MC68HC08AZ32A

Data Sheet

M68HC08
Microcontrollers

MC68HC08AZ32A
Rev. 2
07/2005

freescale.com

freescale™
semiconductor

MC68HC08AZ32A

Data Sheet

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://freescale.com/

Refer to the Revision History for a summary of changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

List of Chapters

Table of Contents

Chapter 1
General Description

Chapter 2
Memory

Chapter 3
Analog-to-Digital Converter (ADC) Module

Chapter 4
Clock Generator Module (CGM)

Chapter 5
Computer Operating Properly (COP) Module

Chapter 6
Central Processor Unit (CPU)

Chapter 7
External Interrupt (IRQ) Module

Chapter 8
Keyboard Interrupt (KBD) Module

Chapter 9
Low-Voltage Inhibit (LVI) Module)

Chapter 10
Mask Options

Chapter 11
MSCAN08 Controller (MSCAN08)

Chapter 12
Programmable Interrupt Timer (PIT)

Chapter 13
Input/Output (I/O) Ports

Chapter 14
Serial Communications Interface (SCI)

Chapter 15
System Integration Module (SIM)

Chapter 16
Serial Peripheral Interface (SPI)

Chapter 17
Timer Interface Module A (TIMA)

Chapter 18
Timer Interface Module B (TIMB)

Chapter 19
Development Support

Chapter 20
Electrical Specifications

Chapter 21
Ordering Information and Mechanical Specifications

Chapter 1
General Description

1.1 Introduction

The MC68HC08AZ32A is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). All MCUs in the Family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

1.2 Features

Features include the following:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- 8.4-MHz internal bus frequency at 125°C
- MSCAN08 controller (scalable CAN) (implementing CAN 2.0b protocol as defined in BOSCH specification September, 1991)
- Available in 64-pin quad flat pack (QFP)
- 32,256 bytes user read-only memory (ROM)
- User ROM data security[1]
- 512 bytes of on-chip electrically erasable programmable read-only memory (EEPROM) with security feature
- 1 Kbyte of on-chip random-access memory (RAM)
- Serial peripheral interface (SPI) module
- Serial communications interface (SCI) module
- 16-bit timer interface module (TIMA) with six input capture/output compare channels
- 16-bit timer interface module (TIMB) with two input capture/output compare channels
- Programmable interrupt timer (PIT)
- Clock generator module (CGM)
- 8-bit, 15-channel analog-to-digital convertor (ADC)
- 5-bit keyboard interrupt module (KBI)

1. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the ROM/EEPROM data difficult for unauthorized users.

- System protection features:
  - Computer operating properly (COP) with optional reset
  - Low-voltage detection with optional reset
  - Illegal opcode detection with optional reset
  - Illegal address detection with optional reset
- Low-power design (fully static with stop and wait modes)
- Master reset pin and power-on reset (POR)

Features of the CPU08 include:

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

Figure 1-1 shows the structure of the MC68HC08AZ32A.

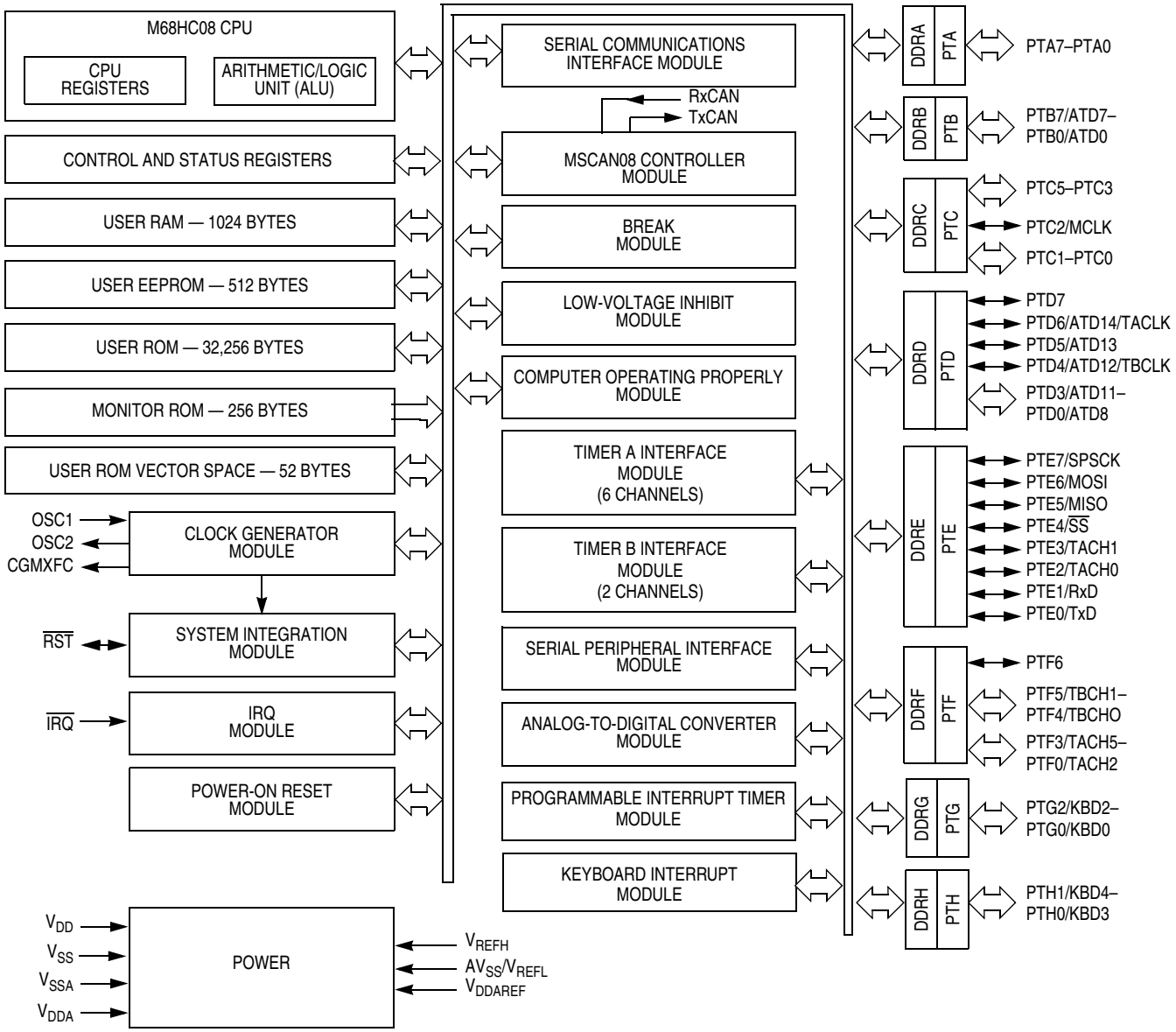


Figure 1-1. MC68HC08AZ32A MCU Block Diagram

1.3 Pin Assignments

Figure 1-2 shows the 64 QFP pin assignments.

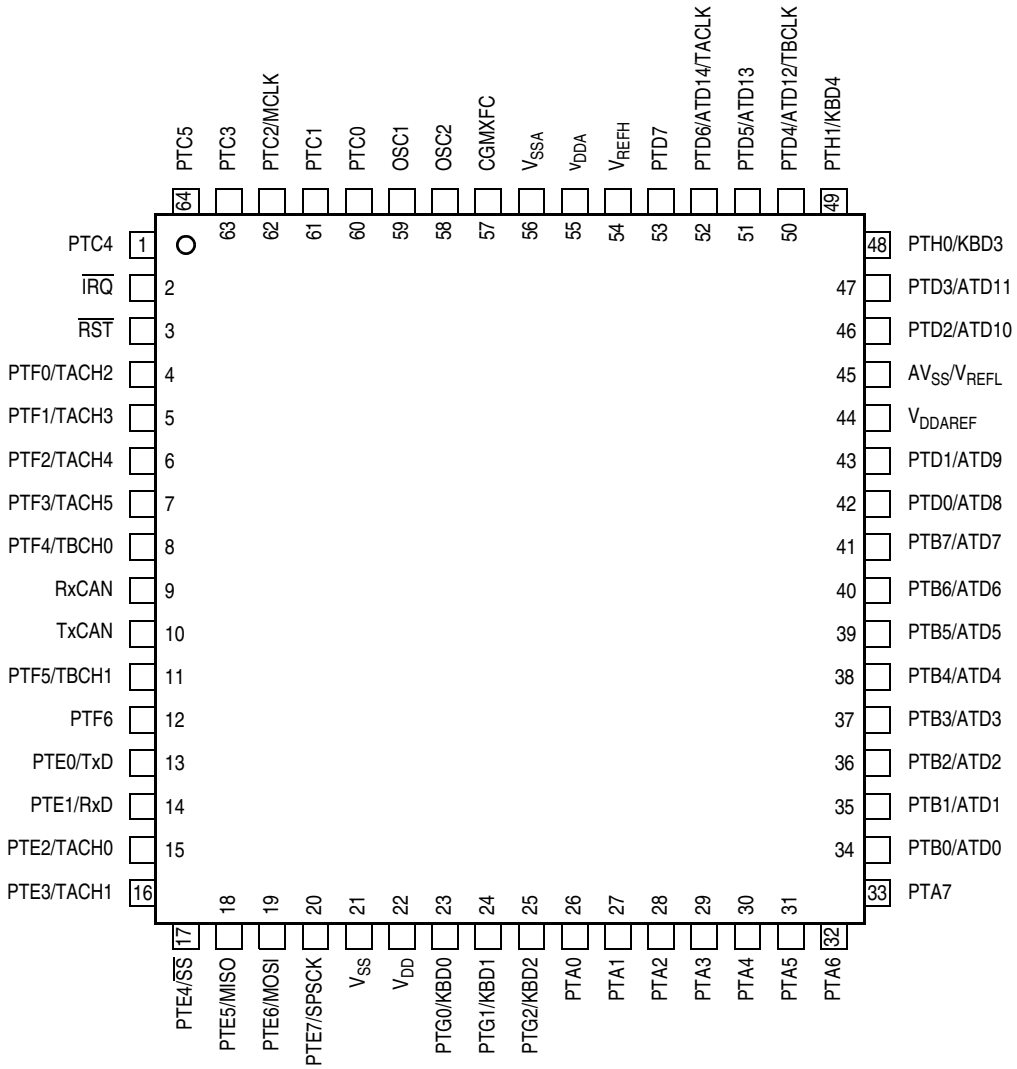
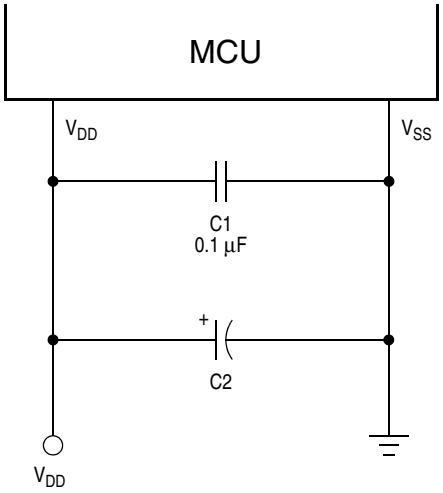


Figure 1-2. 64-Pin QFP Pin Assignments

1.3.1 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 SPI module. See Chapter 16 Serial Peripheral Interface (SPI).

NOTE

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

1.3.2 Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. See Chapter 4 Clock Generator Module (CGM).

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

A 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. See Chapter 15 System Integration Module (SIM).

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

$\overline{IRQ}$ is an asynchronous external interrupt pin. See Chapter 7 External Interrupt (IRQ) Module.

1.3.5 Analog Power Supply Pin (V_{DDA})

V_{DDA} is the power supply pin for the analog portion of the CGM. See Chapter 4 Clock Generator Module (CGM).

1.3.6 Analog Ground Pin (V_{SSA})

V_{SSA} is the ground connection for the analog portion of the CGM. See Chapter 4 Clock Generator Module (CGM).

1.3.7 ADC Analog Ground Pin (AV_{SS}/V_{REFL})

The AV_{SS}/V_{REFL} pin provides both the analog ground connection and the reference low voltage for the ADC. See Chapter 3 Analog-to-Digital Converter (ADC) Module.

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

V_{REFH} provides the reference high voltage for the ADC. See Chapter 3 Analog-to-Digital Converter (ADC) Module.

1.3.9 ADC Analog Power Supply Pin (V_{DDAREF})

V_{DDAREF} is the power supply pin for the analog portion of the ADC. See Chapter 3 Analog-to-Digital Converter (ADC) Module.

1.3.10 External Filter Capacitor Pin (CGMXFC)

CGMXFC is an external filter capacitor connection for the CGM. See Chapter 4 Clock Generator Module (CGM).

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

PTA7–PTA0 are general-purpose bidirectional I/O port pins. See Chapter 13 Input/Output (I/O) Ports.

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

Port B is an 8-bit special function port that shares all eight pins with the ADC. See Chapter 3 Analog-to-Digital Converter (ADC) Module and Chapter 13 Input/Output (I/O) Ports.

1.3.13 Port C I/O Pins (PTC5–PTC0)

PTC5–PTC3 and PTC1–PTC0 are general-purpose bidirectional I/O port pins. PTC2/MCLK is a special function port that shares its pin with the system clock. See Chapter 13 Input/Output (I/O) Ports.

1.3.14 Port D I/O Pins (PTD7–PTD0/ATD8)

Port D is an 8-bit special-function port that shares seven of its pins with the ADC, one of its pins with TIMA, and one more of its pins with TIMB. See Chapter 17 Timer Interface Module A (TIMA), Chapter 18 Timer Interface Module B (TIMB), Chapter 3 Analog-to-Digital Converter (ADC) Module, and Chapter 13 Input/Output (I/O) Ports.

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

Port E is an 8-bit special function port that shares two of its pins with TIMA, four of its pins with the SPI, and two of its pins with the SCI. See Chapter 14 Serial Communications Interface (SCI), Chapter 16 Serial Peripheral Interface (SPI), Chapter 17 Timer Interface Module A (TIMA), and Chapter 13 Input/Output (I/O) Ports.

1.3.16 Port F I/O Pins (PTF6–PTF0/TACH2)

Port F is a 7-bit special function port that shares two of its pins with TIMB. Four of its pins are shared with TIMA. See Chapter 17 Timer Interface Module A (TIMA), Chapter 18 Timer Interface Module B (TIMB), and Chapter 13 Input/Output (I/O) Ports.

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

Port G is a 3-bit special function port that shares all of its pins with the KBD. See Chapter 8 Keyboard Interrupt (KBD) Module and Chapter 13 Input/Output (I/O) Ports.

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

Port H is a 2-bit special-function port that shares all of its pins with the KBD. See Chapter 8 Keyboard Interrupt (KBD) Module and Chapter 13 Input/Output (I/O) Ports.

1.3.19 CAN Transmit Pin (TxCAN)

TxCAN is the digital output from the MSCAN08 module. See Chapter 11 MSCAN08 Controller (MSCAN08).

1.3.20 CAN Receive Pin (RxCAN)

RxCAN is the digital input to the MSCAN08 module. See Chapter 11 MSCAN08 Controller (MSCAN08).

1.4 Clocks

Details of the clock connections to each of the modules on the MC68HC08AZ32A are shown in Table 1-3. A short description of each clock source is also given in Table 1-2.

Table 1-1. External Pins Summary

Pin Name	Function	Driver Type	Hysteresis[1]	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–PTC0	General purpose I/O	Dual state	No	Input (Hi-Z)
PTD7	General purpose I/O	Dual state	No	Input (Hi-Z)
PTD6/ATD14/TACLK	General purpose I/O/ADC channel/ timer A external input clock	Dual state	Yes (only for timer A function)	Input (Hi-Z)
PTD5/ATD13	General purpose I/O/ADC channel	Dual state	No	Input (Hi-Z)
PTD4/ATD12/TBCLK	General purpose I/O/ADC channel/ timer B external input clock	Dual state	Yes (only for timer B function)	Input (Hi-Z)
PTD3/ATD11–PTD0/ATD8	General purpose I/O/ADC channel	Dual state	No	Input (Hi-Z)
PTE7/SPSCK	General purpose I/O/SPI clock	Dual state (open drain)	Yes (only for SPI function)	Input (Hi-Z)
PTE6/MOSI	General purpose I/O/SPI data path	Dual state (open drain)	Yes (only for SPI function)	Input (Hi-Z)
PTE5/MISO	General purpose I/O/SPI data path	Dual state (open drain)	Yes (only for SPI function)	Input (Hi-Z)
PTE4/$\overline{SS}$	General purpose I/O/SPI slave select	Dual state	Yes (only for SPI function)	Input (Hi-Z)
PTE3/TACH1	General purpose I/O/timer A channel 1	Dual state	Yes (only for timer A function)	Input (Hi-Z)
PTE2/TACH0	General purpose I/O/timer A channel 0	Dual state	Yes (only for timer A function)	Input (Hi-Z)
PTE1/RxD	General purpose I/O/SCI receive data	Dual state	Yes (only for SCI function)	Input (Hi-Z)
PTE0/TxD	General purpose I/O/SCI transmit data	Dual state	No	Input (Hi-Z)
PTF6	General purpose I/O	Dual state	No	Input (Hi-Z)
PTF5/TBCH1	General purpose I/O/timer B channel 1	Dual state	Yes (only for timer B function)	Input (Hi-Z)
PTF4/TBCH0	General purpose I/O/ timer B channel 0	Dual state	Yes (only for timer B function)	Input (Hi-Z)
PTF3/TACH5	General purpose I/O/timer A channel 5	Dual state	Yes (only for timer A function)	Input (Hi-Z)
PTF2/TACH4	General purpose I/O/timer A channel 4	Dual state	Yes (only for timer A function)	Input (Hi-Z)
PTF1/TACH3	General purpose I/O/timer A channel 3	Dual state	Yes (only for timer A function)	Input (Hi-Z)
PTF0/TACH2	General purpose I/O/timer A channel 2	Dual state	Yes (only for timer A function)	Input (Hi-Z)

— Continued on next page

Table 1-1. External Pins Summary (Continued)

Pin Name	Function	Driver Type	Hysteresis[1]	Reset State
PTG2/KBD2–PTG0/KBD0	General purpose I/O/keyboard wakeup pin	Dual state	Yes (only for KBD function)	Input (Hi-Z)
PTH1/KBD4–PTH0/KBD3	General purpose I/O/keyboard wakeup pin	Dual state	Yes (only for KBD function)	Input (Hi-Z)
V_{DD}	Logical chip power supply	NA	NA	NA
V_{SS}	Logical chip ground	NA	NA	NA
V_{DDA}	CGM analog power supply	NA	NA	NA
V_{SSA}	CGM analog ground	NA	NA	NA
V_{REFH}	ADC reference high voltage	NA	NA	NA
AV_{SS}/V_{REFL}	ADC GND and reference low 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 (Hi-Z)
$\overline{RST}$	Reset	Open drain	NA	Output low
RxCAN	MSCAN08 serial input	NA	YES	Input (Hi-Z)
TxCAN	MSCAN08 serial output	Output	NA	Output

1. Hysteresis is not 100% tested but is typically a minimum of 300 mV.

Table 1-2. Signal Name Conventions

Signal name	Description
CGMXCLK	Buffered version of OSC1 from CGM
CGMOUT	PLL-based or OSC1-based clock output from CGM
Bus clock	CGMOUT divided by two
SPSCK	SPI serial clock
TACLK	External clock Input for TIMA
TBCLK	External clock Input for TIMB

Table 1-3. Clock Source Summary

Module	Clock Source
ADC	CGMXCLK or bus clock
MSCAN08	CGMXCLK or CGMOUT
COP	CGMXCLK
CPU	Bus clock
EEPROM	CGMXCLK or bus clock
ROM	Bus clock
RAM	Bus clock
SPI	Bus clock/SPSCK
SCI	CGMXCLK
TIMA	Bus clock or PTD6/ATD14/TACLK
TIMB	Bus clock or PTD4/ATD12/TBCLK
PIT	Bus clock
KBI	Bus clock
SIM	CGMOUT and CGMXCLK
IRQ	Bus clock
BRK	Bus clock
LVI	Bus clock
CGM	OSC1 and OSC2

Chapter 2
Memory

2.1 Introduction

The CPU08 can address 64 Kbytes of memory space. The memory map includes:

- 1024 bytes of RAM
- 32,256 bytes of user ROM
- 512 bytes of EEPROM
- 52 bytes of user-defined vectors
- 256 bytes of monitor ROM

The following definitions apply to the memory map representation of reserved and unimplemented locations.

- **Reserved** — Accessing a reserved location can have unpredictable effects on MCU operation.
- **Unimplemented** — Accessing an unimplemented location can cause an illegal address reset (within the constraints as outlined in Chapter 15 System Integration Module (SIM)).

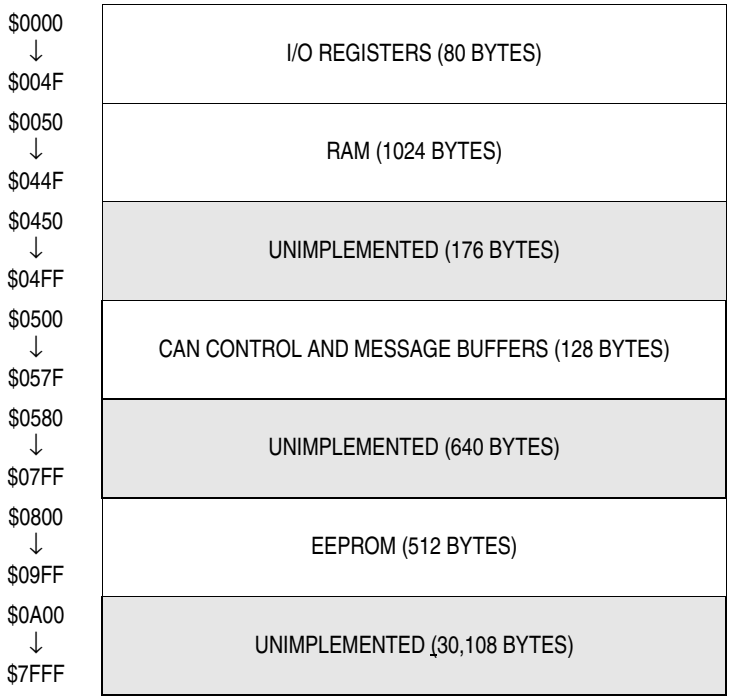


Figure 2-1. MC68HC08AZ32A Memory Map (Sheet 1 of 2)

$8000 ↓ $FDFF	ROM (32,256 BYTES)
$FE00	SIM BREAK STATUS REGISTER (SBSR)
$FE01	SIM RESET STATUS REGISTER (SRSR)
$FE02	RESERVED
$FE03	SIM BREAK FLAG CONTROL REGISTER (SBFCR)
$FE04 ↓ $FE08	RESERVED
$FE09	MORB
$FE0A	RESERVED
$FE0B	RESERVED
$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 EEDIVH NONVOLATILE REGISTER (EEDIVHNVR)
$FE11	EEPROM EEDIVL NONVOLATILE REGISTER (EEDIVLNVR)
$FE12 ↓ $FE19	RESERVED (8 BYTES)
$FE1A	EEPROM EEDIVH REGISTER (EEDIVH)
$FE1B	EEPROM EEDIVL REGISTER (EEDIVL)
$FE1C	EEPROM NONVOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION (EEACR)
$FE20 ↓ $FF1F	MONITOR ROM (256 BYTES)
$FF20 ↓ $FFBF	UNIMPLEMENTED (160 BYTES)
$FFC0 ↓ $FFCB	RESERVED (12 BYTES)
$FFCC ↓ $FFFF	VECTORS (52 BYTES)

Figure 2-1. MC68HC08AZ32A Memory Map (Sheet 2 of 2)

2.2 I/O Section

Addresses $0000–$004F, shown in Figure 2-2, contain the I/O data, status and control registers

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA) See page 155.	Read:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB) See page 157.	Read:	PTB7	PTB6	PTB25	PTB4	PTB3	PTB2	PTB1	PTB0
		Write:								
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC) See page 159.	Read:	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Write:								
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD) See page 161.	Read:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Write:								
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA) See page 155.	Read:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB) See page 157.	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC) See page 159.	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) See page 162.	Read:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE) See page 163.	Read:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Write:								
		Reset:	Unaffected by reset							
$0009	Port F Data Register (PTF) See page 165.	Read:	0	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
		Write:								
		Reset:	Unaffected by reset							
$000A	Port G Data Register (PTG) See page 167.	Read:	0	0	0	0	0	PTG2	PTG1	PTG0
		Write:								
		Reset:	Unaffected by reset							
$000B	Port H Data Register (PTH) See page 169.	Read:	0	0	0	0	0	0	PTH1	PTH0
		Write:								
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved

Figure 2-2. I/O Data, Status and Control Registers (Sheet 1 of 6)

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$000C	Data Direction Register E (DDRE) See page 164.	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) See page 166.	Read:	0	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000E	Data Direction Register G (DDRG) See page 168.	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) See page 170.	Read:	0	0	0	0	0	0	DDRH1	DDRH0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0010	SPI Control Register (SPCR) See page 230.	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) See page 231.	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	1	0	0	0
$0012	SPI Data Register (SPDR) See page 233.	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) See page 186.	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) See page 188.	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) See page 190.	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) See page 191.	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) See page 193.	Read:							BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	SCI Data Register (SCDR) See page 194.	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 2-2. I/O Data, Status and Control Registers (Sheet 2 of 6)

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0019	SCI Baud Rate Register (SCBR) See page 194.	Read:			SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001A	IRQ Status and Control Register (ISCR) See page 97.	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0
$001B	Keyboard Status/Control (KBSCR) See page 103.	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) See page 69.	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) See page 70.	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) See page 71.	Read:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
		Write:								
		Reset:	0	1	1	0	0	1	1	0
$001F	Mask Option Register A (MORA) See page 109.	Read:	LVISTOP	ROMSEC	LVIRST	LVIPWR	SSREC	COPRS	STOP	COPD
		Write:								
		Reset:	Unaffected by reset							
$0020	Timer A Status and Control Register (TASC) See page 247.	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	R			
		Reset:	0	0	1	0	0	0	0	0
$0021	Keyboard Interrupt Enable Register (KBIER) See page 104.	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) See page 248.	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) See page 248.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0024	Timer A Modulo Register High (TAMODH) See page 249.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	Timer A Modulo Register Low (TAMODL) See page 249.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1

= Unimplemented R = Reserved

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

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0026	Timer A Channel 0 Status and Control Register (TASC0) See page 247.	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) See page 249.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0028	Timer A Channel 0 Register Low (TACH0L) See page 249.	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) See page 247.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$002A	Timer A Channel 1 Register High (TACH1H) See page 249.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002B	Timer A Channel 1 Register Low (TACH1L) See page 249.	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) See page 247.	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) See page 249.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002E	Timer A Channel 2 Register Low (TACH2L) See page 249.	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) See page 247.	Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$0030	Timer A Channel 3 Register High (TACH3H) See page 249.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0031	Timer A Channel 3 Register Low (TACH3L) See page 249.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0032	Timer A Channel 4 Status and Control Register (TASC4) See page 247.	Read:	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	TOV4	CH4MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 2-2. I/O Data, Status and Control Registers (Sheet 4 of 6)

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0033	Timer A Channel 4 Register High (TACH4H) See page 249.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0034	Timer A Channel 4 Register Low (TACH4L) See page 249.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0035	Timer A Channel 5 Status and Control Register (TASC5) See page 247.	Read: Write:	CH5F 0	CH5IE	0 R	MS5A	ELS5B	ELS5A	TOV5	CH5MAX
		Reset:	0	0	0	0	0	0	0	0
$0036	Timer A Channel 5 Register High (TACH5H) See page 249.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0037	Timer A Channel 5 Register Low (TACH5L) See page 249.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0038	ADC Status and Control Register (ADSCR) See page 57.	Read: Write:	COCO R	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Reset:	0	0	0	1	1	1	1	1
$0039	ADC Data Register (ADR) See page 59.	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Reset:	Indeterminate after reset							
$003A	ADC Input Clock Register (ADICLK) See page 59.	Read: Write:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
		Reset:	0	0	0	0	0	0	0	0
$003B ↓ $003F	Unimplemented	Read: Write:								
		Reset:								
$0040	Timer B Status and Control Register (TBSC) See page 265.	Read: Write:	TOF 0	TOIE	TSTOP	0 TRST	0 R	PS2	PS1	PS0
		Reset:	0	0	1	0	0	0	0	0
$0041	Timer B Counter Register High (TBCNTH) See page 266.	Read: Write:	Bit 15	14	13	12	11	10	9	8
		Reset:	0	0	0	0	0	0	0	0
$0042	Timer B Counter Register Low (TBCNTL) See page 266.	Read: Write:	Bit 7	6	5	4	3	2	1	0
		Reset:	0	0	0	0	0	0	0	0
$0043	Timer B Modulo Register High (TBMODH) See page 267.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	1	1	1	1	1	1	1	1

= Unimplemented R = Reserved

Figure 2-2. I/O Data, Status and Control Registers (Sheet 5 of 6)

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0044	Timer B Modulo Register Low (TBMODL) See page 267.	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) See page 268.	Read: Write:	CH0F 0	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Reset:	0	0	0	0	0	0	0	0
$0046	Timer B Channel 0 Register High (TBCH0H) See page 271.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0047	Timer B Channel 0 Register Low (TBCH0L) See page 271.	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) See page 268.	Read: Write:	CH1F 0	CH1IE	0 R	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Reset:	0	0	0	0	0	0	0	0
$0049	Timer B Channel 1 Register High (TBCH1H) See page 271.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$004A	Timer B Channel 1 Register Low (TBCH1L) See page 271.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$004B	PIT Status and Control Register (PSC) See page 150.	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) See page 152.	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) See page 152.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$004E	PIT Modulo Register High (PMODH) See page 152.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	1	1	1	1	1	1	1	1
$004F	PIT Modulo Register Low (PMODL) See page 152.	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. I/O Data, Status and Control Registers (Sheet 6 of 6)

2.3 Additional Status and Control Registers

Selected addresses in the range $FE00 to $FFCB contain additional status and control registers as shown in Figure 2-3. A noted exception is the COP control register (COPCTL) at address $FFFF.

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	SIM Break Status Register (SBSR) See page 210.	Read:	R	R	R	R	R	R	BW	R
		Write:							0	
		Reset:							0	
$FE01	SIM Reset Status Register (SRSR) See page 210.	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) See page 211.	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE09	Mask Option Register B (MORB) See page 110.	Read:	EEDIVCLK	0	EESEC	EEMONSEC	AZ32A	0	0	0
		Write:								
		Reset:	Unaffected by reset							
$FE0C	Break Address Register High (BRKH) See page 277.	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) See page 277.	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) See page 276.	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0F	LVI Status Register (LVISR) See page 107.	Read:	LVIOUT	0	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE10	EEDIV High Nonvolatile Register (EEDIVHNVR) See page 50.	Read:	EEDIVS-ECD	R	R	R	R	EEDIV10	EEDIV9	EEDIV8
		Write:								
		Reset:	Unaffected by reset; $FF when blank							
$FE11	EEDIV Low Nonvolatile Register (EEDIVLNVR) See page 50.	Read:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EE2DIV	EEDIV1	EEDIV0
		Write:								
		Reset:	Unaffected by reset; $FF when blank							

= Unimplemented R = Reserved

Figure 2-3. Additional Status and Control Registers (Sheet 1 of 2)

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE1A	EEDIV Divider High Register (EEDIVH) See page 49.	Read:	EEDIV SECD	0	0	0	0	EEDIV10	EEDIV9	EEDIV8
		Write:								
		Reset:	Contents of EEDIVHNVR ($FE10), Bits [6:3] = 0							
$FE1B	EEDIV Divider Low Register (EEDIVL) See page 49.	Read: Write:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EE2DIV	EEDIV1	EEDIV0
		Reset:	Contents of EEDIVLNVR ($FE11)							
$FE1C	EEPROM Nonvolatile Register (EENVR) See page 49.	Read: Write:	R	R	R	EEPRTCT	EEPB3	EEPB2	EEPB1	EEPB0
		Reset:	PV = Programmed value or 1 in the erased state.							
$FE1D	EEPROM Control Register (EECR) See page 46.	Read:	R	0	EEOFF	EERAS1	EERAS0	ELAT	AUTO	EEPGM
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE1F	EEPROM Array Configuration Register (EEACR) See page 47.	Read:	R	R	R	EEPRTCT	EEBP3	EEBP2	EEBP1	EEBP0
		Write:								
		Reset:	Contents of EENVR ($FE1C)							
$FFFF	COP Control Register (COPCTL) See page 79.	Read:	LOW BYTE OF RESET VECTOR							
		Write:	WRITING TO $FFFF CLEARS COP COUNTER							
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved

Figure 2-3. Additional Status and Control Registers (Sheet 2 of 2)

2.4 Vector Addresses and Priority

Addresses in the range $FFCC to $FFFF contain the user-specified vector locations. The vector addresses are shown in Table 2-1.

Table 2-1. Vector Addresses[1]

Low

Address	Vector
$FFCC	TIMA Channel 5 Vector (high)
$FFCD	TIMA Channel 5 Vector (low)
$FFCE	TIMA Channel 4 Vector (high)
$FFCF	TIMA Channel 4 Vector (low)
$FFD0	ADC Vector (high)
$FFD1	ADC 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	MSCAN08 Transmit Vector (High)
$FFDB	MSCAN08 Transmit Vector (Low)
$FFDC	MSCAN08 Receive Vector (High)
$FFDD	MSCAN08 Receive Vector (Low)
$FFDE	MSCAN08 Error Vector (High)
$FFDF	MSCAN08 Error Vector (Low)
$FFE0	MSCAN08 Wakeup Vector (High)
$FFE1	MSCAN08 Wakeup Vector (Low)
$FFE2	SPI Transmit Vector (High)
$FFE3	SPI Transmit Vector (Low)
$FFE4	SPI Receive Vector (High)
$FFE5	SPI Receive Vector (Low)
$FFE6	TIMB Overflow Vector (High)
$FFE7	TIMB Overflow Vector (Low)
$FFE8	TIMB CH1 Vector (High)
$FFE9	TIMB CH1 Vector (Low)
$FFEA	TIMB CH0 Vector (High)
$FFEB	TIMB CH0 Vector (Low)
$FFEC	TIMA Overflow Vector (High)
$FFED	TIMA Overflow Vector (Low)
$FFEE	TIMA CH3 Vector (High)
$FFEF	TIMA CH3 Vector (Low)

— Continued on next page

Table 2-1. Vector Addresses[1] (Continued)

Address	Vector
$FFF0	TIMA CH2 Vector (High)
$FFF1	TIMA CH2 Vector (Low)
$FFF2	TIMA CH1 Vector (High)
$FFF3	TIMA CH1 Vector (Low)
$FFF4	TIMA CH0 Vector (High)
$FFF5	TIMA CH0 Vector (Low)
$FFF6	PIT Vector (High)
$FFF7	PIT Vector (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)
$FFFF	Reset Vector (Low)

High

1. All available ROM locations not defined by the user will by default be filled with the software interrupt (SWI, opcode 83) instruction — see Chapter 6 Central Processor Unit (CPU). Take this into account when defining vector addresses. It is recommended that ALL vector addresses are defined.

2.5 Random-Access Memory (RAM)

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

Before processing an interrupt, the CPU uses 5 bytes of the stack to save the contents of the CPU registers.

NOTE

For M6805 compatibility, the H register is not stacked.

During a subroutine call, the CPU uses 2 bytes of the stack to store the return address. The stack pointer decrements during pushes and increments during pulls.

NOTE

Care should be taken when using nested subroutines. The CPU may overwrite data in the RAM during a subroutine or during the interrupt stacking operation.

2.6 Read-Only Memory (ROM)

The user ROM consists of up to 32, 256 bytes from addresses $8000–$FDFF. The monitor ROM and vectors are located from $FE20–$FF5F.

Fifty-two user vectors, $FFCC–$FFFF, are dedicated to user-defined reset and interrupt vectors.

Security has been incorporated into the MC68HC08AZ32A to prevent external viewing of the ROM contents[1]. This feature is selected by a mask option and ensures that customer-developed software remains propriety. See Chapter 10 Mask Options.

2.7 Electrically Erasable Programmable Read-Only Memory (EEPROM)

The 512 bytes of EEPROM are 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.

Features include:

- 512 bytes nonvolatile memory
- Byte, block, or bulk erasable
- Nonvolatile EEPROM configuration and block protection options
- On-chip charge pump for programming/erasing
- Program/erase protection option
- Read protection option
- AUTO bit driven programming/erasing time feature

2.7.1 EEPROM Configuration

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

- EEPROM timebase reference
- EEPROM program/erase protection
- EEPROM block protection

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

EENVR and EEDIVNVR are nonvolatile 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 divider register (EEDIV: EEDIVH and EEDIVL).

2.7.1.1 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 mask option register B) using a timebase divider circuit controlled by the 16-bit EEPROM timebase divider EEDIV 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 35 µs. The timebase divider value is calculated by using the following formula:

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

This value is written to the EEPROM timebase divider register (EEDIVH and EEDIVL) or programmed into the EEPROM timebase divider nonvolatile register prior to any EEPROM program or erase operations(2.7.1 EEPROM Configuration and 2.7.1.1 EEPROM Timebase Requirements).

2.7.1.2 EEPROM Program/Erase Protection

The EEPROM has a special feature that designates the 16 bytes of addresses from $08F0 to $08FF to be permanently secured. This program/erase protect option is enabled by programming the EEPRTCT bit in the EEPROM nonvolatile register (EENVR) to 0.

Once the EEPRTCT bit is programmed to 0 for the first time:

- Programming and erasing of secured locations $08F0–$08FF is permanently disabled.
- Secured locations $08F0–$08FF can be read as normal.
- Programming and erasing of EENVR is permanently disabled.
- Bulk and block erase operations are disabled for the unprotected locations $0800–$08EF, $0900–$09FF.
- Single byte program and erase operations are still available for locations $0800–$08EF and $0900–$09FF for all bytes that are not protected by the EEPROM block protect EEBPx bits (see 2.7.1.3 EEPROM Block Protection and 2.7.3.2 EEPROM Array Configuration Register).

NOTE

Once armed, the protect option is permanently enabled. As a consequence, all functions in the EENVR will remain in the state they were in immediately before the security was enabled.

2.7.1.3 EEPROM Block Protection

The 512 bytes of EEPROM are 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 2-2 shows the address ranges for the blocks.

Table 2-2. 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 upon read of the 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. Please see 2.7.3.2 EEPROM Array Configuration Register for more information.

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 0 thereafter. Once this security feature is armed, erase and program mode are disabled for EEDIVHNVR and EEDIVLNVR. Modifications to the EEDIVH and EEDIVL registers are also disabled. Therefore, be cautious when programming a value into the EEDIVHNVR.

2.7.2 EEPROM Programming and Erasing

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

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

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

The EEPROM can be programmed such that one or multiple bits are programmed (written to 0) at a time. However, the user may never program the same bit location more than once before erasing the entire byte. In other words, the user is not allowed to program a 0 to a bit that is already programmed (bit state is already 0).

For some applications it might be advantageous to track more than 10K events with a single byte of EEPROM by programming one bit at a time. For that purpose, a special selective bit programming technique is available. An example of this technique is illustrated in Table 2-3.

Table 2-3. Example Selective Bit Programming Description

Description	Program Data in Binary	Result in Binary
Original state of byte (erased)	N/A	1111:1111
First event is recorded by programming bit position 0	1111:1110	1111:1110
Second event is recorded by programming bit position 1	1111:1101	1111:1100
Third event is recorded by programming bit position 2	1111:1011	1111:1000
Fourth event is recorded by programming bit position 3	1111:0111	1111:0000
Events five through eight are recorded in a similar fashion		

NOTE

None of the bit locations are actually programmed more than once although the byte was programmed eight times.

When this technique is utilized, a program/erase cycle is defined as multiple program sequences (up to eight) to a unique location followed by a single erase operation.

2.7.2.1 Program/Erase Using AUTO Bit

An additional feature available for EEPROM program and erase operations is the AUTO mode. When enabled, AUTO mode will activate an internal timer that will automatically terminate the program/erase cycle and clear the EEPGM bit. Please see 2.7.2.2 EEPROM Programming, 2.7.2.3 EEPROM Erasing, and 2.7.3.1 EEPROM Control Register for more information.

2.7.2.2 EEPROM Programming

The unprogrammed or erase state of an EEPROM bit is a 1. Programming changes the state to a 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)

NOTE

If using the AUTO mode, also set the AUTO bit during step 1.

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 time, t_{EEPGM}, to program the byte.
5. Clear EEPGM bit.
6. Wait for 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 bits.(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 write occurs, the last address and data will be latched overriding the previous address and data. Once data is written 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.*

2.7.2.3 EEPROM Erasing

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

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

1. Configure EERAS1 and EERAS0 for byte, block or bulk erase; set EELAT in EECR.(A)

NOTE

If using the AUTO mode, also set the AUTO bit in step 1.

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_{EEBYTE} for byte erase; $t_{EEBLOCK}$ for block erase; $t_{EEBULK.}$ 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 write occurs, the last address and data will be latched overriding the previous address and data. Once data is written 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_{EEBYTE}/$t_{EEBLOCK}$/t_{EEBULK}. 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.*

2.7.3 EEPROM Register Descriptions

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

2.7.3.1 EEPROM Control Register

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

Address: $FE1D

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

= Unimplemented

Figure 2-4. EEPROM Control Register (EECR)

Bit 7— Unused

This read/write bit is software programmable but has no functionality.

EEOFF — EEPROM Power Down

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

EERAS1 and EERAS0 — Erase/Program Mode Select Bits

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

Table 2-4. 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 cannot be cleared if EEPGM is still set. Reset clears this bit.

1 = Buses configured for EEPROM programming 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. For further information, refer to note D under 2.7.2.2 EEPROM Programming and 2.7.2.3 EEPROM Erasing as well as 20.13 EEPROM Memory Characteristics)

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

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 clear EEPGM only to allow time for the removal of high voltage.

2.7.3.2 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 nonvolatile register (EENVR) after a reset.

Address: $FE1F

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

= Unimplemented

Figure 2-5. EEPROM Array Configuration Register (EEACR)

Bit 7:5 — Unused

These read/write bits are software programmable but have no functionality.

EEPRTCT — EEPROM Protection Bit

The EEPRTCT bit is used to enable the security feature in the EEPROM (see 2.7.1.2 EEPROM Program/Erase Protection).

1 = EEPROM security disabled
0 = EEPROM security enabled

This feature is a write-once feature. Once the protection is enabled it may not be disabled.

EEBP[3:0] — EEPROM Block Protection Bits

These bits prevent blocks of EEPROM array from being programmed or erased. Refer to Table 2-5.

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

Table 2-5. EEPROM Block Protection Bits

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

Table 2-6. EEPROM Block Protect and Security Summary

Address Range	EEBPx	EEPRTCT = 1	EEPRTCT = 0
$0800–$087F	EEBP0 = 0	Byte Programming Available Bulk, Block and Byte Erasing Available	Byte Programming Available Only Byte Erasing Available
	EEBP0 = 1	Protected	Protected
$0880–$08EF	EEBP1 = 0	Byte Programming Available Bulk, Block and Byte Erasing Available	Byte Programming Available Only Byte Erasing Available
	EEBP1 = 1	Protected	Protected
$08F0–$08FF	EEBP1 = 0	Byte Programming Available Bulk, Block and Byte Erasing Available	Secured (No Programming or Erasing)
	EEBP1 = 1	Protected	
$0900–$097F	EEBP2 = 0	Byte Programming Available Bulk, Block and Byte Erasing Available	Byte Programming Available Only Byte Erasing Available
	EEBP2 = 1	Protected	Protected
$0980–$09FF	EEBP3 = 0	Byte Programming Available Bulk, Block and Byte Available	Byte Programming Available Only Byte Erasing Available
	EEBP3 = 1	Protected	Protected

2.7.3.3 EEPROM Nonvolatile 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. (See 2.7.3.1 EEPROM Control Register for individual bit descriptions).

Address: $FE1C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	R	R	EEPRTCT	EEBP3	EEBP2	EEBP1	EEBP0
Write:								
Reset:	PV							

R = Reserved PV = Programmed value or 1 in the erased state.

Figure 2-6. EEPROM Nonvolatile Register (EENVR)

NOTE

The EENVR will leave the factory programmed with $F0 such that the full array is available and unprotected.

2.7.3.4 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 35 µs timebase for EEPROM control. These two read/write registers are respectively loaded with the contents of the EEPROM timebase divider on volatile registers (EEDIVHNVR and EEDIVLNVR) after a reset.

Address: $FE1A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EEDIVS-ECD	0	0	0	0	EEDIV10	EEDIV9	EEDIV8
Write:								
Reset:	Contents of EEDIVHNVR ($FE10), Bits [6:3] = 0							

= Unimplemented

Figure 2-7. EEDIV Divider High Register (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 2-8. EEDIV Divider Low Register (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 the EEDIVSECD bit). The EEDIVHNVR and EEDIVLNVR nonvolatile 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 35 µs from the selected reference clock source (CGMXCLK or bus block in the CONFIG-2 register) 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:

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

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

For example, if the reference frequency is 4.9152 MHz, 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.

2.7.3.5 EEPROM Timebase Divider Nonvolatile Register

The 16-bit EEPROM timebase divider nonvolatile 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: Write:	EEDIVS-ECD	R	R	R	R	EEDIV10	EEDIV9	EEDIV8
Reset:	Unaffected by reset; $FF when blank							

R = Reserved

Figure 2-9. EEPROM Divider Nonvolatile Register High (EEDIVHNVR))

Address: $FE11

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

Figure 2-10. EEPROM Divider Nonvolatile Register Low (EEDIVLNVR)

These two registers are protected from erase and program operations if the EEDIVSECD is set to 1 in the EEDIVH (see 2.7.3.4 EEPROM Timebase Divider Register) or programmed to a 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 0 thereafter. Once this security feature is armed, erase and program mode 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.

2.7.4 Low-Power Modes

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

2.7.4.1 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.

2.7.4.2 Stop Mode

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

If stop mode is entered while EELAT and EEPGM are 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.

Chapter 3
Analog-to-Digital Converter (ADC) Module

3.1 Introduction

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

3.2 Features

Features include:

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

3.3 Functional Description

Fifteen ADC channels are available for sampling external sources at pins PTD6/ATD14/TACLK–PTD0/ATD8 and PTB7/ATD7–PTB0/ATD0. An analog multiplexer allows the single ADC converter to select one of the 15 ADC channels as ADC voltage input (ADCVIN). ADCVIN is converted by the successive approximation register-based counters. When the conversion is completed, ADC places the result in the ADC data register and sets a flag or generates an interrupt. See Figure 3-2.

3.3.1 ADC Port I/O Pins

PTD6/ATD14/TACLK–PTD0/ATD8 and PTB7/ATD7–PTB0/ATD0 are general- purpose I/O pins that are shared with the ADC channels.

The channel select bits (ADC status and control register, $0038) 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 which is in use by the ADC will return a 0 if the corresponding DDR bit is at 0. If the DDR bit is at 1, the value in the port data latch is read.

NOTE

Do not use ADC channels ATD14 or ATD12 when using the PTD6/ATD14/TACLK or PTD4/ATD12/TBCLK pins as the clock inputs for the 16-bit timers.

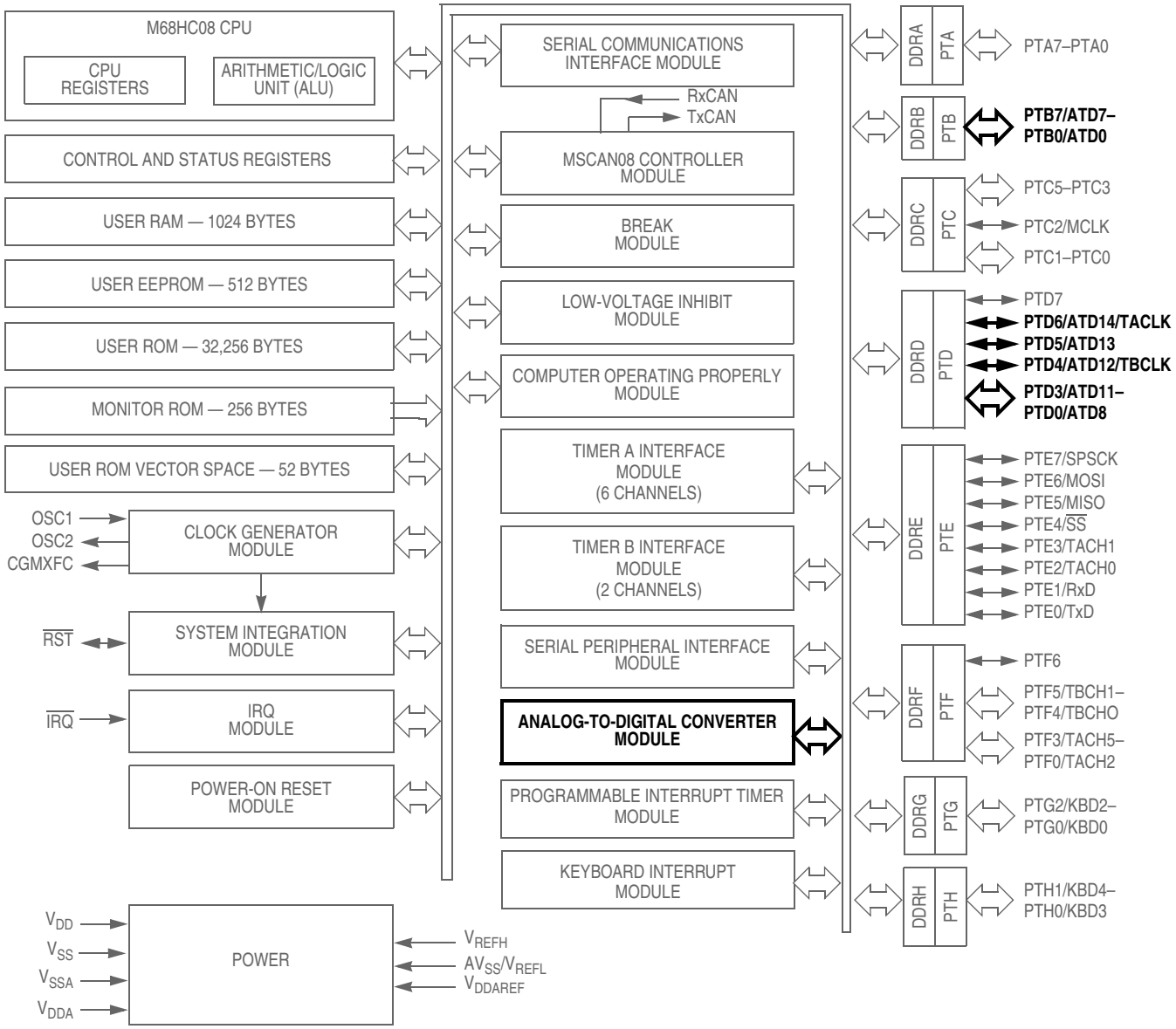


Figure 3-1. Block Diagram Highlighting ADC Block and Pins

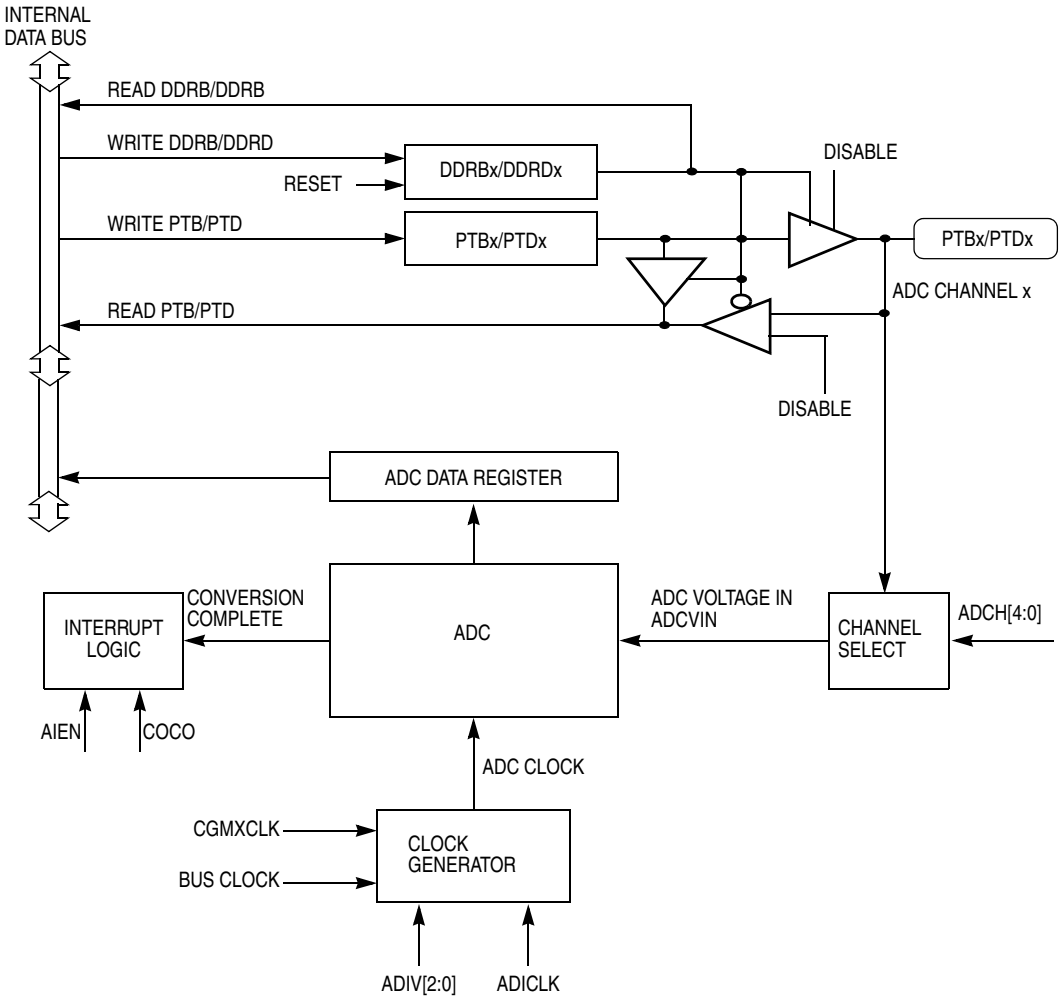


Figure 3-2. ADC Block Diagram

3.3.2 Voltage Conversion

When the input voltage to the ADC equals V_{REFH} (see 20.7 ADC Characteristics), the ADC converts the signal to $FF (full scale). If the input voltage equals $AV_{SS}/V_{REFL,}$ the ADC converts it to $00. Input voltages between V_{REFH} and AV_{SS}/V_{REFL} are a straight-line linear conversion. Conversion accuracy of all other input voltages is not guaranteed. Avoid current injection on unused ADC inputs to prevent potential conversion error.

NOTE

Input voltage should not exceed the analog supply voltages.

3.3.3 Conversion Time

Conversion starts after a write to the ADSCR (ADC status and control register, $0038), and requires between 16 and 17 ADC clock cycles to complete. Conversion time in terms of the number of bus cycles is a function of ADICLK select, CGMXCLK frequency, bus frequency, and ADIV prescaler bits. For example, with a CGMXCLK frequency of 4 MHz, bus frequency of 8 MHz, and fixed ADC clock frequency of 1 MHz, one conversion will take between 16 and 17 µs and there will be between 128 bus cycles between each conversion. Sample rate is approximately 60 kHz.

Refer to 20.7 ADC Characteristics.

$$\text{Conversion Time} = \frac{\text{16 to 17 ADC Clock Cycles}}{\text{ADC Clock Frequency}}$$

Number of Bus Cycles = Conversion Time x Bus Frequency

3.3.4 Continuous Conversion

In the 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 (ADC status and control register, $0038) is cleared. The COCO bit is set after the first conversion and will stay set for the next several conversions until the next write of the ADC status and control register or the next read of the ADC data register.

3.3.5 Accuracy and Precision

The conversion process is monotonic and has no missing codes. See 20.7 ADC Characteristics for accuracy information.

3.4 Interrupts

When the AIEN bit is set, the ADC module is capable of generating a CPU interrupt after each ADC conversion. A CPU interrupt is generated if the COCO bit (ADC status and control register, $0038) is at 0. The COCO bit is not used as a conversion complete flag when interrupts are enabled.

3.5 Low-Power Modes

The following subsections describe the low-power modes.

3.5.1 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 the ADCH[4:0] bits in the ADC status and control register before executing the WAIT instruction.

3.5.2 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. Allow one conversion cycle to stabilize the analog circuitry before attempting a new ADC conversion after exiting stop mode.

3.6 I/O Signals

The ADC module has 15 channels that are shared with I/O ports B and D. Refer to 20.7 ADC Characteristics for voltages referenced below.

3.6.1 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

Route V_{DDAREF} carefully for maximum noise immunity and place bypass capacitors as close as possible to the package. V_{DDAREF} must be present for operation of the ADC.

3.6.2 ADC Analog Ground/ADC Voltage Reference Low Pin (AV_{SS}/V_{REFL})

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

V_{REFL} is the lower reference supply for the ADC.

3.6.3 ADC Voltage Reference Pin (V_{REFH})

V_{REFH} is the high reference voltage for all analog-to-digital conversions.

3.6.4 ADC Voltage In (ADCVIN)

ADCVIN is the input voltage signal from one of the 15 ADC channels to the ADC module.

3.7 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 (ADICLK)

3.7.1 ADC Status and Control Register

The following paragraphs describe the function of the ADC status and control register.

Address: $0038

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

R = Unimplemented

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

COCO — Conversions Complete Bit

In non-interrupt mode (AIEN = 0), COCO is a read-only bit that is set at the end of each conversion. COCO will stay set until cleared by a read of the ADC data register. Reset clears this bit.

In interrupt mode (AIEN = 1), COCO is a read-only bit that is not set at the end of a conversion. It always reads as a 0.

1 = Conversion completed (AIEN = 0)

0 = Conversion not completed (AIEN = 0) or CPU interrupt enabled (AIEN = 1)

NOTE

The write function of the COCO bit is reserved. When writing to the ADSCR register, always have a 0 in the COCO bit position.

AIEN — ADC Interrupt Enable Bit

When this bit is set, an interrupt is generated at the end of an ADC conversion. The interrupt signal is cleared when the data register is read or the status/control register is written. Reset clears the AIEN bit.

1 = ADC interrupt enabled

0 = ADC interrupt disabled

ADCO — ADC Continuous Conversion Bit

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

1 = Continuous ADC conversion

0 = One ADC conversion

ADCH[4:0] — ADC Channel Select Bits

ADCH4, ADCH3, ADCH2, ADCH1, and ADCH0 form a 5-bit field which is used to select one of 15 ADC channels. Channel selection is detailed in Table 3-1. Care should be taken when using a port pin as both an analog and a 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 one. This feature allows for reduced power consumption for the MCU when the ADC is not used. Reset sets these bits.

NOTE

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

Table 3-1. 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	0	0	0	PTD0/ATD8
0	1	0	0	1	PTD1/ATD9
0	1	0	1	0	PTD2/ATD10

— Continued on next page

Table 3-1. Mux Channel Select (Continued)

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	Input Select
0	1	0	1	1	PTD3/ATD11
0	1	1	0	0	PTD4/ATD12/TBCLK
0	1	1	0	1	PTD5/ATD13
0	1	1	1	0	PTD6/ATD14/TACLK
Range 01111 ($0F) to 11010 ($1A)					Unused(1)
					Unused(1)
1	1	0	1	1	Reserved
1	1	1	0	0	Unused(1)
1	1	1	0	1	V_{REFH}(2)
1	1	1	1	0	AV_{SS}/V_{REFL}(2)
1	1	1	1	1	[ADC power off]

1. If any unused channels are selected, the resulting ADC conversion will be unknown.
2. The voltage levels supplied from internal reference nodes as specified in the table are used to verify the operation of the ADC converter both in production test and for user applications.

3.7.2 ADC Data Register

One 8-bit result register 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:	Indeterminate after Reset							

= Unimplemented

Figure 3-4. ADC Data Register (ADR)

3.7.3 ADC Input Clock Register

This register 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 3-5. ADC Input Clock Register (ADICLK)

ADIV2–ADIV0 — ADC Clock Prescaler Bits

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

Table 3-2. 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 Register Bit

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

If the external clock (CGMXCLK) is equal to or greater than 1 MHz, CGMXCLK can be used as the clock source for the ADC. If CGMXCLK is less than 1 MHz, use the PLL-generated bus clock as the clock source. As long as the internal ADC clock is at approximately 1 MHz, correct operation can be guaranteed. See 20.7 ADC Characteristics.

1 = Internal bus clock

0 = External clock (CGMXCLK)

$$1\ \text{MHz} = \frac{f_{XCLK}\ \text{or Bus Frequency}}{\text{ADIV[2:0]}}$$

NOTE

During the conversion process, changing the ADC clock will result in an incorrect conversion.

Chapter 4
Clock Generator Module (CGM)

4.1 Introduction

The clock generator module (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 clocks are derived. 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 high-frequency crystals.

4.2 Features

Features include:

- 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

4.3 Functional Description

The CGM consists of three major submodules:

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

Figure 4-1 shows the structure of the CGM.

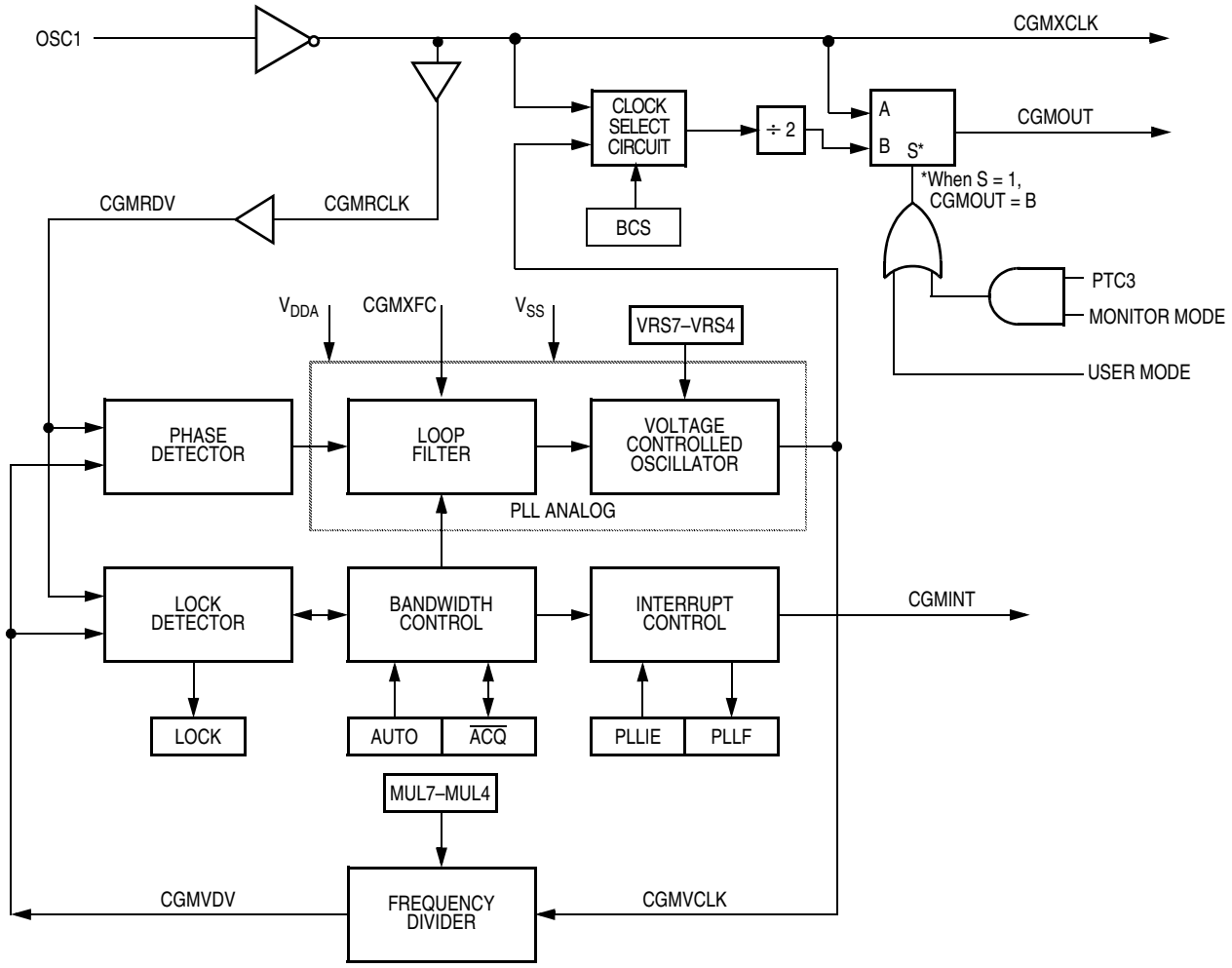


Figure 4-1. CGM Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001C	PLL Control Register (PCTL) See page 69.	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) See page 70.	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) See page 71.	Read:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
		Write:								
		Reset:	0	1	1	0	0	1	1	0

= Unimplemented

Figure 4-2. I/O Register Summary

4.3.1 Crystal Oscillator Circuit

The crystal oscillator circuit consists of an inverting amplifier and an external crystal. The OSC1 pin is the input to the amplifier and the OSC2 pin is the output. The SIMOSCEN signal enables the crystal oscillator circuit.

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

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

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

4.3.2 Phase-Locked Loop Circuit (PLL)

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

4.3.2.1 Circuits

The PLL consists of these 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_{CGMVRS}. Modulating the voltage on the CGMXFC pin changes the frequency within this range. By design, f_{CGMVRS} is equal to the nominal center-of-range frequency, f_{NOM}, (4.9152 MHz) times a linear factor L or (L)f_{NOM}.

CGMRCLK is the PLL reference clock, a buffered version of CGMXCLK. CGMRCLK runs at a frequency, $f_{CGMRCLK}$, and is fed to the PLL through a buffer. The buffer output is the final reference clock, CGMRDV, running at a frequency $f_{CGMRDV} = f_{CGMRCLK}$.

The VCO's output clock, CGMVCLK, running at a frequency $f_{CGMVCLK}$, 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 $f_{CGMVDV} = f_{CGMVCLK}/N$. 4.3.2.4 Programming the PLL for more information.

The phase detector then compares the VCO feedback clock, CGMVDV, with the final reference clock, CGMRDV. A correction pulse is generated based on the phase difference between the two signals. The loop filter then slightly alters the dc voltage on the external capacitor connected to CGMXFC based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, as described in 4.3.2.2 Acquisition and Tracking Modes. The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

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

4.3.2.2 Acquisition and Tracking Modes

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

1. Acquisition mode — In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL startup or when the PLL has suffered a severe noise hit and the VCO frequency is far off the desired frequency. When in acquisition mode, the $\overline{ACQ}$ bit is clear in the PLL bandwidth control register. See 4.5.2 PLL Bandwidth Control Register.
2. Tracking mode — In tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source (see 4.3.3 Base Clock Selector Circuit). The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{ACQ}$ bit is set.

4.3.2.3 Manual and Automatic PLL Bandwidth Modes

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

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT (see 4.5.2 PLL Bandwidth Control Register). If PLL CPU interrupt requests are enabled, the software can wait for a PLL CPU interrupt request and then check the LOCK bit. If CPU interrupts are disabled, software can poll the LOCK bit continuously (during PLL startup, usually) or at periodic intervals. In either case, when the LOCK bit is set, the VCO clock is safe to use as the source for the base clock (see 4.3.3 Base Clock Selector Circuit). If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application (see 4.6 Interrupts).

These conditions apply when the PLL is in automatic bandwidth control mode:

- The $\overline{ACQ}$ bit (see 4.5.2 PLL Bandwidth Control Register) is a read-only indicator of the mode of the filter. Refer to 4.3.2.2 Acquisition and Tracking Modes.
- The $\overline{ACQ}$ bit is set when the VCO frequency is within a certain tolerance, Δ_{TRK}, and is cleared when the VCO frequency is out of a certain tolerance, Δ_{UNT}. See 20.11 CGM Acquisition/Lock Time Information.
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance, Δ_{Lock}, and is cleared when the VCO frequency is out of a certain tolerance, Δ_{UNL}. See 20.11 CGM Acquisition/Lock Time Information.
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. See 20.11 CGM Acquisition/Lock Time Information.

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

The following conditions apply when in manual mode:

- $\overline{ACQ}$ is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the $\overline{ACQ}$ bit must be clear.
- Before entering tracking mode ($\overline{ACQ}$ = 1), software must wait a given time, t_{ACQ} (see 20.11 CGM Acquisition/Lock Time Information), 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

4.3.2.4 Programming the PLL

Use the following 9-step procedure to program the PLL. Table 4-1 shows the variables used and their meaning (please also reference Figure 4-1).

Table 4-1. Variable Definitions

Variable	Definition
f_{BUSDES}	Desired bus clock frequency
$f_{VCLKDES}$	Desired VCO clock frequency
$f_{CGMRCLK}$	Chosen reference crystal frequency
$f_{CGMVCLK}$	Calculated VCO clock frequency
f_{BUS}	Calculated bus clock frequency
f_{NOM}	Nominal VCO center frequency
f_{CGMVRS}	Shifted VCO center frequency

1. Choose the desired bus frequency, f_{BUSDES}.

   Example: f_{BUSDES} = 8 MHz

2. Calculate the desired VCO frequency, $f_{VCLKDES}$.

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

   Example: $f_{VCLKDES} = 4 \times 8\text{ MHz} = 32\text{ MHz}$

3. Using a reference frequency, f_{RCLK}, equal to the crystal frequency, calculate the VCO frequency multiplier, N. Round the result to the nearest integer.

   $$N = \frac{f_{VCLKDES}}{f_{CGMRCLK}}$$

   $$\text{Example:}\quad N = \frac{32\text{ MHz}}{4\text{ MHz}} = 8$$

4. Calculate the VCO frequency, $f_{CGMVCLK}$.

   $$f_{CGMVCLK} = N \times f_{CGMRCLK}$$

   Example: $f_{CGMVCLK} = 8 \times 4\text{ MHz} = 32\text{ MHz}$

5. Calculate the bus frequency, f_{BUS}, and compare f_{BUS} with f_{BUSDES}.

$$f_{Bus} = \frac{f_{CGMVCLK}}{4}$$

$$\text{Example:} \quad f_{BUS} = \frac{32\ \text{MHz}}{4} = 8\ \text{MHz}$$

6. If the calculated f_{BUS} is not within the tolerance limits of your application, select another f_{BUSDES} or another f_{RCLK}.
7. Using the value 4.9152 MHz for f_{NOM}, calculate the VCO linear range multiplier, L. The linear range multiplier controls the frequency range of the PLL.

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

$$\text{Example:} \quad L = \frac{32\ \text{MHz}}{4.9152\ \text{MHz}} = 7$$

8. Calculate the VCO center-of-range frequency, f_{CGMVRS}. The center-of-range frequency is the midpoint between the minimum and maximum frequencies attainable by the PLL.

$$f_{CGMVRS} = L \times f_{NOM}$$

$$\text{Example: } f_{CGMVRS} = 7 \times 4.9152\ \text{MHz} = 34.4\ \text{MHz}$$

NOTE

For proper operation: $|f_{CGMVRS} - f_{CGMVCLK}| \leq \frac{f_{NOM}}{2}$.

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

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

4.3.2.5 Special Programming Exceptions

The programming method described in 4.3.2.4 Programming the PLL, does not account for two possible exceptions. A value of 0 for N or L is meaningless when used in the equations given. To account for these exceptions:

- A 0 value for N is interpreted the same as a value of 1.
- A 0 value for L disables the PLL and prevents its selection as the source for the base clock. See 4.3.3 Base Clock Selector Circuit.

4.3.3 Base Clock Selector Circuit

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

The BCS bit in the PLL control register (PCTL) selects which clock drives CGMOUT. The VCO clock cannot be selected as the base clock source if the PLL is not turned on. The PLL cannot be turned off if the VCO clock is selected. The PLL cannot be turned on or off simultaneously with the selection or deselection of the VCO clock. The VCO clock also cannot be selected as the base clock source if the factor L is programmed to a 0. This value would set up a condition inconsistent with the operation of the PLL, so that the PLL would be disabled and the crystal clock would be forced as the source of the base clock.

4.3.4 CGM External Connections

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

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in Figure 4-3. Figure 4-3 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)

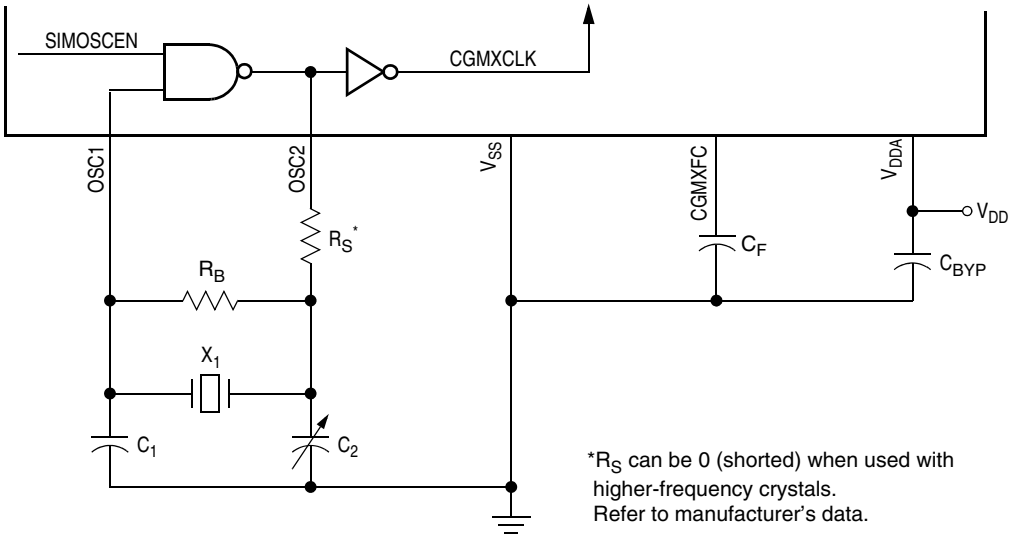


Figure 4-3. CGM External Connections

The series resistor (R_S) may not be required for all ranges of operation, especially with high-frequency crystals. Refer to the crystal manufacturer's data for more information.

Figure 4-3 also shows the external components for the PLL:

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

Routing should be done with great care to minimize signal cross talk and noise. See 20.11 CGM Acquisition/Lock Time Information for routing information and more information on the filter capacitor's value and its effects on PLL performance.

4.4 I/O Signals

The following paragraphs describe the CGM input/output (I/O) signals.

4.4.1 Crystal Amplifier Input Pin (OSC1)

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

4.4.2 Crystal Amplifier Output Pin (OSC2)

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

4.4.3 External Filter Capacitor Pin (CGMXFC)

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

NOTE

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

4.4.4 Analog Power Pin (V_{DDA})

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

NOTE

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

4.4.5 Oscillator Enable Signal (SIMOSCEN)

The SIMOSCEN signal enables the oscillator and PLL.

4.4.6 Crystal Output Frequency Signal (CGMXCLK)

CGMXCLK is the crystal oscillator output signal. It runs at the full speed of the crystal ($f_{CGMXCLK}$) and comes directly from the crystal oscillator circuit. Figure 4-3 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 startup.

4.4.7 CGM Base Clock Output (CGMOUT)

CGMOUT is the clock output of the CGM. This signal is used to generate 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.

4.4.8 CGM CPU Interrupt (CGMINT)

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

4.5 CGM Registers

Three registers control and monitor operation of the CGM:

- PLL control register (PCTL)
- PLL bandwidth control register (PBWC)
- PLL programming register (PPG)

4.5.1 PLL Control Register

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

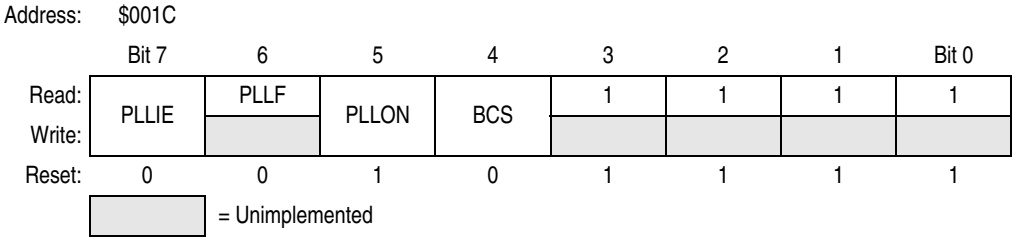


Figure 4-4. PLL Control Register (PCTL)

PLLIE — PLL Interrupt Enable Bit

This read/write bit enables the PLL to generate a CPU 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 CPU interrupt requests enabled
0 = PLL CPU interrupt requests disabled

PLLF — PLL Flag Bit

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

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

NOTE

Do not inadvertently clear the PLLF bit. Be aware that any read or read-modify- write operation on the PLL control register clears the PLLF bit.

PLLON — PLL On Bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). See 4.3.3 Base Clock Selector Circuit. Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

BCS — Base Clock Select Bit

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. See 4.3.3 Base Clock Selector Circuit. Reset and the STOP instruction clear the BCS bit.

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

NOTE

PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMVCLK requires two writes to the PLL control register. See 4.3.3 Base Clock Selector Circuit.

PCTL3–PCTL0 — Unimplemented

These bits provide no function and always read as 1s.

4.5.2 PLL Bandwidth Control Register

The PLL bandwidth control register:

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

Address: $001D

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

= Unimplemented

Figure 4-5. PLL Bandwidth Control Register (PBWC)

AUTO — Automatic Bandwidth Control Bit

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

1 = Automatic bandwidth control
0 = Manual bandwidth control

LOCK — Lock Indicator Bit

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock, CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as 0 and has no meaning. Reset clears the LOCK bit.

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

$\overline{ACQ}$ — Acquisition Mode Bit

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

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

1 = Tracking mode
0 = Acquisition mode

XLD — Crystal Loss Detect Bit

When the VCO output, CGMVCLK, is driving CGMOUT, this read/write bit can indicate whether the crystal reference frequency is active or not.

1 = Crystal reference not active
0 = Crystal reference active

To check the status of the crystal reference, do the following:

1. Write a 1 to XLD.
2. Wait N × 4 cycles. N is the VCO frequency multiplier.
3. Read XLD.

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 0s to bits 3–0 when writing to PBWC.

4.5.3 PLL Programming Register

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 4-6. PLL Programming Register (PPG)

MUL7–MUL4 — Multiplier Select Bits

These read/write bits control the modulo feedback divider that selects the VCO frequency multiplier, N. (See 4.3.2.1 Circuits and 4.3.2.4 Programming the PLL). A value of $0 in the multiplier select bits configures the modulo feedback divider the same as a value of $1. Reset initializes these bits to $6 to give a default multiply value of 6. Refer to Table 4-2.

NOTE

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

Table 4-2. 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

VRS7–VRS4 — 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 4.3.2.1 Circuits, 4.3.2.4 Programming the PLL, and 4.5.1 PLL Control Register.) VRS7–VRS4 cannot be written when the PLLON bit in the PLL control register (PCTL) is set. See 4.3.2.5 Special Programming Exceptions. A value of $0 in the VCO range select bits disables the PLL and clears the BCS bit in the PCTL. (See 4.3.3 Base Clock Selector Circuit and 4.3.2.5 Special Programming Exceptions for more information.) Reset initializes the bits to $6 to give a default range multiply value of 6.

NOTE

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

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

4.6 Interrupts

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

Software should read the LOCK bit after a PLL CPU 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, CPU interrupt requests should be disabled to prevent PLL interrupt service routines from impeding software performance or from exceeding stack limitations.

NOTE

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

4.7 Low-Power Modes

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

4.7.1 Wait Mode

The CGM remains active in wait mode. Before entering wait mode, software can disengage and turn off the PLL by clearing the BCS and PLLON bits in the PLL control register (PCTL). Less power-sensitive applications can disengage the PLL without turning it off. Applications that require the PLL to wake the MCU from wait mode also can deselect the PLL output without turning off the PLL.

4.7.2 Stop Mode

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

If CGMOUT is being driven by CGMVCLK and a STOP instruction is executed; the PLL will clear the BCS bit in the PLL control register, causing CGMOUT to be driven by CGMXCLK. When the MCU recovers from STOP, the crystal clock divided by two drives CGMOUT and BCS remains clear.

4.8 CGM During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See 19.2 Break Module (BRK)).

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

To protect the PLLF bit during the break state, write a 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.

4.9 Acquisition/Lock Time Specifications

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

4.9.1 Acquisition/Lock Time Definitions

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

Other systems refer to acquisition and lock times as the time the system takes to reduce the error between the actual output and the desired output to within specified tolerances. Therefore, the acquisition or lock

time varies according to the original error in the output. Minor errors may not even be registered. Typical PLL applications prefer to use this definition because the system requires the output frequency to be within a certain tolerance of the desired frequency regardless of the size of the initial error.

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

- 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 4.3.2.3 Manual and Automatic PLL Bandwidth Modes), acquisition time expires when the $\overline{ACQ}$ bit becomes set in the PLL bandwidth control register (PBWC).
- Lock time, t_{Lock}, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the lock mode entry tolerance, Δ_{Lock}. Lock time is based on an initial frequency error, $(f_{DES} - f_{ORIG})/f_{DES}$, of not more than ±100%. In automatic bandwidth control mode, lock time expires when the LOCK bit becomes set in the PLL bandwidth control register (PBWC). (See 4.3.2.3 Manual and Automatic PLL Bandwidth Modes).

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

4.9.2 Parametric Influences on Reaction Time

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

The most critical parameter which affects the reaction times of the PLL is the reference frequency, f_{CGMRDV} (please reference Figure 4-1). 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 $f_{CGMXCLK}$.

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 a change in charge) is proportional to the capacitor size. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor is too large, the PLL may not be able to adjust the voltage in a reasonable time. See 4.9.3 Choosing a Filter Capacitor.

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

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

4.9.3 Choosing a Filter Capacitor

As described in 4.9.2 Parametric Influences on Reaction Time, the external filter capacitor, C_F, is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage. The value of the capacitor must, therefore, be chosen with supply potential and reference frequency in mind. For proper operation, the external filter capacitor must be chosen according to this equation:

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

For acceptable values of C_{Fact}, (see 20.9 CGM Operating Conditions). For the value of V_{DDA}, choose the voltage potential at which the MCU is operating. If the power supply is variable, choose a value near the middle of the range of possible supply values.

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

4.9.4 Reaction Time Calculation

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

- Correct selection of filter capacitor, C_F (see 4.9.3 Choosing a Filter Capacitor).
- Room temperature operation
- Negligible external leakage on CGMXFC
- Negligible noise

The K factor in the equations is derived from internal PLL parameters. K_{ACQ} is the K factor when the PLL is configured in acquisition mode, and K_{TRK} is the K factor when the PLL is configured in tracking mode. (See 4.3.2.2 Acquisition and Tracking Modes).

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

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

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

NOTE

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

In automatic bandwidth control mode, the acquisition and lock times are quantized into units based on the reference frequency. (See 4.3.2.3 Manual and Automatic PLL Bandwidth Modes). A certain number of clock cycles, n_{ACQ}, is required to ascertain that the PLL is within the tracking mode entry tolerance, Δ_{TRK}, before exiting acquisition mode. A certain number of clock cycles, n_{TRK}, is required to ascertain that the PLL is within the lock mode entry tolerance, Δ_{Lock}. Therefore, the acquisition time, t_{ACQ}, is an integer

multiple of n_{ACQ}/f_{CGMRDV}, and the acquisition to lock time, t_{AL}, is an integer multiple of n_{TRK}/f_{CGMRDV}. 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 4.3.3 Base Clock Selector Circuit), because the factors described in 4.9.2 Parametric Influences on Reaction Time, may slow the lock time considerably.

When defining a limit in software for the maximum lock time, the value must allow for variation due to all of the factors mentioned in this section, especially due to the C_F capacitor and application specific influences.

The calculated lock time is only an indication and it is the customer's responsibility to allow enough of a guard band for their application. Prior to finalizing any software and while determining the maximum lock time, take into account all device to device differences. Typically, applications set the maximum lock time as an order of magnitude higher than the measured value. This is considered sufficient for all such device to device variation.

Freescale recommends measuring the lock time of the application system by utilizing dedicated software, running in FLASH, EEPROM, or RAM. This should toggle a port pin when the PLL is first configured and switched on, then again when it goes from acquisition to lock mode and finally again when the PLL lock bit is set. The resultant waveform can be captured on an oscilloscope and used to determine the typical lock time for the microcontroller and the associated external application circuit. For example, see Figure 4-7

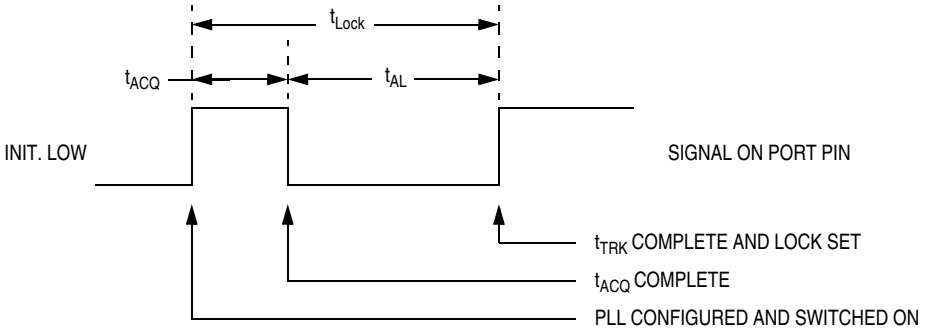


Figure 4-7. Typical Lock Time Waveform Example

NOTE

The filter capacitor should be fully discharged prior to making any measurements.

Chapter 5
Computer Operating Properly (COP) Module

5.1 Introduction

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

5.2 Functional Description

Figure 5-1 shows the structure of the COP module.

The COP counter is a free-running 6-bit counter preceded by the 12-bit system integration module (SIM) counter. COP timeouts are determined strictly by the CGM crystal oscillator clock signal (CGMXCLK), not the CGMOUT signal (see Figure 4-1. CGM Block Diagram).

If not cleared by software, the COP counter overflows and generates an asynchronous reset after 8176 or 262,128 CGMXCLK cycles, depending upon COPS bit in the MORA register ($001F). (See Chapter 10 Mask Options.) With a 4.9152-MHz crystal and the COPS bit in the MORA register ($001F) set to a 1, the COP timeout period is approximately 53.3 ms. Writing any value to location $FFFF before overflow occurs clears the COP counter, clears stages 12 through 5 of the SIM counter, and prevents reset. A CPU interrupt routine can be used to clear the COP.

NOTE

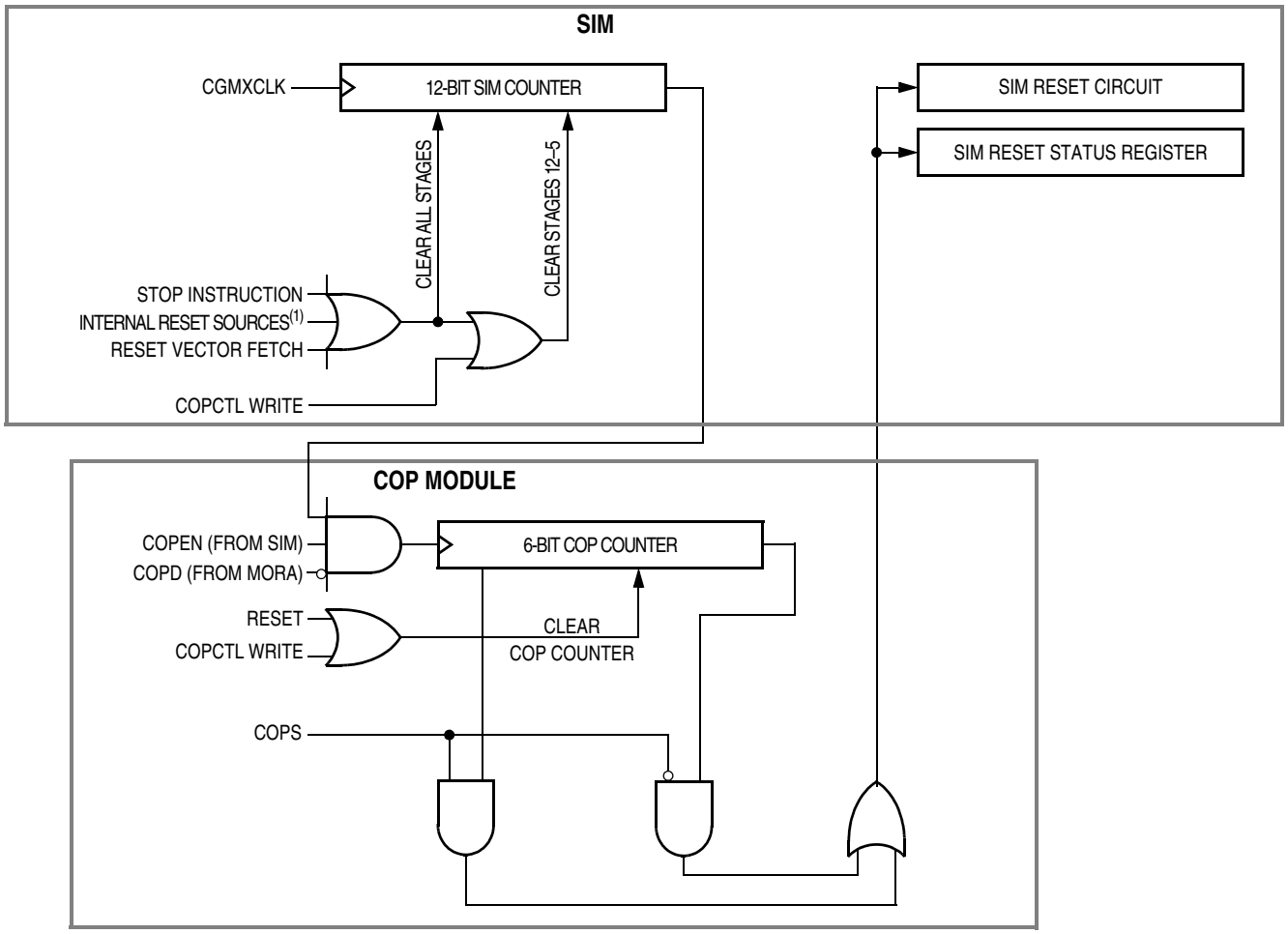
The COP should be serviced as soon as possible out of reset and before entering or after exiting stop mode to guarantee the maximum selected amount of time before the first timeout.

A COP reset pulls the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles and sets the COP bit in the SIM reset status register (SRSR). See 15.7.2 SIM Reset Status Register (SRSR).

While the microcontroller is in monitor mode, the COP module is disabled if the $\overline{\text{RST}}$ pin or the $\overline{\text{IRQ}}$ pin is held at V_{TST} (see 20.5 5.0 Volt DC Electrical Characteristics). During a break state, V_{TST} on the $\overline{\text{RST}}$ pin disables the COP module.

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. The one exception to this is wait mode (see 5.7.1 Wait Mode).



1. See 15.3.2 Active Resets From Internal Sources.

Figure 5-1. COP Block Diagram

5.3 I/O Signals

The following paragraphs describe the signals shown in Figure 5-1.

5.3.1 CGMXCLK

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

5.3.2 STOP Instruction

The STOP instruction clears the COP prescaler.

5.3.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (5.4 COP Control Register (COPCTL)), clears the COP counter and clears stages 12 through 4 of the COP prescaler. Reading the COP control register returns the reset vector.

5.3.4 Power-On Reset

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

5.3.5 Internal Reset

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

5.3.6 Reset Vector Fetch

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

5.3.7 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the mask option register (MORA). See Chapter 10 Mask Options.

5.3.8 COPRS (COP Rate Select)

The COPRS signal reflects the state of the COP rate select bit, COPRS in the MORA register (see Figure 10-1. Mask Option Register A (MORA)).

5.4 COP Control Register (COPCTL)

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

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 5-2. COP Control Register (COPCTL)

5.5 Interrupts

The COP does not generate CPU interrupt requests.

5.6 Monitor Mode

The COP is disabled in monitor mode when V_{TST} (see 20.5 5.0 Volt DC Electrical Characteristics) is present on the $\overline{IRQ}$ pin or on the $\overline{RST}$ pin.

5.7 Low-Power Modes

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

5.7.1 Wait Mode

The COP remains active during wait mode. If COP is enabled, a reset will occur at COP timeout.

5.7.2 Stop Mode

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

The STOP bit in the MORA register ($001F) (see Chapter 10 Mask Options) enables the STOP instruction. To prevent inadvertently turning off the COP with a STOP instruction, disable the STOP instruction by programming the STOP bit to 0.

5.8 COP Module During Break Interrupts

The COP is disabled during a break interrupt when V_{TST} (see 20.5 5.0 Volt DC Electrical Characteristics) is present on the $\overline{RST}$ pin.

Chapter 6
Central Processor Unit (CPU)

6.1 Introduction

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

6.2 Features

Features of the CPU include:

- Object code fully upward-compatible with M68HC05 Family
- 16-bit stack pointer with stack manipulation instructions
- 16-bit index register with x-register manipulation instructions
- 8-MHz CPU internal bus frequency
- 64-Kbyte program/data memory space
- 16 addressing modes
- Memory-to-memory data moves without using accumulator
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- Enhanced binary-coded decimal (BCD) data handling
- Modular architecture with expandable internal bus definition for extension of addressing range beyond 64 Kbytes
- Low-power stop and wait modes

6.3 CPU Registers

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

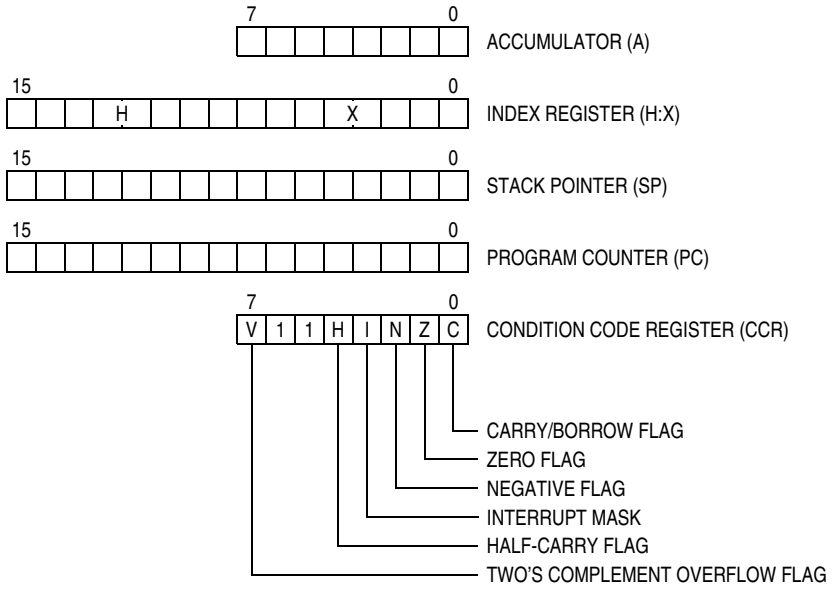


Figure 6-1. CPU Registers

6.3.1 Accumulator

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

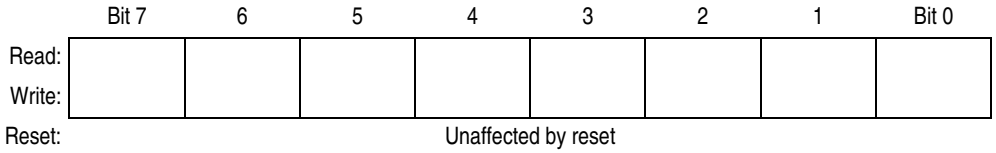


Figure 6-2. Accumulator (A)

6.3.2 Index Register

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

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

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

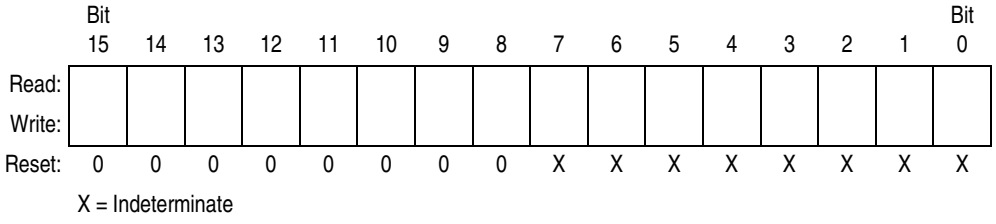


Figure 6-3. Index Register (H:X)

6.3.3 Stack Pointer

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

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

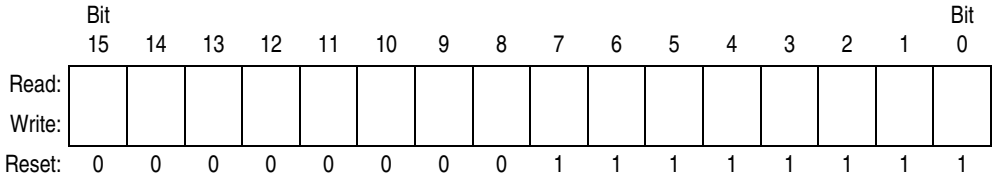


Figure 6-4. Stack Pointer (SP)

NOTE

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

6.3.4 Program Counter

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

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

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

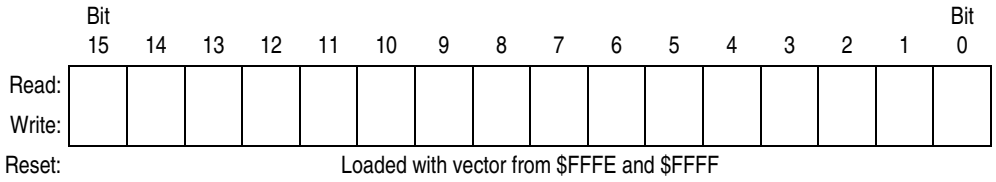


Figure 6-5. Program Counter (PC)

6.3.5 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 1. The following paragraphs describe the functions of the condition code register.

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

X = Indeterminate

Figure 6-6. Condition Code Register (CCR)

V — Overflow Flag

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

1 = Overflow
0 = No overflow

H — Half-Carry Flag

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

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

I — Interrupt Mask

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

1 = Interrupts disabled
0 = Interrupts enabled

NOTE

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

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

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

N — Negative Flag

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

1 = Negative result
0 = Non-negative result

Z — Zero Flag

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

1 = Zero result
0 = Non-zero result

C — Carry/Borrow Flag

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

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

6.4 Arithmetic/Logic Unit (ALU)

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

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

6.5 Low-Power Modes

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

6.5.1 Wait Mode

The WAIT instruction:

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

6.5.2 Stop Mode

The STOP instruction:

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

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

6.6 CPU During Break Interrupts

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

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

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

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

6.7 Instruction Set Summary

Table 6-1 provides a summary of the M68HC08 instruction set.

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

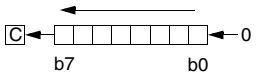
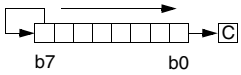
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X ADC *opr*,SP ADC *opr*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
ADD #*opr* ADD *opr* ADD *opr* ADD *opr*,X ADD *opr*,X ADD ,X ADD *opr*,SP ADD *opr*,SP	Add without Carry	A ← (A) + (M)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	AB BB CB DB EB FB 9EEB 9EDB	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
AIS #*opr*	Add Immediate Value (Signed) to SP	SP ← (SP) + (16 « M)	–	–	–	–	–	–	IMM	A7	ii	2
AIX #*opr*	Add Immediate Value (Signed) to H:X	H:X ← (H:X) + (16 « M)	–	–	–	–	–	–	IMM	AF	ii	2
AND #*opr* AND *opr* AND *opr* AND *opr*,X AND *opr*,X AND ,X AND *opr*,SP AND *opr*,SP	Logical AND	A ← (A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A4 B4 C4 D4 E4 F4 9EE4 9ED4	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
ASL *opr* ASLA ASLX ASL *opr*,X ASL ,X ASL *opr*,SP	Arithmetic Shift Left (Same as LSL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
ASR *opr* ASRA ASRX ASR *opr*,X ASR *opr*,X ASR *opr*,SP	Arithmetic Shift Right		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	4 1 1 4 3 5
BCC *rel*	Branch if Carry Bit Clear	PC ← (PC) + 2 + rel ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BCLR *n*, *opr*	Clear Bit n in M	Mn ← 0	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BEQ *rel*	Branch if Equal	PC ← (PC) + 2 + *rel* ? (Z) = 1	–	–	–	–	–	–	REL	27	rr	3
BGE *opr*	Branch if Greater Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) = 0	–	–	–	–	–	–	REL	90	rr	3
BGT *opr*	Branch if Greater Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) ǀ (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) ǀ (Z) = 0	–	–	–	–	–	–	REL	22	rr	3

Table 6-1. Instruction Set Summary (Sheet 2 of 6)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BHS *rel*	Branch if Higher or Same (Same as BCC)	PC ← (PC) + 2 + *rel* ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if $\overline{\text{IRQ}}$ Pin High	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if $\overline{\text{IRQ}}$ Pin Low	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #*opr* BIT *opr* BIT *opr* BIT *opr*,X BIT *opr*,X BIT ,X BIT *opr*,SP BIT *opr*,SP	Bit Test	(A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A5 B5 C5 D5 E5 F5 9EE5 9ED5	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
BLE *opr*	Branch if Less Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) ∣ (N ⊕ V) = 1	–	–	–	–	–	–	REL	93	rr	3
BLO *rel*	Branch if Lower (Same as BCS)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BLS *rel*	Branch if Lower or Same	PC ← (PC) + 2 + *rel* ? (C) ∣ (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT *opr*	Branch if Less Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) =1	–	–	–	–	–	–	REL	91	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	PC ← (PC) + 2 + *rel* ? (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI *rel*	Branch if Minus	PC ← (PC) + 2 + *rel* ? (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	PC ← (PC) + 2 + *rel* ? (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + 2 + *rel* ? (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL *rel*	Branch if Plus	PC ← (PC) + 2 + *rel* ? (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA *rel*	Branch Always	PC ← (PC) + 2 + *rel*	–	–	–	–	–	–	REL	20	rr	3
BRCLR *n*,*opr*,*rel*	Branch if Bit *n* in M Clear	PC ← (PC) + 3 + *rel* ? (Mn) = 0	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	PC ← (PC) + 2	–	–	–	–	–	–	REL	21	rr	3
BRSET *n*,*opr*,*rel*	Branch if Bit *n* in M Set	PC ← (PC) + 3 + *rel* ? (Mn) = 1	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n*,*opr*	Set Bit *n* in M	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	4
CBEQ *opr*,*rel* CBEQA #*opr*,*rel* CBEQX #*opr*,*rel* CBEQ *opr*,X+,*rel* CBEQ X+,*rel* CBEQ *opr*,SP,*rel*	Compare and Branch if Equal	PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (X) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 2 + rel ? (A) – (M) = $00 PC ← (PC) + 4 + rel ? (A) – (M) = $00	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 4 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	I ← 0	–	–	0	–	–	–	INH	9A		2

Table 6-1. Instruction Set Summary (Sheet 3 of 6)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
CLR *opr* CLRA CLRX CLRH CLR *opr*,X CLR ,X CLR *opr*,SP	Clear	M ← $00 A ← $00 X ← $00 H ← $00 M ← $00 M ← $00 M ← $00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	3 1 1 1 3 2 4
CMP #*opr* CMP *opr* CMP *opr* CMP *opr*,X CMP *opr*,X CMP ,X CMP *opr*,SP CMP *opr*,SP	Compare A with M	(A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A1 B1 C1 D1 E1 F1 9EE1 9ED1	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
COM *opr* COMA COMX COM *opr*,X COM ,X COM *opr*,SP	Complement (One's Complement)	M ← ($\overline{M}$) = $FF – (M) A ← ($\overline{A}$) = $FF – (M) X ← ($\overline{X}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M)	0	–	–	↕	↕	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff	4 1 1 4 3 5
CPHX #*opr* CPHX *opr*	Compare H:X with M	(H:X) – (M:M + 1)	↕	–	–	↕	↕	↕	IMM DIR	65 75	ii ii+1 dd	3 4
CPX #*opr* CPX *opr* CPX *opr* CPX ,X CPX *opr*,X CPX *opr*,X CPX *opr*,SP CPX *opr*,SP	Compare X with M	(X) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A3 B3 C3 D3 E3 F3 9EE3 9ED3	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
DAA	Decimal Adjust A	$(A)_{10}$	U	–	–	↕	↕	↕	INH	72		2
DBNZ *opr,rel* DBNZA *rel* DBNZX *rel* DBNZ *opr*,X,*rel* DBNZ X,*rel* DBNZ *opr*,SP,*rel*	Decrement and Branch if Not Zero	A ← (A) – 1 or M ← (M) – 1 or X ← (X) – 1 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 4 + *rel* ? (result) ≠ 0	–	–	–	–	–	–	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr rr ff rr	5 3 3 5 4 6
DEC *opr* DECA DECX DEC *opr*,X DEC ,X DEC *opr*,SP	Decrement	M ← (M) – 1 A ← (A) – 1 X ← (X) – 1 M ← (M) – 1 M ← (M) – 1 M ← (M) – 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff	4 1 1 4 3 5
DIV	Divide	A ← (H:A)/(X) H ← Remainder	–	–	–	–	↕	↕	INH	52		7
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X EOR *opr*,SP EOR *opr*,SP	Exclusive OR M with A	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9EE8 9ED8	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
INC *opr* INCA INCX INC *opr*,X INC ,X INC *opr*,SP	Increment	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1 M ← (M) + 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	4 1 1 4 3 5

Table 6-1. Instruction Set Summary (Sheet 4 of 6)

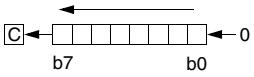
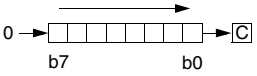
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR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

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

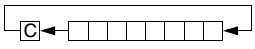
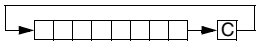
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–	–	INH	88		2
ROL *opr* ROLA ROLX ROL *opr*,X ROL ,X ROL *opr*,SP	Rotate Left through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	4 1 1 4 3 5
ROR *opr* RORA RORX ROR *opr*,X ROR ,X ROR *opr*,SP	Rotate Right through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	4 1 1 4 3 5
RSP	Reset Stack Pointer	SP ← $FF	–	–	–	–	–	–	INH	9C		1
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 Interrupts, Stop Processing, Refer to MCU Documentation	I ← 0; Stop Processing	–	–	0	–	–	–	INH	8E		1
STX *opr* STX *opr* STX *opr*,X STX *opr*,X STX ,X STX *opr*,SP STX *opr*,SP	Store X in M	M ← (X)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	BF CF DF EF FF 9EEF 9EDF	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 5
SUB #*opr* SUB *opr* SUB *opr* SUB *opr*,X SUB *opr*,X SUB ,X SUB *opr*,SP SUB *opr*,SP	Subtract	A ← (A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A0 B0 C0 D0 E0 F0 9EE0 9ED0	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5

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

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
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
WAIT	Enable Interrupts; Wait for Interrupt	I bit ← 0; Inhibit CPU clocking until interrupted	–	–	0	–	–	–	INH	8F		1

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

6.8 Opcode Map

See Table 6-2.

Table 6-2. Opcode Map

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

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

*Pre-byte for stack pointer indexed instructions

MSB / LSB	0
0	5 BRSET0 3 DIR

Low Byte of Opcode in Hexadecimal

High Byte of Opcode in Hexadecimal
Cycles
Opcode Mnemonic
Number of Bytes / Addressing Mode

Chapter 7
External Interrupt (IRQ) Module

7.1 Introduction

The external interrupt (IRQ) module provides the nonmaskable interrupt input.

7.2 Features

Features include:

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

7.3 Functional Description

A falling edge applied to any of the external interrupt pins can latch a CPU interrupt request. Figure 7-1 shows the structure of the IRQ module.

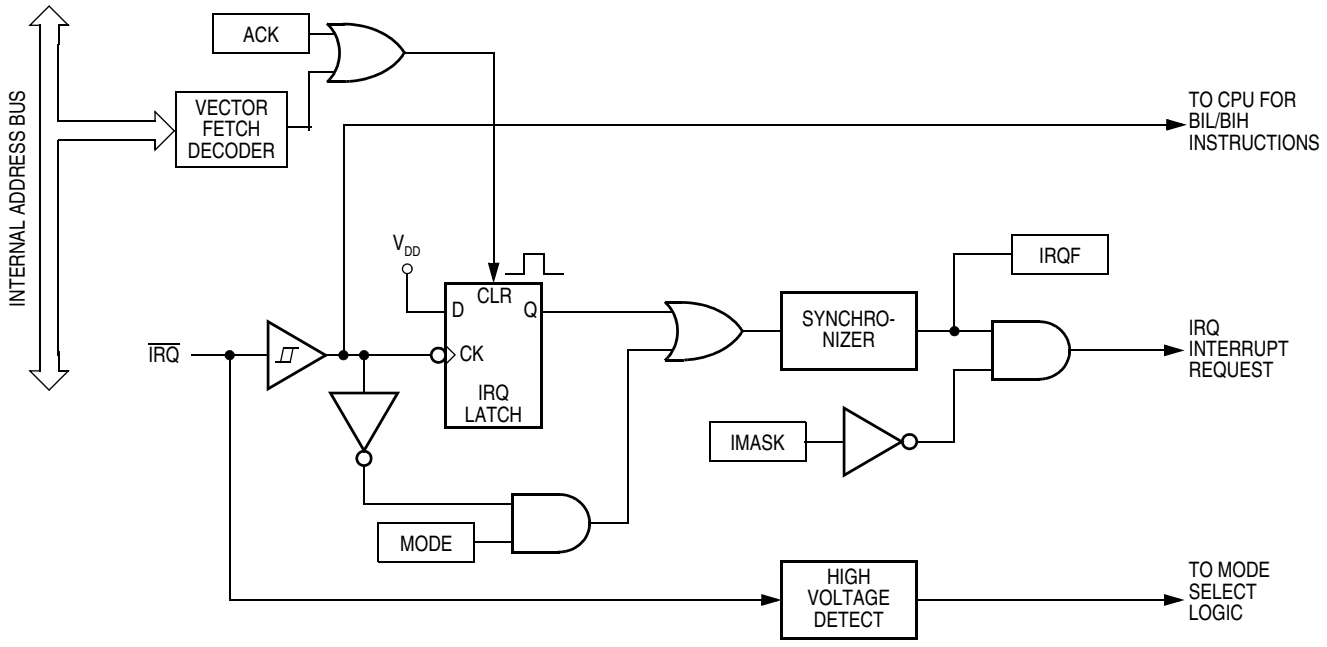


Figure 7-1. IRQ Module Block Diagram

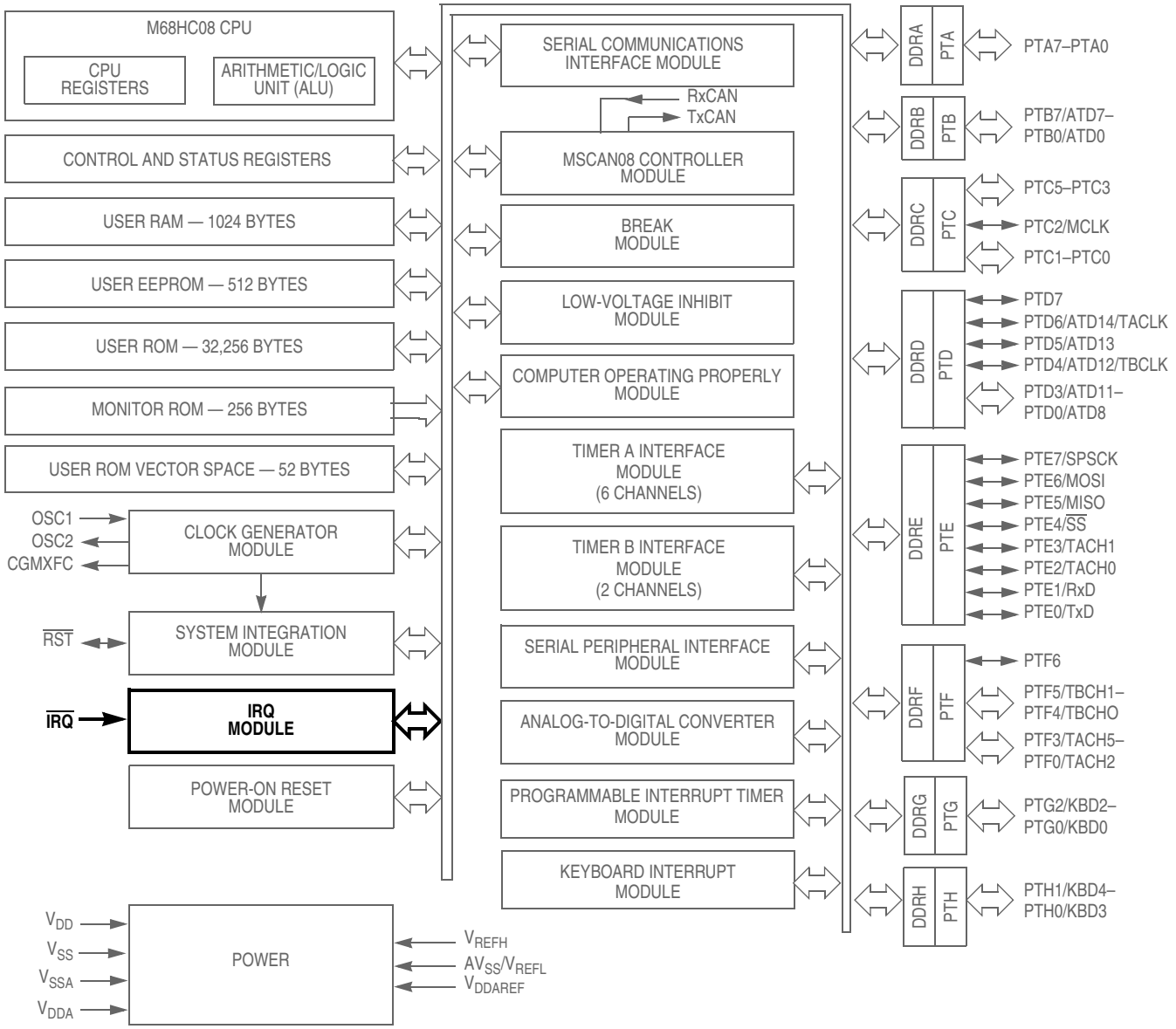


Figure 7-2. Block Diagram Highlighting IRQ Block and Pins

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001A	IRQ Status and Control Register (ISCR) See page 97.	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 7-3. IRQ I/O Register Summary

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

- Vector fetch — a vector fetch automatically generates an interrupt acknowledge signal which clears the latch that caused the vector fetch.
- Software clear — software can clear an interrupt latch by writing to the appropriate acknowledge bit in the interrupt status and control register (ISCR). Writing a 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 both falling-edge and low-level-triggered. The MODE bit in the ISCR controls the triggering sensitivity of the $\overline{\text{IRQ}}$ pin.

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

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

- Vector fetch or software clear
- Return of the interrupt pin to a high level

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

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

NOTE

The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. See Figure 7-4.

7.4 $\overline{\text{IRQ}}$ Pin

A falling edge 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 the IRQ latch:

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

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

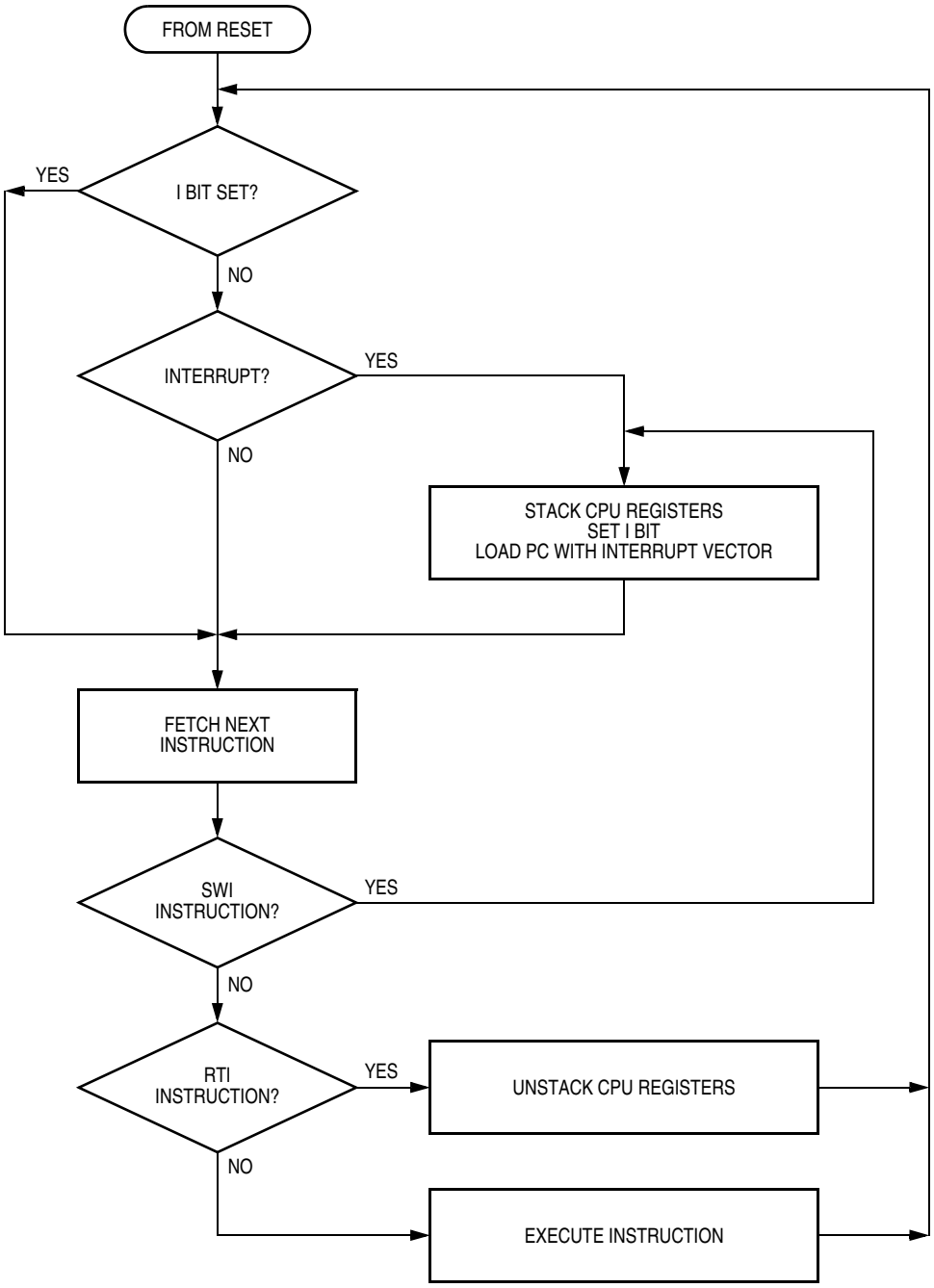


Figure 7-4. IRQ Interrupt Flowchart

If the MODE bit is clear, the $\overline{IRQ}$ 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.

The BIH or BIL instruction is used to read the logic level on the $\overline{IRQ}$ pin.

NOTE

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

7.5 IRQ Module During Break Interrupts

The system integration module (SIM) controls whether the IRQ interrupt latch can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear the latches during the break state. See 15.7.3 SIM Break Flag Control Register (SBFCR).

To allow software to clear the IRQ latch during a break interrupt, write a 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 0 to the BCFE bit. With BCFE at 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.

7.6 IRQ Status and Control Register (ISCR)

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

- Indicates the state of the IRQ interrupt flag
- Clears the IRQ interrupt latch
- Masks IRQ interrupt requests
- Controls triggering sensitivity of the $\overline{IRQ}$ 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 7-5. IRQ Status and Control Register (ISCR)

IRQF — $\overline{IRQ}$ Flag Bit

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

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

ACK — $\overline{IRQ}$ Interrupt Request Acknowledge Bit

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

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

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

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

MODE — $\overline{\text{IRQ}}$ Edge/Level Select Bit

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

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

Chapter 8
Keyboard Interrupt (KBD) Module

8.1 Introduction

The keyboard interrupt module (KBD) provides five independently maskable external interrupt pins.

8.2 Features

Features include:

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

8.3 Functional Description

Writing to the KBIE4–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port G or port H pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its internal pullup device. A low level applied to an enabled keyboard interrupt pin latches a keyboard interrupt request.

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

- If the keyboard interrupt is edge-sensitive only, a falling edge on a keyboard pin does not latch an interrupt request if another keyboard pin is already low. To prevent losing an interrupt request on one pin because another pin is still low, software can disable the latter pin while it is low.
- If the keyboard interrupt is falling edge- and low-level sensitive, an interrupt request is present as long as any keyboard pin is low.

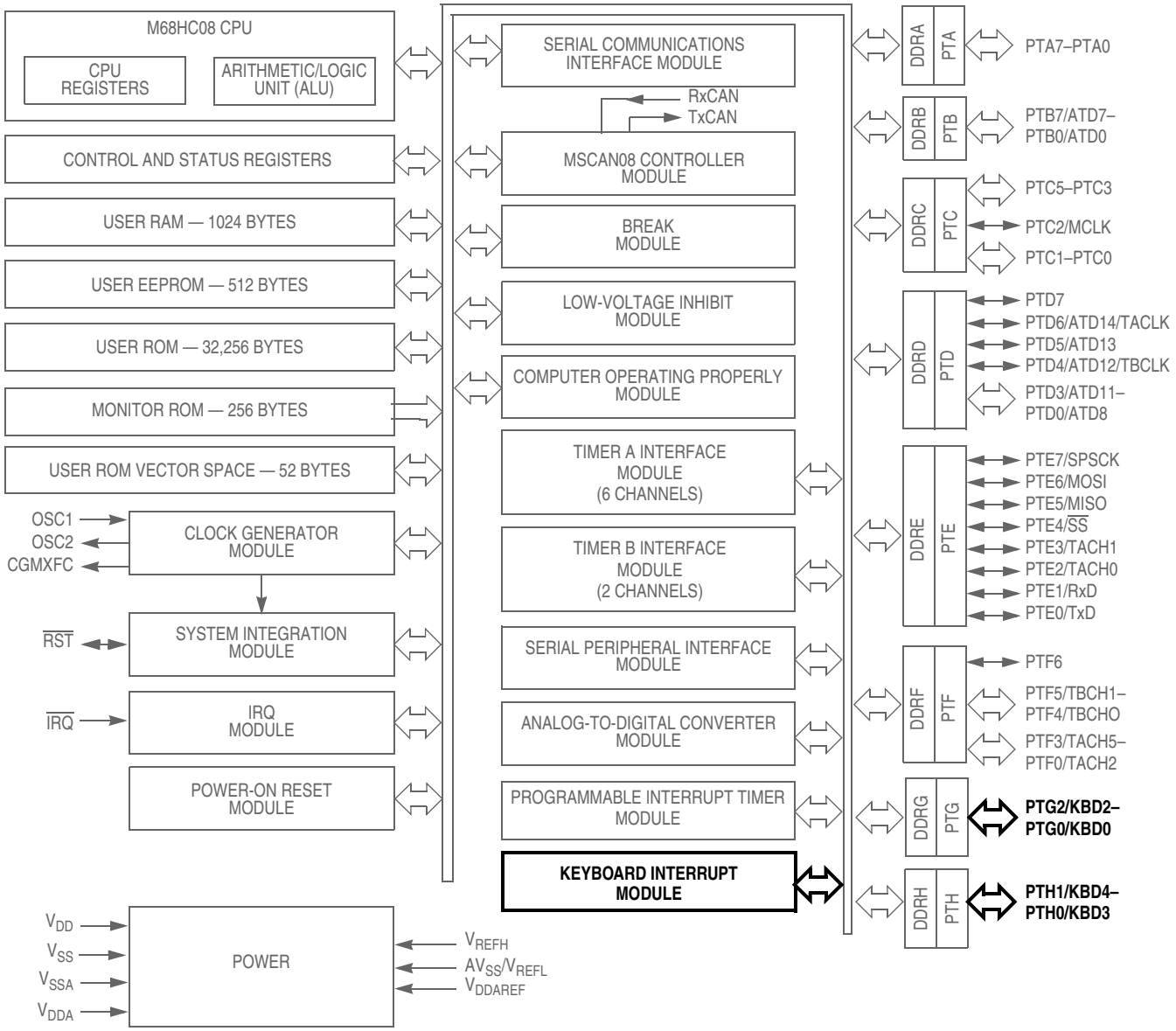


Figure 8-1. Block Diagram Highlighting KBD Block and Pins

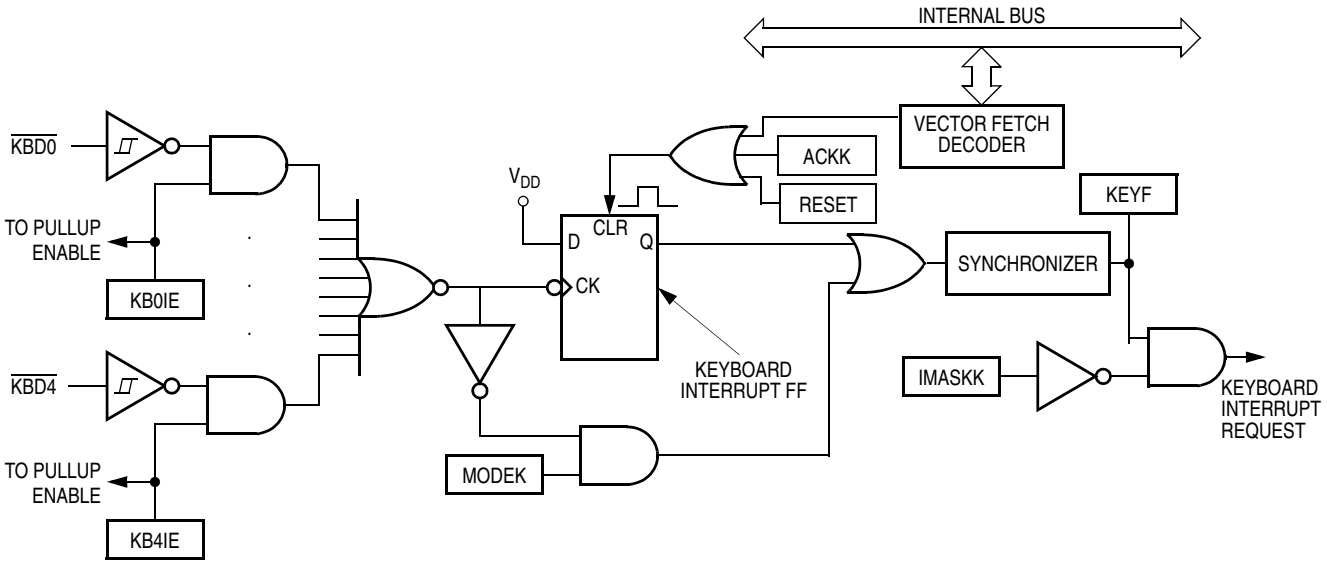


Figure 8-2. Keyboard Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001B	Keyboard Status and Control Register (KBSCR) See page 103.	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) See page 104.	Read:	0	0	0	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 8-3. I/O Register Summary

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

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software may generate the interrupt acknowledge signal by writing a 1 to the ACKK bit in the keyboard status and control register (KBSCR). The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit prior to leaving an interrupt service routine also can prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the KBI vector address.
- Return of all enabled keyboard interrupt pins to a high level. As long as any enabled keyboard interrupt pin is low, the keyboard interrupt remains set.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to a high level 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 low.

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

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

NOTE

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

8.4 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup to reach a 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 DDRG bits in data direction register G.
2. Configure the keyboard pins as outputs by setting the appropriate DDRH bits in data direction register H.
3. Write 1s to the appropriate port G and port H data register bits.
4. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.

8.5 Low-Power Modes

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

8.5.1 Wait Mode

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

8.5.2 Stop Mode

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

8.6 Keyboard Module During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See 19.2 Break Module (BRK).

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

To protect the KEYF bit during the break state, write a 0 to the BCFE bit. With BCFE at 0, writing to the keyboard acknowledge bit (ACKK) in the keyboard status and control register during the break state has no effect. See 8.7.1 Keyboard Status and Control Register.

8.7 I/O Registers

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

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

8.7.1 Keyboard Status and Control Register

The keyboard status and control register:

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

Address: $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 8-4. Keyboard Status and Control Register (KBSCR)

Bits 7–4 — Not used

These read-only bits always read as 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 1 to this write-only bit clears the keyboard interrupt request. ACKK always reads as 0. Reset clears ACKK.

IMASKK — Keyboard Interrupt Mask Bit

Writing a 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

8.7.2 Keyboard Interrupt Enable Register

The keyboard interrupt enable register enables or disables each port G and each port H 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:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 8-5. Keyboard Interrupt Enable Register (KBIER)

KBIE4–KBIE0 — Keyboard Interrupt Enable Bits

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

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

Chapter 9
Low-Voltage Inhibit (LVI) Module)

9.1 Introduction

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

9.2 Features

Features include:

- Programmable LVI reset
- Programmable power consumption
- Digital filtering of V_{DD} pin level

NOTE

If a low-voltage interrupt (LVI) occurs during programming of EEPROM memory, then adequate programming time may not have been allowed to ensure the integrity and retention of the data. It is the responsibility of the user to ensure that in the event of an LVI any addresses being programmed receive specification programming conditions.

9.3 Functional Description

Figure 9-1 shows the structure of the LVI module. The LVI module contains a bandgap reference circuit and comparator. The LVI power bit, LVIPWR, enables the LVI to monitor V_{DD} voltage. The LVI reset bit, LVIRST, enables the LVI module to generate a reset when V_{DD} falls below a voltage, LVI_{TRIPF}, and remains at or below that level for nine or more consecutive CPU cycles.

NOTE

Note that short V_{DD} spikes may not trip the LVI. It is the user's responsibility to ensure a clean V_{DD} signal within the specified operating voltage range if normal microcontroller operation is to be guaranteed.

LVISTOP enables the LVI module during stop mode. This will ensure when the STOP instruction is implemented the LVI will continue to monitor the voltage level on V_{DD}.

LVIPWR, LVIRST, and LVISTOP are mask options (see Chapter 10 Mask Options). Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises above a voltage, LVI_{TRIPR}. V_{DD} must be above LVI_{TRIPR} for only one CPU cycle to bring the MCU out of reset. The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

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

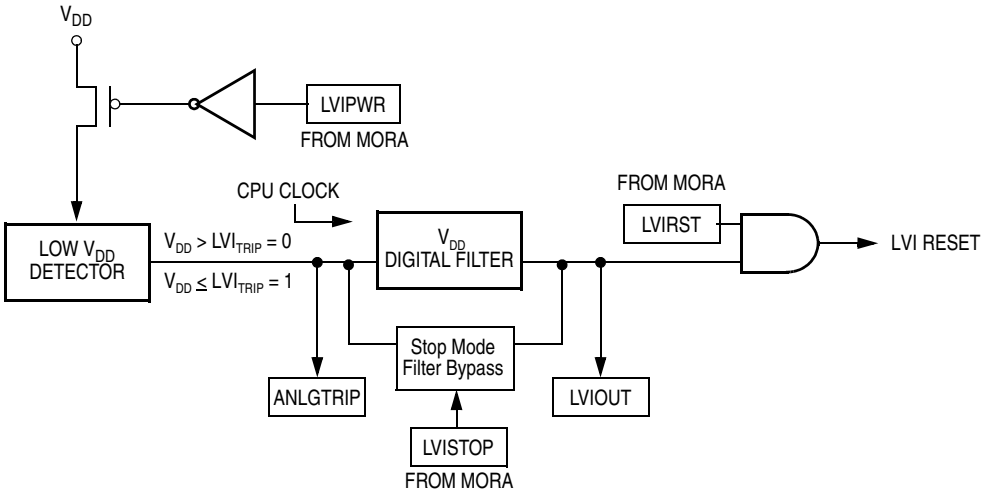


Figure 9-1. LVI Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE0F	LVI Status Register (LVISR) See page 107.	Read:	LVIOUT	0	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
			= Unimplemented							

Figure 9-2. I/O Register Summary

9.3.1 Polled LVI Operation

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

9.3.2 Forced Reset Operation

In applications that require V_{DD} to remain above the LVI_{TRIPF} level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls to the LVI_{TRIPF} level and remains at or below that level for nine or more consecutive CPU cycles. In the mask option register, the LVIPWR and LVIRST bits must be 1s to enable the LVI module and to enable LVI resets.

9.3.3 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 nine or more consecutive CPU cycles. V_{DD} must be above LVI_{TRIPR} for only one CPU cycle to bring the MCU out of reset.

9.4 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 9-3. LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

This read-only flag becomes set when V_{DD} falls below the LVI_{TRIPF} voltage for 32–40 CGMXCLK cycles. (See Table 9-1). Reset clears the LVIOUT bit.

Table 9-1. 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
$V_{DD} < LVI_{TRIPF}$	between 32 and 40 CGMXCLK cycles	0 or 1
$V_{DD} < LVI_{TRIPF}$	> 40 CGMXCLK cycles	1
$LVI_{TRIPF} < V_{DD} < LVI_{TRIPR}$	Any	Previous value

9.5 LVI Interrupts

The LVI module does not generate interrupt requests.

9.6 Low-Power Modes

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

9.6.1 Wait Mode

With the LVIPWR bit in the MORA register programmed to 1, the LVI module is active after a WAIT instruction.

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

9.6.2 Stop Mode

With LVISTOP = 1 and LVIPWR = 1 in the MORA register, the LVI module will be active after a STOP instruction. Because CPU clocks are disabled during stop mode, the LVI trip must bypass the digital filter to generate a reset and bring the MCU out of stop.

With the LVIPWR bit in the MORA register at a 1 and the LVISTOP bit at a 0, the LVI module will be inactive after a STOP instruction.

NOTE

The LVI feature is intended to provide the safe shutdown of the microcontroller and thus protection of related circuitry prior to any application V_{DD} voltage collapsing completely to an unsafe level. Is is not intended that users operate the microcontroller at lower than the specified operating voltage, V_{DD}.

Chapter 10
Mask Options

10.1 Introduction

This section describes the mask options and the mask option registers. The mask options are hardwired connections specified at the same time as the ROM code, which allow the user to customize the MCU. The options control the enable or disable of the following functions:

- Resets caused by the LVI module
- Power to the LVI module
- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- ROM security(1)
- STOP instruction
- Computer operating properly (COP) module enable
- EEPROM read protection(1)
- COP time-out period
- EEPROM reference clock source

10.2 Functional Description

Address: $001F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LVISTOP	ROMSEC	LVIRST	LVIPWR	SSREC	COPRS	STOP	COPD
Write:								
Reset:	Unaffected by reset							

= Unimplemented

Figure 10-1. Mask Option Register A (MORA)

LVISTOP — LVI Stop Mode Enable Bit

LVISTOP enables the LVI module in stop mode. See Chapter 9 Low-Voltage Inhibit (LVI) Module).

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

ROMSEC — ROM Security Bit

ROMSEC enables the ROM security feature. Setting the ROMSEC bit prevents access to the ROM contents.

1 = ROM security enabled
0 = ROM security disabled

1. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the ROM/EEPROM data difficult for unauthorized users.

LVIRST — LVI Reset Enable Bit

LVIRST enables the reset signal from the LVI module. See Chapter 9 Low-Voltage Inhibit (LVI) Module.

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

LVIPWR — LVI Power Enable Bit

LVIPWR enables the LVI module. See Chapter 9 Low-Voltage Inhibit (LVI) Module.

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

SSREC — Short Stop Recovery Bit

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

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

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

COPRS — COP Rate Select

COPRS is similar to COPL (please note that the logic is reversed) as it determines the timeout period for the COP.

1 = COP timeout period is 8176 CGMXCLK cycles.
0 = COP timeout period is 262,128 CGMXCLK cycles.

STOP — STOP Enable Bit

STOP enables the STOP instruction.

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

COPD — COP Disable Bit

COPD disables the COP module. See Chapter 5 Computer Operating Properly (COP) Module.

1 = COP module disabled
0 = COP module enabled

NOTE

Extra care should be exercised when selecting mask option registers since other M68HC08 Family parts may have different options. It is the user's responsibility to correctly define the mask option registers. If in doubt, check with your local field applications representative.

Address: $FE09

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EEDIVCLK	0	EESEC	EEMONSEC	AZ32A	0	0	0
Write:								
Reset:	Unaffected by reset							

= Unimplemented

Figure 10-2. Mask Option Register B (MORB)

EEDIVCLK — EEPROM Timebase Divider Clock Select Bit

EEDIVCLK selects the reference clock source for the EEPROM timebase divider. See 2.7.3.4 EEPROM Timebase Divider Register.

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

EESEC

This bit has no function.

EEMONSEC — EEPROM Read Protection in Monitor Mode Bit

When EEMONSEC is set the entire EEPROM array cannot be accessed in monitor mode unless a valid security code is entered.

1 = EEPROM read protection in monitor mode enabled.
0 = EEPROM read protection in monitor mode disabled.

AZ32A — Device Indicator

This bit is used to distinguish a MC68HC08AZ32A from older non-'A' suffix versions and always reads as 1.

1 = 'A' version
0 = Non-'A' version

NOTE

Extra care should be exercised when selecting mask option registers since other M68HC08 Family parts may have different options. It is the user's responsibility to correctly define the mask option registers. If in doubt, check with your local field applications representative.

Chapter 11
MSCAN08 Controller (MSCAN08)

11.1 Introduction

The MSCAN08 is the specific implementation of the MSCAN concept targeted for the Freescale M68HC08 Microcontroller Family. The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September 1991.

The CAN protocol was primarily, but not exclusively, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the electromagnetic interference (EMI) environment of a vehicle, cost-effectiveness and required bandwidth. MSCAN08 utilizes an advanced buffer arrangement, resulting in a predictable real-time behavior, and simplifies the application software.

11.2 Features

Basic features are:

- Modular architecture
- Implementation of the CAN Protocol — Version 2.0A/B
  - Standard and extended data frames
  - 0–8 bytes data length
  - Programmable bit rate up to 1 Mbps depending on the actual bit timing and the clock jitter of the PLL
- Support for remote frames
- Double-buffered receive storage scheme
- Triple-buffered transmit storage scheme with internal prioritization using a local priority concept
- Flexible maskable identifier filter supports alternatively one full size extended identifier filter, two 16-bit filters, or four 8-bit filters
- Programmable wakeup functionality with integrated low-pass filter
- Programmable loop-back mode supports self-test operation
- Separate signalling and interrupt capabilities for all CAN receiver and transmitter error states (warning, error passive, bus off)
- Programmable MSCAN08 clock source either CPU bus cock or crystal oscillator output
- Programmable link to on-chip timer interface module (TIMB) for time-stamping and network synchronization
- Low-power sleep mode

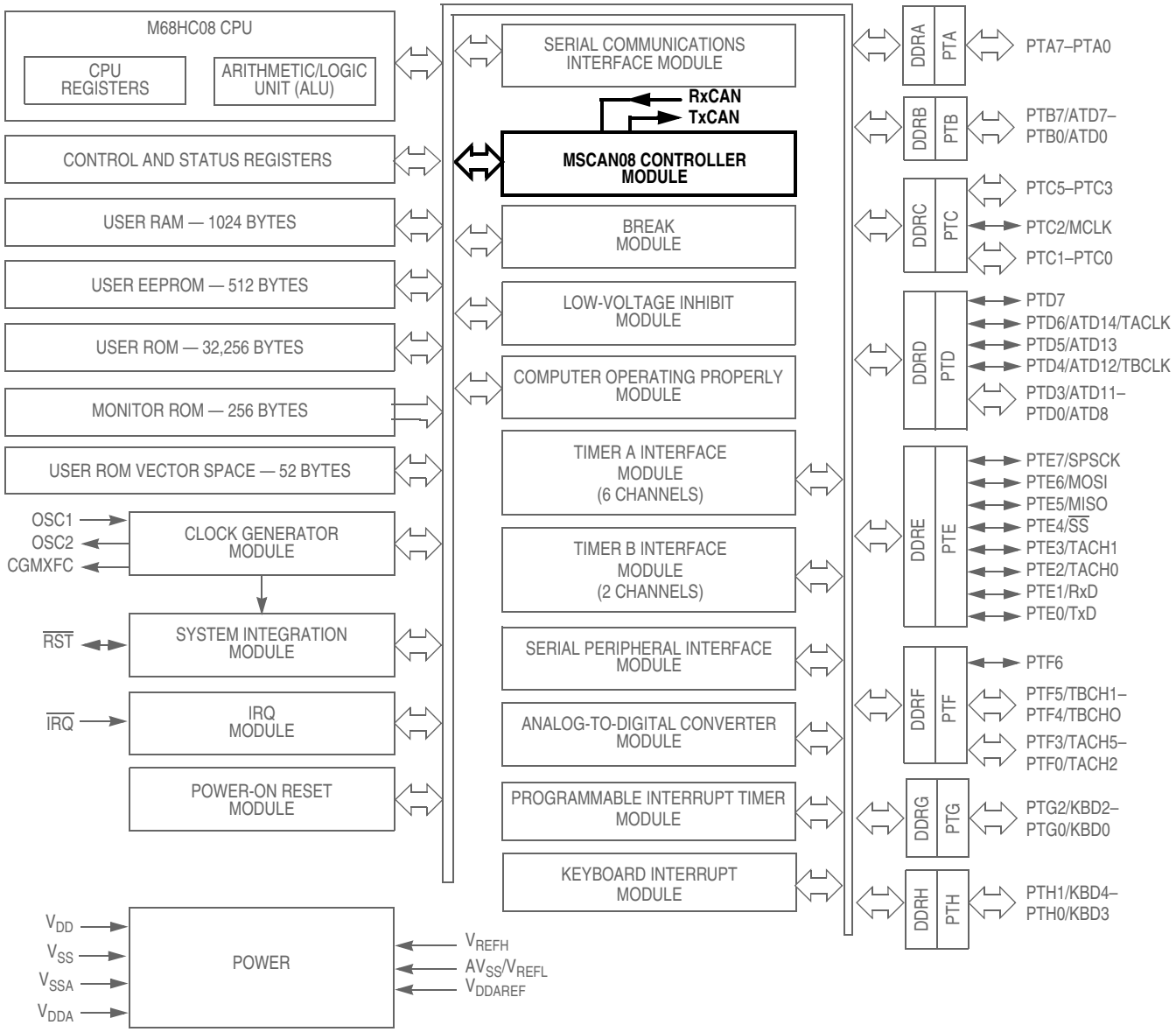


Figure 11-1. Block Diagram Highlighting MSCAN08 Block and Pins

11.3 External Pins

The MSCAN08 uses two external pins, one input (RxCAN) and one output (TxCAN). The TxCAN output pin represents the logic level on the CAN: 0 is for a dominant state and 1 is for a recessive state.

A typical CAN system with MSCAN08 is shown in Figure 11-2.

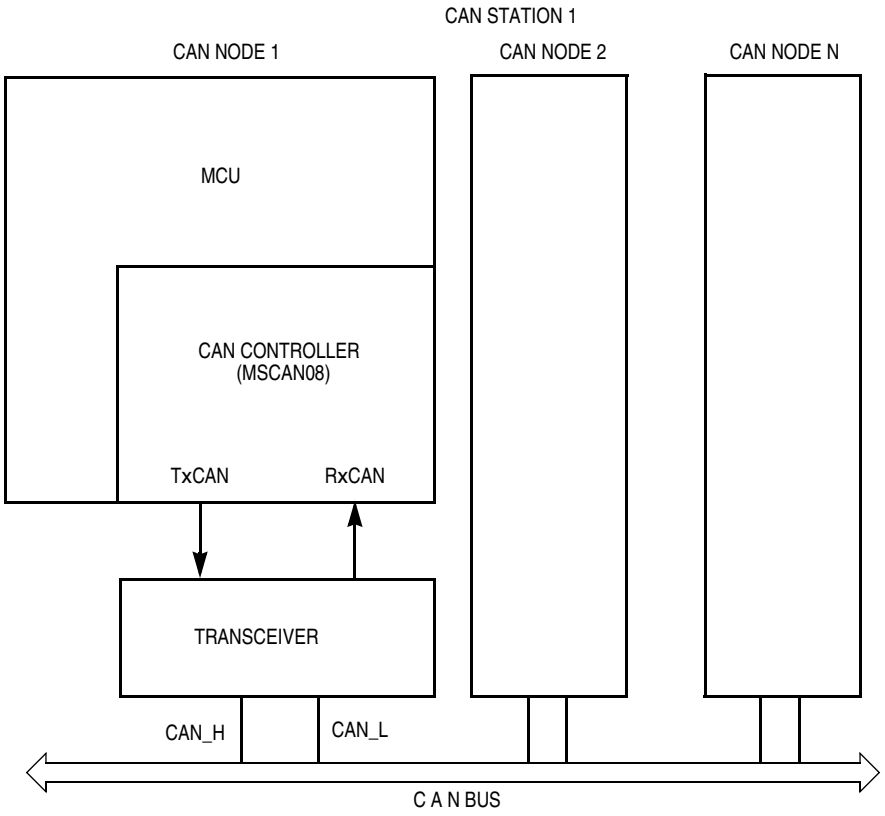


Figure 11-2. The CAN System

Each CAN station is connected physically to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN and has current protection against defective CAN or defective stations.

11.4 Message Storage

MSCAN08 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

11.4.1 Background

Modern application layer software is built under two fundamental assumptions:

1. Any CAN node is able to send out a stream of scheduled messages without releasing the bus between two messages. Such nodes will arbitrate for the bus right after sending the previous message and will only release the bus in case of lost arbitration.
2. The internal message queue within any CAN node is organized as such that the highest priority message will be sent out first if more than one message is ready to be sent.

Above behavior cannot be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message has been sent. This loading process lasts a definite amount of time and has to be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme would de-couple the re-loading of the transmit buffers from the actual message being sent and as such reduces the reactiveness requirements on the CPU. Problems may arise if the sending of a message would be finished just while the CPU re-loads the second buffer. In that case, no buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN08 has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN08 implements with the local priority concept described in 11.4.2 Receive Structures.

11.4.2 Receive Structures

The received messages are stored in a 2-stage input first in first out (FIFO). The two message buffers are mapped using a ping pong arrangement into a single memory area (see Figure 11-3). While the background receive buffer (RxBG) is exclusively associated to the MSCAN08, the foreground receive buffer (RxFG) is addressable by the CPU08. This scheme simplifies the handler software, because only one address area is applicable for the receive process.

Both buffers have a size of 13 bytes to store the CAN control bits, the identifier (standard or extended), and the data content (for details, see 11.12 Programmer's Model of Message Storage).

The receiver full flag (RXF) in the MSCAN08 receiver flag register (CRFLG) (see 11.13.5 MSCAN08 Receiver Flag Register (CRFLG)), signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier, this flag is set.

On reception, each message is checked to see if it passes the filter (for details see 11.5 Identifier Acceptance Filter) and in parallel is written into RxBG. The MSCAN08 copies the content of RxBG into RxFG[1], sets the RXF flag, and generates a receive interrupt to the CPU[2]. The user's receive handler has to read the received message from RxFG and to reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message which can follow immediately after the IFS field of the CAN frame, is received into RxBG. The overwriting of the background buffer is independent of the identifier filter function.

When the MSCAN08 module is transmitting, the MSCAN08 receives its own messages into the background receive buffer, RxBG. It does NOT overwrite RxFG, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loop-back mode (see 11.13.2 MSCAN08 Module Control Register 1), where the MSCAN08 treats its own messages exactly like all other incoming messages. The MSCAN08 receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN08 must be prepared to become a receiver.

1. Only if the RXF flag is not set.
2. The receive interrupt will occur only if not masked. Also, a polling scheme can be applied on RXF.

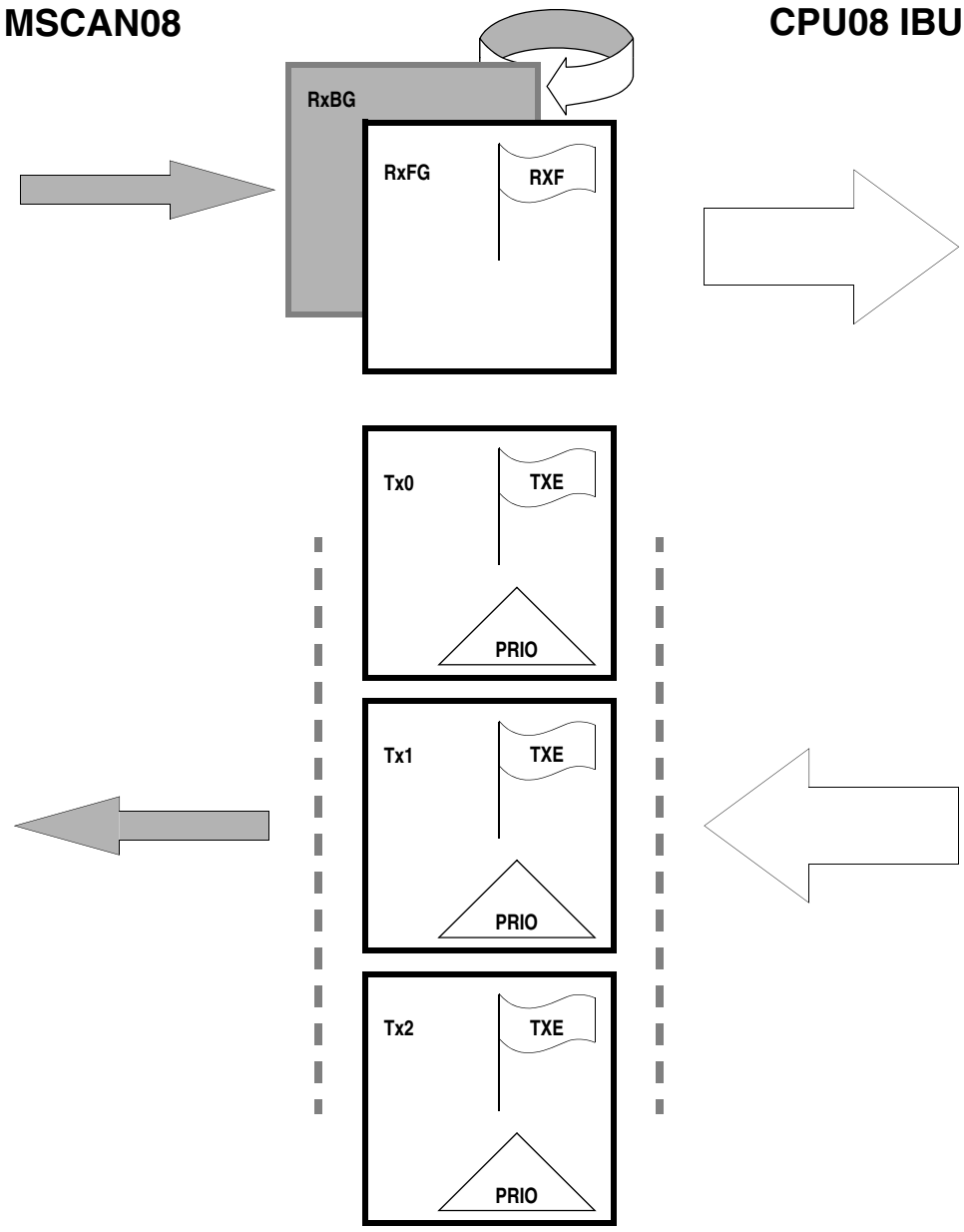


Figure 11-3. User Model for Message Buffer Organization

An overrun condition occurs when both the foreground and the background receive message buffers are filled with correctly received messages with accepted identifiers and another message is correctly received from the bus with an accepted identifier. The latter message will be discarded and an error interrupt with overrun indication will be generated if enabled. The MSCAN08 is still able to transmit messages with both receive message buffers filled, but all incoming messages are discarded.

11.4.3 Transmit Structures

The MSCAN08 has a triple transmit buffer scheme to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in Figure 11-3.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see 11.12 Programmer's Model of Message Storage). An additional transmit buffer priority register (TBPR) contains an 8-bit local priority field (PRIO) (see 11.12.5 Transmit Buffer Priority Registers).

To transmit a message, the CPU08 has to identify an available transmit buffer which is indicated by a set transmit buffer empty (TXE) flag in the MSCAN08 transmitter flag register (CTFLG) (see 11.13.7 MSCAN08 Transmitter Flag Register).

The CPU08 then stores the identifier, the control bits, and the data content into one of the transmit buffers. Finally, the buffer has to be flagged ready for transmission by clearing the TXE flag.

The MSCAN08 then will schedule the message for transmission and will signal the successful transmission of the buffer by setting the TXE flag. A transmit interrupt is generated[1] when TXE is set and can be used to drive the application software to re-load the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN08 uses the local priority setting of the three buffers for prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software sets this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being emitted from this node. The lowest binary value of the PRIO field is defined as the highest priority.

The internal scheduling process takes place whenever the MSCAN08 arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message being set up in one of the three transmit buffers. As messages that are already under transmission cannot be aborted, the user has to request the abort by setting the corresponding abort request flag (ABTRQ) in the transmission control register (CTCR). The MSCAN08 will then grant the request, if possible, by setting the corresponding abort request acknowledge (ABTAK) and the TXE flag in order to release the buffer and by generating a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was actually aborted (ABTAK = 1) or sent (ABTAK = 0).

11.5 Identifier Acceptance Filter

The identifier acceptance registers (CIDAR0–CIDAR3) define the acceptance patterns of the standard or extended identifier (ID10–ID0 or ID28–ID0). Any of these bits can be marked don't care in the identifier mask registers (CIDMR0–CIDMR3).

A filter hit is indicated to the application software by a set RXF (receive buffer full flag, see 11.13.5 MSCAN08 Receiver Flag Register (CRFLG)) and two bits in the Identifier acceptance control register (see 11.13.9 MSCAN08 Identifier Acceptance Control Register). These Identifier hit flags (IDHIT1–IDHIT0) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. In case that more than one hit occurs (two or more filters match) the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

1. Single identifier acceptance filter, each to be applied to a) the full 29 bits of the extended identifier and to the following bits of the CAN frame: RTR, IDE, SRR or b) the 11 bits of the standard identifier

1. The transmit interrupt will occur only if not masked. Also, a polling scheme can be applied on TXE.

plus the RTR and IDE bits of CAN 2.0A/B messages. This mode implements a single filter for a full length CAN 2.0B compliant extended identifier. Figure 11-4 shows how the 32-bit filter bank (CIDAR0–CIDAR3, CIDMR0–CIDMR3) produces a filter 0 hit.

2. Two identifier acceptance filters, each to be applied to a) the 14 most significant bits of the extended identifier plus the SRR and the IDE bits of CAN2.0B messages, or b) the 11 bits of the identifier plus the RTR and IDE bits of CAN 2.0A/B messages. Figure 11-5 shows how the 32-bit filter bank (CIDAR0–CIDAR3, CIDMR0–CIDMR3) produces filter 0 and 1 hits.
3. Four identifier acceptance filters, each to be applied to the first eight bits of the identifier. This mode implements four independent filters for the first eight bits of a CAN 2.0A/B compliant standard identifier. Figure 11-6 shows how the 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces filter 0 to 3 hits.
4. Closed filter. No CAN message will be copied into the foreground buffer RxFG, and the RXF flag will never be set.

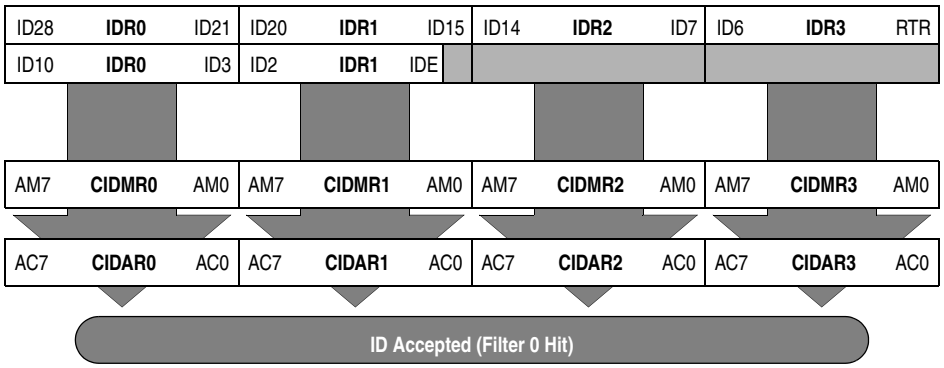


Figure 11-4. Single 32-Bit Maskable Identifier Acceptance Filter

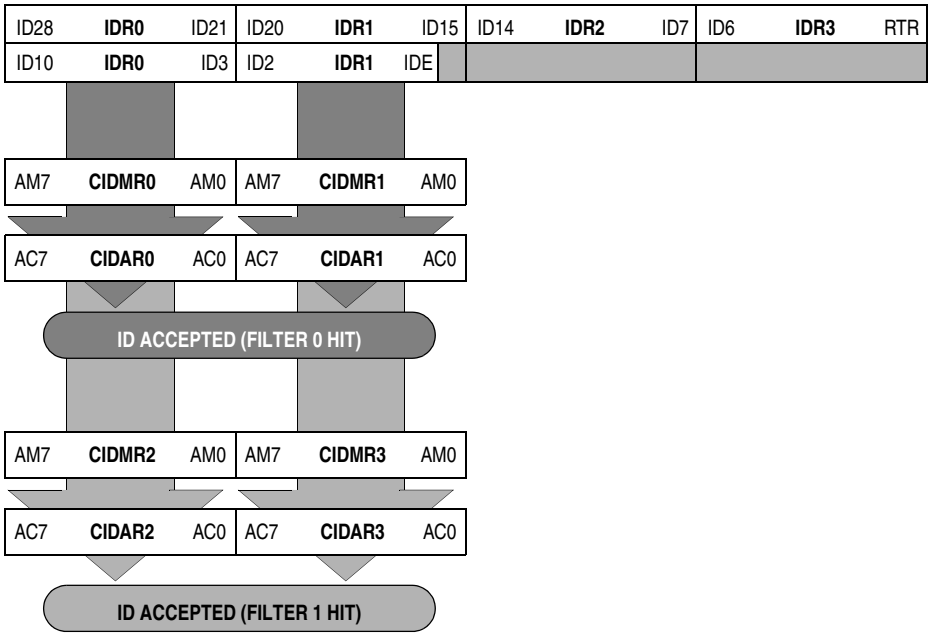


Figure 11-5. Dual 16-Bit Maskable Acceptance Filters

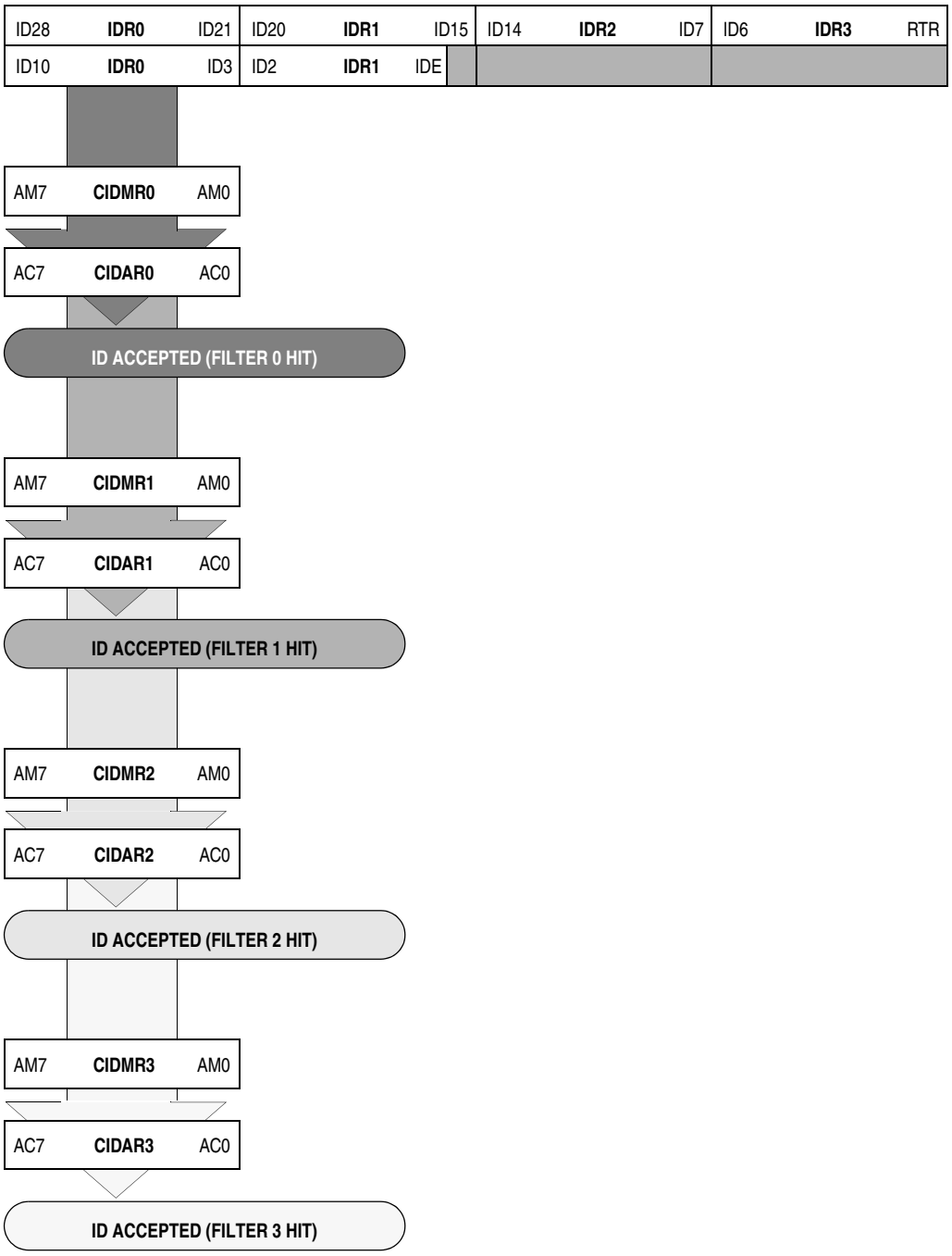


Figure 11-6. Quadruple 8-Bit Maskable Acceptance Filters

11.5.1 MSCAN Extended ID Rejected if Stuff Bit Between ID16 and ID15

For 32-bit and 16-bit identifier acceptance modes, an extended ID CAN frame with a stuff bit between ID16 and ID15 can be erroneously rejected, depending on IDAR0, IDAR1, and IDMR1.

Extended IDs (ID28–ID0) which generate a stuff bit between ID16 and ID15:

IDAR0	IDAR1	IDAR2	IDAR3
********	***1111x	xxxxxxxx	xxxxxxxx

where: x = 0 or 1 (don't care)
* = pattern for ID28 to ID18 (see the following).

Affected extended IDs (ID28 - ID18) patterns:

a) xxxxxxxxx01 exceptions: 00000000001 01111100001
xxxx1000001 exception: 11111000001

b) xxxxx100000 exception: 01111100000

c) xxxx0111111 exception: 00000111111

d) x0111110000

e) 10000000000

f) 11111111111

g) 10000011111

When an affected ID is received, an incorrect value is compared to the 2nd byte of the filter (IDAR1 and IDAR5, plus IDAR3 and IDAR7 in 16-bit mode). This incorrect value is the shift register contents before ID15 is shifted in (i.e., right shifted by 1).

Workaround

If the problematic IDs cannot be avoided, the workaround is to mask certain bits with IDMR1 (and IDMR5, plus IDMR3 and IDMR7 in 16-bit mode).

Example 1: to receive the message IDs

xxxx xxxx x011 111x xxxx xxxx xxxx xxxx

IDMR1 etc. must be 111x xxx1, i.e. ID20, 19, 18, 15 must be masked.

Example 2: to receive the message IDs

xxxx 0111 1111 111x xxxx xxxx xxxx xxxx

IDMR1 etc. must be 1xxx xxx1, i.e. ID20 and ID15 must be masked.

In general, using IDMR1 etc. 1111 xxx1, i.e. masking ID20, ID19, ID18, SRR, ID15, hides the problem.

11.6 Interrupts

The MSCAN08 supports four interrupt vectors mapped onto eleven different interrupt sources, any of which can be individually masked (for details see 11.13.5 MSCAN08 Receiver Flag Register (CRFLG) to 11.13.8 MSCAN08 Transmitter Control Register).

- *Transmit Interrupt*: At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXE flags of the empty message buffers are set.
- *Receive Interrupt*: A message has been received successfully and loaded into the foreground receive buffer. This interrupt will be emitted immediately after receiving the EOF symbol. The RXF flag is set.

- *Wakeup Interrupt*: An activity on the CAN bus occurred during MSCAN08 internal sleep mode or power-down mode (provided SLPAK = WUPIE = 1).
- *Error Interrupt*: An overrun, error, or warning condition occurred. The receiver flag register (CRFLG) will indicate one of the following conditions:
  - *Overrun:* An overrun condition as described in 11.4.2 Receive Structures has occurred.
  - *Receiver Warning*: The receive error counter has reached the CPU warning limit of 96.
  - *Transmitter Warning*: The transmit error counter has reached the CPU warning limit of 96.
  - *Receiver Error Passive*: The receive error counter has exceeded the error passive limit of 127 and MSCAN08 has gone to error passive state.
  - *Transmitter Error Passive*: The transmit error counter has exceeded the error passive limit of 127 and MSCAN08 has gone to error passive state.
  - *Bus Off*: The transmit error counter has exceeded 255 and MSCAN08 has gone to bus off state.

11.6.1 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN08 receiver flag register (CRFLG) or the MSCAN08 transmitter flag register (CTFLG). Interrupts are pending as long as one of the corresponding flags is set. The flags in the above registers must be reset within the interrupt handler in order to handshake the interrupt. The flags are reset through writing a 1 to the corresponding bit position. A flag cannot be cleared if the respective condition still prevails.

NOTE

Bit manipulation instructions (`BSET`) shall not be used to clear interrupt flags.

11.6.2 Interrupt Vectors

The MSCAN08 supports four interrupt vectors as shown in Table 11-1. The vector addresses and the relative interrupt priority are defined in **Table 2-1. Vector Addresses**

Table 11-1. MSCAN08 Interrupt Vector Addresses

Function	Source	Local Mask	Global Mask
Wakeup	WUPIF	WUPIE	I Bit
Error Interrupts	RWRNIF	RWRNIE	
	TWRNIF	TWRNIE	
	RERRIF	RERRIE	
	TERRIF	TERRIE	
	BOFFIF	BOFFIE	
	OVRIF	OVRIE	
Receive	RXF	RXFIE	
Transmit	TXE0	TXEIE0	
	TXE1	TXEIE1	
	TXE2	TXEIE2	

11.7 Protocol Violation Protection

The MSCAN08 will protect the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN08 can not be modified while the MSCAN08 is on-line. The SFTRES bit in the MSCAN08 module control register (see 11.13.1 MSCAN08 Module Control Register 0) serves as a lock to protect the following registers:
  - MSCAN08 module control register 1 (CMCR1)
  - MSCAN08 bus timing register 0 and 1 (CBTR0 and CBTR1)
  - MSCAN08 identifier acceptance control register (CIDAC)
  - MSCAN08 identifier acceptance registers (CIDAR0–CIDAR3)
  - MSCAN08 identifier mask registers (CIDMR0–CIDMR3)
- The TxCAN pin is forced to recessive when the MSCAN08 is in any of the low-power modes.

11.8 Low Power Modes

In addition to normal mode, the MSCAN08 has three modes with reduced power consumption:

- Sleep mode
- Soft reset mode
- Power down modes

In sleep and soft reset mode, power consumption is reduced by stopping all clocks except those to access the registers. In power down mode, all clocks are stopped and no power is consumed.

The WAIT and STOP instructions put the MCU in low-power consumption standby modes. Table 11-2 summarizes the combinations of MSCAN08 and CPU modes. A particular combination of modes is entered for the given settings of the bits SLPAK and SFTRES. For all modes, an MSCAN wakeup interrupt can occur only if SLPAK = WUPIE = 1.

Table 11-2. MSCAN08 versus CPU Operating Modes

MSCAN Mode	CPU Mode	
	STOP	WAIT or RUN
Power Down	SLPAK = X[1] SFTRES = X	
Sleep		SLPAK = 1 SFTRES = 0
Soft Reset		SLPAK = 0 SFTRES = 1
Normal		SLPAK = 0 SFTRES = 0

1. ‘X’ means don’t care.

11.8.1 MSCAN08 Sleep Mode

The CPU can request the MSCAN08 to enter the low-power mode by asserting the SLPRQ bit in the module configuration register (see Figure 11-7). The time when the MSCAN08 enters sleep mode depends on its activity:

- If it is transmitting, it continues to transmit until there is no more messages to be transmitted, and then goes into sleep mode
- If it is receiving, it waits for the end of this message and then goes into sleep mode
- If it is neither transmitting or receiving, it will immediately go into sleep mode

NOTE

The application software must avoid setting up a transmission (by clearing one or more TXE flags) and immediately request sleep mode (by setting SLPRQ). It then depends on the exact sequence of operations as to whether MSCAN08 starts transmitting or goes into sleep mode directly.

During sleep mode, the SLPAK flag is set. The application software should use SLPAK as a handshake indication for the request (SLPRQ) to go into sleep mode. When in sleep mode, the MSCAN08 stops its internal clocks. However, clocks to allow register accesses still run. If the MSCAN08 is in bus-off state, it stops counting the 128*11 consecutive recessive bits due to the stopped clocks. The TxCAN pin stays in recessive state. If RXF = 1, the message can be read and RXF can be cleared. Copying RxGB into RxFG doesn't take place while in sleep mode. It is possible to access the transmit buffers and to clear the TXE flags. No message abort takes place while in sleep mode.

The MSCAN08 leaves sleep mode (wakeup) when:

- Bus activity occurs, or
- The MCU clears the SLPRQ bit, or
- The MCU sets the SFTRES bit

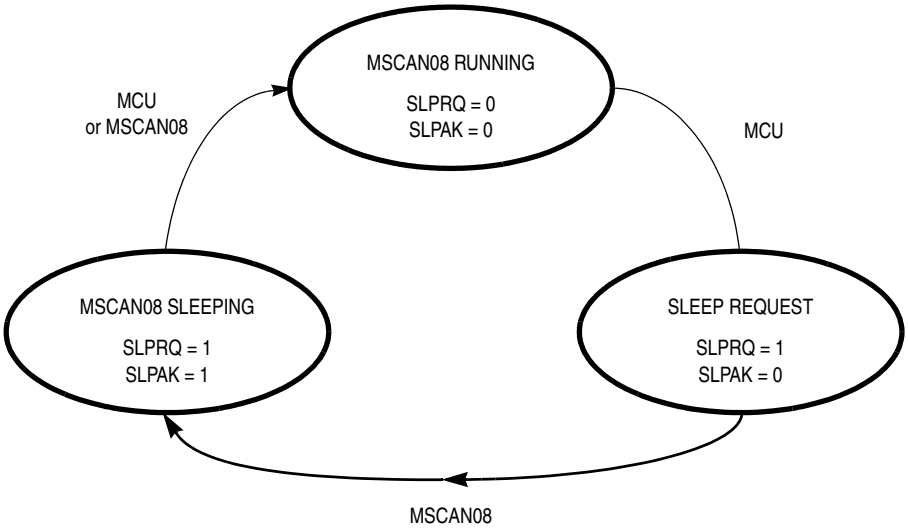


Figure 11-7. Sleep Request/Acknowledge Cycle

NOTE

The MCU cannot clear the SLPRQ bit before the MSCAN08 is in sleep mode (SLPAK = 1).

After wakeup, the MSCAN08 waits for 11 consecutive recessive bits to synchronize to the bus. As a consequence, if the MSCAN08 is woken by a CAN frame, this frame is not received. The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions are executed upon wakeup: copying of RxBG into RxFG, message aborts, and message transmissions. If the MSCAN08 is still in bus-off state after sleep mode was left, it continues counting the 128*11 consecutive recessive bits.

11.8.2 MSCAN08 Soft Reset Mode

In soft reset mode, the MSCAN08 is stopped although registers can still be accessed. This mode is used to initialize the module configuration, bit timing, and the CAN message filter. See 11.13.1 MSCAN08 Module Control Register 0 for a complete description of the soft reset mode.

When setting the SFTRES bit, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations.

NOTE

The user is responsible to take care that the MSCAN08 is not active when soft reset mode is entered. The recommended procedure is to bring the MSCAN08 into sleep mode before the SFTRES bit is set.

11.8.3 MSCAN08 Power Down Mode

The MSCAN08 is in power down mode when the CPU is in stop mode. When entering the power down mode, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations.

NOTE

The user is responsible to take care that the MSCAN08 is not active when power down mode is entered. The recommended procedure is to bring the MSCAN08 into sleep mode before the STOP instruction is executed.

To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN08 drives the TxCAN pin into recessive state.

In power down mode, no registers can be accessed.

MSCAN08 bus activity can wake the MCU from CPU stop/MSCAN08 power-down mode. However, until the oscillator starts up and synchronization is achieved the MSCAN08 will not respond to incoming data.

11.8.4 CPU Wait Mode

The MSCAN08 module remains active during CPU wait mode. The MSCAN08 will stay synchronized to the CAN bus and generates transmit, receive, and error interrupts to the CPU, if enabled. Any such interrupt will bring the MCU out of wait mode.

11.8.5 Programmable Wakeup Function

The MSCAN08 can be programmed to apply a low-pass filter function to the RxCAN input line while in internal sleep mode (see information on control bit WUPM in 11.13.2 MSCAN08 Module Control Register 1). This feature can be used to protect the MSCAN08 from wakeup due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic inference within noisy environments.

11.9 Timer Link

The MSCAN08 will generate a timer signal whenever a valid frame has been received. Because the CAN specification defines a frame to be valid if no errors occurred before the EOF field has been transmitted successfully, the timer signal will be generated right after the EOF. A pulse of one bit time is generated. As the MSCAN08 receiver engine also receives the frames being sent by itself, a timer signal also will be generated after a successful transmission.

The previously described timer signal can be routed into the on-chip timer interface module B (TIMB). This signal is connected to the channel 0 input under the control of the timer link enable (TLNKEN) bit in CMCR0.

After the TIMB module has been programmed to capture rising edge events, it can be used under software control to generate 16-bit time stamps which can be stored with the received message.

11.10 Clock System

Figure 11-8 shows the structure of the MSCAN08 clock generation circuitry and its interaction with the clock generation module (CGM). With this flexible clocking scheme the MSCAN08 is able to handle CAN bus rates ranging from 10 kbps up to 1 Mbps.

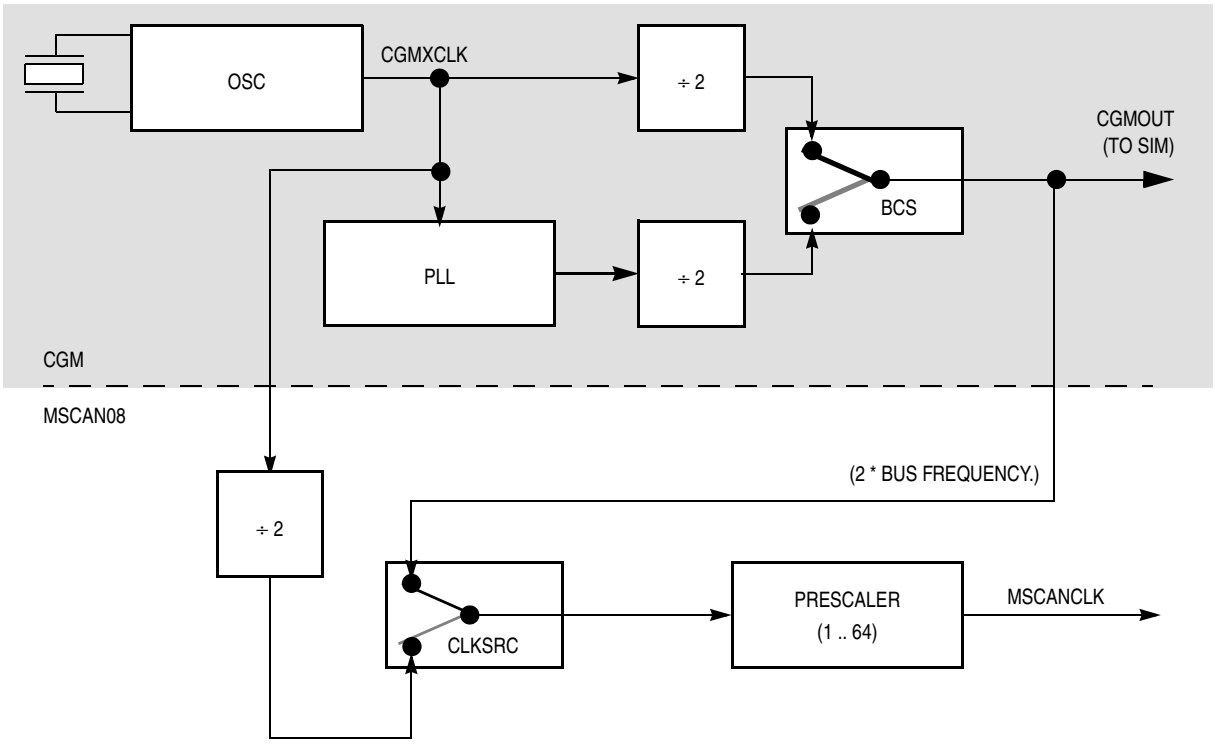


Figure 11-8. Clocking Scheme

The clock source bit (CLKSRC) in the MSCAN08 module control register (CMCR1) (see 11.13.1 MSCAN08 Module Control Register 0) defines whether the MSCAN08 is connected to the output of the crystal oscillator or to the PLL output.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met.

NOTE

If the system clock is generated from a PLL, it is recommended to select the crystal clock source rather than the system clock source due to jitter considerations, especially at faster CAN bus rates.

A programmable prescaler is used to generate out of the MSCAN08 clock the time quanta (Tq) clock. A time quantum is the atomic unit of time handled by the MSCAN08.

$$f_{Tq} = \frac{f_{MSCANCLK}}{\text{Presc value}}$$

A bit time is subdivided into three segments[1] (see Figure 11-9):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time segment 2: This segment represents PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

$$\text{Bit rate} = \frac{f_{Tq}}{\text{No. of time quanta}}$$

The synchronization jump width (SJW) can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

The above parameters can be set by programming the bus timing registers, CBTR0 and CBTR1. See 11.13.3 MSCAN08 Bus Timing Register 0 and 11.13.4 MSCAN08 Bus Timing Register 1.

NOTE

It is the user's responsibility to make sure that the bit timing settings are in compliance with the CAN standard,

Table 11-8 gives an overview on the CAN conforming segment settings and the related parameter values.

1. For further explanation of the underlying concepts please refer to ISO/DIS 11 519-1, Section 10.3.

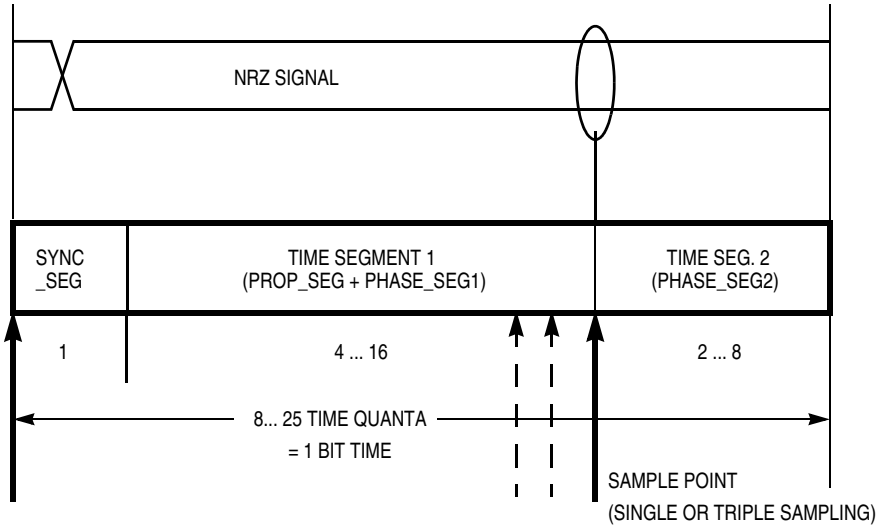


Figure 11-9. Segments Within the Bit Time

Table 11-3. Time Segment Syntax

SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit point	A node in transmit mode will transfer a new value to the CAN bus at this point.
Sample point	A node in receive mode will sample the bus at this point. If the three samples per bit option is selected then this point marks the position of the third sample.

Table 11-4. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronized Jump Width	SJW
5 .. 10	4 .. 9	2	1	1 .. 2	0 .. 1
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

11.11 Memory Map

The MSCAN08 occupies 128 bytes in the CPU08 memory space. The absolute mapping is implementation dependent with the base address being a multiple of 128.

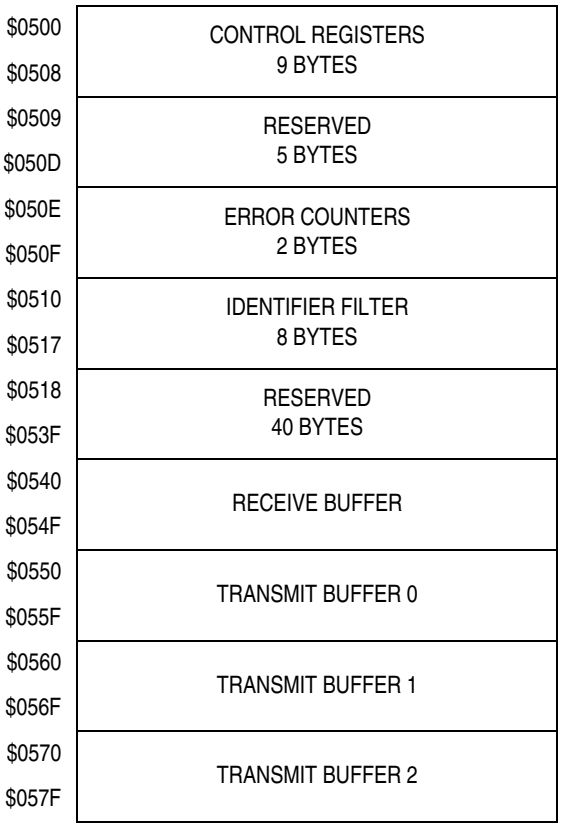


Figure 11-10. MSCAN08 Memory Map

11.12 Programmer's Model of Message Storage

This section details the organization of the receive and transmit message buffers and the associated control registers. For reasons of programmer interface simplification, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13-byte data structure. An additional transmit buffer priority register (TBPR) is defined for the transmit buffers.

Addr[1]	Register Name
$05b0	IDENTIFIER REGISTER 0
$05b1	IDENTIFIER REGISTER 1
$05b2	IDENTIFIER REGISTER 2
$05b3	IDENTIFIER REGISTER 3
$05b4	DATA SEGMENT REGISTER 0
$05b5	DATA SEGMENT REGISTER 1
$05b6	DATA SEGMENT REGISTER 2
$05b7	DATA SEGMENT REGISTER 3
$05b8	DATA SEGMENT REGISTER 4
$05b9	DATA SEGMENT REGISTER 5
$05bA	DATA SEGMENT REGISTER 6
$05bB	DATA SEGMENT REGISTER 7
$05bC	DATA LENGTH REGISTER
$05bD	TRANSMIT BUFFER PRIORITY REGISTER[2]
$05bE	UNUSED
$05bF	UNUSED

1. Where b equals the following:
   b = 4 for receive buffer
   b = 5 for transmit buffer 0
   b = 6 for transmit buffer 1
   b = 7 for transmit buffer 2
2. Not applicable for receive buffers

Figure 11-11. Message Buffer Organization

11.12.1 Message Buffer Outline

Figure 11-12 shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure 11-13. All bits of the 13-byte data structure are undefined out of reset.

NOTE

The foreground receive buffer can be read anytime but cannot be written.
The transmit buffers can be read or written anytime.

Addr.	Register		Bit 7	6	5	4	3	2	1	Bit 0
$05b0	IDR0	Read: Write:	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
$05b1	IDR1	Read: Write:	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
$05b2	IDR2	Read: Write:	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
$05b3	IDR3	Read: Write:	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
$05b4	DSR0	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05b5	DSR1	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05b6	DSR2	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05b7	DSR3	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05b8	DSR4	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05b9	DSR5	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05bA	DSR6	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05bB	DSR7	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$05bC	DLR	Read: Write:					DLC3	DLC2	DLC1	DLC0

= Unimplemented

Figure 11-12. Receive/Transmit Message Buffer Extended Identifier (IDRn)

Addr.	Register		Bit 7	6	5	4	3	2	1	Bit 0
$05b0	IDR0	Read: Write:	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
$05b1	IDR1	Read: Write:	ID2	ID1	ID0	RTR	IDE (=0)			
$05b2	IDR2	Read: Write:								
$05b3	IDR3	Read: Write:								

= Unimplemented

Figure 11-13. Standard Identifier Mapping

11.12.2 Identifier Registers

The identifiers consist of either 11 bits (ID10–ID0) for the standard, or 29 bits (ID28–ID0) for the extended format. ID10/28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

SRR — Substitute Remote Request

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and will be stored as received on the CAN bus for receive buffers.

IDE — ID Extended

This flag indicates whether the extended or standard identifier format is applied in this buffer. In case of a receive buffer, the flag is set as being received and indicates to the CPU how to process the buffer identifier registers. In case of a transmit buffer, the flag indicates to the MSCAN08 what type of identifier to send.

1 = Extended format, 29 bits
0 = Standard format, 11 bits

RTR — Remote Transmission Request

This flag reflects the status of the remote transmission request bit in the CAN frame. In case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.

1 = Remote frame
0 = Data frame

11.12.3 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

DLC3–DLC0 — Data Length Code Bits

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table 11-5 shows the effect of setting the DLC bits.

Table 11-5. Data Length Codes

Data Length Code				Data Byte Count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

11.12.4 Data Segment Registers (DSRn)

The eight data segment registers contain the data to be transmitted or received. The number of bytes to be transmitted or being received is determined by the data length code in the corresponding DLR.

11.12.5 Transmit Buffer Priority Registers

Address: $05bD

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
Reset:	Unaffected by reset							

Figure 11-14. Transmit Buffer Priority Register (TBPR)

PRIO7–PRIO0 — Local Priority

This field defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the MSCAN08 and is defined to be highest for the smallest binary number. The MSCAN08 implements the following internal prioritization mechanism:

- All transmission buffers with a cleared TXE flag participate in the prioritization right before the SOF is sent.
- The transmission buffer with the lowest local priority field wins the prioritization.
- In case more than one buffer has the same lowest priority, the message buffer with the lower index number wins.

11.13 Programmer's Model of Control Registers

The programmer's model has been laid out for maximum simplicity and efficiency. Figure 11-15 gives an overview on the control register block of the MSCAN08.

Addr.	Register		Bit 7	6	5	4	3	2	1	Bit 0
$0500	CMCR0	Read:	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
		Write:								
$0501	CMCR1	Read:	0	0	0	0	0	LOOPB	WUPM	CLKSRC
		Write:								
$0502	CBTR0	Read:	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
		Write:								
$0503	CBTR1	Read:	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		Write:								
$0504	CRFLG	Read:	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
		Write:								
$0505	CRIER	Read:	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
		Write:								
$0506	CTFLG	Read:	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
		Write:								
$0507	CTCR	Read:	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
		Write:								
$0508	CIDAC	Read:	0	0	IDAM1	IDAM0	0	0	IDHIT1	IDHIT0
		Write:								
$0509	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
$050E	CRXERR	Read:	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
		Write:								
$050F	CTXERR	Read:	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
		Write:								
$0510	CIDAR0	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
$0511	CIDAR1	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
$0512	CIDAR2	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
$0513	CIDAR3	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
$0514	CIDMR0	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								

= Unimplemented R = Reserved

Figure 11-15. MSCAN08 Control Register Structure

Addr.	Register		Bit 7	6	5	4	3	2	1	Bit 0
$0515	CIDMR1	Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
$0516	CIDMR2	Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
$0517	CIDMR3	Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0

= Unimplemented R = Reserved

Figure 11-15. MSCAN08 Control Register Structure (Continued)

11.13.1 MSCAN08 Module Control Register 0

Address: $0500

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
Write:								
Reset:	0	0	0	0	0	0	0	1

= Unimplemented

Figure 11-16. Module Control Register 0 (CMCR0)

SYNCH — Synchronized Status

This bit indicates whether the MSCAN08 is synchronized to the CAN bus and as such can participate in the communication process.

1 = MSCAN08 synchronized to the CAN bus
0 = MSCAN08 not synchronized to the CAN bus

TLNKEN — Timer Enable

This flag is used to establish a link between the MSCAN08 and the on-chip timer (see 11.9 Timer Link).

1 = The MSCAN08 timer signal output is connected to the timer input.
0 = The port is connected to the timer input.

SLPAK — Sleep Mode Acknowledge

This flag indicates whether the MSCAN08 is in module internal sleep mode. It shall be used as a handshake for the sleep mode request (see 11.8.1 MSCAN08 Sleep Mode). If the MSCAN08 detects bus activity while in sleep mode, it clears the flag.

1 = Sleep – MSCAN08 in internal sleep mode
0 = Wakeup – MSCAN08 is not in sleep mode

SLPRQ — Sleep Request, Go to Internal Sleep Mode

This flag requests the MSCAN08 to go into an internal power-saving mode (see 11.8.1 MSCAN08 Sleep Mode).

1 = Sleep — The MSCAN08 will go into internal sleep mode.
0 = Wakeup — The MSCAN08 will function normally.

SFTRES — Soft Reset

When this bit is set by the CPU, the MSCAN08 immediately enters the soft reset state. Any ongoing transmission or reception is aborted and synchronization to the bus is lost.

The following registers enter and stay in their hard reset state:

CMCR0, CRFLG, CRIER, CTFLG, and CTCR.

The registers CMCR1, CBTR0, CBTR1, CIDAC, CIDAR0–CIDAR3, and CIDMR0–CIDMR3 can only be written by the CPU when the MSCAN08 is in soft reset state. The values of the error counters are not affected by soft reset.

When this bit is cleared by the CPU, the MSCAN08 tries to synchronize to the CAN bus. If the MSCAN08 is not in bus-off state, it will be synchronized after 11 recessive bits on the bus; if the MSCAN08 is in bus-off state, it continues to wait for 128 occurrences of 11 recessive bits.

Clearing SFTRES and writing to other bits in CMCR0 must be in separate instructions.

1 = MSCAN08 in soft reset state

0 = Normal operation

11.13.2 MSCAN08 Module Control Register 1

Address: $0501

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

= Unimplemented

Figure 11-17. Module Control Register (CMCR1)

LOOPB — Loop Back Self-Test Mode

When this bit is set, the MSCAN08 performs an internal loop back which can be used for self-test operation: the bit stream output of the transmitter is fed back to the receiver internally. The CAN_{RX} input pin is ignored and the CAN_{TX} output goes to the recessive state (logic 1). The MSCAN08 behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state the MSCAN08 ignores the bit sent during the ACK slot of the CAN frame Acknowledge field to insure proper reception of its own message. Both transmit and receive interrupts are generated.

1 = Activate loop back self-test mode

0 = Normal operation

WUPM — Wakeup Mode

This flag defines whether the integrated low-pass filter is applied to protect the MSCAN08 from spurious wakeups (see 11.8.5 Programmable Wakeup Function).

1 = MSCAN08 will wakeup the CPU only in cases of a dominant pulse on the bus which has a length of at least t_{wup}.

0 = MSCAN08 will wakeup the CPU after any recessive-to-dominant edge on the CAN bus.

CLKSRC — Clock Source

This flag defines which clock source the MSCAN08 module is driven from (see 11.10 Clock System).

1 = The MSCAN08 clock source is CGMOUT (see Figure 11-8).

0 = The MSCAN08 clock source is CGMXCLK/2 (see Figure 11-8).

NOTE

The CMCR1 register can be written only if the SFTRES bit in the MSCAN08 module control register is set

11.13.3 MSCAN08 Bus Timing Register 0

Address: $0502

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
Reset:	0	0	0	0	0	0	0	0

Figure 11-18. Bus Timing Register 0 (CBTR0)

SJW1 and SJW0 — Synchronization Jump Width

The synchronization jump width (SJW) defines the maximum number of time quanta (T_q) clock cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see Table 11-6).

Table 11-6. Synchronization Jump Width

SJW1	SJW0	Synchronization Jump Width
0	0	1 T_q cycle
0	1	2 T_q cycle
1	0	3 T_q cycle
1	1	4 T_q cycle

BRP5–BRP0 — Baud Rate Prescaler

These bits determine the time quanta (T_q) clock, which is used to build up the individual bit timing, according to Table 11-7.

Table 11-7. Baud Rate Prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Prescaler Value (P)
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
:	:	:	:	:	:	:
:	:	:	:	:	:	:
1	1	1	1	1	1	64

NOTE

The CBTR0 register can be written only if the SFTRES bit in the MSCAN08 module control register is set.

11.13.4 MSCAN08 Bus Timing Register 1

Address: $0503

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
Reset:	0	0	0	0	0	0	0	0

Figure 11-19. Bus Timing Register 1 (CBTR1)

SAMP — Sampling

This bit determines the number of serial bus samples to be taken per bit time. If set, three samples per bit are taken, the regular one (sample point) and two preceding samples, using a majority rule. For higher bit rates, SAMP should be cleared, which means that only one sample will be taken per bit.

1 = Three samples per bit(1)
0 = One sample per bit

TSEG22–TSEG10 — Time Segment

Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point. Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in Table 11-8.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (T_q) clock cycles per bit as shown in Table 11-4).

$$\text{Bit time} = \frac{\text{Pres value}}{f_{MSCANCLK}} \bullet \text{number of time quanta}$$

NOTE

The CBTR1 register can only be written if the SFTRES bit in the MSCAN08 module control register is set.

Table 11-8. Time Segment Values

TSEG13	TSEG12	TSEG11	TSEG10	Time Segment 1
0	0	0	0	1 T_q Cycle(1)
0	0	0	1	2 T_q Cycles(1)
0	0	1	0	3T_q Cycles(1)
0	0	1	1	4 T_q Cycles
.	.	.	.	.
.	.	.	.	.
1	1	1	1	16 T_q Cycles

TSEG22	TSEG21	TSEG20	Time Segment 2
0	0	0	1 T_q Cycle(1)
0	0	1	2 T_q Cycles
.	.	.	.
.	.	.	.
1	1	1	8T_q Cycles

1. This setting is not valid. Please refer to Table 11-4 for valid settings.

1. In this case PHASE_SEG1 must be at least 2 time quanta.

11.13.5 MSCAN08 Receiver Flag Register (CRFLG)

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. A flag can be cleared only when the condition which caused the setting is valid no more. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CRIER register. A hard or soft reset will clear the register.

Address: $0504

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
Reset:	0	0	0	0	0	0	0	0

Figure 11-20. Receiver Flag Register (CRFLG)

WUPIF — Wakeup Interrupt Flag

If the MSCAN08 detects bus activity while in sleep mode, it sets the WUPIF flag. If not masked, a wakeup interrupt is pending while this flag is set.

1 = MSCAN08 has detected activity on the bus and requested wakeup.
0 = No wakeup interrupt has occurred.

RWRNIF — Receiver Warning Interrupt Flag

This flag is set when the MSCAN08 goes into warning status due to the receive error counter (REC) exceeding 96 and neither one of the error interrupt flags or the bus-off interrupt flag is set(1). If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 has gone into receiver warning status.
0 = No receiver warning status has been reached.

TWRNIF — Transmitter Warning Interrupt Flag

This flag is set when the MSCAN08 goes into warning status due to the transmit error counter (TEC) exceeding 96 and neither one of the error interrupt flags or the bus-off interrupt flag is set(2). If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 has gone into transmitter warning status.
0 = No transmitter warning status has been reached.

RERRIF — Receiver Error Passive Interrupt Flag

This flag is set when the MSCAN08 goes into error passive status due to the receive error counter exceeding 127 and the bus-off interrupt flag is not set(3). If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 has gone into receiver error passive status.
0 = No receiver error passive status has been reached.

1. Condition to set the flag: RWRNIF = (96 → REC) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
2. Condition to set the flag: TWRNIF = (96 → TEC) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
3. Condition to set the flag: RERRIF = (127 → REC → 255) & $\overline{\text{BOFFIF}}$

TERRIF — Transmitter Error Passive Interrupt Flag

This flag is set when the MSCAN08 goes into error passive status due to the transmit error counter exceeding 127 and the bus-off interrupt flag is not set[1]. If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 went into transmit error passive status.
0 = No transmit error passive status has been reached.

BOFFIF — Bus-Off Interrupt Flag

This flag is set when the MSCAN08 goes into bus-off status, due to the transmit error counter exceeding 255. It cannot be cleared before the MSCAN08 has monitored 128 times 11 consecutive ‘recessive’ bits on the bus. If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08has gone into bus-off status.
0 = No bus-off status has been reached.

OVRIF — Overrun Interrupt Flag

This flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set.

1 = A data overrun has been detected since last clearing the flag.
0 = No data overrun has occurred.

RXF — Receive Buffer Full

The RXF flag is set by the MSCAN08 when a new message is available in the foreground receive buffer. This flag indicates whether the buffer is loaded with a correctly received message. After the CPU has read that message from the receive buffer the RXF flag must be cleared to release the buffer. A set RXF flag prohibits the exchange of the background receive buffer into the foreground buffer. If not masked, a receive interrupt is pending while this flag is set.

1 = The receive buffer is full. A new message is available.
0 = The receive buffer is released (not full).

NOTE

To ensure data integrity, no registers of the receive buffer shall be read while the RXF flag is cleared.

The CRFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.

11.13.6 MSCAN08 Receiver Interrupt Enable Register

Address: $0505

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
Reset:	0	0	0	0	0	0	0	0

Figure 11-21. Receiver Interrupt Enable Register (CRIER)

WUPIE — Wakeup Interrupt Enable

1 = A wakeup event will result in a wakeup interrupt.
0 = No interrupt will be generated from this event.

1. Condition to set the flag: TERRIF = (128 → TEC → 255) & $\overline{\text{BOFFIF}}$

RWRNIE — Receiver Warning Interrupt Enable

1 = A receiver warning status event will result in an error interrupt.
0 = No interrupt is generated from this event.

TWRNIE — Transmitter Warning Interrupt Enable

1 = A transmitter warning status event will result in an error interrupt.
0 = No interrupt is generated from this event.

RERRIE — Receiver Error Passive Interrupt Enable

1 = A receiver error passive status event will result in an error interrupt.
0 = No interrupt is generated from this event.

TERRIE — Transmitter Error Passive Interrupt Enable

1 = A transmitter error passive status event will result in an error interrupt.
0 = No interrupt is generated from this event.

BOFFIE — Bus-Off Interrupt Enable

1 = A bus-off event will result in an error interrupt.
0 = No interrupt is generated from this event.

OVRIE — Overrun Interrupt Enable

1 = An overrun event will result in an error interrupt.
0 = No interrupt is generated from this event.

RXFIE — Receiver Full Interrupt Enable

1 = A receive buffer full (successful message reception) event will result in a receive interrupt.
0 = No interrupt will be generated from this event.

NOTE

The CRIER register is held in the reset state when the SFTRES bit in CMCR0 is set.

11.13.7 MSCAN08 Transmitter Flag Register

The abort acknowledge flags are read only. The transmitter buffer empty flags are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag setting. The transmitter buffer empty flags each have an associated interrupt enable bit in the CTCR register. A hard or soft reset will resets the register.

Address:	$0506	5						
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
Write:								
Reset:	0	0	0	0	0	1	1	1

= Unimplemented

Figure 11-22. Transmitter Flag Register (CTFLG)

ABTAK2–ABTAK0 — Abort Acknowledge

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent. The ABTAKx flag is cleared implicitly whenever the corresponding TXE flag is cleared.

1 = The message has been aborted.
0 = The message has not been aborted, thus has been sent out.

TXE2–TXE0 — Transmitter Empty

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The MSCAN08 sets the flag after the message has been sent successfully. The flag is also set by the MSCAN08 when the transmission request was successfully aborted due to a pending abort request (see 11.12.5 Transmit Buffer Priority Registers). If not masked, a receive interrupt is pending while this flag is set. Clearing a TXEx flag also clears the corresponding ABTAKx flag (ABTAK, see above). When a TXEx flag is set, the corresponding ABTRQx bit (ABTRQ) is cleared. See 11.13.8 MSCAN08 Transmitter Control Register

1 = The associated message buffer is empty (not scheduled).

0 = The associated message buffer is full (loaded with a message due for transmission).

NOTE

To ensure data integrity, no registers of the transmit buffers should be written to while the associated TXE flag is cleared.

The CTFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.

11.13.8 MSCAN08 Transmitter Control Register

Address: $0507

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-23. Transmitter Control Register (CTCR)

ABTRQ2–ABTRQ0 — Abort Request

The CPU sets an ABTRQx bit to request that an already scheduled message buffer (TXE = 0) be aborted. The MSCAN08 will grant the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted the associated TXE and the abort acknowledge flag (ABTAK) (see 11.13.7 MSCAN08 Transmitter Flag Register) will be set and an TXE interrupt is generated if enabled. The CPU cannot reset ABTRQx. ABTRQx is cleared implicitly whenever the associated TXE flag is set.

1 = Abort request pending

0 = No abort request

NOTE

The software must not clear one or more of the TXE flags in CTFLG and simultaneously set the respective ABTRQ bit(s).

TXEIE2–TXEIE0 — Transmitter Empty Interrupt Enable

1 = A transmitter empty (transmit buffer available for transmission) event results in a transmitter empty interrupt.

0 = No interrupt is generated from this event.

NOTE

The CTCR register is held in the reset state when the SFTRES bit in CMCR0 is set.

11.13.9 MSCAN08 Identifier Acceptance Control Register

Address: $0508

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	IDAM1	IDAM0	0	0	IDHIT1	IDHIT0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-24. Identifier Acceptance Control Register (CIDAC)

IDAM1–IDAM0— Identifier Acceptance Mode

The CPU sets these flags to define the identifier acceptance filter organization (see 11.5 Identifier Acceptance Filter). Table 11-9 summarizes the different settings. In "filter closed" mode no messages will be accepted so that the foreground buffer will never be reloaded.

Table 11-9. Identifier Acceptance Mode Settings

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	Single 32-bit acceptance filter
0	1	Two 16-bit acceptance filter
1	0	Four 8-bit acceptance filters
1	1	Filter closed

IDHIT1–IDHIT0— Identifier Acceptance Hit Indicator

The MSCAN08 sets these flags to indicate an identifier acceptance hit (see 11.5 Identifier Acceptance Filter). Table 11-9 summarizes the different settings.

Table 11-10. Identifier Acceptance Hit Indication

IDHIT1	IDHIT0	Identifier Acceptance Hit
0	0	Filter 0 hit
0	1	Filter 1 hit
1	0	Filter 2 hit
1	1	Filter 3 hit

The IDHIT indicators are always related to the message in the foreground buffer. When a message gets copied from the background to the foreground buffer, the indicators are updated as well.

NOTE

The CIDAC register can be written only if the SFTRES bit in the CMCR0 is set.

11.13.10 MSCAN08 Receive Error Counter

Address: $050E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-25. Receiver Error Counter (CRXERR)

This read-only register reflects the status of the MSCAN08 receive error counter.

11.13.11 MSCAN08 Transmit Error Counter

Address: $050F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-26. Transmit Error Counter (CTXERR)

This read-only register reflects the status of the MSCAN08 transmit error counter.

NOTE

Both error counters may only be read when in sleep or soft reset mode.

11.13.12 MSCAN08 Identifier Acceptance Registers

On reception each message is written into the background receive buffer. The CPU is only signalled to read the message, however, if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message will be overwritten by the next message (dropped).

The acceptance registers of the MSCAN08 are applied on the IDR0 to IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers only the first two (CIDMR0/CIDMR1 and CIDAR0/CIDAR1) are applied.

CIDAR0 Address: $0510

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset:	Unaffected by reset							

CIDAR1 Address: $050511

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset:	Unaffected by reset							

CIDAR2 Address: $0512

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset:	Unaffected by reset							

CIDAR3 Address: $0513

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset:	Unaffected by reset							

Figure 11-27. Identifier Acceptance Registers (CIDAR0–CIDAR3)

AC7–AC0 — Acceptance Code Bits

AC7–AC0 comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

NOTE

The CIDAR0–CIDAR3 registers can be written only if the SFTRES bit in CMCR0 is set

11.13.13 MSCAN08 Identifier Mask Registers (CIDMR0–CIDMR3)

The identifier mask registers specify which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. For standard identifiers it is required to program the last three bits (AM2–AM0) in the mask register CIDMR1 to ‘don’t care’.

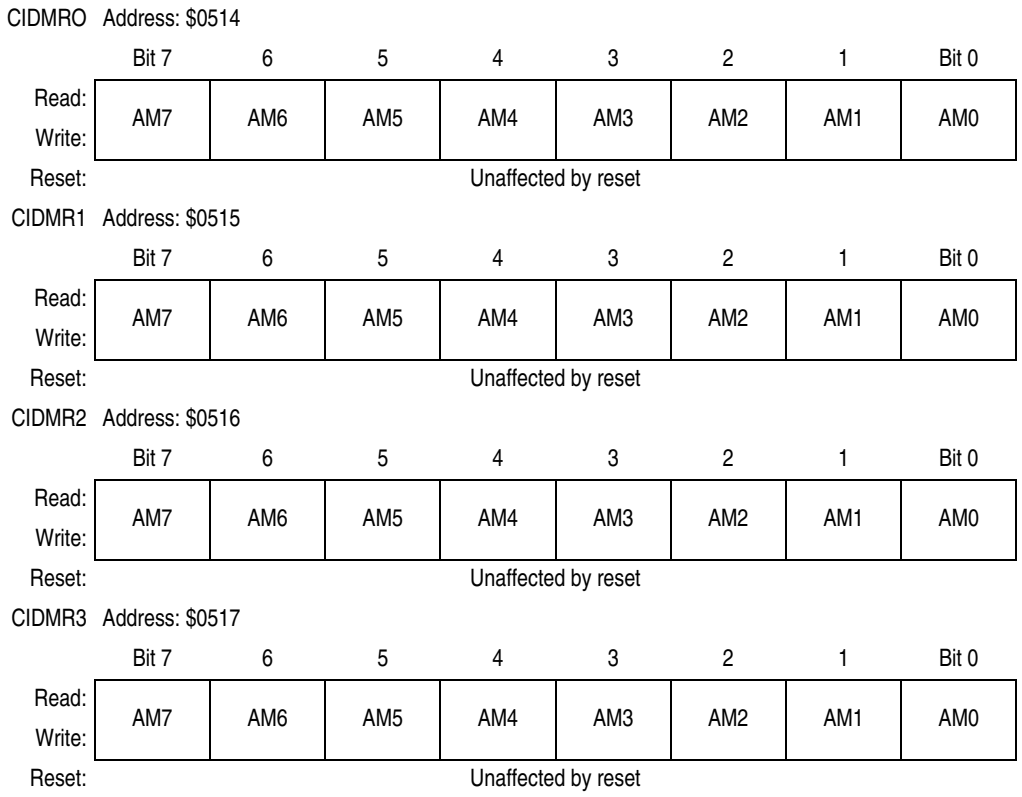


Figure 11-28. Identifier Mask Registers (CIDMR0–CIDMR3)

AM7–AM0 — Acceptance Mask Bits

If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match will be detected. The message will be accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted.

1 = Ignore corresponding acceptance code register bit.

0 = Match corresponding acceptance code register and identifier bits.

NOTE

The CIDMR0–CIDMR3 registers can be written only if the SFTRES bit in the CMCR0 is set

Chapter 12
Programmable Interrupt Timer (PIT)

12.1 Introduction

This section describes the programmable interrupt timer (PIT) which is a periodic interrupt timer whose counter is clocked internally via software programmable options. Figure 12-1 is a block diagram of the PIT.

For further information regarding timers on M68HC08 Family devices, please consult the *HC08 Timer Reference Manual* (Freescale document order number TIM08RM/AD).

12.2 Features

Features include:

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

12.3 Functional Description

Figure 12-1 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.

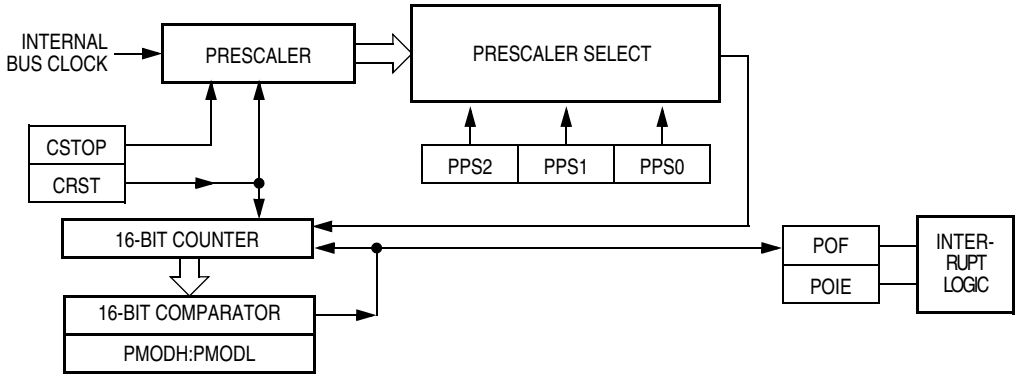


Figure 12-1. PIT Block Diagram

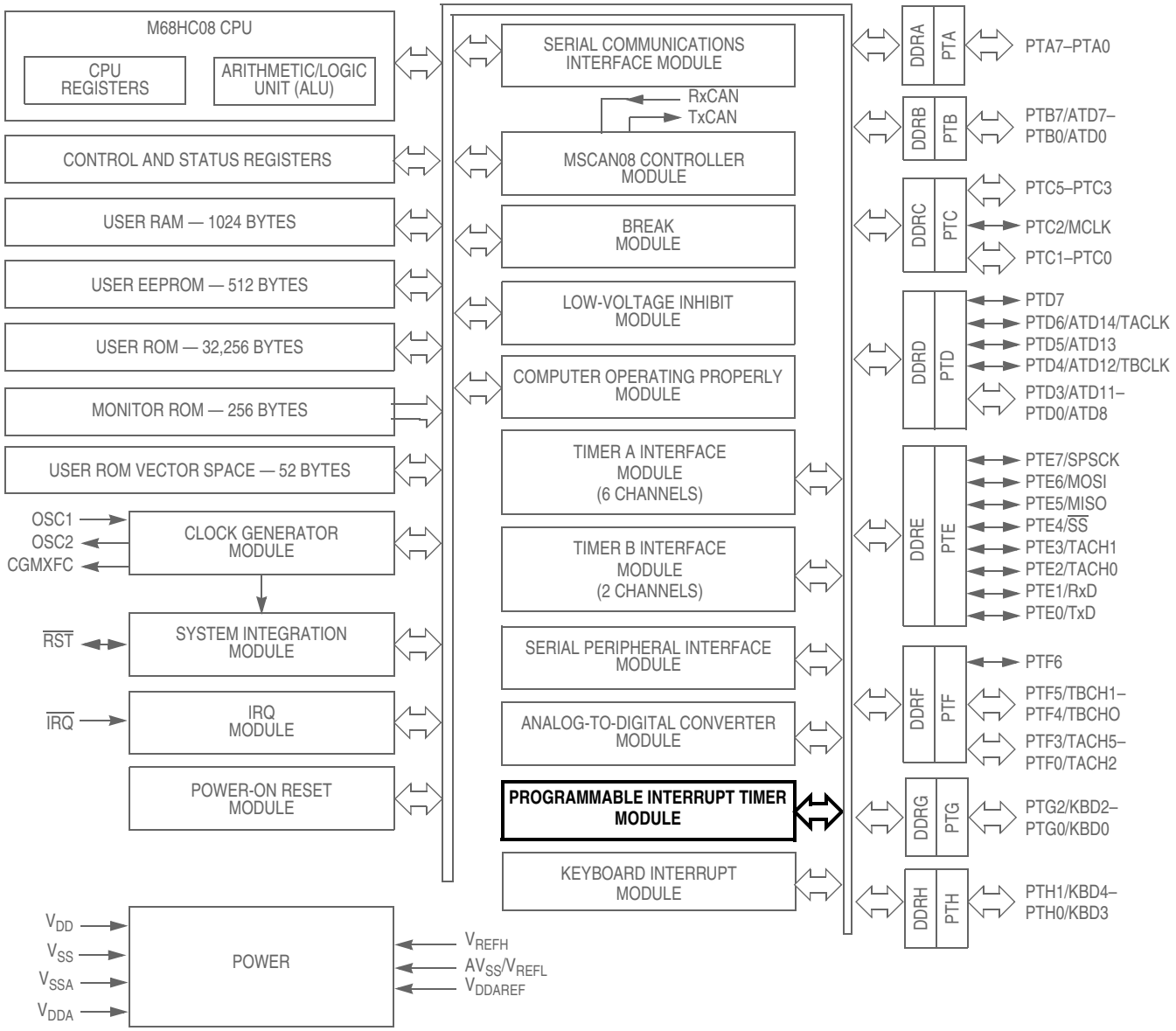


Figure 12-2. Block Diagram Highlighting PIT Block

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$004B	PIT Status and Control Register (PSC) See page 150.	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) See page 152.	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) See page 152.	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) See page 152.	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) See page 152.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	1	1	1	1	1	1	1	1

=Unimplemented

Figure 12-3. PIT I/O Register Summary

12.4 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 reaches the modulo value programmed 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.

12.5 Low-Power Modes

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

12.5.1 Wait Mode

The 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.

12.5.2 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.

12.6 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 Figure 15-17. SIM Break Status Register (SBSR)).

To allow software to clear status bits during a break interrupt, write a 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 0 to the BCFE bit. With BCFE at 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 0. After the break, doing the second step clears the status bit.

12.7 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)

12.7.1 PIT Status and Control Register

The PIT status and control register:

- 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 12-4. PIT Status and Control Register (PSC)

POF — PIT Overflow Flag Bit

This read/write flag is set when the PIT counter reaches 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 0 to POF. If another PIT overflow occurs before the clearing sequence is complete, then writing 0 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 1 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] — Prescaler Select Bits

These read/write bits select one of the seven prescaler outputs as the input to the PIT counter as Table 12-1 shows. Reset clears the PPS[2:0] bits.

Table 12-1. Prescaler Selection

PPS[2:0]	PIT Clock Source
000	Internal bus clock ÷1
001	Internal bus clock ÷ 2
010	Internal bus clock ÷ 4
011	Internal bus clock ÷ 8
100	Internal bus clock ÷ 16
101	Internal bus clock ÷ 32
110	Internal bus clock ÷ 64
111	Internal bus clock ÷ 64

12.7.2 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

Address: $004D

	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. PIT Counter Registers (PCNTH–PCNTL)

12.7.3 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 timer 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

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 12-6. PIT Counter Modulo Registers (PMODH–PMODL)

NOTE

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

Chapter 13
Input/Output (I/O) Ports

13.1 Introduction

Fifty 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.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA) See page 155.	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB) See page 157.	Read: Write:	PTB7	PTB6	PTB25	PTB4	PTB3	PTB2	PTB1	PTB0
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC) See page 159.	Read:	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Write:								
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD) See page 161.	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA) See page 155.	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) See page 157.	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Reset:	0	0	0	0	0	0	0	0
		= Unimplemented								

Figure 13-1. I/O Port Register Summary

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0006	Data Direction Register C (DDRC) See page 159.	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) See page 161.	Read:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE) See page 163.	Read:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Write:								
		Reset:	Unaffected by reset							
$0009	Port F Data Register (PTF) See page 165.	Read:	0	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
		Write:								
		Reset:	Unaffected by reset							
$000A	Port G Data Register (PTG) See page 167.	Read:	0	0	0	0	0	PTG2	PTG1	PTG0
		Write:								
		Reset:	Unaffected by reset							
$000B	Port H Data Register (PTH) See page 169.	Read:	0	0	0	0	0	0	PTH1	PTH0
		Write:								
		Reset:	Unaffected by reset							
$000C	Data Direction Register E (DDRE) See page 164.	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) See page 166.	Read:	0	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000E	Data Direction Register G (DDRG) See page 168.	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) See page 170.	Read:	0	0	0	0	0	0	DDRH1	DDRH0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-1. I/O Port Register Summary (Continued)

13.2 Port A

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

13.2.1 Port A Data Register (PTA)

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

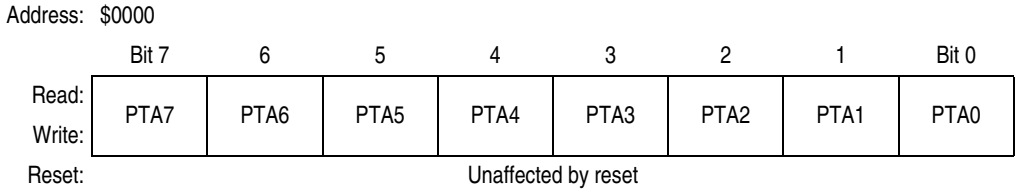


Figure 13-2. Port A Data Register (PTA)

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.

13.2.2 Data Direction Register A (DDRA)

Data direction register A determines whether each port A pin is an input or an output. Writing a 1 to a DDRA bit enables the output buffer for the corresponding port A pin; a 0 disables the output buffer.

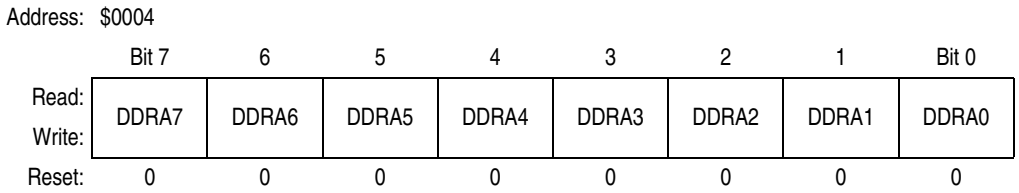


Figure 13-3. Data Direction Register A (DDRA)

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 13-4 shows the port A I/O logic.

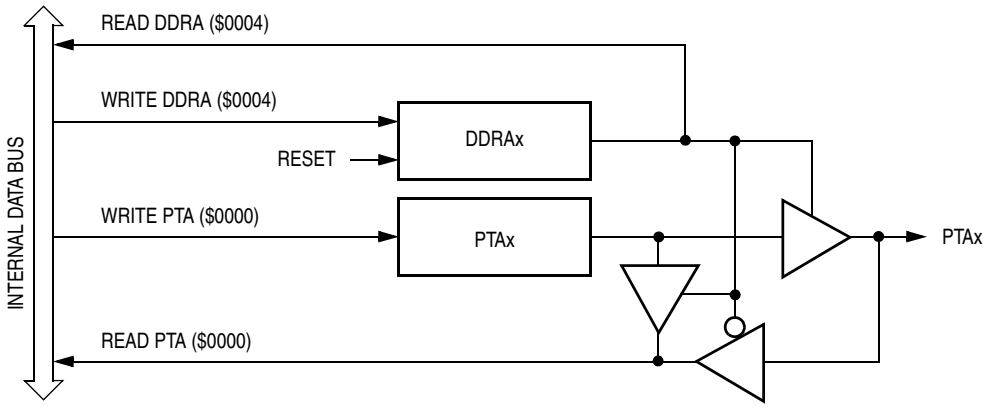


Figure 13-4. Port A I/O Circuit

When bit DDRAx is a 1, reading address $0000 reads the PTAx data latch. When bit DDRAx is a 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 13-1 summarizes the operation of the port A pins.

Table 13-1. Port A Pin Functions

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

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

13.3 Port B

Port B is an 8-bit special function port that shares all of its pins with the analog-to- digital converter (ADC).

13.3.1 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 Functions:	ATD7	ATD6	ATD5	ATD4	ATD3	ATD2	ATD1	ATD0

Figure 13-5. 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

NOTE

PTB7/ATD7–PTB0/ATD0 are eight of the analog-to-digital converter (ADC) channels. The ADC channel select bits, CH[4:0], determine whether the PTB7/ATD7–PTB0/ATD0 pins are ADC channels or general-purpose I/O pins. If an ADC channel is selected and a read of the corresponding bit in the port B data register occurs, the data will be 0 if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch. Data direction register B (DDRB) does not affect the data direction of the port B pins that are being used by the ADC. However, the DDRB bits always determine whether reading port B returns the states of the latches or a 0.

13.3.2 Data Direction Register B (DDRB)

Data direction register B determines whether each port B pin is an input or an output. Writing 1 to a DDRB bit enables the output buffer for the corresponding port B pin; a 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 13-6. 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 13-7 shows the port B I/O logic.

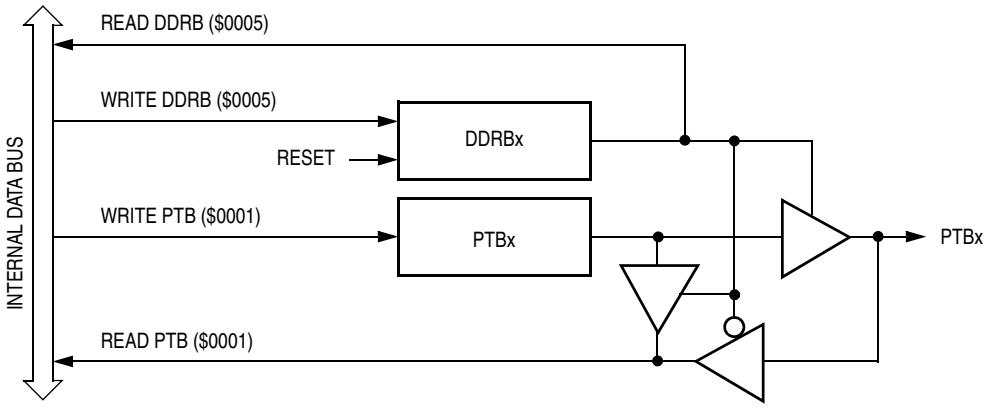


Figure 13-7. Port B I/O Circuit

When bit DDRBx is a 1, reading address $0001 reads the PTBx data latch. When bit DDRBx is a 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 13-2 summarizes the operation of the port B pins.

Table 13-2. Port B Pin Functions

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

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

13.4 Port C

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

13.4.1 Port C Data Register (PTC)

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

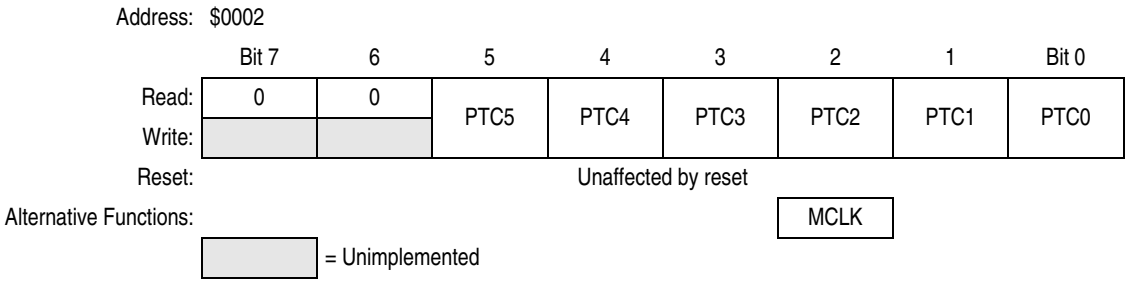


Figure 13-8. Port C Data Register (PTC)

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 — System Clock Bit

The system clock is driven out of PTC2 when enabled by MCLKEN in PTC DDR7.

13.4.2 Data Direction Register C (DDRC)

Data direction register C determines whether each port C pin is an input or an output. Writing a 1 to a DDRC bit enables the output buffer for the corresponding port C pin; a 0 disables the output buffer.

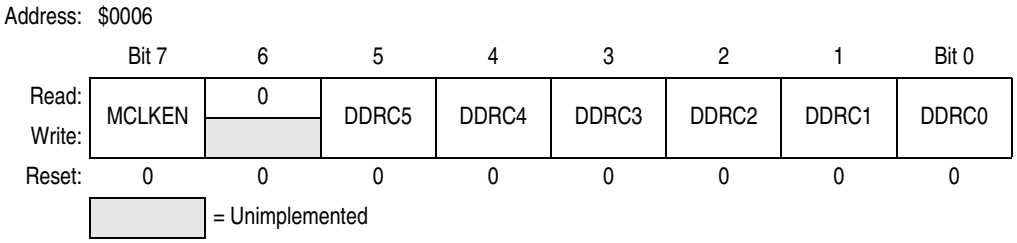


Figure 13-9. Data Direction Register C (DDRC)

MCLKEN — MCLK Enable Bit

This read/write bit enables MCLK to be an output signal on PTC2. If MCLK is enabled, DDRC2 has no effect. Reset clears this bit.

1 = MCLK output enabled
0 = MCLK output disabled

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

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

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

NOTE

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

Figure 13-10 shows the port C I/O logic.

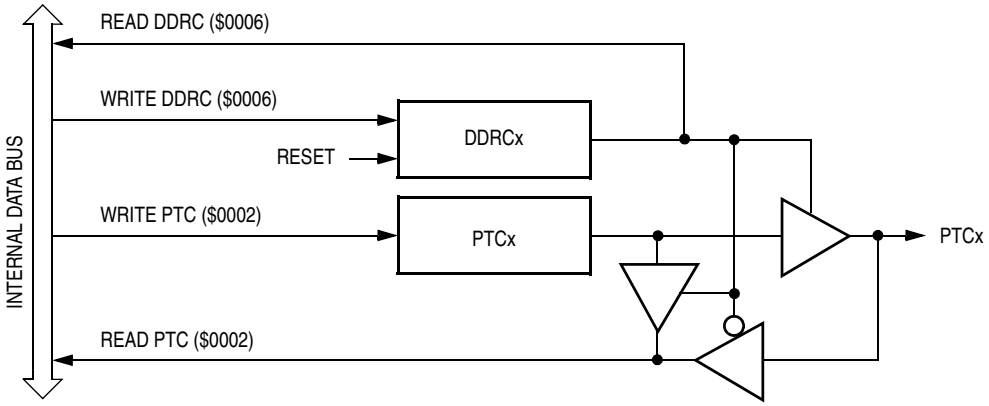


Figure 13-10. Port C I/O Circuit

When bit DDRCx is a 1, reading address $0002 reads the PTCx data latch. When bit DDRCx is a 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 13-3 summarizes the operation of the port C pins.

Table 13-3. Port C Pin Functions

DDRC Bit	PTC Bit	I/O Pin Mode	Accesses to DDRC	Accesses to PTC	
			Read/Write	Read	Write
0	2	Input, Hi-Z	DDRC[7]	Pin	PTC2
1	2	Output	DDRC[7]	0	—
0	X(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]

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

13.5 Port D

Port D is an 8 -bit special function port that shares seven of it's pins with the ADC module and two with the TIMA and TIMB modules

13.5.1 Port D Data Register (PTD)

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

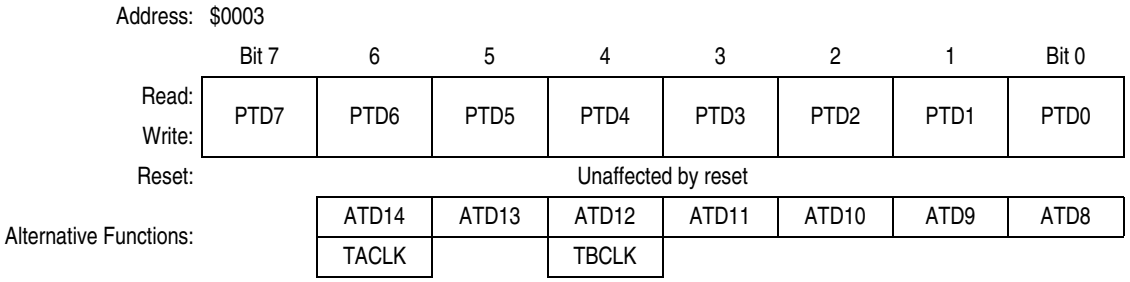


Figure 13-11. Port D Data Register (PTD)

PTD[7:0] — Port D Data Bits

PTD[7:0] are read/write, software programmable bits. Data direction of PTD[7:0] pins are under the control of the corresponding bit in data direction register D.

ATD[14:8] — ADC Channel Status Bits

PTD6/ATD14/TACLK–PTD0/ATD8 are seven of the 15 ADC channels. The ATD channel select bits, CH[4:0], determine whether the PTD6/ATD14/ TACLK–PTD0/ATD8 pins are ADC channels or general-purpose I/O pins. If an ADC channel is selected and a read of this corresponding bit in the port B data register occurs, the data will be 0 if the data direction for this bit is programmed as an input. Otherwise the data will reflect the value in the data latch.

NOTE

Data direction register D (DDRD) does not affect the data direction of port D pins that are being used by the TIMA or TIMB. However, the DDRD bits always determine whether reading port D returns the states of the latches to a 0.

TACLK/TBCLK — Timer Clock Input

The PTD6/ATD14/TACLK pin is the external clock input for the TIMA. The PTD4/TBCLK pin is the external clock input for the TIMB.The prescaler select bits, PS[2:0], select PTD6/ATD14/TACLK or PTD4/TBCLK as the TIM clock input (see 17.8.1 TIMA Status and Control Register and 18.8.1 TIMB Status and Control Register). When not selected as the TIM clock, PTD6/TACLK and PTD4/TBCLK are available for general-purpose I/O. While TACLK/TBCLK are selected, corresponding DDRD bits have no effect.

13.5.2 Data Direction Register D (DDRD)

Data direction register D determines whether each port D pin is an input or an output. Writing a 1 to a DDRD bit enables the output buffer for the corresponding port D pin; a 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 13-12. 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 13-13 shows the port D I/O logic.

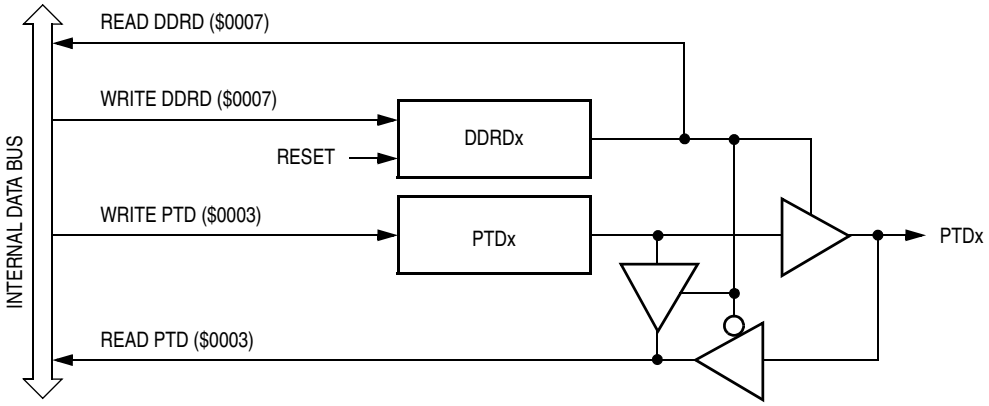


Figure 13-13. Port D I/O Circuit

When bit DDRDx is 1, reading address $0003 reads the PTDx data latch. When bit DDRDx is a 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 13-4 summarizes the operation of the port D pins.

Table 13-4. Port D Pin Functions

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

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

13.6 Port E

Port E is an 8-bit special function port that shares two of its pins with the TIMA, two of its pins with the serial communications interface module (SCI), and four of its pins with the serial peripheral interface module (SPI).

13.6.1 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 Functions:	SPSCK	MOSI	MISO	SS	TACH1	TACH0	RxD	TxD

Figure 13-14. Port E Data Register (PTE)

PTE[7:0] — Port E Data Bits

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

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.

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 16.12.1 SPI Control Register (SPCR).

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 16.12.1 SPI Control Register (SPCR).

$\overline{\text{SS}}$ — Slave Select

The PTE4/$\overline{\text{SS}}$ 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 and MODFEN bit is low, the PTE4/$\overline{\text{SS}}$ pin is available for general-purpose I/O. See 16.12.1 SPI Control Register (SPCR). When the SPI is enabled as a slave, the DDRF0 bit in data direction register E (DDRE) has no effect on the PTE4/$\overline{\text{SS}}$ pin.

NOTE

Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SPI 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 13-5.

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 17.8.1 TIMA Status and Control Register.

NOTE

Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the TIMA. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. See Table 13-5.

RxD — SCI Receive Data Input

The PTE1/RxD pin is the receive data input for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE1/RxD pin is available for general-purpose I/O. See 14.8.1 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 14.8.2 SCI Control Register 2.

NOTE

Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SCI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. See Table 13-5.

13.6.2 Data Direction Register E (DDRE)

Data direction register E determines whether each port E pin is an input or an output. Writing a 1 to a DDRE bit enables the output buffer for the corresponding port E pin; a 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 13-15. 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 13-16 shows the port E I/O logic.

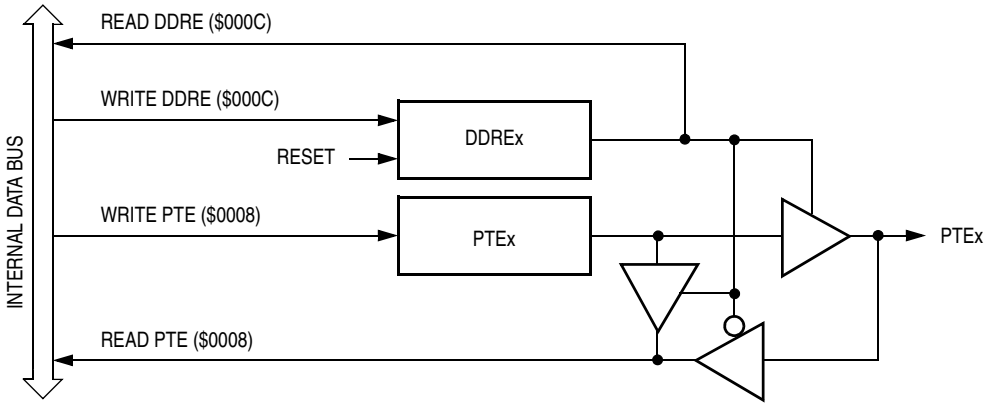


Figure 13-16. Port E I/O Circuit

When bit DDREx is a 1, reading address $0008 reads the PTEx data latch. When bit DDREx is a 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 13-5 summarizes the operation of the port E pins.

Table 13-5. Port E Pin Functions

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

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

13.7 Port F

Port F is a 7-bit special function port that shares four of its pins with TIMA and two of its pins with TIMB.

13.7.1 Port F Data Register (PTF)

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

Address: $0009

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	PTF6	PTF5	PTF4	PTF3	PTF2	PTF1	PTF0
Write:								
Reset:	Unaffected by reset							
Alternative Functions:			TBCH1	TBCH0	TACH5	TACH4	TACH3	TACH2

= Unimplemented

Figure 13-17. Port F Data Register (PTF)

PTF[6:0] — Port F Data Bits

These read/write bits are software programmable. Data direction of each port F pin is under the control of the corresponding bit in data direction register F. Reset has no effect on PTF[6:0].

TACH[5:2] — Timer A Channel I/O Bits

The PTF3/TACH5–PTF0/TACH2 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF3/TACH5–PTF0/TACH2 pins are timer channel I/O pins or general-purpose I/O pins.

TBCH[1:0] — Timer B Channel I/O Bits

The PTF5/TBCH1–PTF4/TBCH0 pins are the TIMB input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF5/TBCH1–PTF4/TBCH0 pins are timer channel I/O pins or general-purpose I/O pins. See 18.8.1 TIMB Status and Control Register.

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 13-6.

13.7.2 Data Direction Register F (DDRF)

Data direction register F determines whether each port F pin is an input or an output. Writing a 1 to a DDRF bit enables the output buffer for the corresponding port F pin; a 0 disables the output buffer.

Address: $000D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	DDRF6	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-18. Data Direction Register F (DDRF)

DDRF[6:0] — Data Direction Register F Bits

These read/write bits control port F data direction. Reset clears DDRF[6:0], 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 13-19 shows the port F I/O logic.

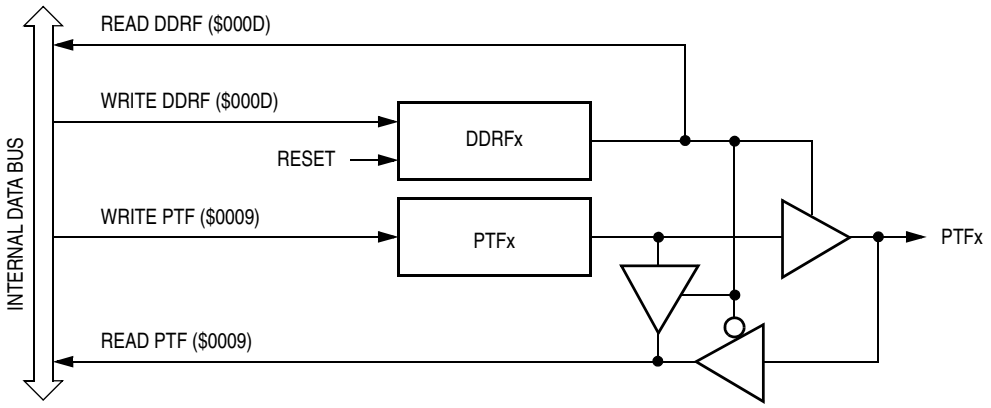


Figure 13-19. Port F I/O Circuit

When bit DDRFx is a 1, reading address $0009 reads the PTFx data latch. When bit DDRFx is a 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 13-6 summarizes the operation of the port F pins.

Table 13-6. Port F Pin Functions

DDRF Bit	PTF Bit	I/O Pin Mode	Accesses to DDRF	Accesses to PTF	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRF[6:0]	Pin	PTF[6:0](3)
1	X	Output	DDRF[6:0]	PTF[6:0]	PTF[6:0]

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

13.8 Port G

Port G is a 3-bit special function port that shares all of its pins with the KBD.

13.8.1 Port G Data Register (PTG)

The port G data register 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 Functions:						KBD2	KBD1	KBD0

= Unimplemented

Figure 13-20. 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 PTG[2:0].

KBD[2:0] — Keyboard Wakeup Pins

The keyboard interrupt enable bits, KBIE[2:0], in the keyboard interrupt control register (KBICR), enable the port G pins as external interrupt pins. See Chapter 8 Keyboard Interrupt (KBD) Module. Enabling an external interrupt pin will override the corresponding DDRGx.

13.8.2 Data Direction Register G (DDRG)

Data direction register G determines whether each port G pin is an input or an output. Writing a 1 to a DDRG bit enables the output buffer for the corresponding port G pin; a 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

= Unimplemented

Figure 13-21. 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 13-22 shows the port G I/O logic.

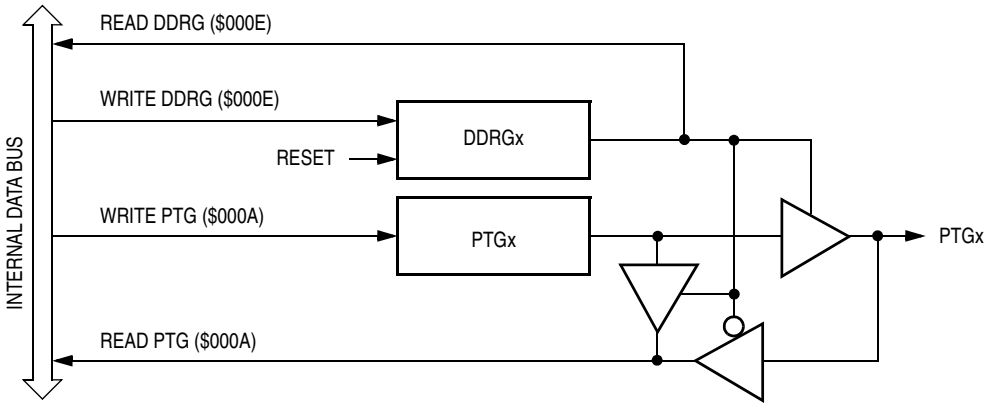


Figure 13-22. Port G I/O Circuit

When bit DDRGx is a 1, reading address $000A reads the PTGx data latch. When bit DDRGx is a 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. Table 13-7 summarizes the operation of the port G pins.

Table 13-7. Port G Pin Functions

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

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

13.9 Port H

Port H is a 2-bit special function port that shares all of its pins with the KBD.

13.9.1 Port H Data Register (PTH)

The port H data register contains a data latch for each of the two port H pins.

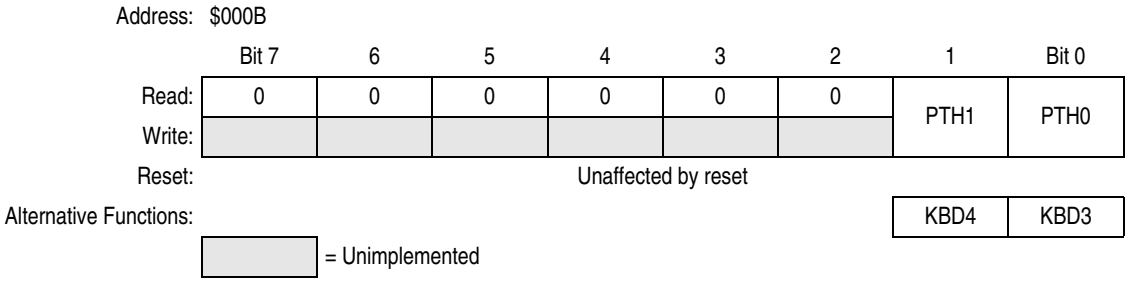


Figure 13-23. Port H Data Register (PTH)

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] — Keyboard Wakeup Pins

The keyboard interrupt enable bits, KBIE[4:3], in the keyboard interrupt control register (KBICR), enable the port H pins as external interrupt pins. See Chapter 8 Keyboard Interrupt (KBD) Module.

13.9.2 Data Direction Register H (DDRH)

Data direction register H determines whether each port H pin is an input or an output. Writing a 1 to a DDRH bit enables the output buffer for the corresponding port H pin; a 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

= Unimplemented

Figure 13-24. 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 13-25 shows the port H I/O logic.

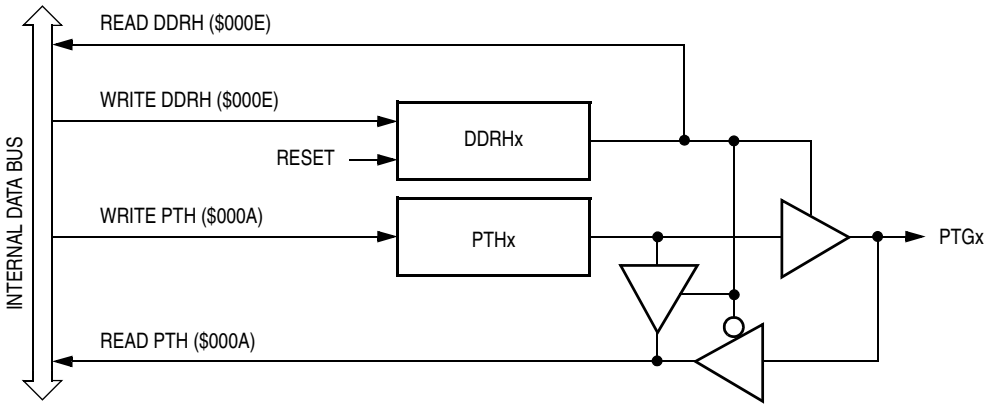


Figure 13-25. Port H I/O Circuit

When bit DDRHx is a 1, reading address $000B reads the PTHx data latch. When bit DDRHx is a 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. Table 13-8 summarizes the operation of the port H pins.

Table 13-8. Port H Pin Functions

DDRH Bit	PTH Bit	I/O Pi 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]

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

Chapter 14
Serial Communications Interface (SCI)

14.1 Introduction

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

14.2 Features

Features include:

- 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

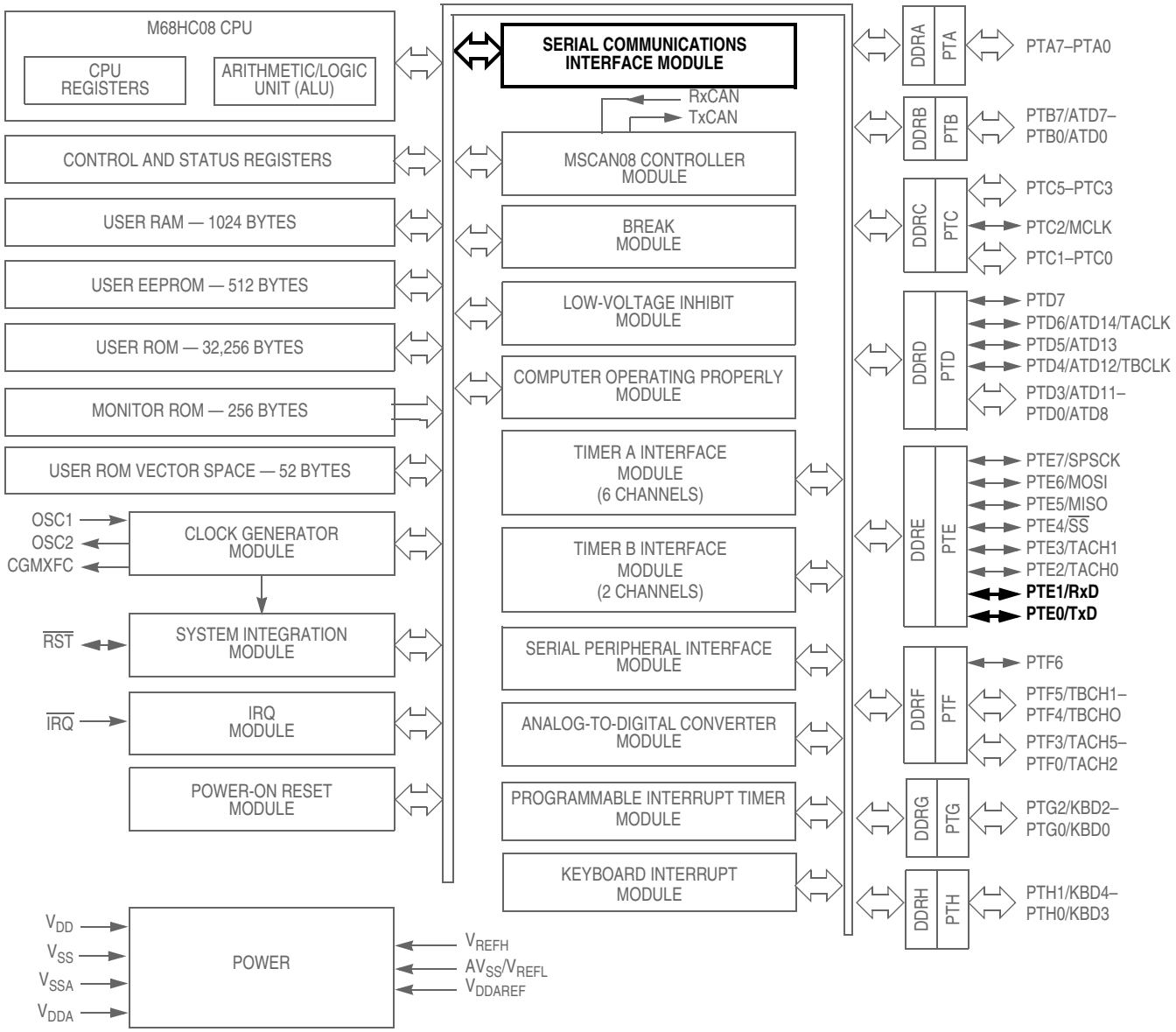


Figure 14-1. Block Diagram Highlighting SCI Block and Pins

14.3 Pin Name Conventions

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

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

SCI I/O 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 14-1 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 14-1. Pin Name Conventions

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

14.4 Functional Description

Figure 14-3 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.

14.4.1 Data Format

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

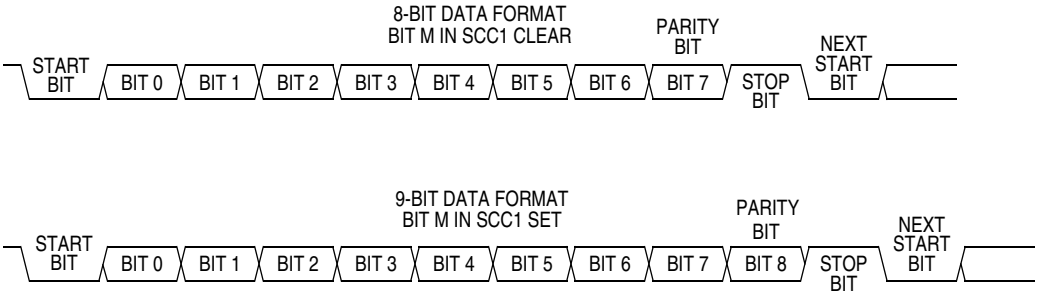


Figure 14-2. SCI Data Formats

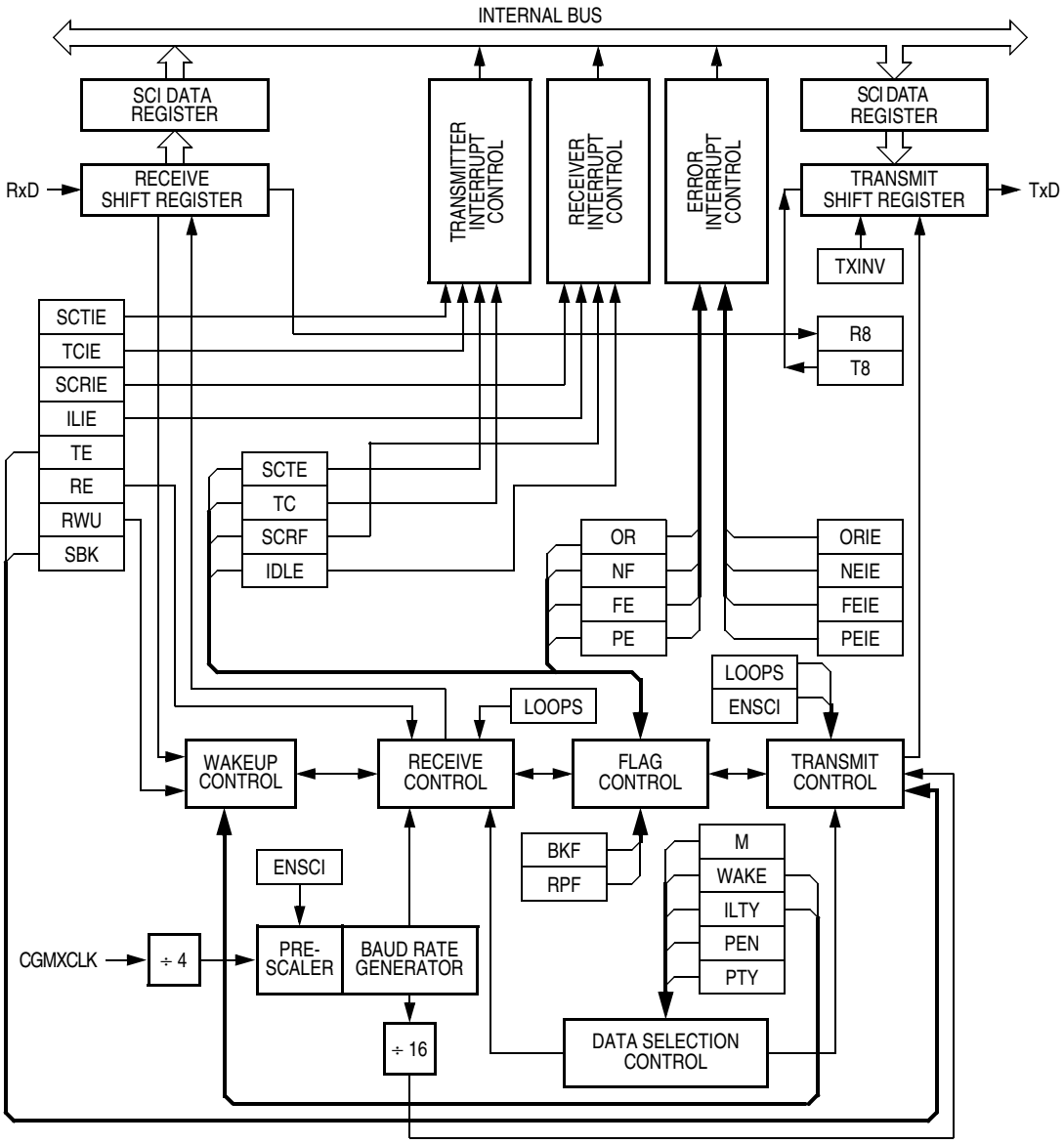


Figure 14-3. SCI Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0013	SCI Control Register 1 (SCC1) See page 186.	Read: Write:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Reset:	0	0	0	0	0	0	0	0
$0014	SCI Control Register 2 (SCC2) See page 188.	Read: Write:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Reset:	0	0	0	0	0	0	0	0
$0015	SCI Control Register 3 (SCC3) See page 190.	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) See page 191.	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) See page 193.	Read:							BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	SCI Data Register (SCDR) See page 194.	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) See page 194.	Read:			SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Figure 14-4. SCI I/O Register Summary

14.4.2 Transmitter

Figure 14-5 shows the structure of the SCI transmitter.

14.4.2.1 Character Length

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

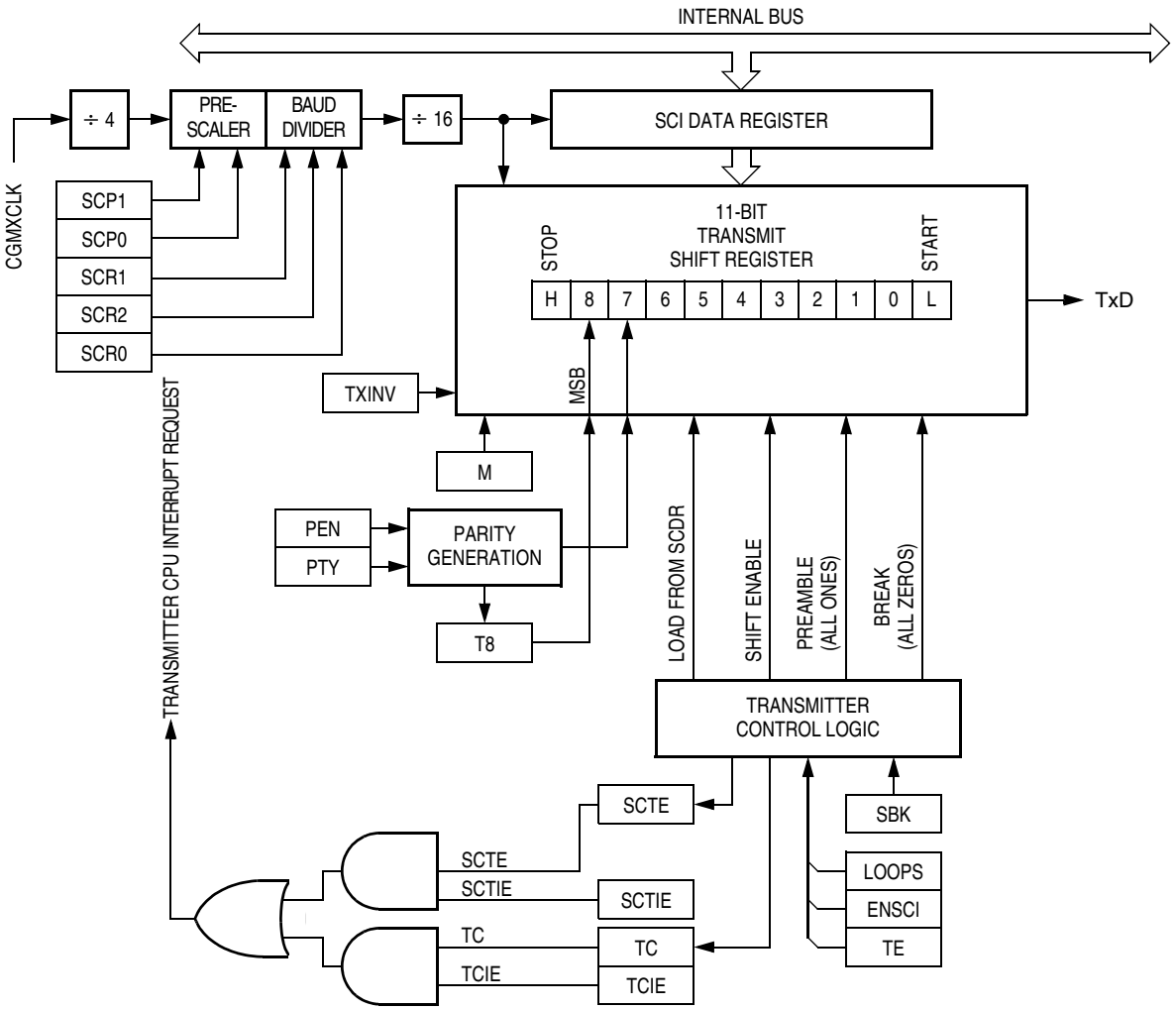


Figure 14-5. SCI Transmitter

14.4.2.2 Character Transmission

During an SCI transmission, the transmit shift register shifts a character out to the 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 1 to the enable SCI bit (ENSCI) in SCI control register 1 (SCC1).
2. Enable the transmitter by writing a 1 to the transmitter enable bit (TE) in SCI control register 2 (SCC2).
3. Clear the SCI transmitter empty bit (SCTE) 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 1s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A 0 start bit automatically goes into the least significant bit position of the transmit shift register. A 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 TxD pin goes to the idle condition, 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.

14.4.2.3 Break Characters

Writing a 1 to the send break bit, SBK, in SCC2 loads the transmit shift register with a break character. A break character contains all 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 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 1. The automatic 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 0 data bits and a 0 where the stop bit should be.

Receiving a break character has the following 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

14.4.2.4 Idle Characters

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

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

NOTE

This note does not apply to the L32X mask set of the MC68HC08AZ32A. When a break sequence is followed immediately by an idle character, this SCI design exhibits a condition in which the break character length is reduced by one half bit time. In this instance, the break sequence will consist of a valid start bit, eight or nine data bits (as defined by the M bit in SCC1) of 0 and one half data bit length of 0 in the stop bit position followed immediately by the idle character. To ensure a break character of the proper length is transmitted, always queue up a byte of data to be transmitted while the final break sequence is in progress.

NOTE

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

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

14.4.2.5 Inversion of Transmitted Output

The transmit inversion bit (TXINV) in SCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values, including idle, break, start, and stop bits, are inverted when TXINV is at 1. See 14.8.1 SCI Control Register 1.

14.4.2.6 Transmitter Interrupts

The following conditions can generate CPU interrupt requests from the SCI transmitter:

- SCI transmitter empty (SCTE) — The SCTE bit in SCS1 indicates that the SCDR has transferred a character to the transmit shift register. SCTE can generate 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.

14.4.3 Receiver

Figure 14-6 shows the structure of the SCI receiver.

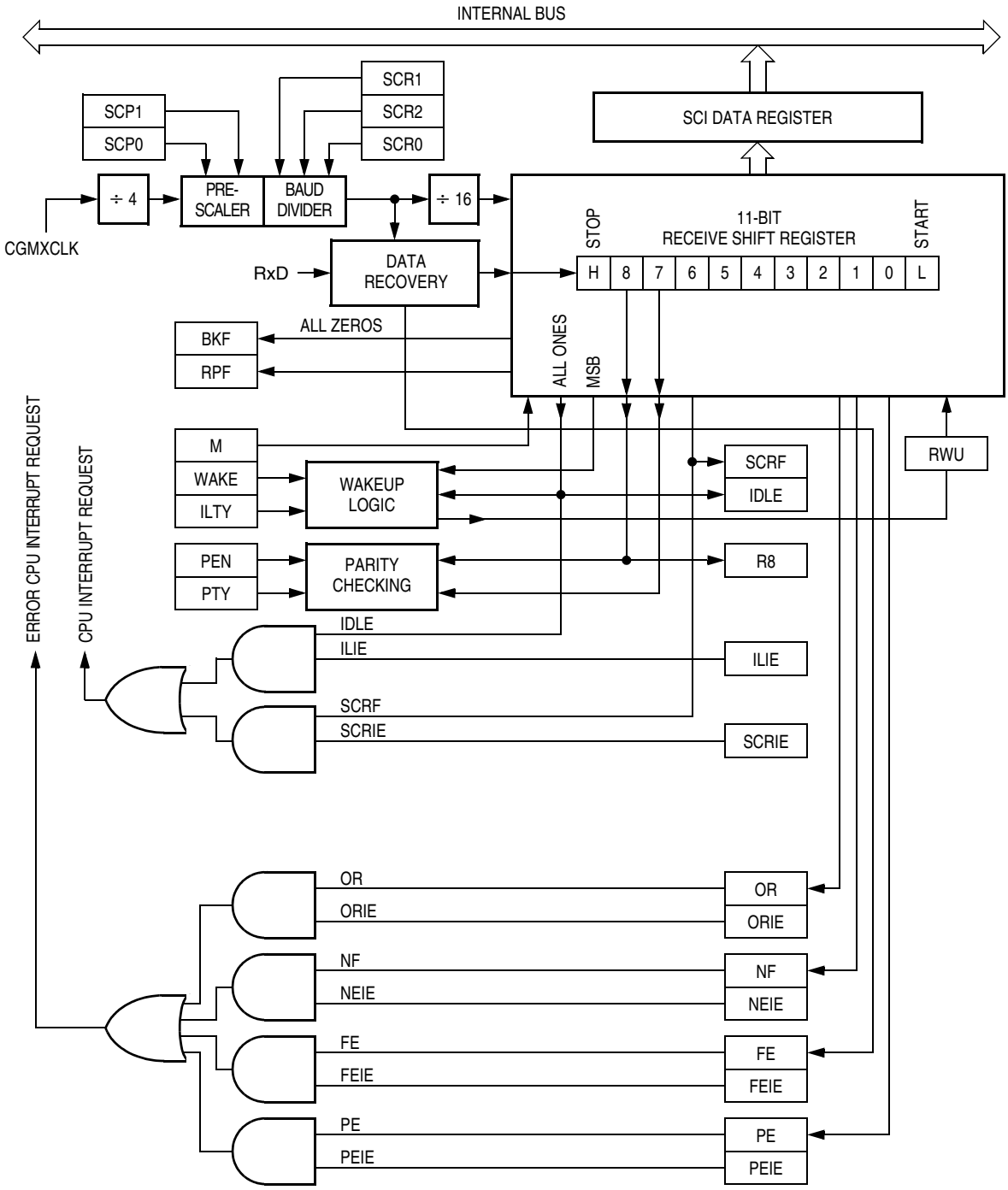


Figure 14-6. SCI Receiver Block Diagram

14.4.3.1 Character Length

The receiver can accommodate either 8-bit or 9-bit data. The state of the M bit in SCI control register 1 (SCC1) determines character length. When receiving 9-bit data, bit R8 in SCI control register 2 (SCC2) is the ninth bit (bit 8). When receiving 8-bit data, bit R8 is a copy of the eighth bit (bit 7).

14.4.3.2 Character Reception

During an SCI reception, the receive shift register shifts characters in from the 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.

14.4.3.3 Data Sampling

The receiver samples the RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at the following times (see Figure 14-7):

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

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

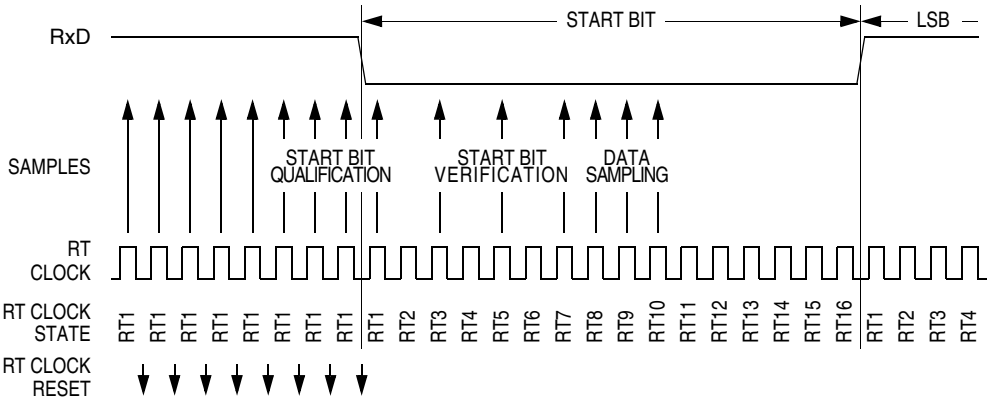


Figure 14-7. Receiver Data Sampling

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

Table 14-2. Start Bit Verification

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

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

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

Table 14-3. Data Bit Recovery

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

NOTE

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

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

Table 14-4. Stop Bit Recovery

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

14.4.3.4 Framing Errors

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

14.4.3.5 Baud Rate Tolerance

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

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

Slow Data Tolerance

Figure 14-8 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.

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

With the misaligned character shown in Figure 14-8, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit times × 16 RT cycles + 3 RT cycles = 147 RT cycles.

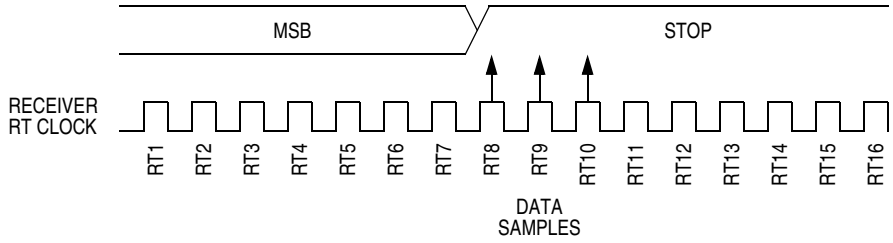


Figure 14-8. Slow Data

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

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

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

With the misaligned character shown in Figure 14-8, the receiver counts 170 RT cycles at the point when the count of the transmitting device is
10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles.

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

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

Fast Data Tolerance

Figure 14-9 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

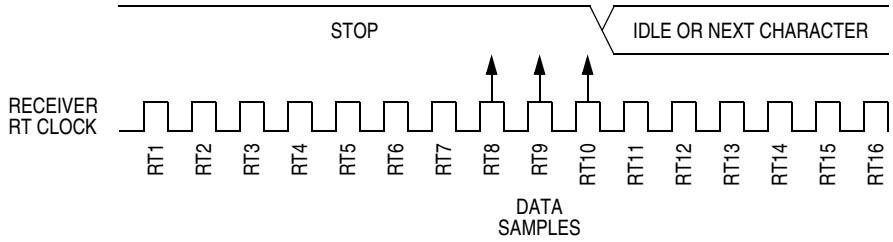


Figure 14-9. Fast Data

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

With the misaligned character shown in Figure 14-9, 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

$$\left|\frac{154 - 160}{154}\right| \times 100 = 3.90\dot{0}\%$$

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

With the misaligned character shown in Figure 14-9, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times × 16 RT cycles = 176 RT cycles.

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

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

14.4.3.6 Receiver Wakeup

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

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

- Address mark — An address mark is a 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 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 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.

14.4.3.7 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 1s shifted in from the RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables the IDLE bit to generate CPU interrupt requests.

14.4.3.8 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 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.

14.5 Low-Power Modes

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

14.5.1 Wait Mode

The SCI module remains active in wait mode. 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.

14.5.2 Stop Mode

The SCI module is inactive in stop mode. The STOP instruction does not affect SCI register states. Any enabled CPU interrupt request from the SCI module does not bring the MCU out of Stop mode. SCI module operation resumes after the MCU exits stop mode.

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

14.6 SCI During Break Module Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See 19.2 Break Module (BRK).

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

14.7 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

14.7.1 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).

14.7.2 PTE1/RxD (Receive Data)

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

14.8 I/O Registers

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

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

14.8.1 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	ILLTY	PEN	PTY
Reset:	0	0	0	0	0	0	0	0

Figure 14-10. SCI Control Register 1 (SCC1)

LOOPS — Loop Mode Select Bit

This read/write bit enables loop mode operation. In loop mode the RxD pin is disconnected from the 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 14-5).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 1 (address mark) in the most significant bit position of a received character or an idle condition on the RxD pin. Reset clears the WAKE bit.

1 = Address mark wakeup
0 = Idle line wakeup

ILTY — Idle Line Type Bit

This read/write bit determines when the SCI starts counting 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 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 14-5). When enabled, the parity function inserts a parity bit in the most significant bit position. (See Table 14-4). Reset clears the PEN bit.

1 = Parity function enabled
0 = Parity function disabled

PTY — Parity Bit

This read/write bit determines whether the SCI generates and checks for odd parity or even parity. (See Table 14-5). 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 14-5. Character Format Selection

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

14.8.2 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 14-11. 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. Setting the SCTIE bit in SCC3 enables the SCTE bit to generate CPU interrupt requests. Reset clears the SCTIE bit.

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

TCIE — Transmission Complete Interrupt Enable Bit

This read/write bit enables the TC bit to generate 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. Setting the SCRIE bit in SCC3 enables the SCRF bit to generate CPU interrupt requests. Reset clears the SCRIE bit.

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

ILIE — Idle Line Interrupt Enable Bit

This read/write bit enables the IDLE bit to generate 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 1s from the transmit shift register to the TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the TxD returns to the idle condition (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 1. The 1 after the break character guarantees recognition of a valid start bit. If SBK remains set, the transmitter continuously transmits break characters with no 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.

14.8.3 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 the following interrupts:
  - Receiver overrun interrupts
  - Noise error interrupts
  - Framing error interrupts
  - Parity error interrupts

Figure 14-12. 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. Reset clears PEIE.

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

14.8.4 SCI Status Register 1

SCI status register 1 contains flags to signal the following conditions:

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

Address: $0016

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

= Unimplemented

Figure 14-13. 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 cleared automatically when data, preamble, or break is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queueing data, preamble, and break and the transmission actually starting. Reset sets the TC bit.

1 = No transmission in progress
0 = Transmission in progress

SCRF — 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 the SCRF generates a CPU interrupt request. In normal operation, clear the SCRF bit by reading SCS1 with SCRF set and then reading the SCDR. Reset clears SCRF.

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

IDLE — Receiver Idle Bit

This clearable, read-only bit is set when 10 or 11 consecutive 1s appear on the receiver input. IDLE generates an SCI receiver 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 14-14 shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

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

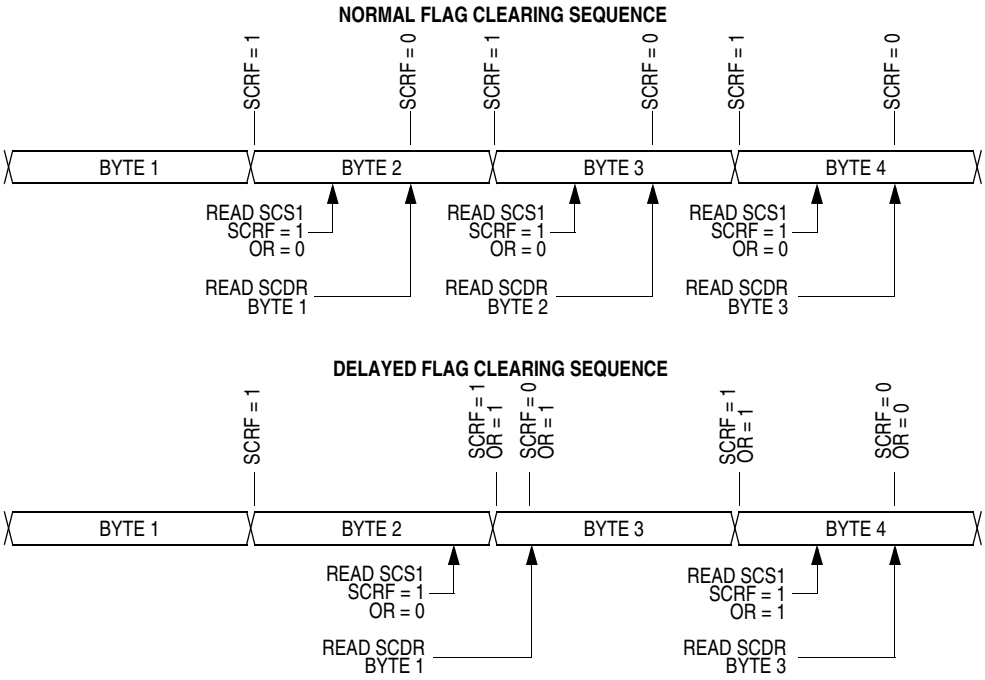


Figure 14-14. Flag Clearing Sequence

NF — Receiver Noise Flag Bit

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

1 = Noise detected
0 = No noise detected

FE — Receiver Framing Error Bit

This clearable, read-only bit is set when a 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

PE — Receiver Parity Error Bit

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

1 = Parity error detected
0 = No parity error detected

14.8.5 SCI Status Register 2

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

- Break character detected
- Incoming data

Address: $0017

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

= Unimplemented

Figure 14-15. 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 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 1s again appear on the RxD pin followed by another break character. Reset clears the BKF bit.

1 = Break character detected
0 = No break character detected

RPF — Reception in Progress Flag Bit

This read-only bit is set when the receiver detects a 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

14.8.6 SCI Data Register

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

Address: $0018

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

Figure 14-16. 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.

14.8.7 SCI Baud Rate Register

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

Address: $0019

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

= Unimplemented R = Reserved

Figure 14-17. SCI Baud Rate Register (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in Table 14-6. Reset clears SCP1 and SCP0.

Table 14-6. SCI Baud Rate Prescaling

SCP[1:0]	Prescaler Divisor (PD)
00	1
01	3
10	4
11	13

SCR2 – SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in Table 14-7. Reset clears SCR2–SCR0.

Table 14-7. SCI Baud Rate Selection

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

Use the following formula to calculate the SCI baud rate:

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

where:

$f_{Crystal}$ = crystal frequency
PD = prescaler divisor
BD = baud rate divisor

Table 14-8 shows the SCI baud rates that can be generated with a 4.9152-MHz crystal.

Table 14-8. SCI Baud Rate Selection Examples

SCP[1:0]	Prescaler Divisor (PD)	SCR[2:1:0]	Baud Rate Divisor (BD)	Baud Rate ($f_{Crystal}$ = 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

Chapter 15
System Integration Module (SIM)

15.1 Introduction

This section describes the system integration module, 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 15-2. Table 15-1 is a summary of the SIM I/O registers. The SIM is a system state controller that coordinates CPU and exception timing.

The SIM is responsible for:

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

A block diagram of the SIM is shown in Figure 15-2.

Figure 15-3 is a summary of the SIM input/output (I/O) registers.

Table 15-1 shows the internal signal names used in this section.

Table 15-1. Signal Naming Conventions

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

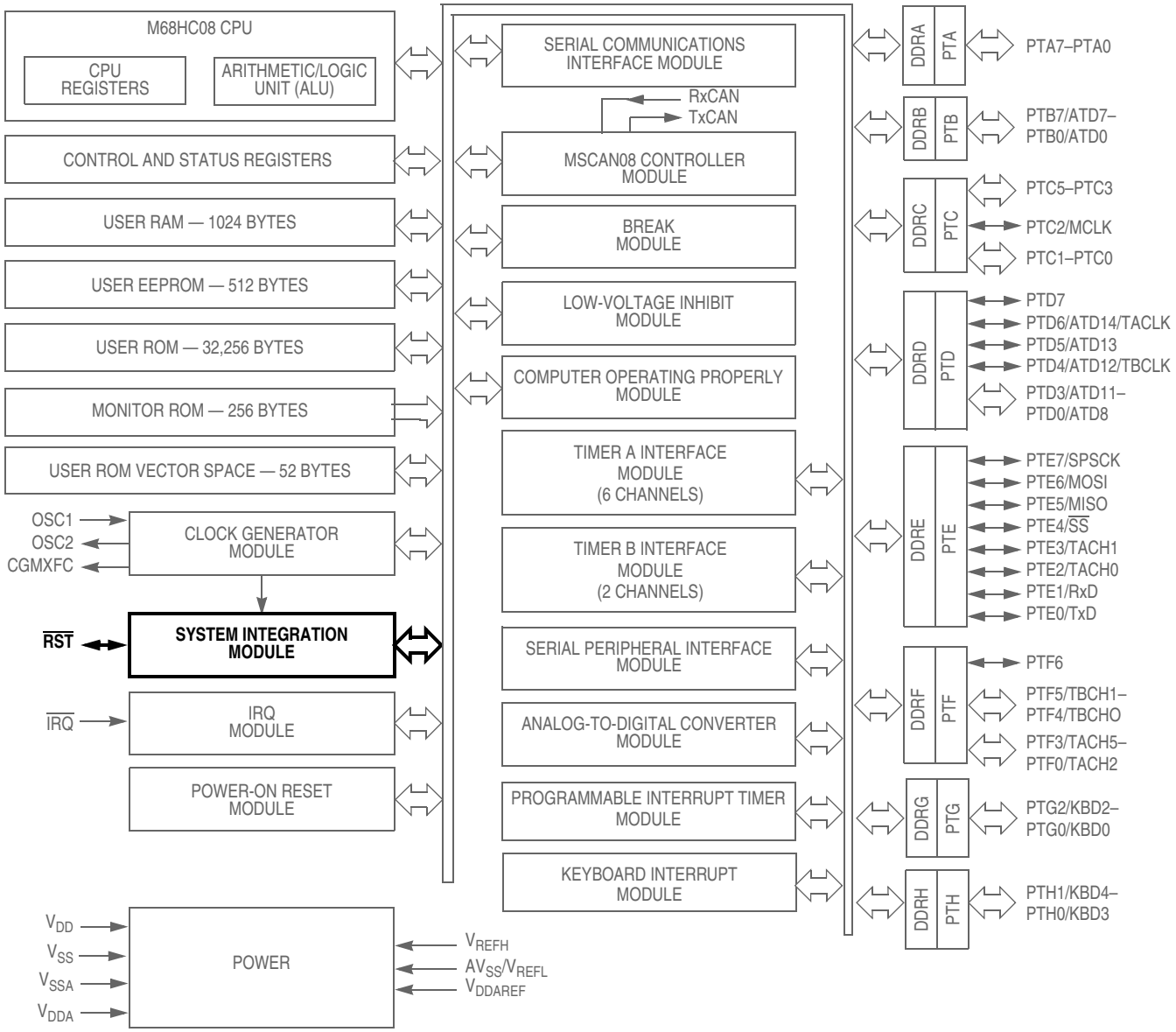


Figure 15-1. Block Diagram Highlighting SIM Block and Pin

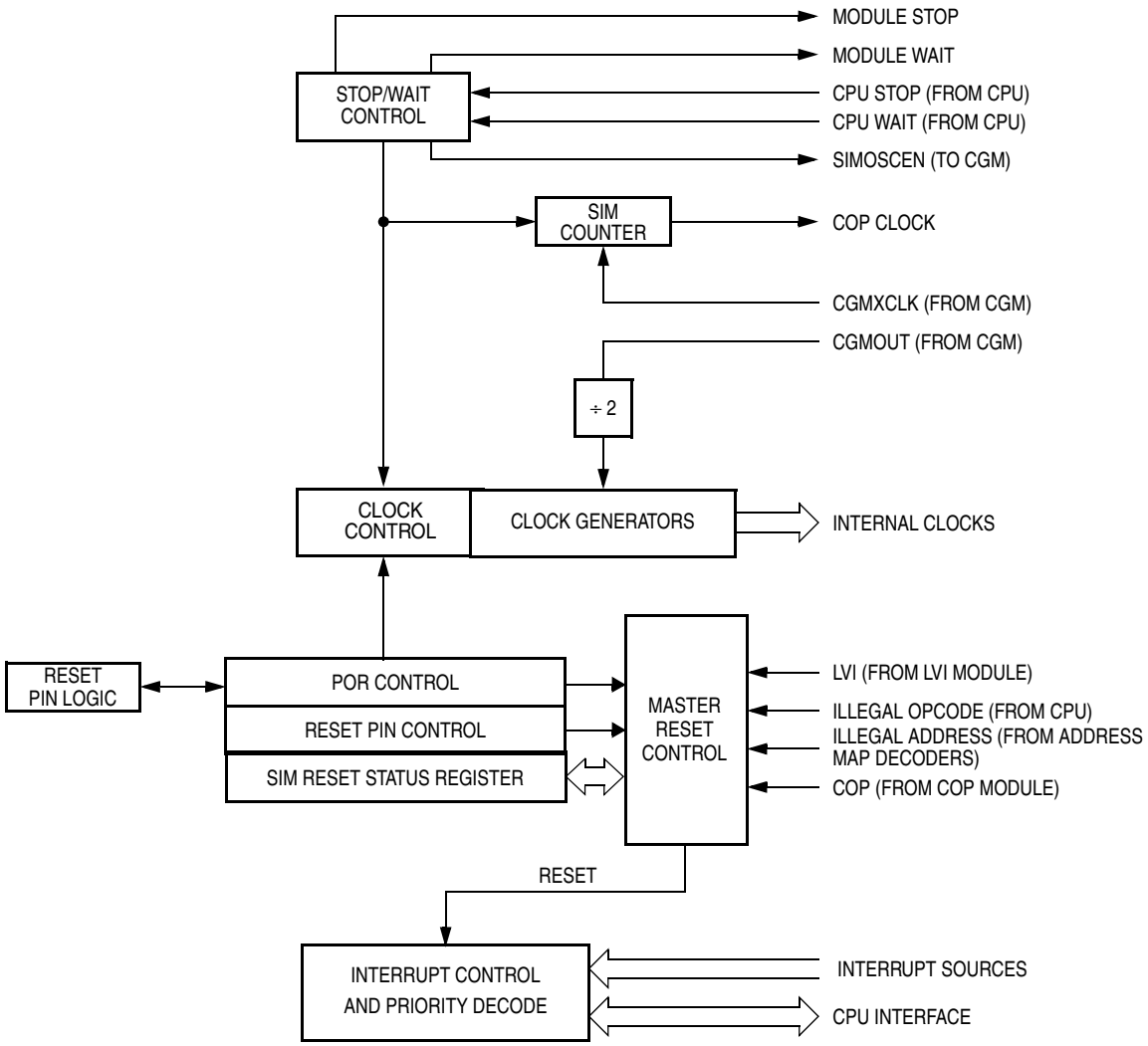


Figure 15-2. SIM Block Diagram

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	SIM Break Status Register (SBSR) See page 210.	Read:	R	R	R	R	R	R	BW	R
		Write:							0	
		Reset:							0	
$FE01	SIM Reset Status Register (SRSR) See page 210.	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) See page 211.	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							

R = Reserved = Unimplemented

Figure 15-3. SIM I/O Register Summary

15.2 SIM Bus Clock Control and Generation

The bus clock generator provides system clock signals for the CPU and peripherals on the MCU. The system clocks are generated from an incoming clock, CGMOUT, as shown in Figure 15-4. This clock can come from either an external oscillator or from the on-chip PLL. Chapter 4 Clock Generator Module (CGM).

15.2.1 Bus Timing

In user mode, the internal bus frequency is either the crystal oscillator output (CGMXCLK) divided by four or the PLL output (CGMVCLK) divided by four. Chapter 4 Clock Generator Module (CGM).

15.2.2 Clock Start-Up From POR or LVI Reset

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

15.2.3 Clocks in Stop and Wait Mode

Upon exit from stop mode by an interrupt or reset, the SIM allows CGMXCLK to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay timeout. This timeout is selectable as 4096 or 32 CGMXCLK cycles. (See 15.6.2 Stop Mode.)

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

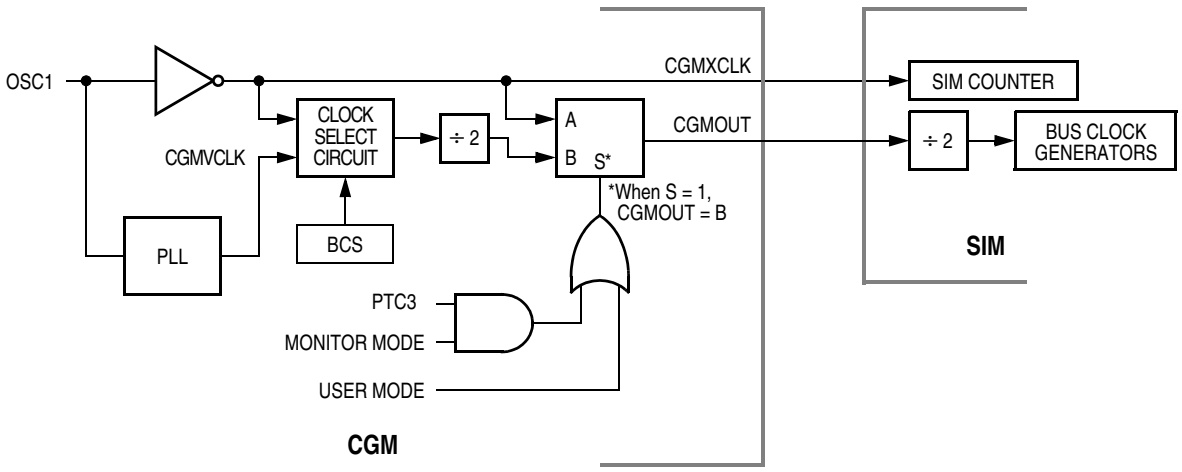


Figure 15-4. CGM Clock Signals

15.3 Reset and System Initialization

The MCU has these reset sources:

- Power-on reset module (POR)
- External reset pin ($\overline{RST}$)
- 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, 15.3.3 SIM Counter, but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See 15.7 SIM Registers.)

15.3.1 External Pin Reset

Pulling the asynchronous $\overline{RST}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{RST}$ is held low for at least the minimum t_{RL} time. Figure 15-5 shows the relative timing.

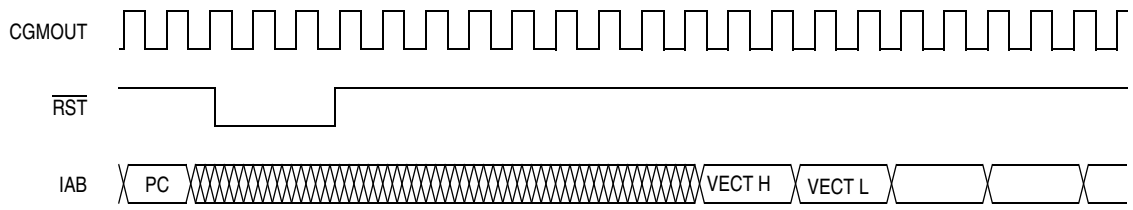


Figure 15-5. External Reset Timing

15.3.2 Active Resets From Internal Sources

All internal reset sources actively pull the $\overline{RST}$ 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 15-6. An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR. See Figure 15-7. Note that for LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles, during which the SIM forces the $\overline{RST}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{RST}$ as shown in Figure 15-6.

The COP reset is asynchronous to the bus clock.

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

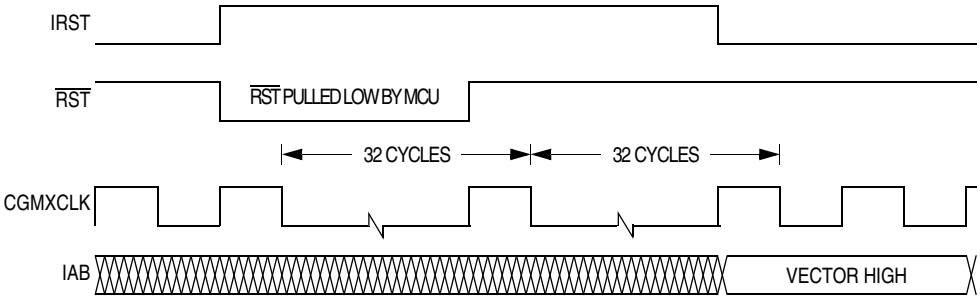


Figure 15-6. Internal Reset Timing

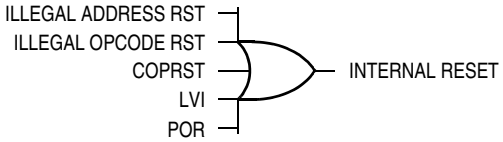


Figure 15-7. Sources of Internal Reset

Table 15-2. Reset Recovery Timing

Reset Recovery Type	Actual Number of Cycles
POR/LVI	4163 (4096 + 64 + 3)
All Others	67 (64 + 3)

15.3.2.1 Power-On Reset

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

At power-on, these events occur:

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

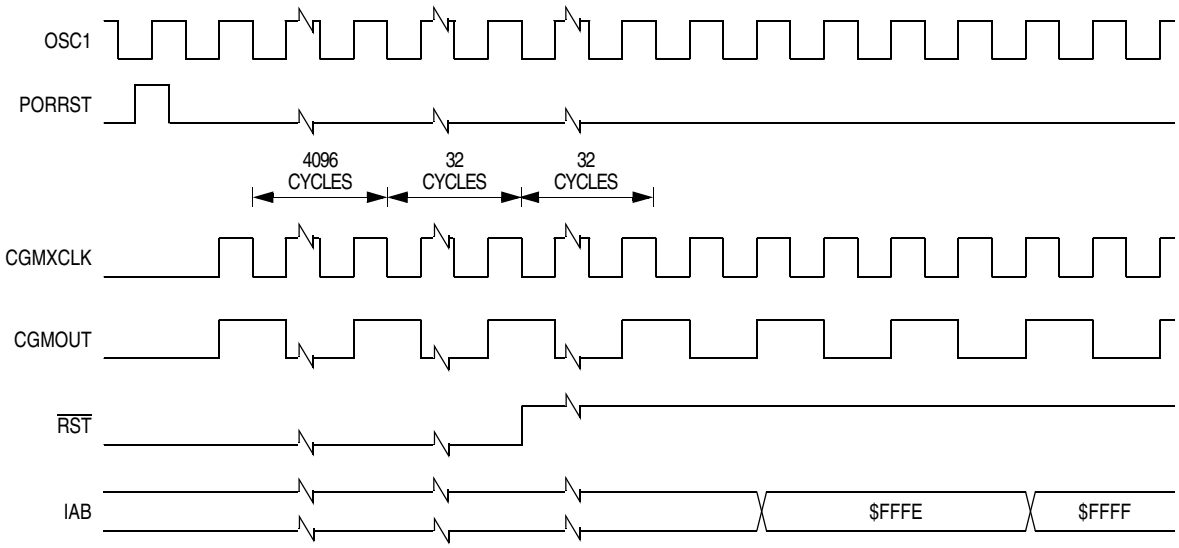


Figure 15-8. POR Recovery

15.3.2.2 Computer Operating Properly (COP) Reset

An input to the SIM is reserved for the COP reset signal. The overflow of the COP counter causes an internal reset and sets the COP bit in the SIM reset status register (SRSR). The SIM actively pulls down the $\overline{\text{RST}}$ 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 8176 CGMXCLK cycles, drives the COP counter. The COP should be serviced as soon as possible out of reset to guarantee the maximum amount of time before the first timeout.

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

15.3.2.3 Illegal Opcode Reset

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

If the stop enable bit, STOP, in the mask option register is 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.

15.3.2.4 Illegal Address Reset

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

NOTE

Extra care should be exercised if code in this part has been taken from another M68HC08 with a different memory map since some legal addresses could become illegal addresses on a smaller ROM. It is the user's responsibility to check their code for illegal addresses. Older M68HC08s may have a different illegal address reset specification.

15.3.2.5 Low-Voltage Inhibit (LVI) Reset

The low-voltage inhibit module (LVI) asserts its output to the SIM when the V_{DD} voltage falls to the $LVI_{\overline{TRIPF}}$ voltage. 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.

15.3.3 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.

15.3.4 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.

15.3.5 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 or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the mask option register. If the SSREC bit is a 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 startup times from stop mode. External crystal applications should use the full stop recovery time, that is, with SSREC cleared.

15.3.6 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See 15.6.2 Stop Mode for details.) The SIM counter is free-running after all reset states, see 15.3.2 Active Resets From Internal Sources for counter control and internal reset recovery sequences.

15.4 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

15.4.1 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 15-9 shows interrupt entry timing, and Figure 15-10 shows interrupt recovery timing.

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt may take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared). See Figure 15-10.

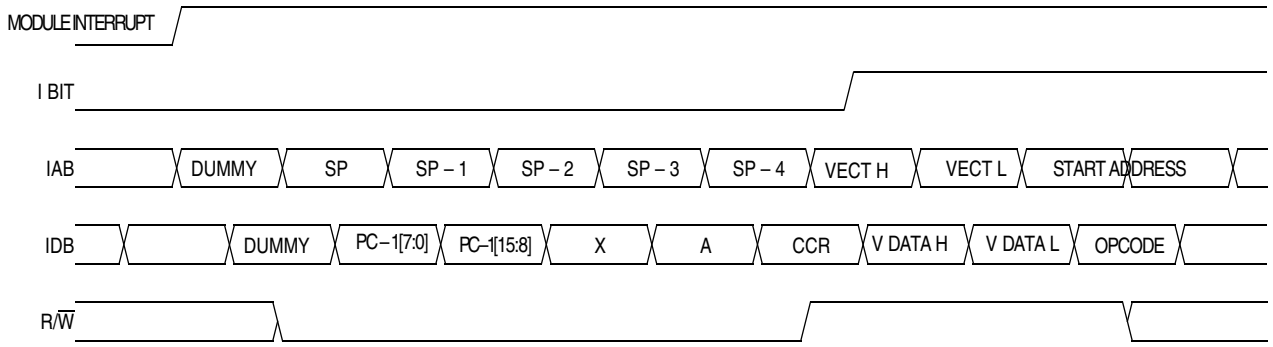


Figure 15-9. Interrupt Entry

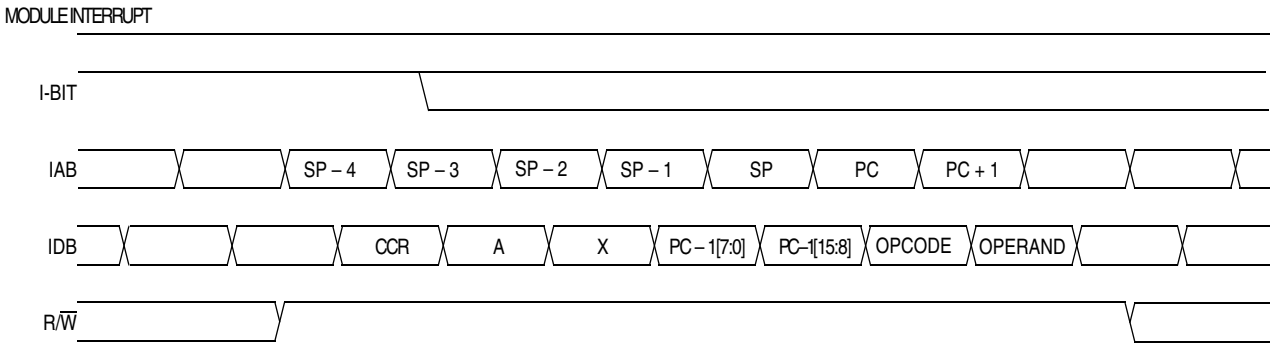


Figure 15-10. Interrupt Recovery

15.4.1.1 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 15-11 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.

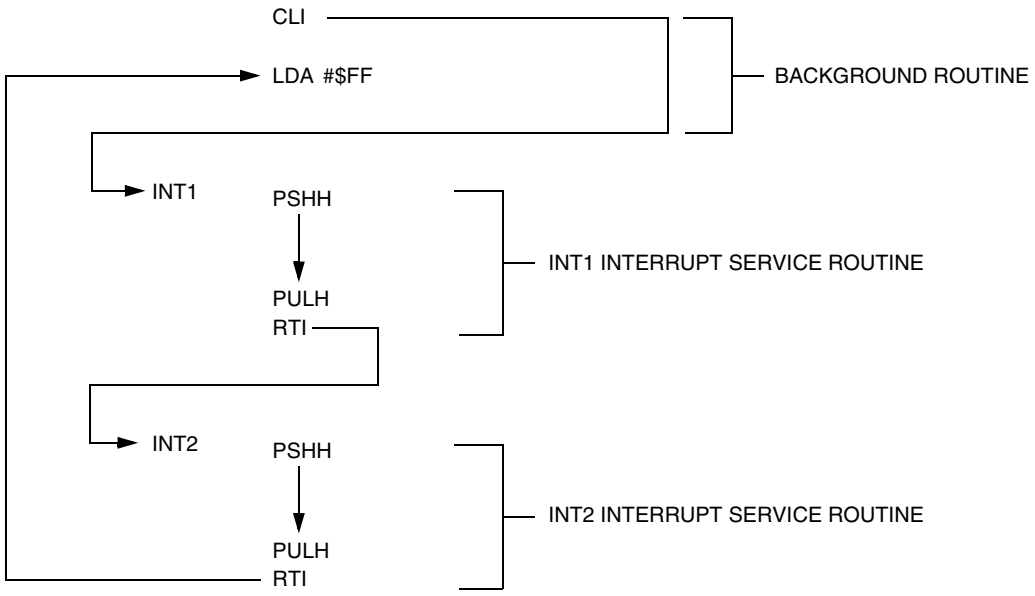


Figure 15-11. Interrupt Recognition Example

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.

15.4.1.2 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.*

15.4.2 Reset

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

15.5 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. See 19.2 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.

15.5.1 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.

15.6 Low-Power Modes

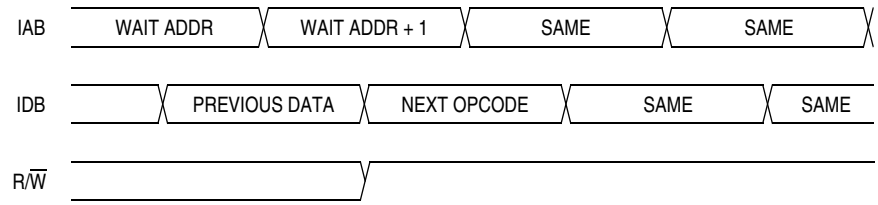
Executing the STOP/WAIT instruction puts the MCU in a low power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts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

15.6.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. Figure 15-12 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.

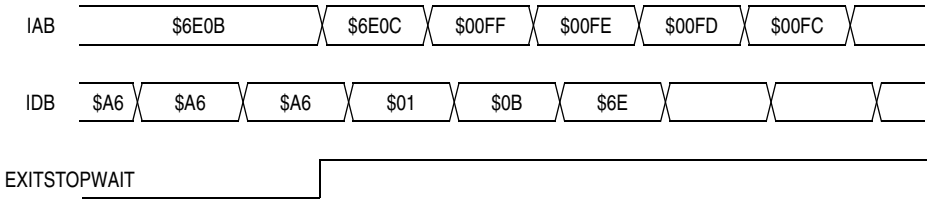
Wait mode can also be exited by a reset or break. A break interrupt during wait mode sets the SIM break WAIT bit, BW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the mask option register is ‘0’, then the computer operating properly (COP) module is enabled and remains active in wait mode.



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

Figure 15-12. WAIT Mode Entry Timing

Figure 15-13 and Figure 15-14 show the timing for wait recovery.



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

Figure 15-13. Wait Recovery from Interrupt

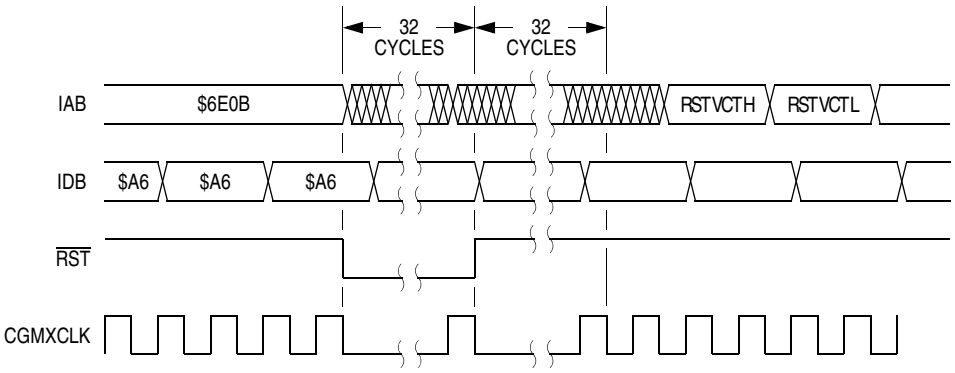


Figure 15-14. Wait Recovery from Internal Reset

15.6.2 Stop Mode

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

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

NOTE

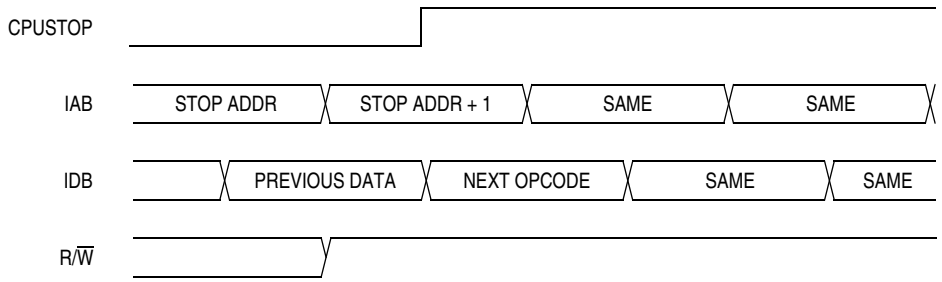
External crystal applications should use the full stop recovery time by clearing the SSREC bit.

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

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 15-15 shows stop mode entry timing.

NOTE

To minimize stop current, all pins configured as inputs should be driven to a logic 1 or logic 0.



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

Figure 15-15. Stop Mode Entry Timing

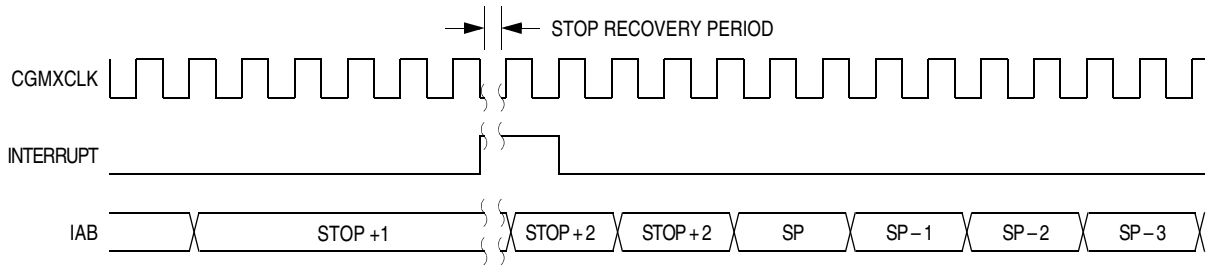


Figure 15-16. Stop Mode Recovery from Interrupt

15.7 SIM Registers

The SIM has three memory mapped registers. Table 15-3 shows the mapping of these registers.

Table 15-3. SIM Registers

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

15.7.1 SIM Break Status Register (SBSR)

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

Address: $FE00

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

R = Reserved

1. Writing a 0 clears BW.

Figure 15-17. SIM Break Status Register (SBSR)

BW — SIM Break Wait Bit

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

1 = Wait mode was exited by break interrupt
0 = Wait mode was not exited by break interrupt

15.7.2 SIM Reset Status Register (SRSR)

The SRSR register contains flags that show the source of the last reset. The status register will automatically clear after reading SRSR. A power-on reset sets the POR bit and clears all other bits in the register. All other reset sources set the individual flag bits but do not clear the register. More than one reset source can be flagged at any time depending on the conditions at the time of the internal or external reset. For example, the POR and LVI bit can both be set if the power supply has a slow rise time.

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 15-18. 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 ($\overline{\text{RST}}$)
0 = POR or read of SRSR

COP — Computer Operating Properly Reset Bit

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

ILOP — Illegal Opcode Reset Bit

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

ILAD — Illegal Address Reset Bit (opcode fetches only)

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

LVI — Low-Voltage Inhibit Reset Bit

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

15.7.3 SIM Break Flag Control Register (SBFCR)

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

Address: $FE03

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

R = Reserved

Figure 15-19. 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

Chapter 16
Serial Peripheral Interface (SPI)

16.1 Introduction

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

16.2 Features

Features include:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four master mode frequencies (maximum = bus frequency ÷ 2)
- Maximum slave mode frequency = bus frequency
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts with CPU service:
  - SPRF (SPI receiver full)
  - SPTE (SPI transmitter empty)
- Mode fault error flag with CPU interrupt capability
- Overflow error flag with CPU interrupt capability
- Programmable wired-OR mode
- I^2C (inter-integrated circuit) compatibility

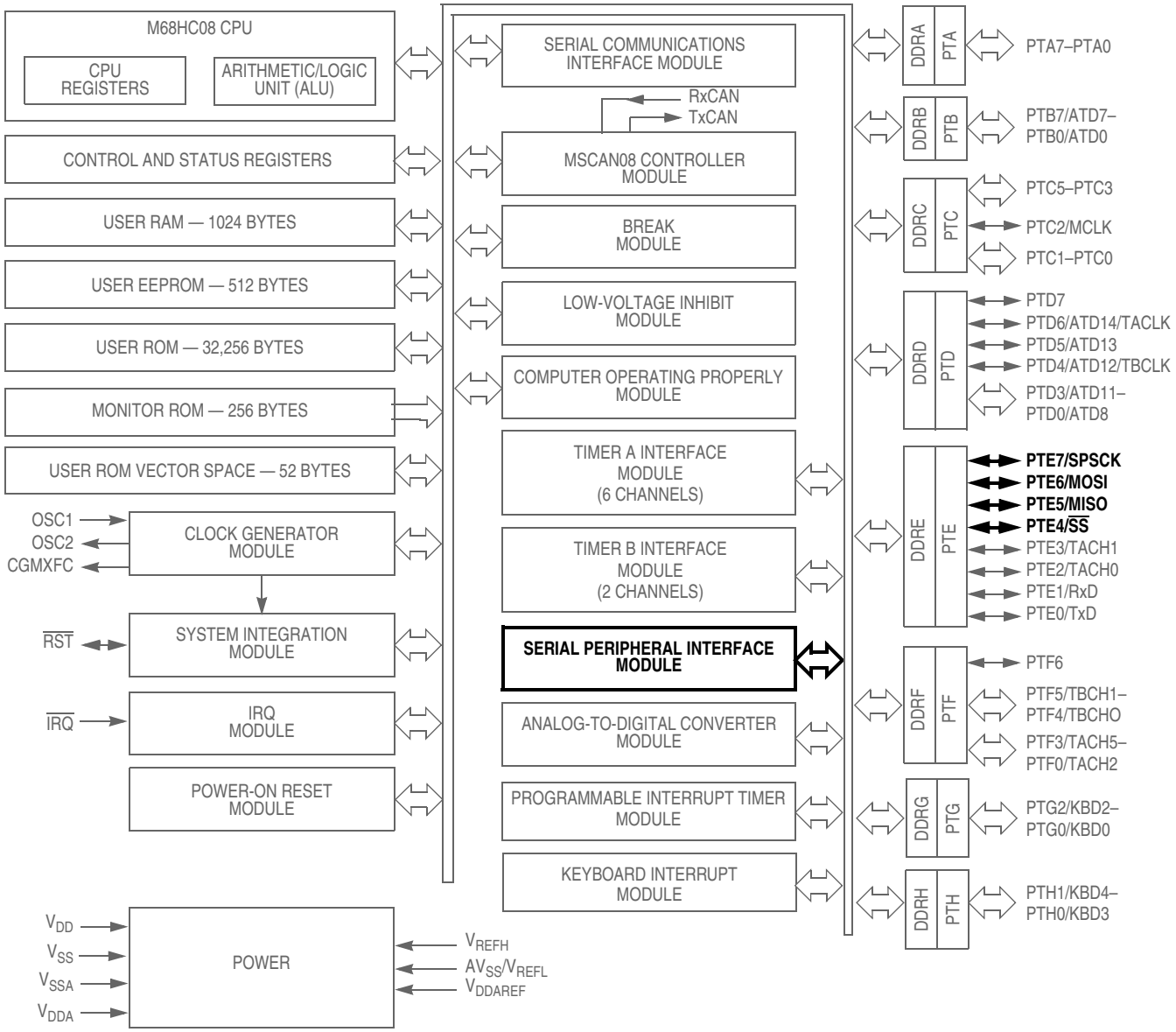


Figure 16-1. Block Diagram Highlighting SPI Block and Pins

16.3 Pin Name Conventions and I/O Register Addresses

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

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

The SPI shares four I/O pins with a parallel I/O port. The full name of an SPI pin reflects the name of the shared port pin. Table 16-1 shows the full names of the SPI I/O pins. The generic pin names appear in the text that follows.

Table 16-1. Pin Name Conventions

SPI Generic Pin Names:	MISO	MOSI	$\overline{SS}$	SCK
Full SPI Pin Names:	PTE5/MISO	PTE6/MOSI	PTE4/$\overline{SS}$	PTE7/SPSCK

16.4 Functional Description

Figure 16-2 summarizes the SPI I/O registers and Figure 16-3 shows the structure of the SPI module.

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

The following paragraphs describe the operation of the SPI module.

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0010	SPI Control Register (SPCR) See page 230.	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) See page 231.	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	1	0	0	0
$0012	SPI Data Register (SPDR) See page 233.	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							

R = Reserved = Unimplemented

Figure 16-2. SPI I/O Register Summary

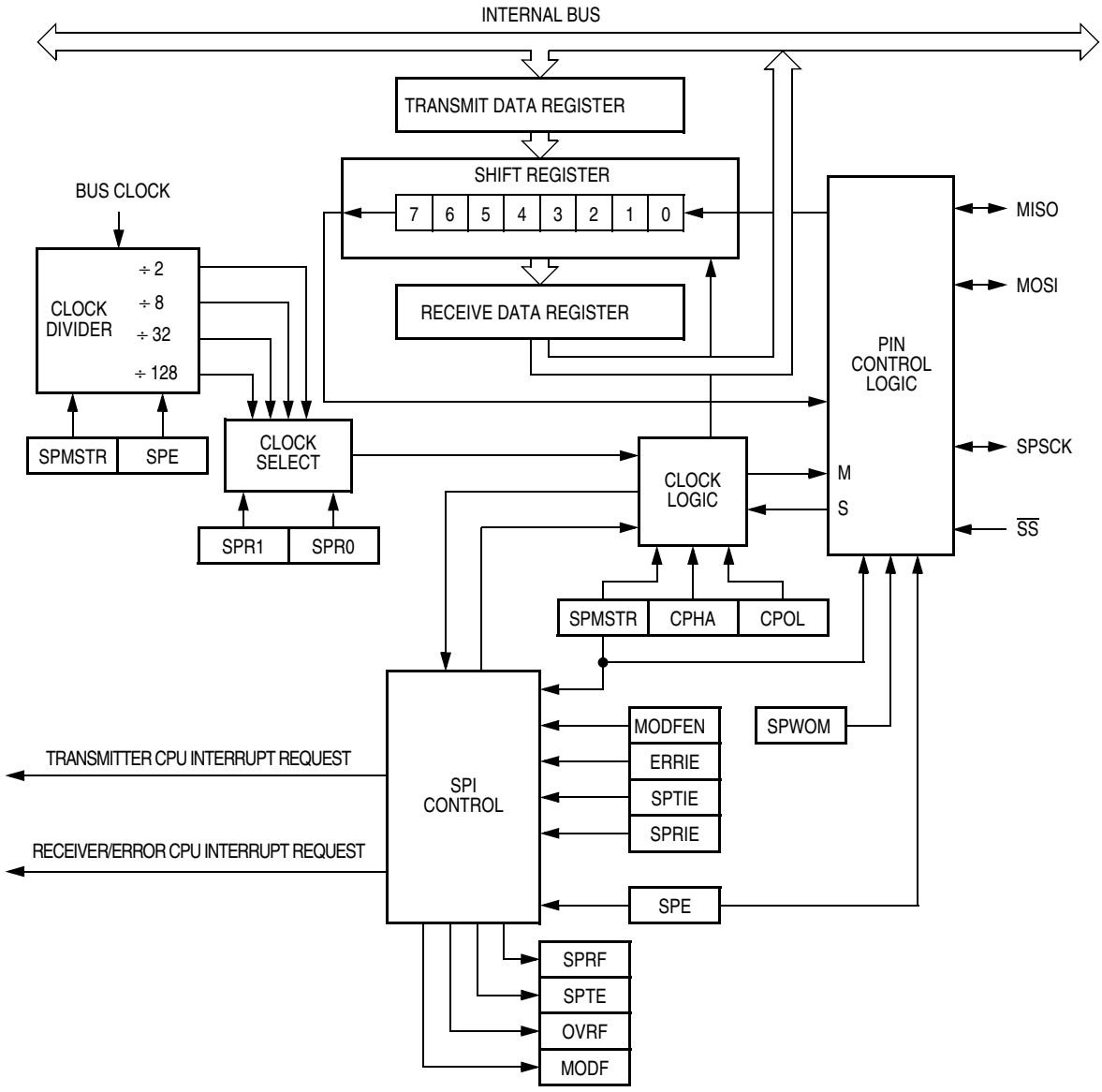


Figure 16-3. SPI Module Block Diagram

16.4.1 Master Mode

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

NOTE

Configure the SPI modules as master and slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. See 16.12.1 SPI Control Register (SPCR).

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the SPI data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The byte begins shifting out on the MOSI pin under the control of the serial clock. See Figure 16-4.

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. See 16.12.2 SPI Status and Control Register (SPSCR). Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

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

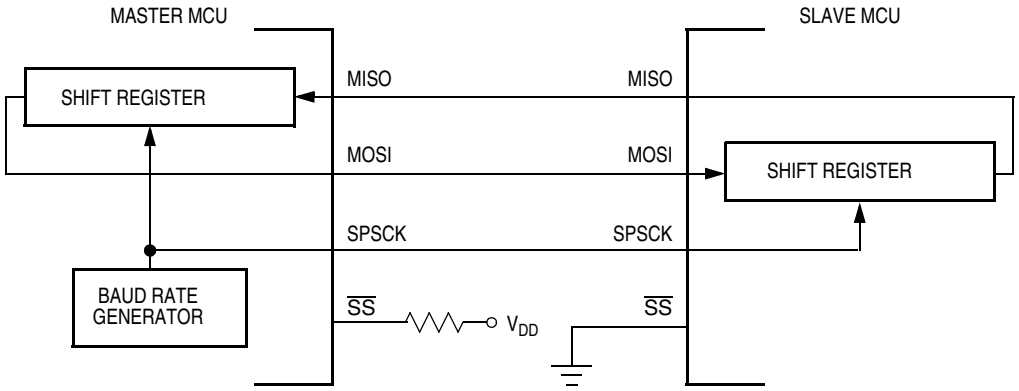


Figure 16-4. Full-Duplex Master Slave Connections

16.4.2 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 MCU must be at low. $\overline{SS}$ must remain low until the transmission is complete. See 16.5.7 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 must then read the SPI data register before another byte enters the shift register.

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

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

When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transmission. See 16.5 Transmission Formats.

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

NOTE

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

16.5 Transmission Formats

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

16.5.1 Clock Phase and Polarity Controls

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

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

NOTE

Before writing to the CPOL bit or the CPHA bit, the SPI should be disabled by clearing the SPI enable bit (SPE).

16.5.2 Transmission Format When CPHA = 0

Figure 16-5 shows an SPI transmission in which CPHA is 0. The figure should not be used as a replacement for data sheet parametric information.Two waveforms are shown for SCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is low, 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 16.5.7 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 transmission. The $\overline{SS}$ pin must be toggled high and then low again between each byte transmitted.

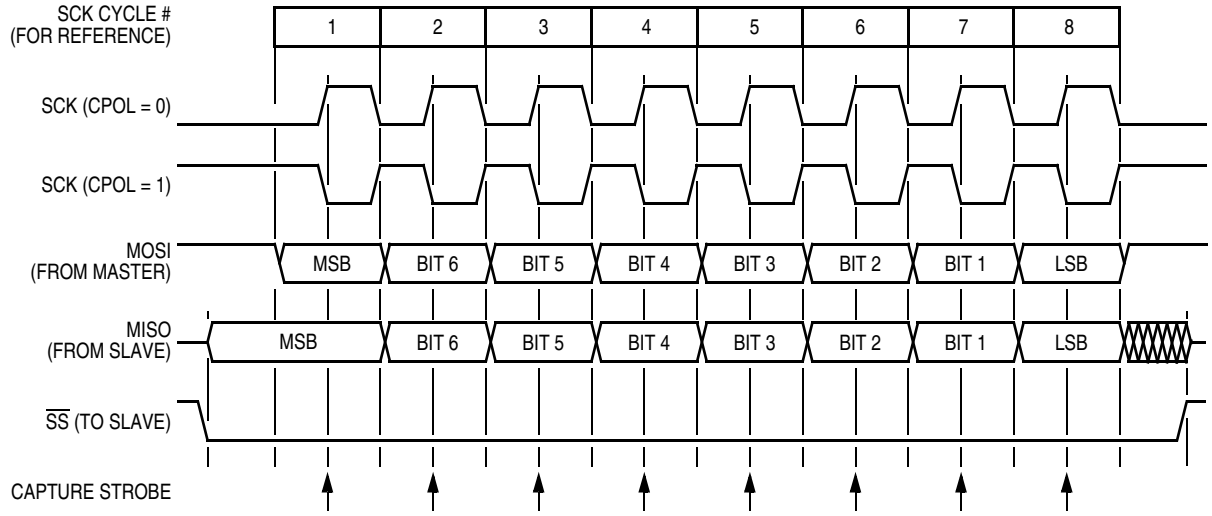


Figure 16-5. Transmission Format (CPHA = 0)

16.5.3 Transmission Format When CPHA = 1

Figure 16-6 shows an SPI transmission in which CPHA is 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is low, 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 16.5.7 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.

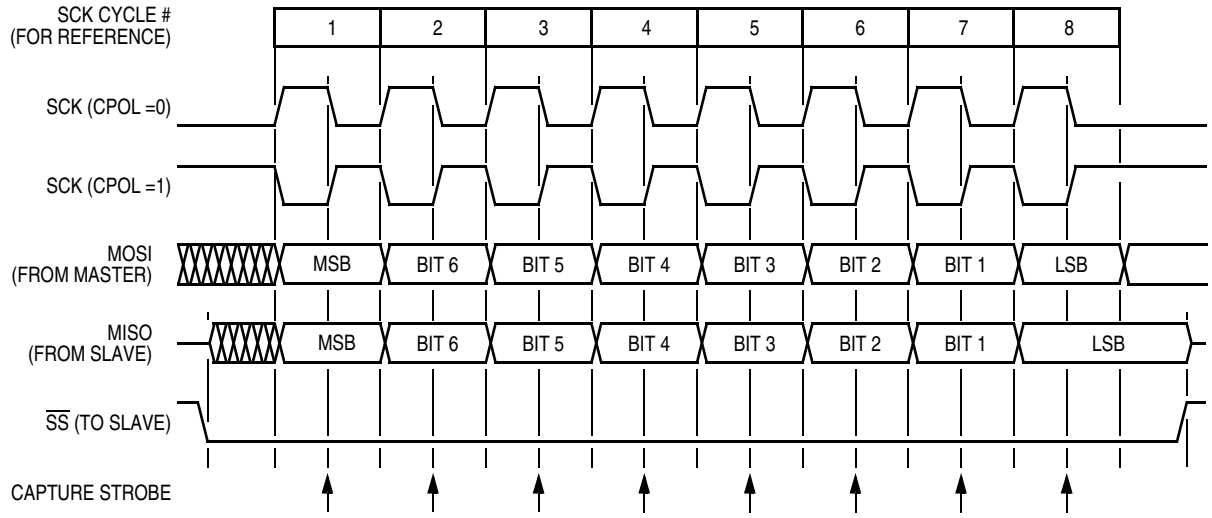


Figure 16-6. Transmission Format (CPHA = 1)

16.5.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), transmissions are started by a software write to the SPDR. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SCK signal. When CPHA = 0, the SCK signal remains inactive for the first half of the first SCK cycle. When CPHA = 1, the first SCK cycle begins with an edge on the SCK line from its inactive to its active level. The SPI clock rate (selected by SPR1:SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. See Figure 16-7. The internal SPI clock in the master is a free-running derivative of the internal MCU clock. It is only enabled when both the SPE and SPMSTR bits are set to conserve power. SCK edges occur halfway through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR will occur relative to the slower SCK. This uncertainty causes the variation in the initiation delay shown in Figure 16-7. This delay will be no longer than a single SPI bit time. That is, the maximum delay is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

16.5.5 Error Conditions

The following flags signal SPI error conditions:

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

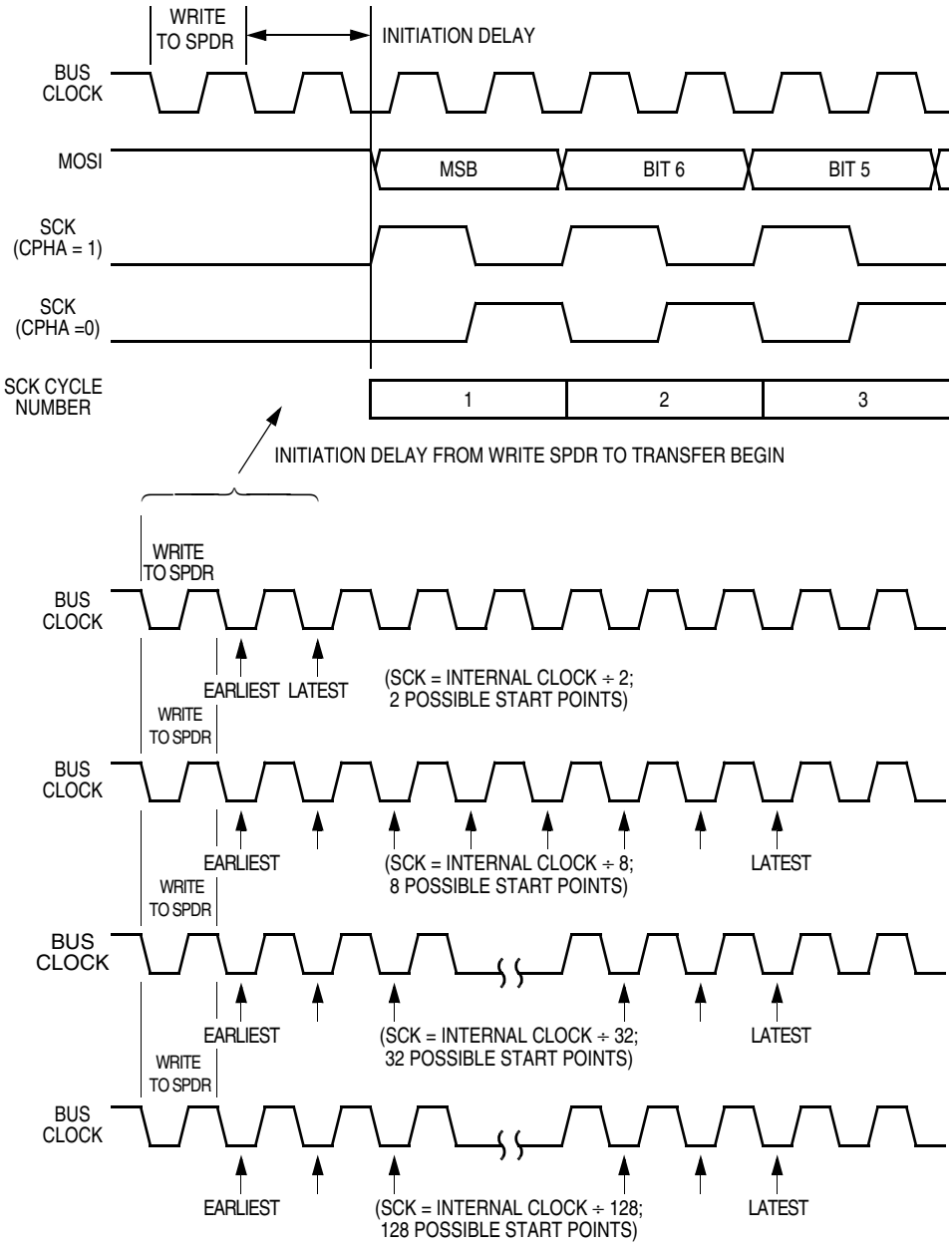


Figure 16-7. Transmission Start Delay (Master)

16.5.6 Overflow Error

The overflow flag (OVRF) becomes set if the SPI receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. See Figure 16-5 and Figure 16-6. If an overflow occurs, the data being received is not transferred to the receive data register so that the unread data can still be read. Therefore, an overflow error always indicates the loss of data.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. MODF and OVRF can generate a receiver/error CPU interrupt request. See Figure 16-10. It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

If an end-of-block transmission interrupt was meant to pull the MCU out of wait, having an overflow condition without overflow interrupts enabled causes the MCU to hang in wait mode. If the OVRF is enabled to generate an interrupt, it can pull the MCU out of wait mode instead.

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. Figure 16-8 shows how it is possible to miss an overflow.

The first part of Figure 16-8 shows how to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF flag can be set in the interval between SPSCR and SPDR being read.

In this case, an overflow can easily be missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it will not be obvious that bytes are being lost as more transmissions are completed. To prevent this, the OVRF interrupt should be enabled, or alternatively another read of the SPSCR should be carried out following the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions will terminate with an SPRF interrupt. Figure 16-9 illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit.

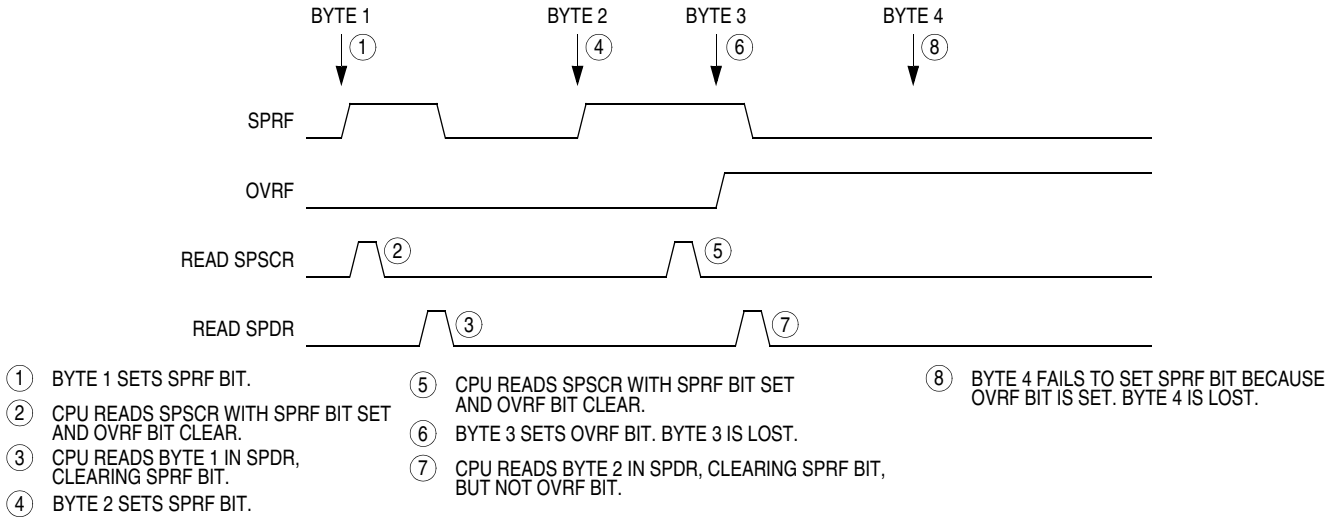


Figure 16-8. Missed Read of Overflow Condition

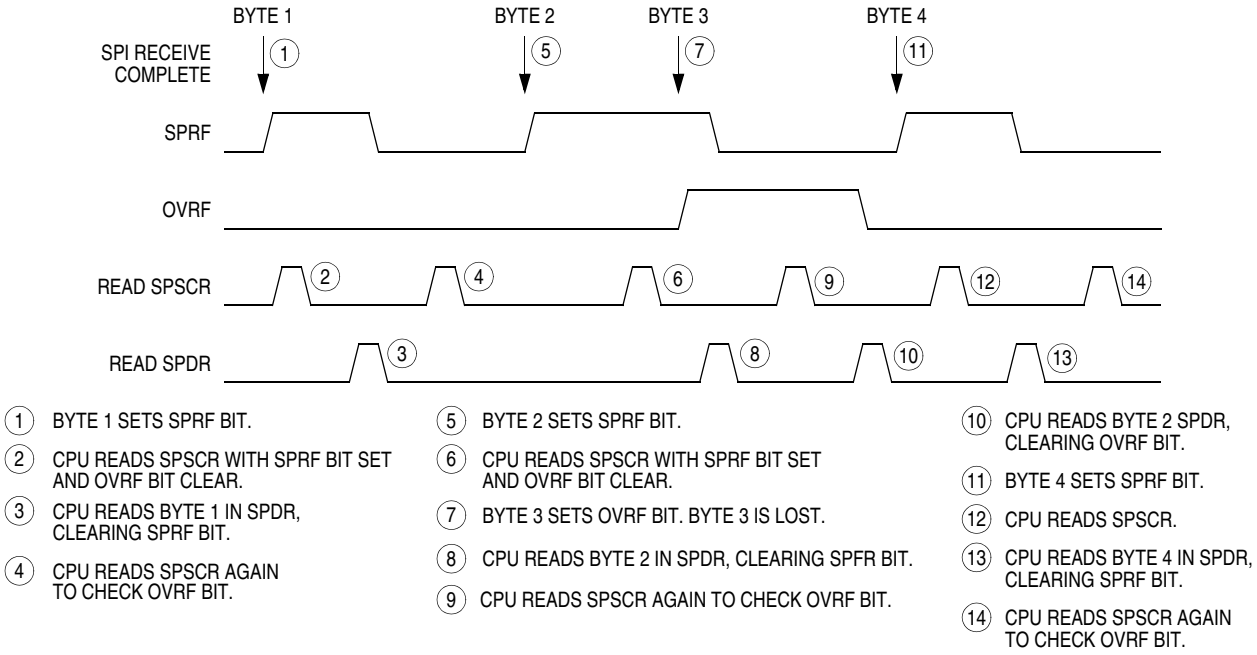


Figure 16-9. Clearing SPRF When OVRF Interrupt is Not Enabled

16.5.7 Mode Fault Error

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

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. MODF and OVRF can generate a receiver/error CPU interrupt request. See Figure 16-10. It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes low. A mode fault in a master SPI causes the following events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error CPU interrupt request.
- The SPE bit is cleared.
- The SPTE bit is set.
- The SPI state counter is cleared.
- The data direction register of the shared I/O port regains control of port drivers.

NOTE

To prevent bus contention with another master SPI after a mode fault error, clear all data direction register (DDR) bits associated with the SPI shared port pins.

Setting the MODF flag (SPSCR) does not clear the SPMSTR bit. Reading SPMSTR when MODF = 1 will indicate a MODE fault error occurred in either master mode or slave mode.

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{SS}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{SS}$ goes low and ends once the incoming SPSCK 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 16.5 Transmission Formats.

NOTE

When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is low) and later deselected ($\overline{SS}$ is high) even if no SPSCK is sent to that slave. This happens because $\overline{SS}$ at 0 indicates the start of the transmission (MISO driven out with the value of MSB) for CPHA = 0. When CPHA = 1, a slave can be selected and then later deselected with no transmission occurring. Therefore, MODF does not occur since a transmission was never begun.

In a slave SPI (MSTR = 0), the MODF bit generates an SPI receiver/error CPU interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transmission by toggling the SPE bit of the slave.

NOTE

A high on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if a transmission has begun.

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

16.6 Interrupts

Four SPI status flags can be enabled to generate CPU interrupt requests. See Table 16-2 and Figure 16-10.

Table 16-2. SPI Interrupts

Flag	Request
SPTE (transmitter empty)	SPI transmitter CPU interrupt request (SPTIE = 1)
SPRF (receiver full)	SPI receiver CPU interrupt request (SPRIE = 1)
OVRF (overflow)	SPI receiver/error interrupt request (SPRIE = 1, ERRIE = 1)
MODF (mode fault)	SPI receiver/error interrupt request (SPRIE = 1, ERRIE = 1, MODFEN = 1)

The SPI transmitter interrupt enable bit (SPTIE) enables the SPTE flag to generate transmitter CPU interrupt requests.

The SPI receiver interrupt enable bit (SPRIE) enables the SPRF bit to generate receiver CPU interrupt requests, provided that the SPI is enabled (SPE = 1).

The error interrupt enable bit (ERRIE) enables both the MODF and OVRF flags to generate a receiver/error CPU interrupt request.

The mode fault enable bit (MODFEN) can prevent the MODF flag from being set so that only the OVRF flag is enabled to generate receiver/error CPU interrupt requests.

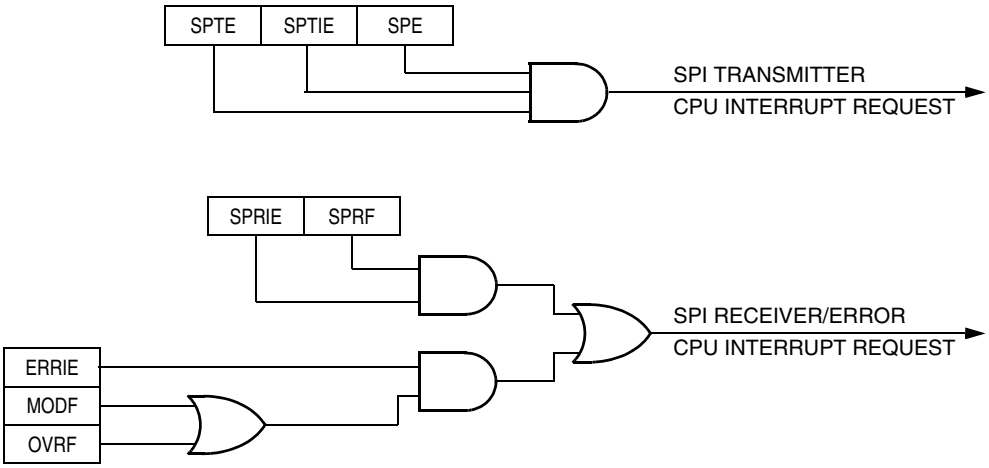


Figure 16-10. SPI Interrupt Request Generation

Two 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 can generate an SPI receiver/error CPU interrupt request.
- SPI transmitter empty (SPTE) — the SPTE bit becomes set every time a byte transfers from the transmit data register to the shift register. If the SPI transmit interrupt enable bit, SPTIE, is also set, SPTE can generate an SPTE CPU interrupt request.

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 SPI data register only when the SPTE bit is high. Figure 16-11 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA: CPOL = 1:0).

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

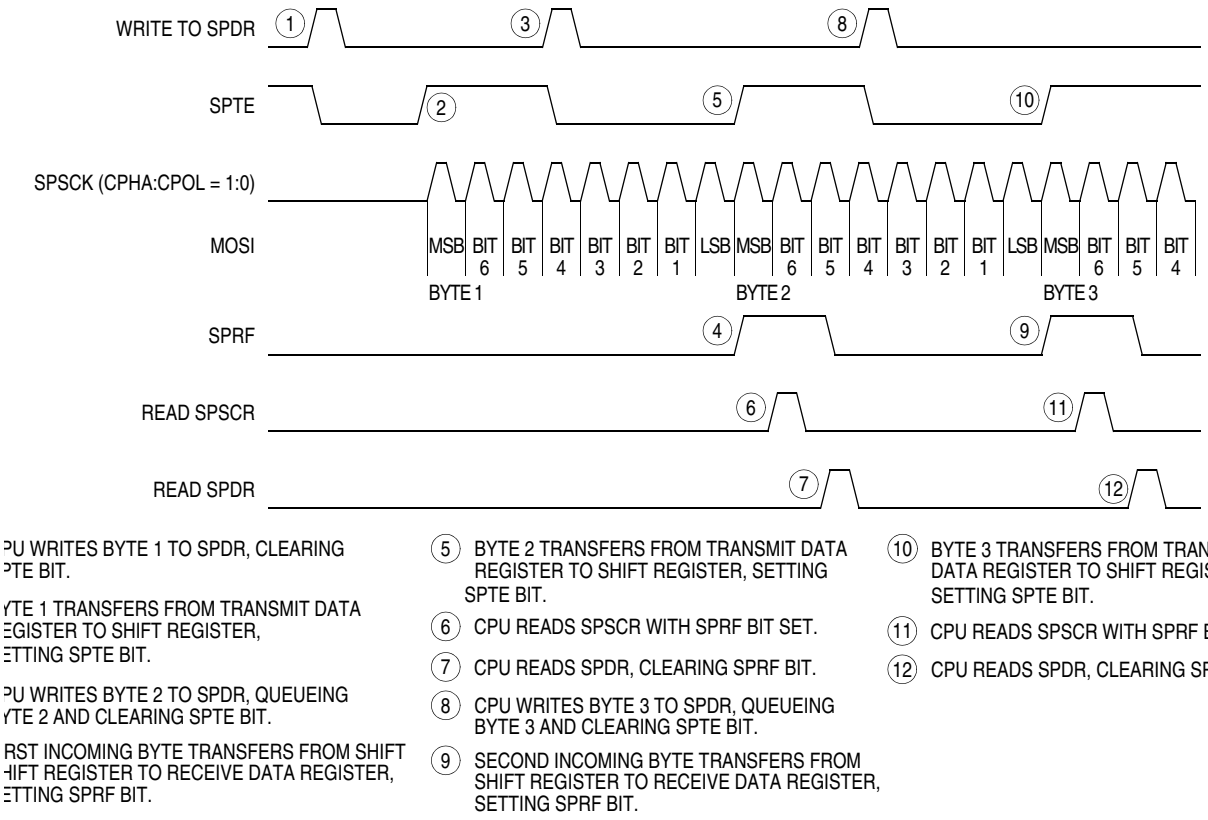


Figure 16-11. SPRF/SPTE CPU Interrupt Timing

16.8 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is low. Whenever SPE is low, the following occurs:

- The SPTE flag is set
- Any transmission currently in progress is aborted
- The shift register is cleared
- The SPI state counter is cleared, making it ready for a new complete transmission
- All the SPI port logic is defaulted back to being general purpose I/O.

The following items are reset only by a system reset:

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

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

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

16.9 Low-Power Modes

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

16.9.1 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, power consumption can be reduced by disabling the SPI module before executing the WAIT instruction.

To exit WAIT mode when an overflow condition occurs, the OVRF bit should be enabled to generate CPU interrupt requests by setting the error interrupt enable bit (ERRIE). See 16.6 Interrupts.

16.9.2 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.

16.10 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 15.7.3 SIM Break Flag Control Register (SBFCR).

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

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

16.11 I/O Signals

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

- MISO — data received
- MOSI — data transmitted
- SPSCK — serial clock
- $\overline{SS}$ — slave select
- V_{SS} — clock ground

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

16.11.1 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 0 and its $\overline{SS}$ pin is at 1. To support a multiple-slave system, a 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.

16.11.2 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.

16.11.3 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.

16.11.4 $\overline{SS}$ (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 16.5 Transmission Formats. Since it is used to indicate the start of a transmission, the $\overline{SS}$ must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low throughout the transmission for the CPHA = 1 format. See Figure 16-12.

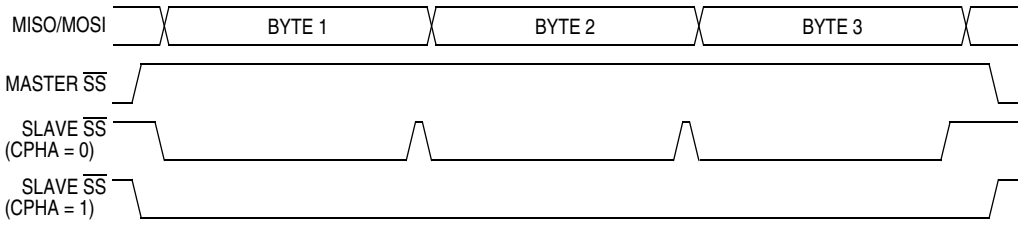


Figure 16-12. CPHA/$\overline{SS}$ Timing

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 16.12.2 SPI Status and Control Register (SPSCR).

NOTE

A 1 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 transmission has already begun.

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 16.5.7 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 data register. See Table 16-3.

Table 16-3. SPI Configuration

SPE	SPMSTR	MODFEN	SPI CONFIGURATION	STATE OF $\overline{SS}$ LOGIC
0	X	X	Not Enabled	General-purpose I/O; $\overline{SS}$ ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	General-purpose I/O; $\overline{SS}$ ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

X = don't care

16.11.5 V_{SS} (Clock Ground)

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

16.12 I/O Registers

Three registers control and monitor SPI operation:

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

16.12.1 SPI Control Register (SPCR)

The SPI control register does the following:

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

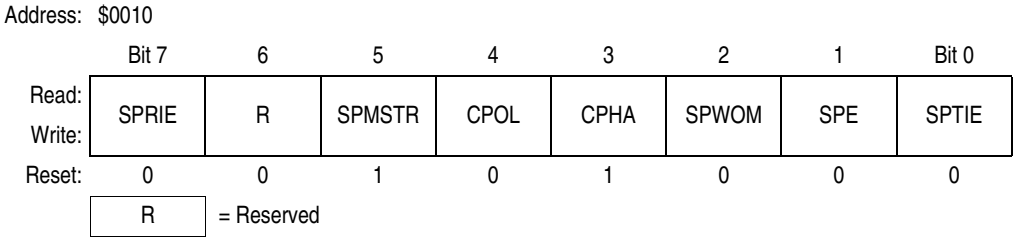


Figure 16-13. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable

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

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

This read/write bit determines the logic state of the SPSCK pin between transmissions. See Figure 16-5 and Figure 16-6. To transmit data between SPI modules, the SPI modules must have identical CPOL bits. Reset clears the CPOL bit.

CPHA — Clock Phase

This read/write bit controls the timing relationship between the serial clock and SPI data. See Figure 16-5 and Figure 16-6. To transmit data between SPI modules, the SPI modules must have identical CPHA bits. When CPHA = 0, the $\overline{SS}$ pin of the slave SPI module must be set to logic one between bytes. See Figure 16-12. Reset sets the CPHA bit.

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

When CPHA = 1 for a slave, the first edge of the SPSCK indicates the beginning of the transmission. The same applies when $\overline{SS}$ is high for a slave. The MISO pin is held in a high-impedance state, and the incoming SPSCK is ignored. In certain cases, it may also cause the MODF flag to be set. See 16.5.7 Mode Fault Error. A 1 on the $\overline{SS}$ pin does not affect the state of the SPI state machine in any way.

SPWOM — SPI Wired-OR Mode

This read/write bit disables the pull-up 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 16.8 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

16.12.2 SPI Status and Control Register (SPSCR)

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

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

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

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

Address: $0011

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
Write:								
Reset:	0	0	0	0	1	0	0	0

= Unimplemented

Figure 16-14. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full

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

During an SPRF CPU interrupt, the CPU clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register. Any read of the SPI data register clears the SPRF bit.

Reset clears the SPRF bit.

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

ERRIE — Error Interrupt Enable

This read-only bit enables the MODF and OVRF flags to generate CPU interrupt requests. Reset clears the ERRIE bit.

1 = MODF and OVRF can generate CPU interrupt requests
0 = MODF and OVRF cannot generate CPU interrupt requests

OVRF — Overflow Flag

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

1 = Overflow
0 = No overflow

MODF — Mode Fault

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

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

SPTE — SPI Transmitter Empty

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

The SPI data register should not be written to unless the SPTE bit is high.

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

Reset sets the SPTE bit.

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

MODFEN — Mode Fault Enable

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 16.11.4 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 16.5.7 Mode Fault Error.

SPR1 and SPR0 — SPI Baud Rate Select

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:SPR0	Baud Rate Divisor (BD)
0 0	2
0 1	8
1 0	32
1 1	128

The following formula is used 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

16.12.3 SPI Data Register (SPDR)

The SPI data register is the read/write buffer for the receive data register and the transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate buffers that can contain different values. See Figure 16-3.

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:	Indeterminate after 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 buffer read is not the same as the buffer written.

Chapter 17
Timer Interface Module A (TIMA)

17.1 Introduction

This section describes the timer interface module (TIMA). The TIMA is a 6-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. Figure 17-2 is a block diagram of the TIMA.

For further information regarding timers on M68HC08 family devices, please consult the *HC08 Timer Reference Manual*, Freescale order number TIM08RM/AD.

17.2 Features

Features of the TIMA include:

- Six 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 (4 MHz maximum frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMA counter stop and reset bits

17.3 Functional Description

Figure 17-2 shows the TIMA structure. The central component of the TIMA is the 16-bit TIMA counter that can operate as a free-running counter or a modulo up-counter. The TIMA counter provides the timing reference for the input capture and output compare functions. The TIMA counter modulo registers, TAMODH–TAMODL, control the modulo value of the TIMA counter. Software can read the TIMA counter value at any time without affecting the counting sequence.

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

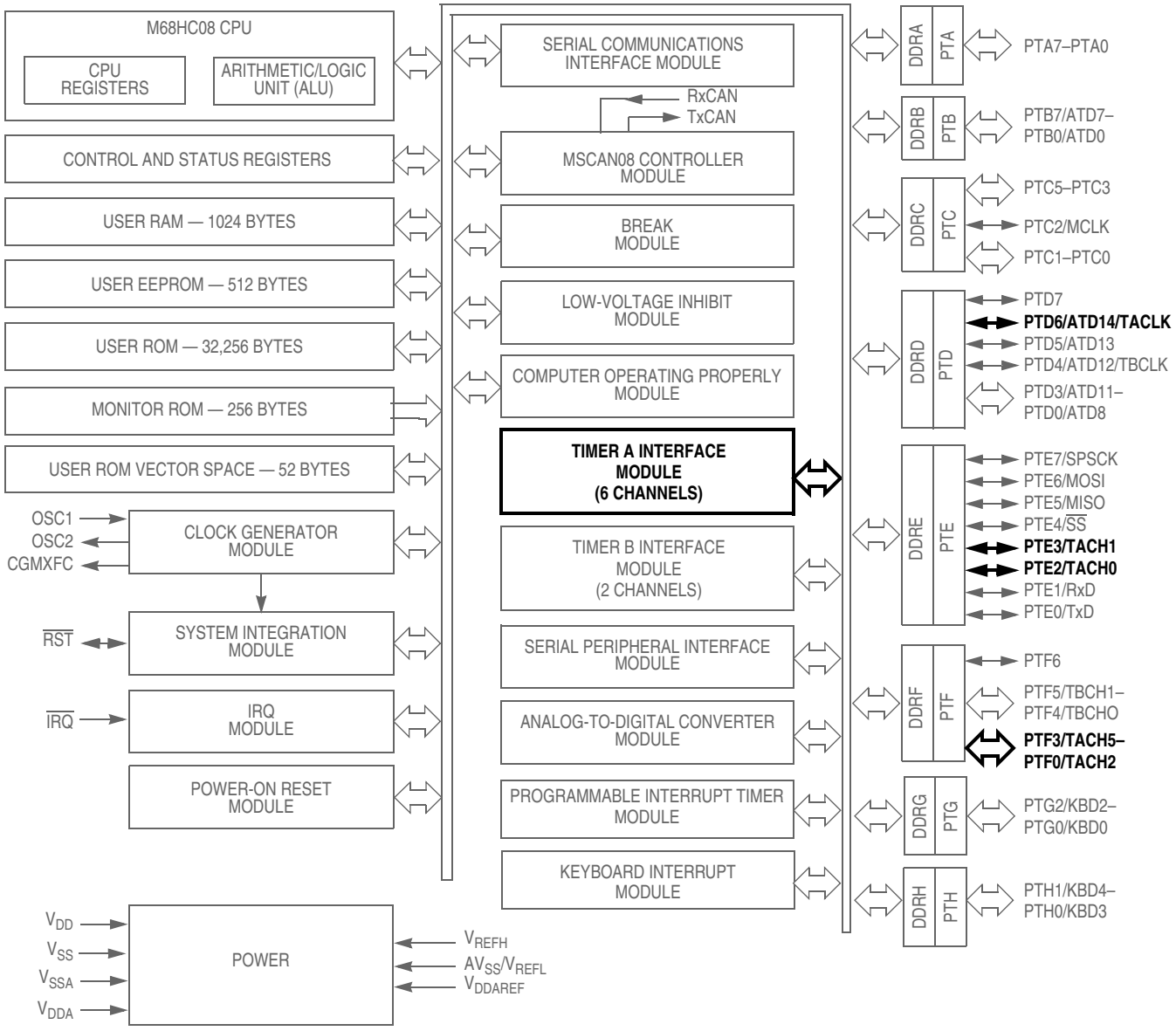


Figure 17-1. Block Diagram Highlighting TIMA Block and Pins

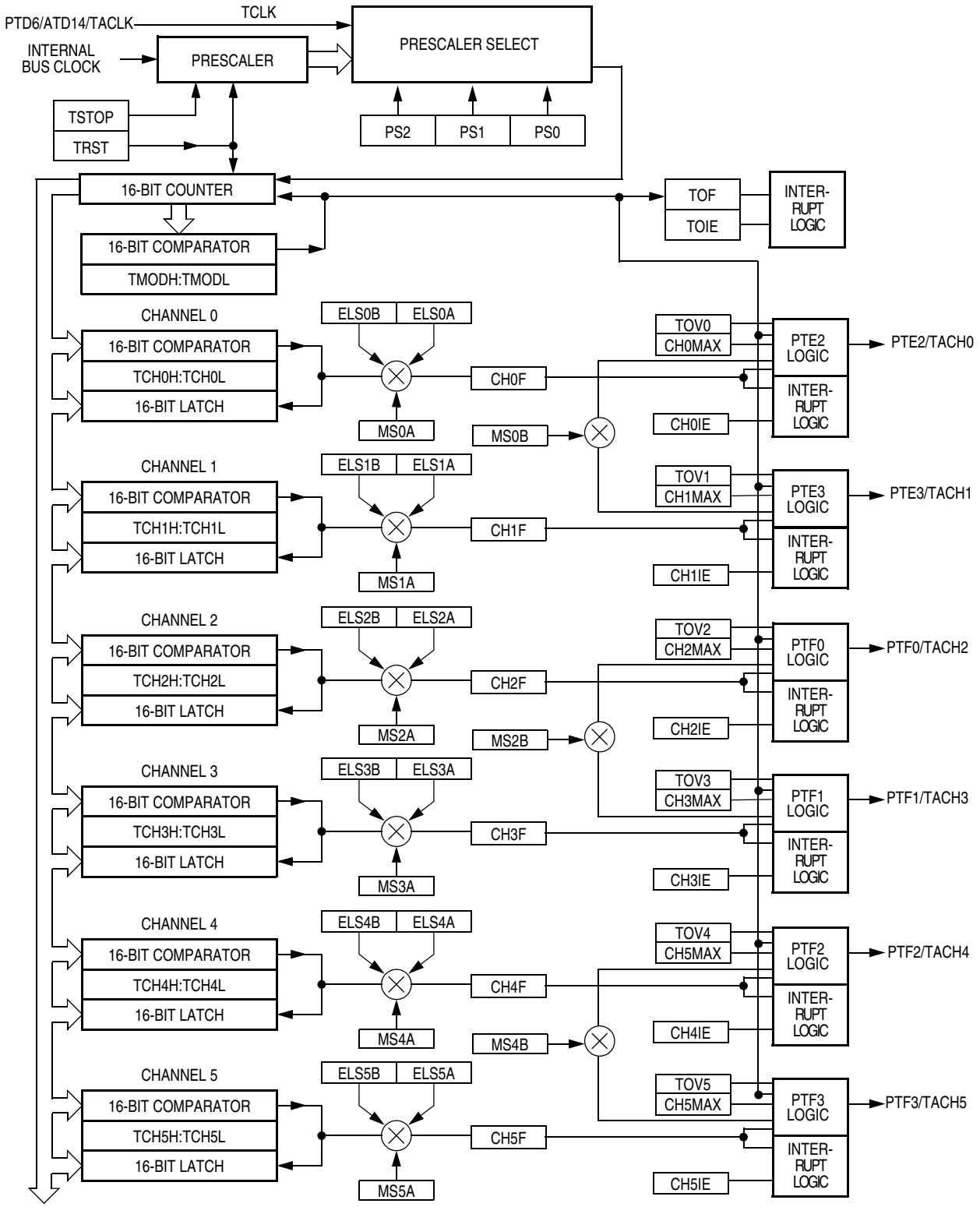


Figure 17-2. TIMA Block Diagram

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0020	Timer A Status and Control Register (TASC) See page 247.	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	R			
		Reset:	0	0	1	0	0	0	0	0
$0022	Timer A Counter Register High (TACNTH) See page 248.	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) See page 248.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0024	Timer A Modulo Register High (TAMODH) See page 249.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	Timer A Modulo Register Low (TAMODL) See page 249.	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) See page 249.	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) See page 253.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0028	Timer A Channel 0 Register Low (TACH0L) See page 253.	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) See page 249.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$002A	Timer A Channel 1 Register High (TACH1H) See page 253.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002B	Timer A Channel 1 Register Low (TACH1L) See page 253.	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) See page 249.	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) See page 253.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							

= Unimplemented R = Reserved

Figure 17-3. TIMA I/O Register Summary

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$002E	Timer A Channel 2 Register Low (TACH2L) See page 253.	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) See page 249.	Read: Write:	CH3F 0	CH3IE	0 R	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Reset:	0	0	0	0	0	0	0	0
$0030	Timer A Channel 3 Register High (TACH3H) See page 253.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0031	Timer A Channel 3 Register Low (TACH3L) See page 253.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0032	Timer A Channel 4 Status and Control Register (TASC4) See page 249.	Read: Write:	CH4F 0	CH4IE	MS4B	MS4A	ELS4B	ELS4A	TOV4	CH4MAX
		Reset:	0	0	0	0	0	0	0	0
$0033	Timer A Channel 4 Register High (TACH4H) See page 253.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0034	Timer A Channel 4 Register Low (TACH4L) See page 253.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0035	Timer A Channel 5 Status and Control Register (TASC5) See page 249.	Read: Write:	CH5F 0	CH5IE	0 R	MS5A	ELS5B	ELS5A	TOV5	CH5MAX
		Reset:	0	0	0	0	0	0	0	0
$0036	Timer A Channel 5 Register High (TACH5H) See page 253.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0037	Timer A Channel 5 Register Low (TACH5L) See page 253.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							

= Unimplemented R = Reserved

Figure 17-3. TIMA I/O Register Summary (Continued)

17.3.1 TIMA Counter Prescaler

The TIMA clock source can be one of the seven prescaler outputs or the TIMA clock pin, PTD6/ATD14/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.

17.3.2 Input Capture

An input capture function has three basic parts: edge select logic, an input capture latch and a 16-bit counter. Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined

transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TASC0 through TASC5 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMA latches the contents of the TIMA counter into the TIMA channel registers, TACHxH–TACHxL. Input captures can generate TIMA CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The free-running counter contents are transferred to the TIMA channel register (TACHxH–TACHxL see 17.8.5 TIMA Channel Registers) on each proper signal transition regardless of whether the TIMA channel flag (CH0F–CH5F in TASC0–TASC5 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register 2 bus cycles after the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see 17.8.5 TIMA Channel Registers). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the TIMA channel register (TACHxH–TACHxL).

17.3.3 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.

17.3.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in 17.3.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the 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 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.

17.3.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the 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.

Channels 4 and 5 can be linked to form a buffered output compare channel whose output appears on the PTF2 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS4B bit in TIMA channel 4 status and control register (TASC4) links channel 4 and channel 5. The output compare value in the TIMA channel 4 registers initially controls the output on the PTF2 pin. Writing to the TIMA channel 5 registers enables the TIMA channel 5 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (4 or 5) that control the output are the ones written to last. TASC4 controls and monitors the buffered output compare function and TIMA channel 5 status and control register (TASC5) is unused. While the MS4B bit is set, the channel 5 pin, PTF3, is available as a general-purpose I/O pin.

NOTE

In buffered output compare operation, do not write new output compare values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered output compares.

17.3.4 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 17-4 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 1. Program the TIMA to set the pin if the state of the PWM pulse is 0.

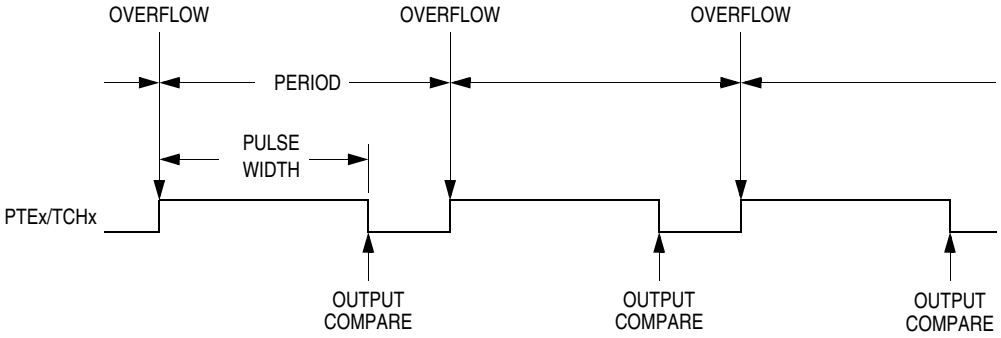


Figure 17-4. PWM Period and Pulse Width

The value in the TIMA counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIMA counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000 (see 17.8.1 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%.

17.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in 17.3.4 Pulse Width Modulation (PWM). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIMA channel registers.

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

Use 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 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.

17.3.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the 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.

Channels 4 and 5 can be linked to form a buffered PWM channel whose output appears on the PTF2 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS4B bit in TIMA channel 4 status and control register (TASC4) links channel 4 and channel 5. The TIMA channel 4 registers initially control the pulse width on the PTF2 pin. Writing to the TIMA channel 5 registers enables the TIMA channel 5 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (4 or 5) that control the pulse width are the ones written to last. TASC4 controls and monitors the buffered PWM function and TIMA channel 5 status and control register (TASC5) is unused. While the MS4B bit is set, the channel 5 pin, PTF3, is available as a general-purpose I/O pin.

NOTE

In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. User software should track the

currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.

17.3.4.3 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 and prescaler by setting the TIMA stop bit, TSTOP.
   b. Reset the TIMA counter and prescaler by setting the TIMA reset bit, TRST.
2. In the TIMA counter modulo registers (TAMODH–TAMODL) write the value for the required PWM period.
3. In the TIMA channel x registers (TACHxH–TACHxL) write the value for the required pulse width.
4. In TIMA channel x status and control register (TASCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA (see Table 17-2).
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB–ELSxA. The output action on compare must force the output to the complement of the pulse width level (see Table 17-2).

   NOTE

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

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMA channel 0 registers (TACH0H–TACH0L) initially control the buffered PWM output. TIMA status control register 0 (TASC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Setting MS2B links channels 2 and 3 and configures them for buffered PWM operation. The TIMA channel 2 registers (TACH2H–TACH2L) initially control the buffered PWM output. TIMA status control register 2 (TASC2) controls and monitors the PWM signal from the linked channels. MS2B takes priority over MS2A.

Setting MS4B links channels 4 and 5 and configures them for buffered PWM operation. The TIMA channel 4 registers (TACH4H–TACH4L) initially control the buffered PWM output. TIMA status control register 4 (TASC4) controls and monitors the PWM signal from the linked channels. MS4B takes priority over MS4A.

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

Setting the channel x maximum duty cycle bit (CHxMAX) and setting the TOVx bit generates a 100% duty cycle output (see 17.8.4 TIMA Channel Status and Control Registers).

17.4 Interrupts

The following TIMA sources can generate interrupt requests:

- TIMA overflow flag (TOF) — The TOF bit is set when the TIMA counter reaches the modulo value programmed in the TIMA counter modulo registers. The TIMA overflow interrupt enable bit, TOIE, enables TIMA overflow CPU interrupt requests. TOF and TOIE are in the TIMA status and control register.
- TIMA channel flags (CH5F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMA CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

17.5 Low-Power Modes

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

17.5.1 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.

17.5.2 Stop Mode

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

17.6 TIMA During Break Interrupts

A break interrupt stops the TIMA counter and inhibits input captures.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state (see 15.7.3 SIM Break Flag Control Register (SBFCR)).

To allow software to clear status bits during a break interrupt, write a 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 0 to the BCFE bit. With BCFE at 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 0. After the break, doing the second step clears the status bit.

17.7 I/O Signals

Port D shares one of its pins with the TIMA. Port E shares two of its pins with the TIMA and port F shares four of its pins with the TIMA. PTD6/ATD14/TACLK is an external clock input to the TIMA prescaler. The six TIMA channel I/O pins are PTE2/TACH0, PTE3/TACH1, PTF0/TACH2, PTF1/TACH3, PTF2, and PTF3.

17.7.1 TIMA Clock Pin (PTD6/ATD14/TACLK)

PTD6/ATD14/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/ATD14/TACLK input by writing 1s to the three prescaler select bits, PS[2:0] (see 17.8.1 TIMA Status and Control Register). The minimum TCLK pulse width, $TCLK_{LMIN}$ or $TCLK_{HMIN}$, is:

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

The maximum TCLK frequency is the least: 4 MHz or bus frequency ÷ 2.

PTD6/ATD14/TACLK is available as a general-purpose I/O pin or ADC channel when not used as the TIMA clock input. When the PTD6/ATD14/TACLK pin is the TIMA clock input, it is an input regardless of the state of the DDRD6 bit in data direction register D.

17.7.2 TIMA Channel I/O Pins (PTF3–PTF0/TACH2 and PTE3/TACH1–PTE2/TACH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTE2/TACH0, PTF0/TACH2 and PTF2 can be configured as buffered output compare or buffered PWM pins.

17.8 I/O Registers

These I/O registers control and monitor TIMA operation:

- TIMA status and control register (TASC)
- TIMA control registers (TACNTH–TACNTL)
- TIMA counter modulo registers (TAMODH–TAMODL)
- TIMA channel status and control registers (TASC0–TASC5)
- TIMA channel registers (TACH0H–TACH0L through TACH5H–TACH5L)

17.8.1 TIMA Status and Control Register

The TIMA status and control register:

- 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	R			
Reset:	0	0	1	0	0	0	0	0

R = Reserved

Figure 17-5. TIMA Status and Control Register (TASC)

TOF — TIMA Overflow Flag Bit

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

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

TOIE — TIMA Overflow Interrupt Enable Bit

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

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

TSTOP — TIMA Stop Bit

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

1 = TIMA counter stopped
0 = TIMA counter active

NOTE

Do not set the TSTOP bit before entering wait mode if the TIMA is required to exit wait mode. Also, when the TSTOP bit is set and input capture mode is enabled, input captures are inhibited until TSTOP is cleared.

When using TSTOP to stop the timer counter, see if any timer flags are set. If a timer flag is set, it must be cleared by clearing TSTOP, then clearing the flag, then setting TSTOP again.

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 0. Reset clears the TRST bit.

1 = Prescaler and TIMA counter cleared
0 = No effect

NOTE

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

PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD6/ATD14/TACLK pin or one of the seven prescaler outputs as the input to the TIMA counter as Table 17-1 shows. Reset clears the PS[2:0] bits.

Table 17-1. Prescaler Selection

PS[2:0]	TIMA Clock Source
000	Internal Bus Clock ÷1
001	Internal Bus Clock ÷ 2
010	Internal Bus Clock ÷ 4
011	Internal Bus Clock ÷ 8
100	Internal Bus Clock ÷ 16
101	Internal Bus Clock ÷ 32
110	Internal Bus Clock ÷ 64
111	PTD6/ATD14/TACLK

17.8.2 TIMA Counter Registers

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

NOTE

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

Register Name and Address: TACNTH — $0022

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

Register Name and Address: TACNTL — $0023

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

= Unimplemented

Figure 17-6. TIMA Counter Registers (TACNTH and TACNTL)

17.8.3 TIMA Counter Modulo Registers

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

Register Name and Address TAMODH — $0024

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

Register Name and Address TAMODL — $0025

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

Figure 17-7. TIMA Counter Modulo Registers (TAMODH and TAMODL)

NOTE

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

17.8.4 TIMA Channel Status and Control Registers

Each of the TIMA channel status and control registers:

- Flags input captures and output compares
- Enables input capture and output compare interrupts
- Selects input capture, output compare or PWM operation
- Selects high, low or toggling output on output compare
- Selects rising edge, falling edge or any edge as the active input capture trigger
- Selects output toggling on TIMA overflow
- Selects 0% and 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Register Name and Address TASC0 — $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

Register Name and Address TASC1 — $0029

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

R = Reserved

Figure 17-8. TIMA Channel Status and Control Registers (TASC0–TASC5) (Sheet 1 of 2)

Register Name and Address: TASC2 — $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

Register Name and Address: TASC3 — $002F

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

Register Name and Address: TASC4 — $0032

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	TOV4	CH4MAX
Write:	0							
Reset:	0	0	0	0	0	0	0	0

Register Name and Address: TASC5 — $0035

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CH5F	CH5IE	0	MS5A	ELS5B	ELS5A	TOV5	CH5MAX
Write:	0		R					
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 17-8. TIMA Channel Status and Control Registers (TASC0–TASC5) (Sheet 2 of 2)

CHxF — Channel x Flag Bit

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

When CHxIE = 1, clear CHxF by reading TIMA channel x status and control register with CHxF set and then writing a 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a 1 to CHxF has no effect.

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

CHxIE — Channel x Interrupt Enable Bit

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

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

MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIMA channel 0, TIMA channel 2 and TIMA channel 4 status and control registers.

Setting MS0B disables the channel 1 status and control register and reverts TACH1 pin to general-purpose I/O.

Setting MS2B disables the channel 3 status and control register and reverts TACH3 pin to general-purpose I/O.

Setting MS4B disables the channel 5 status and control register and reverts TACH5 pin to general-purpose I/O.

Reset clears the MSxB bit.

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

MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. See Table 17-2.

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

When ELSxB:A = 00, this read/write bit selects the initial output level of the TACHx pin once PWM, output compare mode or input capture mode is enabled. See Table 17-2. Reset clears the MSxA bit.

1 = Initial output level low
0 = Initial output level high

NOTE

Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIMA status and control register (TASC).

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port E or port F and pin PTEx/TACHx or pin PTFx/TACHx is available as a general-purpose I/O pin. However, channel x is at a state determined by these bits and becomes transparent to the respective pin when PWM, input capture mode or output compare operation mode is enabled. Table 17-2 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

NOTE

Before enabling a TIMA channel register for input capture operation, make sure that the PTEx/TACHx pin or PTFx/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 1 and clear output on compare is selected, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 17-9 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at 100% duty cycle level until the cycle after CHxMAX is cleared.

NOTE

The 100% PWM duty cycle is defined as a continuous high level if the PWM polarity is 1 and a continuous low level if the PWM polarity is 0. Conversely, a 0% PWM duty cycle is defined as a continuous low level if the PWM polarity is 1 and a continuous high level if the PWM polarity is 0.

Table 17-2. 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	0	Output compare or PWM	Software compare only
0	1	0	1		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

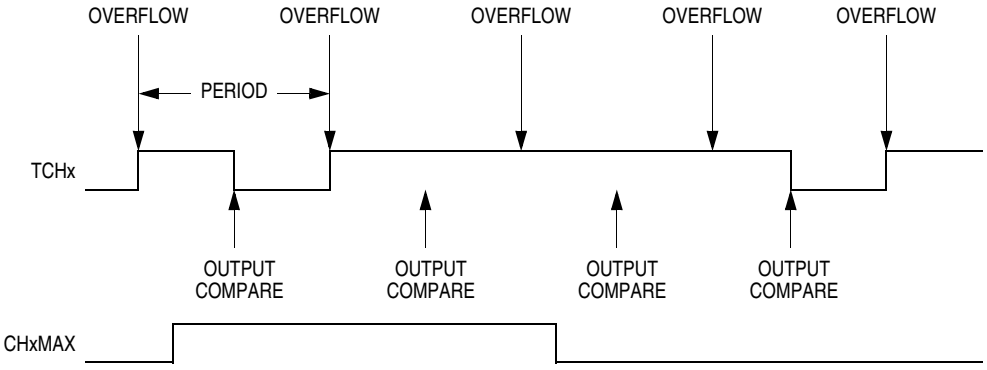


Figure 17-9. CHxMAX Latency

17.8.5 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 and the CHxF bit until the low byte (TACHxL) is written.

Register Name and Address: TACH0H — $0027

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

Register Name and Address: TACH0L — $0028

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

Register Name and Address: TACH1H — $002A

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

Register Name and Address: TACH1L — $002B

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

Register Name and Address: TACH2H — $002D

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

Figure 17-10. TIMA Channel Registers (TACH0H/L–TACH5H/L) (Sheet 1 of 2)

Register Name and Address TACH2L — $002E

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

Register Name and Address TACH3H — $0030

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

Register Name and Address TACH3L — $0031

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

Register Name and Address TACH4H — $0033

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

Register Name and Address TACH4L — $0034

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

Register Name and Address TACH5H — $0036

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

Register Name and Address TACH5L — $0037

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

Figure 17-10. TIMA Channel Registers (TACH0H/L–TACH5H/L) (Sheet 2 of 2)

Chapter 18
Timer Interface Module B (TIMB)

18.1 Introduction

This section describes the timer interface module (TIMB). The TIMB is a 2-channel timer that provides a timing reference with input capture, output compare, and pulse width modulation functions. Figure 18-2 is a block diagram of the TIMB.

For further information regarding timers on M68HC08 family devices, please consult the *HC08 Timer Reference Manual*, Freescale document order number TIM08RM/AD.

18.2 Features

Features include:

- Two input capture/output compare channels
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
- Buffered and unbuffered pulse width modulation (PWM) signal generation
- Programmable TIMB clock input
  - Seven frequency internal bus clock prescaler selection
  - External TIMB clock input (4 MHz maximum frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMB counter stop and reset bits

18.3 Functional Description

Figure 18-2 shows the TIMB structure. The central component of the TIMB is the 16-bit TIMB counter that can operate as a free-running counter or a modulo up-counter. The TIMB counter provides the timing reference for the input capture and output compare functions. The TIMB counter modulo registers, TBMODH–TBMODL, control the modulo value of the TIMB counter. Software can read the TIMB counter value at any time without affecting the counting sequence.

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

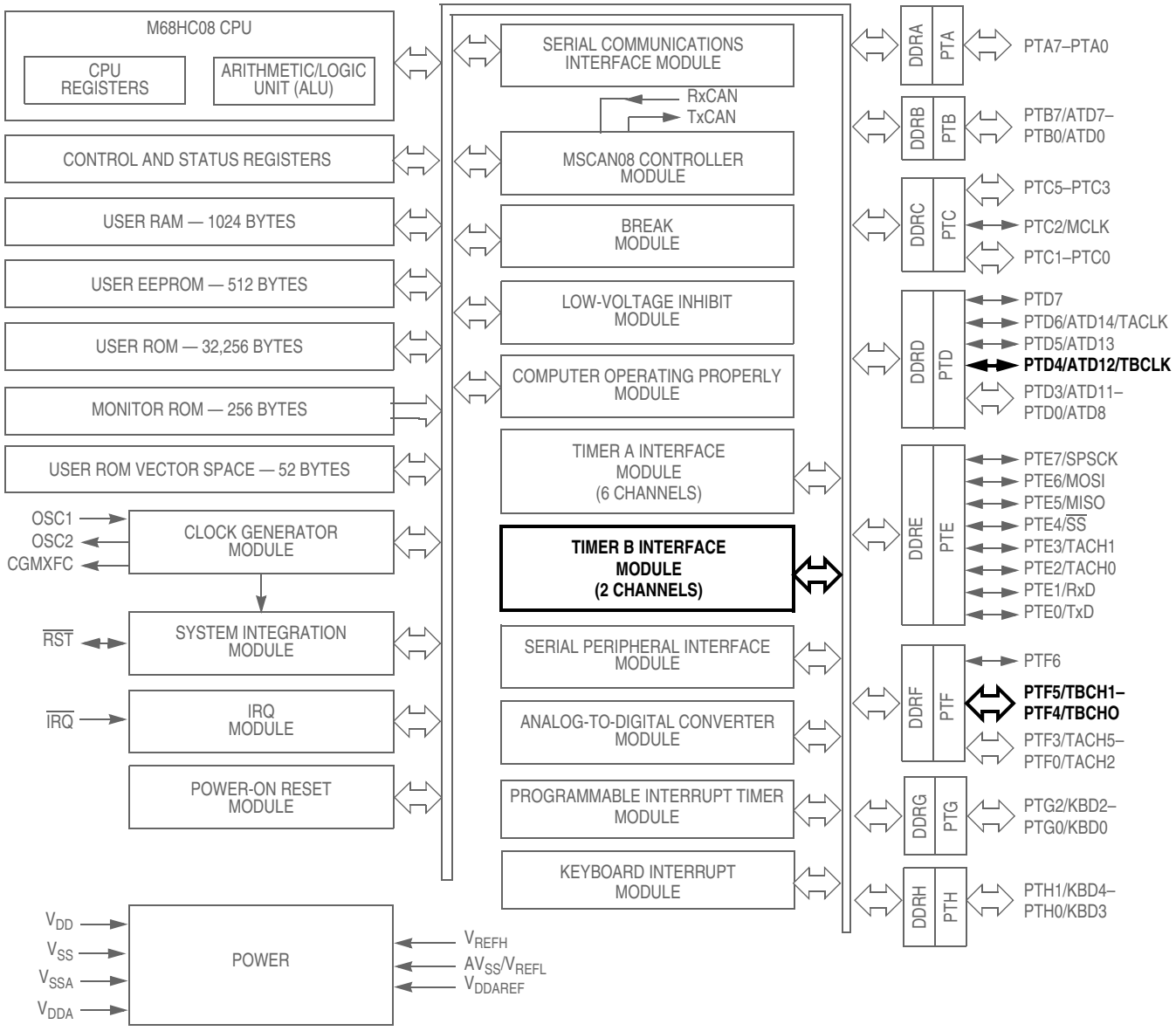


Figure 18-1. Block Diagram Highlighting TIMB Block and Pins

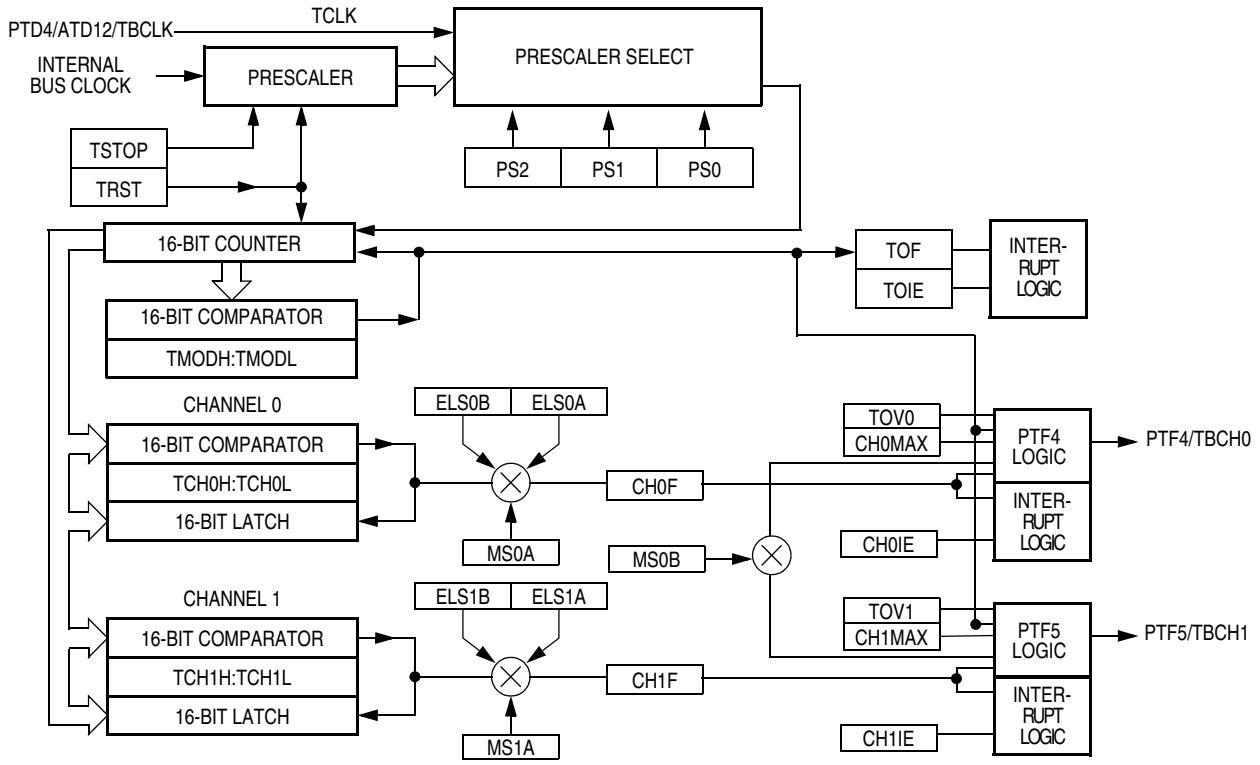


Figure 18-2. TIMB Block Diagram

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0040	Timer B Status and Control Register (TBSC) See page 265.	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	R			
		Reset:	0	0	1	0	0	0	0	0
$0041	Timer B Counter Register High (TBCNTH) See page 266.	Read:	Bit 15	14	13	12	11	10	9	8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0042	Timer B Counter Register Low (TBCNTL) See page 266.	Read:	Bit 7	6	5	4	3	2	1	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0043	Timer B Modulo Register High (TBMODH) See page 267.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0044	Timer B Modulo Register Low (TBMODL) See page 267.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1

= Unimplemented R = Reserved

Figure 18-3. TIMB I/O Register Summary

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0045	Timer B Channel 0 Status and Control Register (TBSC0) See page 268.	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) See page 271.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0047	Timer B Channel 0 Register Low (TBCH0L) See page 271.	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) See page 268.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$0049	Timer B Channel 1 Register High (TBCH1H) See page 271.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$004A	Timer B Channel 1 Register Low (TBCH1L) See page 271.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							

= Unimplemented R = Reserved

Figure 18-3. TIMB I/O Register Summary (Continued)

18.3.1 TIMB Counter Prescaler

The TIMB clock source can be one of the seven prescaler outputs or the TIMB clock pin, PTD4/ATD12/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.

18.3.2 Input Capture

An input capture function has three basic parts: edge select logic, an input capture latch and a 16-bit counter. Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TBSC0 through TBSC1 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMB latches the contents of the TIMB counter into the TIMB channel registers, TBCHxH–TBCHxL. Input captures can generate TIMB CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The free-running counter contents are transferred to the TIMB channel register (TBCHxH–TBCHxL, see 18.8.5 TIMB Channel Registers) on each proper signal transition regardless of whether the TIMB channel flag (CH0F–CH1F in TBSC0–TBSC1 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register 2 bus cycles after the actual event occurs, user software can

respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see 18.8.5 TIMB Channel Registers). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the input capture channel register (TBCHxH–TBCHxL).

18.3.3 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.

18.3.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in 18.3.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the 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 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.

18.3.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the 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.

NOTE

In buffered output compare operation, do not write new output compare values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered output compares.

18.3.4 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 18-4 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 1. Program the TIMB to set the pin if the state of the PWM pulse is 0.

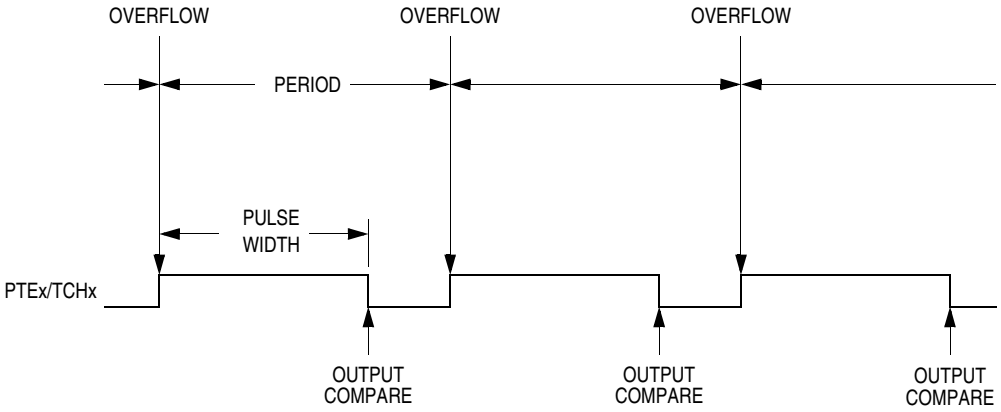


Figure 18-4. PWM Period and Pulse Width

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

18.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in 18.3.4 Pulse Width Modulation (PWM). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIMB channel registers.

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

Use 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 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.

18.3.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the 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.

NOTE

In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. User software should track the

currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.

18.3.4.3 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 and prescaler by setting the TIMB reset bit, TRST.
2. In the TIMB counter modulo registers (TBMODH–TBMODL) write the value for the required PWM period.
3. In the TIMB channel x registers (TBCHxH–TBCHxL) write the value for the required pulse width.
4. In TIMB channel x status and control register (TBSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA (see Table 18-2).
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB–ELSxA. The output action on compare must force the output to the complement of the pulse width level (see Table 18-2).

   NOTE

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

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMB channel 0 registers (TBCH0H–TBCH0L) initially control the buffered PWM output. TIMB status control register 0 (TBSC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

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

Setting the channel x maximum duty cycle bit (CHxMAX) and setting the TOVx bit generates a 100% duty cycle output (see 18.8.4 TIMB Channel Status and Control Registers).

18.4 Interrupts

The following TIMB sources can generate interrupt requests:

- TIMB overflow flag (TOF) — The TOF bit is set when the TIMB counter value reaches 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 (CH1F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMB CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

18.5 Low-Power Modes

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

18.5.1 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.

18.5.2 Stop Mode

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

18.6 TIMB During Break Interrupts

A break interrupt stops the TIMB counter and inhibits input captures.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state (see 15.7.3 SIM Break Flag Control Register (SBFCR)).

To allow software to clear status bits during a break interrupt, write a 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 0 to the BCFE bit. With BCFE at 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 0. After the break, doing the second step clears the status bit.

18.7 I/O Signals

Port D shares one of its pins with the TIMB. Port F shares two of its pins with the TIMB. PTD4/ATD12/TBCLK is an external clock input to the TIMB prescaler. The two TIMB channel I/O pins are PTF4/TBCH0 and PTF5/TBCH1.

18.7.1 TIMB Clock Pin (PTD4/ATD12/TBCLK)

PTD4/ATD12/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/ATD12/TBCLK input by writing 1s to the three prescaler select bits, PS[2:0] (see TIMB Status and Control Register). The minimum TCLK pulse width, $TCLK_{LMIN}$ or $TCLK_{HMIN}$, is:

$$\frac{1}{\text{Bus Frequency}} + t_{SU}$$

The maximum TCLK frequency is the least: 4 MHz or bus frequency ÷ 2.

PTD4/ATD12/TBCLK is available as a general-purpose I/O pin or ADC channel when not used as the TIMB clock input. When the PTD4/ATD12/TBCLK pin is the TIMB clock input, it is an input regardless of the state of the DDRD4 bit in data direction register D.

18.7.2 TIMB Channel I/O Pins (PTF5/TBCH1–PTF4/TBCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTF4/TBCH0 and PTF5/TBCH1 can be configured as buffered output compare or buffered PWM pins.

18.8 I/O Registers

These I/O registers control and monitor TIMB operation:

- TIMB status and control register (TBSC)
- TIMB control registers (TBCNTH–TBCNTL)
- TIMB counter modulo registers (TBMODH–TBMODL)
- TIMB channel status and control registers (TBSC0 and TBSC1)
- TIMB channel registers (TBCH0H–TBCH0L and TBCH1H–TBCH1L)

18.8.1 TIMB Status and Control Register

The TIMB status and control register:

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

Address:	$0040							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
Write:	0			TRST	R			
Reset:	0	0	1	0	0	0	0	0
	R	= Reserved						

Figure 18-5. TIMB Status and Control Register (TBSC)

TOF — TIMB Overflow Flag Bit

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

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

TOIE — TIMB Overflow Interrupt Enable Bit

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

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

TSTOP — TIMB Stop Bit

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

1 = TIMB counter stopped
0 = TIMB counter active

NOTE

Do not set the TSTOP bit before entering wait mode if the TIMB is required to exit wait mode. Also, when the TSTOP bit is set and the timer is configured for input capture operation, input captures are inhibited until TSTOP is cleared.

When using TSTOP to stop the timer counter, see if any timer flags are set. If a timer flag is set, it must be cleared by clearing TSTOP, then clearing the flag, then setting TSTOP again.

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 0. Reset clears the TRST bit.

1 = Prescaler and TIMB counter cleared
0 = No effect

NOTE

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

PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD4/ATD12/TBCLK pin or one of the seven prescaler outputs as the input to the TIMB counter as Table 18-1 shows. Reset clears the PS[2:0] bits.

Table 18-1. Prescaler Selection

PS[2:0]	TIMB Clock Source
000	Internal bus clock ÷1
001	Internal bus clock ÷ 2
010	Internal bus clock ÷ 4
011	Internal bus clock ÷ 8
100	Internal bus clock ÷ 16
101	Internal bus clock ÷ 32
110	Internal bus clock ÷ 64
111	PTD4/ATD12/TBCLK

18.8.2 TIMB Counter Registers

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

NOTE

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

Register Name and Address TBCNTH — $0041

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

Register Name and Address TBCNTL — $0042

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

R = Reserved

Figure 18-6. TIMB Counter Registers (TBCNTH and TBCNTL)

18.8.3 TIMB Counter Modulo Registers

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

Register Name and Address: TBMODH — $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

Register Name and Address: TBMODL — $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 18-7. TIMB Counter Modulo Registers (TBMODH and TBMODL)

NOTE

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

18.8.4 TIMB Channel Status and Control Registers

Each of the TIMB channel status and control registers:

- Flags input captures and output compares
- Enables input capture and output compare interrupts
- Selects input capture, output compare or PWM operation
- Selects high, low or toggling output on output compare
- Selects rising edge, falling edge or any edge as the active input capture trigger
- Selects output toggling on TIMB overflow
- Selects 0% and 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Register Name and Address TBSC0 — $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

Register Name and Address TBSC1 — $0048

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

R = Reserved

Figure 18-8. TIMB Channel Status and Control Registers (TBSC0–TBSC1)

CHxF — Channel x Flag Bit

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

When CHxIE = 1, clear CHxF by reading TIMB channel x status and control register with CHxF set, and then writing a 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a 1 to CHxF has no effect.

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

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMB CPU interrupts on channel x.

Reset clears the CHxIE bit.

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

MSxB — Mode Select Bit B

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

Setting MS0B disables the channel 1 status and control register and reverts TBCH1 to general-purpose I/O.

Reset clears the MSxB bit.

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

MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation (see Table 18-2).

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

When ELSxB:A = 00, this read/write bit selects the initial output level of the TBCHx pin once PWM, input capture or output compare operation is enabled (see Table 18-2). Reset clears the MSxA bit.

1 = Initial output level low
0 = Initial output level high

NOTE

Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIMB status and control register (TBSC).

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port F and pin PTFx/TBCHx is available as a general-purpose I/O pin. However, channel x is at a state determined by these bits and becomes transparent to the respective pin when PWM, input capture, or output compare mode is enabled. Table 18-2 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

NOTE

Before enabling a TIMB channel register for input capture operation, make sure that the PTFx/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.

Table 18-2. 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	0	Output compare or PWM	Software compare only
0	1	0	1		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

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at 1 and clear output on compare is selected, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 18-9 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at 100% duty cycle level until the cycle after CHxMAX is cleared.

NOTE

The 100% PWM duty cycle is defined as a continuous high level if the PWM polarity is 1 and a continuous low level if the PWM polarity is 0. Conversely, a 0% PWM duty cycle is defined as a continuous low level if the PWM polarity is 1 and a continuous high level if the PWM polarity is 0.

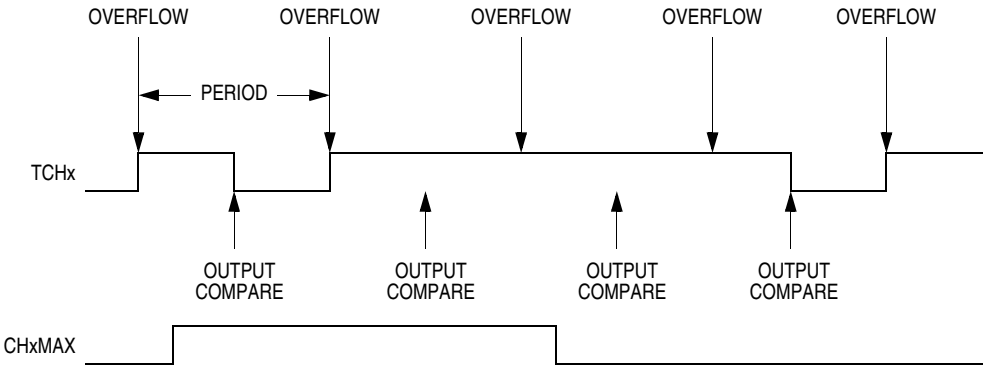


Figure 18-9. CHxMAX Latency

18.8.5 TIMB Channel Registers

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

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

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

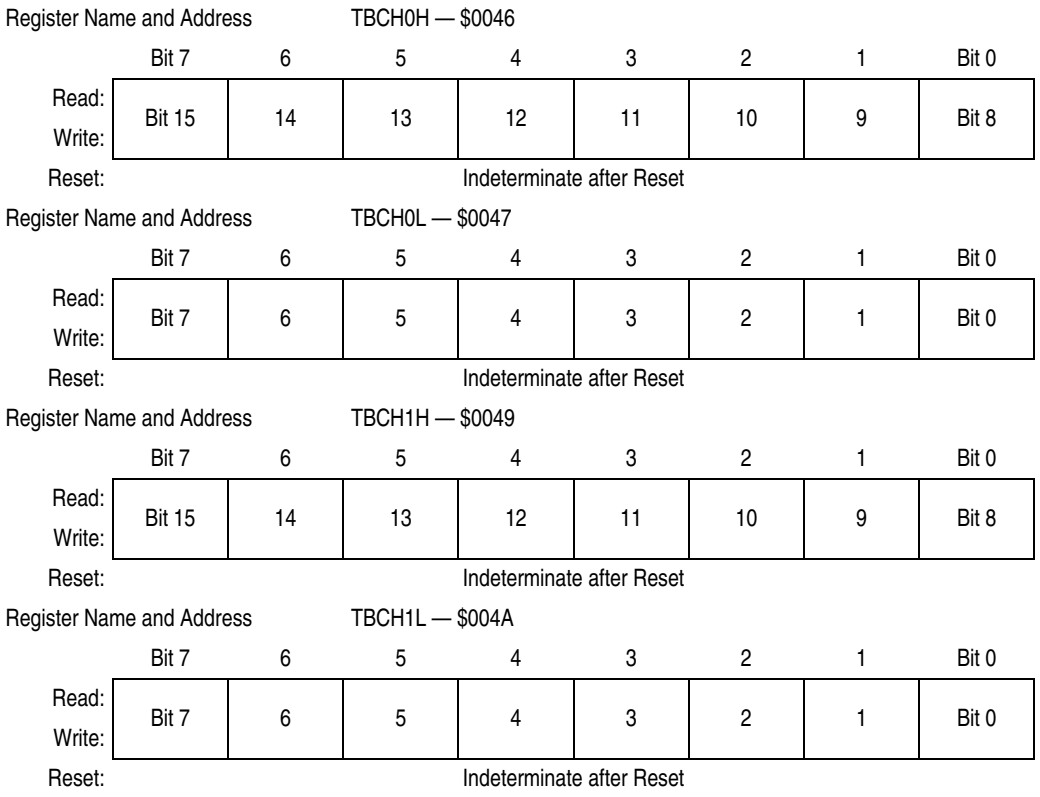


Figure 18-10. TIMB Channel Registers (TBCH0H/L–TBCH1H/L)

Chapter 19
Development Support

19.1 Introduction

This section describes the break module, the monitor read-only memory (MON), and the monitor mode entry methods.

19.2 Break Module (BRK)

The break module can generate a break interrupt which stops normal program flow at a defined address in order to begin execution of a background program.

Features include:

- Accessible I/O registers during the break interrupt
- CPU-generated break Interrupts
- Software-generated break interrupts
- COP disabling during break interrupts

19.2.1 Functional Description

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

The following events can cause a break interrupt to occur:

- A CPU-generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a 1 to the BRKA bit in the break status and control register (BRKSCR).

When a CPU-generated address matches the contents of the break address registers, the break interrupt begins after the CPU completes its current instruction. A return from interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. Figure 19-2 shows the structure of the break module.

When the internal address bus matches the value written in the break address registers or when software writes a 1 to the BRKA bit in the break status and control register, 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)

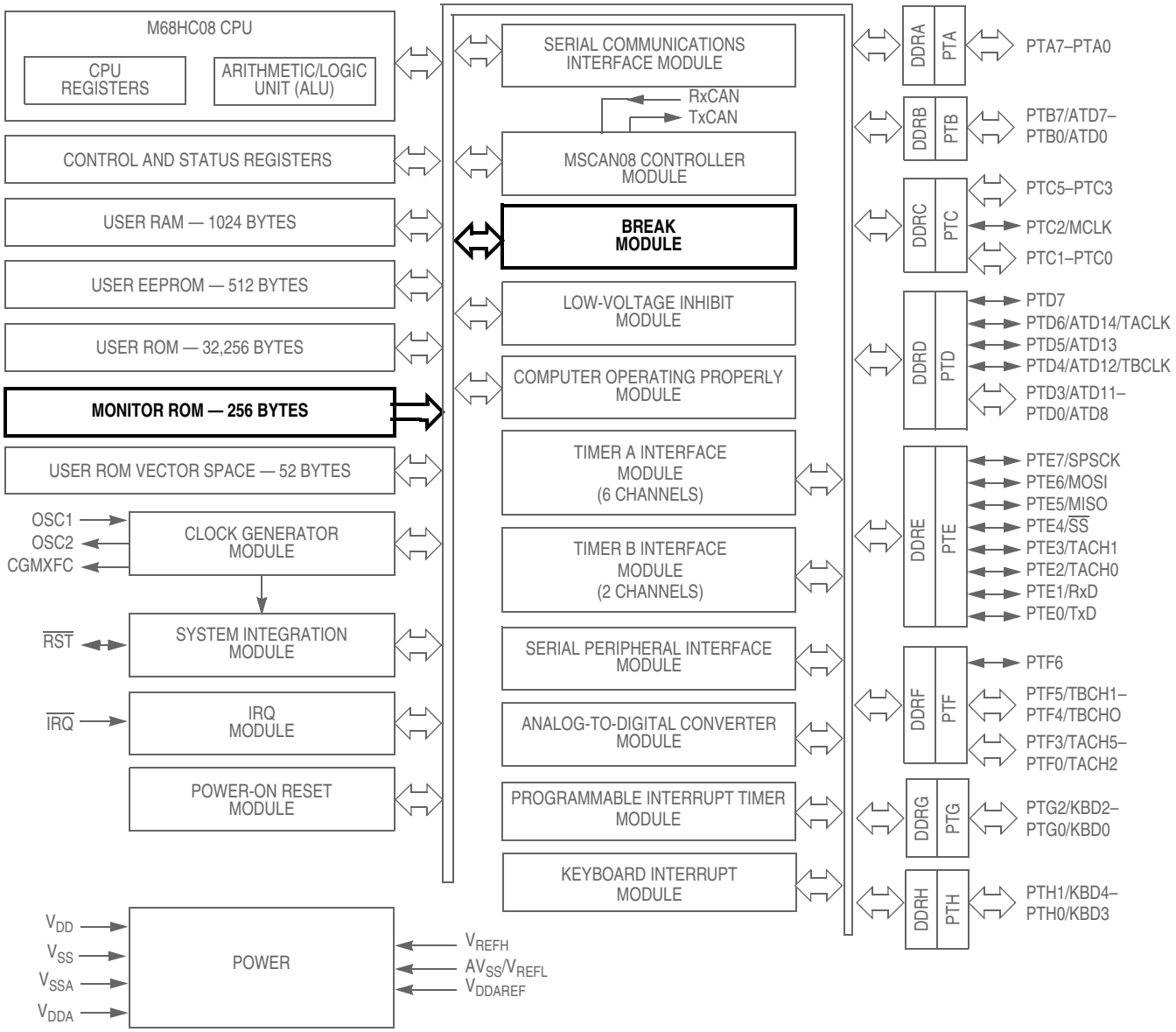


Figure 19-1. Block Diagram Highlighting BRK and MON Blocks

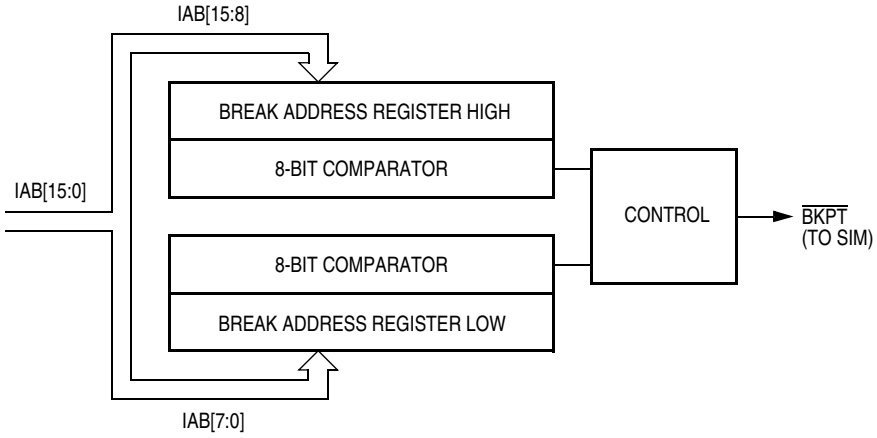


Figure 19-2. Break Module Block Diagram

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE0C	Break Address Register High (BRKH) See page 277.	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) See page 277.	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) See page 276.	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:			R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
			R = Reserved							

Figure 19-3. Break I/O Register Summary

The break interrupt timing is:

- When a break address is placed at the address of the instruction opcode, the instruction is not executed until after completion of the break interrupt routine.
- When a break address is placed at an address of an instruction operand, the instruction is executed before the break interrupt.
- When software writes a 1 to the BRKA bit, the break interrupt occurs just before the next instruction is executed.

By updating a break address and clearing the BRKA bit in a break interrupt routine, a break interrupt can be generated continuously.

CAUTION

A break address should be placed at the address of the instruction opcode. When software does not change the break address and clears the BRKA bit in the first break interrupt routine, the next break interrupt will not be generated after exiting the interrupt routine even when the internal address bus matches the value written in the break address registers.

19.2.1.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether or not module status bits can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. See Chapter 15 System Integration Module (SIM), and the **Break Interrupts** subsection for each module.

19.2.1.2 TIM and PIT During Break Interrupts

A break interrupt stops the timer counter.

19.2.1.3 COP During Break Interrupts

The COP is disabled during a break interrupt when V_{TST} is present on the $\overline{RST}$ pin. For V_{TST} see 20.5 5.0 Volt DC Electrical Characteristics.

19.2.2 Break Module Registers

Three registers control and monitor operation of the break module:

- Break status and control register (BRKSCR)
- Break address register high (BRKH)
- Break address register low (BRKL)

19.2.2.1 Break Status and Control Register (BRKSCR)

The break status and control register 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:			R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 19-4. 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 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 1 to BRKA generates a break interrupt. Clear BRKA by writing a 0 to it before exiting the break routine. Reset clears the BRKA bit.

1 = Break address match
0 = No break address match

19.2.2.2 Break Address Registers (BRKH and BRKL)

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

Address:	$FE0C							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BIT 15	BIT 13	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
Reset:	0	0	0	0	0	0	0	0

Figure 19-5. Break Address Register (BRKH)

Address:	$FE0D							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Reset:	0	0	0	0	0	0	0	0

Figure 19-6. Break Address Register (BRKL)

19.2.3 Low-Power Modes

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

19.2.3.1 Wait Mode

If enabled, the break module is active in wait mode. The SIM break wait bit (BW) in the SIM break status register indicates whether wait was exited by a break interrupt. If so, the user can modify the return address on the stack by subtracting one from it. See Chapter 15 System Integration Module (SIM).

19.2.3.2 Stop Mode

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

19.3 Monitor Module (MON)

This subsection describes the monitor module (MON) and the monitor mode entry methods. The monitor allows debugging and programming of the microcontroller unit (MCU) through a single-wire interface with a host computer.

Features include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between monitor ROM and host computer
- Standard mark/space non-return-to-zero (NRZ) communication with host computer
- Up to 28.8K baud communication with host computer
- Execution of code in RAM or ROM
- EEPROM programming
- ROM security (read protection)[1]
- EEPROM read protection[1]

1. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the ROM/EEPROM data difficult for unauthorized users

19.3.1 Functional Description

Figure 19-7 shows a simplified diagram of monitor mode entry.

The monitor module receives and executes commands from a host computer. Figure 19-8 shows an example circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

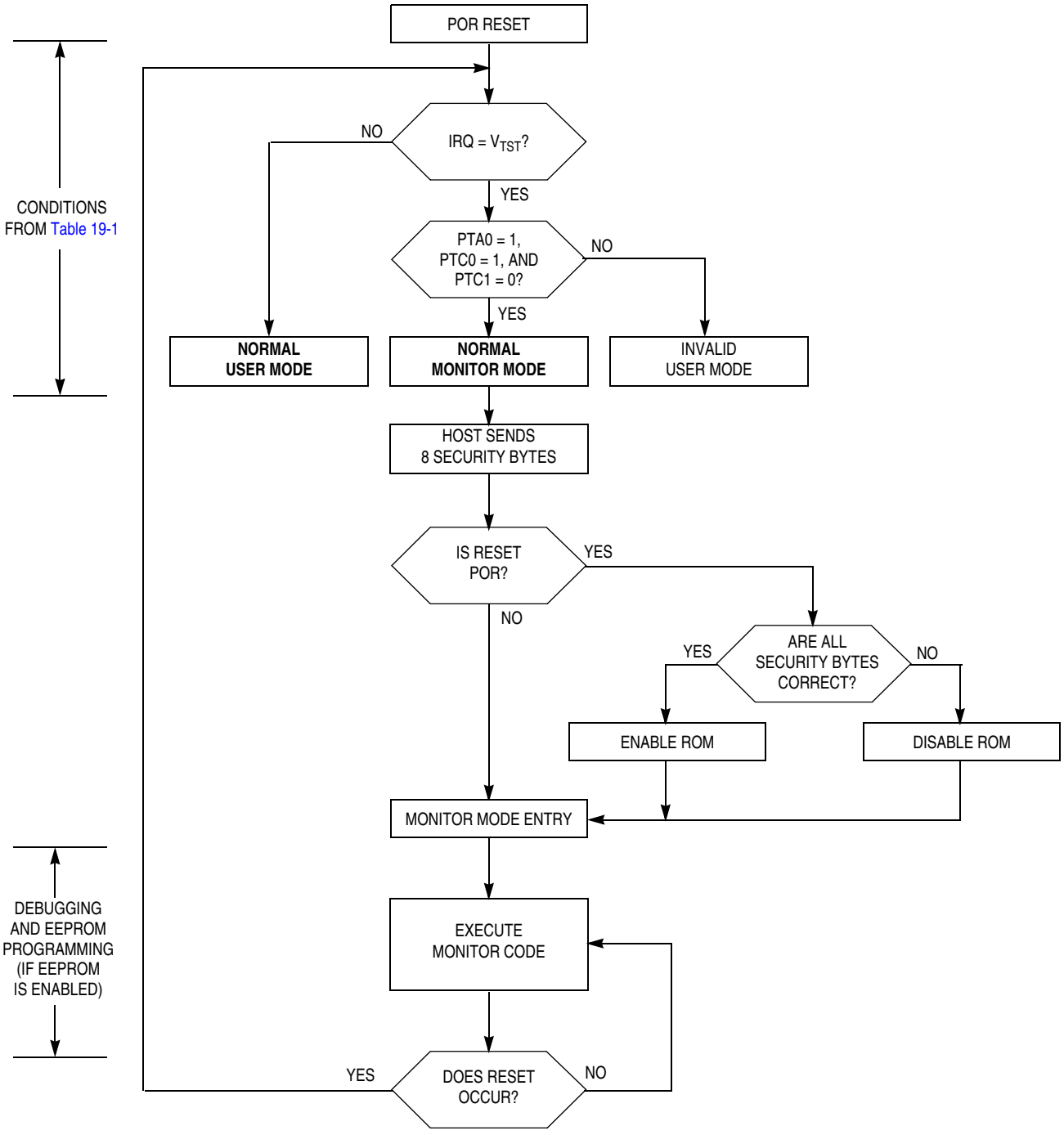


Figure 19-7. Simplified Monitor Mode Entry Flowchart

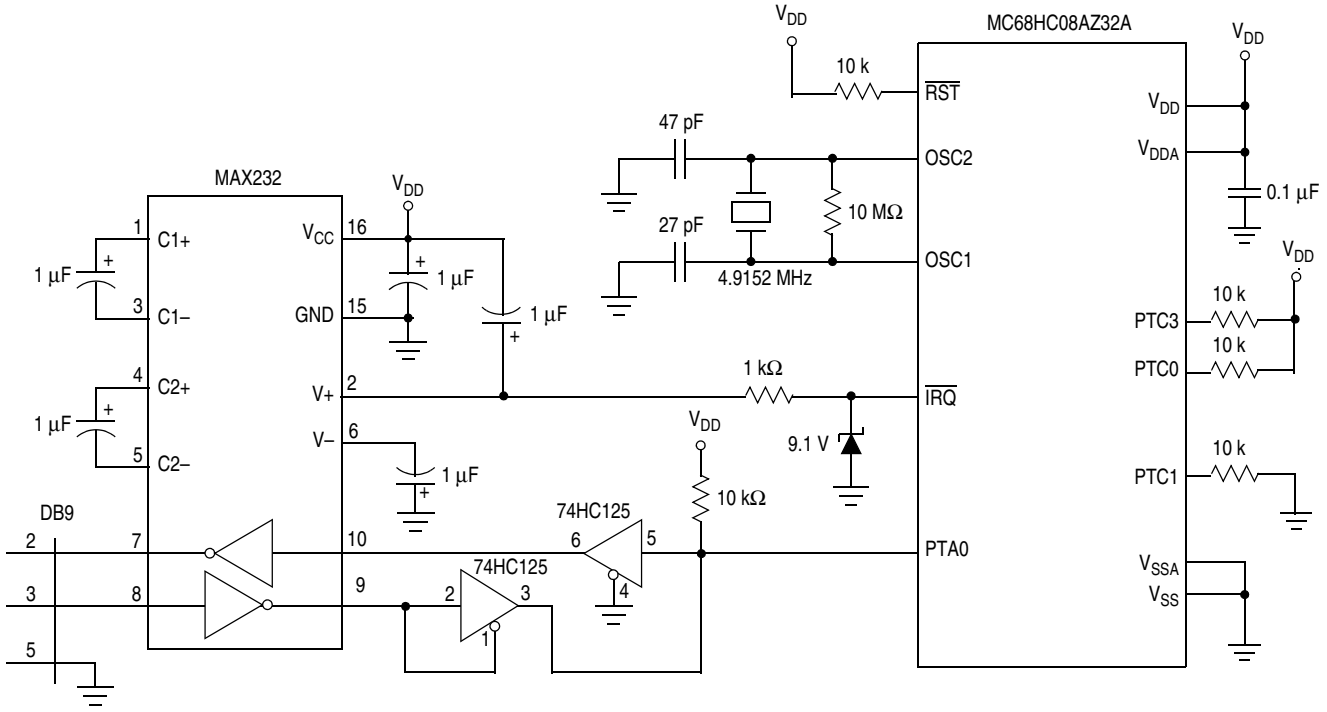


Figure 19-8. Normal Monitor Mode Circuit

Simple monitor commands can access any memory address. In monitor mode, the MCU can execute code downloaded into RAM by a host computer while most MCU pins retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.

19.3.1.1 Monitor Mode Entry

Table 19-1 shows the pin conditions for entering monitor mode. As specified in the table, monitor mode may be entered after a power-on reset (POR) and will allow communication provided the pin and clock conditions are met.

The rising edge of the internal $\overline{\text{RST}}$ signal latches the monitor mode. Once monitor mode is latched, the values on PTC0, PTC1, and PTC3 pins can be changed.

Once out of reset, the MCU waits for the host to send eight security bytes (see 19.3.2 Security). After the security bytes, the MCU sends a break signal (10 consecutive logic 0s) to the host, indicating that it is ready to receive a command.

Table 19-1. Mode Selection

$\overline{\text{IRQ}}$	PTC0	PTC1	PTA0	PTC3	Mode	CGMOUT	Bus Frequency
V_{TSTI}(1)	1	0	1	1	Monitor	$\frac{\text{CGMXCLK}}{2}$ or $\frac{\text{CGMVCLK}}{2}$	$\frac{\text{CGMOUT}}{2}$
V_{TSTI}(1)	1	0	1	0	Monitor	CGMXCLK	$\frac{\text{CGMOUT}}{2}$

1. For V_{TST}, 20.5 5.0 Volt DC Electrical Characteristics and 20.2 Maximum Ratings.

19.3.1.2 Monitor Vectors

In monitor mode, the MCU uses different vectors for reset, SWI (software interrupt), and break interrupt than those for user mode. 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} is applied to either the $\overline{IRQ}$ pin or the $\overline{RST}$ pin.

Table 19-2 summarizes the differences between user mode and monitor mode regarding vectors.

Table 19-2. Mode Differences

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

1. If the high voltage (V_{TST}) is removed from the $\overline{IRQ}$ pin while in monitor mode, the SIM asserts its COP enable output. The COP is a mask option enabled or disabled by the COPD bit in the configuration register. See 20.5 5.0 Volt DC Electrical Characteristics.

19.3.1.3 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. Transmit and receive baud rates must be identical.

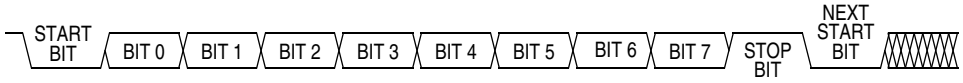


Figure 19-9. Monitor Data Format

19.3.1.4 Break Signal

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

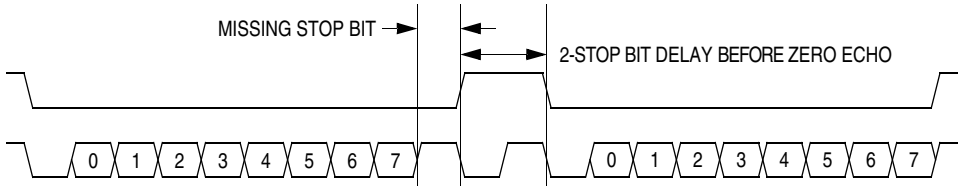


Figure 19-10. Break Transaction

19.3.1.5 Commands

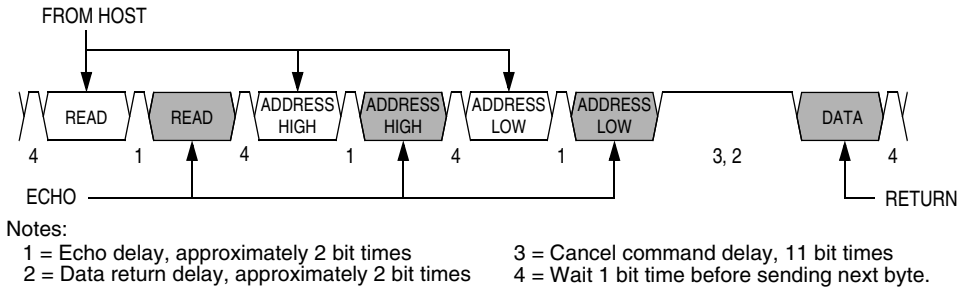
The monitor ROM firmware uses these commands:

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

The monitor ROM firmware echoes each received byte back to the PTA0 pin for error checking. An 11-bit delay at the end of each command allows the host to send a break character to cancel the command. A delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. The data returned by a read command appears after the echo of the last byte of the command.

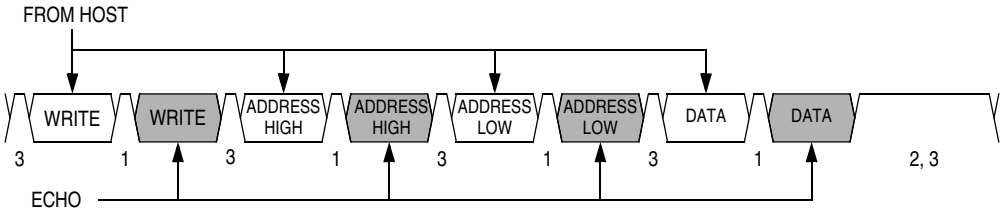
NOTE

Wait one bit time after each echo before sending the next byte.



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

Figure 19-11. Read Transaction



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

Figure 19-12. Write Transaction

A brief description of each monitor mode command is given in Table 19-3 through Table 19-8.

Table 19-3. READ (Read Memory) Command

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

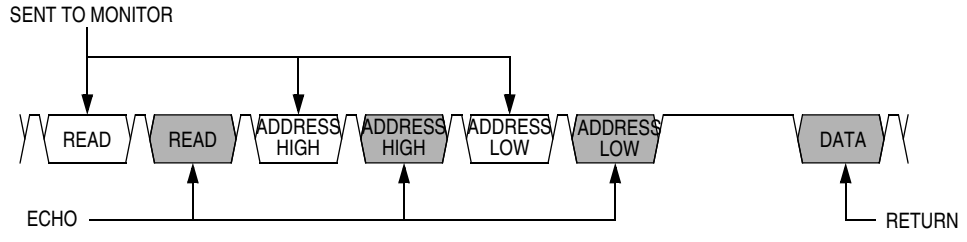


Table 19-4. WRITE (Write Memory) Command

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

Command Sequence

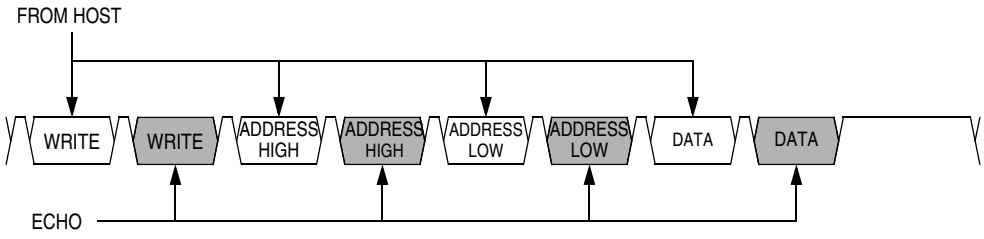


Table 19-5. IREAD (Indexed Read) Command

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

Command Sequence

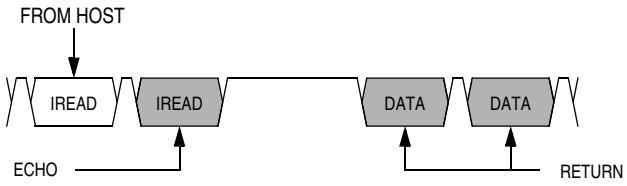
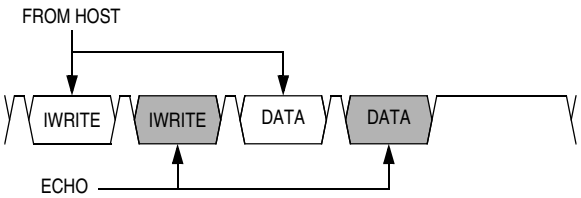


Table 19-6. IWRITE (Indexed Write) Command

Description	Write to last address accessed + 1
Operand	Single data byte
Data Returned	None
Opcode	$19

Command Sequence



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

Table 19-7. READSP (Read Stack Pointer) Command

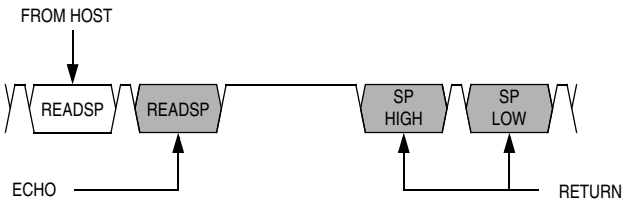
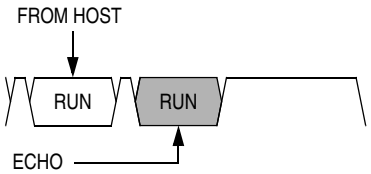
Description	Reads stack pointer
Operand	None
Data Returned	Returns incremented stack pointer value (SP + 1) in high-byte:low-byte order
Opcode	$0C
Command Sequence	

Table 19-8. RUN (Run User Program) Command

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

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

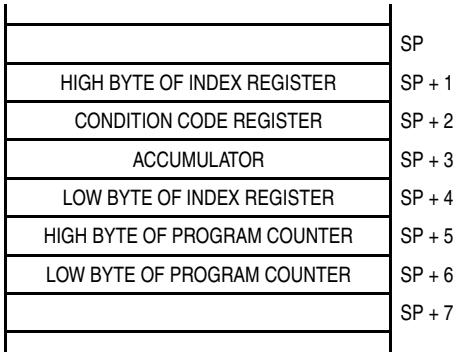


Figure 19-13. Stack Pointer at Monitor Mode Entry

19.3.2 Security

A security feature discourages unauthorized reading of ROM 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 ROM locations and execute code from ROM. 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 19-14.

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 ROM location returns an invalid value and trying to execute code from ROM 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.

NOTE

The MCU does not transmit a break character until after the host sends the eight security bytes.

If the security sequence fails, the device should be reset by a power-on reset and brought up in monitor mode to attempt another entry.

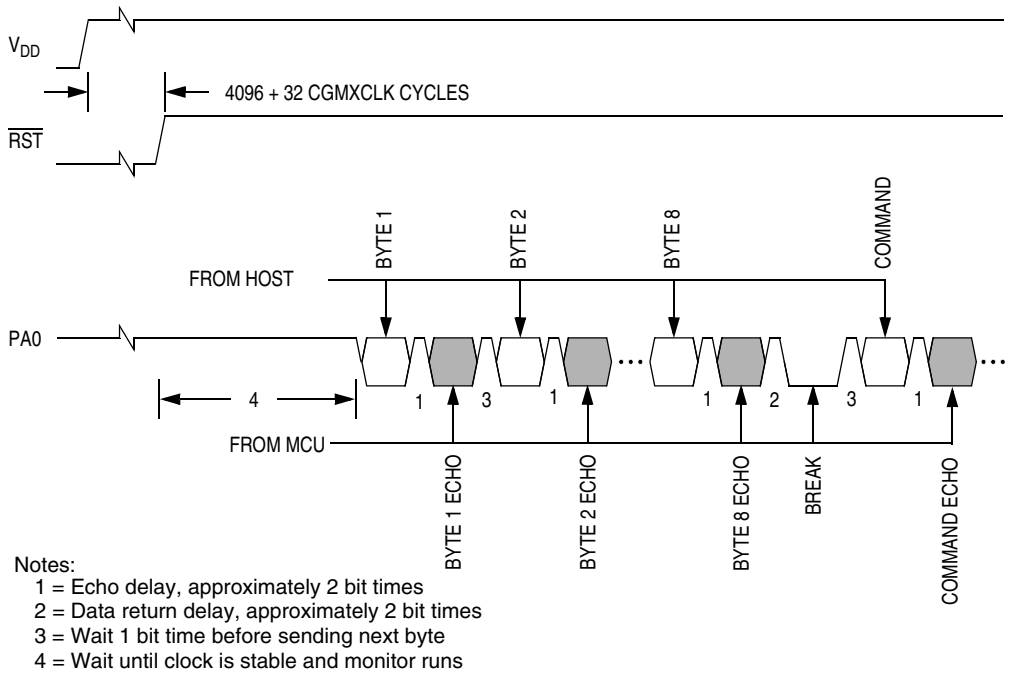


Figure 19-14. Monitor Mode Entry Timing

19.3.2.1 Baud Rate

The MC68HC08AZ32A features a monitor mode which is optimized to operate with either a 4.9152-MHz crystal clock source (or multiples of 4.9152 MHz) or a 4-MHz crystal (or multiples of 4 MHz). This supports designs which use the MSCAN08 module, which is generally clocked from a 4 MHz, 8 MHz, or 16 MHz internal reference clock. The table below outlines the available baud rates for a range of crystals and how they can match to a PC baud rate.

Table 19-9. Monitor Baud Rate Selection

Clock Frequency	Baud Rate		Closest PC Baud Rate		Error Percent	
	PTC3 = 0	PTC3 = 1	PTC3 = 0	PTC3 = 1	PTC3 = 0	PTC3 = 1
32 kHz	57.97	28.98	57.6	28.8	0.64	0.63
1 MHz	1811.59	905.80	1800	900	0.64	0.64
2 MHz	3623.19	1811.59	3600	1800	0.64	0.64
4 MHz	7246.37	3623.19	7200	3600	0.64	0.64
4.194 MHz	7597.83	3798.91	7680	3840	1.08	1.08
4.9152 MHz	8904.35	4452.17	8861	4430	0.49	0.50
8 MHz	14492.72	7246.37	14400	7200	0.64	0.64
16 MHz	28985.51	14492.75	28800	14400	0.64	0.64

WARNING

Care should be taken when setting the baud rate since incorrect baud rate setting can result in communications failure.

Chapter 20
Electrical Specifications

20.1 Introduction

This section contains electrical and timing specifications.

20.2 Maximum Ratings

Maximum ratings are the extreme limits to which the microcontroller unit (MCU) can be exposed without permanently damaging it.

NOTE

This device is not guaranteed to operate at the maximum ratings. Plese refer to 20.5 5.0 Volt DC Electrical Characteristics for guaranteed operating conditions.

Rating[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} and V_{SS}	I	± 25	mA
Storage temperature	T_{STG}	–55 to +150	°C
Maximum current out of V_{SS}	I_{MVSS}	100	mA
Maximum current into V_{DD}	I_{MVDD}	100	mA
$\overline{RST}$ and $\overline{IRQ}$ input voltage	V_{TST}	V_{DD} + 4.5	V

1. Voltages are referencecd 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}).

20.3 Functional Operating Range

Rating	Symbol	Value	Unit
Operating temperature range(1)	T_A	–40 to $T_{A(MAX)}$	°C
Operating voltage range	V_{DD}	5.0 ± 0.5	V

1. $T_{A\ (MAX)}$ =125°C for part suffix MFU
105°C for part suffix VFU
85°C for part suffix CFU

NOTE

For applications which use the LVI, Freescale guarantee the functionality of the device down to the LVI trip point (V_{LVI}) within the constraints outlined in Chapter 9 Low-Voltage Inhibit (LVI) Module).

20.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance QFP (64 pins)	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

1. Power dissipation is a function of temperature.
2. K is a constant unique to the device. K can be determined from a known T_A and measured P_D. With this value of K, P_D and T_J can be determined for any value of T_A.

20.5 5.0 Volt DC Electrical Characteristics

Characteristic(1)	Symbol	Min	Max	Unit
Output high voltage I_{Load} = –2.0 mA (all ports) I_{Load} = –5.0 mA (all ports)	V_{OH}	 V_{DD} –0.8 V_{DD} –1.5	 — —	V
Total source current	$I_{OH(TOT)}$	—	10	mA
Output low voltage I_{Load} = 1.6 mA (all ports) I_{Load} = 10.0 mA (all ports)	V_{OL}	 — —	 0.4 1.5	V
Total sink current	$I_{OL(TOT)}$	—	15	mA

— Contined on next page

Characteristic[1]	Symbol	Min	Max	Unit
Input high voltage (all ports, $\overline{IRQ}$, $\overline{RST}$, OSC1)	V_{IH}	0.7 x V_{DD}	V_{DD}	V
Input low voltage (all ports, $\overline{IRQ}$, $\overline{RST}$, OSC1)	V_{IL}	V_{SS}	0.3 x V_{DD}	V
V_{DD} + V_{DDA} supply current Run[2] [3] Wait[4] [3] Stop[5] 25°C –40°C to +125°C 25°C with LVI enabled –40°C to +125°C with LVI enabled	I_{DD}	 — — — — — —	 30 14 50 100 400 500	 mA mA µA µA µA µA
I/O ports Hi-Z leakage current	I_L	–1	1	µA
I/O ports Hi-Z leakage current[6]	I_L	–10	10	µA
Input current	I_{In}	–1	1	µA
Input current	I_{In}	–10	10	µA
Capacitance Ports (as input or output)	C_{Out} C_{In}	— —	12 8	pF
Low-voltage reset inhibit Trip Recover	V_{LVI}	 3.80 —	 — 4.49	V
POR re-arm 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
High COP disable voltage[10]	V_{TST}	V_{DD} + 3	V_{DD} + 4.5	V
Monitor mode entry voltage on $\overline{IRQ}$[11]	V_{TST}	V_{DD} + 3	V_{DD} + 4.5	V

1. V_{DD} = 5.0 Vdc ± 0.5 V, V_{SS} = 0 Vdc, T_A = –40°C to $T_{A\ (MAX)}$, unless otherwise noted.
2. 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.
3. Although I_{DD} is proportional to bus frequency, a current of several mA is present even at very low frequencies.
4. Wait I_{DD} measured using external square wave clock source (f_{Bus} = 8.4 MHz). All inputs 0.2 Vdc 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 all modules enabled.
5. Stop I_{DD} measured with OSC1 = V_{SS}.
6. When subjected to a Human Body Model (HBM) ESD event as specified in AEC Q100-002 these pins may exhibit recoverable leakage values within the specification indicated.
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.
10. See Chapter 5 Computer Operating Properly (COP) Module. V_{TST} applied to $\overline{RST}$.
11. See monitor mode description within Chapter 5 Computer Operating Properly (COP) Module. V_{TST} applied to $\overline{IRQ}$ or $\overline{RST}$.

20.6 Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Bus operating frequency (4.5–5.5 V — V_{DD} only)	f_{Bus}	—	8.4	MHz
Internal clock period ($1/f_{Bus}$)	t_{CYC}	119	—	ns
$\overline{RST}$ pulse width low	t_{RL}	1.5	—	t_{CYC}
$\overline{IRQ}$ interrupt pulse width low (edge triggered)	t_{ILHI}	1.5	—	t_{CYC}
$\overline{IRQ}$ Interrupt Pulse Period	t_{ILIL}	Note 3	—	t_{CYC}
16-bit timer Input capture pulse width[2] Input capture period Input clock pulse width	 t_{TH}, t_{TL} t_{TLTL} t_{TCH}, t_{TCL}	 2 Note[3] $(1/f_{OP}) + 5$	 — — —	 t_{CYC} t_{CYC} ns
MSCAN wakeup filter pulse width[4]	t_{WUP}	2	5	µs

1. V_{DD} = 5.0 Vdc ± 0.5 V, V_{SS} = 0 Vdc, T_A = –40°C to $T_{A\ (MAX)}$, unless otherwise noted.
2. Refer to Table 17-2. Mode, Edge, and Level Selection, Table 18-2. Mode, Edge, and Level Selection, and supporting notes.
3. The minimum period t_{TLTL} or t_{ILIL} should not be less than the number of cycles it takes to execute the capture interrupt service routine plus t_{CYC}.
4. The minimum pulse width to wake up the MSCAN module is guaranteed by design but not tested.

20.7 ADC Characteristics

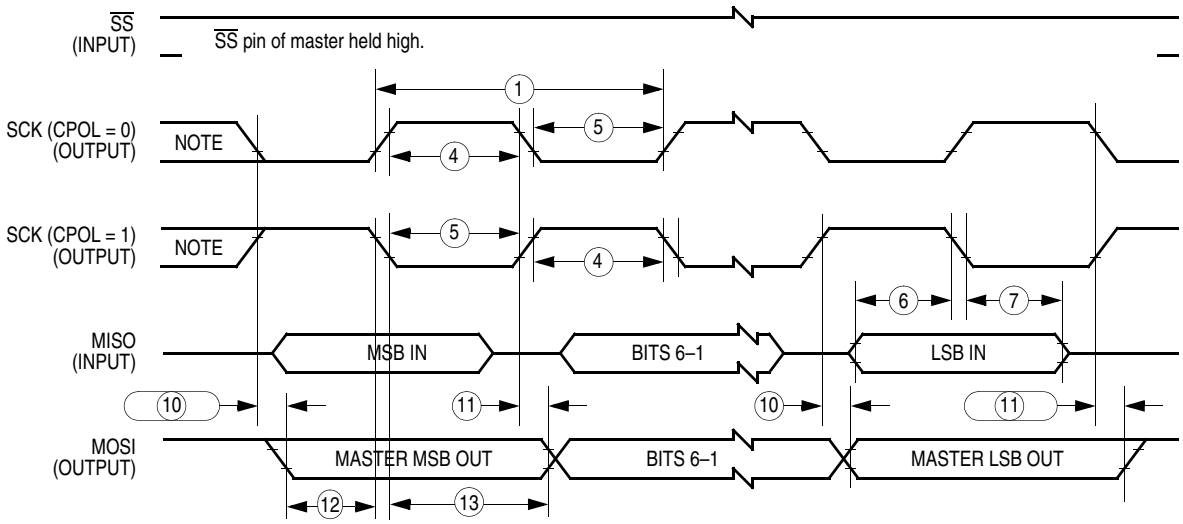
Characteristic[1]	Min	Max	Unit	Comments
Resolution	8	8	Bits	
Absolute accuracy (V_{REFL} = 0 V, V_{DDA} = V_{REFH} = 5 V ± 0.5 V)	–1	+1	LSB	Includes quantization
Conversion range	V_{REFL}	V_{REFH}	V	V_{REFL} = V_{SSA}
Powerup time	16	17	µs	Conversion time period
Input leakage[2] (ports B and D)	– 1	1	µA	
Input leakage[3] (ports B and D)	–10	10	µA	
Conversion Time	16	17	ADC clock cycles	Includes sampling time
Monotonicity	Inherent within total error			
Zero input reading	00	01	Hex	V_{IN} = V_{REFL}
Full-scale reading	FE	FF	Hex	V_{IN} = V_{REFH}
Sample time[4]	5	—	ADC clock cycles	
Input capacitance	—	8	pF	Not tested
ADC internal clock	500 k	1.048 M	Hz	Tested only at 1 MHz
Analog input voltage	V_{REFL}	V_{REFH}	V	

1. V_{DD} = 5.0 Vdc ± 0.5 V, V_{SS} = 0 Vdc, V_{DDA}/V_{DDAREF} = 5.0 Vdc ± 0.5 V, V_{SSA} = 0 Vdc, V_{REFH} = 5.0 Vdc ± 0.5 V
2. The external system error caused by input leakage current is approximately equal to the product of R source and input current.
3. When subjected to a Human Body Model (HBM) ESD event as specified in AEC Q100-002 these pins may exhibit recoverable leakage values within the specification indicated.
4. Source impedances greater than 10 kΩ adversely affect internal RC charging time during input sampling.

20.8 5.0 Vdc ± 0.5 V Serial Peripheral Interface (SPI) Timing

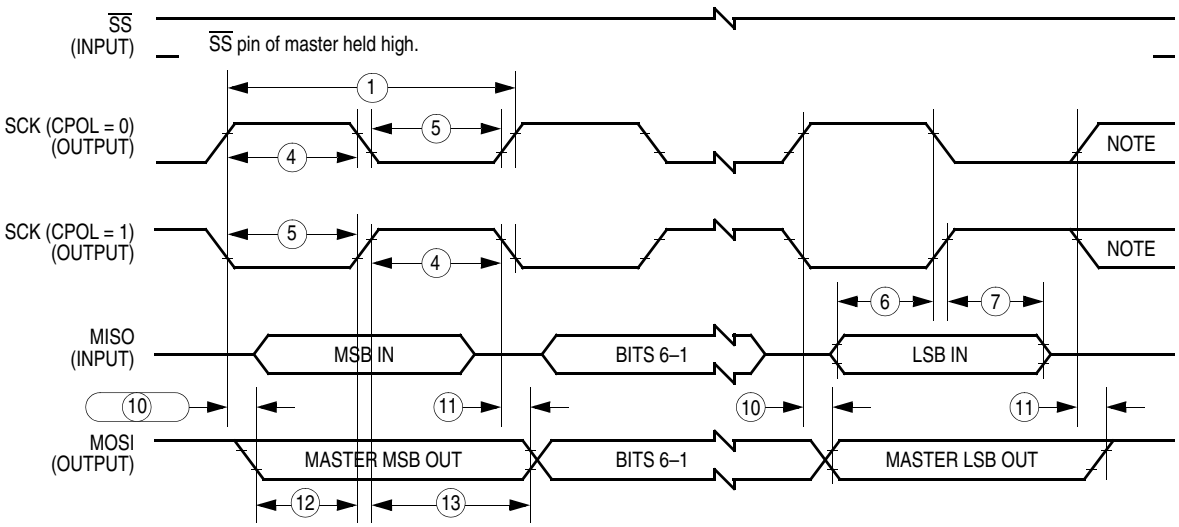
Num(1)	Characteristic(2)	Symbol	Min	Max	Unit
	Operating frequency(3) Master Slave	 $f_{BUS(M)}$ $f_{Bus(S)}$	 $f_{BUS}/128$ dc	 $f_{BUS}/2$ f_{BUS}	MHz
1	Cycle time Master Slave	 $t_{CYC(M)}$ $t_{CYC(S)}$	 2 1	 128 —	t_{CYC}
2	Enable lead time	t_{Lead}	15	—	ns
3	Enable lag time	t_{Lag}	15	—	ns
4	Clock (SCK) high time Master Slave	 $t_{W(SCKH)M}$ $t_{W(SCKH)S}$	 100 50	 — —	ns
5	Clock (SCK) low time Master Slave	 $t_{W(SCKL)M}$ $t_{W(SCKL)S}$	 100 50	 — —	ns
6	Data setup time (inputs) Master Slave	 $t_{SU(M)}$ $t_{SU(S)}$	 45 5	 — —	ns
7	Data hold time (inputs) Master Slave	 $t_{H(M)}$ $t_{H(S)}$	 0 15	 — —	ns
8	Access time, slave(4) CPHA = 0 CPHA = 1	 $t_{A(CP0)}$ $t_{A(CP1)}$	 0 0	 40 20	ns
9	Slave disable time (hold time to high-impedance state)	t_{DIS}	—	25	ns
10	Enable edge kead time to data valid(5) Master Slave	 $t_{EV(M)}$ $t_{EV(S)}$	 — —	 10 40	ns
11	Data hold time (outputs, after enable edge) Master Slave	 $t_{HO(M)}$ $t_{HO(S)}$	 0 5	 — —	ns
12	Data valid Master (before capture edge)	$t_{V(M)}$	90	—	ns
13	Data hold time (outputs) Master (after capture edge)	$t_{HO(M)}$	100	—	ns

1. Item numbers refer to dimensions in Figure 20-1 and Figure 20-2.
2. All timing is shown with respect to 30% V_{DD} and 70% V_{DD}, unless otherwise noted; assumes 100 pF load on all SPI pins.
3. f_{Bus} = the currently active bus frequency for the microcontroller.
4. Time to data active from high-impedance state.
5. With 100 pF on all SPI pins



NOTE: This first clock edge is generated internally, but is not seen at the SCK pin.

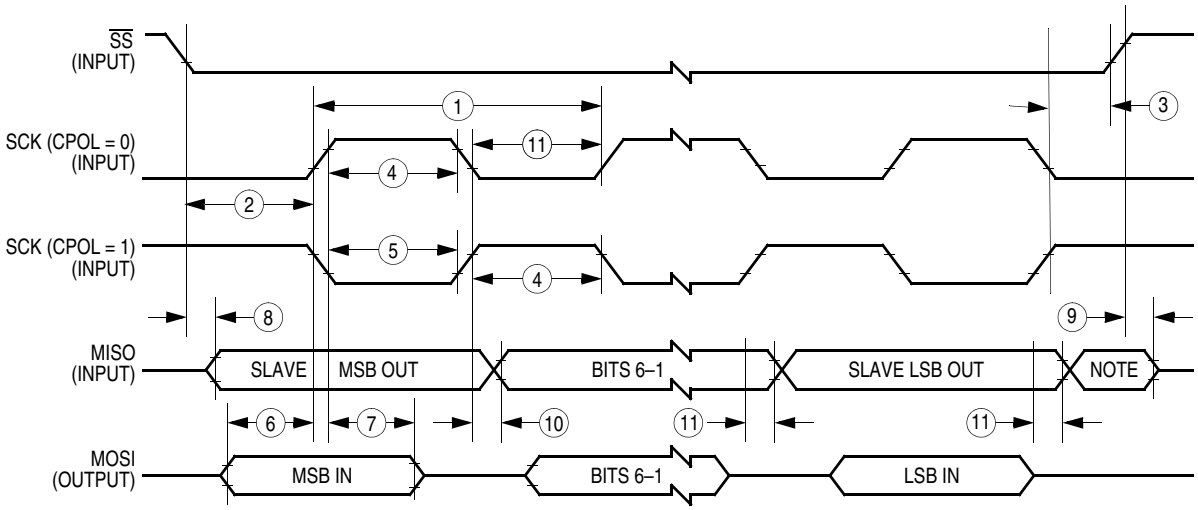
a) SPI Master Timing (CPHA = 0)



NOTE: This last clock edge is generated internally, but is not seen at the SCK pin.

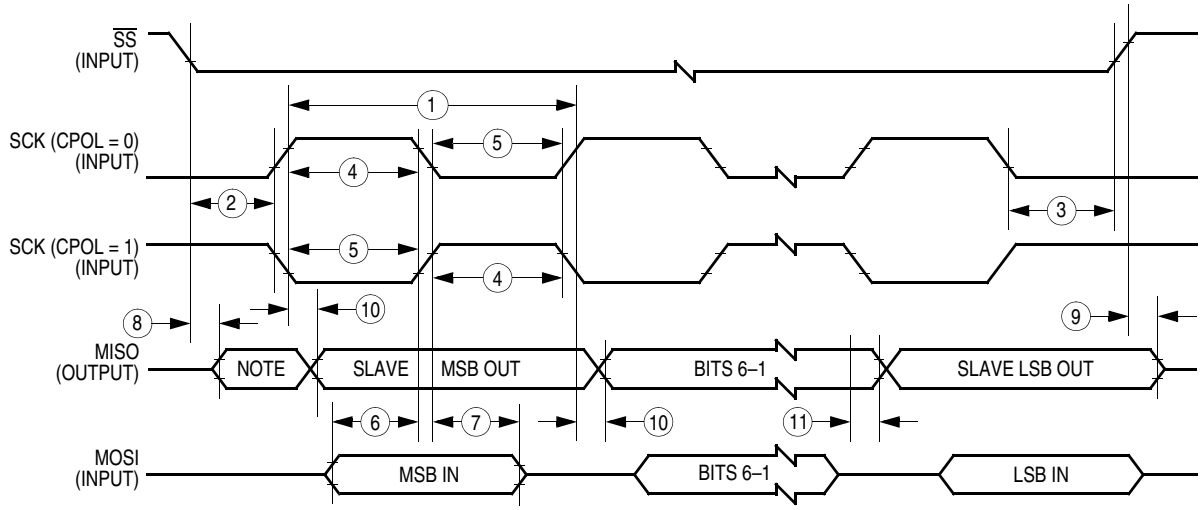
b) SPI Master Timing (CPHA = 1)

Figure 20-1. SPI Master Timing Diagram



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 20-2. SPI Slave Timing Diagram

20.9 CGM Operating Conditions

Characteristic	Symbol	Min	Typ	Max	Unit
Operating voltage	V_{DDA}	V_{DD}–0.3	—	V_{DD}+0.3	V
	V_{SSA}	V_{SS} –0.3	—	V_{SS}+0.3	V
Crystal reference frequency	$f_{CGMRCLK}$	1	4.9152	8	MHz
Module crystal reference frequency(1)	$f_{CGMXCLK}$	—	4.9152	—	MHz
Range nominal multiplier	f_{NOM}	—	4.9152	—	MHz
VCO center-of-range frequency	f_{CGMVRS}	4.9152	—	Note(2)	MHz
VCO operating frequency	$f_{CGMVCLK}$	4.9152	—	32.0	

1. Same frequency as $f_{CGMRCLK}$.
2. f_{CGMVRS} is a nominal value described and calculated as an example in Chapter 4 Clock Generator Module (CGM) for the desired VCO operating frequency, $f_{CGMVCLK}$.

20.10 CGM Component Information

Description	Symbol	Min	Typ	Max	Unit
Crystal load capacitance(1)	C_L	—	—	—	—
Crystal fixed capacitance(1)	C1	—	2 x CL	—	—
Crystal tuning capacitance(1)	C2	—	2 x CL	—	—
Filter capacitor multiply factor	C_{FACT}	—	0.0154	—	F/s V
Filter capacitor(2)	C_F	—	C_{FACT} x (V_{DDA}/$f_{CGMXCLK}$)	—	—
Bypass capacitor(3)	C_{BYP}	—	0.1 µF	—	µF

1. Consult crystal manufacturer's data
2. See 4.4.3 External Filter Capacitor Pin (CGMXFC).
3. CBYP must provide low AC impedance from f = $f_{CGMXCLK}$/100 to 100 x $f_{CGMVCLK}$. So, series resistance must be considered.

20.11 CGM Acquisition/Lock Time Information

Characteristic[1]	Symbol	Min	Typ[2]	Max[2]	Unit
Manual mode time to stable[3]	t_{ACQ}	—	$(8 \times V_{DDA}) / (f_{CGMXCLK} \times K_{ACQ})$	—	s
Manual stable to lock time[3]	t_{AL}	—	$(4 \times V_{DDA}) / (f_{CGMXCLK} \times K_{TRK})$	—	s
Manual acquisition time	t_{Lock}	—	$t_{ACQ} + t_{AL}$	—	s
Tracking mode entry frequency tolerance	D_{TRK}	0	—	± 3.6	%
Acquisition mode entry frequency tolerance	D_{UNT}	± 6.3	—	± 7.2	%
LOCK entry freqency tolerance	D_{LOCK}	0	—	± 0.9	%
LOCK Exit freqency 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[3]	t_{ACQ}	$n_{ACQ}/f_{CGMXCLK}$	$(8 \times V_{DDA}) / (f_{CGMXCLK} \times K_{ACQ})$	—	s
Automatic stable to lock time[3]	t_{AL}	$n_{TRK}/f_{CGMXCLK}$	$(4 \times V_{DDA}) / (f_{CGMXCLK} \times K_{TRK})$	—	s
Automatic lock time	t_{Lock}	—	0.65	25	ms
PLL jitter, deviation of average bus frequency over 2 ms [4] [5]		0	—	± (f_{CRYS}) x (.025%) x (N/4)	%
K value for automatic mode time to stable	K_{ACQ}	—	0.2	—	—
K value	K_{TRK}	—	0.004	—	—

1. V_{DD} = 5.0 Vdc ± 0.5 V, V_{SS} = 0 Vdc, T_A = –40°C to $T_{A(MAX)}$, unless otherwise noted.
2. Conditions for typical and maximum values are for run mode with $f_{CGMXCLK}$ = 8 MHz, f_{BUSDES} = 8 MHz, N = 4, L = 7, discharged C_F = 15 nF, V_{DD} = 5 Vdc.
3. If C_F is chosen correctly.
4. Guaranteed but not tested. Refer to 4.3.2 Phase-Locked Loop Circuit (PLL) for guidance on the use of the PLL.
5. N = VCO frequency multiplier.

20.12 RAM Memory Characteristics

Characteristic	Symbol	Min	Max	Unit
RAM data retention voltage	V_{RDR}	0.7	—	V

20.13 EEPROM Memory Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
EEPROM programming time per byte	t_{EEPGM}	10	—	—	ms
EEPROM erasing time per byte	t_{EEBYTE}	10	—	—	ms
EEPROM erasing time per block	$t_{EEBLOCK}$	10	—	—	ms
EEPROM erasing time per bulk	t_{EEBULK}	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 programming maximum time to AUTO bit set	—	—	—	500	µs
EEPROM erasing maximum time to AUTO bit set	—	—	—	8	ms
EEPROM endurance[2]	—	10K	>100K	—	Cycles
EEPROM data retention[3]	—	15	>100	—	Years

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.
2. Typical endurance was evaluated for this product family. For additional information on how Freescale defines *Typical Endurance*, please refer to Engineering Bulletin EB619.
3. Typical data retention values are based on intrinsic capability of the technology measured at high temperature and de-rated to 25°C using the Arrhenius equation. For additional information on how Freescale defines *Typical Data Retention*, please refer to Engineering Bulletin EB618.

Chapter 21
Ordering Information and Mechanical Specifications

21.1 Introduction

This section provides ordering information for the MC68HC08AZ32A along with the dimensions for the 64-pin quad flat pack (QFP)

The following figure shows the latest package drawing at the time of this publication. To make sure that you have the latest package specifications, contact your local Freescale sales office

21.2 MC Order Numbers

These part numbers are generic numbers only. To place an order, ROM code must be submitted to the ROM Processing Center (RPC). Refer to the Customer Interface Tool (CIT) link at:
http://freescale.com

Table 21-1. MC Order Numbers

MC Order Number	Operating Temperature Range
MC68HC08AZ32ACFU	–40°C to +85°C
MC68HC08AZ32AVFU	–40°C to +105°C
MC68HC08AZ32AMFU	–40°C to +125°C

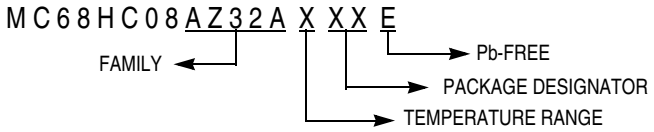
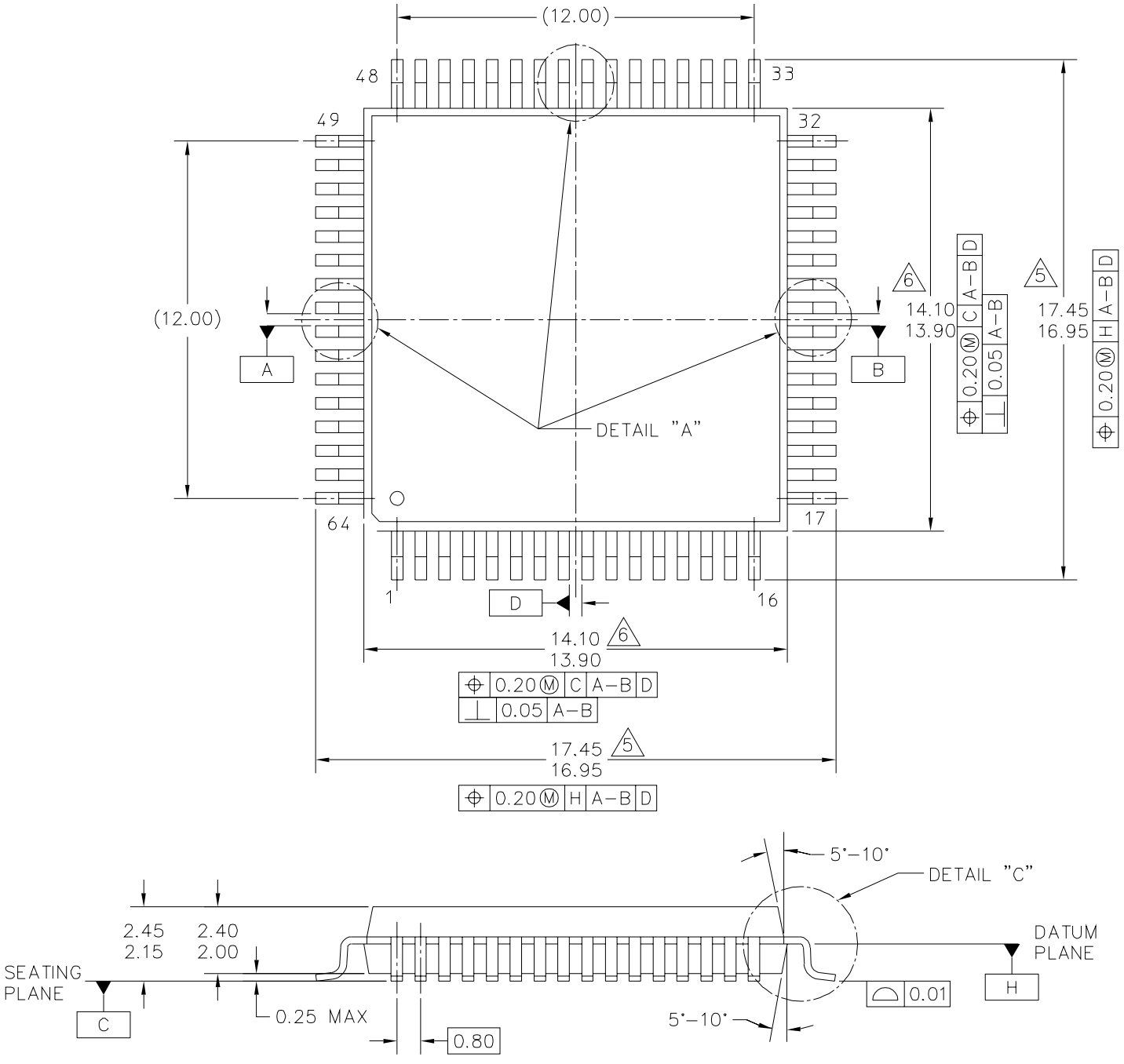


Figure 21-1. Device Numbering System

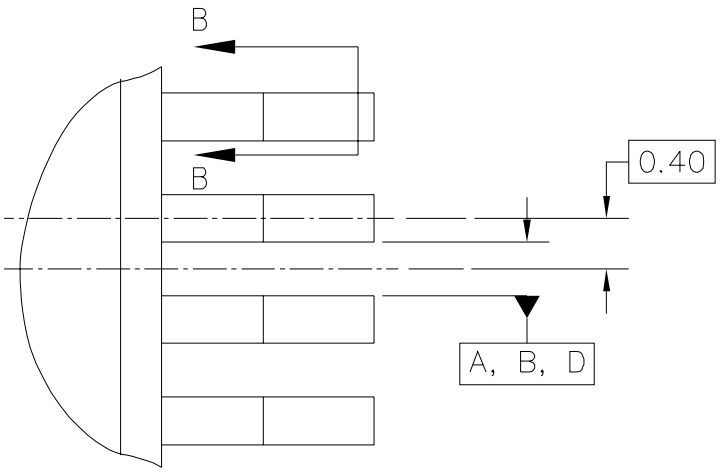
21.3 Package Dimensions

Refer to the following pages for detailed package dimensions.

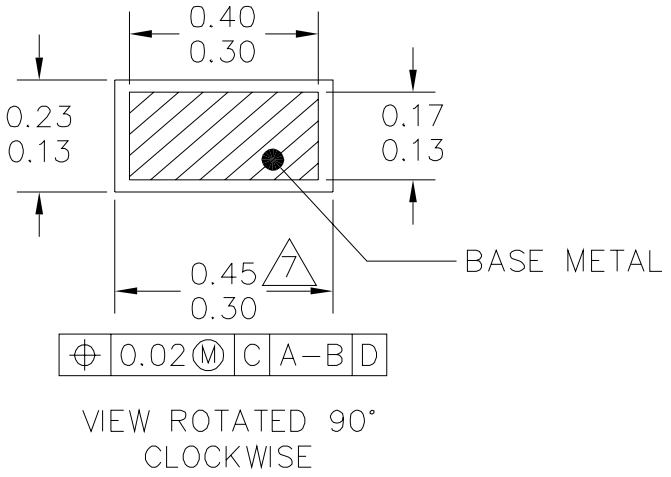


MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

TITLE:	DOCUMENT NO: 98ASB42844B	REV: A
64LD QFP (14 X 14)	CASE NUMBER: 840B-02	06 APR 2005
	STANDARD: NON-JEDEC	

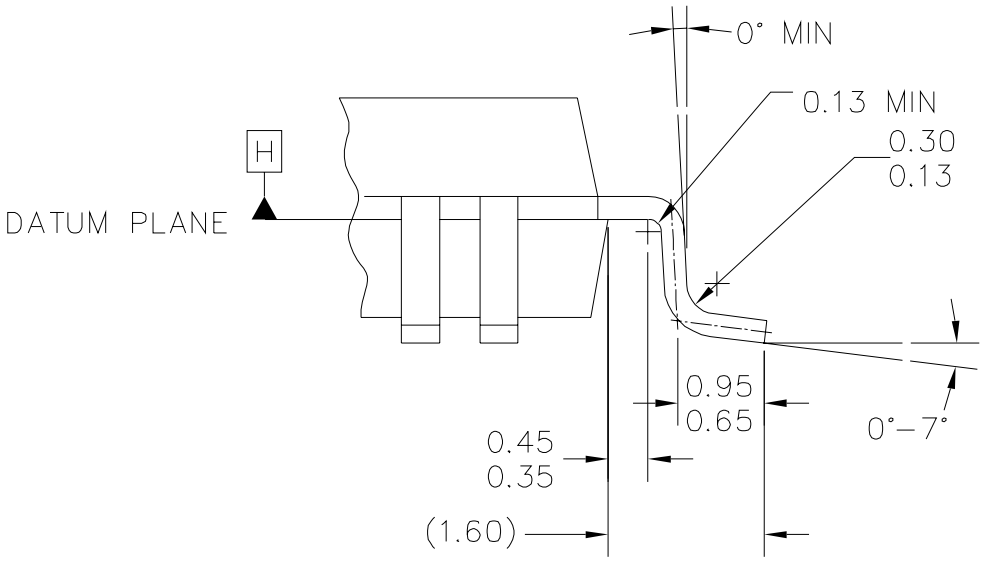


DETAIL "A"



VIEW ROTATED 90° CLOCKWISE

SECTION B–B



DETAIL "C"

MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

TITLE:	DOCUMENT NO: 98ASB42844B	REV: A
64LD QFP (14 X 14)	CASE NUMBER: 840B-02	06 APR 2005
	STANDARD: NON-JEDEC	

NOTES:

1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.

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 TO BE DETERMINED AT SEATING PLANE −C−.

6. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. DIMENSIONS DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE −H−.

7. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08mm TOTAL IN EXCESS OF THE DIMENSION AT MAXIMUM MATERIAL CONDICTION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

TITLE:	DOCUMENT NO: 98ASB42844B	REV: A
64LD QFP (14 X 14)	CASE NUMBER: 840B-02	06 APR 2005
	STANDARD: NON-JEDEC	

Glossary

A — See “accumulator (A).”

accumulator (A) — An 8-bit general-purpose register in the CPU08. The CPU08 uses the accumulator to hold operands and results of arithmetic and logic operations.

acquisition mode — A mode of PLL operation during startup before the PLL locks on a frequency. Also see "tracking mode."

address bus — The set of wires that the CPU or DMA uses to read and write memory locations.

addressing mode — The way that the CPU determines the operand address for an instruction. The M68HC08 CPU has 16 addressing modes.

ALU — See “arithmetic logic unit (ALU).”

arithmetic logic unit (ALU) — The portion of the CPU that contains the logic circuitry to perform arithmetic, logic, and manipulation operations on operands.

asynchronous — Refers to logic circuits and operations that are not synchronized by a common reference signal.

baud rate — The total number of bits transmitted per unit of time.

BCD — See “binary-coded decimal (BCD).”

binary — Relating to the base 2 number system.

binary number system — The base 2 number system, having two digits, 0 and 1. Binary arithmetic is convenient in digital circuit design because digital circuits have two permissible voltage levels, low and high. The binary digits 0 and 1 can be interpreted to correspond to the two digital voltage levels.

binary-coded decimal (BCD) — A notation that uses 4-bit binary numbers to represent the 10 decimal digits and that retains the same positional structure of a decimal number. For example,

234 (decimal) = 0010 0011 0100 (BCD)

bit — A binary digit. A bit has a value of either logic 0 or logic 1.

branch instruction — An instruction that causes the CPU to continue processing at a memory location other than the next sequential address.

break module — A module in the M68HC08 Family. The break module allows software to halt program execution at a programmable point in order to enter a background routine.

breakpoint — A number written into the break address registers of the break module. When a number appears on the internal address bus that is the same as the number in the break address registers, the CPU executes the software interrupt instruction (SWI).

break interrupt — A software interrupt caused by the appearance on the internal address bus of the same value that is written in the break address registers.

bus — A set of wires that transfers logic signals.

bus clock — The bus clock is derived from the CGMOUT output from the CGM. The bus clock frequency, f_{op}, is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

byte — A set of eight bits.

C — The carry/borrow bit in the condition code register. The CPU08 sets the carry/borrow bit when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the carry/borrow bit (as in bit test and branch instructions and shifts and rotates).

CCR — See “condition code register.”

central processor unit (CPU) — The primary functioning unit of any computer system. The CPU controls the execution of instructions.

CGM — See “clock generator module (CGM).”

clear — To change a bit from logic 1 to logic 0; the opposite of set.

clock — A square wave signal used to synchronize events in a computer.

clock generator module (CGM) — A module in the M68HC08 Family. The CGM generates a base clock signal from which the system clocks are derived. The CGM may include a crystal oscillator circuit and or phase-locked loop (PLL) circuit.

comparator — A device that compares the magnitude of two inputs. A digital comparator defines the equality or relative differences between two binary numbers.

computer operating properly module (COP) — A counter module in the M68HC08 Family that resets the MCU if allowed to overflow.

condition code register (CCR) — An 8-bit register in the CPU08 that contains the interrupt mask bit and five bits that indicate the results of the instruction just executed.

control bit — One bit of a register manipulated by software to control the operation of the module.

control unit — One of two major units of the CPU. The control unit contains logic functions that synchronize the machine and direct various operations. The control unit decodes instructions and generates the internal control signals that perform the requested operations. The outputs of the control unit drive the execution unit, which contains the arithmetic logic unit (ALU), CPU registers, and bus interface.

COP — See "computer operating properly module (COP)."

counter clock — The input clock to the TIM counter. This clock is the output of the TIM prescaler.

CPU — See “central processor unit (CPU).”

CPU08 — The central processor unit of the M68HC08 Family.

CPU clock — The CPU clock is derived from the CGMOUT output from the CGM. The CPU clock frequency is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

CPU cycles — A CPU cycle is one period of the internal bus clock, normally derived by dividing a crystal oscillator source by two or more so the high and low times will be equal. The length of time required to execute an instruction is measured in CPU clock cycles.

CPU registers — Memory locations that are wired directly into the CPU logic instead of being part of the addressable memory map. The CPU always has direct access to the information in these registers. The CPU registers in an M68HC08 are:

- A (8-bit accumulator)
- H:X (16-bit index register)
- SP (16-bit stack pointer)
- PC (16-bit program counter)
- CCR (condition code register containing the V, H, I, N, Z, and C bits)

cycle time — The period of the operating frequency: $t_{CYC} = 1/f_{OP}$.

decimal number system — Base 10 numbering system that uses the digits zero through nine.

direct memory access module (DMA) — A M68HC08 Family module that can perform data transfers between any two CPU-addressable locations without CPU intervention. For transmitting or receiving blocks of data to or from peripherals, DMA transfers are faster and more code-efficient than CPU interrupts.

DMA — See "direct memory access module (DMA)."

DMA service request — A signal from a peripheral to the DMA module that enables the DMA module to transfer data.

duty cycle — A ratio of the amount of time the signal is on versus the time it is off. Duty cycle is usually represented by a percentage.

EEPROM — Electrically erasable, programmable, read-only memory. A nonvolatile type of memory that can be electrically reprogrammed.

EPROM — Erasable, programmable, read-only memory. A nonvolatile type of memory that can be erased by exposure to an ultraviolet light source and then reprogrammed.

exception — An event such as an interrupt or a reset that stops the sequential execution of the instructions in the main program.

external interrupt module (IRQ) — A module in the M68HC08 Family with both dedicated external interrupt pins and port pins that can be enabled as interrupt pins.

fetch — To copy data from a memory location into the accumulator.

firmware — Instructions and data programmed into nonvolatile memory.

free-running counter — A device that counts from zero to a predetermined number, then rolls over to zero and begins counting again.

full-duplex transmission — Communication on a channel in which data can be sent and received simultaneously.

H — The upper byte of the 16-bit index register (H:X) in the CPU08.

H — The half-carry bit in the condition code register of the CPU08. This bit indicates a carry from the low-order four bits of the accumulator value to the high-order four bits. The half-carry bit is required for binary-coded decimal arithmetic operations. The decimal adjust accumulator (DAA) instruction uses the state of the H and C bits to determine the appropriate correction factor.

hexadecimal — Base 16 numbering system that uses the digits 0 through 9 and the letters A through F.

high byte — The most significant eight bits of a word.

illegal address — An address not within the memory map

illegal opcode — A nonexistent opcode.

I — The interrupt mask bit in the condition code register of the CPU08. When I is set, all interrupts are disabled.

index register (H:X) — A 16-bit register in the CPU08. The upper byte of H:X is called H. The lower byte is called X. In the indexed addressing modes, the CPU uses the contents of H:X to determine the effective address of the operand. H:X can also serve as a temporary data storage location.

input/output (I/O) — Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal and writes to an output to change the level on an external signal.

instructions — Operations that a CPU can perform. Instructions are expressed by programmers as assembly language mnemonics. A CPU interprets an opcode and its associated operand(s) and instruction.

interrupt — A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine.

interrupt request — A signal from a peripheral to the CPU intended to cause the CPU to execute a subroutine.

I/O — See “input/output (I/0).”

IRQ — See "external interrupt module (IRQ)."

jitter — Short-term signal instability.

latch — A circuit that retains the voltage level (logic 1 or logic 0) written to it for as long as power is applied to the circuit.

latency — The time lag between instruction completion and data movement.

least significant bit (LSB) — The rightmost digit of a binary number.

logic 1 — A voltage level approximately equal to the input power voltage (V_{DD}).

logic 0 — A voltage level approximately equal to the ground voltage (V_{SS}).

low byte — The least significant eight bits of a word.

low voltage inhibit module (LVI) — A module in the M68HC08 Family that monitors power supply voltage.

LVI — See "low voltage inhibit module (LVI)."

M68HC08 — A Freescale family of 8-bit MCUs.

mark/space — The logic 1/logic 0 convention used in formatting data in serial communication.

mask — 1. A logic circuit that forces a bit or group of bits to a desired state. 2. A photomask used in integrated circuit fabrication to transfer an image onto silicon.

mask option — A optional microcontroller feature that the customer chooses to enable or disable.

mask option register (MOR) — An EPROM location containing bits that enable or disable certain MCU features.

MCU — Microcontroller unit. See “microcontroller.”

memory location — Each M68HC08 memory location holds one byte of data and has a unique address. To store information in a memory location, the CPU places the address of the location on the address bus, the data information on the data bus, and asserts the write signal. To read information from a memory location, the CPU places the address of the location on the address bus and asserts the read signal. In response to the read signal, the selected memory location places its data onto the data bus.

memory map — A pictorial representation of all memory locations in a computer system.

microcontroller — Microcontroller unit (MCU). A complete computer system, including a CPU, memory, a clock oscillator, and input/output (I/O) on a single integrated circuit.

modulo counter — A counter that can be programmed to count to any number from zero to its maximum possible modulus.

monitor ROM — A section of ROM that can execute commands from a host computer for testing purposes.

MOR — See "mask option register (MOR)."

most significant bit (MSB) — The leftmost digit of a binary number.

multiplexer — A device that can select one of a number of inputs and pass the logic level of that input on to the output.

N — The negative bit in the condition code register of the CPU08. The CPU sets the negative bit when an arithmetic operation, logical operation, or data manipulation produces a negative result.

nibble — A set of four bits (half of a byte).

object code — The output from an assembler or compiler that is itself executable machine code, or is suitable for processing to produce executable machine code.

opcode — A binary code that instructs the CPU to perform an operation.

open-drain — An output that has no pullup transistor. An external pullup device can be connected to the power supply to provide the logic 1 output voltage.

operand — Data on which an operation is performed. Usually a statement consists of an operator and an operand. For example, the operator may be an add instruction, and the operand may be the quantity to be added.

oscillator — A circuit that produces a constant frequency square wave that is used by the computer as a timing and sequencing reference.

OTPROM — One-time programmable read-only memory. A nonvolatile type of memory that cannot be reprogrammed.

overflow — A quantity that is too large to be contained in one byte or one word.

page zero — The first 256 bytes of memory (addresses $0000–$00FF).

parity — An error-checking scheme that counts the number of logic 1s in each byte transmitted. In a system that uses odd parity, every byte is expected to have an odd number of logic 1s. In an even parity system, every byte should have an even number of logic 1s. In the transmitter, a parity generator appends an extra bit to each byte to make the number of logic 1s odd for odd parity or even for even parity. A parity checker in the receiver counts the number of logic 1s in each byte. The parity checker generates an error signal if it finds a byte with an incorrect number of logic 1s.

PC — See “program counter (PC).”

peripheral — A circuit not under direct CPU control.

phase-locked loop (PLL) — A oscillator circuit in which the frequency of the oscillator is synchronized to a reference signal.

PLL — See "phase-locked loop (PLL)."

pointer — Pointer register. An index register is sometimes called a pointer register because its contents are used in the calculation of the address of an operand, and therefore points to the operand.

polarity — The two opposite logic levels, logic 1 and logic 0, which correspond to two different voltage levels, V_{DD} and V_{SS}.

polling — Periodically reading a status bit to monitor the condition of a peripheral device.

port — A set of wires for communicating with off-chip devices.

prescaler — A circuit that generates an output signal related to the input signal by a fractional scale factor such as 1/2, 1/8, 1/10 etc.

program — A set of computer instructions that cause a computer to perform a desired operation or operations.

program counter (PC) — A 16-bit register in the CPU08. The PC register holds the address of the next instruction or operand that the CPU will use.

pull — An instruction that copies into the accumulator the contents of a stack RAM location. The stack RAM address is in the stack pointer.

pullup — A transistor in the output of a logic gate that connects the output to the logic 1 voltage of the power supply.

pulse-width — The amount of time a signal is on as opposed to being in its off state.

pulse-width modulation (PWM) — Controlled variation (modulation) of the pulse width of a signal with a constant frequency.

push — An instruction that copies the contents of the accumulator to the stack RAM. The stack RAM address is in the stack pointer.

PWM period — The time required for one complete cycle of a PWM waveform.

RAM — Random access memory. All RAM locations can be read or written by the CPU. The contents of a RAM memory location remain valid until the CPU writes a different value or until power is turned off.

RC circuit — A circuit consisting of capacitors and resistors having a defined time constant.

read — To copy the contents of a memory location to the accumulator.

register — A circuit that stores a group of bits.

reserved memory location — A memory location that is used only in special factory test modes. Writing to a reserved location has no effect. Reading a reserved location returns an unpredictable value.

reset — To force a device to a known condition.

ROM — Read-only memory. A type of memory that can be read but cannot be changed (written). The contents of ROM must be specified before manufacturing the MCU.

SCI — See "serial communication interface module (SCI)."

serial — Pertaining to sequential transmission over a single line.

serial communications interface module (SCI) — A module in the M68HC08 Family that supports asynchronous communication.

serial peripheral interface module (SPI) — A module in the M68HC08 Family that supports synchronous communication.

set — To change a bit from logic 0 to logic 1; opposite of clear.

shift register — A chain of circuits that can retain the logic levels (logic 1 or logic 0) written to them and that can shift the logic levels to the right or left through adjacent circuits in the chain.

signed — A binary number notation that accommodates both positive and negative numbers. The most significant bit is used to indicate whether the number is positive or negative, normally logic 0 for positive and logic 1 for negative. The other seven bits indicate the magnitude of the number.

software — Instructions and data that control the operation of a microcontroller.

software interrupt (SWI) — An instruction that causes an interrupt and its associated vector fetch.

SPI — See "serial peripheral interface module (SPI)."

stack — A portion of RAM reserved for storage of CPU register contents and subroutine return addresses.

stack pointer (SP) — A 16-bit register in the CPU08 containing the address of the next available storage location on the stack.

start bit — A bit that signals the beginning of an asynchronous serial transmission.

status bit — A register bit that indicates the condition of a device.

stop bit — A bit that signals the end of an asynchronous serial transmission.

subroutine — A sequence of instructions to be used more than once in the course of a program. The last instruction in a subroutine is a return from subroutine (RTS) instruction. At each place in the main program where the subroutine instructions are needed, a jump or branch to subroutine (JSR or BSR) instruction is used to call the subroutine. The CPU leaves the flow of the main program to execute the instructions in the subroutine. When the RTS instruction is executed, the CPU returns to the main program where it left off.

synchronous — Refers to logic circuits and operations that are synchronized by a common reference signal.

TIM — See "timer interface module (TIM)."

timer interface module (TIM) — A module used to relate events in a system to a point in time.

timer — A module used to relate events in a system to a point in time.

toggle — To change the state of an output from a logic 0 to a logic 1 or from a logic 1 to a logic 0.

tracking mode — Mode of low-jitter PLL operation during which the PLL is locked on a frequency. Also see "acquisition mode."

two’s complement — A means of performing binary subtraction using addition techniques. The most significant bit of a two’s complement number indicates the sign of the number (1 indicates negative). The two’s complement negative of a number is obtained by inverting each bit in the number and then adding 1 to the result.

unbuffered — Utilizes only one register for data; new data overwrites current data.

unimplemented memory location — A memory location that is not used. Writing to an unimplemented location has no effect. Reading an unimplemented location returns an unpredictable value. Executing an opcode at an unimplemented location causes an illegal address reset.

V —The overflow bit in the condition code register of the CPU08. The CPU08 sets the V bit when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow bit.

variable — A value that changes during the course of program execution.

VCO — See "voltage-controlled oscillator."

vector — A memory location that contains the address of the beginning of a subroutine written to service an interrupt or reset.

voltage-controlled oscillator (VCO) — A circuit that produces an oscillating output signal of a frequency that is controlled by a dc voltage applied to a control input.

waveform — A graphical representation in which the amplitude of a wave is plotted against time.

wired-OR — Connection of circuit outputs so that if any output is high, the connection point is high.

word — A set of two bytes (16 bits).

write — The transfer of a byte of data from the CPU to a memory location.

X — The lower byte of the index register (H:X) in the CPU08.

Z — The zero bit in the condition code register of the CPU08. The CPU08 sets the zero bit when an arithmetic operation, logical operation, or data manipulation produces a result of $00.

Revision History

Major Changes Between Revision 2.0 and Revision 1.0

The following table lists the major changes between the current revision of the MC68HC08AZ32A Technical Data Book and revision 1.0.

Section Affected	Description of Change
Throughout	Reformatted document to current publications standards

Major Changes Between Revision 1.0 and Revision 0.0

The following table lists the major changes between the revision 1.0 of the MC68HC08AZ32A Technical Data Book and the initial release at revision 0.0.

Section Affected	Description of Change
General Description	Corrected text in numerous pin desriptions. Corrected Table 1-1 - External Pins Summary with which pins have hysteresis. Added missing modules to Table 1-3 - Clock Source Summary
Memory Map	Corrected type errors. Corrected various addresses and register names in **Figure 2-1 - Memory Map**. Corrected numerous register bit descriptions in **Figure 2-2 - I/O Data, Status and Control Registers** to match module sections. Added **Additional Status and Control Registers** section and moved register descriptions accordingly. Corrected bit descriptions to match module sections.
EEPROM	Section altered significantly to better align module descriptions across groups within Freescale using 0.5µ TSMC/SST FLASH. Numerous additions submitted by applications engineering for further clarification of functional operation.
Clock Generator Module (CGM)	Corrected clock signal names and associated timing parameters for consistency and to match signal naming conventions. Additional textual description added to **Reaction Time Calculation** subsection.
Mask Options	Corrected descriptions of LVIRST and LVIPWR bits
Break Module	Corrected description of BRKSCR register
System Integration Module (SIM)	Corrected various type errors in SBSR and SBFCR register bit descriptions
Monitor ROM (MON)	Modified **Figure 11-1 - Monitor Mode Circuit** based upon recommendations from applications engineering. Corrected type errors. Corrected **Figure 11-6 - Monitor Mode Entry Timing**.
Computer Operating Properly (COP)	Corrected references to COPL (now COPRS). Corrected type errors.
Low Voltage Inhibit (LVI)	Corrected functional description and revised Figure 13-1 - LVI Module Block Diagram

Section Affected	Description of Change
External Interrupt Module (IRQ)	Corrected ISCR register bit descriptions.
Timer Interface Module B (TIMB)	Corrected numerous type and grammatical errors. Corrected numerous pin and register name errors within text. Corrected references to TIMB overflow interrupts (removed "channel x" references as they are incorrect). Corrected functional description on TOF flag.
Programmable Interrupt Timer (PIT)	Corrected type and grammatical errors. Corrected PIT Overflow Interrupt Enable Bit acronym from PIE to POIE.
MSCAN08	Included Extended ID errata information as new subsection 20.6.1 Added address definitions to Figure 20-9 - MSCAN08 Memory Map.
Keyboard Module (KBD)	Added Low Power Modes subsection.
Timer Interface Module A (TIMA)	Corrected numerous type and grammatical errors. Corrected numerous pin and register name errors within text. Corrected references to TIMA overflow interrupts (removed "channel x" references as they are incorrect). Corrected functional description of TOF flag.
Electrical Specifications	Corrected type errors. Increased V_{HI} specification in **Maximum Ratings** to V_{DD} + 4.5V. Decreased LVI trip voltage specification to 3.80V and increased LVI recovery voltage to 4.49V in **5.0 Volt DC Electrical Characteristics**. Increased VHI specification to minimum of V_{DD} + 3.0V and maximum of V_{DD} + 4.5V in **5.0 Volt DC Electrical Characteristics**. Added Unit columns to all CGM specification tables and adjusted text accordingly. Corrected Operating Voltage specification in **CGM Operating Conditions**. Added typical specifications for K_{acq} and K_{trk} parameters in **CGM Acquisition/Lock Time Information**. Split **Memory Characteristics** table into separate **RAM Memory Characteristics and EEPROM Memory Characteristics**. Added maximum specification for EEPROM AUTO bit set for each of program and erase operation in **EEPROM Memory Characteristics**.
MC68HC08AZ32A Changes	Added subsection highlighting operation of IAR function. Added subsection highlighting change of Monitor Mode entry and COP disable voltage change. Added subsection highlighting change in LVI trip and recovery voltage specifications. Added subsection highlighting revised K values.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

RoHS-compliant and/or Pb- free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb- free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale.s Environmental Products program, go to http://www.freescale.com/epp.

MC68HC08AZ32A
Rev. 2, 07/2005

freescale™
semiconductor