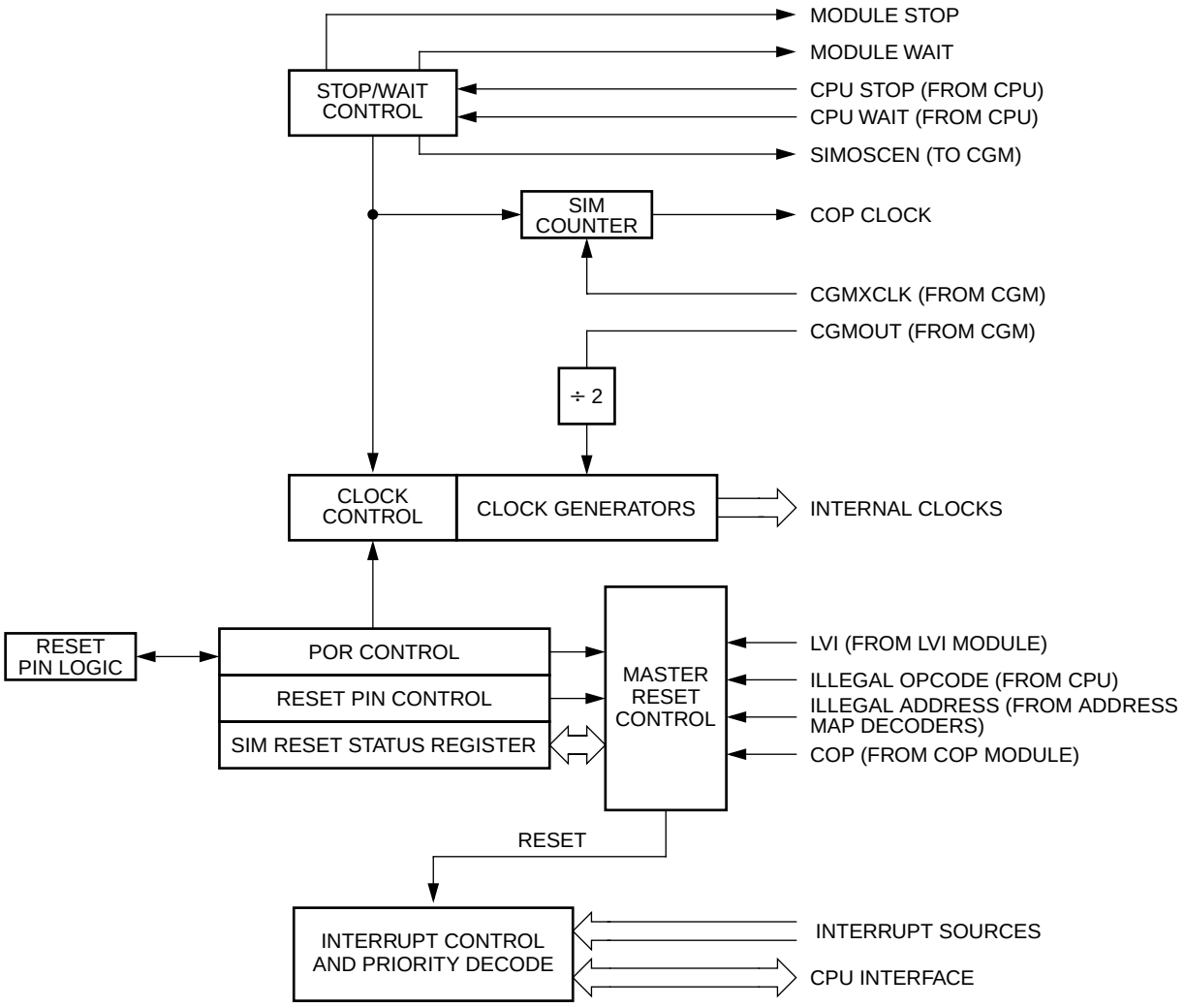


Figure 8-1. SIM Block Diagram

Table 8-1. Signal naming 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

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

= Unimplemented R = Reserved

Figure 8-2. SIM I/O Register Summary

8.3 SIM Bus Clock Control and Generation

The bus clock generator provides system clock signals for the CPU and peripherals on the MCU. The system clocks are generated from an incoming clock, CGMOUT, as shown in **Figure 8-3**. This clock can come from either an external oscillator or from the on-chip PLL.
See **Section 9. Clock Generator Module (CGM)**.

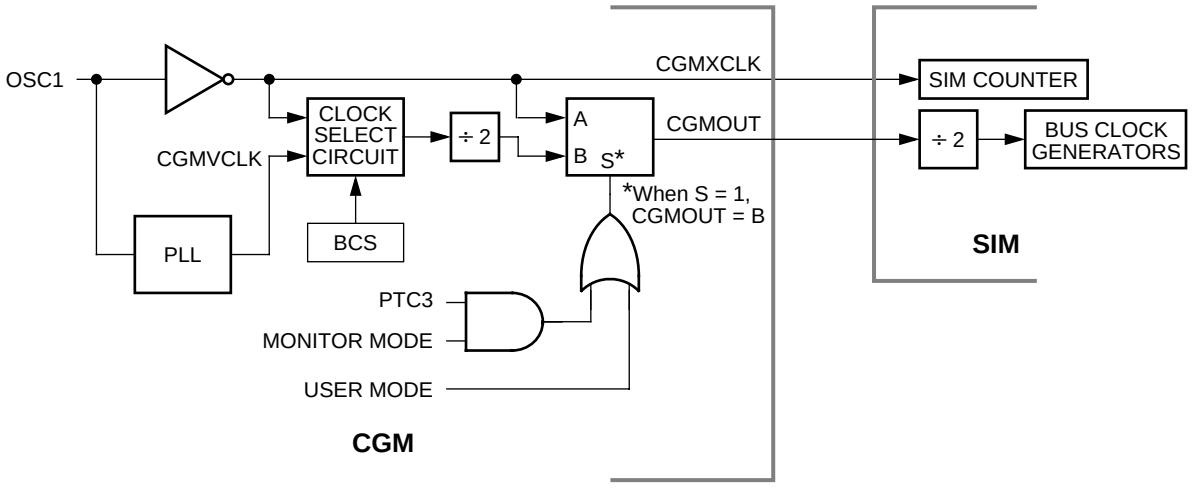


Figure 8-3. CGM Clock Signals

8.3.1 Bus Timing

In user mode, the internal bus frequency is either the crystal oscillator output (CGMXCLK) divided by four or the PLL output (CGMVCLK) divided by four. See **Section 9. Clock Generator Module (CGM)**.

8.3.2 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 been 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.

8.3.3 Clocks in Stop and Wait Modes

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 **8.7.2 Stop Mode**.)

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

8.4 Reset and System Initialization

The MCU has the following reset sources:

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

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

An internal reset clears the SIM counter (see **8.5 SIM Counter**), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See **8.8 SIM Registers**.)

8.4.1 External Pin Reset

Pulling the asynchronous $\overline{\text{RST}}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{\text{RST}}$ is held low for a minimum of 67 CGMXCLK cycles, assuming that neither the POR nor the LVI was the source of the reset. See **Table 8-2** for details. **Figure 8-4** shows the relative timing.

Table 8-2. PIN Bit Set Timing

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

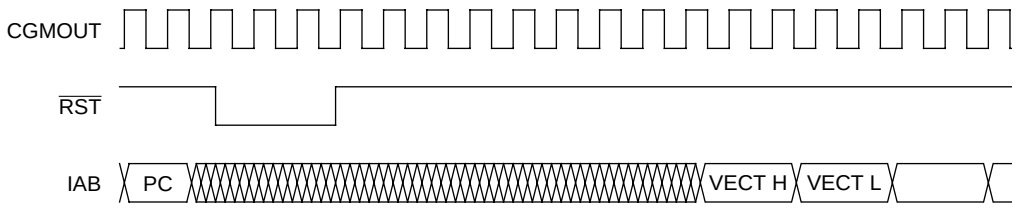


Figure 8-4. External Reset Timing

8.4.2 Active Resets from Internal Sources

All internal reset sources actively pull the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles to allow for resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles. See **Figure 8-5**. An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR. See **Figure 8-6**. Note that for LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles, during which the SIM forces the $\overline{\text{RST}}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{\text{RST}}$ as shown in **Figure 8-5**.

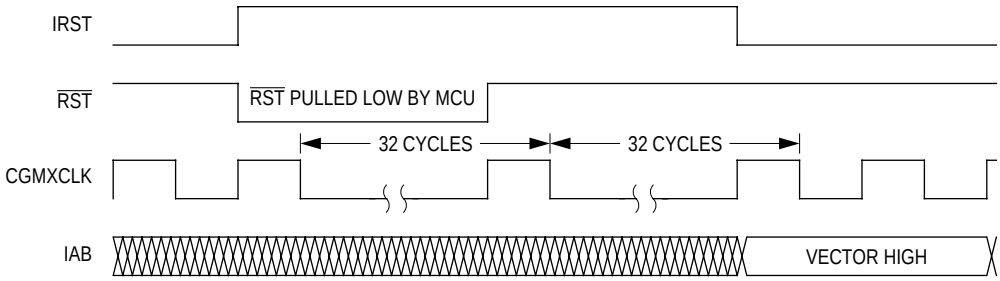


Figure 8-5. Internal reset timing

The COP reset is asynchronous to the bus clock.

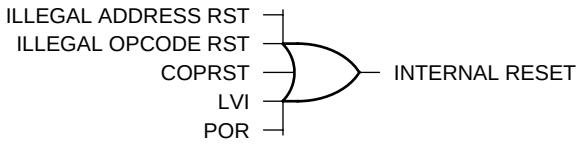


Figure 8-6. Sources of Internal Reset

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

8.4.2.1 Power-On Reset

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

At power-on, the following events occur:

- A POR pulse is generated
- The internal reset signal is asserted
- The SIM enables CGMOUT
- Internal clocks to the CPU and modules are held inactive for 4096 CGMXCLK cycles to allow the oscillator to stabilize

- 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

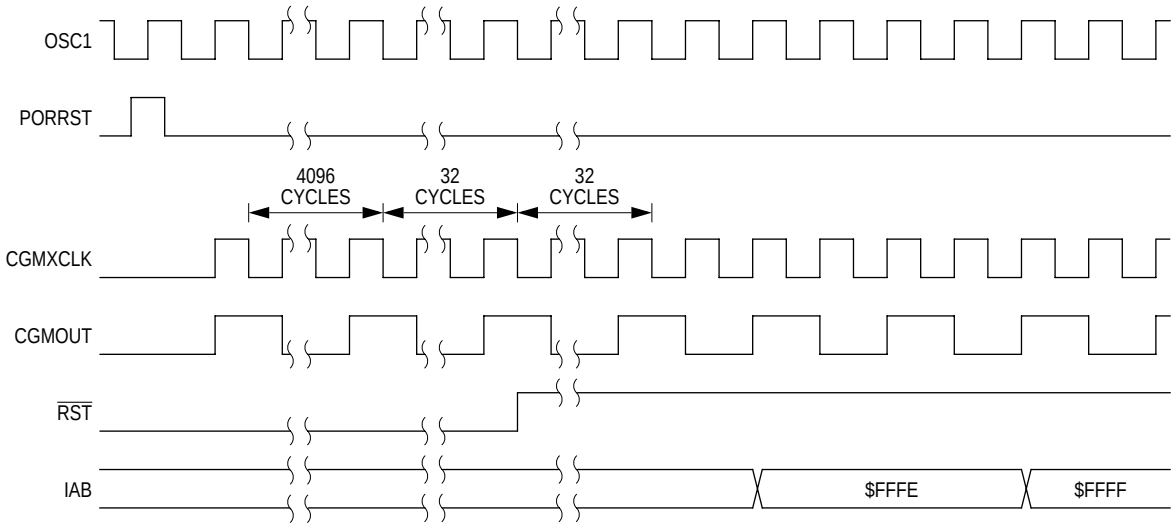


Figure 8-7. POR Recovery

8.4.2.2 Computer Operating Properly (COP) Reset

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

To prevent a COP module timeout, a value (any value) should be written 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 $\overline{IRQ}$ pin is held at V_{TST} while the MCU is in monitor mode. The COP module can be disabled only through combinational logic conditioned with the high voltage signal on the $\overline{RST}$ or the $\overline{IRQ}$ pin. This prevents the COP from becoming disabled as a result of external noise. During a break state, V_{TST} on the $\overline{RST}$ pin disables the COP module.

8.4.2.3 Illegal Opcode Reset

The SIM decodes signals from the CPU to detect illegal instructions. An illegal instruction sets the ILOP bit in the SIM reset status register (SRSR) and causes a reset.

If the STOP enable bit, STOP, in the configuration register 1 (CONFIG1) is logic 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

8.4.2.4 Illegal Address Reset

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

8.4.2.5 Low-Voltage Inhibit (LVI) Reset

The low-voltage inhibit module (LVI) asserts its output to the SIM when the V_{DD} voltage falls to the trip voltage, V_{LVIl}. The LVI bit in the SIM reset status register (SRSR) is set, and the external reset pin ($\overline{\text{RST}}$) is held low while the SIM counter counts out 4096 CGMXCLK cycles. 64 CGMXCLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

8.5 SIM Counter

The SIM counter is used by the power-on reset module (POR) and in stop mode recovery to allow the oscillator time to stabilize before enabling the internal bus (IBUS) clocks. The SIM counter also serves as a prescaler for the computer operating properly (COP) module. 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.

8.5.1 SIM Counter during Power-On Reset

The power-on reset (POR) module 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.

8.5.2 SIM Counter during Stop Mode Recovery

The SIM counter is also used for stop mode recovery. The STOP instruction clears the SIM counter. After an interrupt, break, or reset, the SIM senses the state of the short STOP recovery bit, SSREC, in the configuration register 1 (CONFIG1). If the SSREC bit is a logic 1, 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.

8.5.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See **8.7.2 Stop Mode** for details.) The SIM counter is free-running after all reset states. (See **8.4.2 Active Resets from Internal Sources** for counter control and internal reset recovery sequences.)

8.6 Exception Control

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

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

8.6.1 Interrupts

At the beginning of an interrupt, the CPU saves the CPU register contents onto 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 8-8** shows interrupt entry timing, and **Figure 8-9** shows interrupt recovery timing.

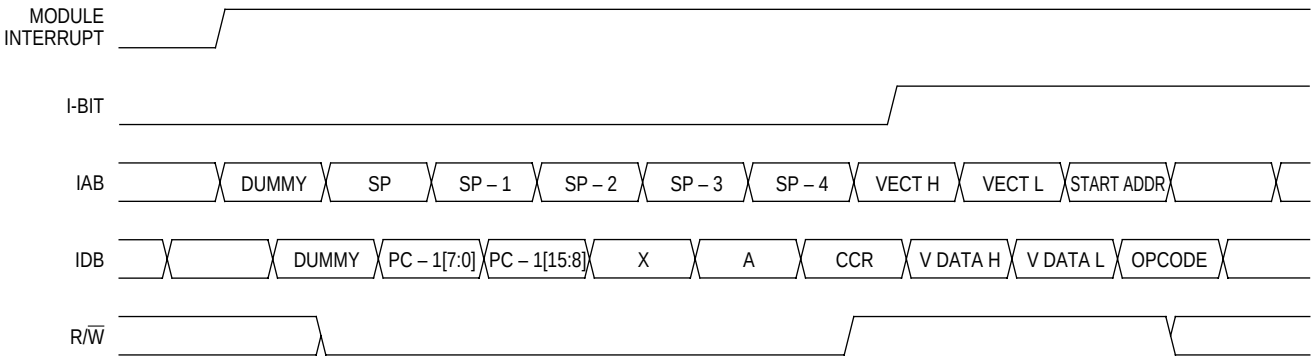


Figure 8-8. Interrupt Entry Timing

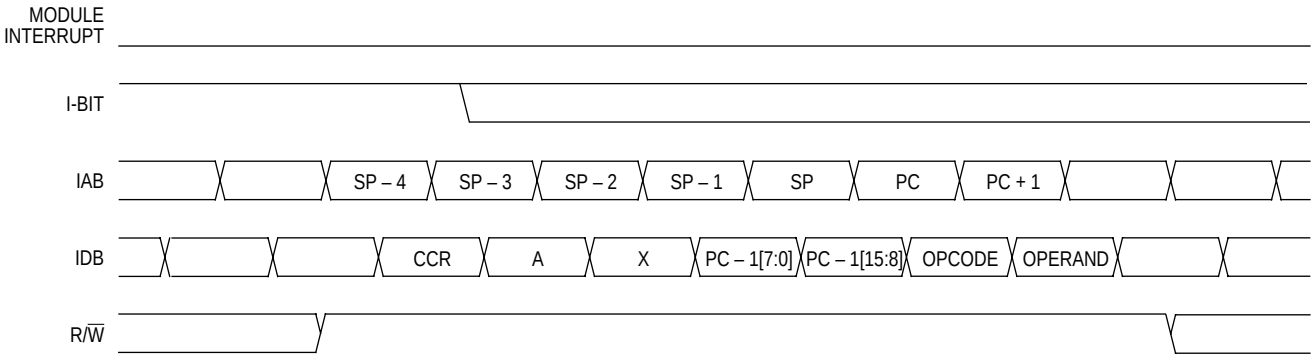


Figure 8-9. Interrupt Recovery Timing

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt may take precedence, regardless of priority, until the latched interrupt is serviced (or the I-bit is cleared).
(See **Figure 8-10**.)

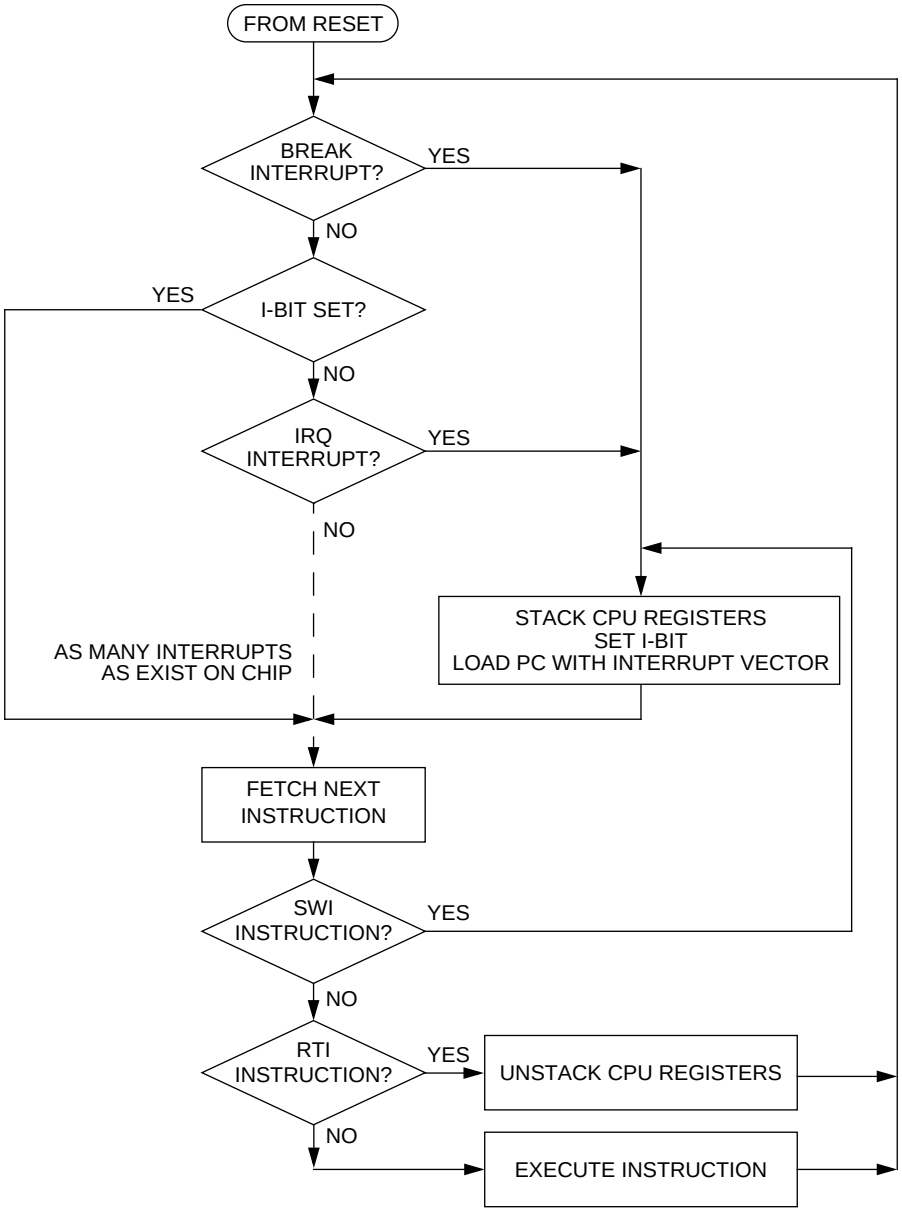


Figure 8-10. Interrupt Processing

8.6.1.1 Hardware Interrupts

Processing of a hardware interrupt begins after completion of the current instruction. When the 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 8-11** demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

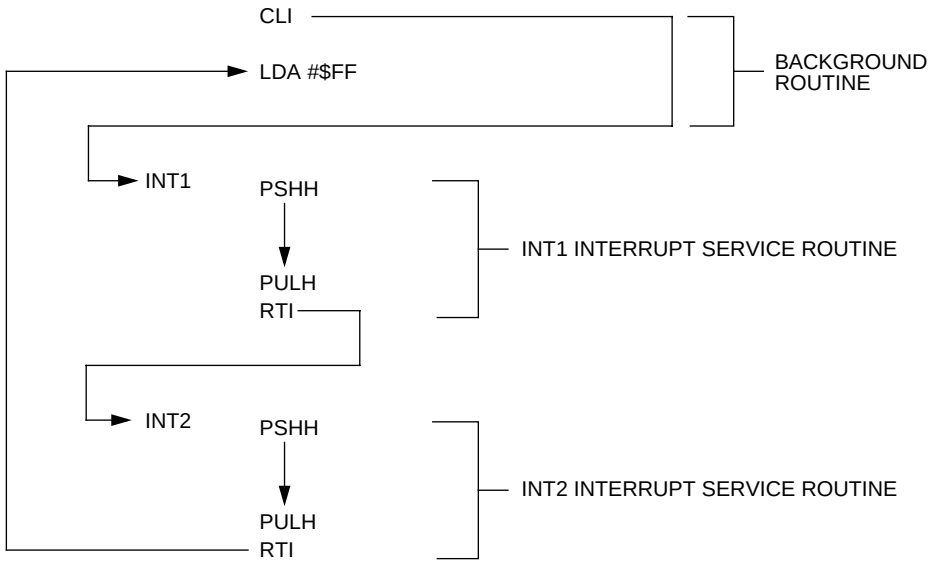


Figure 8-11. Interrupt Recognition Example

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

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

8.6.1.2 SWI Instruction

The SWI instruction is a non-maskable instruction that causes an interrupt regardless of the state of the interrupt mask (I-bit) in the condition code register.

NOTE: *A software interrupt pushes PC onto the stack. A software interrupt does **not** push PC – 1, as a hardware interrupt does.*

Table 8-3. Vector Addresses

Vector Priority	Address	Vector
Lowest	$FFD0	ADC Conversion Complete Vector (High)
	$FFD1	ADC Conversion Complete Vector (Low)
	$FFD2	Keyboard Vector (High)
	$FFD3	Keyboard Vector (Low)
	$FFD4	SCI Transmit Vector (High)
	$FFD5	SCI Transmit Vector (Low)
	$FFD6	SCI Receive Vector (High)
	$FFD7	SCI Receive Vector (Low)
	$FFD8	SCI Error Vector (High)
	$FFD9	SCI Error Vector (Low)
	$FFDA	Reserved
	$FFDB	Reserved
	$FFDC	Reserved
	$FFDD	Reserved
	$FFDE	Timer B Channel 3 Vector (High)
	$FFDF	Timer B Channel 3 Vector (Low)
	$FFE0	Timer B Channel 2 Vector (High)
	$FFE1	Timer B Channel 2 Vector (Low)
	$FFE2	SPI Transmit Vector (High)
	$FFE3	SPI Transmit Vector (Low)
	$FFE4	SPI Receive Vector (High)
	$FFE5	SPI Receive Vector (Low)
	$FFE6	Timer B Overflow Vector (High)
	$FFE7	Timer B Overflow Vector (Low)
	$FFE8	Timer B Channel 1 Vector (High)
	$FFE9	Timer B Channel 1 Vector (Low)
	$FFEA	Timer B Channel 0 Vector (High)
	$FFEB	Timer B Channel 0 Vector (Low)
	$FFEC	Timer A Overflow Vector (High)
	$FFED	Timer A Overflow Vector (Low)
	$FFEE	Timer A Channel 3 Vector (High)
	$FFEF	Timer A Channel 3 Vector (Low)
	$FFF0	Timer A Channel 2 Vector (High)
	$FFF1	Timer A Channel 2 Vector (Low)
	$FFF2	Timer A Channel 1 Vector (High)
	$FFF3	Timer A Channel 1 Vector (Low)
	$FFF4	Timer A Channel 0 Vector (High)
	$FFF5	Timer A Channel 0 Vector (Low)
	$FFF6	Programmable Interrupt Timer (High)
	$FFF7	Programmable Interrupt Timer (Low)
	$FFF8	PLL Vector (High)
	$FFF9	PLL Vector (Low)
	$FFFA	IRQ Vector (High)
	$FFFB	IRQ Vector (Low)
	$FFFC	SWI Vector (High)
	$FFFD	SWI Vector (Low)
	$FFFE	Reset Vector (High)
Highest	$FFFF	Reset Vector (Low)

8.6.2 Reset

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

8.6.3 Break Interrupts

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

8.6.4 Status Flag Protection in Break Mode

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

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

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

8.7 Low-Power Modes

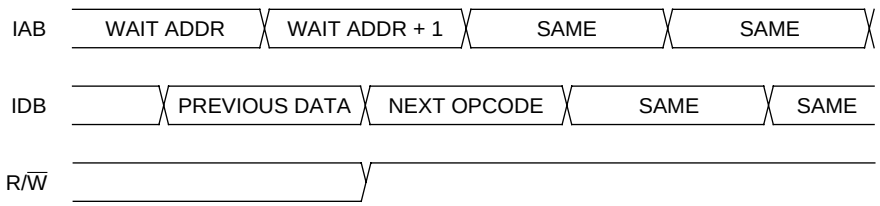
Executing the STOP or WAIT instruction puts the MCU in a low-power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts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

8.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. **Figure 8-12** 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.

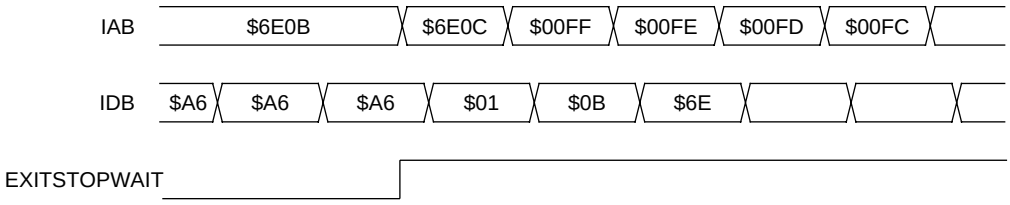
wait mode can also be exited by a reset or break. A break interrupt during wait mode sets the SIM break STOP/WAIT bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the configuration register 1 (CONFIG1) is 0, then the computer operating properly (COP) module is enabled and remains active in wait mode.



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

Figure 8-12. Wait Mode Entry Timing

Figure 8-13 and **Figure 8-14** show the timing for wait recovery.



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

Figure 8-13. Wait Recovery from Interrupt or Break

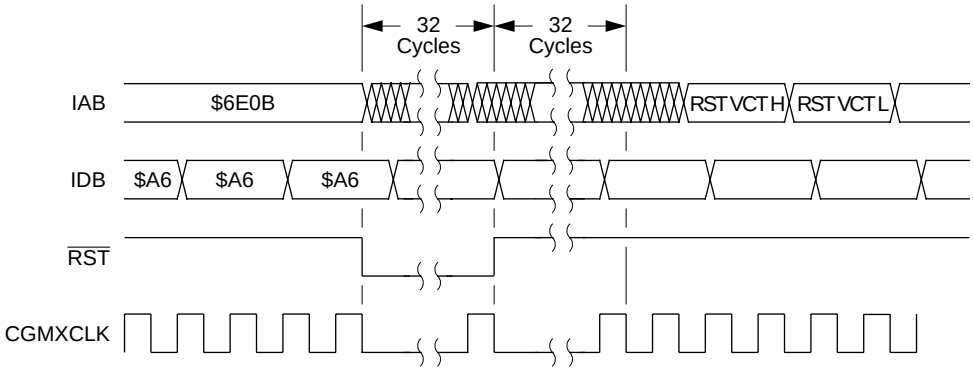


Figure 8-14. Wait Recovery from Internal Reset

8.7.2 Stop Mode

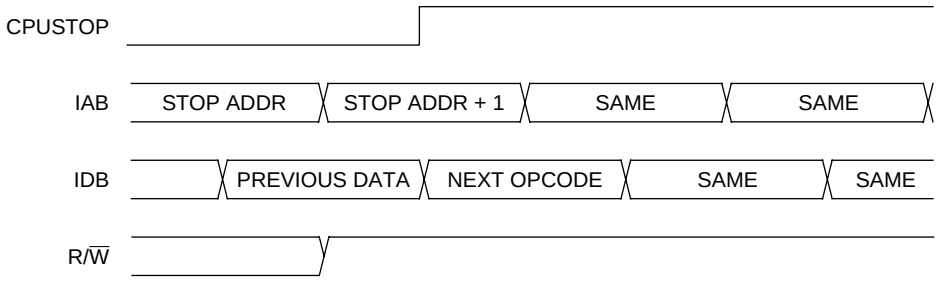
In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset or break also causes an exit from stop mode.

The SIM disables the clock generator module outputs (CGMOUT and CGMXCLK) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in the configuration register 1 (CONFIG1). 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 8-15** shows stop mode entry timing.



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

Figure 8-15. Stop Mode Entry Timing

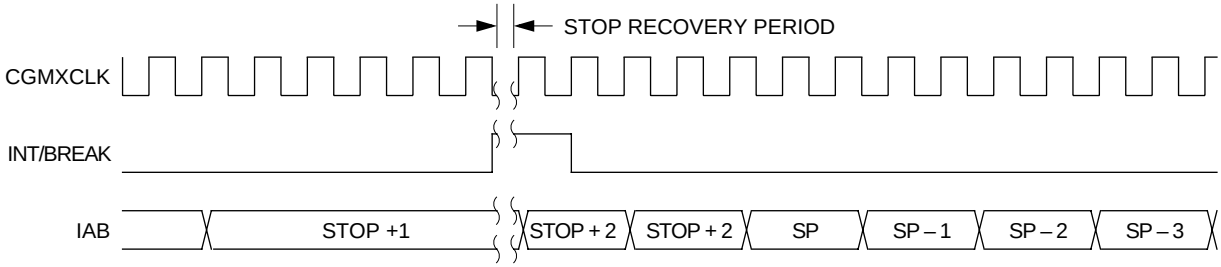


Figure 8-16. Stop Mode Recovery from Interrupt or Break

8.8 SIM Registers

The SIM has three memory mapped registers. **Table 8-4** shows the mapping of these registers.

Table 8-4. SIM Registers

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

8.8.1 SIM Break Status Register

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

Address: $FE00

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

R = Reserved

Note: 1. Writing a logic 0 clears SBSW.

Figure 8-17. SIM Break Status Register (SBSR)

SBSW — SIM Break STOP/WAIT

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

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

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

```
; 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 STOP or WAIT mode was exited by
                                   ; break.

          TST    LOBYTE,SP         ; If RETURNLO is not 0,

          BNE    DOLO              ; then just decrement low byte.

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

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

  RETURN  PULH                     ; Restore H register.
          RTI
```

8.8.2 SIM Reset Status Register

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

Address: $FE01

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

= Unimplemented

Figure 8-18. SIM Reset Status Register (SRSR)

POR — Power-On Reset Bit

1 = Last reset caused by POR circuit

0 = Read of SRSR

PIN — External Reset Bit

1 = Last reset caused by external reset pin ($\overline{RST}$)
0 = POR or read of SRSR

COP — Computer Operating Properly Reset Bit

1 = Last reset caused by COP counter
0 = POR or read of SRSR

ILOP — Illegal Opcode Reset Bit

1 = Last reset caused by an illegal opcode
0 = POR or read of SRSR

ILAD — Illegal Address Reset Bit (opcode fetches only)

1 = Last reset caused by an opcode fetch from an illegal address
0 = POR or read of SRSR

LVI — Low-Voltage Inhibit Reset Bit

1 = Last reset was caused by the LVI circuit
0 = POR or read of SRSR

8.8.3 SIM Break Flag Control Register

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

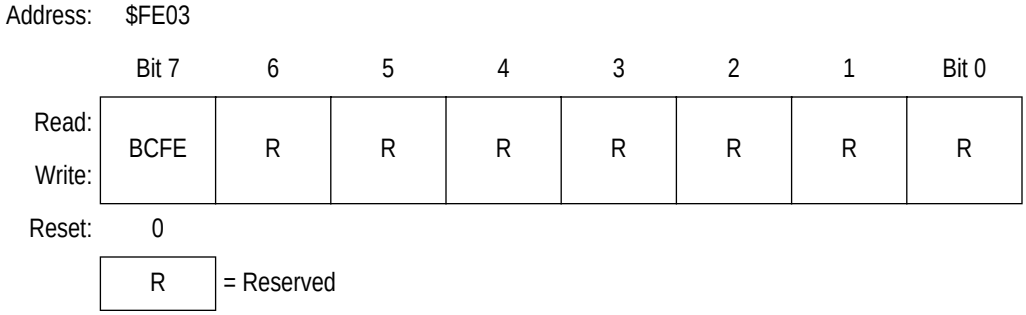


Figure 8-19. SIM Break Flag Control Register (SBFCR)

BCFE — Break Clear Flag Enable Bit

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

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

Section 9. Clock Generator Module (CGM)

9.1 Contents

9.2 Introduction

This section describes the clock generator module (CGM). 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 8 MHz crystals or ceramic resonators. The PLL can generate an 8 MHz bus frequency without using a higher frequency crystal.

9.3 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

9.4 Functional Description

The CGM consists of three major sub-modules:

- Crystal oscillator circuit which generates the constant crystal frequency clock, CGMXCLK.
- Phase-locked loop (PLL) which 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 9-1 shows the structure of the CGM.

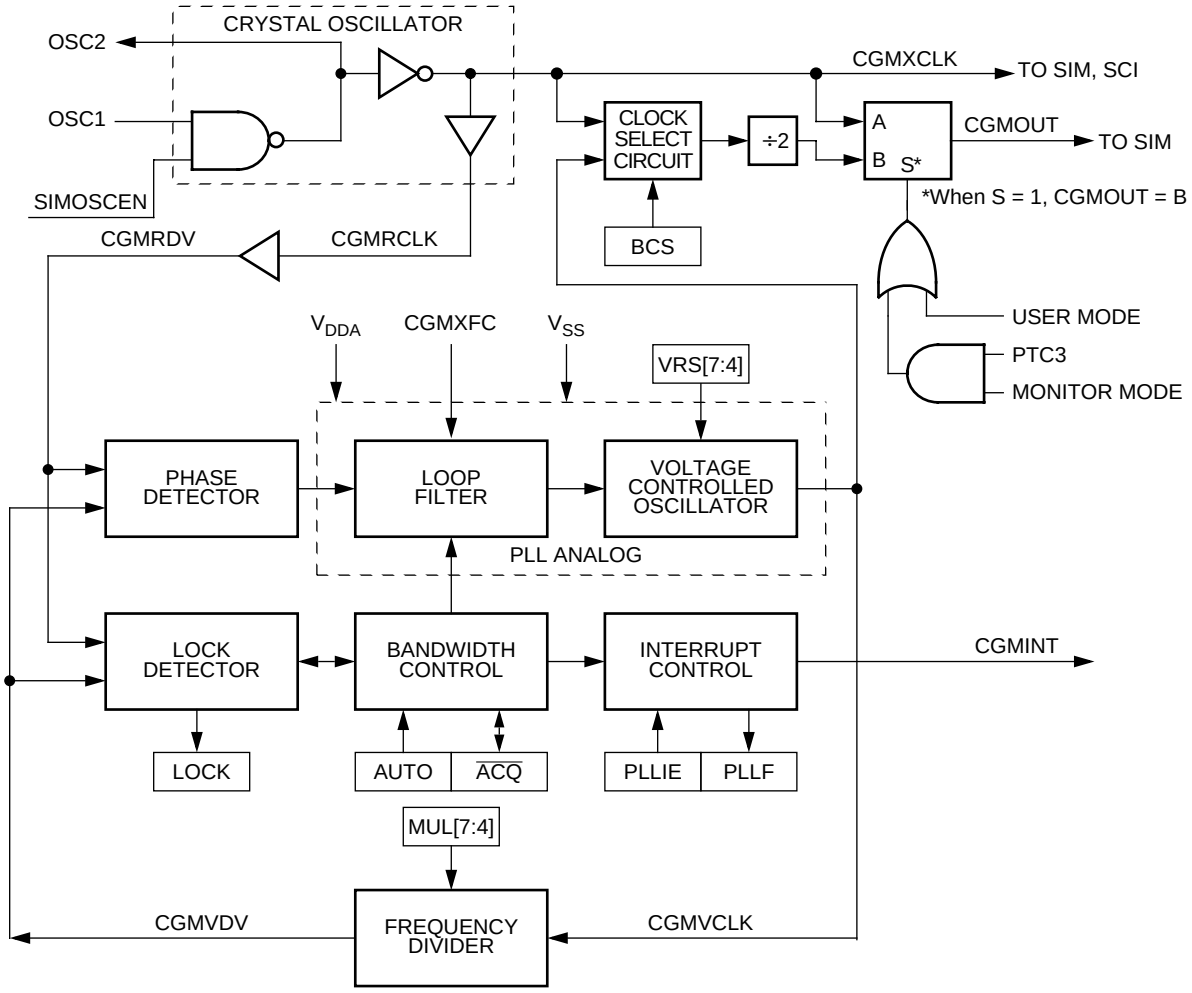


Figure 9-1. CGM Block Diagram

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

= Unimplemented

Figure 9-2. CGM I/O Register Summary

9.4.1 Crystal Oscillator Circuit

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

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

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

An externally generated clock can also feed the OSC1 pin of the crystal oscillator circuit. For this configuration, the external clock should be connected to the OSC1 pin and the OSC2 pin allowed to float.

9.4.2 Phase-Locked Loop (PLL) Circuit

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.

9.4.2.1 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}, (4.9152MHz) 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 **9.4.2.4 Programming the PLL** for more information).

The phase detector then compares the VCO feedback clock, CGMVDV, with the final reference clock, CGMRDV. A correction pulse is generated based on the phase difference between the two signals. The loop filter

then slightly alters the DC voltage on the external capacitor connected to CGMXFC based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, described in **9.4.2.2 Acquisition and Tracking Modes**. The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

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

9.4.2.2 Acquisition and Tracking Modes

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

- Acquisition mode — in acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL start-up or when the PLL has suffered a severe noise hit and the resulting VCO frequency is much different from the desired frequency. When in acquisition mode, the $\overline{ACQ}$ bit is clear in the PLL bandwidth control register. See **9.6.2 PLL Bandwidth Control Register (PBWC)**.
- Tracking mode — in tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source. See **9.4.3 Base Clock Selector Circuit**. The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{ACQ}$ bit is set.

9.4.2.3 Manual and Automatic PLL Bandwidth Modes

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

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode is used also to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT. See **9.6.2 PLL Bandwidth Control Register (PBWC)**. If PLL interrupts are enabled, the software can wait for a PLL interrupt request and then check the LOCK bit. If interrupts are disabled, software can poll the LOCK bit continuously (during PLL start-up, usually) or at periodic intervals. In either case, when the LOCK bit is set, the VCO clock is safe to use as the source for the base clock. See **9.4.3 Base Clock Selector Circuit**. If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application. (See **9.7 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 **9.6.2 PLL Bandwidth Control Register (PBWC)**) is a read-only indicator of the mode of the filter. (See **9.4.2.2 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 **9.10 Acquisition/Lock Time Specifications**)
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance Δ_{LOCK} and is cleared when the VCO frequency is out of a certain tolerance Δ_{UNL}. (See **9.10 Acquisition/Lock Time Specifications**)
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. (See **9.6.1 PLL Control Register (PCTL)**)

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

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 **9.10 Acquisition/Lock Time Specifications**), after turning on the PLL by setting PLLON in the PLL control register (PCTL).
- Software must wait a given time, t_{AL}, after entering tracking mode before selecting the PLL as the clock source to CGMOUT (BCS = 1).
- The LOCK bit is disabled.
- CPU interrupts from the CGM are disabled.

9.4.2.4 Programming the PLL

The following procedure shows how to program the PLL.

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

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

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

3. Choose a practical PLL reference frequency, f_{RCLK}.
4. Select 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} = 4.9152MHz

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 cause the MCU to "crash".*

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.

9.4.2.5 Special Programming Exceptions

The programming method described in **9.4.2.4 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 **9.4.3 Base Clock Selector Circuit**)

9.4.3 Base Clock Selector Circuit

This circuit is used to select either the crystal clock, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the base clock, CGMOUT. The two input clocks go through a transition control circuit that waits up to three CGMXCLK cycles and three CGMVCLK cycles to change from one clock source to the other. During this time, CGMOUT is held in stasis. The output of the transition control circuit is then divided by two to correct the duty cycle. Therefore, the bus clock frequency, which is one-half of the base clock frequency, is one-fourth the frequency of the selected clock (CGMXCLK or CGMVCLK).

The BCS bit in the PLL control register (PCTL) selects which clock drives CGMOUT. The VCO clock cannot be selected as the base clock source if the PLL is not turned on. The PLL cannot be turned off if the VCO clock is selected. The PLL cannot be turned on or off simultaneously with the selection or deselection of the VCO clock. The VCO clock also cannot be selected as the base clock source if the factor L is programmed to a 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.

9.4.4 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 9-3**. This figure shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

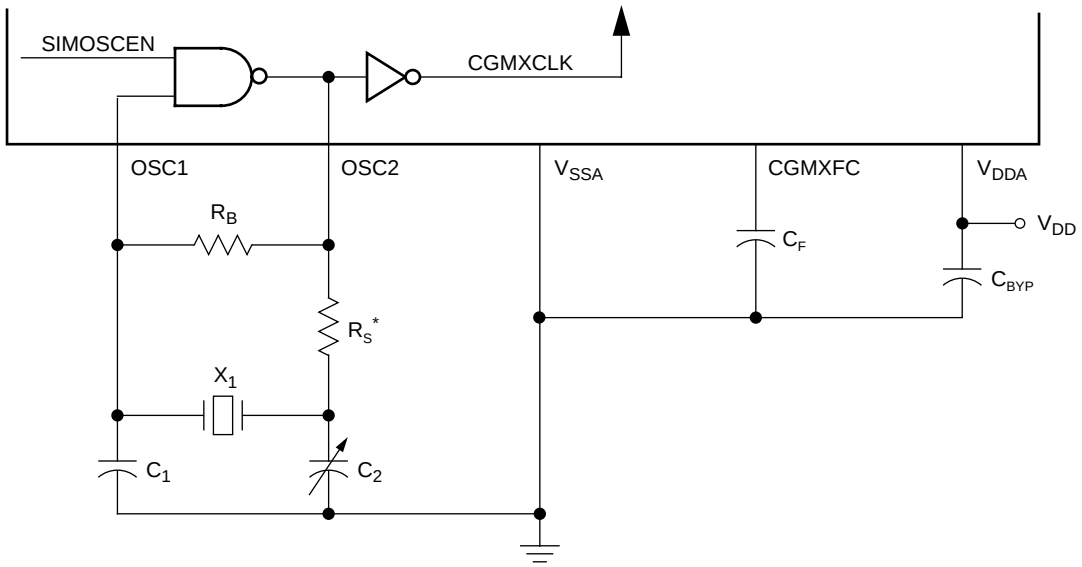
- Crystal, X_1
- Fixed capacitor, C_1
- Tuning capacitor, C_2 (can also be a fixed capacitor)
- Feedback resistor, R_B
- Series resistor, R_S (optional)

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 9-3 also shows the external components for the PLL:

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

Care should be taken with routing in order to minimize signal cross talk and noise. (See **9.10 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 9-3. CGM External Connections

9.5 I/O Signals

The following paragraphs describe the CGM I/O signals.

9.5.1 Crystal Amplifier Input Pin (OSC1)

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

9.5.2 Crystal Amplifier Output Pin (OSC2)

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

9.5.3 External Filter Capacitor Pin (CGMXFC)

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

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

9.5.4 PLL Analog Power Pin (V_{DDA})

V_{DDA} is a power pin used by the analog portions of the PLL. The pin should be connected 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.*

9.5.5 Oscillator Enable Signal (SIMOSCEN)

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

9.5.6 Crystal Output Frequency Signal (CGMXCLK)

CGMXCLK is the crystal oscillator output signal. It runs at the full speed of the crystal (f_{XCLK}) and is generated directly from the crystal oscillator circuit. **Figure 9-3** 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.

9.5.7 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.

9.5.8 CGM CPU Interrupt (CGMINT)

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

9.6 CGM Registers

The following registers control and monitor operation of the CGM:

- PLL control register (PCTL). (See **9.6.1 PLL Control Register (PCTL)**)
- PLL bandwidth control register (PBWC). (See **9.6.2 PLL Bandwidth Control Register (PBWC)**)
- PLL programming register (PPG). (See **9.6.3 PLL Programming Register (PPG)**)

Figure 9-4 is a summary of the CGM registers.

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

= 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, $\overline{\text{ACQ}}$ 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 9-4. CGM I/O Register Summary

9.6.1 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.

Address: $001C

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

= Unimplemented

Figure 9-5. 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 0. Reset clears the PLLIE bit.

1 = PLL interrupts enabled
0 = PLL interrupts disabled

PLLF — PLL Interrupt Flag Bit

This read-only bit is set whenever the LOCK bit toggles. PLLF generates an interrupt request if the PLLIE bit is set also. PLLF always reads as 0 when the AUTO bit in the PLL bandwidth control register (PBWC) is clear. The PLLF bit should be cleared by reading the PLL control register. Reset clears the PLLF bit.

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

NOTE: *The PLLF bit should not be inadvertently cleared. 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 **9.4.3 Base Clock Selector Circuit**. Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

BCS — Base Clock Select Bit

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to

the other. During the transition, CGMOUT is held in stasis. See **9.4.3 Base Clock Selector Circuit**. Reset and the STOP instruction clear the BCS bit.

1 = CGMOUT driven by CGMVCLK/2
0 = CGMOUT driven by CGMXCLK/2

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 **9.4.3 Base Clock Selector Circuit**.*

Bits [3:0] — Unimplemented bits

These bits provide no function and always read as 1.

9.6.2 PLL Bandwidth Control Register (PBWC)

The PLL bandwidth control register does the following:

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

Address: $001D

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

= Unimplemented

Figure 9-7. 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), the $\overline{\text{ACQ}}$ bit should be cleared before turning the PLL on. 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 0 and has no meaning. Reset clears the LOCK bit.

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

$\overline{\text{ACQ}}$ — Acquisition Mode Bit

When the AUTO bit is set, $\overline{\text{ACQ}}$ is a read-only bit that indicates whether the PLL is in acquisition mode or tracking mode. When the AUTO bit is clear, $\overline{\text{ACQ}}$ 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 indicates whether the crystal reference frequency is active or not. To check the status of the crystal reference, the following procedure should be followed:

1. Write a 1 to XLD.
2. Wait $4 \times N$ 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 0.

Bits [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 Bits [3:0] whenever writing to PBWC.

9.6.3 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.

Address: $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 9-8. 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 **9.4.2.1 PLL Circuits** and **9.4.2.4 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 9-1. 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

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 **9.4.2.1 PLL Circuits**, **9.4.2.4 Programming the PLL**, and **9.6.1 PLL Control Register (PCTL)**). VRS[7:4] cannot be written when the PLLON bit in the PLL control register (PCTL) is set. (See **9.4.2.5 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 **9.4.3 Base Clock Selector Circuit** and **9.4.2.5 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.

9.7 Interrupts

When the AUTO bit is set in the PLL bandwidth control register (PBWC), the PLL can generate a CPU interrupt request every time the LOCK bit changes state. The PLLIE bit in the PLL control register (PCTL) enables CPU interrupts from the PLL. PLLF, the interrupt flag in the PCTL, becomes set whether interrupts are enabled or not. When the AUTO bit is clear, CPU interrupts from the PLL are disabled and PLLF reads as 0.

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

NOTE: *Software can select CGMVCLK/2 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.*

9.8 Low-Power Modes

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

9.8.1 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.

9.8.2 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.

9.9 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 **Section 8. System Integration Module (SIM)**.

To allow software to clear status bits during a break interrupt, a 1 should be written 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 0 to the BCFE bit. With BCFE at 0 (its default state), software can read and write the PLL control register during the break state without affecting the PLLF bit.

9.10 Acquisition/Lock Time Specifications

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

9.10.1 Acquisition/Lock Time Definitions

Typical control systems refer to the acquisition time or lock time as the reaction time of the system, within specified tolerances, 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 percentage of the step input or when the output settles to the desired value plus or minus a percentage 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 0Hz to 1MHz, the acquisition time is the time taken for the frequency to reach 1MHz ± 50kHz. 50kHz = 5% of the 1MHz step input. If the system is operating at 1MHz and suffers a –100kHz noise hit, the acquisition time is the time taken to return from 900kHz to 1MHz ± 5kHz. 5kHz = 5% of the 100kHz 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 **9.4.2.3 Manual and Automatic PLL Bandwidth Modes**), acquisition time expires when the $\overline{ACQ}$ bit becomes set in the PLL bandwidth control register (PBWC).

- Lock time, t_{LOCK}, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the lock mode entry tolerance Δ_{LOCK}. Lock time is based on an initial frequency error, $(f_{DES} - f_{ORIG})/f_{DES}$, of not more than ±100%. In automatic bandwidth control mode, lock time expires when the LOCK bit becomes set in the PLL bandwidth control register (PBWC). See **9.4.2.3 Manual and Automatic PLL Bandwidth Modes**.

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

9.10.2 Parametric Influences On Reaction Time

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

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

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 **9.10.3 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.

9.10.3 Choosing a Filter Capacitor

As described in **9.10.2 Parametric Influences On Reaction Time**, the external filter capacitor C_F is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage. The value of the capacitor must, therefore, be chosen with supply potential and reference frequency in mind. For proper operation, the external filter capacitor must be chosen according to the following equation:

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

For the value of V_{DDA}, the voltage potential at which the MCU is operating should be used. 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.

9.10.4 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 **9.10.3 Choosing a Filter Capacitor**)
- Room temperature operation
- Negligible external leakage on CGMXFC
- Negligible noise

The K factor in the equations is derived from internal PLL parameters. K_{ACQ} is the K factor when the PLL is configured in acquisition mode, and K_{TRK} is the K factor when the PLL is configured in tracking mode. See **9.4.2.2 Acquisition and Tracking Modes**.

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

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

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

Note 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 **9.4.2.3 Manual and Automatic PLL Bandwidth Modes**. A certain number of clock cycles, n_{ACQ}, is required to ascertain whether the PLL is within the tracking mode entry tolerance Δ_{TRK}, before exiting acquisition mode. Also, a certain number of clock cycles, n_{TRK}, is required to ascertain

whether 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 **9.4.3 Base Clock Selector Circuit**), because the factors described in **9.10.2 Parametric Influences On Reaction Time** may slow the lock time considerably.

Section 10. Monitor ROM (MON)

10.1 Contents

10.2 Introduction

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

10.3 Features

Features of the monitor ROM include the following:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between monitor ROM and host computer
- Standard mark/space non-return-to-zero (NRZ) communication with host computer
- 4800 baud to 28.8 k-baud communication with host computer
- FLASH memory security feature[1]
- Execution of code in RAM or FLASH

10.4 Functional Description

The monitor ROM receives and executes commands from a host computer. **Figure 10-1** shows a example circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

While simple monitor commands can access any memory address, the MCU has a FLASH security feature to prevent external viewing of the contents of FLASH. Proper procedures must be followed to verify FLASH content. Access to the FLASH is denied to unauthorized users of customer specified software.

In monitor mode, the MCU can execute host-computer code in RAM while all MCU pins except PTA0 retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.

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

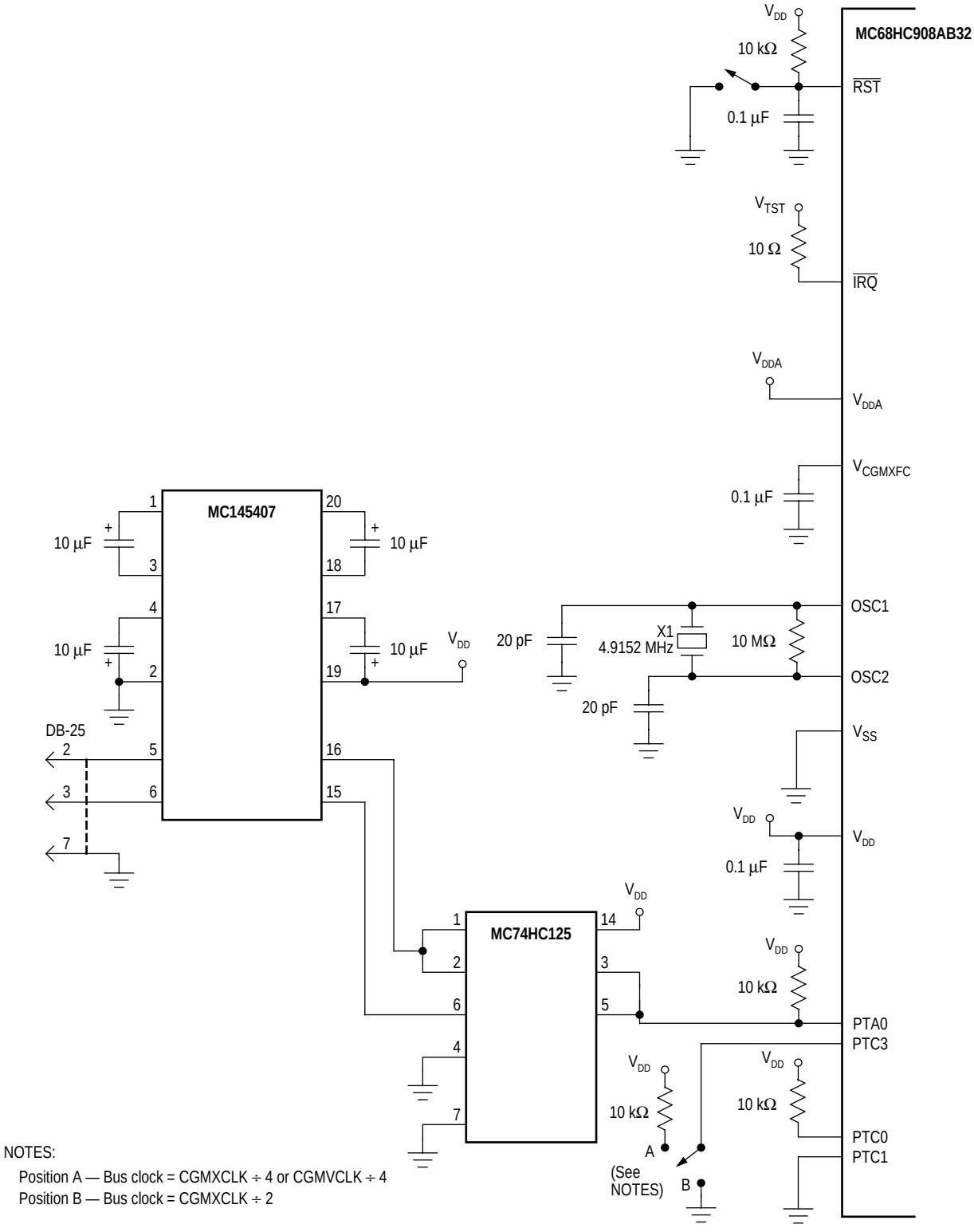


Figure 10-1. Monitor Mode Circuit

10.4.1 Entering Monitor Mode

Table 10-1 shows the pin conditions for entering monitor mode.

Table 10-1. Monitor Mode Entry Conditions

$\overline{IRQ}$ Pin	PTC0 Pin	PTC1 Pin	PTA0 Pin	PTC3 Pin	CGMOUT	Bus Frequency (CGMOUT ÷ 2)
V_{TST}(1)	1	0	1	1	CGMXCLK ÷ 2 or CGMVCLK ÷ 2	CGMXCLK ÷ 4 or CGMVCLK ÷ 4
V_{TST}	1	0	1	0	CGMXCLK	CGMXCLK ÷ 2

Notes:
1. For V_{TST}, see **Section 23. Electrical Specifications**.

Enter monitor mode by either

- Executing a software interrupt instruction (SWI) or
- Applying a logic 0 and then a logic 1 to the $\overline{RST}$ pin.

The MCU sends a break signal (10 consecutive logic 0s) to the host computer, indicating that it is ready to receive a command. The break signal also provides a timing reference to allow the host to determine the necessary baud rate.

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_{TST} (see **Section 23. Electrical Specifications**) is applied to either the $\overline{IRQ}$ pin or the $\overline{RST}$ pin. (See **Section 8. System Integration Module (SIM)** for more information on modes of operation.)

NOTE: *Holding the PTC3 pin low when entering monitor mode causes a bypass of a divide-by-two stage at the oscillator. The CGMOUT frequency is equal to the CGMXCLK frequency, and the OSC1 input directly generates internal bus clocks. In this case, the OSC1 signal must have a 50% duty cycle at maximum bus frequency.*

Table 10-2 is a summary of the differences between user mode and monitor mode.

Table 10-2. Mode Differences

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

Notes:
1. If the high voltage (V_{TST}) is removed from the $\overline{IRQ}$ pin while in monitor mode, the SIM asserts its COP enable output. The COP can be enabled or disabled by the COPD bit in the configuration register 1 (CONFIG1). (See **23.6 5.0-V DC Electrical Characteristics**.)

10.4.2 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. (See **Figure 10-2** and **Figure 10-3**.)

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

Figure 10-2. Monitor Data Format

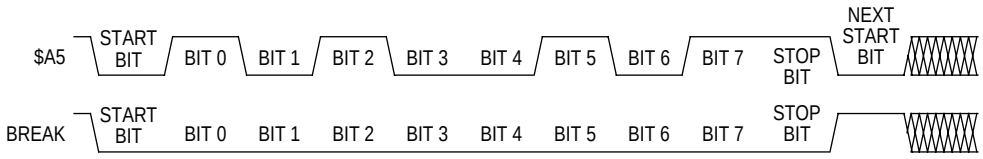


Figure 10-3. Sample Monitor Waveforms

10.4.3 Echoing

As shown in **Figure 10-4**, the monitor ROM immediately echoes each received byte back to the PTA0 pin for error checking.

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

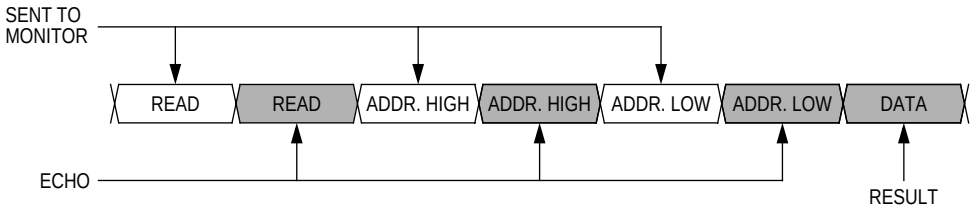


Figure 10-4. Read Transaction

10.4.4 Break Signal

A start bit followed by nine low bits is a break signal. (See **Figure 10-5**.) When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits before echoing the break signal.

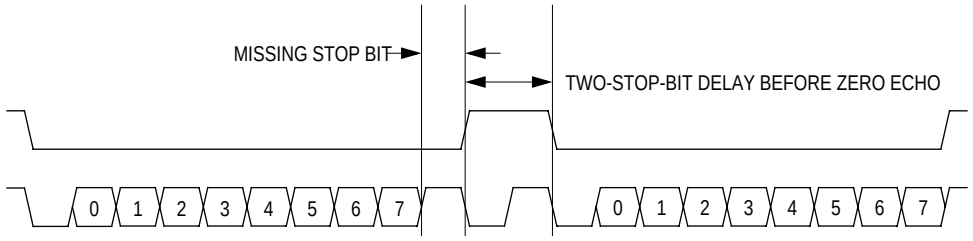


Figure 10-5. Break Transaction

10.4.5 Commands

The monitor ROM uses these commands:

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

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

Table 10-3. 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

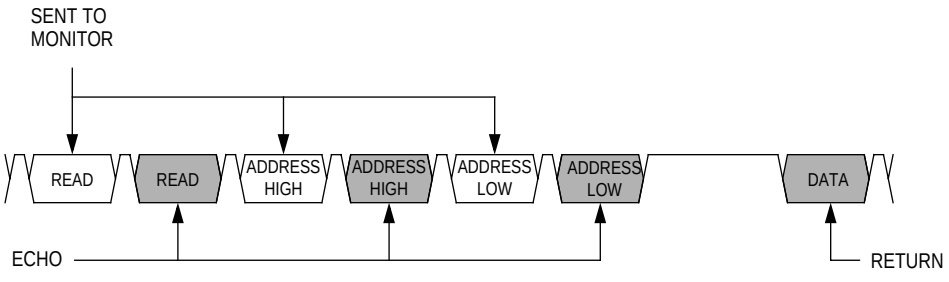


Table 10-4. WRITE (Write Memory) Command

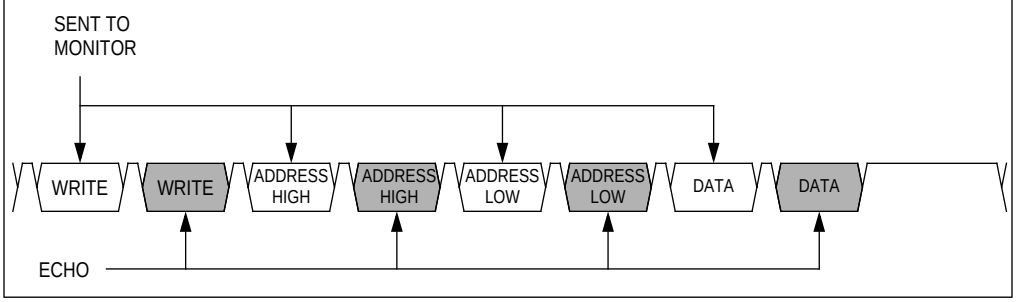
Description	Write byte to memory
Operand	Specifics 2-byte address in high byte:low byte order; low byte followed by data byte
Data Returned	None
Opcode	$49
Command Sequence	

Table 10-5. IREAD (Indexed Read) Command

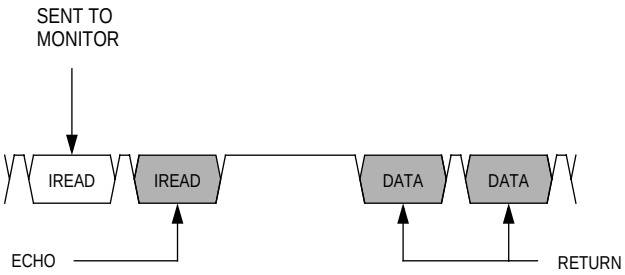
Description	Read next 2 bytes in memory from last address accessed
Operand	Specifies 2-byte address in high byte:low byte order
Data Returned	Returns contents of next two addresses
Opcode	$1A
Command Sequence	

Table 10-6. IWRITE (Indexed Write) Command

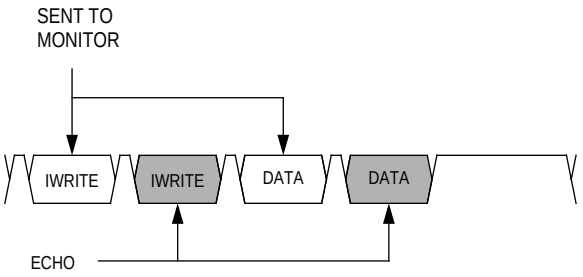
Description	Write to last address accessed + 1
Operand	Specifies single data byte
Data Returned	None
Opcode	$19
Command Sequence	

Table 10-7. READSP (Read Stack Pointer) Command

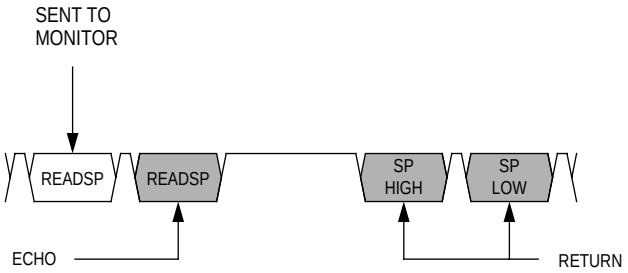
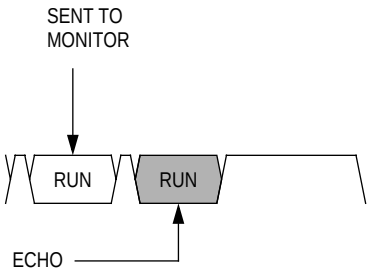
Description	Reads stack pointer
Operand	None
Data Returned	Returns stack pointer in high byte:low byte order
Opcode	$0C
Command Sequence	

Table 10-8. RUN (Run User Program) Command

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

10.4.6 Baud Rate

With a 4.9152-MHz crystal and the PTC3 pin at logic 1 during reset, data is transferred between the monitor and host at 4800 baud. If the PTC3 pin is at logic 0 during reset, the monitor baud rate is 9600. When the CGM output, CGMOUT, is driven by the PLL, the baud rate is determined by the MUL[7:4] bits in the PLL programming register (PPG). (See **Section 9. Clock Generator Module (CGM)**.)

Table 10-9. Monitor Baud Rate Selection

Monitor Baud Rate	VCO Frequency Multiplier (N)					
	1	2	3	4	5	6
4.9152 MHz	4800	9600	14,400	19,200	24,000	28,800
4.194 MHz	4096	8192	12,288	16,384	20,480	24,576

10.5 Security

A security feature discourages unauthorized reading of FLASH locations while in monitor mode. The host can bypass the security feature at monitor mode entry by sending eight security bytes that match the bytes at locations $FFF6–$FFFD. Locations $FFF6–$FFFD contain user-defined data.

NOTE: *Do not leave locations $FFF6–$FFFD blank. For security reasons, program locations $FFF6–$FFFD even if they are not used for vectors.*

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PTA0. If the received bytes match those at locations $FFF6–$FFFD, the host bypasses the security feature and can read all FLASH locations and execute code from FLASH. Security remains bypassed until a power-on reset occurs. If the reset was not a power-on reset, security remains bypassed and security code entry is not required. (See **Figure 10-6**.)

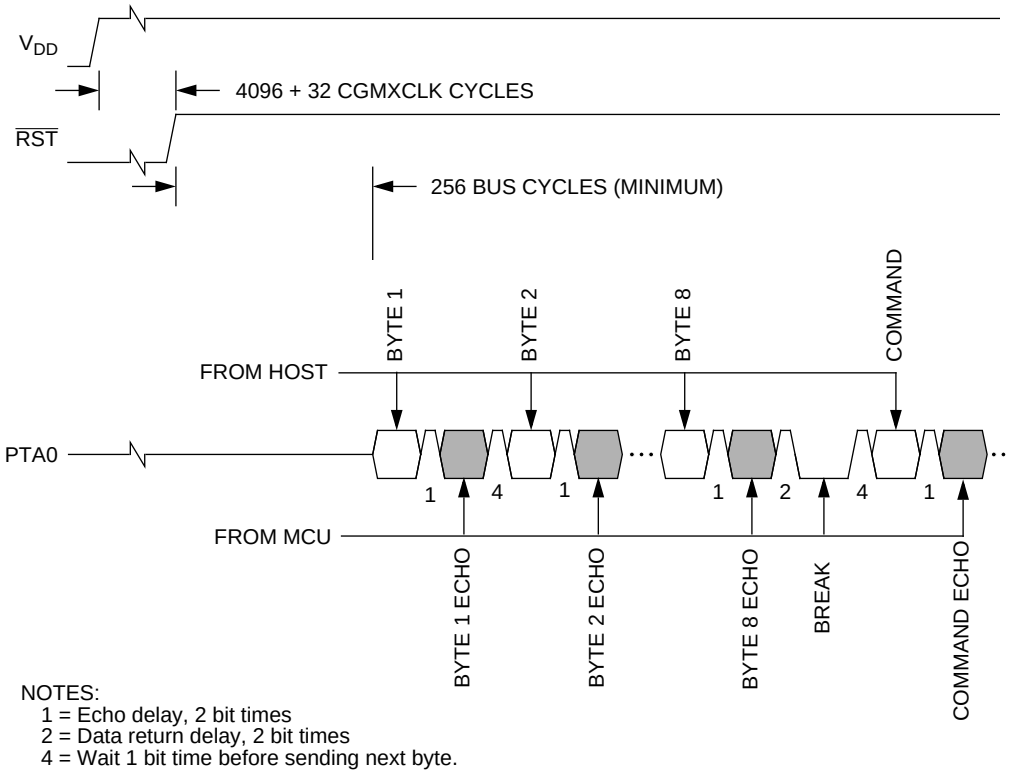


Figure 10-6. Monitor Mode Entry Timing

Upon power-on reset, if the received bytes of the security code do not match the data at locations $FFF6–$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. After receiving the eight security bytes from the host, the MCU transmits a break character, signifying that it is ready to receive a command.

To determine whether the security code entered is correct, check to see if bit 6 of RAM address $50 is set. If it is, then the correct security code has been entered and FLASH can be accessed.

NOTE: *The MCU does not transmit a break character until after the host sends the eight security bits.*

If the host fails the security bypass as described in **10.5 Security**, the MCU remains in monitor mode, but reading a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. The MCU monitor commands are still valid and user software can execute from RAM. A bulk-erase operation is possible, erasing the entire FLASH memory, including the security bytes at $FFF6–$FFFD.

10.6 Extended Security

To further disable monitor mode functions, an extended security command keyword can be programmed at FLASH locations $FFC0–$FFC7. The keyword is eight bytes long with a 7-byte ASCII string and 1-byte $00 delimiter. The keyword for the MC68HC908AB32 MCU is "PSWDOPT" + $00.

Entry to monitor mode with extended security command keyword programmed, the MCU stops communicating with the host after transmitting a break character if the host fails the security bypass as described in **10.5 Security**.

NOTE: *Once the extended security command keyword is programmed, the FLASH memory cannot be erased without a valid security code (matching $FFF6–$FFFD). Therefore, the extended security command keyword should not be programmed during software development.*

Section 11. Timer Interface Module A (TIMA)

11.1 Contents

11.2 Introduction

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

11.3 Features

Features of the TIMA 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 TIMA clock input:
  - Seven-frequency internal bus clock prescaler selection
  - External TIMA clock input (4MHz maximum frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMA counter stop and reset bits
- Modular architecture expandable to eight channels

11.4 Pin Name Conventions

The TIMA share five I/O pins with port D, E, and F I/O pins. The full name of the TIMA I/O pin is listed in **Table 11-1**. The generic pin name appear in the text that follows.

Table 11-1. Pin Name Conventions

TIMA Generic Pin Names:	Full TIMA Pin Names:
TACLK	PTD6/TACLK
TACH0	PTE2/TACH0
TACH1	PTE3/TACH1
TACH2	PTF0/TACH2
TACH3	PTF1/TACH3

11.5 Functional Description

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

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

11.5.1 TIMA Counter Prescaler

The TIMA clock source can be one of the seven prescaler outputs or the TIMA clock pin, PTD6/TACLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMA status and control register select the TIMA clock source.

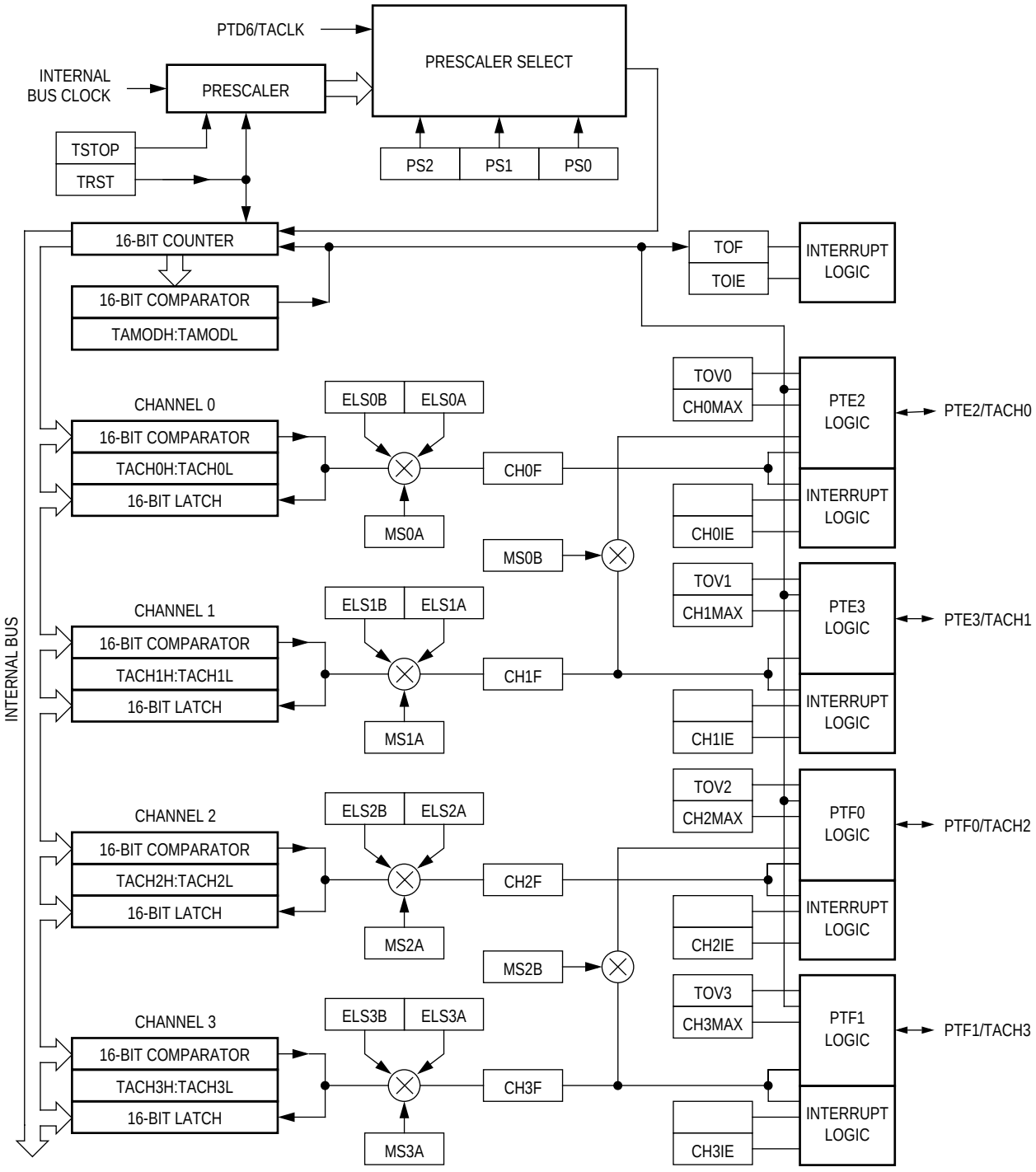


Figure 11-1. TIMA Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0020	Timer A Status and Control Register (TASC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0022	Timer A Counter Register High (TACNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0023	Timer A Counter Register Low (TACNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0024	Timer A Counter Modulo Register High (TAMODH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	Timer A Counter Modulo Register Low (TAMODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0026	Timer A Channel 0 Status and Control Register (TASC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0027	Timer A Channel 0 Register High (TACH0H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0028	Timer A Channel 0 Register Low (TACH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0029	Timer A Channel 1 Status and Control Register (TASC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$002A	Timer A Channel 1 Register High (TACH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							

= Unimplemented

Figure 11-2. TIMA I/O Register Summary (Sheet 1 of 2)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002B	Timer A Channel 1 Register Low (TACH1L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$002C	Timer A Channel 2 Status and Control Register (TASC2)	Read: Write:	CH2F 0	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		Reset:	0	0	0	0	0	0	0	0
$002D	Timer A Channel 2 Register High (TACH2H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$002E	Timer A Channel 2 Register Low (TACH2L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$002F	Timer A Channel 3 Status and Control Register (TASC3)	Read: Write:	CH3F 0	CH3IE	0 —	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Reset:	0	0	0	0	0	0	0	0
$0030	Timer A Channel 3 Register High (TACH3H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0031	Timer A Channel 3 Register Low (TACH3L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							

— = Unimplemented

Figure 11-2. TIMA I/O Register Summary (Sheet 2 of 2)

11.5.2 Input Capture

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

11.5.3 Output Compare

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

11.5.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in **11.5.3 Output Compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMA channel registers.

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

Use 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 TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

11.5.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTE2/TACH0 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The output compare value in the TIMA channel 0 registers initially controls the output on the PTE2/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the output are the ones written to last. TASC0 controls and monitors the buffered output compare function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TACH1, 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 PTF0/TACH2 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIMA channel 2 status and control register (TASC2) links channel 2 and channel 3. The output compare value in the TIMA channel 2 registers initially controls the output on the PTF0/TACH2 pin. Writing to the TIMA channel 3 registers enables the TIMA channel 3 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (2 or 3) that control the output are the ones written to last. TASC2 controls and monitors the buffered output compare function, and TIMA channel 3 status and control register (TASC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF1/TACH3, is available as a general-purpose I/O pin.

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

11.5.4 Pulse Width Modulation (PWM)

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

As **Figure 11-3** shows, the output compare value in the TIMA channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMA to clear the channel pin on output compare if the state of the PWM pulse is logic one. Program the TIMA to set the pin if the state of the PWM pulse is logic zero.

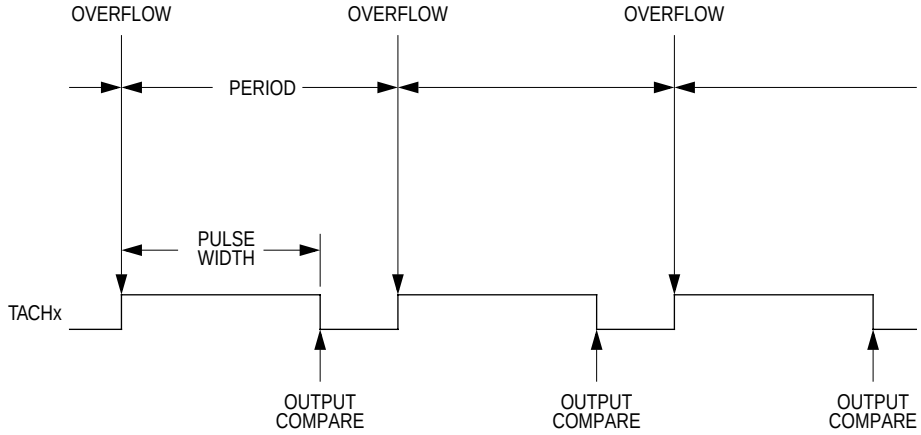


Figure 11-3PWM Period and Pulse Width

The value in the TIMA counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIMA counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000. See **11.10.1 TIMA Status and Control Register**.

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

11.5.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **11.5.4 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 TIMA channel registers.

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

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 TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

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

11.5.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTE2/TACH0 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The TIMA channel 0 registers initially control the pulse width on the PTE2/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the pulse width are the ones written to last. TASC0 controls and monitors the buffered PWM function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TACH1, 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 PTF0/TACH2 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS2B bit in TIMA channel 2 status and control register (TASC2) links channel 2 and channel 3. The TIMA channel 2 registers initially control the pulse width on the PTF0/TACH2 pin. Writing to the TIMA channel 3 registers enables the TIMA channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (2 or 3) that control the pulse width are the ones written to last. TASC2 controls and monitors the buffered PWM function, and TIMA channel 3 status and control register (TASC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF1/TACH3, is available as a general-purpose I/O pin.

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

11.5.4.3 PWM Initialization

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

1. In the TIMA status and control register (TASC):
   a. Stop the TIMA counter by setting the TIMA stop bit, TSTOP.
   b. Reset the TIMA counter by setting the TIMA reset bit, TRST.
2. In the TIMA counter modulo registers (TAMODH:TAMODL), write the value for the required PWM period.
3. In the TIMA channel x registers (TACHxH:TACHxL), write the value for the required pulse width.
4. In TIMA channel x status and control register (TASCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. See **Table 11-3**.
   a. Write 1 to the toggle-on-overflow bit, TOVx.
   b. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB:ELSxA. The output action on compare must force the output to the complement of the pulse width level. See **Table 11-3**.

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

5. In the TIMA status control register (TASC), clear the TIMA stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMA channel 0 registers (TACH0H:TACH0L) initially control the buffered PWM output. TIMA channel 0 status and control register 0 (TASC0) 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 TIMA channel 2 registers (TACH2H:TACH2L) initially control the PWM output. TIMA channel 2 status and control register (TASC2) 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 TIMA overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100% duty cycle output. See **11.10.4 TIMA Channel Status and Control Registers**.

11.6 Interrupts

The following TIMA sources can generate interrupt requests:

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

11.7 Low-Power Modes

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

11.7.1 Wait Mode

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

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

11.7.2 Stop Mode

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

11.8 TIMA During Break Interrupts

A break interrupt stops the TIMA 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 **8.8.3 SIM Break Flag Control Register**.

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

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

11.9 I/O Signals

Ports E and F each share two pins with the TIMA and port D shares one. PTD6/TACLK is an external clock input to the TIMA prescaler. The four TIMA channel I/O pins are PTE2/TACH0, PTE3/TACH1, PTF0/TACH2, and PTF1/TACH3.

11.9.1 TIMA Clock Pin

PTD6/TACLK is an external clock input that can be the clock source for the TIMA counter instead of the prescaled internal bus clock. Select the PTD6/TACLK input by writing logic 1s to the three prescaler select bits, PS[2:0]. See **11.10.1 TIMA Status and Control Register**. The minimum TACLK pulse width, $TACLK_{LMIN}$ or $TACLK_{HMIN}$, is:

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

The maximum TACLK frequency is:

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

PTD6/TACLK is available as a general-purpose I/O pin when not used as the TIMA clock input. When the PTD6/TACLK pin is the TIMA clock input, it is an input regardless of the state of the DDRD6 bit in data direction register D.

11.9.2 TIMA Channel I/O Pins

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTF0/TACH2 and PTE3/TACH1 can be configured as buffered output compare or buffered PWM pins.

11.10 I/O Registers

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

- TIMA status and control register (TASC)
- TIMA counter registers (TACNTH:TACNTL)
- TIMA counter modulo registers (TAMODH:TAMODL)
- TIMA channel status and control registers (TASC0, TASC1, TASC2, and TASC3)
- TIMA channel registers (TACH0H:TACH0L, TACH1H:TACH1L, TACH2H:TACH2L, and TACH3H:TACH3L)

11.10.1 TIMA Status and Control Register

The TIMA status and control register does the following:

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

Address: $0020

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

= Unimplemented

Figure 11-4. TIMA Status and Control Register (TASC)

TOF — TIMA Overflow Flag Bit

This read/write flag is set when the TIMA counter resets to $0000 after reaching the modulo value programmed in the TIMA counter modulo registers. Clear TOF by reading the TIMA status and control register when TOF is set and then writing a logic zero to TOF. If another TIMA 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 = TIMA counter has reached modulo value
0 = TIMA counter has not reached modulo value

TOIE — TIMA Overflow Interrupt Enable Bit

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

1 = TIMA overflow interrupts enabled
0 = TIMA overflow interrupts disabled

TSTOP — TIMA Stop Bit

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

1 = TIMA counter stopped
0 = TIMA counter active

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

TRST — TIMA Reset Bit

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

1 = Prescaler and TIMA counter cleared
0 = No effect

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

PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD6/TACLK pin or one of the seven prescaler outputs as the input to the TIMA counter as **Table 11-2** shows. Reset clears the PS[2:0] bits.

Table 11-2. Prescaler Selection

PS2	PS1	PS0	TIM Clock Source
0	0	0	Internal Bus Clock ÷1
0	0	1	Internal Bus Clock ÷ 2
0	1	0	Internal Bus Clock ÷ 4
0	1	1	Internal Bus Clock ÷ 8
1	0	0	Internal Bus Clock ÷ 16
1	0	1	Internal Bus Clock ÷ 32
1	1	0	Internal Bus Clock ÷ 64
1	1	1	PTD6/TACLK

11.10.2 TIMA Counter Registers

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

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

Address: $0022

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

= Unimplemented

Figure 11-5. TIMA Counter Register High (TACNTH)

Address: $0023

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

= Unimplemented

Figure 11-6. TIMA Counter Register Low (TACNTL)

11.10.3 TIMA Counter Modulo Registers

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

Address: $0024

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

Figure 11-7. TIMA Counter Modulo Register High (TAMODH)

Address: $0025

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

Figure 11-8. TIMA Counter Modulo Register Low (TAMODL)

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

11.10.4 TIMA Channel Status and Control Registers

Each of the TIMA 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 TIMA overflow
- Selects 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Address: $0026

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

Figure 11-9. TIMA Channel 0 Status and Control Register (TASC0)

Address: $0029

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 11-10. TIMA Channel 1 Status and Control Register (TASC1)

Address: $002C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 11-11. TIMA Channel 2 Status and Control Register (TASC2)

Address: $002F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 11-12. TIMA Channel 3 Status and Control Register (TASC3)

CHxF — Channel x Flag Bit

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

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

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

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

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMA CPU interrupts on channel x. Reset clears the CHxIE bit.

1 = Channel x CPU interrupt requests enabled
0 = Channel x CPU interrupt requests disabled

MSxB — Mode Select Bit B

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

Setting MS0B disables the channel 1 status and control register and reverts TCH1B to general-purpose I/O.

Setting MS2B disables the channel 3 status and control register and reverts TCH3B to general-purpose I/O.

Reset clears the MSxB bit.

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

MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. See **Table 11-3**.

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

When ELSxB:A = 00, this read/write bit selects the initial output level of the TBCHx pin. See **Table 11-3**. 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 TIMA status and control register (TASC).*

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to the port I/O, and pin TACHx is available as a general-purpose I/O pin. **Table 11-3** shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

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

MSxB	MSxA	ELSxB	ELSxA	Mode	Configuration
X	0	0	0	Output Preset	Pin under Port Control; Initial Output Level High
X	1	0	0		Pin under Port Control; Initial Output Level Low
0	0	0	1	Input Capture	Capture on Rising Edge Only
0	0	1	0		Capture on Falling Edge Only
0	0	1	1		Capture on Rising or Falling Edge
0	1	0	1	Output Compare or PWM	Toggle Output on Compare
0	1	1	0		Clear Output on Compare
0	1	1	1		Set Output on Compare
1	X	0	1	Buffered Output Compare or Buffered PWM	Toggle Output on Compare
1	X	1	0		Clear Output on Compare
1	X	1	1		Set Output on Compare

NOTE: *Before enabling a TIMA channel register for input capture operation, make sure that the TACHx pin is stable for at least two bus clocks.*

TOVx — Toggle-On-Overflow Bit

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

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

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

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic zero, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As **Figure 11-13** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.

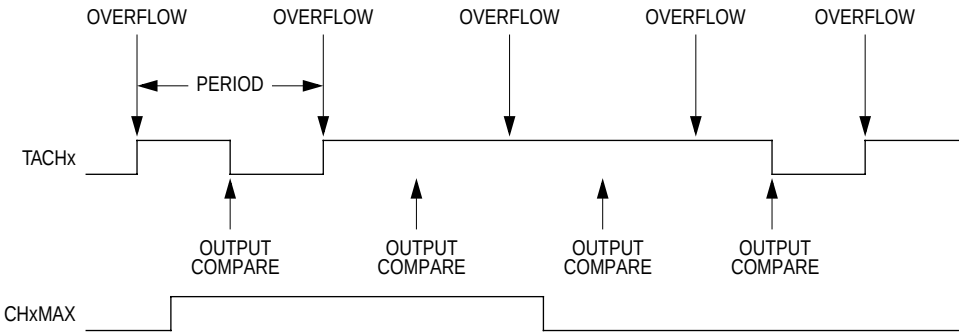


Figure 11-13. CHxMAX Latency

11.10.5 TIMA Channel Registers

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

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

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

Address: $0027

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

Figure 11-14. TIMA Channel 0 Register High (TACH0H)

Address: $0028

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

Figure 11-15. TIMA Channel 0 Register Low (TACH0L)

Address: $002A

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

Figure 11-16. TIMA Channel 1 Register High (TACH1H)

Address: $002B

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

Figure 11-17. TIMA Channel 1 Register Low (TACH1L)

Address: $002D

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

Figure 11-18. TIMA Channel 2 Register High (TACH2H)

Address: $002E

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

Figure 11-19. TIMA Channel 2 Register Low (TACH2L)

Address: $0030

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

Figure 11-20. TIMA Channel 3 Register High (TACH3H)

Address: $0031

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

Figure 11-21. TIMA Channel 3 Register Low (TACH3L)

Section 12. Timer Interface Module B (TIMB)

12.1 Contents

12.2 Introduction

This section describes the timer interface module A (TIMB). The TIMB 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 TIMB.

12.3 Features

Features of the TIMB 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 TIMB clock input:
  - Seven-frequency internal bus clock prescaler selection
  - External TIMB clock input (4MHz maximum frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMB counter stop and reset bits
- Modular architecture expandable to eight channels

12.4 Pin Name Conventions

The TIMB share five I/O pins with port D and F I/O pins. The full name of the TIMB I/O pin is listed in **Table 12-1**. The generic pin name appear in the text that follows.

Table 12-1. Pin Name Conventions

TIMB Generic Pin Names:	Full TIMB Pin Names:
TBCLK	PTD4/TBCLK
TBCH0	PTF4/TBCH0
TBCH1	PTF5/TBCH1
TBCH2	PTF2/TBCH2
TBCH3	PTF3/TBCH3

12.5 Functional Description

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

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

12.5.1 TIMB Counter Prescaler

The TIMB clock source can be one of the seven prescaler outputs or the TIMB clock pin, PTD4/TBCLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMB status and control register select the TIMB clock source.

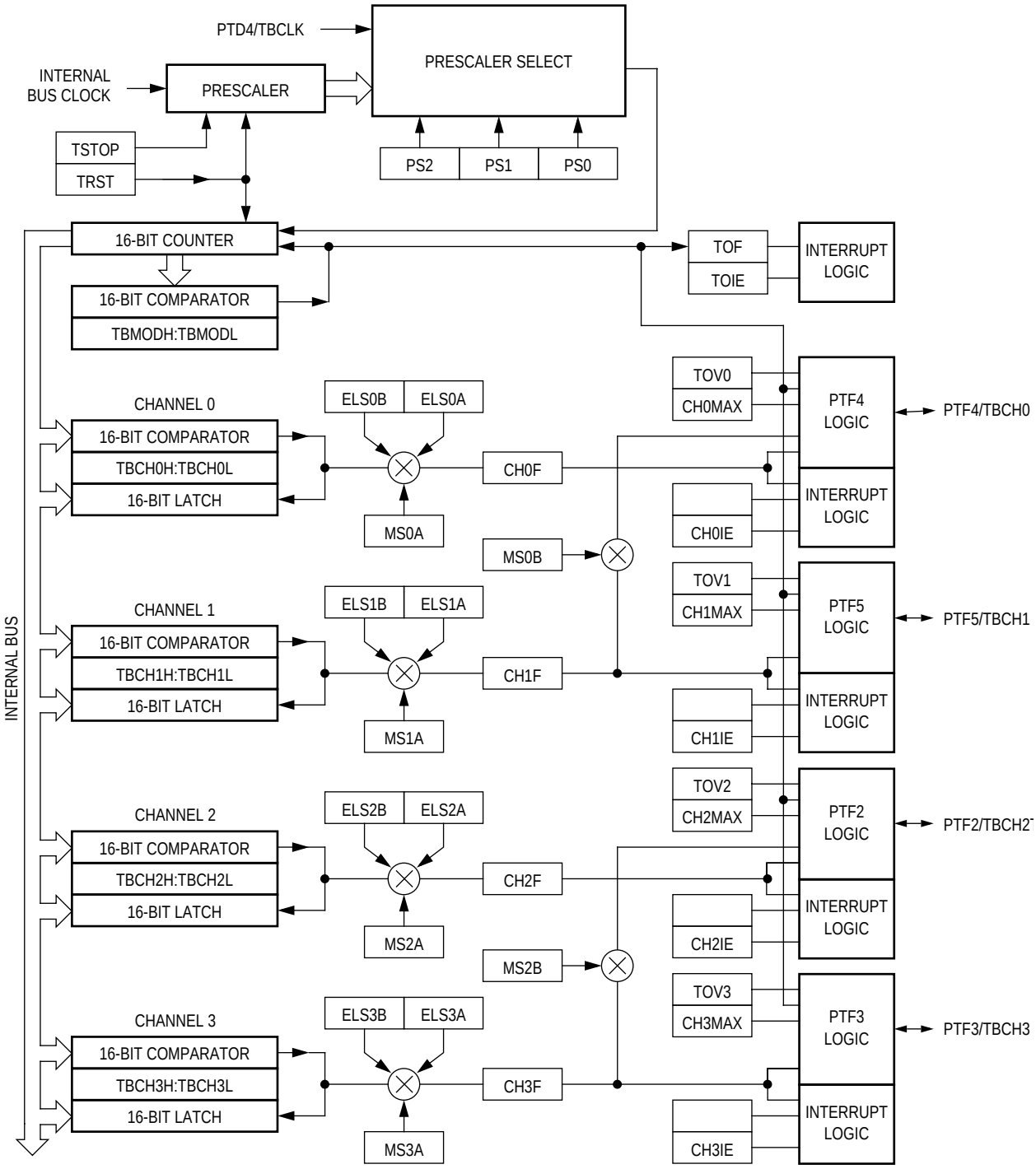


Figure 12-1. TIMB Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0040	Timer B Status and Control Register (TBSC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0041	Timer B Counter Register High (TBCNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0042	Timer B Counter Register Low (TBCNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0043	Timer B Counter Modulo Register High (TBMODH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0044	Timer B Counter Modulo Register Low (TBMODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0045	Timer B Channel 0 Status and Control Register (TBSC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0046	Timer B Channel 0 Register High (TBCH0H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0047	Timer B Channel 0 Register Low (TBCH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0048	Timer B Channel 1 Status and Control Register (TBSC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0049	Timer B Channel 1 Register High (TBCH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							

= Unimplemented

Figure 12-2. TIMB I/O Register Summary (Sheet 1 of 2)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$004A	Timer B Channel 1 Register Low (TBCH1L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0032	Timer B Channel 2 Status and Control Register (TBSC2)	Read: Write:	CH2F 0	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		Reset:	0	0	0	0	0	0	0	0
$0033	Timer B Channel 2 Register High (TBCH2H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0034	Timer B Channel 2 Register Low (TBCH2L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0035	Timer B Channel 3 Status and Control Register (TBSC3)	Read: Write:	CH3F 0	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Reset:	0	0	0	0	0	0	0	0
$0036	Timer B Channel 3 Register High (TBCH3H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0037	Timer B Channel 3 Register Low (TBCH3L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							

= Unimplemented

Figure 12-2. TIMB I/O Register Summary (Sheet 2 of 2)

12.5.2 Input Capture

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

12.5.3 Output Compare

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

12.5.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in **12.5.3 Output Compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMB channel registers.

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

Use 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 TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

12.5.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTF4/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The output compare value in the TIMB channel 0 registers initially controls the output on the PTF4/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the output after the TIMB overflows. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the output are the ones written to last. TBSC0 controls and monitors the buffered output compare function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTF5/TBCH1, 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 PTF2/TBCH2 pin. The TIMB channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIMB channel 2 status and control register (TBSC2) links channel 2 and channel 3. The output compare value in the TIMB channel 2 registers initially controls the output on the PTF2/TBCH2 pin. Writing to the TIMB channel 3 registers enables the TIMB channel 3 registers to synchronously control the output after the TIMB overflows. At each subsequent overflow, the TIMB channel registers (2 or 3) that control the output are the ones written to last. TBSC2 controls and monitors the buffered output compare function, and TIMB channel 3 status and control register (TBSC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF3/TBCH3, 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.*

12.5.4 Pulse Width Modulation (PWM)

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

As **Figure 12-3** shows, the output compare value in the TIMB channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMB to clear the channel pin on output compare if the state of the PWM pulse is logic one. Program the TIMB to set the pin if the state of the PWM pulse is logic zero.

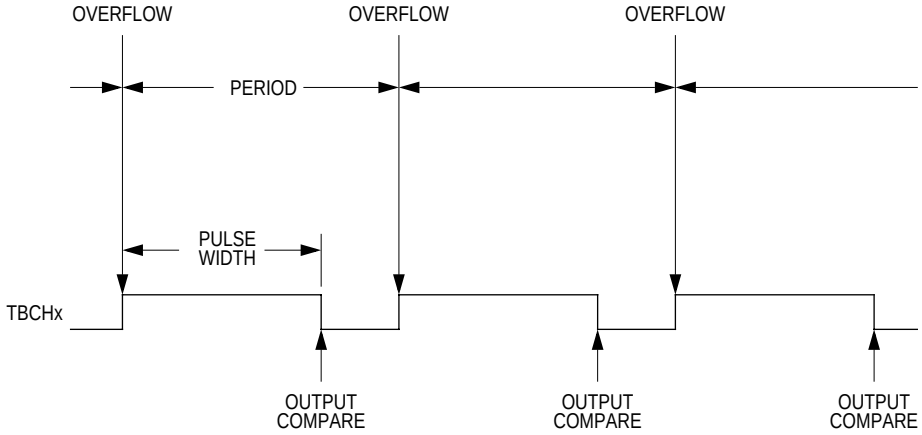


Figure 12-3PWM Period and Pulse Width

The value in the TIMB counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIMB counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000. See **12.10.1 TIMB Status and Control Register**.

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

12.5.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **12.5.4 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 TIMB channel registers.

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

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 TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

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

12.5.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTF4/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The TIMB channel 0 registers initially control the pulse width on the PTF4/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the pulse width are the ones written to last. TBSC0 controls and monitors the buffered PWM function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTF5/TBCH1, 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 PTF2/TBCH2 pin. The TIMB channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS2B bit in TIMB channel 2 status and control register (TBSC2) links channel 2 and channel 3. The TIMB channel 2 registers initially control the pulse width on the PTF2/TBCH2 pin. Writing to the TIMB channel 3 registers enables the TIMB channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMB channel registers (2 or 3) that control the pulse width are the ones written to last. TBSC2 controls and monitors the buffered PWM function, and TIMB channel 3 status and control register (TBSC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF3/TBCH3, 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.*

12.5.4.3 PWM Initialization

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

1. In the TIMB status and control register (TBSC):
   a. Stop the TIMB counter by setting the TIMB stop bit, TSTOP.
   b. Reset the TIMB counter by setting the TIMB reset bit, TRST.
2. In the TIMB counter modulo registers (TBMODH:TBMODL), write the value for the required PWM period.
3. In the TIMB channel x registers (TBCHxH:TBCHxL), write the value for the required pulse width.
4. In TIMB channel x status and control register (TBSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. See **Table 12-3**.
   a. Write 1 to the toggle-on-overflow bit, TOVx.
   b. 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 TIMB status control register (TBSC), clear the TIMB stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMB channel 0 registers (TBCH0H:TBCH0L) initially control the buffered PWM output. TIMB channel 0 status and control register (TBSC0) 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 TIMB channel 2 registers (TBCH2H:TBCH2L) initially control the PWM output. TIMB channel 2 status and control register (TBSC2) 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 TIMB overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100% duty cycle output. See **12.10.4 TIMB Channel Status and Control Registers**.

12.6 Interrupts

The following TIMB sources can generate interrupt requests:

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

12.7 Low-Power Modes

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

12.7.1 Wait Mode

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

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

12.7.2 Stop Mode

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

12.8 TIMB During Break Interrupts

A break interrupt stops the TIMB 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 **8.8.3 SIM Break Flag Control Register**.

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.

12.9 I/O Signals

Port F shares four pins with the TIMB and port D shares one. PTD4/TBCLK is an external clock input to the TIMB prescaler. The four TIMB channel I/O pins are PTF4/TBCH0, PTF5/TBCH1, PTF2/TBCH2, and PTF3/TBCH3.

12.9.1 TIMB Clock Pin

PTD4/TBCLK is an external clock input that can be the clock source for the TIMB counter instead of the prescaled internal bus clock. Select the PTD4/TBCLK input by writing logic 1s to the three prescaler select bits, PS[2:0]. See **12.10.1 TIMB Status and Control Register**. The minimum TBCLK pulse width, $TBCLK_{LMIN}$ or $TBCLK_{HMIN}$, is:

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

The maximum TBCLK frequency is:

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

PTD4/TBCLK is available as a general-purpose I/O pin when not used as the TIMB clock input. When the PTD4/TBCLK pin is the TIMB clock input, it is an input regardless of the state of the DDRD6 bit in data direction register D.

12.9.2 TIMB Channel I/O Pins

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTF2/TBCH2 and PTF5/TBCH1 can be configured as buffered output compare or buffered PWM pins.

12.10 I/O Registers

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

- TIMB status and control register (TBSC)
- TIMB counter registers (TBCNTH:TBCNTL)
- TIMB counter modulo registers (TBMODH:TBMODL)
- TIMB channel status and control registers (TBSC0, TBSC1, TBSC2, and TBSC3)
- TIMB channel registers (TBCH0H:TBCH0L, TBCH1H:TBCH1L, TBCH2H:TBCH2L, and TBCH3H:TBCH3L)

12.10.1 TIMB Status and Control Register

The TIMB status and control register does the following:

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

Address: $0040

	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 12-4. TIMB Status and Control Register (TBSC)

TOF — TIMB Overflow Flag Bit

This read/write flag is set when the TIMB counter resets to $0000 after reaching the modulo value programmed in the TIMB counter modulo registers. Clear TOF by reading the TIMB status and control register when TOF is set and then writing a logic zero to TOF. If another TIMB 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 = TIMB counter has reached modulo value
0 = TIMB counter has not reached modulo value

TOIE — TIMB Overflow Interrupt Enable Bit

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

1 = TIMB overflow interrupts enabled
0 = TIMB overflow interrupts disabled

TSTOP — TIMB Stop Bit

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

1 = TIMB counter stopped
0 = TIMB counter active

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

TRST — TIMB Reset Bit

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

1 = Prescaler and TIMB counter cleared
0 = No effect

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

PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD4/TBCLK pin or one of the seven prescaler outputs as the input to the TIMB counter as **Table 12-2** shows. Reset clears the PS[2:0] bits.

Table 12-2. Prescaler Selection

PS2	PS1	PS0	TIM Clock Source
0	0	0	Internal Bus Clock ÷1
0	0	1	Internal Bus Clock ÷ 2
0	1	0	Internal Bus Clock ÷ 4
0	1	1	Internal Bus Clock ÷ 8
1	0	0	Internal Bus Clock ÷ 16
1	0	1	Internal Bus Clock ÷ 32
1	1	0	Internal Bus Clock ÷ 64
1	1	1	PTD4/TBCLK

12.10.2 TIMB Counter Registers

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

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

Address: $0041

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

= Unimplemented

Figure 12-5. TIMB Counter Register High (TBCNTH)

Address: $0042

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

= Unimplemented

Figure 12-6. TIMB Counter Register Low (TBCNTL)

12.10.3 TIMB Counter Modulo Registers

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

Address: $0043

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

Figure 12-7. TIMB Counter Modulo Register High (TBMODH)

Address: $0044

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

Figure 12-8. TIMB Counter Modulo Register Low (TBMODL)

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

12.10.4 TIMB Channel Status and Control Registers

Each of the TIMB 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 TIMB overflow
- Selects 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Address: $0045

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

Figure 12-9. TIMB Channel 0 Status and Control Register (TBSC0)

Address: $0048

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 12-10. TIMB Channel 1 Status and Control Register (TBSC1)

Address: $0032

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 12-11. TIMB Channel 2 Status and Control Register (TBSC2)

Address: $0035

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Figure 12-12. TIMB Channel 3 Status and Control Register (TBSC3)

CHxF — Channel x Flag Bit

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

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

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

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

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMB CPU interrupts on channel x.
Reset clears the CHxIE bit.

1 = Channel x CPU interrupt requests enabled
0 = Channel x CPU interrupt requests disabled

MSxB — Mode Select Bit B

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

Setting MS0B disables the channel 1 status and control register and reverts TCH1B to general-purpose I/O.

Setting MS2B disables the channel 3 status and control register and reverts TCH3B 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**.

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 TBCHx 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 TIMB status and control register (TBSC).*

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to the port I/O, and pin TBCHx 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
X	0	0	0	Output Preset	Pin under Port Control; Initial Output Level High
X	1	0	0		Pin under Port Control; Initial Output Level Low
0	0	0	1	Input Capture	Capture on Rising Edge Only
0	0	1	0		Capture on Falling Edge Only
0	0	1	1		Capture on Rising or Falling Edge
0	1	0	1	Output Compare or PWM	Toggle Output on Compare
0	1	1	0		Clear Output on Compare
0	1	1	1		Set Output on Compare
1	X	0	1	Buffered Output Compare or Buffered PWM	Toggle Output on Compare
1	X	1	0		Clear Output on Compare
1	X	1	1		Set Output on Compare

NOTE: *Before enabling a TIMB channel register for input capture operation, make sure that the TBCHx pin is stable for at least two bus clocks.*

TOVx — Toggle-On-Overflow Bit

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

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

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

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic zero, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As **Figure 12-13** 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.

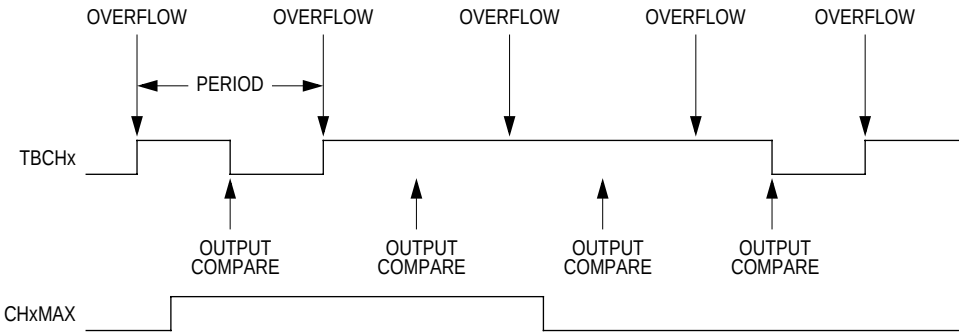


Figure 12-13. CHxMAX Latency

12.10.5 TIMB Channel Registers

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

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

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

Address: $0046

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

Figure 12-14. TIMB Channel 0 Register High (TBCH0H)

Address: $0047

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

Figure 12-15. TIMB Channel 0 Register Low (TBCH0L)

Address: $0049

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

Figure 12-16. TIMB Channel 1 Register High (TBCH1H)

Address: $004A

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

Figure 12-17. TIMB Channel 1 Register Low (TBCH1L)

Address: $0033

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

Figure 12-18. TIMB Channel 2 Register High (TBCH2H)

Address: $0034

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

Figure 12-19. TIMB Channel 2 Register Low (TBCH2L)

Address: $0036

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

Figure 12-20. TIMB Channel 3 Register High (TBCH3H)

Address: $0037

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

Figure 12-21. TIMB Channel 3 Register Low (TBCH3L)

Section 13. Programmable Interrupt Timer (PIT)

13.1 Contents

13.2 Introduction

This section describes the programmable interrupt timer (PIT), which is a timer whose counter is clocked internally via software programmable options. **Figure 13-1** is a block diagram of the PIT.

13.3 Features

Features of the PIT include the following:

- Programmable PIT clock input
- Free-running or modulo up-count operation
- PIT counter stop and reset bits

13.4 Functional Description

Figure 13-1 shows the structure of the PIT. The central component of the PIT is the 16-bit PIT counter that can operate as a free-running counter or a modulo up-counter. The counter provides the timing reference for the interrupt. The PIT counter modulo registers, PMODH:PMODL, control the modulo value of the counter. Software can read the counter value at any time without affecting the counting sequence.

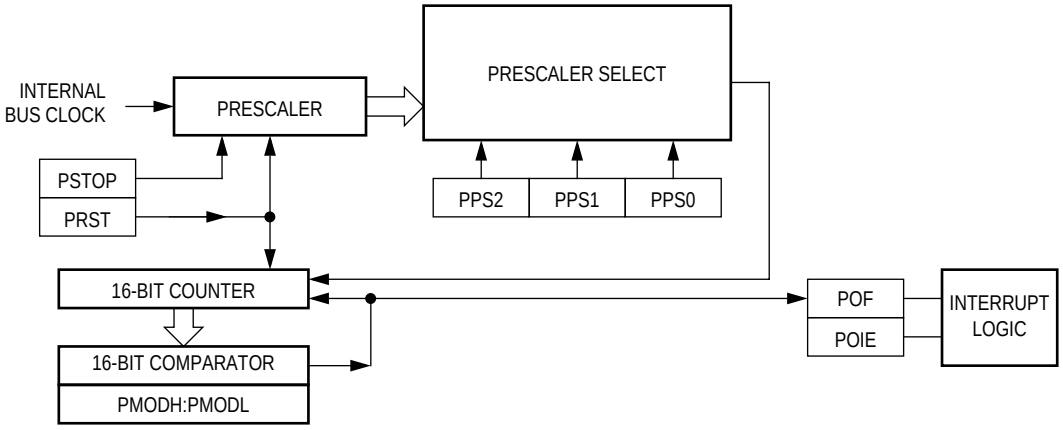


Figure 13-1. PIT Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$004B	PIT Status and Control Register (PSC)	Read:	POF	POIE	PSTOP	0	0	PPS2	PPS1	PPS0
		Write:	0			PRST				
		Reset:	0	0	1	0	0	0	0	0
$004C	PIT Counter Register High (PCNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004D	PIT Counter Register Low (PCNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004E	PIT Counter Modulo Register High (PMODH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$004F	PIT Counter Modulo Register Low (PMODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1

= Unimplemented

Figure 13-2. PIT I/O Register Summary

13.4.1 PIT Counter Prescaler

The clock source can be one of the seven prescaler outputs. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PPS[2:0] in the status and control register select the PIT clock source.

The value in the PIT counter modulo registers and the selected prescaler output determines the frequency of the Periodic Interrupt. The PIT overflow flag (POF) is set when the PIT counter value rolls over to $0000 after matching the value in the PIT counter modulo registers. The PIT interrupt enable bit, POIE, enables PIT overflow CPU interrupt requests. POF and POIE are in the PIT status and control register.

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

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

13.5.2 Stop Mode

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

13.6 PIT During Break Interrupts

A break interrupt stops the PIT 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 **8.8.3 SIM Break Flag Control Register**.

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 Registers

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

- PIT status and control register (PSC)
- PIT counter registers (PCNTH:PCNTL)
- PIT counter modulo registers (PMODH:PMODL)

13.7.1 PIT Status and Control Register

The PIT status and control register does the following:

- Enables PIT interrupt
- Flags PIT overflows
- Stops the PIT counter
- Resets the PIT counter
- Prescales the PIT counter clock

Address: $004B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	POF	POIE	PSTOP	0	0	PPS2	PPS1	PPS0
Write:	0			PRST				
Reset:	0	0	1	0	0	0	0	0

= Unimplemented

Figure 13-3. PIT Status and Control Register (PSC)

POF — PIT Overflow Flag Bit

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

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

POIE — PIT Overflow Interrupt Enable Bit

This read/write bit enables PIT overflow interrupts when the POF bit becomes set. Reset clears the POIE bit.

1 = PIT overflow interrupts enabled
0 = PIT overflow interrupts disabled

PSTOP — PIT Stop Bit

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

1 = PIT counter stopped
0 = PIT counter active

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

PRST — PIT Reset Bit

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

1 = Prescaler and PIT counter cleared
0 = No effect

NOTE: *Setting the PSTOP and PRST bits simultaneously stops the PIT counter at a value of $0000.*

PPS[2:0] — PIT Prescaler Select Bits

These read/write bits select one of the seven prescaler outputs as the input to the PIT counter as **Table 13-1** shows. Reset clears the PPS[2:0] bits.

Table 13-1. PIT Prescaler Selection

PPS2	PPS1	PPS0	PIT Clock Source
0	0	0	Internal Bus Clock ÷1
0	0	1	Internal Bus Clock ÷ 2
0	1	0	Internal Bus Clock ÷ 4
0	1	1	Internal Bus Clock ÷ 8
1	0	0	Internal Bus Clock ÷ 16
1	0	1	Internal Bus Clock ÷ 32
1	1	0	Internal Bus Clock ÷ 64
1	1	1	Internal Bus Clock ÷ 64

13.7.2 PIT Counter Registers

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

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

Address: $004C

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

= Unimplemented

Figure 13-4. PIT Counter Register High (PCNTH)

Address: $004D

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

= Unimplemented

Figure 13-5. PIT Counter Register Low (PCNTL)

13.7.3 PIT Counter Modulo Registers

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

Address: $004E

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

Figure 13-6. PIT Counter Modulo Register High (PMODH)

Address: $004F

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

Figure 13-7. PIT Counter Modulo Register Low (PMODL)

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

Section 14. Analog-to-Digital Converter (ADC)

14.1 Contents

14.2 Introduction

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

14.3 Features

Features of the ADC module include:

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0038	ADC Status and Control Register (ADSCR)	Read: / Write:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Reset:	0	0	0	1	1	1	1	1
$0039	ADC Data Register (ADR)	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$003A	ADC Clock Register (ADCLK)	Read:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-1. ADC Register Summary

14.4 Functional Description

The ADC provides eight pins for sampling external sources at pins PTB7/ATD7–PTB0/ATD0. An analog multiplexer allows the single ADC converter to select one of eight ADC channels as ADC voltage in (V_{ADIN}). V_{ADIN} is converted by the successive approximation register-based analog-to-digital converter. When the conversion is completed, ADC places the result in the ADC data register and sets a flag or generates an interrupt. (See **Figure 14-2**.)

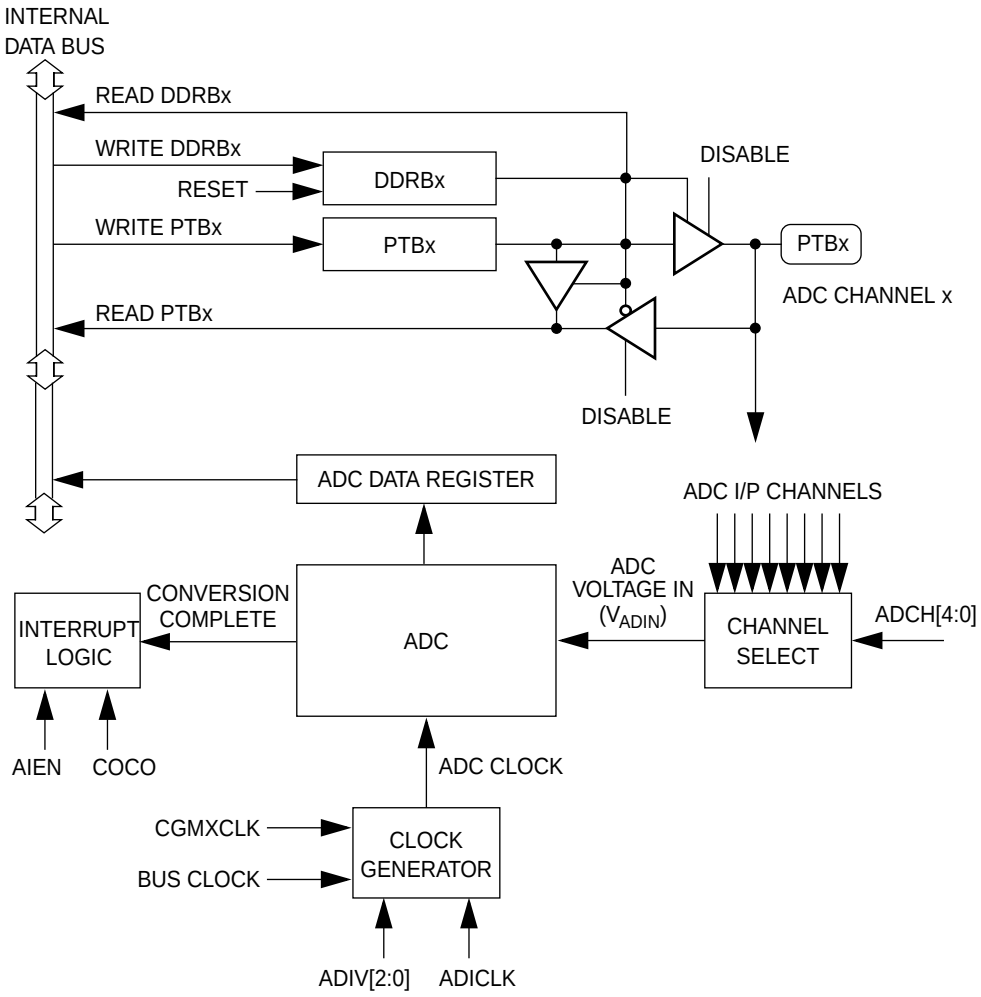


Figure 14-2. ADC Block Diagram

14.4.1 ADC Port I/O Pins

PTB7/ATD7–PTB0/ATD0 are general-purpose I/O (input/output) pins that share with the ADC channels. The channel select bits define which ADC channel/port pin will be used as the input signal. The ADC overrides the port I/O logic by forcing that pin as input to the ADC. The remaining ADC channels/port pins are controlled by the port I/O logic and can be used as general-purpose I/O. Writes to the port register or DDR will not have any affect on the port pin that is selected by the ADC. Read of a port pin in use by the ADC will return a logic 0.

14.4.2 Voltage Conversion

When the input voltage to the ADC equals V_{REFH}, the ADC converts the signal to $FF (full scale). If the input voltage equals V_{REFL}, the ADC converts it to $00. Input voltages between V_{REFH} and V_{REFL} are a straight-line linear conversion.

14.4.3 Conversion Time

Conversion starts after a write to the ADSCR. One conversion will take between 16 and 17 ADC clock cycles. The ADIVx and ADICLK bits should be set to provide a 1-MHz ADC clock frequency.

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

$$\text{Number of bus cycles} = \text{conversion time} \times \text{bus frequency}$$

14.4.4 Conversion

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

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

14.4.5 Accuracy and Precision

The conversion process is monotonic and has no missing codes.

14.5 Interrupts

When the AIEN bit is set, the ADC module is capable of generating CPU interrupts after each ADC conversion. A CPU interrupt is generated if the COCO bit is at logic 0. The COCO bit is not used as a conversion complete flag when interrupts are enabled.

14.6 Low-Power Modes

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

14.6.1 Wait Mode

The ADC continues normal operation during wait mode. Any enabled CPU interrupt request from the ADC can bring the MCU out of wait mode. If the ADC is not required to bring the MCU out of wait mode, power down the ADC by setting ADCH[4:0] bits in the ADC status and control register before executing the WAIT instruction.

14.6.2 Stop Mode

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

14.7 I/O Signals

The ADC module has eight pins shared with port B, PTB7/ATD7–PTB0/ATD0.

14.7.1 ADC Analog Power Pin (V_{DDAREF})

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

NOTE: *For maximum noise immunity, route V_{DDAREF} carefully and place bypass capacitors as close as possible to the package.*

14.7.2 ADC Analog Ground Pin (A_{VSS}/V_{REFL})

The ADC analog portion uses A_{VSS}/V_{REFL} as its ground pin. Connect the A_{VSS}/V_{REFL} pin to the same voltage potential as V_{SS}.

NOTE: *Route A_{VSS}/V_{REFL} cleanly to avoid any offset errors.*

14.7.3 ADC Voltage Reference High Pin (V_{REFH})

V_{REFH} is the reference voltage for the ADC.

14.7.4 ADC Voltage In (V_{ADIN})

V_{ADIN} is the input voltage signal from one of the eight ADC channels to the ADC module.

14.8 I/O Registers

These I/O registers control and monitor ADC operation:

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

14.8.1 ADC Status and Control Register (ADSCR)

Function of the ADC status and control register is described here.

Address: $0038

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

Figure 14-3. ADC Status and Control Register (ADSCR)

COCO — Conversions Complete

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

If the AIEN bit is a logic 1, the COCO becomes a read/write bit, which should be cleared to logic 0 for CPU to service the ADC interrupt request. Reset clears the COCO bit.

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

AIEN — ADC Interrupt Enable Bit

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

1 = ADC interrupt enabled
0 = ADC interrupt disabled

ADCO — ADC Continuous Conversion Bit

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

1 = Continuous ADC conversion
0 = One ADC conversion

ADCH[4:0] — ADC Channel Select Bits

ADCH[4:0] form a 5-bit field which is used to select one of the eight ADC channels, ATD7–ATD0. The channels are detailed in **Table 14-1**. Care should be taken when using a port pin as both an analog and digital input simultaneously to prevent switching noise from corrupting the analog signal.

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

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

The voltage levels supplied from internal reference nodes, as specified in **Table 14-1**, are used to verify the operation of the ADC converter both in production test and for user applications.

Table 14-1. Mux Channel Select

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	Input Select
0	0	0	0	0	PTB0/ATD0
0	0	0	0	1	PTB1/ATD1
0	0	0	1	0	PTB2/ATD2
0	0	0	1	1	PTB3/ATD3
0	0	1	0	0	PTB4/ATD4
0	0	1	0	1	PTB5/ATD5
0	0	1	1	0	PTB6/ATD6
0	0	1	1	1	PTB7/ATD7
0 ↓ 1	1 ↓ 1	0 ↓ 1	0 ↓ 0	0 ↓ 0	Reserved
1	1	1	0	1	V_{REFH}
1	1	1	1	0	V_{REFL}
1	1	1	1	1	ADC power off

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

14.8.2 ADC Data Register (ADR)

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

Address: $0039

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

= Unimplemented

Figure 14-4. ADC Data Register (ADR)

14.8.3 ADC Clock Register (ADCLK)

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

Address: $003A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-5. ADC Clock Register (ADCLK)

ADIV[2:0] — ADC Clock Prescaler Bits

ADIV[2:0] form a 3-bit field which selects the divide ratio used by the ADC to generate the internal ADC clock. **Table 14-2** shows the available clock configurations. The ADC clock should be set to approximately 1 MHz.

Table 14-2. ADC Clock Divide Ratio

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

X = don’t care

ADICLK — ADC Input Clock Select Bit

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

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

1 = Internal bus clock

0 = External clock (CGMXCLK)

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

Section 15. Serial Communications Interface Module (SCI)

15.1 Contents

15.2 Introduction

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

NOTE: *References to DMA (direct-memory access) and associated functions are only valid if the MCU has a DMA module. This MCU does not have the DMA function. Any DMA-related register bits should be left in their reset state for normal MCU operation.*

15.3 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
- Programmable transmitter output polarity

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

15.4 Pin Name Conventions

The generic names of the SCI I/O pins are:

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

SCI I/O (input/output) lines are implemented by sharing parallel I/O port pins. The full name of an SCI input or output reflects the name of the shared port pin. **Table 15-1** shows the full names and the generic names of the SCI I/O pins.

The generic pin names appear in the text of this section.

Table 15-1. Pin Name Conventions

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

15.5 Functional Description

Figure 15-1 shows the structure of the SCI module. The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The transmitter and receiver of the SCI operate independently, although they use the same baud rate generator. During normal operation, the CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

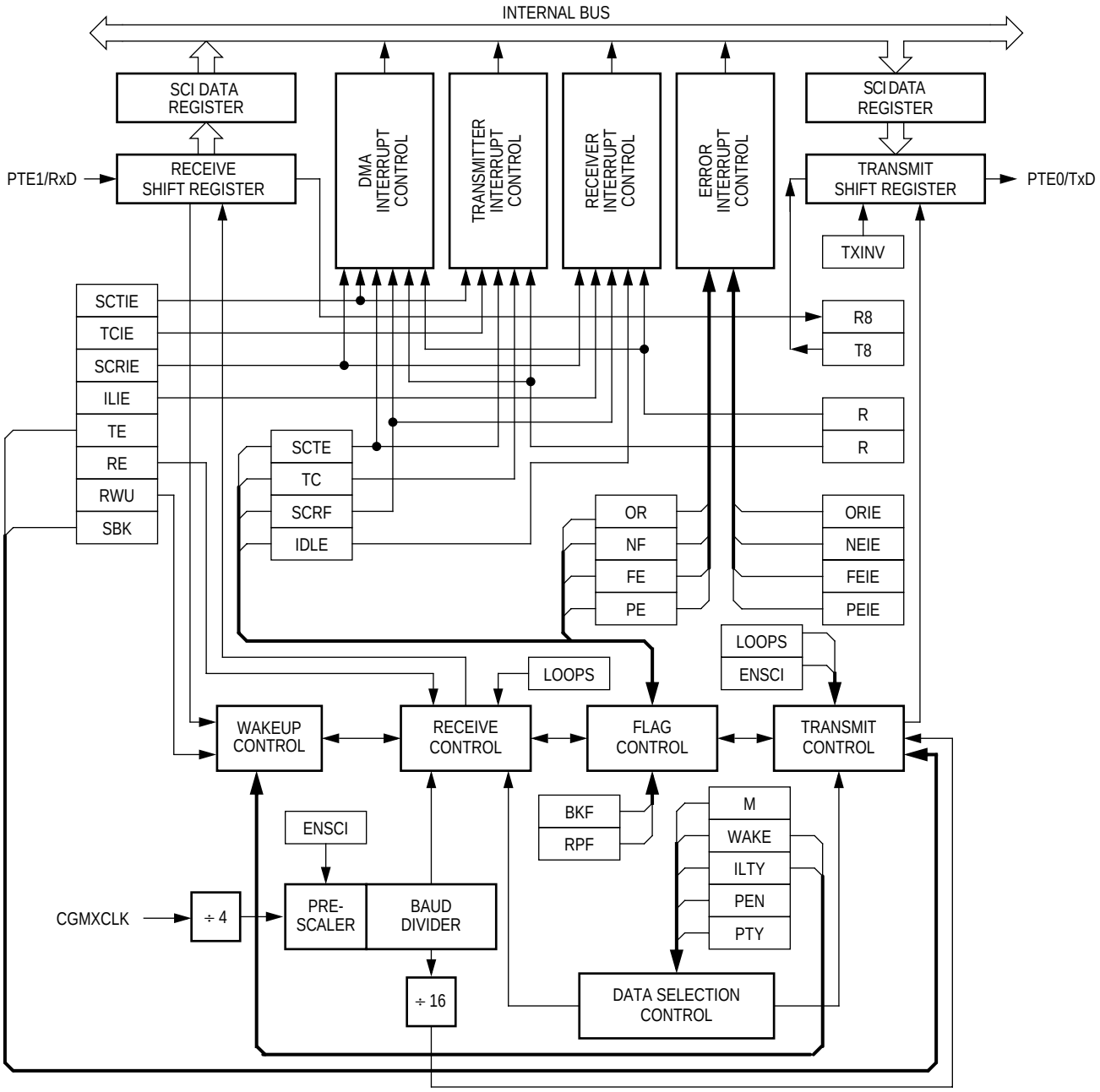


Figure 15-1. SCI Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0013	SCI Control Register 1 (SCC1)	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0014	SCI Control Register 2 (SCC2)	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0015	SCI Control Register 3 (SCC3)	Read:	R8	T8	R	R	ORIE	NEIE	FEIE	PEIE
		Write:								
		Reset:	U	U	0	0	0	0	0	0
$0016	SCI Status Register 1 (SCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$0017	SCI Status Register 2 (SCS2)	Read:							BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	SCI Data Register (SCDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0019	SCI Baud Rate Register (SCBR)	Read:			SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Figure 15-2. SCI I/O Register Summary

15.5.1 Data Format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in **Figure 15-3**.

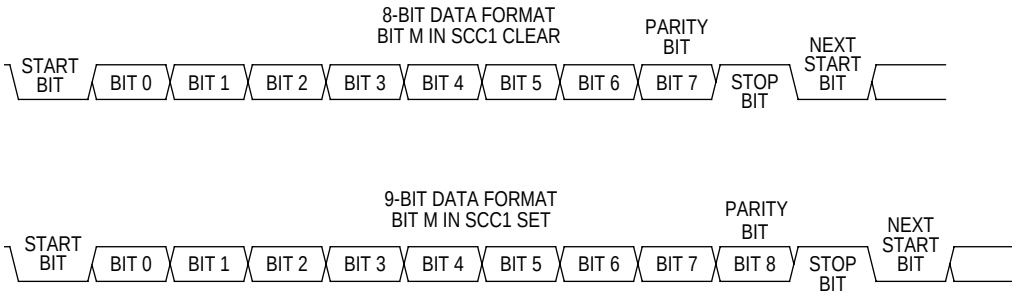


Figure 15-3. SCI Data Formats

15.5.2 Transmitter

Figure 15-4 shows the structure of the SCI transmitter.

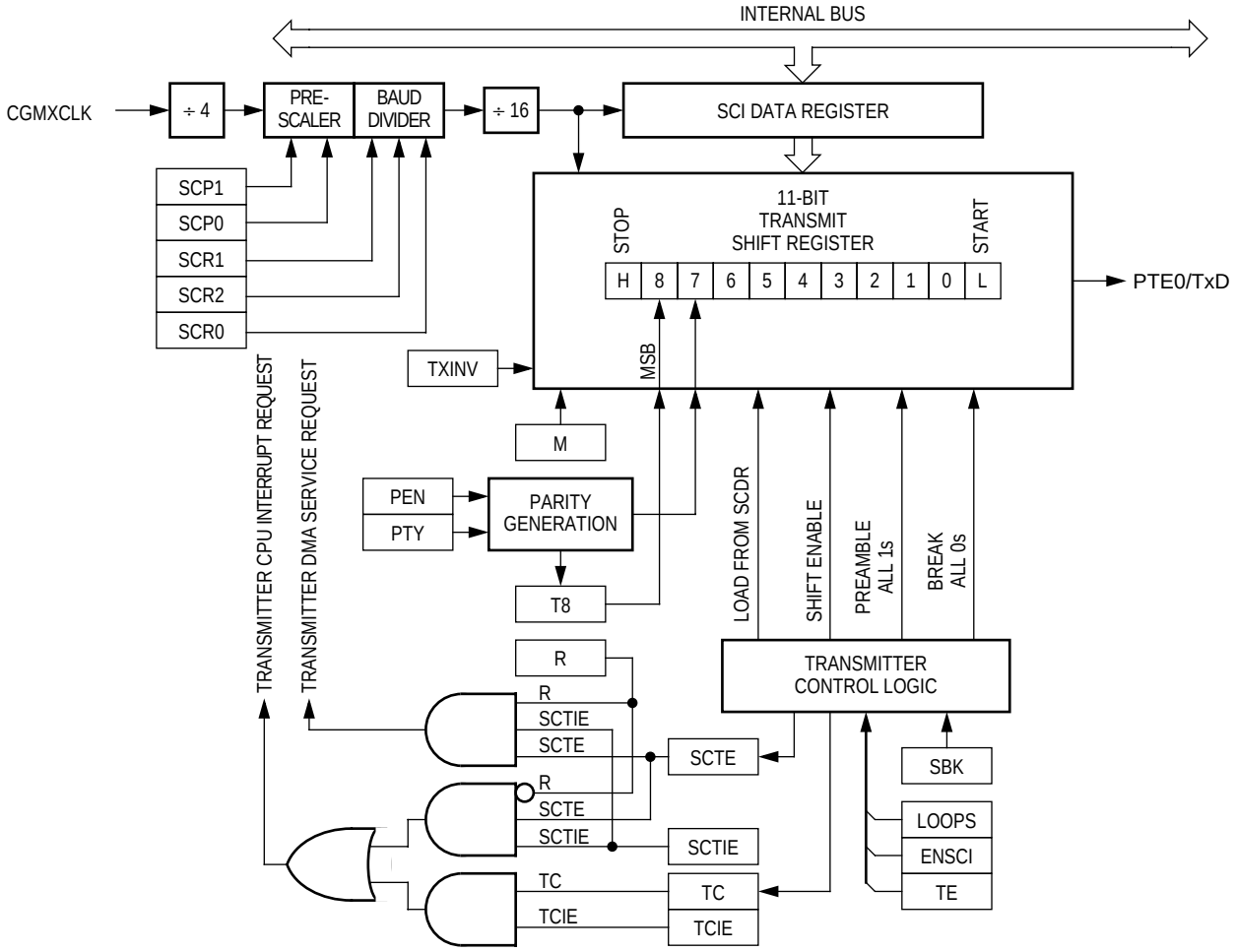


Figure 15-4. SCI Transmitter

15.5.2.1 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).

15.5.2.2 Character Transmission

During an SCI transmission, the transmit shift register shifts a character out to the PTE0/TxD pin. The SCI data register (SCDR) is the write-only buffer between the internal data bus and the transmit shift register. To initiate an SCI transmission:

1. Enable the SCI by writing a logic 1 to the enable SCI bit (ENSCI) in SCI control register 1 (SCC1).
2. Enable the transmitter by writing a logic 1 to the transmitter enable bit (TE) in SCI control register 2 (SCC2).
3. Clear the SCI transmitter empty bit by first reading SCI status register 1 (SCS1) and then writing to the SCDR.
4. Repeat step 3 for each subsequent transmission.

At the start of a transmission, transmitter control logic automatically loads the transmit shift register with a preamble of logic 1s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 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 a transmitter CPU interrupt request.

When the transmit shift register is not transmitting a character, the PTE0/TxD pin goes to the idle condition, logic 1. If at any time software clears the ENSCI bit in SCI control register 1 (SCC1), the transmitter and receiver relinquish control of the port E pins.

15.5.2.3 Break Characters

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

The SCI recognizes a break character when a start bit is followed by eight or nine logic 0 data bits and a logic 0 where the stop bit should be.

Receiving a break character has these effects on SCI registers:

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

15.5.2.4 Idle Characters

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

If the TE bit is cleared during a transmission, the PTE0/TxD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the character currently being transmitted.

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

Toggle the TE bit for a queued idle character when the SCTE bit becomes set and just before writing the next byte to the SCDR.

15.5.2.5 Inversion of Transmitted Output

The transmit inversion bit (TXINV) in SCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values, including idle, break, start, and stop bits, are inverted when TXINV is at logic 1. (See **15.9.1 SCI Control Register 1**.)

15.5.2.6 Transmitter Interrupts

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

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

15.5.3 Receiver

Figure 15-5 shows the structure of the SCI receiver.

15.5.3.1 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). When receiving 8-bit data, bit R8 is a copy of the eighth bit (bit 7).

15.5.3.2 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 a receiver CPU interrupt request.

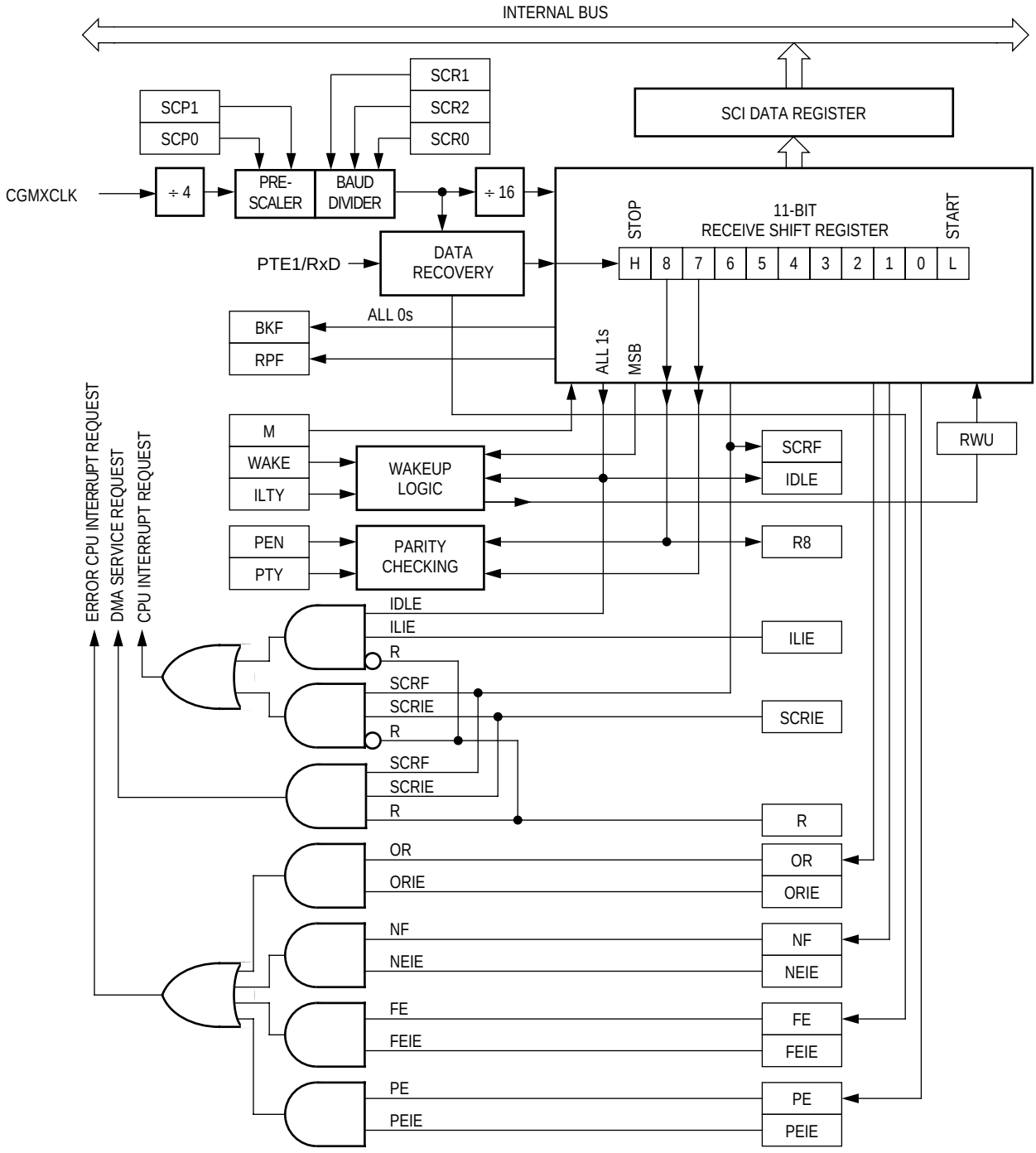


Figure 15-5. SCI Receiver Block Diagram

15.5.3.3 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. To adjust for baud rate mismatch, the RT clock is resynchronized at the following times (see **Figure 15-6**):

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

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

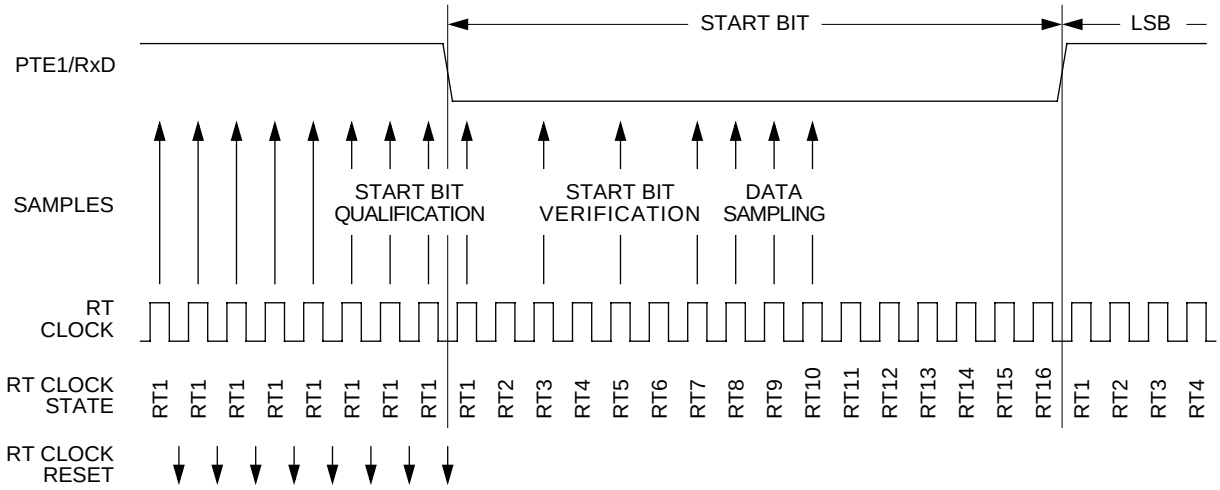


Figure 15-6. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. **Table 15-2** summarizes the results of the start bit verification samples.

Table 15-2. Start Bit Verification

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

Start bit verification is not successful if any two of the three verification samples are logic 1s. If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. **Table 15-3** summarizes the results of the data bit samples.

Table 15-3. Data Bit Recovery

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

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

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. **Table 15-4** summarizes the results of the stop bit samples.

Table 15-4. Stop Bit Recovery

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

15.5.3.4 Framing Errors

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

15.5.3.5 Baud Rate Tolerance

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

tolerance is much more than the degree of misalignment that is likely to occur.

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

Slow Data Tolerance

Figure 15-7 shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

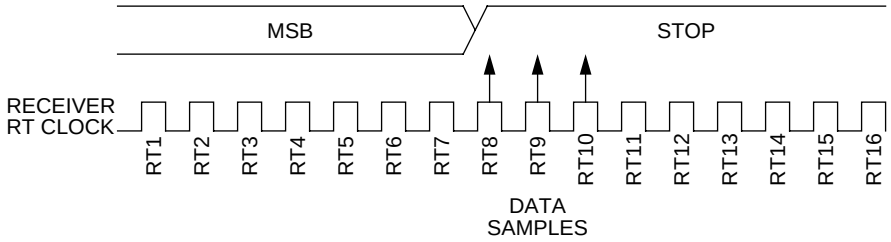


Figure 15-7. Slow Data

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

With the misaligned character shown in **Figure 15-7**, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit times × 16 RT cycles + 3 RT cycles = 147 RT cycles.

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

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

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

With the misaligned character shown in **Figure 15-7**, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles.

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

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

Fast Data Tolerance

Figure 15-8 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

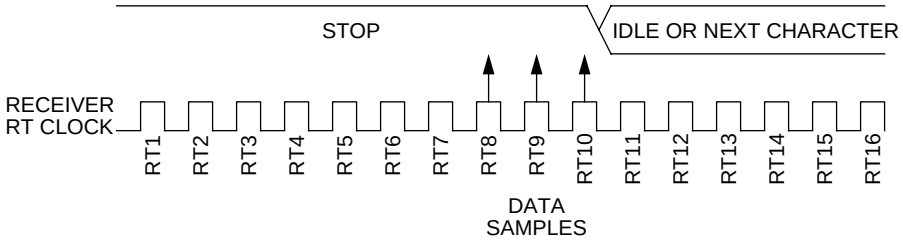


Figure 15-8. Fast Data

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

With the misaligned character shown in **Figure 15-8**, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles = 160 RT cycles.

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

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

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

With the misaligned character shown in **Figure 15-8**, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times × 16 RT cycles = 176 RT cycles.

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

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

15.5.3.6 Receiver Wakeup

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

Depending on the state of the WAKE bit in SCC1, either of two conditions on the PTE1/RxD pin can bring the receiver out of the standby state:

- Address mark — An address mark is a logic 1 in the most significant bit position of a received character. When the WAKE bit is set, an address mark wakes the receiver from the standby state by clearing the RWU bit. The address mark also sets the SCI receiver full bit, SCRF. Software can then compare the character containing the address mark to the user-defined address of the receiver. If they are the same, the receiver remains awake and processes the characters that follow. If they are not the same, software can set the RWU bit and put the receiver back into the standby state.
- Idle input line condition — When the WAKE bit is clear, an idle character on the PTE1/RxD pin wakes the receiver from the standby state by clearing the RWU bit. The idle character that wakes the receiver does not set the receiver idle bit, IDLE, or the

SCI receiver full bit, SCRF. The idle line type bit, ILTY, determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit.

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

15.5.3.7 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 a receiver CPU interrupt request. Setting the SCI receive interrupt enable bit, SCRIE, in SCC2 enables the SCRF bit to generate receiver CPU interrupts.
- Idle input (IDLE) — The IDLE bit in SCS1 indicates that 10 or 11 consecutive logic 1s shifted in from the PTE1/RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables the IDLE bit to generate CPU interrupt requests.

15.5.3.8 Error Interrupts

The following receiver error flags in SCS1 can generate CPU interrupt requests:

- Receiver overrun (OR) — The OR bit indicates that the receive shift register shifted in a new character before the previous character was read from the SCDR. The previous character remains in the SCDR, and the new character is lost. The overrun interrupt enable bit, ORIE, in SCC3 enables OR to generate SCI error CPU interrupt requests.
- Noise flag (NF) — The NF bit is set when the SCI detects noise on incoming data or break characters, including start, data, and stop bits. The noise error interrupt enable bit, NEIE, in SCC3 enables NF to generate SCI error CPU interrupt requests.

- Framing error (FE) — The FE bit in SCS1 is set when a logic 0 occurs where the receiver expects a stop bit. The framing error interrupt enable bit, FEIE, in SCC3 enables FE to generate SCI error CPU interrupt requests.
- 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 to generate SCI error CPU interrupt requests.

15.6 Low-Power Modes

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

15.6.1 Wait Mode

The 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.

Refer to **8.7 Low-Power Modes** for information on exiting wait mode.

15.6.2 Stop Mode

The SCI module is inactive after the execution of a STOP instruction, and thus the SCI cannot cause an interrupt to exit stop mode. The STOP instruction does not affect SCI register states. SCI module operation resumes after an external interrupt.

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

Refer to **8.7 Low-Power Modes** for information on exiting stop mode.

15.7 SCI During Break Module 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.

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

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

15.8 I/O Signals

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

- PTE0/TxD — Transmit data
- PTE1/RxD — Receive data

15.8.1 PTE0/TxD (Transmit Data)

The PTE0/TxD pin is the serial data output from the SCI transmitter. The SCI shares the PTE0/TxD pin with port E. When the SCI is enabled, the PTE0/TxD pin is an output regardless of the state of the DDRE2 bit in data direction register E (DDRE).

15.8.2 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).

15.9 I/O Registers

These 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)

15.9.1 SCI Control Register 1

SCI control register 1:

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

Address: $0013

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

Figure 15-9. SCI Control Register 1 (SCC1)

LOOPS — Loop Mode Select Bit

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

1 = Loop mode enabled
0 = Normal operation enabled

ENSCI — Enable SCI Bit

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

1 = SCI enabled
0 = SCI disabled

TXINV — Transmit Inversion Bit

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

1 = Transmitter output inverted
0 = Transmitter output not inverted

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

M — Mode (Character Length) Bit

This read/write bit determines whether SCI characters are eight or nine bits long. (See **Table 15-5**.) The ninth bit can serve as an extra stop bit, as a receiver wakeup signal, or as a parity bit. Reset clears the M bit.

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

WAKE — Wakeup Condition Bit

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

1 = Address mark wakeup
0 = Idle line wakeup

ILTY — Idle Line Type Bit

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

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

PEN — Parity Enable Bit

This read/write bit enables the SCI parity function. (See **Table 15-5**.) When enabled, the parity function inserts a parity bit in the most significant bit position. (See **Figure 15-3**.) Reset clears the PEN bit.

1 = Parity function enabled
0 = Parity function disabled

PTY — Parity Bit

This read/write bit determines whether the SCI generates and checks for odd parity or even parity. (See **Table 15-5**.) Reset clears the PTY bit.

1 = Odd parity
0 = Even parity

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

Table 15-5. Character Format Selection

Control Bits		Character Format				
M	**PEN and 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

15.9.2 SCI Control Register 2

SCI control register 2:

- Enables the following CPU interrupt requests:
  - Enables the SCTE bit to generate transmitter CPU interrupt requests
  - Enables the TC bit to generate transmitter CPU interrupt requests
  - Enables the SCRF bit to generate receiver CPU interrupt requests
  - Enables the IDLE bit to generate receiver CPU interrupt requests

- Enables the transmitter
- Enables the receiver
- Enables SCI wakeup
- Transmits SCI break characters

Address: $0014

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

Figure 15-10. SCI Control Register 2 (SCC2)

SCTIE — SCI Transmit Interrupt Enable Bit

This read/write bit enables the SCTE bit to generate SCI transmitter CPU interrupt requests. Reset clears the SCTIE bit.

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

TCIE — Transmission Complete Interrupt Enable Bit

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

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

SCRIE — SCI Receive Interrupt Enable Bit

This read/write bit enables the SCRF bit to generate SCI receiver CPU interrupt requests. Reset clears the SCRIE bit.

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

ILIE — Idle Line Interrupt Enable Bit

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

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

TE — Transmitter Enable Bit

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 logic 1s from the transmit shift register to the PTE0/TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the PTE0/TxD returns to the idle condition (logic 1). Clearing and then setting TE during a transmission queues an idle character to be sent after the character currently being transmitted. Reset clears the TE bit.

1 = Transmitter enabled
0 = Transmitter disabled

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

RE — Receiver Enable Bit

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

1 = Receiver enabled
0 = Receiver disabled

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

RWU — Receiver Wakeup Bit

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

1 = Standby state
0 = Normal operation

SBK — Send Break Bit

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

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

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

15.9.3 SCI Control Register 3

SCI control register 3:

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

Address: $0015

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

= Unimplemented R = Reserved U = Unaffected

Figure 15-11. SCI Control Register 3 (SCC3)

R8 — Received Bit 8

When the SCI is receiving 9-bit characters, R8 is the read-only ninth bit (bit 8) of the received character. R8 is received at the same time that the SCDR receives the other 8 bits.
When the SCI is receiving 8-bit characters, R8 is a copy of the eighth bit (bit 7). Reset has no effect on the R8 bit.

T8 — Transmitted Bit 8

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

ORIE — Receiver Overrun Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the receiver overrun bit, OR.
1 = SCI error CPU interrupt requests from OR bit enabled
0 = SCI error CPU interrupt requests from OR bit disabled

NEIE — Receiver Noise Error Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the noise error bit, NE. Reset clears NEIE.
1 = SCI error CPU interrupt requests from NE bit enabled
0 = SCI error CPU interrupt requests from NE bit disabled

FEIE — Receiver Framing Error Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the framing error bit, FE. Reset clears FEIE.
1 = SCI error CPU interrupt requests from FE bit enabled
0 = SCI error CPU interrupt requests from FE bit disabled

PEIE — Receiver Parity Error Interrupt Enable Bit

This read/write bit enables SCI receiver CPU interrupt requests generated by the parity error bit, PE. (See **15.9.4 SCI Status Register 1**.) Reset clears PEIE.
1 = SCI error CPU interrupt requests from PE bit enabled
0 = SCI error CPU interrupt requests from PE bit disabled

15.9.4 SCI Status Register 1

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

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

Address: $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 15-12. SCI Status Register 1 (SCS1)

SCTE — SCI Transmitter Empty Bit

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

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

TC — Transmission Complete Bit

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

1 = No transmission in progress
0 = Transmission in progress

SCRF — 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 SCI receiver CPU interrupt request. When the SCRIE bit in SCC2 is set, SCRF generates a CPU interrupt request. In normal operation, clear the SCRF bit by reading SCS1 with SCRF set and then reading the SCDR. Reset clears SCRF.

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

IDLE — Receiver Idle Bit

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

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

OR — Receiver Overrun Bit

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

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

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

Software latency may allow an overrun to occur between reads of SCS1 and SCDR in the flag-clearing sequence. **Figure 15-13** shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

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

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 0 is accepted as the stop bit. FE generates an SCI error CPU interrupt request if the FEIE bit in SCC3 also is set. Clear the FE bit by reading SCS1 with FE set and then reading the SCDR. Reset clears the FE bit.

1 = Framing error detected
0 = No framing error detected

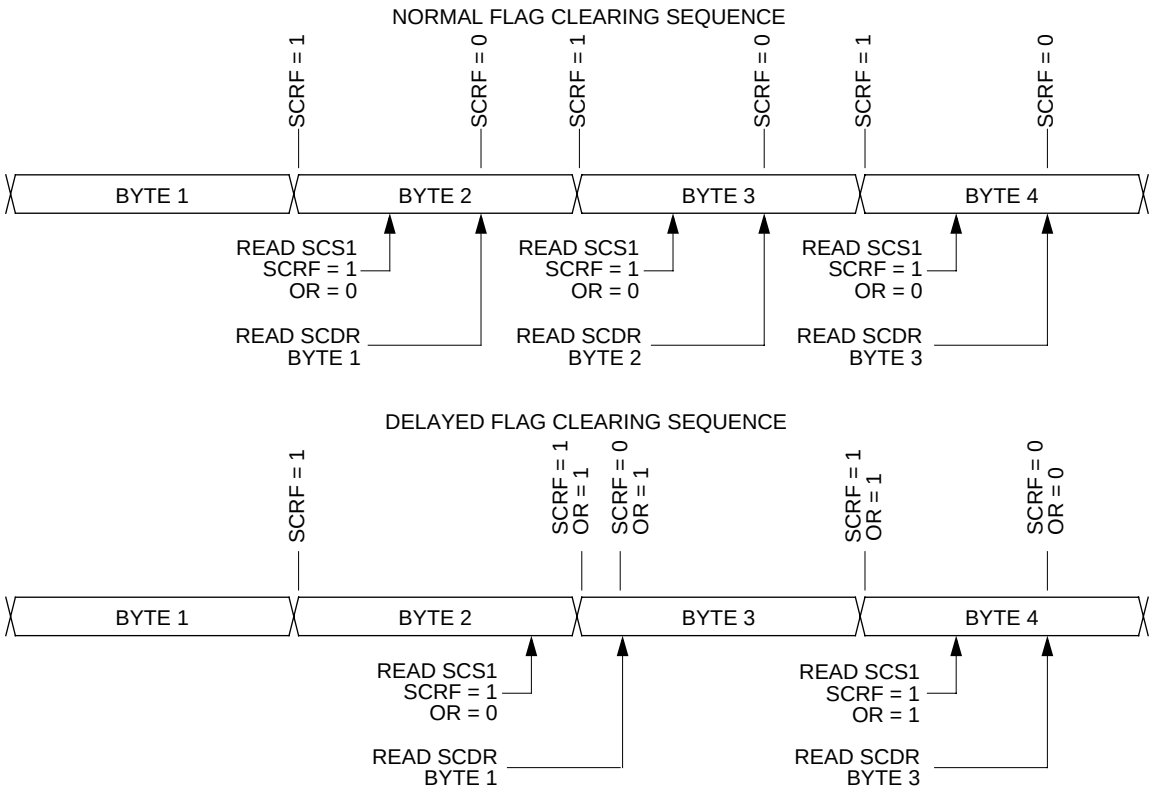


Figure 15-13. Flag Clearing Sequence

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

15.9.5 SCI Status Register 2

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

- Break character detected
- Incoming data

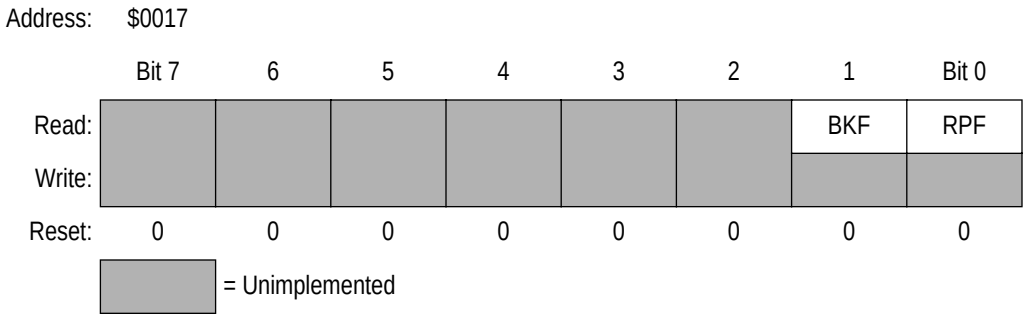


Figure 15-14. SCI Status Register 2 (SCS2)

BKF — Break Flag Bit

This clearable, read-only bit is set when the SCI detects a break character on the PTE1/RxD pin. In SCS1, the FE and SCRF bits are also set. In 9-bit character transmissions, the R8 bit in SCC3 is cleared. BKF does not generate a CPU interrupt request. Clear BKF by reading SCS2 with BKF set and then reading the SCDR. Once cleared, BKF can become set again only after logic 1s again appear on the 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 when the receiver detects a logic 0 during the RT1 time period of the start bit search. RPF does not generate an interrupt request. RPF is reset after the receiver detects false start bits (usually from noise or a baud rate mismatch) or when the receiver detects an idle character. Polling RPF before disabling the SCI module or entering stop mode can show whether a reception is in progress.

1 = Reception in progress
0 = No reception in progress

15.9.6 SCI Data Register

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

Address: $0018

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

Figure 15-15. 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.

NOTE: *Do not use read/modify/write instructions on the SCI data register.*

15.9.7 SCI Baud Rate Register

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

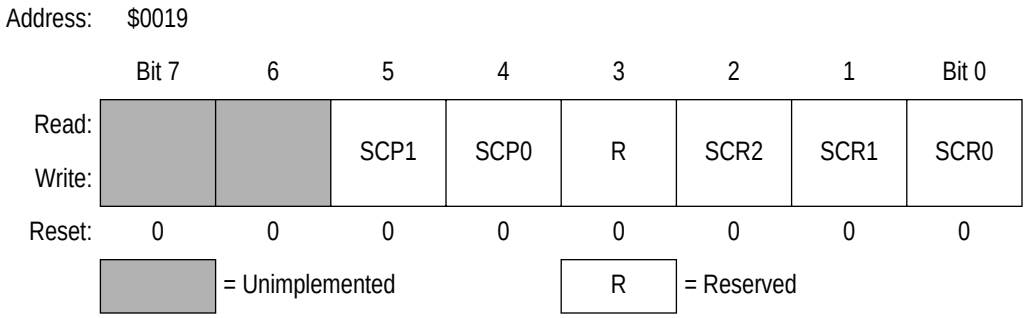


Figure 15-16. SCI Baud Rate Register (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in **Table 15-6**. Reset clears SCP1 and SCP0.

Table 15-6. SCI Baud Rate Prescaling

SCP1 and SCP0	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 15-7**. Reset clears SCR2–SCR0.

Table 15-7. SCI Baud Rate Selection

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

Use this formula to calculate the SCI baud rate:

$$\text{baud rate} = \frac{\text{SCI clock source}}{64 \times PD \times BD}$$

where:

SCI clock source = CGMXCLK
(See **9.5.6 Crystal Output Frequency Signal (CGMXCLK)**.)
PD = prescaler divisor
BD = baud rate divisor

This makes the formula:

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

Table 15-8 shows the SCI baud rates that can be generated with a 4.9152MHz CGMXCLK.

Table 15-8. SCI Baud Rate Selection Examples

SCP1 and SCP0	Prescaler Divisor (PD)	SCR2, SCR1, and SCR0	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 16. Serial Peripheral Interface Module (SPI)

16.1 Contents

16.2 Introduction

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

16.3 Features

Features of the SPI module include the following:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four master mode frequencies (maximum = bus frequency ÷ 2)
- Maximum slave mode frequency = bus frequency
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts:
  - SPRF (SPI receiver full)
  - SPTE (SPI transmitter empty)
- Mode fault error flag with CPU interrupt capability
- Overflow error flag with CPU interrupt capability
- Programmable wired-OR mode
- I^2C (inter-integrated circuit) compatibility
- I/O (input/output) port bit(s) software configurable with pullup device(s) if configured as input port bit(s)

16.4 Pin Name Conventions and I/O Register Addresses

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

The full names of the SPI I/O pins are shown in **Table 16-1**. The generic pin names appear in the text that follows.

Table 16-1. Pin Name Conventions

SPI Generic Pin Names:		MISO	MOSI	$\overline{SS}$	SPSCK	CGND
Full SPI Pin Names:	SPI	PTE5/MISO	PTE6/MOSI	PTE4/$\overline{SS}$	PTE7/SPSCK	V_{SS}

16.5 Functional Description

Figure 16-1 summarizes the SPI I/O registers and **Figure 16-2** shows the structure of the SPI module.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0010	SPI Control Register (SPCR)	Read: / Write:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Reset:	0	0	1	0	1	0	0	0
$0011	SPI Status and Control Register (SPSCR)	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	1	0	0	0
$0012	SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved

Figure 16-1. SPI I/O Register Summary

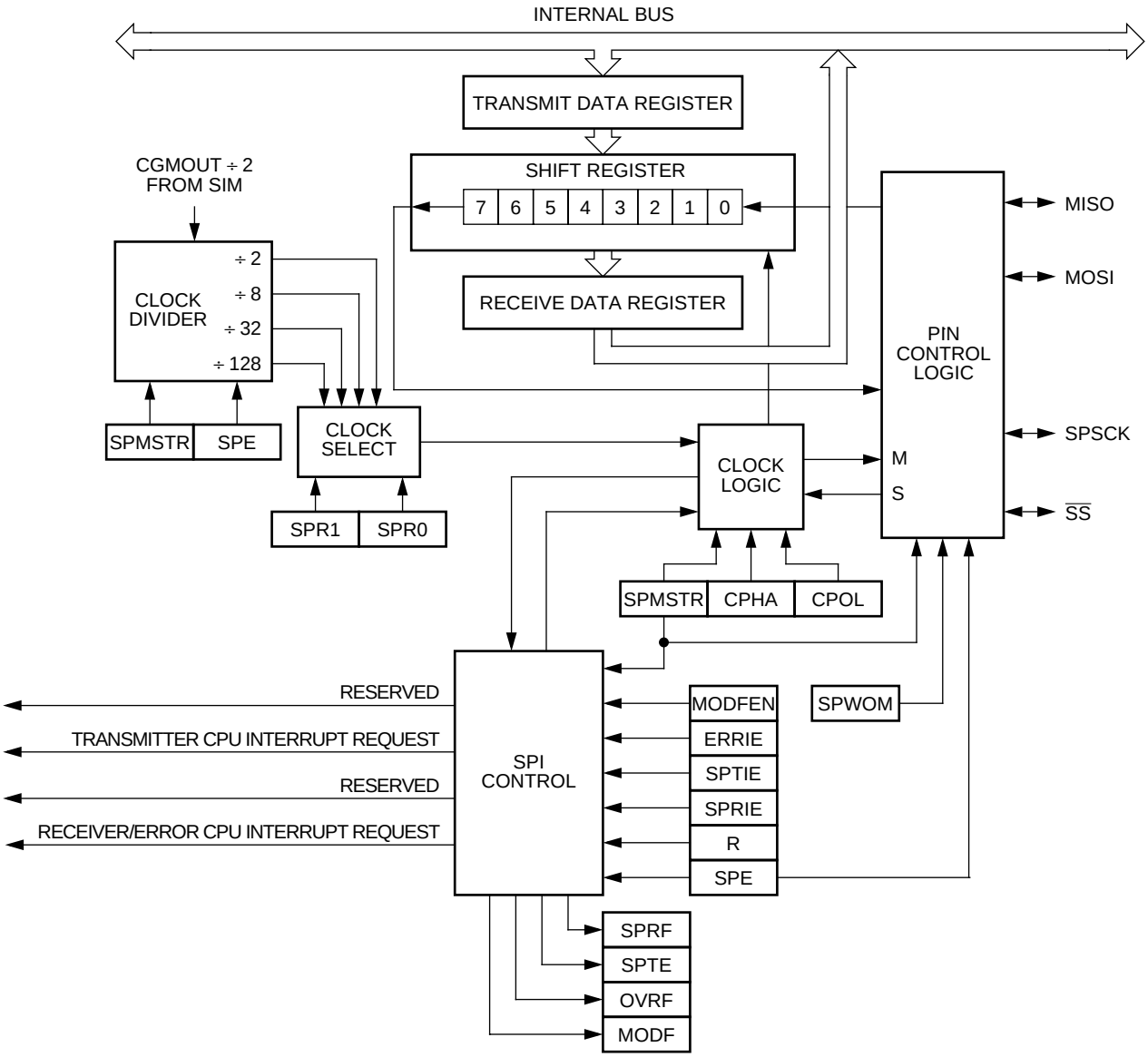


Figure 16-2. SPI Module Block Diagram

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

The following paragraphs describe the operation of the SPI module.

16.5.1 Master Mode

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

NOTE: *Configure the SPI modules as master or slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. (See **16.14.1 SPI Control Register**.)*

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the transmit data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The byte begins shifting out on the MOSI pin under the control of the serial clock. (See **Figure 16-3**.)

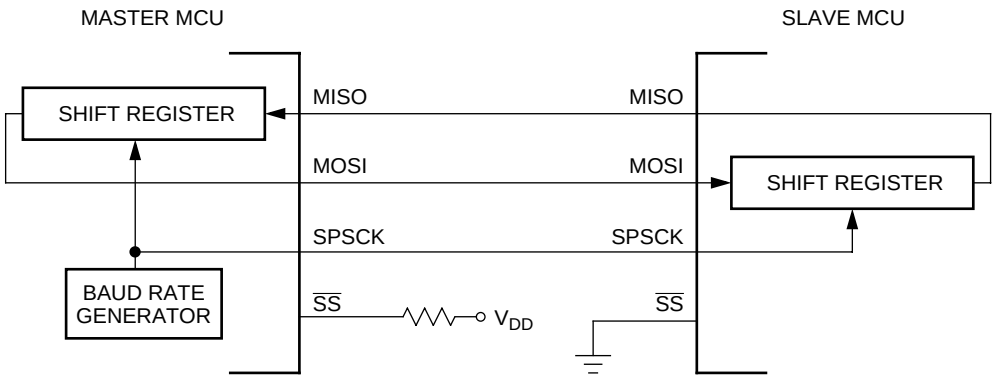


Figure 16-3. Full-Duplex Master-Slave Connections

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See **16.14.2 SPI Status and Control Register**.) Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

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

16.5.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit is clear. In slave mode, the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be at logic 0. $\overline{SS}$ must remain low until the transmission is complete. (See **16.8.2 Mode Fault Error**.)

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

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

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

When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transmission. (See **16.6 Transmission Formats**.)

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

16.6 Transmission Formats

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

16.6.1 Clock Phase and Polarity Controls

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

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

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

16.6.2 Transmission Format When CPHA = 0

Figure 16-4 shows an SPI transmission in which CPHA is logic 0. The figure should not be used as a replacement for data sheet parametric information.

Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See **16.8.2 Mode Fault Error**.) When CPHA = 0, the first SPSCK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SPSCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the slave data transmission. The slave's $\overline{SS}$ pin must be toggled back to high and then low again between each byte transmitted as shown in **Figure 16-5**.

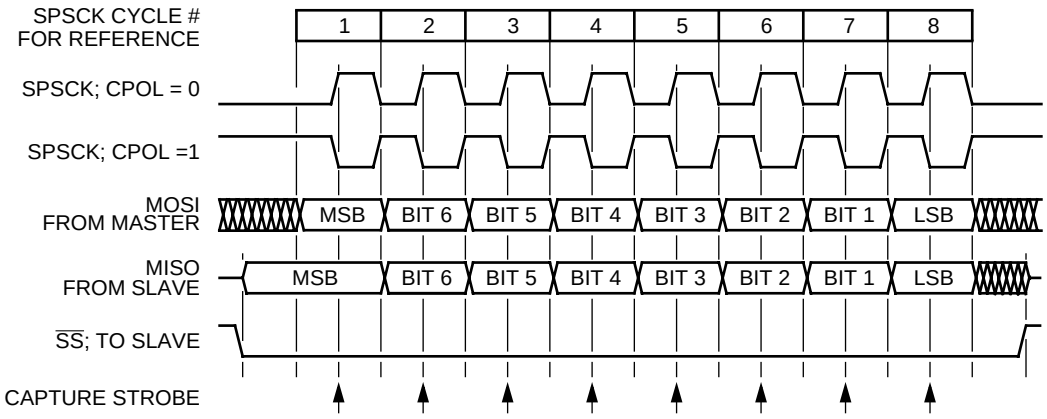


Figure 16-4. Transmission Format (CPHA = 0)

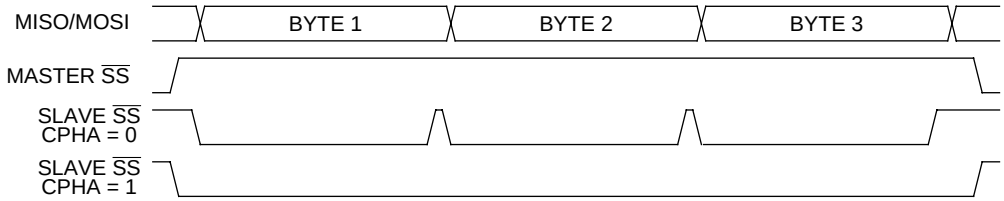


Figure 16-5. CPHA/SS Timing

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

16.6.3 Transmission Format When CPHA = 1

Figure 16-6 shows an SPI transmission in which CPHA is logic 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See **16.8.2 Mode Fault Error**.) When CPHA = 1, the master begins driving its MOSI pin on the first SPSCK edge. Therefore, the slave uses the first SPSCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

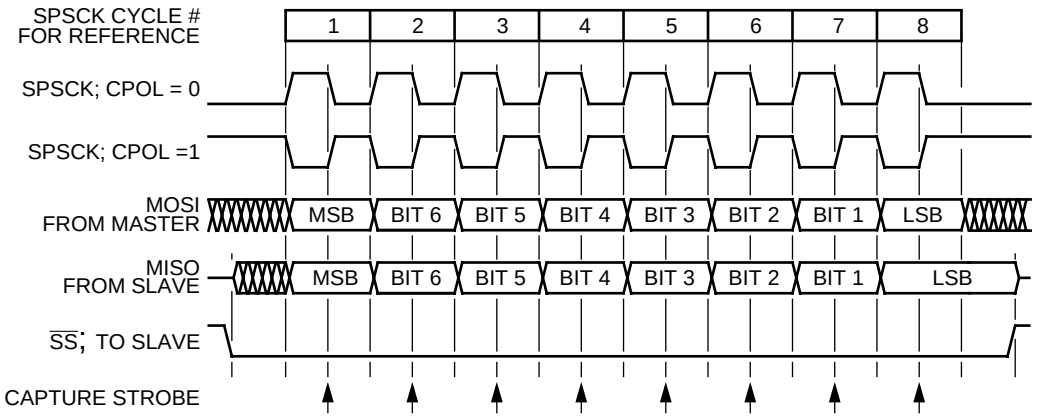


Figure 16-6. Transmission Format (CPHA = 1)

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

16.6.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), writing to the SPDR starts a transmission. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SPSCK signal. When CPHA = 0, the SPSCK signal remains inactive for the first half of the first SPSCK cycle. When CPHA = 1, the first SPSCK cycle begins with an edge on the SPSCK line from its inactive to its active level. The SPI clock rate (selected by SPR1:SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See **Figure 16-7**.) The internal SPI clock in the master is a free-running derivative of the internal MCU clock. To conserve power, it is enabled only when both the SPE and SPMSTR bits are set. SPSCK edges occur halfway through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR occurs relative to the slower SPSCK. This uncertainty causes the variation in the initiation delay shown in **Figure 16-7**. This delay is no longer than a single SPI bit time. That is, the maximum delay is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

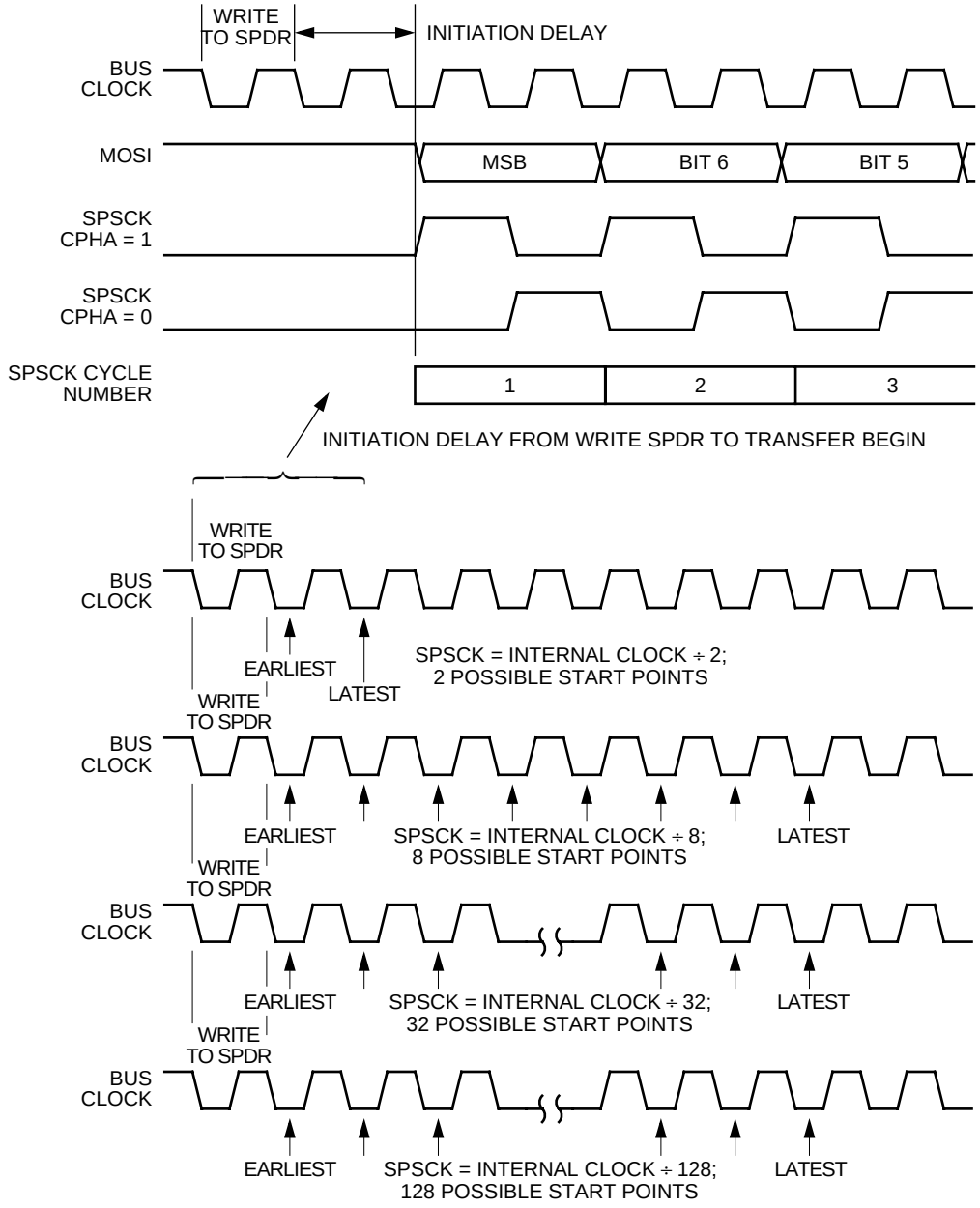


Figure 16-7. Transmission Start Delay (Master)

16.7 Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the transmit data register only when the SPTE bit is high. **Figure 16-8** shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA: CPOL = 1:0).

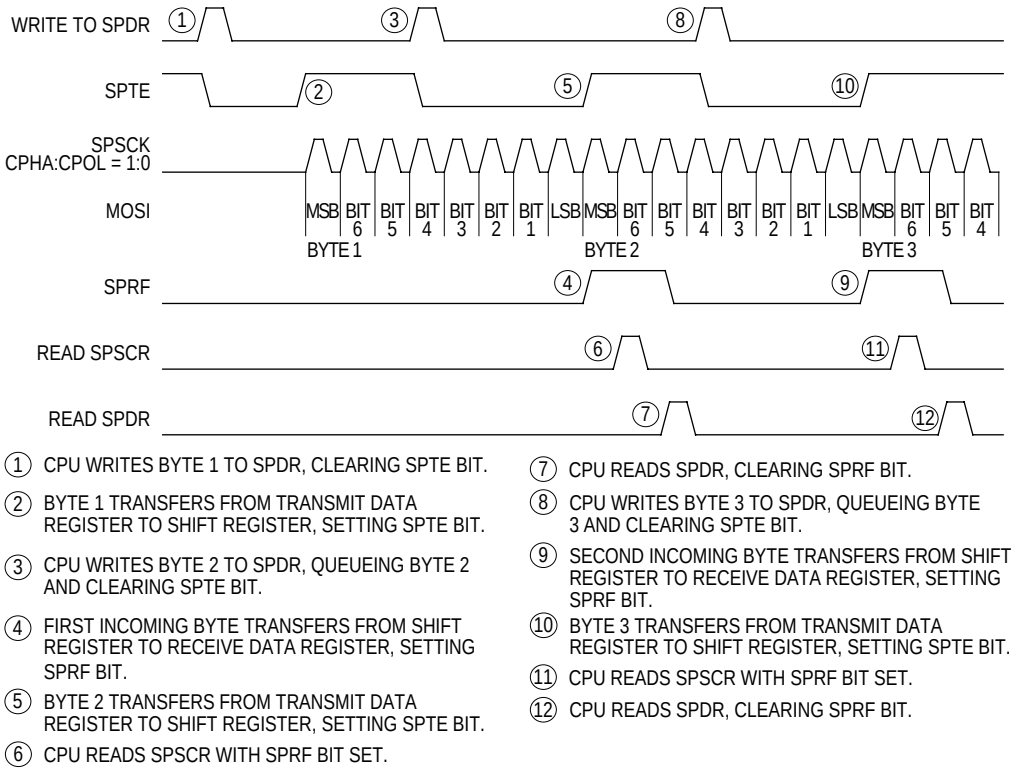


Figure 16-8. SPRF/SPTE CPU Interrupt Timing

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

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

16.8 Error Conditions

The following flags signal SPI error conditions:

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

16.8.1 Overflow Error

The overflow flag (OVRF) becomes set if the receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. The bit 1 capture strobe occurs in the middle of SPSCK cycle 7. (See **Figure 16-4** and **Figure 16-6**.) If an overflow occurs, all data received after the overflow and before the OVRF bit is cleared does not transfer to the receive data register and does not set the SPI receiver full bit (SPRF). The unread data that transferred to the receive data register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPI status and control register and then reading the SPI data register.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF

interrupts share the same CPU interrupt vector. (See **Figure 16-11**.) It is not possible to enable MODF or OVRF individually to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. **Figure 16-9** shows how it is possible to miss an overflow. The first part of **Figure 16-9** shows how it is possible to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF bit can be set in between the time that SPSCR and SPDR are read.

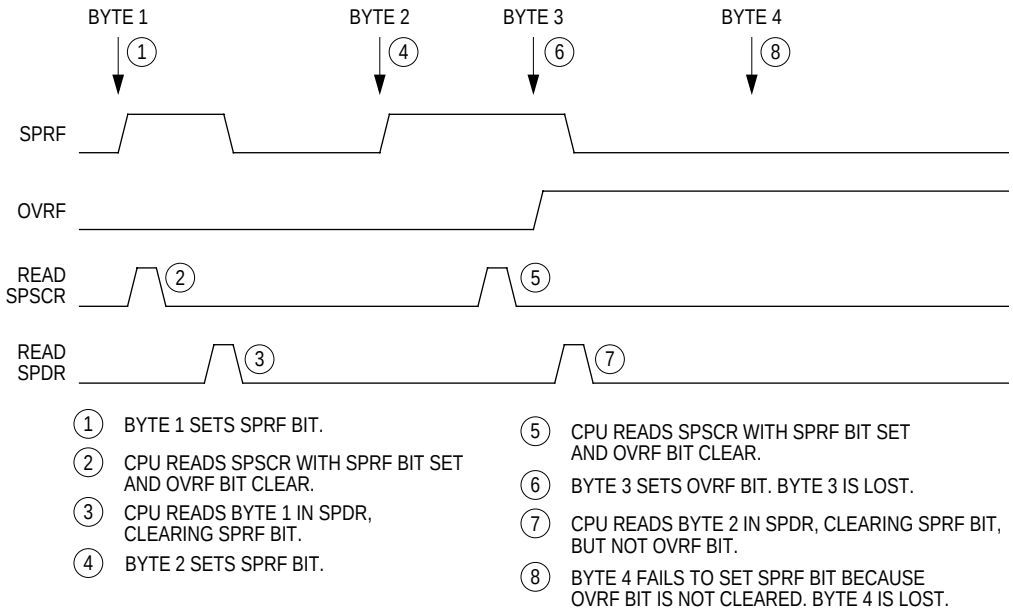


Figure 16-9. Missed Read of Overflow Condition

In this case, an overflow can be missed easily. Since no more SPRF interrupts can be generated until this OVRF is serviced, it is not obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR following the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions can set the SPRF bit. **Figure 16-10** illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit.

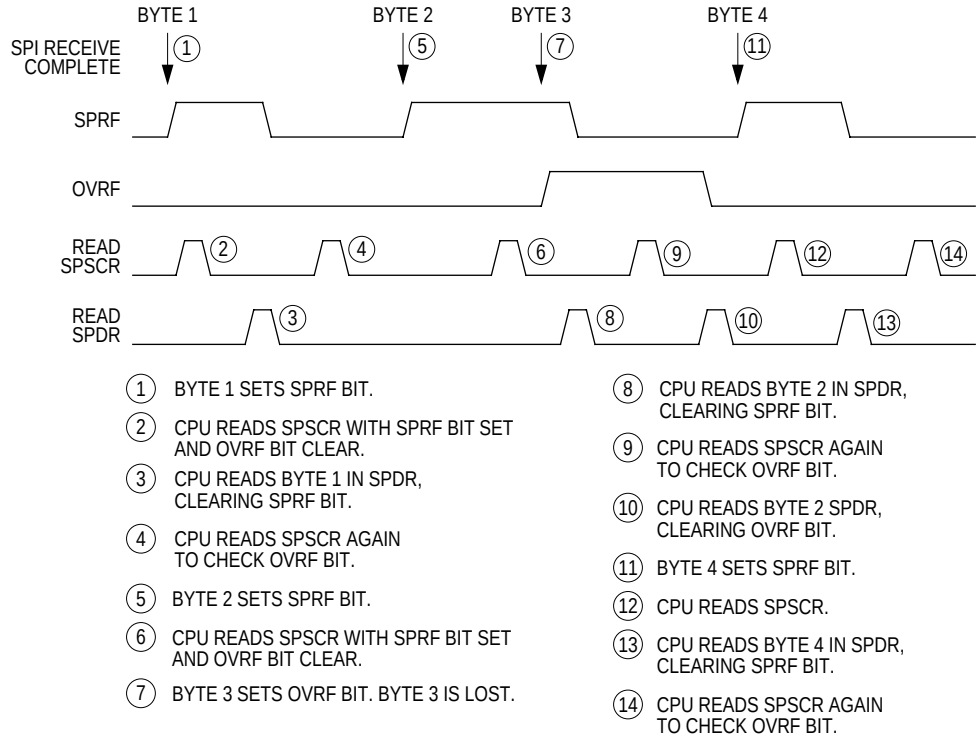


Figure 16-10. Clearing SPRF When OVRF Interrupt Is Not Enabled

16.8.2 Mode Fault Error

Setting the SPMSTR bit selects master mode and configures the SPSCK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SPSCK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, becomes set any time the state of the slave select pin, $\overline{SS}$, is inconsistent with the mode selected by SPMSTR.

To prevent SPI pin contention and damage to the MCU, a mode fault error occurs if:

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

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

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. (See **Figure 16-11**.) It is not possible to enable MODF or OVRF individually to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes to logic 0. A mode fault in a master SPI causes the following events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error CPU interrupt request.
- The SPE bit is cleared.
- The SPTE bit is set.
- The SPI state counter is cleared.
- The data direction register of the shared I/O port regains control of port drivers.

NOTE: *To prevent bus contention with another master SPI after a mode fault error, clear all SPI bits of the data direction register of the shared I/O port before enabling the SPI.*

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{SS}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{SS}$ goes low and ends once the incoming SPSCK goes back to its idle level following the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SPSCK returns to its idle level following the shift of the last data bit. (See **16.6 Transmission Formats**.)

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

When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is at logic 0) and later unselected ($\overline{SS}$ is at logic 1) even if no SPSCK is sent to that

slave. This happens because $\overline{SS}$ at logic 0 indicates the start of the transmission (MISO driven out with the value of MSB) for CPHA = 0. When CPHA = 1, a slave can be selected and then later unselected with no transmission occurring. Therefore, MODF does not occur since a transmission was never begun.

In a slave SPI (MSTR = 0), the MODF bit generates an SPI receiver/error CPU interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transmission by clearing the SPE bit of the slave.

NOTE: *A logic 1 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.*

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

16.9 Interrupts

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

Table 16-2. SPI Interrupts

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

Reading the SPI status and control register with SPRF set and then reading the receive data register clears SPRF. The clearing mechanism for the SPTE flag is always just a write to the transmit data register.

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

The SPI receiver interrupt enable bit (SPRIE) enables the SPRF bit to generate receiver CPU interrupt requests, regardless of the state of the SPE bit. (See **Figure 16-11**.)

The error interrupt enable bit (ERRIE) enables both the MODF and OVRF bits to generate a receiver/error CPU interrupt request.

The mode fault enable bit (MODFEN) can prevent the MODF flag from being set so that only the OVRF bit is enabled by the ERRIE bit to generate receiver/error CPU interrupt requests.

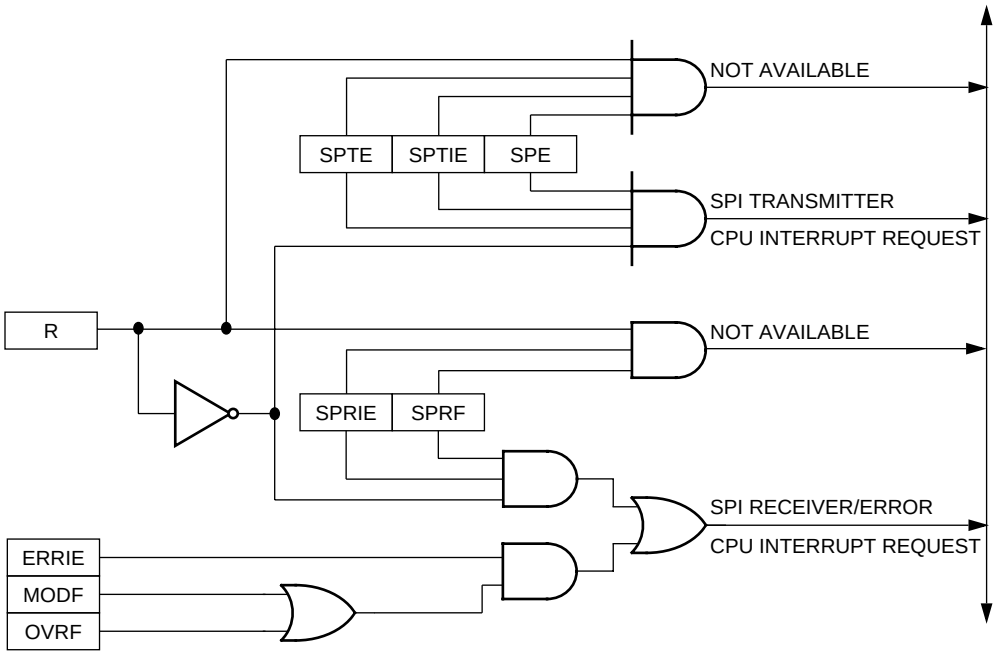


Figure 16-11. SPI Interrupt Request Generation

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

- SPI receiver full bit (SPRF) — The SPRF bit becomes set every time a byte transfers from the shift register to the receive data register. If the SPI receiver interrupt enable bit, SPRIE, is also set, SPRF generates an SPI receiver/error CPU interrupt request.
- SPI transmitter empty (SPTE) — The SPTE bit becomes set every time a byte transfers from the transmit data register to the shift register. If the SPI transmit interrupt enable bit, SPTIE, is also set, SPTE generates an SPTE CPU interrupt request.

16.10 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is low. Whenever SPE is low, the following occurs:

- The SPTE flag is set.
- Any transmission currently in progress is aborted.
- The shift register is cleared.
- The SPI state counter is cleared, making it ready for a new complete transmission.
- All the SPI port logic is defaulted back to being general-purpose I/O.

These items are reset only by a system reset:

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

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

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

16.11 Low-Power Modes

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

16.11.1 Wait Mode

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

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

To exit wait mode when an overflow condition occurs, enable the OVRF bit to generate CPU interrupt requests by setting the error interrupt enable bit (ERRIE). (See **16.9 Interrupts**.)

16.11.2 Stop Mode

The SPI module is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions. SPI operation resumes after an external interrupt. If stop mode is exited by reset, any transfer in progress is aborted, and the SPI is reset.

16.12 SPI During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **Section 8. System Integration Module (SIM)**.)

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

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

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

16.13 I/O Signals

The SPI module has five I/O pins and shares four of them with a parallel I/O port. They are:

- MISO — Data received
- MOSI — Data transmitted
- SPSCK — Serial clock
- $\overline{SS}$ — Slave select
- CGND — Clock ground (internally connected to V_{SS})

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

16.13.1 MISO (Master In/Slave Out)

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

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

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

16.13.2 MOSI (Master Out/Slave In)

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

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

16.13.3 SPSCK (Serial Clock)

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

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

16.13.4 $\overline{SS}$ (Slave Select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{SS}$ is used to select a slave. For CPHA = 0, the $\overline{SS}$ is used to define the start of a transmission. (See **16.6 Transmission Formats**.) Since it is used to indicate the start of a transmission, the $\overline{SS}$ must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low between transmissions for the CPHA = 1 format. See **Figure 16-12**.

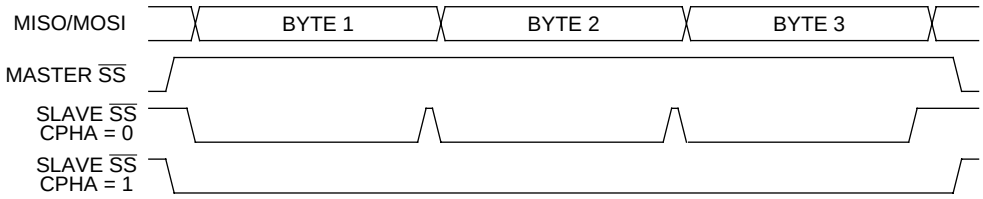


Figure 16-12. CPHA/$\overline{SS}$ Timing

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. It cannot be used as a general-purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of the $\overline{SS}$ from creating a MODF error. (See **16.14.2 SPI Status and Control Register**.)

NOTE: *A logic 1 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.*

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. (See **16.8.2 Mode Fault Error**.) For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is low for an SPI master, the $\overline{SS}$ pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{SS}$ pin by configuring the appropriate pin as an input and reading the port data register. (See **Table 16-3**.)

Table 16-3. SPI Configuration

SPE	SPMSTR	MODFEN	SPI Configuration	State of $\overline{SS}$ Logic
0	X(1)	X	Not enabled	General-purpose I/O; $\overline{SS}$ ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	General-purpose I/O; $\overline{SS}$ ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

Note 1. X = Don’t care

16.13.5 CGND (Clock Ground)

CGND is the ground return for the serial clock pin, SPSCK, and the ground for the port output buffers. It is internally connected to V_{SS} as shown in **Table 16-1**.

16.14 I/O Registers

Three registers control and monitor SPI operation:

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

16.14.1 SPI Control Register

The SPI control register:

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

Address: $0010

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

= Unimplemented R = Reserved

Figure 16-13. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

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

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

SPMSTR — SPI Master Bit

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

1 = Master mode
0 = Slave mode

CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the SPSCK pin between transmissions. (See **Figure 16-4** and **Figure 16-6**.) To transmit data between SPI modules, the SPI modules must have identical CPOL values. Reset clears the CPOL bit.

CPHA — Clock Phase Bit

This read/write bit controls the timing relationship between the serial clock and SPI data. (See **Figure 16-4** and **Figure 16-6**.) To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the $\overline{SS}$ pin of the slave SPI module must be set to logic 1 between bytes. (See **Figure 16-12**.) Reset sets the CPHA bit.

SPWOM — SPI Wired-OR Mode Bit

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

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

SPE — SPI Enable

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. (See **16.10 Resetting the SPI**.) Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE— SPI Transmit Interrupt Enable

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

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

16.14.2 SPI Status and Control Register

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

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

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

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

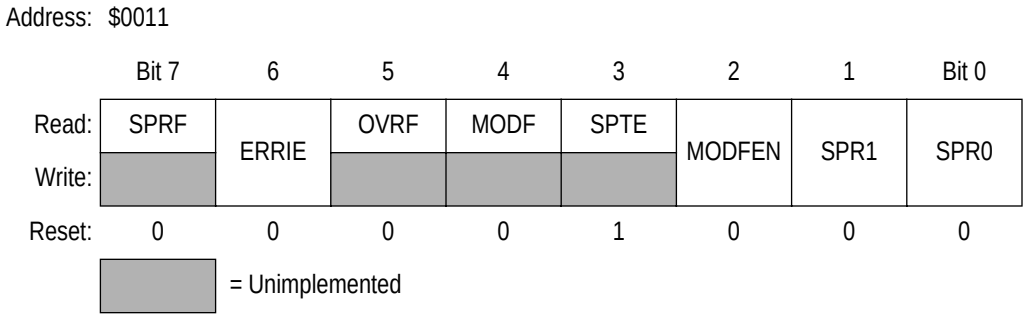


Figure 16-14. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

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

During an SPRF CPU interrupt, the CPU clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register. Reset clears the SPRF bit.

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

ERRIE — Error Interrupt Enable Bit

This read/write bit enables the MODF and OVRF bits to generate CPU interrupt requests. Reset clears the ERRIE bit.

1 = MODF and OVRF can generate CPU interrupt requests
0 = MODF and OVRF cannot generate CPU interrupt requests

OVRF — Overflow Bit

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

1 = Overflow
0 = No overflow

MODF — Mode Fault Bit

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

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

SPTE — SPI Transmitter Empty Bit

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

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

During an SPTE CPU interrupt, the CPU clears the SPTE bit by writing to the transmit data register.

Reset sets the SPTE bit.

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

MODFEN — Mode Fault Enable Bit

This read/write bit, when set to 1, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag. If the SPI is enabled as a master and the MODFEN bit is low, then the $\overline{SS}$ pin is available as a general-purpose I/O.

If the MODFEN bit is set, then this pin is not available as a general-purpose I/O. When the SPI is enabled as a slave, the $\overline{SS}$ pin is not available as a general-purpose I/O regardless of the value of MODFEN. (See **16.13.4 SS (Slave Select)**.)

If the MODFEN bit is low, the level of the $\overline{SS}$ pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation. (See **16.8.2 Mode Fault Error**.)

SPR1 and SPR0 — SPI Baud Rate Select Bits

In master mode, these read/write bits select one of four baud rates as shown in **Table 16-4**. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 16-4. SPI Master Baud Rate Selection

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

Use this formula to calculate the SPI baud rate:

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

where:

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

16.14.3 SPI Data Register

The SPI data register consists of the read-only receive data register and the write-only transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate registers that can contain different values. (See **Figure 16-2**.)

Address: $0012

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

Figure 16-15. SPI Data Register (SPDR)

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

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

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

17.1 Contents

17.2 Introduction

Fifty-one 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.*

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read: Write:	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA)	Read: Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB)	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Reset:	0	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC)	Read: Write:	MCLKEN	0	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD)	Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Reset:	0	0	0	0	0	0	0	0

Figure 17-1. I/O Port Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0008	Port E Data Register (PTE)	Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Reset:	Unaffected by reset							
$0009	Port F Data Register (PTF)	Read: Write:	PTF7	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
		Reset:	Unaffected by reset							
$000A	Port G Data Register (PTG)	Read:	0	0	0	0	0	PTG2	PTG1	PTG0
		Write:								
		Reset:	Unaffected by reset							
$000B	Port H Data Register (PTH)	Read:	0	0	0	0	0	0	PTH1	PTH0
		Write:								
		Reset:	Unaffected by reset							
$000C	Data Direction Register E (DDRE)	Read: Write:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		Reset:	0	0	0	0	0	0	0	0
$000D	Data Direction Register F (DDRF)	Read: Write:	DDRF7	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
		Reset:	0	0	0	0	0	0	0	0
$000E	Data Direction Register G (DDRG)	Read:	0	0	0	0	0	DDRG2	DDRG1	DDRG0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000F	Data Direction Register H (DDRH)	Read:	0	0	0	0	0	0	DDRH1	DDRH0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$003D	Port D Input Pullup Enable Register (PTDPUE)	Read: Write:	PTDPUE7	PTDPUE6	PTDPUE5	PTDPUE4	PTDPUE3	PTDPUE2	PTDPUE1	PTDPUE0
		Reset:	0	0	0	0	0	0	0	0
$003E	Port F Input Pullup Enable Register (PTFPUE)	Read: Write:	PTFPUE7	PTFPUE6	PTFPUE5	PTFPUE4	PTFPUE3	PTFPUE2	PTFPUE1	PTFPUE0
		Reset:	0	0	0	0	0	0	0	0

Figure 17-1. I/O Port Register Summary

Table 17-1. Port Control Register Bits Summary (Sheet 1 of 2)

Port	Bit	DDR	Module Control			Pin
			Module	Register	Control Bit	
A	0	DDRA0	—	—	—	PTA0
	1	DDRA1				PTA1
	2	DDRA2				PTA2
	3	DDRA3				PTA3
	4	DDRA4				PTA4
	5	DDRA5				PTA5
	6	DDRA6				PTA6
	7	DDRA7				PTA7
B	0	DDRB0	ADC	ADSCR $0038	ADCH[4:0]	PTB0/ATD0
	1	DDRB1				PTB1/ATD1
	2	DDRB2				PTB2/ATD2
	3	DDRB3				PTB3/ATD3
	4	DDRB4				PTB4/ATD4
	5	DDRB5				PTB5/ATD5
	6	DDRB6				PTB6/ATD6
	7	DDRB7				PTB7/ATD7
C	0	DDRC0	—	—	—	PTC0
	1	DDRC1				PTC1
	2	DDRC2	—	DDRC $0006	MCLKEN	PTC2/MCLK
	3	DDRC3	—	—	—	PTC3
	4	DDRC4				PTC4
	5	DDRC5				PTC5
D	0	DDRD0	—	—	—	PTD0
	1	DDRD1				PTD1
	2	DDRD2				PTD2
	3	DDRD3				PTD3
	4	DDRD4	TIMB	TBSC $0040	PS[2:0]	PTD4/TBCLK
	5	DDRD5	—	—	—	PTD5
	6	DDRD6	TIMA	TASC $0020	PS[2:0]	PTD6/TACLK
	7	DDRD7	—	—	—	PTD7

Table 17-1. Port Control Register Bits Summary (Sheet 2 of 2)

Port	Bit	DDR	Module Control			Pin
			Module	Register	Control Bit	
E	0	DDRE0	SCI	SCC1 $0013	ENSCI	PTE0/TxD
	1	DDRE1				PTE1/RxD
	2	DDRE2	TIMA	TASC0 $0026	ELS0B:ELS0A	PTE2/TACH0
	3	DDRE3		TASC1 $0029	ELS1B:ELS1A	PTE3/TACH1
	4	DDRE4	SPI	SPCR $0010	SPE, SPMSTR	PTE4/$\overline{SS}$
	5	DDRE5			SPE	PTE5/MISO
	6	DDRE6				PTE6/MOSI
	7	DDRE7				PTE7SPSCK
F	0	DDRF0	TIMA	TASC2 $002C	ELS2B:ELS2A	PTF0/TACH2
	1	DDRF1		TASC3 $002F	ELS3B:ELS3A	PTF1/TACH3
	2	DDRF2	TIMB	TBSC2 $0032	ELS2B:ELS2A	PTF2/TBCH2
	3	DDRF3		TBSC3 $0035	ELS3B:ELS3A	PTF3/TBCH3
	4	DDRF4		TBSC0 $0045	ELS0B:ELS0A	PTF4/TBCH0
	5	DDRF5		TBSC1 $0048	ELS1B:ELS1A	PTF5/TBCH1
	6	DDRF6	—	—	—	PTF6
	7	DDRF7				PTF7
G	0	DDRG0	KBI	KBIER $0021	KBIE0	PTG0/KBD0
	1	DDRG1			KBIE1	PTG1/KBD1
	2	DDRG2			KBIE2	PTG2/KBD2
H	0	DDRH0			KBIE3	PTH0/KBD3
	1	DDRH1			KBIE4	PTH1/KBD4

17.3 Port A

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

17.3.1 Port A Data Register (PTA)

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

Address: $0000

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

Figure 17-2. Port A Data Register (PTA)

PTA[7:0] — Port A Data Bits

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

17.3.2 Data Direction Register A (DDRA)

Data direction register A determines whether each port A pin is an input or an output. Writing a logic 1 to a DDRA bit enables the output buffer for the corresponding port A pin; a logic 0 disables the output buffer.

Address: $0004

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

Figure 17-3. Data Direction Register A (DDRA)

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 17-4*** *shows the port A I/O logic.*

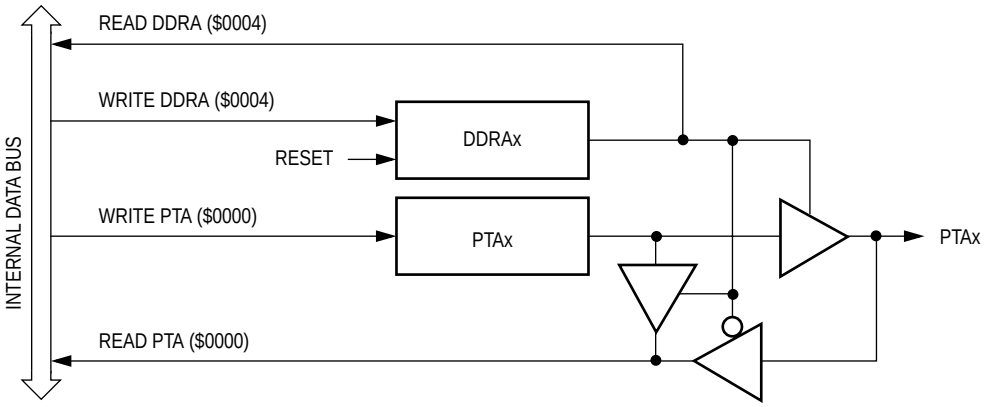


Figure 17-4. Port A I/O Circuit

When DDRAx is a logic 1, reading address $0000 reads the PTAx data latch. When DDRAx is a logic 0, reading address $0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit.

Table 17-2 summarizes the operation of the port A pins.

Table 17-2. Port A Pin Functions

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

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

17.4 Port B

Port B is an 8-bit special function port that shares all eight of its port pins with the analog-to-digital converter (ADC) module (see **Section 14. Analog-to-Digital Converter (ADC)**).

17.4.1 Port B Data Register (PTB)

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

Address: $0001

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
Reset:	Unaffected by reset							
Alternative Function:	ATD7	ATD6	ATD5	ATD4	ATD3	ATD2	ATD2	ATD0

Figure 17-5. Port B Data Register (PTB)

PTB[7:0] — Port B Data Bits

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

ATD[7:0] — ADC channels

ATD[7:0] are pins used for the input channels to the analog-to-digital converter module. The channel select bits in the ADC status and control register define which port B pin will be used as an ADC input and overrides any control from the port I/O logic by forcing that pin as the input to the analog circuitry.

NOTE: *Care must be taken when reading port B while applying analog voltages to ATD[7:0] pins. If the appropriate ADC channel is not enabled, excessive current drain may occur if analog voltages are applied to the PTBx/ATDx pin, while PTB is read as a digital input. Those ports not selected as analog input channels are considered digital I/O ports.*

17.4.2 Data Direction Register B (DDRB)

Data direction register B determines whether each port B pin is an input or an output. Writing a logic 1 to a DDRB bit enables the output buffer for the corresponding port B pin; a logic 0 disables the output buffer.

Address: $0005

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

Figure 17-6. Data Direction Register B (DDRB)

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 17-7*** *shows the port B I/O logic.*

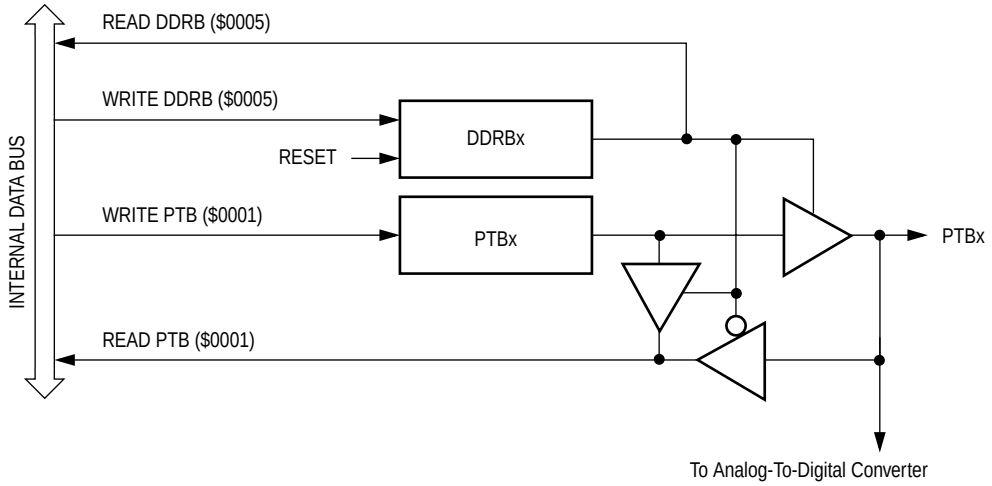


Figure 17-7. Port B I/O Circuit

When DDRBx is a logic 1, reading address $0001 reads the PTBx data latch. When DDRBx is a logic 0, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit.

Table 17-3 summarizes the operation of the port B pins.

Table 17-3. Port B Pin Functions

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

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

17.5 Port C

Port C is a 6-bit general-purpose bidirectional I/O port. PTC2 pin can be configured as an output pin for the system MCLK clock.

17.5.1 Port C Data Register (PTC)

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

Address: $0002

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
Write:								
Reset:	Unaffected by reset							
Alternative Function:						MCLK		

Figure 17-8. Port C Data Register (PTC)

PTC[5:0] — Port C Data Bits

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

MCLK — T12 System Clock

The system clock is driven out of the PTC2 pin when MCLKEN is set.

17.5.2 Data Direction Register C (DDRC)

Data direction register C determines whether each port C pin is an input or an output. Writing a logic 1 to a DDRC bit enables the output buffer for the corresponding port C pin; a logic 0 disables the output buffer.

Address: $0006

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	MCLKEN	0	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 17-9. Data Direction Register B (DDRB)

MCLKEN — T12 System Clock Enable Bit

This read/write bit enables MCLK to be an output signal on PTC2 pin. Reset clears MCLKEN.

1 = PTC2 pin configured as MCLK output
0 = PTC2 pin configured as standard I/O pin

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

These read/write bits control port C data direction. Reset clears DDRC[5:0], 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 17-10*** *shows the port C I/O logic.*

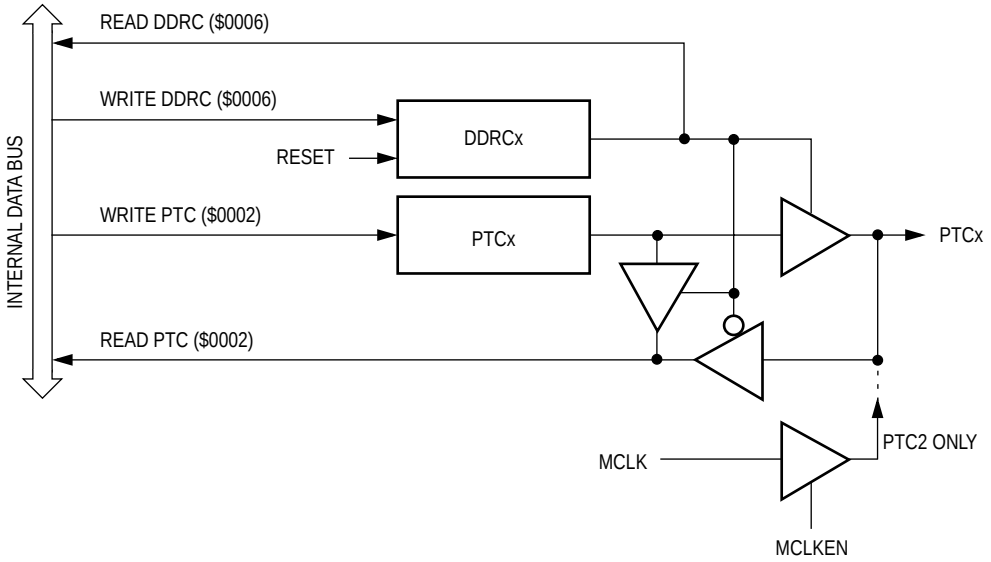


Figure 17-10. Port C I/O Circuit

When DDRCx is a logic 1, reading address $0002 reads the PTCx data latch. When DDRCx is a logic 0, reading address $0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit.

Table 17-4 summarizes the operation of the port C pins.

Table 17-4. Port C Pin Functions

DDRC Bit	PTC Bit	I/O Pin Mode	Accesses to DDRC	Accesses to PTC	
			Read/Write	Read	Write
0	2	Input, Hi-Z	DDRC7	Pin	PTC2
1	2	Output	DDRC7	0	—
0	X(1)	Input, Hi-Z(2)	DDRC[5:0]	Pin	PTC[5:0](3)
1	X	Output	DDRC[5:0]	PTC[5:0]	PTC[5:0]

Notes:
1. X = don't care; except PTC2.
2. Hi-Z = high impedance.
3. Writing affects data register, but does not affect input.

17.6 Port D

Port D is an 8-bit special function port that shares two of its pins with the timer interface module (see **Section 11. Timer Interface Module A (TIMA)** and **Section 12. Timer Interface Module B (TIMB)**). Each port D pin has 15mA current drive (sink) and programmable pullup.

17.6.1 Port D Data Register (PTD)

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

Address: $0003

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
Reset:	Unaffected by reset							
Alternative Function:		TACLK		TBCLK				
Additional Functions:	15mA sink	15mA sink	15mA sink	15mA sink	15mA sink	15mA sink	15mA sink	15mA sink
	Input pullup	Input pullup	Input pullup	Input pullup	Input pullup	Input pullup	Input pullup	Input pullup

Figure 17-11. Port D Data Register (PTD)

PTD[7:0] — Port D Data Bits

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

TACLK — Timer A Clock Input

The PTD6 pin becomes TACLK, the timer A (TIMA) external clock input when the TIMA prescaler select bits, PS[2:0] = 111. See **Section 11. Timer Interface Module A (TIMA)**.

TBCLK — Timer B Clock Input

The PTD4 pin becomes TBCLK, the timer B (TIMB) external clock input when the TIMB prescaler select bits, PS[2:0] = 111. See **Section 12. Timer Interface Module B (TIMB)**.

17.6.2 Data Direction Register D (DDRD)

Data direction register D determines whether each port D pin is an input or an output. Writing a logic 1 to a DDRD bit enables the output buffer for the corresponding port D pin; a logic 0 disables the output buffer.

Address: $0007

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

Figure 17-12. Data Direction Register D (DDRD)

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

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

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

NOTE: *Avoid glitches on port D pins by writing to the port D data register before changing data direction register D bits from 0 to 1.* ***Figure 17-13*** *shows the port D I/O logic.*

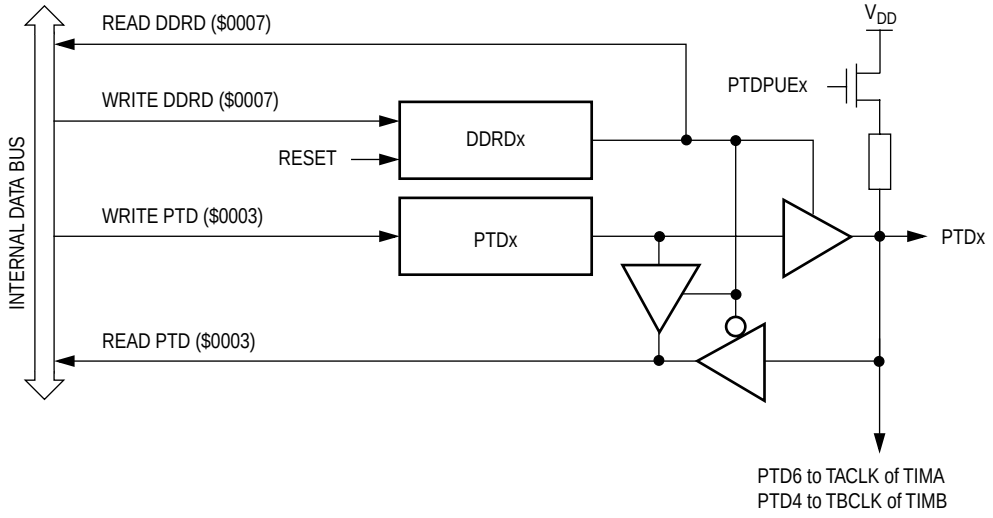


Figure 17-13. Port D I/O Circuit

When DDRDx is a logic 1, reading address $0003 reads the PTDx data latch. When DDRDx is a logic 0, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit.

Table 17-5 summarizes the operation of the port D pins.

Table 17-5. Port D Pin Functions

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

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

17.6.3 Port D Input Pullup Enable Register (PTDPUE)

The port D input pullup enable register (PTDPUE) controls the input pullup device for each of the eight port D pins. Each bit is individually configurable and requires that the data direction register, DDRD, bit be configured as an input. Each pullup is automatically and dynamically disabled when a port bit's DDRD is configured for output mode.

Address: $003D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTDPUE7	PTDPUE6	PTDPUE5	PTDPUE4	PTDPUE3	PTDPUE2	PTDPUE1	PTDPUE0
Reset:	0	0	0	0	0	0	0	0

Figure 17-14. Port D Input Pullup Enable Register (PTDPUE)

PTDPUE[7:0] — Port D Input Pullup Enable Bits

These writable bits are software programmable to enable pullup devices on an input port pin.

1 = Corresponding port D pin configured to have internal pullup
0 = Corresponding port D pin internal pullup disconnected

17.7 Port E

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

17.7.1 Port E Data Register (PTE)

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

Address: $0008

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
Reset:	Unaffected by reset							
Alternative Function:	SPSCK	MOSI	MISO	$\overline{SS}$	TACH1	TACH0	RxD	TxD

Figure 17-15. Port E Data Register (PTE)

PTE[7:0] — Port E Data Bits

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

SPSCK — SPI Serial Clock

The PTE7/SPSCK pin is the serial clock input of a SPI slave module and serial clock output of a SPI master modules. When the SPE bit is clear, the PTE7/SPSCK pin is available for general-purpose I/O. See **16.14.1 SPI Control Register**.

MOSI — Master Out/Slave In

The PTE6/MOSI pin is the master out/slave in terminal of the SPI module. When the SPE bit is clear, the PTE6/MOSI pin is available for general-purpose I /O. See **16.14.1 SPI Control Register**.

MISO — Master In/Slave Out

The PTE5/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 PTE5/MISO pin is available for general-purpose I/O. See **16.14.1 SPI Control Register**.

$\overline{SS}$ — Slave Select

The PTE4/$\overline{SS}$ 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 PTE4/$\overline{SS}$ pin is available for general-purpose I/O. See **16.14.1 SPI Control Register**. When the SPI is enabled as a slave, the DDRE4 bit in data direction register E (DDRE) has no effect on the PTE4/$\overline{SS}$ pin.

TACH[1:0] — Timer A Channel I/O Bits

The PTE3/TACH1–PTE2/TACH0 pins are the TIMA input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTE3/TACH1–PTE2/TACH0 pins are timer channel I/O pins or general-purpose I/O pins. See **11.10.4 TIMA Channel Status and Control Registers**.

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 **15.9.1 SCI Control Register 1**.

TxD — SCI Transmit Data Output

The PTE0/TxD pin is the transmit data output for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE0/TxD pin is available for general-purpose I/O. See **15.9.1 SCI Control Register 1**.

NOTE: *Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SPI module, TIMA, and 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 17-6**.*

17.7.2 Data Direction Register E (DDRE)

Data direction register E determines whether each port E pin is an input or an output. Writing a logic 1 to a DDRE bit enables the output buffer for the corresponding port E pin; a logic 0 disables the output buffer.

Address: $000C

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

Figure 17-16. 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 17-17*** *shows the port E I/O logic.*

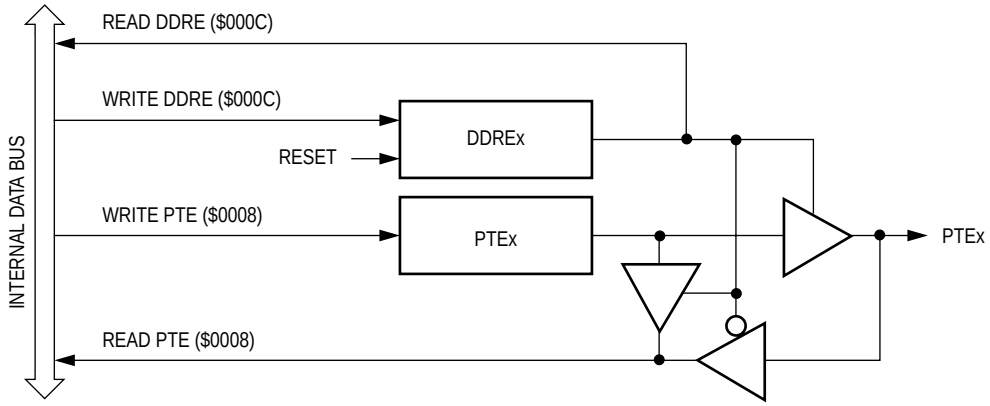


Figure 17-17. Port E I/O Circuit

When DDREx is a logic 1, reading address $0008 reads the PTEx data latch. When DDREx is a logic 0, reading address $0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit.

Table 17-6 summarizes the operation of the port E pins.

Table 17-6. 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]

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

17.8 Port F

Port F is an 8-bit special function port that shares six of its pins with the timer interface modules (TIMA and TIMB).

17.8.1 Port F Data Register (PTF)

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

Address: $0009

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTF7	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
Reset:	Unaffected by reset							
Alternative Function:			TBCH1	TBCH0	TBCH3	TBCH2	TACH3	TACH2
Additional Function:	Input pullup	Input pullup	Input pullup	Input pullup	Input pullup	Input pullup	Input pullup	Input pullup

Figure 17-18. Port F Data Register (PTF)

PTF[7:0] — 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 port F data.

TACH[3:2] and TBCH[3:0] — Timer channel I/O bits

The PTF5/TBCH1–PTF0/TACH2 pins are the TIMA and TIMB input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF5/TBCH1–PTF0/TACH2 pins are timer channel I/O pins or general-purpose I/O pins. See **11.10.4 TIMA Channel Status and Control Registers** and **12.10.4 TIMB Channel Status and Control Registers**.

NOTE: *Data direction register F (DDRF) does not affect the data direction of port F pins that are being used by TIMA and TIMB. However, the DDRF bits always determine whether reading port F returns the states of the latches or the states of the pins. See **Table 17-7**.*

17.8.2 Data Direction Register F (DDRF)

Data direction register F determines whether each port F pin is an input or an output. Writing a logic 1 to a DDRF bit enables the output buffer for the corresponding port F pin; a logic 0 disables the output buffer.

Address: $000D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRF7	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
Reset:	0	0	0	0	0	0	0	0

Figure 17-19. Data Direction Register F (DDRF)

DDRF[7:0] — Data Direction Register F Bits

These read/write bits control port F data direction. Reset clears DDRF[7:0], 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 17-20*** *shows the port F I/O logic.*

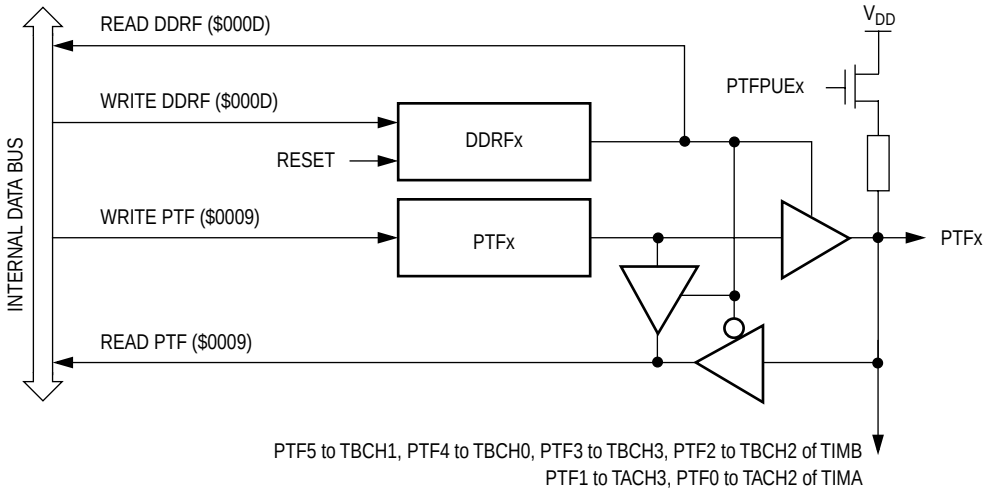


Figure 17-20. Port F I/O Circuit

When DDRFx is a logic 1, reading address $0009 reads the PTFx data latch. When DDRFx is a logic 0, 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 17-7 summarizes the operation of the port F pins.

Table 17-7. 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[7:0]	Pin	PTF[7:0](3)
1	X	Output	DDRF[7:0]	PTF[7:0]	PTF[7:0]

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

17.8.3 Port F Input Pullup Enable Register (PTFPUE)

The port F input pullup enable register (PTFPUE) controls the input pullup device for each of the eight port F pins. Each bit is individually configurable and requires that the data direction register, DDRF, bit be configured as an input. Each pullup is automatically and dynamically disabled when a port bit's DDRF is configured for output mode.

Address: $003E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTFPUE7	PTFPUE6	PTFPUE5	PTFPUE4	PTFPUE3	PTFPUE2	PTFPUE1	PTFPUE0
Reset:	0	0	0	0	0	0	0	0

Figure 17-21. Port F Input Pullup Enable Register (PTFPUE)

PTFPUE[7:0] — Port F Input Pullup Enable Bits

These writable bits are software programmable to enable pullup devices on an input port pin.

1 = Corresponding port F pin configured to have internal pullup
0 = Corresponding port F pin internal pullup disconnected

17.9 Port G

Port G is a 3-bit special-function port that shares all three of its pins with the keyboard interrupt (KBI) module.

17.9.1 Port G Data Register (PTG)

The port G data register (PTG) contains a data latch for each of the three port G pins.

Address: $000A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	PTG2	PTG1	PTG0
Write:								
Reset:	Unaffected by reset							
Alternative Function:						KBD2	KBD1	KBD0

Figure 17-22. Port G Data Register (PTG)

PTG[2:0] — Port G Data Bits

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

KBD[2:0] — The keyboard interrupt enable bits, KBIE[2:0], in the keyboard interrupt enable register (KBIER), enable the port G pins as external interrupt pins. See **Section 19. Keyboard Interrupt Module (KBI)**.

17.9.2 Data Direction Register G (DDRG)

Data direction register G determines whether each port G pin is an input or an output. Writing logic 1 to a DDRG bit enables the output buffer for the corresponding port G pin; a logic 0 disables the output buffer.

Address: $000E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	DDRG2	DDRG1	DDRG0
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 17-23. Data Direction Register G (DDRG)

DDRG[2:0] — Data Direction Register G Bits

These read/write bits control port G data direction. Reset clears DDRG[2:0], 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 17-24*** *shows the port G I/O logic.*

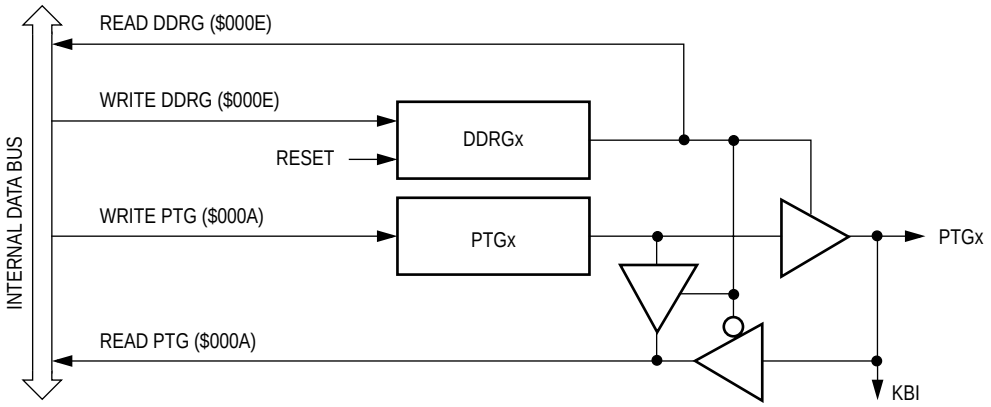


Figure 17-24. Port G I/O Circuit

When DDRGx is a logic 1, reading address $000A reads the PTGx data latch. When DDRGx is a logic 0, 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 17-6 summarizes the operation of the port G pins.

Table 17-8. 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[2:0]	Pin	PTG[2:0](3)
1	X	Output	DDRG[2:0]	PTG[2:0]	PTG[2:0]

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

17.10 Port H

Port H is a 2-bit special-function port that shares all two of its pins with the keyboard interrupt (KBI) module.

17.10.1 Port H Data Register (PTH)

The port H data register (PTH) contains a data latch for each of the two port H pins.

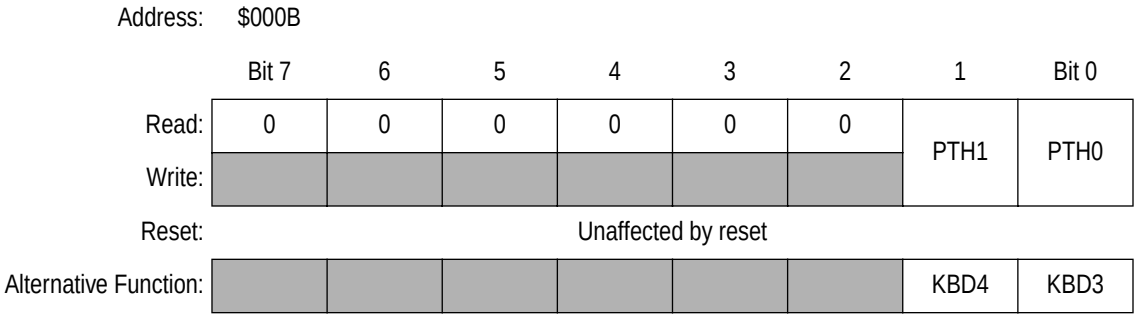


Figure 17-25. Port H Data Register (PTH)

PTH[1:0] — Port H Data Bits

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

KBD[4:3] — The keyboard interrupt enable bits, KBIE[4:3], in the keyboard interrupt enable register (KBIER), enable the port H pins as external interrupt pins. See **Section 19. Keyboard Interrupt Module (KBI)**.

17.10.2 Data Direction Register H (DDRH)

Data direction register H determines whether each port H pin is an input or an output. Writing logic 1 to a DDRH bit enables the output buffer for the corresponding port H pin; a logic 0 disables the output buffer.

Address: $000F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	DDRH1	DDRH0
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 17-26. Data Direction Register H (DDRH)

DDRH[1:0] — Data Direction Register H Bits

These read/write bits control port H data direction. Reset clears DDRH[1:0], 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 data direction register H bits from 0 to 1.* ***Figure 17-27*** *shows the port H I/O logic.*

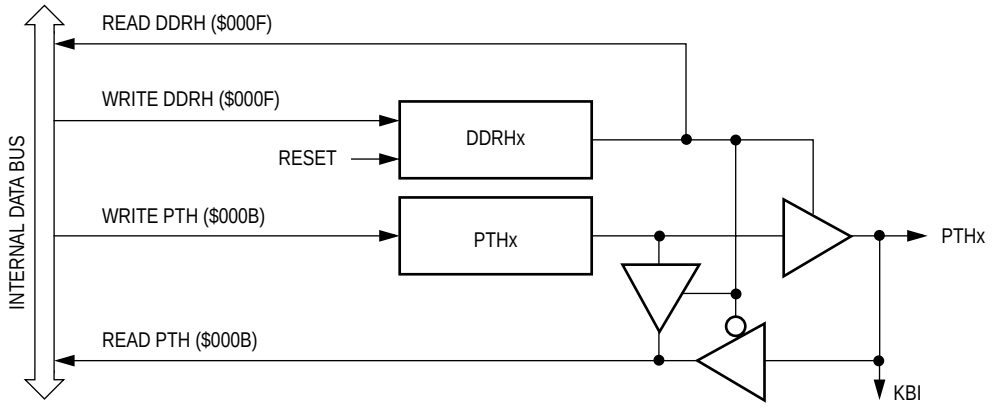


Figure 17-27. Port H I/O Circuit

When DDRHx is a logic 1, reading address $000B reads the PTHx data latch. When DDRHx is a logic 0, 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 17-6 summarizes the operation of the port H pins.

Table 17-9. 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[1:0]	Pin	PTH[1:0](3)
1	X	Output	DDRH[1:0]	PTH[1:0]	PTH[1:0]

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

Section 18. External Interrupt (IRQ)

18.1 Contents

18.2 Introduction

The IRQ (external interrupt) module provides a maskable interrupt input.

18.3 Features

Features of the IRQ module include the following:

- A dedicated external interrupt pin, $\overline{\text{IRQ}}$
- IRQ interrupt control bits
- Hysteresis buffer
- Programmable edge-only or edge and level interrupt sensitivity
- Automatic interrupt acknowledge
- Internal pullup resistor

18.4 Functional Description

A logic 0 applied to the external interrupt pin can latch a CPU interrupt request. **Figure 18-1** shows the structure of the IRQ module.

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

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

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

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

When the interrupt pin is both falling-edge and low-level-triggered, the CPU interrupt request remains set until both of the following occur:

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

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

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

NOTE: *The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. (See **8.6 Exception Control**.)*

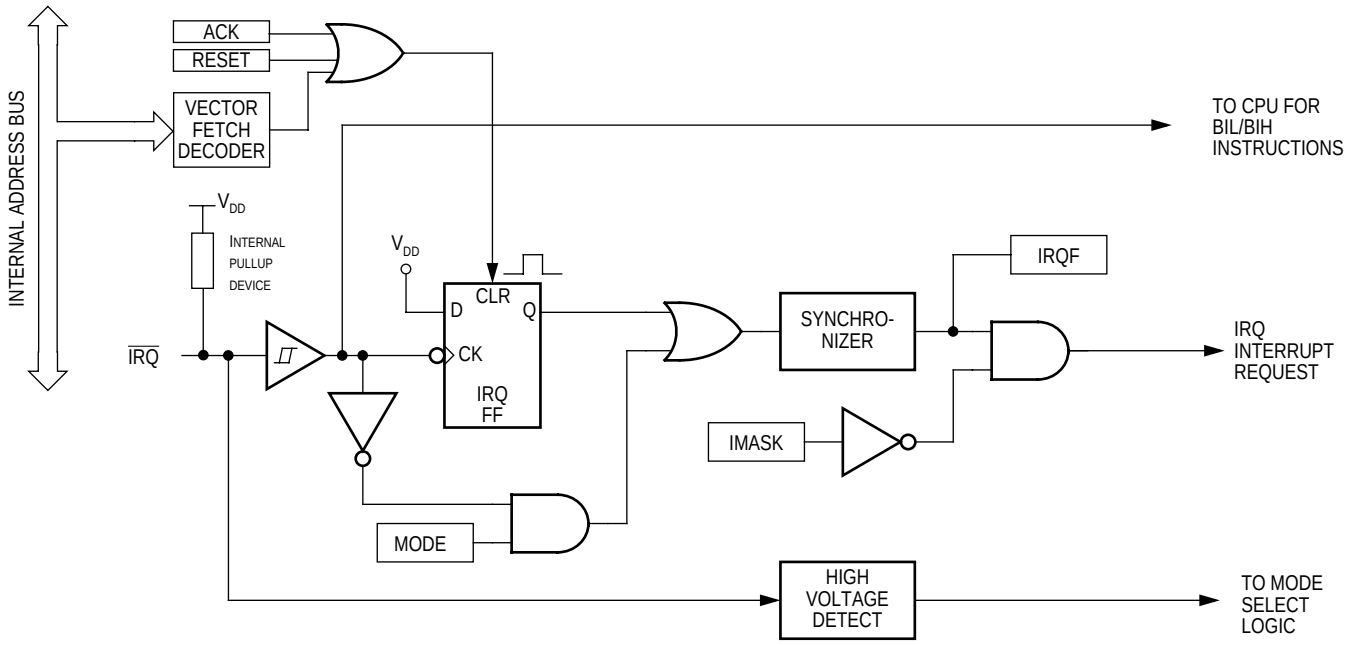


Figure 18-1. IRQ Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001A	IRQ Status and Control Register (ISCR)	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 18-2. IRQ I/O Register Summary

18.4.1 $\overline{\text{IRQ}}$ Pin

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

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

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

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

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

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

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

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

18.5 IRQ Status and Control Register (ISCR)

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

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

Address: $001A

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

= Unimplemented

Figure 18-3. IRQ Status and Control Register (ISCR)

IRQF — IRQ Flag

This read-only status bit is high when the IRQ interrupt is pending.
1 = IRQ interrupt pending
0 = IRQ interrupt not pending

ACK — IRQ Interrupt Request Acknowledge Bit

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

IMASK — IRQ Interrupt Mask Bit

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

MODE — IRQ Edge/Level Select Bit

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

18.6 IRQ Module During Break Interrupts

The system integration module (SIM) controls whether the IRQ latch can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear the latches during the break state. (See **Section 8. System Integration Module (SIM)**.)

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

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

Section 19. Keyboard Interrupt Module (KBI)

19.1 Contents

19.2 Introduction

The keyboard interrupt module (KBI) provides five independently maskable external interrupts which are accessible via PTG0–PTG2 and PTH0–PTH1 pins.

19.3 Features

Features of the keyboard interrupt module include the following:

- Five keyboard interrupt pins with pullup devices
- Separate keyboard interrupt enable bits and one keyboard interrupt mask
- Built-in pull-up device if input pin is configured as input port bit
- Programmable edge-only or edge- and level- interrupt sensitivity
- Exit from low-power modes

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001B	Keyboard Status and Control Register (KBSCR)	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
		Write:						ACKK		
		Reset:	0	0	0	0	0	0	0	0
$0021	Keyboard Interrupt Enable Register (KBIER)	Read:	0	0	0	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 19-1. KBI I/O Register Summary

19.4 I/O Pins

The five keyboard interrupt pins are shared with standard port I/O pins. The full name of the KBI pins are listed in **Table 19-1**. The generic pin name appear in the text that follows.

Table 19-1. Pin Name Conventions

KBI Generic Pin Name	Full MCU Pin Name	Pin Selected for KBI Function by KBIEx Bit in KBIER
KBD0	PTG0/KBD0	KBIE0
KBD1	PTG1/KBD1	KBIE1
KBD2	PTG2/KBD2	KBIE2
KBD3	PTH0/KBD3	KBIE3
KBD4	PTH1/KBD4	KBIE4

19.5 Functional Description

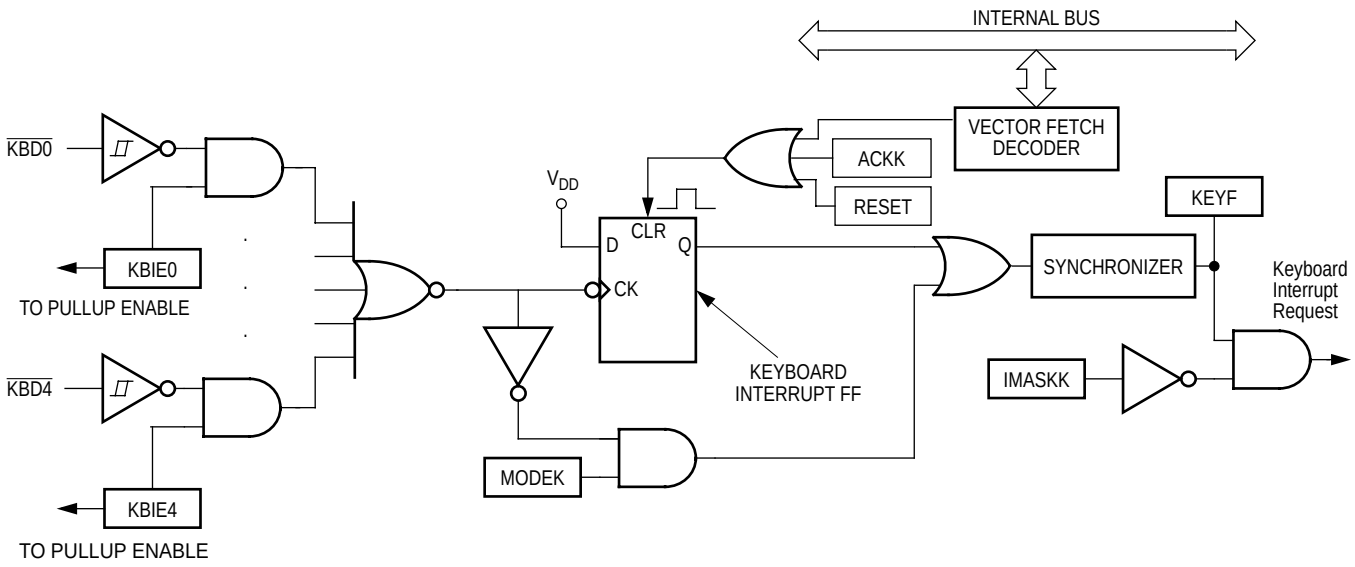


Figure 19-2. Keyboard Interrupt Block Diagram

Writing to the KBIE4–KBIE0 bits in the keyboard interrupt enable register independently enables or disables the corresponding port pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its internal pull-up device. A logic 0 applied to an enabled keyboard interrupt pin latches a keyboard interrupt request.

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

- If the keyboard interrupt is edge-sensitive only, a falling edge on a keyboard pin does not latch an interrupt request if another keyboard pin is already low. To prevent losing an interrupt request on one pin because another pin is still low, software can disable the latter pin while it is low.
- If the keyboard interrupt is falling edge- and low level-sensitive, an interrupt request is present as long as any keyboard pin is low.

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

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACKK bit in the keyboard status and control register KBSCR. The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the vector address at locations $FFD2 and $FFD3.
- Return of all enabled keyboard interrupt pins to logic 1 — As long as any enabled keyboard interrupt pin is at logic 0, the keyboard interrupt remains set.

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

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

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

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

To determine the logic level on a keyboard interrupt pin, disable the pull-up device, use the data direction register to configure the pin as an input and then read the data register.

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

19.5.1 Keyboard Initialization

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

To prevent a false interrupt on keyboard initialization:

1. Mask keyboard interrupts by setting the IMASKK bit in the keyboard status and control register.
2. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.
3. Write to the ACKK bit in the keyboard status and control register to clear any false interrupts.
4. Clear the IMASKK bit.

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

Another way to avoid a false interrupt:

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

19.5.2 Keyboard Status and Control Register

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

Address: $001B

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

= Unimplemented

Figure 19-3. Keyboard Status and Control Register (KBSCR)

Bits 7–4 — Not used

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

KEYF — Keyboard Flag Bit

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

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

ACKK — Keyboard Acknowledge Bit

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

IMASKK— Keyboard Interrupt Mask Bit

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

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

MODEK — Keyboard Triggering Sensitivity Bit

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

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

19.5.3 Keyboard Interrupt Enable Register

The keyboard interrupt enable register enables or disables the corresponding port pin to operate as a keyboard interrupt pin.

Address: $0021

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
Write:								
Port Pin:				PTH1/ KBD4	PTH0/ KBD3	PTG2/ KBD2	PTG1/ KBD1	PTG0 /KBD0
Reset:	0	0	0	0	0	0	0	0

Figure 19-4. Keyboard Interrupt Enable Register (KBIER)

KBIE4–KBIE0 — Keyboard Interrupt Enable Bits

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

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

19.6 Wait Mode

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

19.7 Stop Mode

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

19.8 Keyboard Module During Break Interrupts

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

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

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

Section 20. Computer Operating Properly (COP)

20.1 Contents

20.2 Introduction

The computer operating properly (COP) module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by clearing the COP counter periodically. The COP module can be disabled through the COPD bit in configuration register 1 (CONFIG1).

20.3 Functional Description

Figure 20-1 shows the structure of the COP module.

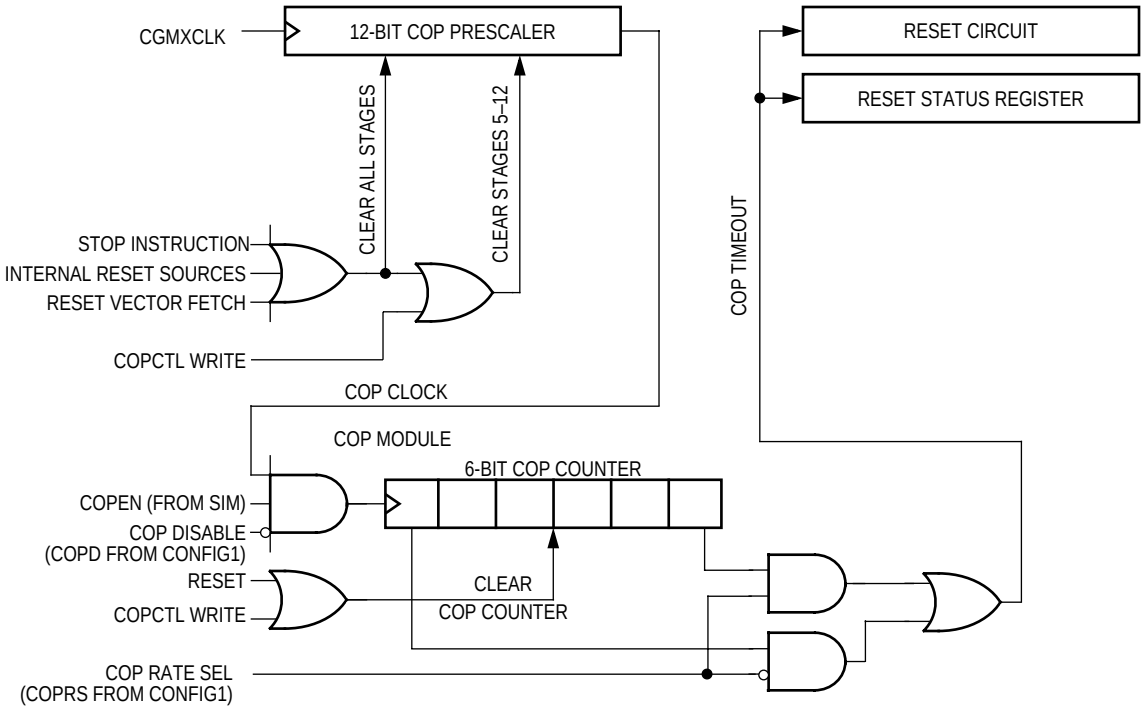


Figure 20-1. COP Block Diagram

The COP counter is a free-running 6-bit counter preceded by a 12-bit prescaler counter. If not cleared by software, the COP counter overflows and generates an asynchronous reset after $2^{18} - 2^4$ or $2^{13} - 2^4$ CGMXCLK cycles, depending on the state of the COP rate select bit, COPRS, in configuration register 1. With a $2^{18} - 2^4$ CGMXCLK cycle overflow option, a 4.9152MHz crystal gives a COP timeout period of 53.3ms. Writing any value to location $FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 12 through 5 of the prescaler.

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

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

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

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

20.4 I/O Signals

The following paragraphs describe the signals shown in **Figure 20-1**.

20.4.1 CGMXCLK

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

20.4.2 STOP Instruction

The STOP instruction clears the COP prescaler.

20.4.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see **20.5 COP Control Register**) clears the COP counter and clears bits 12 through 5 of the prescaler. Reading the COP control register returns the low byte of the reset vector.

20.4.4 Power-On Reset

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

20.4.5 Internal Reset

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

20.4.6 Reset Vector Fetch

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

20.4.7 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register 1. (See **Figure 20-2**.)

20.4.8 COPRS (COP Rate Select)

The COPRS signal reflects the state of the COP rate select bit (COPRS) in the configuration register 1. (See **Figure 20-2**.)

Address: $001F

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

R = Reserved

Figure 20-2. Configuration Register 1 (CONFIG1)

COPRS — COP Rate Select Bit

COPRS selects the COP timeout period. Reset clears COPRS.

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

COPD — COP Disable Bit

COPD disables the COP module.

1 = COP module disabled
0 = COP module enabled

20.5 COP Control Register

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

Address: $FFFF

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Low byte of reset vector							
Write:	Clear COP counter							
Reset:	Unaffected by reset							

Figure 20-3. COP Control Register (COPCTL)

20.6 Interrupts

The COP does not generate CPU interrupt requests.

20.7 Monitor Mode

When monitor mode is entered with V_{TST} on the $\overline{IRQ}$ pin, the COP is disabled as long as V_{TST} remains on the $\overline{IRQ}$ pin or the $\overline{RST}$ pin. When monitor mode is entered by having blank reset vectors and not having V_{TST} on the $\overline{IRQ}$ pin, the COP is automatically disabled until a POR occurs.

20.8 Low-Power Modes

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

20.8.1 Wait Mode

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

20.8.2 Stop Mode

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

To prevent inadvertently turning off the COP with a STOP instruction, a configuration option is available that disables the STOP instruction. When the STOP bit in the configuration register has the STOP instruction is disabled, execution of a STOP instruction results in an illegal opcode reset.

20.9 COP Module During Break Mode

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

Section 21. Low-Voltage Inhibit (LVI)

21.1 Contents

21.2 Introduction

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

21.3 Features

Features of the LVI module include the following:

- Programmable LVI reset
- Programmable LVI module power
- Programmable stop mode operation

21.4 Functional Description

Figure 21-1 shows the structure of the LVI module. The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. The LVI power bit, LVIPWRD, enables the LVI to monitor V_{DD} voltage. The LVI reset bit, LVIRSTD, enables the LVI module to generate a reset when V_{DD} falls below a voltage, LVI_{TRIPF}, and remains at or below that level for 9 or more consecutive CPU cycles. Setting the LVI enable in stop mode bit, LVISTOP, enables the LVI to operate in stop mode.

LVISTOP, LVIPWRD, and LVIRSTD are in the configuration register 1 (CONFIG1). See **Section 6. Configuration Register (CONFIG)** for details of the LVI's configuration bits. Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises above a voltage, LVI_{TRIPR}, which causes the MCU to exit reset. See **8.4.2.5 Low-Voltage Inhibit (LVI) Reset** for details of the interaction between the SIM and the LVI. 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.

NOTE: *Where LVI trip falling voltage $LVI_{TRIPF} = V_{LVII}$ and LVI trip rising voltage $LVI_{TRIPR} = V_{LVII} + H_{LVI}$ (See **Section 23. Electrical Specifications**.)*

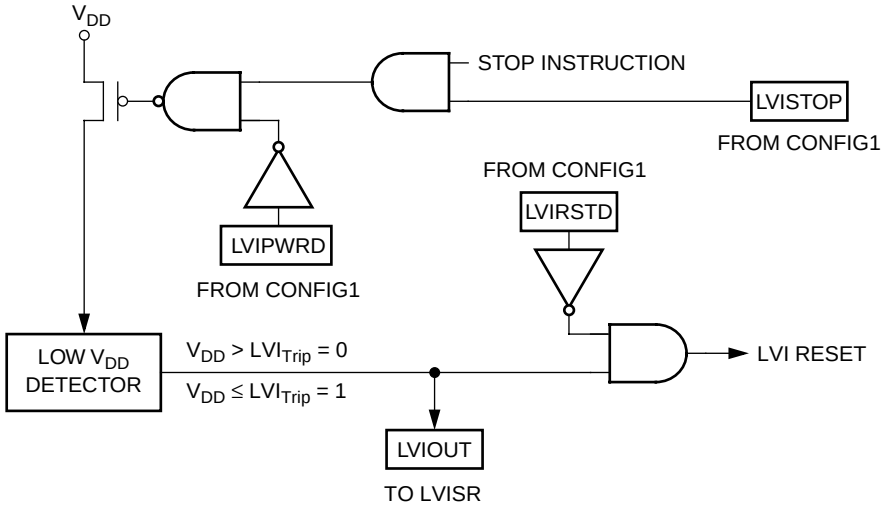


Figure 21-1. LVI Module Block Diagram

Address: $FE0F

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

= Unimplemented

Figure 21-2. LVI I/O Register Summary

21.4.1 Polled LVI Operation

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

21.4.2 Forced Reset Operation

In applications that require V_{DD} to remain above the LVI_{TRIPF} level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls below the LVI_{TRIPF} level and remains at or below that level for 9 or more consecutive CPU cycles. In configuration register 1, the LVIPWRD and LVIRSTD bits must be at logic 0 to enable the LVI module and to enable LVI resets.

21.4.3 False Reset Protection

The V_{DD} pin level is digitally filtered to reduce false resets due to power supply noise. In order for the LVI module to reset the MCU, V_{DD} must remain at or below the LVI_{TRIPF} level for 9 or more consecutive CPU cycles. V_{DD} must be above LVI_{TRIPR} for only one CPU cycle to bring the MCU out of reset.

21.5 LVI Status Register (LVISR)

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

Address: $FE0F

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

= Unimplemented

Figure 21-3. LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

This read-only flag becomes set when V_{DD} falls below the LVI_{TRIPF} voltage for 32 to 40 CGMXCLK cycles. (See **Table 21-1**.) Reset clears the LVIOUT bit.

Table 21-1. LVIOUT Bit Indication

V_{DD}		LVIOUT
At level:	**For number of CGMXCLK cycles:**	
$V_{DD} > LVI_{TRIPR}$	Any	0
$V_{DD} < LVI_{TRIPF}$	< 32 CGMXCLK cycles	0
$V_{DD} < LVI_{TRIPF}$	32 to 40 CGMXCLK cycles	0 or 1
$V_{DD} < LVI_{TRIPF}$	> 40 CGMXCLK cycles	1
$LVI_{TRIPF} < V_{DD} < LVI_{TRIPR}$	Any	Previous value

21.6 LVI Interrupts

The LVI module does not generate interrupt requests.

21.7 Low-Power Modes

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

21.7.1 Wait Mode

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

21.7.2 Stop Mode

If enabled in stop mode (LVISTOP set), the LVI module remains active in stop mode. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of stop mode.

Section 22. Break Module (BRK)

22.1 Contents

22.2 Introduction

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

22.3 Features

Features of the break module include:

- Accessible input/output (I/O) registers during the break interrupt
- CPU-generated break interrupts
- Software-generated break interrupts
- COP disabling during break interrupts

22.4 Functional Description

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

The following events can cause a break interrupt to occur:

- A CPU-generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a logic 1 to the BRKA bit in the break status and control register.

When a CPU-generated address matches the contents of the break address registers, the break interrupt begins after the CPU completes its current instruction. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. **Figure 22-1** shows the structure of the break module.

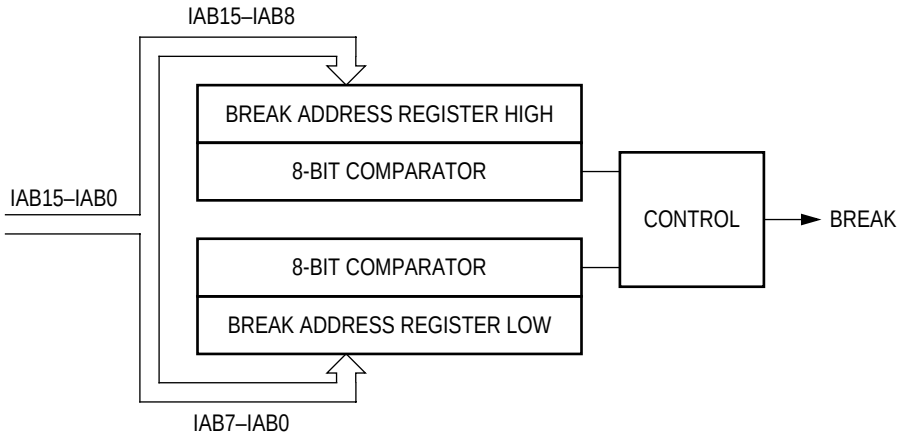


Figure 22-1. Break Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	SIM Break Status Register (SBSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							Note	
		Reset:	0	0	0	0	0	0	0	0
$FE03	SIM Break Flag Control Register (SBFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE0C	Break Address Register High (BRKH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0D	Break Address Register Low (BRKL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0E	Break Status and Control Register (BRKSCR)	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

Note: Writing a logic 0 clears SBSW.

= Unimplemented R = Reserved

Figure 22-2. Break Module I/O Register Summary

22.4.1 Flag Protection During Break Interrupts

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

22.4.2 CPU During Break Interrupts

The CPU starts a break interrupt by:

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

22.4.3 PIT, TIMA, and TIMB During Break Interrupts

A break interrupt stops all timer counters.

22.4.4 COP During Break Interrupts

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

22.5 Low-Power Modes

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

22.5.1 Wait Mode

If enabled, the break module is active in wait mode. In the break routine, the user can subtract one from the return address on the stack if SBSW is set (see **Section 8. System Integration Module (SIM)**). Clear the SBSW bit by writing logic 0 to it.

22.5.2 Stop Mode

A break interrupt causes exit from stop mode and sets the SBSW bit in the break status register.

22.6 Break Module Registers

These registers control and monitor operation of the break module:

- Break status and control register (BRKSCR)
- Break address register high (BRKH)
- Break address register low (BRKL)
- SIM Break status register (SBSR)
- SIM Break flag control register (SBFCR)

22.6.1 Break Status and Control Register

The break status and control register (BRKSCR) contains break module enable and status bits.

Address: $FE0E

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

= Unimplemented

Figure 22-3. Break Status and Control Register (BRKSCR)

BRKE — Break Enable Bit

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

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

BRKA — Break Active Bit

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

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

22.6.2 Break Address Registers

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

Address: $FE0C

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

Figure 22-4. Break Address Register High (BRKH)

Address: $FE0D

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

Figure 22-5. Break Address Register Low (BRKL)

22.6.3 SIM Break Status Register

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

Address: $FE00

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

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

Figure 22-6. SIM Break Status Register (SBSR)

SBSW — SIM Break Stop/Wait Bit

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

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

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

```
;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
```

22.6.4 SIM Break Flag Control Register

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

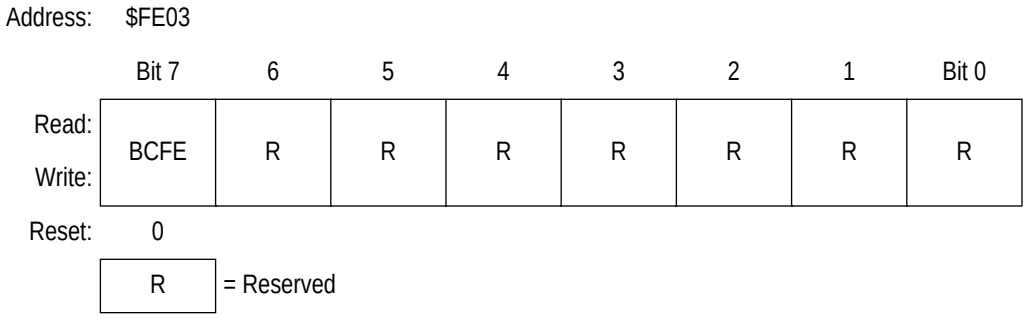


Figure 22-7. SIM Break Flag Control Register (SBFCR)

BCFE — Break Clear Flag Enable Bit

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

1 = Status bits clearable during break

0 = Status bits not clearable during break

Section 23. Electrical Specifications

23.1 Contents

23.2 Introduction

This section contains electrical and timing specifications.

23.3 Absolute Maximum Ratings

Maximum ratings are the extreme limits to which the MCU can be exposed without permanently damaging it.

NOTE: *This device is not guaranteed to operate properly at the maximum ratings. Refer to* ***23.6 5.0-V DC Electrical Characteristics*** *for guaranteed operating conditions.*

Characteristic(1)	Symbol	Value	Unit
Supply voltage	V_{DD}	–0.3 to + 6.0	V
Input voltage	V_{In}	V_{SS} – 0.3 to V_{DD} + 0.3	V
Maximum current per pin excluding V_{DD}, V_{SS}, and PTD0–PTD7	I	± 15	mA
Maximum current for pins PTD0–PTD7	$I_{PTD0–PTD7}$	± 25	mA
Maximum current into V_{DD}	I_{mvdd}	100	mA
Maximum current out of V_{SS}	I_{mvss}	100	mA
Storage temperature	T_{stg}	–55 to +150	°C

Notes:
1. Voltages referenced to V_{SS}

NOTE: *This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. For proper operation, it is recommended that V_{In} and V_{Out} be constrained to the range $V_{SS} \le (V_{In}$ or $V_{Out}) \le V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either V_{SS} or V_{DD}).*

23.4 Functional Operating Range

Characteristic	Symbol	Value		Unit
Operating temperature range	T_A	–40 to +85	–40 to +125	°C
Operating voltage range	V_{DD}	5.0 ±10%	5.0 ±10%	V

23.5 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance QFP (64-pin)	θ_{JA}	70	°C/W
I/O pin power dissipation	$P_{I/O}$	User-Determined	W
Power dissipation(1)	P_D	$P_D = (I_{DD} \times V_{DD}) + P_{I/O} = K/(T_J + 273\ °C)$	W
Constant(2)	K	$P_D \times (T_A + 273\ °C) + P_D^2 \times \theta_{JA}$	W/°C
Average junction temperature	T_J	$T_A + (P_D \times \theta_{JA})$	°C

Notes:
1. Power dissipation is a function of temperature.
2. K is a constant unique to the device. K can be determined for a known T_A and measured P_D. With this value of K, P_D and T_J can be determined for any value of T_A.

23.6 5.0-V DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Output high voltage					
(I_{Load} = –2.0 mA) all I/O pins	V_{OH}	V_{DD} – 0.8	—	—	V
(I_{Load} = –10.0 mA) all I/O pins	V_{OH}	V_{DD} – 1.5	—	—	V
(I_{Load} = –10.0 mA) pins PTD0–PTD7 only	V_{OH}	V_{DD} – 0.8	—	—	V
Maximum combined I_{OH} for port C, port E, port F, port G, port H	I_{OH1}	—	—	50	mA
Maximum combined I_{OH} for port D, port A, port B	I_{OH2}	—	—	50	mA
Maximum total I_{OH} for all port pins	I_{OHT}	—	—	100	mA
Output low voltage					
(I_{Load} = 1.6 mA) all I/O pins	V_{OL}	—	—	0.4	V
(I_{Load} = 10 mA) all I/O pins	V_{OL}	—	—	1.5	V
(I_{Load} = 15 mA) pins PTD0–PTD7 only	V_{OL}	—	—	1.0	V
Maximum combined I_{OL} for port C, port E, port F, port G, port H	I_{OL1}	—	—	50	mA
Maximum combined I_{OL} for port D, port A, port B	I_{OL2}	—	—	50	mA
Maximum total I_{OL} for all port pins	I_{OLT}	—	—	100	mA
Input high voltage All ports, $\overline{IRQ}$, $\overline{RST}$, OSC1	V_{IH}	$0.7 \times V_{DD}$	—	V_{DD}	V
Input low voltage All ports, $\overline{IRQ}$, $\overline{RST}$, OSC1	V_{IL}	V_{SS}	—	$0.3 \times V_{DD}$	V
V_{DD} supply current					
Run[3]	I_{DD}	—	—	30	mA
Wait[4]		—	—	12	mA
Stop[5]					
LVI enabled, T_A = 25 °C		—	300	400	µA
LVI disabled, T_A = 25 °C		—	20	50	µA
LVI enabled, T_A = –40 °C to 125 °C		—	400	500	µA
LVI disabled, T_A = –40 °C to 125 °C		—	50	100	µA
I/O ports Hi-Z leakage current[6]	I_{IL}	—	1	±10	µA
Input current	I_{In}	—	—	±1	µA
Pullup resistors (as input only)	R_{PU}	20	33	50	kΩ
Capacitance Ports (as input or output)	C_{Out} C_{In}	— —	— —	12 8	pF
Monitor mode entry voltage	V_{TST}	V_{DD} + 2.5	—	8	V
Low-voltage inhibit, trip falling voltage – target	V_{LVII}	—	4.11	—	V

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Low-voltage inhibit reset/recover hysteresis – target	H_{LVI}	100	150	—	mV
POR rearm voltage[7]	V_{POR}	0	—	200	mV
POR reset voltage[8]	V_{PORRST}	0	—	800	mV
POR rise time ramp rate[9]	R_{POR}	0.02	—	—	V/ms

Notes:
1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted
2. Typical values reflect average measurements at midpoint of voltage range, 25 °C only.
3. Run (operating) I_{DD} measured using external square wave clock source (f_{BUS} = 8.4 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run I_{DD}. Measured with all modules enabled.
4. Wait I_{DD} measured using external square wave clock source (f_{BUS} = 8.4 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait I_{DD}. Measured with PLL and LVI enabled.
5. Stop I_{DD} is measured with OSC1 = V_{SS}.
6. Pullups are disabled. Port B leakage is specified in **23.10 ADC Characteristics**.
7. Maximum is highest voltage that POR is guaranteed.
8. Maximum is highest voltage that POR is possible.
9. If minimum V_{DD} is not reached before the internal POR reset is released, $\overline{RST}$ must be driven low externally until minimum V_{DD} is reached.

23.7 EEPROM and Memory Characteristics

Characteristic	Symbol	Min	Max	Unit
RAM data retention voltage	V_{RDR}	0.7	—	V
EEPROM programming time per byte	t_{EEPGM}	10	—	ms
EEPROM erasing time per byte	t_{EBYTE}	10	—	ms
EEPROM erasing time per block	t_{EBLOCK}	10	—	ms
EEPROM erasing time per bulk	t_{EBULK}	10	—	ms
EEPROM programming voltage discharge period	t_{EEFPV}	100	—	µs
Number of programming operations to the same EEPROM byte before erase[1]	—	—	8	—
EEPROM write/erase cycles at 10 ms write time (85°C)	—	10,000	—	Cycles
EEPROM data retention after 10,000 write/erase cycles	—	10	—	Years

Notes:
1. Programming a byte more times than the specified maximum may affect the data integrity of that byte. The byte must be erased before it can be programmed again.

23.8 5.0-V Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Frequency of operation Crystal option[2] External clock option[3]	f_{OSC}	 1 dc[4]	 8.4 33.6	 MHz MHz
Internal operating frequency	f_{BUS}	Note[5]	8.4	MHz
$\overline{\text{RESET}}$ input pulse width low[6]	t_{IRL}	1.5	—	t_{cyc}
$\overline{\text{IRQ}}$ interrupt pulse width low[7] (Edge-triggered)	t_{ILIH}	1.5	—	t_{cyc}
$\overline{\text{IRQ}}$ interrupt pulse period	t_{ILIL}	Note[8]	—	t_{cyc}
16-bit timer Input capture pulse width Input capture period	 t_{TH},t_{TL} t_{TLTL}	 2 Note[9]	 — —	 t_{cyc} t_{cyc}

Notes:
1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted
2. See **23.12 Clock Generation Module Characteristics** for more information.
3. No more than 10% duty cycle deviation from 50%
4. Some modules may require a minimum frequency greater than dc for proper operation. See appropriate table for this information.
5. Some modules may require a minimum frequency greater than dc for proper operation. See appropriate table for this information.
6. Minimum pulse width reset is guaranteed to be recognized. It is possible for a smaller pulse width to cause a reset.
7. Minimum pulse width is for guaranteed interrupt. It is possible for a smaller pulse width to be recognized.
8. Minimum pulse width is for guaranteed interrupt. It is possible for a smaller pulse width to be recognized. The minimum period, t_{ILIL} or t_{TLTL}, should not be less than the number of cycles it takes to execute the interrupt service routine plus TBD t_{cyc}.
9. The minimum period, t_{ILIL} or t_{TLTL}, should not be less than the number of cycles it takes to execute the interrupt service routine plus TBD t_{cyc}.

23.9 Timer Interface Module Characteristics

Characteristic	Symbol	Min	Max	Unit
Input capture pulse width	t_{TIH}, t_{TIL}	125	—	ns
Input clock pulse width	t_{TCH}, t_{TCL}	$(1/f_{BUS})+5$	—	ns

23.10 ADC Characteristics

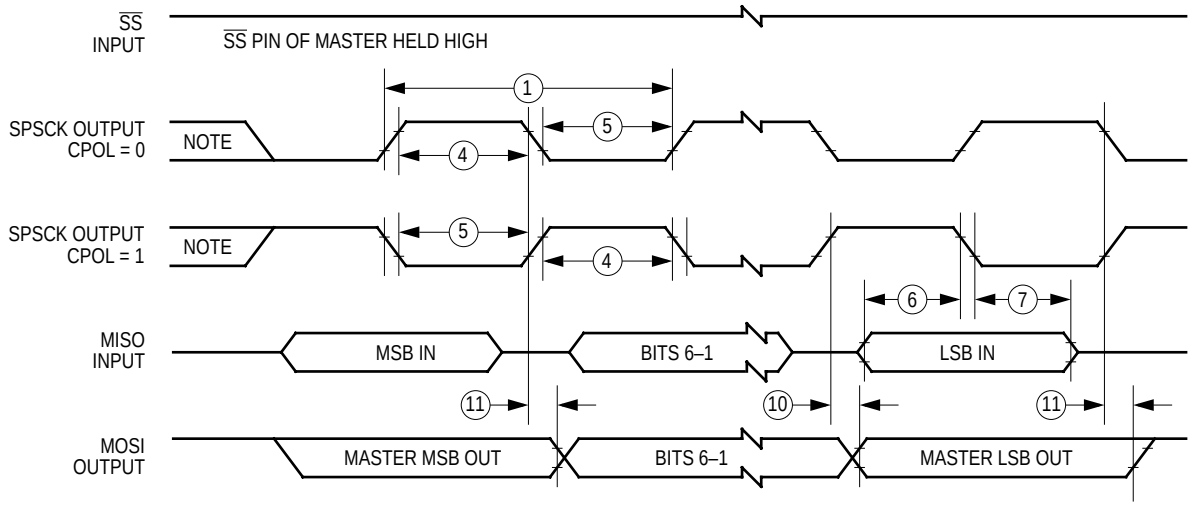
Characteristic[1]	Symbol	Min	Max	Unit	Comments
Supply voltage	V_{DDAD}	4.5 (V_{DD} min)	5.5 (V_{DD} max)	V	V_{DDAREF} should be tied to the same potential as V_{DD} via separate traces.
Input voltages	V_{ADIN} V_{REFH}	0 1.5	V_{DDAREF} V_{DDAREF}	V	$V_{ADIN} \leq V_{REFH}$
Resolution	B_{AD}	8	8	Bits	
Absolute accuracy (V_{REFL} = 0 V, V_{REFH} = V_{DDAD} = 5 V ± 10%)	A_{AD}	± 1/2	± 1	LSB	Includes quantization
ADC internal clock	f_{ADIC}	0.5	1.048	MHz	$t_{AIC} = 1/f_{ADIC}$, tested only at 1 MHz
Conversion range	R_{AD}	V_{REFL}	V_{REFH}	V	$V_{REFL} = V_{SSA}$
Power-up time	t_{ADPU}	16		t_{AIC} cycles	
Conversion time	t_{ADC}	16	17	t_{AIC} cycles	
Sample time[2]	t_{ADS}	5	—	t_{AIC} cycles	
Zero input reading[3]	Z_{ADI}	00	01	Hex	$V_{IN} = V_{REFL}$
Full-scale reading[3]	F_{ADI}	FE	FF	Hex	$V_{IN} = V_{REFH}$
Input capacitance	C_{ADI}	—	(20) 8	pF	Not tested
Input leakage[4] Port B	—	—	± 1	µA	

Notes:
1. V_{DD} = 5.0 Vdc ± 10%, V_{DDA} = V_{DDAREF} = 5.0 Vdc ± 10%, V_{REFH} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, V_{REFL} = V_{SSA} = 0 Vdc
2. Source impedances greater than 10 kΩ adversely affect internal RC charging time during input sampling.
3. Zero-input/full-scale reading requires sufficient decoupling measures for accurate conversions.
4. The external system error caused by input leakage current is approximately equal to the product of R source and input current.

23.11 SPI Characteristics

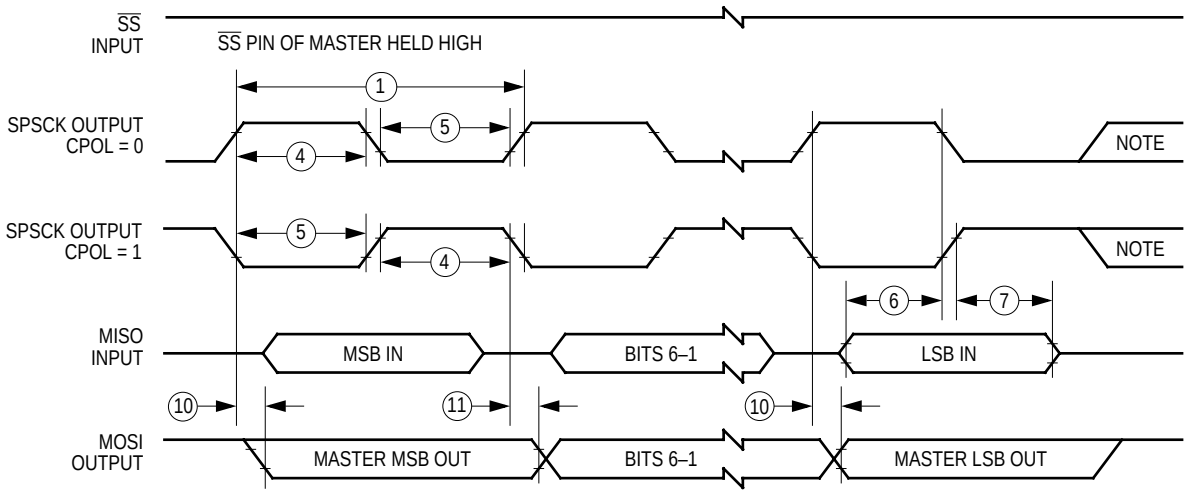
Diagram Number[1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	 $f_{BUS(M)}$ $f_{BUS(S)}$	 $f_{BUS}/128$ DC	 $f_{BUS}/2$ f_{BUS}	 MHz MHz
1	Cycle time Master Slave	 $t_{cyc(M)}$ $t_{cyc(S)}$	 2 1	 128 —	 t_{cyc} t_{cyc}
2	Enable lead time	$t_{Lead(S)}$	15	—	ns
3	Enable lag time	$t_{Lag(S)}$	15	—	ns
4	Clock (SPSCK) high time Master Slave	 $t_{SCKH(M)}$ $t_{SCKH(S)}$	 100 50	 — —	 ns ns
5	Clock (SPSCK) low time Master Slave	 $t_{SCKL(M)}$ $t_{SCKL(S)}$	 100 50	 — —	 ns ns
6	Data setup time (inputs) Master Slave	 $t_{SU(M)}$ $t_{SU(S)}$	 45 5	 — —	 ns ns
7	Data hold time (inputs) Master Slave	 $t_{H(M)}$ $t_{H(S)}$	 0 15	 — —	 ns ns
8	Access time, slave[3] CPHA = 0 CPHA = 1	 $t_{A(CP0)}$ $t_{A(CP1)}$	 0 0	 40 20	 ns ns
9	Disable time, slave[4]	$t_{DIS(S)}$	—	25	ns
10	Data valid time, after enable edge Master Slave[5]	 $t_{V(M)}$ $t_{V(S)}$	 — —	 10 40	 ns ns
11	Data hold time, outputs, after enable edge Master Slave	 $t_{HO(M)}$ $t_{HO(S)}$	 0 5	 — —	 ns ns

Notes:
1. Numbers refer to dimensions in **Figure 23-1** and **Figure 23-2**.
2. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; 100 pF load on all SPI pins.
3. Time to data active from high-impedance state
4. Hold time to high-impedance state
5. With 100 pF on all SPI pins



Note: This first clock edge is generated internally, but is not seen at the SPSCK pin.

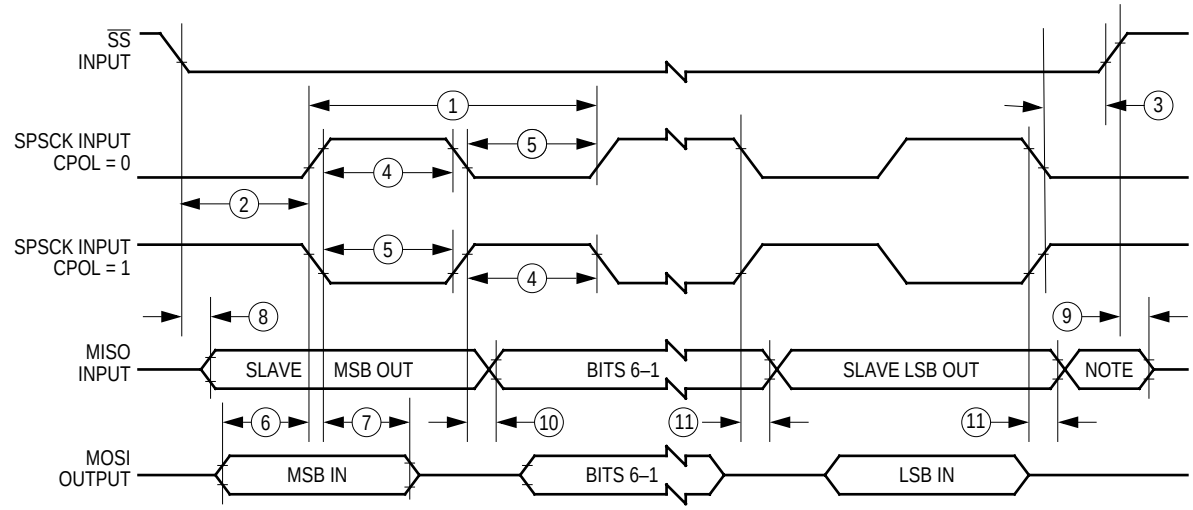
a) SPI Master Timing (CPHA = 0)



Note: This last clock edge is generated internally, but is not seen at the SPSCK pin.

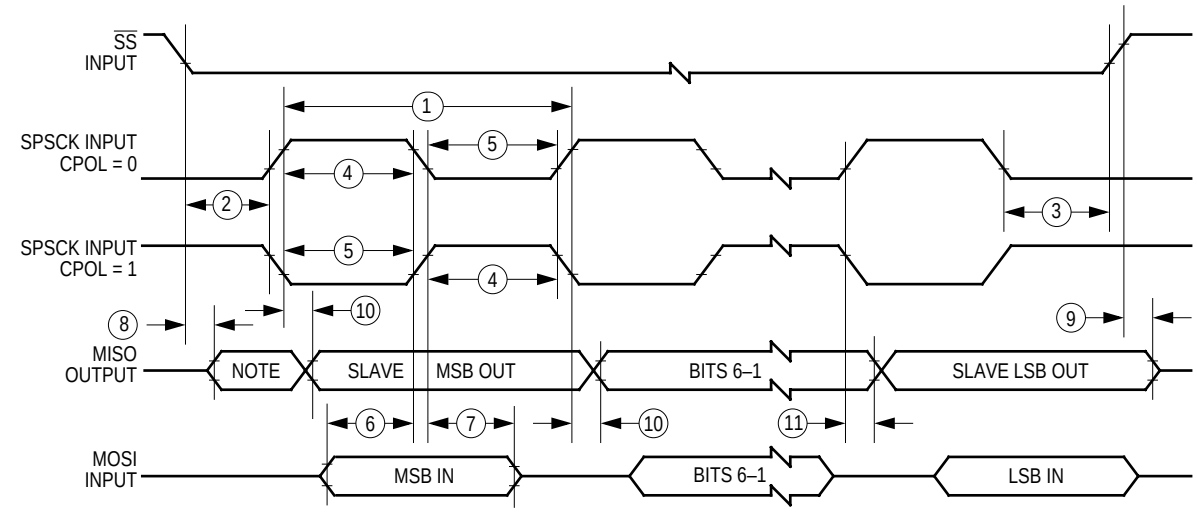
b) SPI Master Timing (CPHA = 1)

Figure 23-1. SPI Master Timing



Note: Not defined but normally MSB of character just received

a) SPI Slave Timing (CPHA = 0)



Note: Not defined but normally LSB of character previously transmitted

b) SPI Slave Timing (CPHA = 1)

Figure 23-2. SPI Slave Timing

23.12 Clock Generation Module Characteristics

23.12.1 CGM Operating Conditions

Characteristic	Symbol	Min	Typ	Max	Comments
Operating Voltage	V_{DD}	4.5 V	—	5.5 V	
Crystal Reference Frequency	f_{RCLK}	1	—	8.4	
Module Crystal Reference Frequency	f_{XCLK}	—	4.9152 MHz	—	Same Frequency as f_{RCLK}
Range Nominal Multiplier	f_{NOM}	—	4.9152 MHz	—	4.5 to 5.5 V, V_{DD} only
VCO Center-of-Range Frequency (MHz)	f_{VRS}	4.9152	—	32.0	4.5 to 5.5 V, V_{DD} only
VCO Operating Frequency (MHz)	f_{VCLK}	4.9152	—	32.0	

23.12.2 CGM Component Information

Characteristic	Symbol	Min	Typ	Max	Unit
Crystal load capacitance	C_L	—	—	—	Consult crystal manufacturer's data
Crystal fixed capacitance	C_1	—	$2 \times C_L$	—	Consult crystal manufacturer's data
Crystal tuning capacitance	C_2	—	$2 \times C_L$	—	Consult crystal manufacturer's data
Feedback bias resistor	R_B	—	22MΩ	—	
Series resistor	R_S	—	330kΩ	1MΩ	Not required
Filter capacitor multiply factor	C_{Fact}	—	0.0154	—	F/s V
Filter capacitor	C_F	—	$C_{Fact} \times (V_{DDA}/f_{XCLK})$	—	See **9.5.3 External Filter Capacitor Pin (CGMXFC)**.
Bypass capacitor	C_{BYP}	—	0.1 µF	—	C_{BYP} must provide low AC impedance from f = $f_{XCLK}/100$ to $100 \times f_{VCLK}$, so series resistance must be considered.

23.12.3 CGM Acquisition/Lock Time Information

Description[1]	Symbol	Min	Typ	Max	Notes
Manual mode time to stable	t_{ACQ}	—	$(8 \times V_{DDA})/(f_{XCLK} \times K_{ACQ})$	—	If C_F chosen correctly
Manual stable to lock time	t_{AL}	—	$(4 \times V_{DDA})/(f_{XCLK} \times K_{TRK})$	—	If C_F chosen correctly
Manual acquisition time	t_{Lock}	—	$t_{ACQ}+t_{AL}$	—	
Tracking mode entry frequency tolerance	D_{TRK}	0	—	± 3.6%	
Acquisition mode entry frequency tolerance	D_{UNT}	± 6.3%	—	± 7.2%	
LOCK entry freq. tolerance	D_{LOCK}	0	—	± 0.9%	
LOCK exit freq. tolerance	D_{UNL}	± 0.9%	—	± 1.8%	
Reference cycles per Acquisition mode measurement	n_{ACQ}	—	32	—	
Reference cycles per Tracking mode measurement	n_{TRK}	—	128	—	
Automatic mode time to stable	t_{ACQ}	n_{ACQ}/f_{XCLK}	$(8 \times V_{DDA})/(f_{XCLK} \times K_{ACQ})$		If C_F Chosen Correctly
Automatic stable to lock time	t_{AL}	n_{TRK}/f_{XCLK}	$(4 \times V_{DDA})/(f_{XCLK} \times K_{TRK})$	—	If C_F Chosen Correctly
Automatic lock time	t_{Lock}	—	$t_{ACQ} + t_{AL}$	—	
PLL jitter, deviation of average bus frequency over 2 ms		0	—	± (f_{CRYS}) × (.025%) × (N/4)	N = VCO Freq. Mult. (GBNT)[2]

Notes:

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted.
2. GBNT guaranteed but not tested.

23.13 FLASH Memory Characteristics

Characteristic	Symbol	Min	Max	Unit
FLASH program bus clock frequency	—	1	—	MHz
FLASH read bus clock frequency	f_{Read} (1)	32k	8.4M	Hz
FLASH page erase time	t_{Erase} (2)	1	—	ms
FLASH mass erase time	t_{MErase} (3)	4	—	ms
FLASH PGM/ERASE to HVEN set up time	t_{nvs}	10	—	µs
FLASH high-voltage hold time	t_{nvh}	5	—	µs
FLASH high-voltage hold time (mass erase)	t_{nvhl}	100	—	µs
FLASH program hold time	t_{pgs}	5	—	µs
FLASH program time	t_{PROG}	30	40	µs
FLASH return to read time	t_{rcv} (4)	1	—	µs
FLASH cumulative program hv period	t_{HV} (5)	—	4	ms
FLASH row erase endurance (6)	—	10,000	—	Cycles
FLASH row program endurance (7)	—	10,000	—	Cycles
FLASH data retention time (8)	—	10	—	Years

Notes:

1. f_{Read} is defined as the frequency range for which the FLASH memory can be read.
2. If the page erase time is longer than t_{Erase} (Min), there is no erase-disturb, but it reduces the endurance of the FLASH memory.
3. If the mass erase time is longer than t_{MErase} (Min), there is no erase-disturb, but it reduces the endurance of the FLASH memory.
4. t_{rcv} is defined as the time it needs before the FLASH can be read after turning off the high voltage charge pump, by clearing HVEN to logic 0.
5. t_{HV} is defined as the cumulative high voltage programming time to the same row before next erase. t_{HV} must satisfy this condition: $t_{nvs} + t_{nvh} + t_{pgs} + (t_{PROG} \times 64) \leq t_{HV}$ max.
6. The minimum row endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many erase / program cycles.
7. The minimum row endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many erase / program cycles.
8. The FLASH is guaranteed to retain data over the entire operating temperature range for at least the minimum time specified.

Section 24. Mechanical Specifications

24.1 Contents

24.2 Introduction

This section gives the dimensions for:

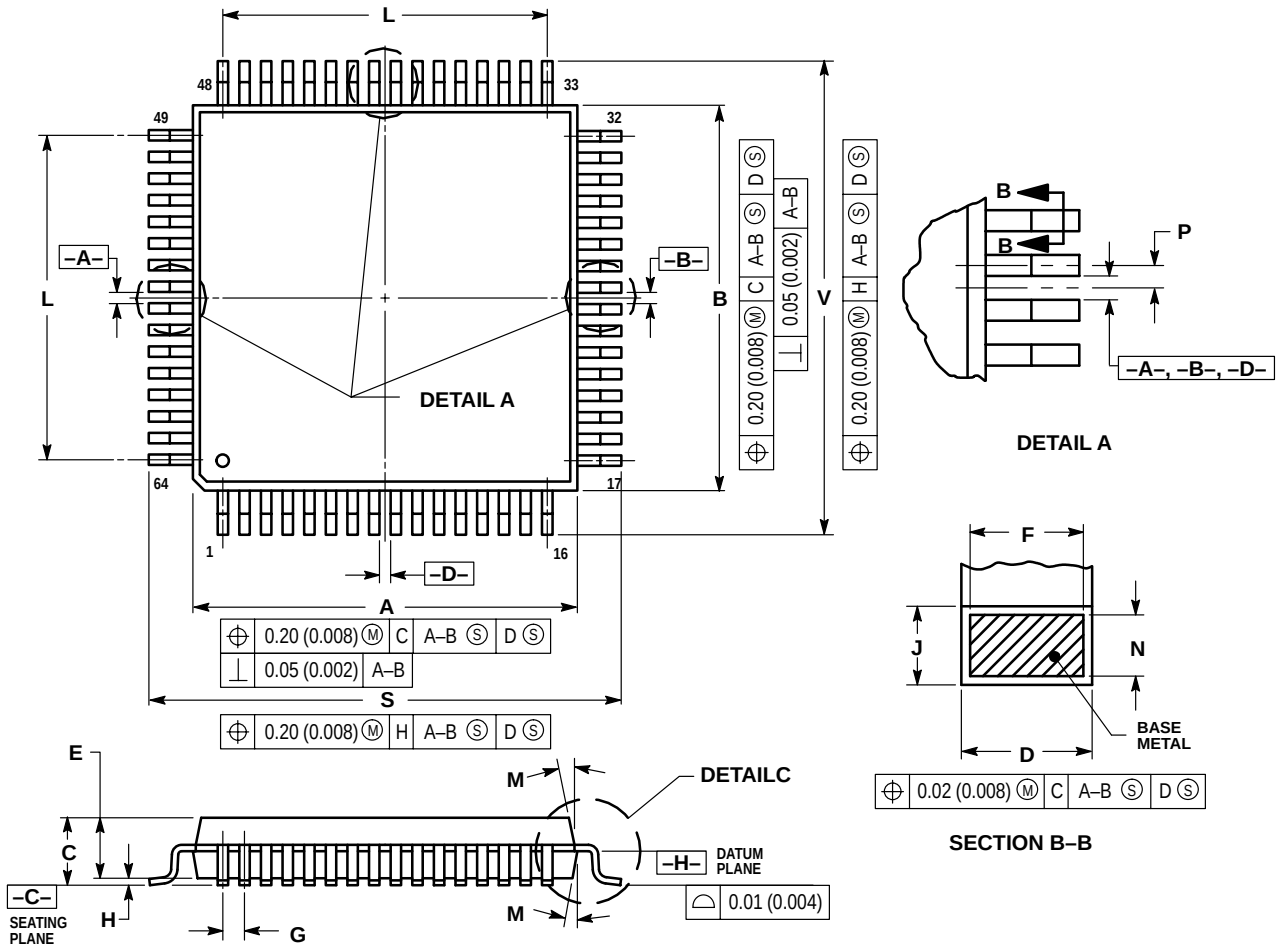
- 64-pin plastic quad flat pack (case 840B-01)

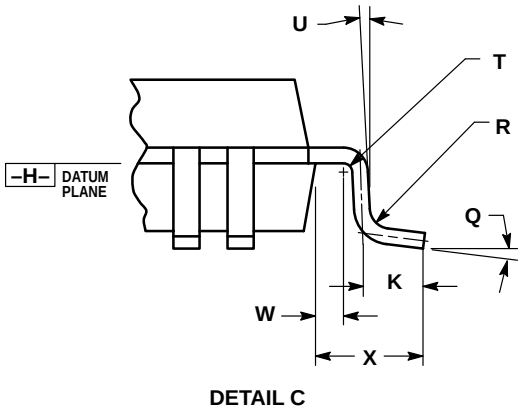
Figure 24-1 shows the latest package drawing at the time of this publication. To make sure that you have the latest package specifications, contact one of the following:

- Local Motorola Sales Office
- World Wide Web at http://www.motorola.com/semiconductors/

Follow the World Wide Web on-line instructions to retrieve the current mechanical specifications.

24.3 64-Pin Plastic Quad Flat Pack (QFP)





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) PER SIDE. 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 24-1. 64-Pin Plastic Quad Flat Pack (QFP)

Section 25. Ordering Information

25.1 Contents

25.2 Introduction

This section contains ordering numbers for the MC68HC908AB32.

25.3 MC Order Numbers

Table 25-1. MC Order Numbers

MC order number[1]	Operating temperature range
MC68HC908AB32CFU	–40 °C to +85 °C
MC68HC908AB32MFU	–40 °C to +125 °C

Notes:
1. FU = Quad Flat Pack
   C = Operating temperature range: –40 °C to +85 °C
   M = Operating temperature range: –40 °C to +125 °C

Ordering Information

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140 or 1-800-441-2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3-20-1, Minami-Azabu, Minato-ku, Tokyo 106-8573 Japan. 81-3-3440-3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852-26668334

Technical Information Center: 1-800-521-6274

HOME PAGE: http://www.motorola.com/semiconductors/

68HC908AB32 : Microcontroller

The MC68HC908AB32 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). The M68HC08 Family is based on the customer-specified integrated circuit (CSIC) design strategy. All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

68HC908AB32 Features

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 families
- Memory map and pin functions compatible with MC68HC08AB32 and MC68HC08AB16
- 8MHz internal bus frequency at 85°C
- 32K-bytes user program FLASH memory with security feature
- 512 bytes of on-chip EEPROM with security feature
- 1K byte of on-chip RAM
- Clock generator module (CGM)
- Two 16-bit timer interface modules (TIMA and TIMB) with selectable input capture, output compare, and PWM capability on each channel
- Periodic Interrupt Timer (PIT)
- Clock Generator Module (CGM)

[top]

68HC908AB32 Parametrics

ROM (Bytes)	RAM (Bytes)	Flash (KBytes)	EEPROM (Bytes)	Timer	I/O	Serial	Multiplexing	A/D	PWM	Operating Voltage (V)	Bus Frequency (Max) (MHz)	Availability
-	1K	32	512	4-CH + 4-CH 16-Bit I/C, O/C, or PWM	51	SCI SPI	-	8-CH 8-Bit	See Timer	5.0	8.0	Production

[top]

68HC908AB32 Documentation

Application Note

ID	Name	Format	Size K	Rev #	Date Last Modified	Order	Availability
AN1050/D	Designing for Electromagnetic Compatibility (EMC) with HCMOS Microcontrollers	pdf	82	0	1/01/2000		
AN1218/D	HC05 to HC08 Optimization	pdf	347	2	1/01/1993		
AN1219/D	M68HC08 Integer Math Routines	pdf	177	1	1/01/1997		
AN1219SW	Software Files for AN1219 zipped	zip	77	0	1/01/1995		-
AN1221/D	Hamming Error Control Coding Techniques with the HC08 MCU	pdf	63	0	1/01/1993		
AN1221SW	Software Files for AN1221 zipped	zip	55	0	1/01/1995		-
AN1222/D	Arithmetic Waveform Synthesis with the HC05/08 MCUs	pdf	24	0	1/01/1993		
AN1222SW	Software Files for AN1222 zipped	zip	20	0	1/01/1995		-
AN1259/D	System Design and Layout Techniques for Noise Reduction in MCU-Based Systems	pdf	78	0	1/01/1995		
AN1263/D	Designing for Electromagnetic Compatibility with Single-Chip Microcontrollers	pdf	104	0	1/01/1995		
AN1274/D	HC08 SCI Operation with Various Input Clocks	pdf	47	0	1/01/1996		
AN1705/D	Noise Reduction Techniques for Microcontroller-Based Systems	pdf	67	0	1/01/1999		
AN1744/D	Resetting Microcontrollers During Power Transitions	pdf	80	0	1/01/1998		
AN1752/D	Data Structures for 8-Bit Microcontrollers	pdf	213	1	5/07/2001		
AN1771/D	Precision Sine-Wave Tone Synthesis Using 8-Bit MCUs	pdf	250	0	1/01/1998		
AN1775/D	Expanding Digital Input with an A/D Converter	pdf	86	1	1/01/1998		
AN1783/D	Determining MCU Oscillator Start-Up Parameters	pdf	48	1	1/01/1999		
AN1818/D	Software SCI Routines with the 16-Bit Timer Module	pdf	84	0	1/01/1999		
AN1820/D	Software I2C Communications	pdf	55	0	1/01/1999		
AN1820SW	Software files for AN1820 zipped	zip	2	0	1/01/1998		-
AN1837/D	Non-Volatile Memory Technology Overview	pdf	116	0	3/27/2000		
AN1843/D	Vacuum Cleaner Reference Platform	pdf	326	0	9/13/2000		
AN1853/D	Embedding Microcontrollers in Domestic Refrigeration Appliances	pdf	221	0	6/22/2000		
AN2093/D	Creating Efficient C Code for the MC68HC08	pdf	36	0	1/01/2000		
AN2103/D	Local Interconnect Network (LIN) Demonstration	pdf	953	0	12/01/2000		

AN2120/D	Connecting an M68HC08 Family Microcontroller to an Internet Service Provider (ISP) Using the Point-to-Point Protocol (PPP)	pdf	741	0	5/20/2001	
AN2120SW	Software for AN2120, zip format	zip	31	1.0	7/31/2002	-
AN2149/D	Compressor Induction Motor Stall and Rotation Detection using Microcontrollers	pdf	127	0	5/30/2001	
AN2159/D	Digital Direct Current Ignition System Using HC08 Microcontrollers	pdf	129	0	11/20/2001	
AN2159SW	AN2159SW	zip	182	1	3/08/2002	-
AN2295/D	Developer's Serial Bootloader for M68HC08	pdf	324	1	10/04/2002	
AN2295SW	Software for AN2295	zip	210	1	10/07/2002	-
AN2321/D	Designing for Board Level Electromagnetic Compatibility	pdf	1628	0	8/15/2002	
AN2342/D	Opto Isolation Circuits For In Circuit Debugging of 68HC9(S)12 and 68HC908 Microcontrollers	pdf	156	0	9/25/2002	
AN4006/D	Digital Captive Discharge Ignition System Using HC05/HC08 8-Bit Microcontrollers	pdf	61	0	3/27/2000	

Brochure

ID	Name	Format	Size K	Rev #	Date Last Modified	Order Availability
8-16BITPAK/D	8-16 Bit Microcontrollers Product Portfolio	html	1	0	10/15/2002	
BR1822	Embedded Flash MCU Overview	pdf	174	-	-	-
BR1853/D	BR1853 68HC908AB32 Brochure	pdf	154	0	8/07/2000	-
BR68HC08FAMAM/D	68HC08 Family: High Performance and Flexibility	pdf	65	1	4/04/2002	
CWDEVSTUDFACT	CodeWarrior™ Development Studio for 68HC08, Special Edition Brochure	pdf	49	0	4/22/2002	-
FLYREMBEDFLASH/D	Embedded Flash: Changing the Technology World for the Better	pdf	621	1	4/03/2002	

Data Sheets

ID	Name	Format	Size K	Rev #	Date Last Modified	Order Availability
MC68HC908AB32/D	MC68HC908AB32 HCMOS Microcontroller Unit	pdf	3706	1.0	8/26/2000	

Engineering Bulletin

ID	Name	Format	Size K	Rev #	Date Last Modified	Order Availability
EB349/D	RAM Data Retention Considerations for Motorola Microcontrollers	pdf	45	1	6/22/2000	

EB387/D	MMDS and MMEVS Changes for MC68HC908ASxxA, MC68HC908AZxxA, and MC68HC908AB32	pdf	61	0	7/16/2001	
EB389/D	TOF Consideration when Measuring a Long Input Capture Event	pdf	55	1	4/15/2002	
EB390/D	Porting the AN2120/D UDP/IP Code to the Avnet Evaluation Board	pdf	1501	0	5/09/2002	
EB396/D	Use of OSC2/XTAL as a Clock Output on Motorola Microcontrollers	pdf	49	0	6/19/2002	
EB398/D	Techniques to Protect MCU Applications Against Malfunction Due to Code Run-Away	pdf	87	0	8/13/2002	
EB608/D	Interrupt Handling Considerations When Modifying EEPROM on HC08 Microcontrollers	pdf	96	0	8/14/2002	

Miscellaneous

ID	Name	Format	Size K	Rev #	Date Last Modified	Order Availability
68HC908AB32PB/D	8-bit Microcontroller	pdf	54	1	4/03/2002	

Product Change Notices

ID	Name	Format	Size K	Rev #	Date Last Modified	Order Availability
PCN7855	14X14 QFP ASSY MOVE FROM SHC TO KLM, PT 1 OF 2	htm	24	0	8/05/2002	-
PCN7856	14X14 QFP ASSY MOVE FROM SHC TO KLM, PT 2 OF 2	htm	22	0	8/05/2002	-
PCN8103	10X10 LQFP ASSY MOVE FROM SHC TO BAT3	htm	16	0	10/08/2002	-

Reference Manual

ID	Name	Format	Size K	Rev #	Date Last Modified	Order Availability
ADCRM/AD	Analog-to-Digital Reference Manual	pdf	231	0	1/01/1996	
CPU08RM/AD	CPU08RM Central Processor Unit Reference Manual	pdf	2666	3	4/03/2002	
DMA08RM/D	Direct Memory Access Reference Manual	pdf	463	1	2/23/2000	-
DRM001/D	Passive Infrared (PIR) Intruder Detection Using the MC68HC908JK1/3, Incorporating Remote Control Adjustment Using the MC68HC908GP32	pdf	2504	0	2/20/2001	
DRM002/D	USB08 Universal Serial Bus Evaluation Board Using the MC68HC908JB8 Designer Reference Manual	pdf	1845	0	4/12/2001	
M68HC08RG/D	HC08 Family Reference Guide	pdf	142	2	4/08/2002	-
TIM08RM/AD	TIM08 Timer Interface Module Reference Manual	pdf	771	1.0	1/10/1996	

Selector Guide

ID	Name	Format	Size K	Rev #	Date Last Modified	Order Availability
SG1006/D	Microcontrollers SPS Sales Guide	pdf	600	0	9/26/2002	
SG1011/D	Software and Development Tools Sales Guide	pdf	259	1	9/26/2002	
SG187/D	Automotive Selector Guide2Q, 2002	pdf	1098	10	3/28/2002	
SG2000CR/D	Application Selector Guide Index and Cross-Reference.	pdf	62	0	6/24/2002	
SG2036/D	Application Summary Home Appliances - Cooking Products. Microcontrollers provide intelligent management programs delivering high precision control over the cooking process.	pdf	78	0	6/24/2002	
SG2037/D	Application Summary - Home Appliances - Dishwashers. Microcontrollers enable the electronic control which is used to provide a range of dishwasher appliance features.	pdf	77	0	6/24/2002	
SG2038/D	Application Summary - Home Appliances - Refrigerators and Freezers. Microcontrollers maximize appliance efficiency while supporting a variety of features in refrigerators and freezers.	pdf	77	0	6/24/2002	
SG2040/D	Home Appliances Washing Machines can use as many as three microcontrollers adding intelligence for increased functionality and user control.	pdf	91	0	6/24/2002	
SG2044/D	Application Summary - Home Appliances. Dryers. New dryer features make this application more energy efficient and better able to meet consumer demands for improved control.	pdf	90	0	6/24/2002	

Users Guide

ID	Name	Format	Size K	Rev #	Date Last Modified	Order Availability
CDSWHC08QS	CodeWarrior™ Development Studio for 68HC08 Quick Start Guide	pdf	2847	2.1	9/20/2002	-

[top]

68HC908AB32 Development Tools/Boards

ID	Name	Vendor ID	Order Availability
M68MULTILINK08	MON08 Multilink	METROWERKS	
M68CYCLONE08	MON08 Cyclone	METROWERKS	
M68EML08AB32	Emulation Module	MOTOROLA	

KITMMDS08AB32	Modular Development System (MMDS) Kits	MOTOROLA	
KITMMEVS08AB32	Modular Evaluation System (MMEVS)	MOTOROLA	
M68ICS08AB	M68ICS08AB Development Tool Kit	MOTOROLA	
CDCWSEHC08	CodeWarrior Development Studio for 68HC08 Special Edition	METROWERKS	-
CWHC08PRO	CodeWarrior Development Studio for Motorola HC08 Microcontrollers Professional Edition	METROWERKS	
CWHC08STD	CodeWarrior Development Studio for Motorola HC08 Microcontrollers Standard Edition	METROWERKS	

[top]

Design Tools

Software

ID	Name	Vendor ID	Format	Size K	Rev #
HC08DRIVEWAYSW	Aisys Driveway for 68HC08	AISYS	exe	19653	2.3

Software/Application Software/Code Examples

ID	Name	Vendor ID	Format	Size K	Rev #
HC08DELAYSW	HC08 Software Example: Subroutine that delays for a whole number of milliseconds	MOTOROLA	zip	2	-
HC08EXSW	HC08 Software Example: Library containing software examples in assembly for 68HC08	MOTOROLA	zip	14	-

[top]

Orderable Parts Information

PartNumber	Package Info	Life Cycle Description (code)	Remarks	Budgetary Price QTY 1000+ ($US)	Order Availability
KMC908AB32CFU	20x20 mm Plastic Quad Flat Pack (QFP)	PRODUCT RAPID GROWTH(2)	-40 to +85 C	$5.95	
KMC908AB32MFU	20x20 mm Plastic Quad Flat Pack (QFP)	PRODUCT RAPID GROWTH(2)	-40 to +125 C	$6.54	

KMC908AB32VFU	20x20 mm Plastic Quad Flat Pack (QFP)	PRODUCT RAPID GROWTH(2)	-40 to +105 C	-	-
MC68HC908AB32CFU	20x20 mm Plastic Quad Flat Pack (QFP)	PRODUCT RAPID GROWTH(2)	-40 to +85 C	$5.95	
MC68HC908AB32MFU	20x20 mm Plastic Quad Flat Pack (QFP)	PRODUCT RAPID GROWTH(2)	-40 to +125 C	$6.54	
MC68HC908AB32VFU	20x20 mm Plastic Quad Flat Pack (QFP)	PRODUCT RAPID GROWTH(2)	-40 to +105 C	$6.24	
MCHC908AB32CFUR2	20x20 mm Plastic Quad Flat Pack (QFP)	PRODUCT RAPID GROWTH(2)	-40 to +85 C	$6.09	
MCHC908AB32MFUR2	20x20 mm Plastic Quad Flat Pack (QFP)	PRODUCT RAPID GROWTH(2)	-40 to +125 C	$6.69	
MCHC908AB32VFUR2	20x20 mm Plastic Quad Flat Pack (QFP)	PRODUCT RAPID GROWTH(2)	-40 to +125 C	$6.40	
KMC908AB32CPB	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
KMC908AB32MPB	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
KMC908AB32VPB	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MCHC908AB32VPBR2	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MCHC908AB32MPBR2	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MCHC908AB32CPBR2	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MC68HC908AB32CPB	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MC68HC908AB32MPB	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MC68HC908AB32VPB	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MC908AB32MFU	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-

MC908AB32MFUR2	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MC908AB32VFU	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MC908AB32VFUR2	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MC908AB32CFU	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-
MC908AB32CFUR2	-	PRODUCT NEWLY INTRO'D/RAMP-UP(1)	-	-	-

[top]

Motorola Home | Semiconductors | Login | Support | Contact Us | Site Map
Products | Documentation | Tools | Design Resources | Applications

68HC908AB32 : Microcontroller

The MC68HC908AB32 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). The M68HC08 Family is based on the customer-specified integrated circuit (CSIC) design strategy. All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

Block Diagram

68HC908AB32 Features

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 families
- Memory map and pin functions compatible with MC68HC08AB32 and MC68HC08AB16
- 8MHz internal bus frequency at 85°C
- 32K-bytes user program FLASH memory with security feature
- 512 bytes of on-chip EEPROM with security feature
- 1K byte of on-chip RAM
- Clock generator module (CGM)
- Two 16-bit timer interface modules (TIMA and TIMB) with selectable input capture, output compare, and PWM capability on each channel
- Periodic Interrupt Timer (PIT)
- Clock Generator Module (CGM)

Return to Top

Page Contents:
- Features
- Documentation
- Reference Designs
- Tools
- Rich Media
- Orderable Parts
- Related Links

Other Info:
- FAQs
- 3rd Party Design Help
- Training
- 3rd Party Tool Vendors

68HC908AB32 Documentation

Documentation

Application Note

ID	Name	Vendor ID	Format	Size K	Rev #	Date Last Modified	Order Availability
AN1050_D	Designing for Electromagnetic Compatibility (EMC) with HCMOS Microcontrollers	MOTOROLA	pdf	82	0	1/01/2000	-
AN1218/D	HC05 to HC08 Optimization	MOTOROLA	pdf	347	2	1/01/1993	ORDER
AN1219/D	M68HC08 Integer Math Routines	MOTOROLA	pdf	177	1	1/01/1997	ORDER
AN1219SW	Software Files for AN1219 zipped	MOTOROLA	zip	77	0	1/01/1995	-
AN1221/D	Hamming Error Control Coding Techniques with the HC08 MCU	MOTOROLA	pdf	63	0	1/01/1993	ORDER
AN1221SW	Software Files for AN1221 zipped	MOTOROLA	zip	55	0	1/01/1995	-
AN1222/D	Arithmetic Waveform Synthesis with the HC05/08 MCUs	MOTOROLA	pdf	24	0	1/01/1993	ORDER

AN1222SW	Software Files for AN1222 zipped	MOTOROLA	zip	20	0	1/01/1995	-
AN1259/D	System Design and Layout Techniques for Noise Reduction in MCU-Based Systems	MOTOROLA	pdf	78	0	1/01/1995	ORDER
AN1263/D	Designing for Electromagnetic Compatibility with Single-Chip Microcontrollers	MOTOROLA	pdf	104	0	1/01/1995	ORDER
AN1274/D	HC08 SCI Operation with Various Input Clocks	MOTOROLA	pdf	47	0	1/01/1996	ORDER
AN1516/D	Liquid Level Control Using a Motorola Pressure Sensor	MOTOROLA	pdf	77	2	1/24/2003	ORDER
AN1705/D	Noise Reduction Techniques for Microcontroller-Based Systems	MOTOROLA	pdf	67	0	1/01/1999	ORDER
AN1744/D	Resetting Microcontrollers During Power Transitions	MOTOROLA	pdf	80	0	1/01/1998	ORDER
AN1752/D	Data Structures for 8-Bit Microcontrollers	MOTOROLA	pdf	213	1	5/07/2001	ORDER
AN1771/D	Precision Sine-Wave Tone Synthesis Using 8-Bit MCUs	MOTOROLA	pdf	250	0	1/01/1998	ORDER
AN1775/D	Expanding Digital Input with an A/D Converter	MOTOROLA	pdf	86	1	1/01/1998	ORDER
AN1783/D	Determining MCU Oscillator Start-Up Parameters	MOTOROLA	pdf	48	1	1/01/1999	ORDER
AN1818/D	Software SCI Routines with the 16-Bit Timer Module	MOTOROLA	pdf	84	0	1/01/1999	ORDER
AN1820/D	Software I2C Communications	MOTOROLA	pdf	55	0	1/01/1999	ORDER
AN1820SW	Software files for AN1820 zipped	MOTOROLA	zip	2	0	1/01/1998	-
AN1837/D	Non-Volatile Memory Technology Overview	MOTOROLA	pdf	116	0	3/27/2000	ORDER
AN1853/D	Embedding Microcontrollers in Domestic Refrigeration Appliances	MOTOROLA	pdf	221	0	6/22/2000	ORDER
AN2093/D	Creating Efficient C Code for the MC68HC08	MOTOROLA	pdf	36	0	1/01/2000	ORDER
AN2103/D	Local Interconnect Network (LIN) Demonstration	MOTOROLA	pdf	953	0	12/01/2000	ORDER
AN2120/D	Connecting an M68HC08 Family Microcontroller to an Internet Service Provider (ISP) Using the Point-to-Point Protocol (PPP)	MOTOROLA	pdf	741	0	5/20/2001	ORDER
AN2120SW	Software for AN2120, zip format	MOTOROLA	zip	31	1.0	7/31/2002	-
AN2149/D	Compressor Induction Motor Stall and Rotation Detection using Microcontrollers	MOTOROLA	pdf	127	0	5/30/2001	ORDER
AN2159/D	Digital Direct Current Ignition System Using HC08 Microcontrollers	MOTOROLA	pdf	129	0	11/20/2001	ORDER
AN2159SW	AN2159SW	MOTOROLA	zip	182	1	3/08/2002	-
AN2295	Developer's Serial Bootloader for M68HC08	MOTOROLA	pdf	738	4	10/29/2003	ORDER
AN2295SW	Software for AN2295	MOTOROLA	zip	725	4.0	10/21/2003	-
AN2321/D	Designing for Board Level Electromagnetic Compatibility	MOTOROLA	pdf	1628	0	8/15/2002	ORDER
AN2342	Opto Isolation Circuits For In Circuit Debugging of 68HC9(S)12 and 68HC908 Microcontrollers	MOTOROLA	pdf	155	0	9/25/2002	ORDER
AN2438/D	ADC Definitions and Specifications	MOTOROLA	pdf	297	0	2/21/2003	ORDER

AN2504	On-Chip FLASH Programming API for CodeWarrior	MOTOROLA	pdf	530	0	10/15/2003	ORDER
AN2504SW	Software files for application note AN2504	MOTOROLA	zip	59	0	10/21/2003	-
AN4006/D	Digital Captive Discharge Ignition System Using HC05/HC08 8-Bit Microcontrollers	MOTOROLA	pdf	61	0	3/27/2000	ORDER

Brochure

ID	Name	Vendor ID	Format	Size K	Rev #	Date Last Modified	Order Availability
BR1822	Embedded Flash MCU Overview	MOTOROLA	pdf	174	-	-	ORDER
BR1853/D	BR1853 68HC908AB32 Brochure	MOTOROLA	pdf	154	0	8/07/2000	-
BR68HC08FAMAM/D	68HC08 Family: High Performance and Flexibility	MOTOROLA	pdf	57	2	5/21/2003	ORDER
FLYREMBEDFLASH/D	Embedded Flash: Changing the Technology World for the Better	MOTOROLA	pdf	68	2	5/21/2003	ORDER

Data Sheets

ID	Name	Vendor ID	Format	Size K	Rev #	Date Last Modified	Order Availability
MC68HC908AB32/D	MC68HC908AB32 HCMOS Microcontroller Unit	MOTOROLA	pdf	3706	1.0	8/26/2000	ORDER

Engineering Bulletin

ID	Name	Vendor ID	Format	Size K	Rev #	Date Last Modified	Order Availability
EB349/D	RAM Data Retention Considerations for Motorola Microcontrollers	MOTOROLA	pdf	45	1	6/22/2000	ORDER
EB387/D	MMDS and MMEVS Changes for MC68HC908ASxxA, MC68HC908AZxxA, and MC68HC908AB32	MOTOROLA	pdf	61	0	7/16/2001	ORDER
EB389/D	TOF Consideration when Measuring a Long Input Capture Event	MOTOROLA	pdf	55	1	4/15/2002	ORDER
EB390/D	Porting the AN2120/D UDP/IP Code to the Avnet Evaluation Board	MOTOROLA	pdf	1501	0	5/09/2002	ORDER
EB394/D	Initializing the MC68HC908AB32 Output Compare Feature	MOTOROLA	pdf	381	0	11/18/2002	ORDER
EB396/D	Use of OSC2/XTAL as a Clock Output on Motorola Microcontrollers	MOTOROLA	pdf	49	0	6/19/2002	ORDER
EB398	Techniques to Protect MCU Applications Against Malfunction Due to Code Run-Away	MOTOROLA	pdf	0	0	8/13/2002	ORDER
EB608/D	Interrupt Handling Considerations When Modifying EEPROM on HC08 Microcontrollers	MOTOROLA	pdf	96	0	8/14/2002	ORDER
EB612/D	Differences Between the HC08AB16A and HC908AB32	MOTOROLA	pdf	68	0	11/15/2002	ORDER

Fact Sheets

ID	Name	Vendor ID	Format	Size K	Rev #	Date Last Modified	Order Availability
68HC908AB32PB/D	8-bit Microcontroller	MOTOROLA	pdf	47	2	5/21/2003	ORDER
CWDEVSTUDFACTHC08	Development Studio	MOTOROLA	pdf	48	2	5/13/2002	-

Product Change Notices

ID	Name	Vendor ID	Format	Size K	Rev #	Date Last Modified	Order Availability
PCN7855	14X14 QFP ASSY MOVE FROM SHC TO KLM, PT 1 OF 2	MOTOROLA	htm	24	0	8/05/2002	-
PCN7856	14X14 QFP ASSY MOVE FROM SHC TO KLM, PT 2 OF 2	MOTOROLA	htm	22	0	8/05/2002	-
PCN8103	10X10 LQFP ASSY MOVE FROM SHC TO BAT3	MOTOROLA	htm	16	0	10/08/2002	-
PCN8341	MC68HC908AB32CFU 64QFP TEST MOVE TO OHT	MOTOROLA	htm	5	0	12/04/2002	-
PCN8627	MC68HC908AB32 SINGLE TEST FLOW	MOTOROLA	htm	4	0	3/21/2003	-
PCN8886	REDESIGN OF NH1414 2.2 14X14 MQFP TRAY	MOTOROLA	htm	36	0	5/19/2003	-

Reference Manual

ID	Name	Vendor ID	Format	Size K	Rev #	Date Last Modified	Order Availability
ADCRM/AD	Analog-to-Digital Reference Manual	MOTOROLA	pdf	231	0	1/01/1996	ORDER
CPU08RM/AD	CPU08RM Central Processor Unit Reference Manual	MOTOROLA	pdf	2666	3	4/03/2002	ORDER
DRM001/D	Passive Infrared (PIR) Intruder Detection Using the MC68HC908JK1/3, Incorporating Remote Control Adjustment Using the MC68HC908GP32	MOTOROLA	pdf	2504	0	2/20/2001	ORDER
DRM002/D	USB08 Universal Serial Bus Evaluation Board Using the MC68HC908JB8 Designer Reference Manual	MOTOROLA	pdf	1845	0	4/12/2001	ORDER
TIM08RM/AD	TIM08 Timer Interface Module Reference Manual	MOTOROLA	pdf	771	1.0	1/10/1996	ORDER

Selector Guide

ID	Name	Vendor ID	Format	Size K	Rev #	Date Last Modified	Order Availability
SG1002	Analog Selector Guide - Quarter 4, 2003	MOTOROLA	pdf	579	0	10/24/2003	ORDER
SG1006	Microcontrollers Selector Guide - Quarter 4, 2003	MOTOROLA	pdf	826	0	10/24/2003	ORDER
SG1010	Sensors Selector Guide - Quarter 4, 2003	MOTOROLA	pdf	219	0	10/24/2003	ORDER
SG1011	Software and Development Tools Selector Guide - Quarter 4, 2003	MOTOROLA	pdf	287	0	10/24/2003	ORDER
SG2000CR	Application Selector Guide Index and Cross-Reference.	MOTOROLA	pdf	95	3	11/11/2003	ORDER
SG2036	Application Summary Home Appliances - Cooking Products. Microcontrollers provide intelligent management programs delivering high precision control over the cooking process.	MOTOROLA	pdf	0	1	12/16/2002	ORDER
SG2037	Application Selector Guide - Home Appliances DISHWASHERS	MOTOROLA	pdf	0	2	6/17/2003	ORDER
SG2038	Application Summary - Home Appliances - Refrigerators and Freezers. Microcontrollers maximize appliance efficiency while supporting a variety of features in refrigerators and freezers.	MOTOROLA	pdf	0	1	12/16/2002	ORDER
SG2039	Application Selector Guide - Vacuum Cleaners Vacuum Cleaners	MOTOROLA	pdf	0	0	6/17/2003	ORDER
SG2040	Application Selector Guide - Home Applicances WASHING MACHINES	MOTOROLA	pdf	0	2	6/17/2003	ORDER

SG2044	Application Summary - Home Appliances. Dryers. New dryer features make this application more energy efficient and better able to meet consumer demands for improved control.	MOTOROLA	pdf	0	1	12/16/2002	ORDER

Return to Top

68HC908AB32 Reference Designs

Reference Designs

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
RD68HC908PLKNX	Konnex PL132 Over Power Line Based On Motorola's M68HC08 Family	MOTOROLA	-	-	-	-

Return to Top

68HC908AB32 Tools

Hardware Tools

Emulators/Probes/Wigglers

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
AX-6811	AX-6811	HITEX	-	-	-	-
IC10000	iC1000 PowerEmulator	ISYS	-	-	-	-
IC20000	iC2000 PowerEmulator	ISYS	-	-	-	-
IC40000	iC4000 ActiveEmulator	ISYS	-	-	-	-
INDART-HC08/D	In-Circuit, Real-Time Debugger/Programmer for Motorola 68HC08 Family (USB)	SOFTEC	-	-	-	-

Evaluation/Development Boards and Systems

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
KITMMDS08AB32	Modular Development System (MMDS) Kits	MOTOROLA	-	-	-	BUY
KITMMEVS08AB32	Modular Evaluation System (MMEVS)	MOTOROLA	-	-	-	BUY
M68CBL05C	Low-noise Flex Cable	MOTOROLA	-	-	-	BUY
M68EML08AB32	Emulation Module	MOTOROLA	-	-	-	BUY
M68ICS08AB	M68ICS08AB Development Tool Kit	MOTOROLA	-	-	-	BUY
M68CYCLONE08	MON08 Cyclone	METROWERKS	-	-	-	BUY
M68MULTILINK08	MON08 Multilink	METROWERKS	-	-	-	BUY

Programmers

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
MP8011A	Gang Programmer Base Unit	SOFTEC	-	-	-	-
AP520	Automated Programming System	SYSGEN	-	-	-	-
POWERLAB	Universal Programmer	SYSGEN	-	-	-	-
T9600	High-speed universal gang programmer	SYSGEN	-	-	-	-

Software

Application Software

Code Examples

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
HC08DELAYSW	HC08 Software Example: Subroutine that delays for a whole number of milliseconds	MOTOROLA	zip	2	-	-
HC08EXSW	HC08 Software Example: Library containing software examples in assembly for 68HC08	MOTOROLA	zip	14	-	-

Operating Systems

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
CMX-TINY+	CMX-Tiny+	CMX	-	-	-	-

Software Tools

Assemblers

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
ADX-08	ADX-08 Macro Assembler-Linker and IDE	AVOCET	-	-	-	-
AX6808	AX6808 relocatable and absolute macro assembler for HC08 and HCS08	COSMIC	-	-	-	-

Compilers

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
CX6808S	CX6808 C Cross Compiler for HC08 and HCS08	COSMIC	-	-	-	-
ICC08	ICC08 V6 STD	IMAGE	-	-	-	-

Debuggers

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
ZAP 6808 MON08	ZAP 6808 MON08 Debugger and Flash Programmer ZAP 6808 MON08 debugger uses the 68HC08's on-chip monitor interface to provide a real-time ANSI C and assembly source level debugger including FLASH programming, FLASH security and hardware breakpoint support.	MOTOROLA	-	-	-	-
ZAP 6808 MMDS	ZAP 6808 MMDS Debugger	COSMIC	-	-	-	-
ZAP 6808 SIM	ZAP 6808 Simulator Debugger	COSMIC	-	-	-	-
AX-6811	AX-6811	HITEX	-	-	-	-
NOICE08	NoICE08	IMAGE	-	-	-	-

IDE (Integrated Development Environment)

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
CDCWSEHC08	CodeWarrior Development Studio™ for HC(S)08 Special Edition	METROWERKS	-	-	-	-
CWHC08PRO	CodeWarrior Development Studio for Motorola HC08 Microcontrollers Professional Edition	METROWERKS	-	-	-	BUY
CWHC08STD	CodeWarrior Development Studio for Motorola HC08 Microcontrollers Standard Edition	METROWERKS	-	-	-	BUY
CX6808LT4	HC08 Development Tool Suite 4K Lite (FREE)	COSMIC	-	-	-	-
IDEA08	IDEA08 integrated development environment for HC08 and HCS08	COSMIC	-	-	-	-
IC-SW-OPR	winIDEA	ISYS	-	-	-	-

Performance and Testing

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
AX-6811	AX-6811	HITEX	-	-	-	-

Return to Top

Rich Media

Rich Media

Webcast

ID	Name	Vendor ID	Format	Size K	Rev #	Order Availability
RMWC_CODEWARRIOR	CodeWarrior Development Tools for 68HC08 and HCS12 Microcontrollers. Listen to our webcast for an overview of some of the challenges that developers face and an explanation of the CodeWarrior tools that help to address these challenges.	MOTOROLA	html	4	0.0	-
RMWC_QFAMILY	8-bit Microcontroller Overview and Q-Family of Flash Microcontrollers Listen to our companion webcasts to learn about Motorola's recent 8-bit products and services-especially the HC08 Q-Family-that offer maximum design flexibility while helping you get to market fast.	MOTOROLA	htm	5	1.1	-

Return to Top

Orderable Parts Information

PartNumber	Package Info	Tape and Reel	Life Cycle Description (code)	Budgetary Price QTY 1000+ ($US)	Additional Info	Order Availability
KMC908AB32CFU	QFP 64 14*14*2.2P0.8	No	PRODUCT RAPID GROWTH(2)	$5.95	more	BUY
KMC908AB32CPB	LQFP 64 10*10*1.4P0.5	No	PRODUCT RAPID GROWTH(2)	-	more	-
KMC908AB32MFU	QFP 64 14*14*2.2P0.8	No	PRODUCT RAPID GROWTH(2)	$6.54	more	BUY
KMC908AB32MPB	LQFP 64 10*10*1.4P0.5	No	PRODUCT RAPID GROWTH(2)	-	more	-
KMC908AB32VFU	QFP 64 14*14*2.2P0.8	No	PRODUCT RAPID GROWTH(2)	-	more	-
KMC908AB32VPB	LQFP 64 10*10*1.4P0.5	No	PRODUCT RAPID GROWTH(2)	-	more	-
MC68HC908AB32CFU	QFP 64 14*14*2.2P0.8	No	PRODUCT RAPID GROWTH(2)	$5.95	more	BUY
MC68HC908AB32MFU	QFP 64 14*14*2.2P0.8	No	PRODUCT RAPID GROWTH(2)	$6.54	more	BUY
MC68HC908AB32MPB	LQFP 64 10*10*1.4P0.5	No	PRODUCT RAPID GROWTH(2)	-	more	BUY

MC68HC908AB32VFU	QFP 64 14*14*2.2P0.8	No	PRODUCT RAPID GROWTH(2)	$6.24	more	BUY
MC68HC908AB32VPB	LQFP 64 10*10*1.4P0.5	No	PRODUCT RAPID GROWTH(2)	-	more	-
MC908AB32CFU	QFP 64 14*14*2.2P0.8	No	PRODUCT RAPID GROWTH(2)	-	more	-
MC908AB32CFUR2	QFP 64 14*14*2.2P0.8	Yes	PRODUCT RAPID GROWTH(2)	-	more	-
MC908AB32MFU	QFP 64 14*14*2.2P0.8	No	PRODUCT RAPID GROWTH(2)	-	more	-
MC908AB32MFUR2	QFP 64 14*14*2.2P0.8	Yes	PRODUCT RAPID GROWTH(2)	-	more	-
MC908AB32VFU	QFP 64 14*14*2.2P0.8	No	PRODUCT RAPID GROWTH(2)	-	more	-
MC908AB32VFUR2	QFP 64 14*14*2.2P0.8	Yes	PRODUCT RAPID GROWTH(2)	-	more	-
MCHC908AB32CFUR2	QFP 64 14*14*2.2P0.8	Yes	PRODUCT RAPID GROWTH(2)	$6.09	more	BUY
MCHC908AB32CPBR2	LQFP 64 10*10*1.4P0.5	Yes	PRODUCT RAPID GROWTH(2)	-	more	-
MCHC908AB32MFUR2	QFP 64 14*14*2.2P0.8	Yes	PRODUCT RAPID GROWTH(2)	$6.69	more	BUY
MCHC908AB32MPBR2	LQFP 64 10*10*1.4P0.5	Yes	PRODUCT RAPID GROWTH(2)	-	more	-
MCHC908AB32VFUR2	QFP 64 14*14*2.2P0.8	Yes	PRODUCT RAPID GROWTH(2)	$6.40	more	BUY
MCHC908AB32VPBR2	LQFP 64 10*10*1.4P0.5	Yes	PRODUCT RAPID GROWTH(2)	-	more	-

NOTE: Are you looking for an obsolete orderable part? Click **HERE** to check our distributors' inventory.

Return to Top

Related Links

- Automotive
- Microcontrollers
- Motor Control
- Sensors

Return to Top